Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj desktopové aplikace pro správu skladu

Vypracoval: Zdeněk Dolák
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Zdeněk DOLÁK
Osobní číslo:	E17439
Studijní program:	B6209 Systémové inženýrství a informatika
Studijní obor:	Ekonomická informatika
Téma práce:	Vývoj desktopové aplikace pro správu skladu
Zadávající katedra:	Katedra aplikované matematiky a informatiky

Zásady pro vypracování

Cílem bakalářské je vytvořit desktopovou aplikaci pro správu skladu. Aplikace bude umožňovat provádět operace s položkami skladu a o každé položce bude uchovávat rozšiřující informace. Položky budou uchovány v databázi. Aplikace umožní znázornit stav skladu (např. dostupný počet vybrané položky v průběhu času, upozornění na nízké množství položek apod.).

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje.
3. Implementace výsledné aplikace.
4. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
5. Závěry a doporučení.

Rozsah pracovní zprávy:	40 – 50 stran
Rozsah grafických prací:	dle potřeby
Forma zpracování bakalářské práce:	tištěná

Seznam doporučené literatury:

1. Albahari, J. & Albahari, B. (2017). *C# 7.0 in a Nutshell.* Sebastopol, California (USA): O'Reilly Media.
2. Harrington, J. L. (2016). *Relational Database Design and Implementation.* Cambridge, MA, USA: Morgan Kaufmann.
3. Nagel, C. (2016). *Professional C# 6 and .NET Core 1.0.* Indianapolis, Indiana (USA): John Wiley & Sons.
4. Posadas, M. (2016). *Mastering C# and .NET Framework.* Birmingham, UK: Packt.
5. Price, M. J. (2017). *C# 7.1 and .NET Core 2.0: Modern Cross-Platform Development.* Birmingham, UK: Packt Publishing.

Vedoucí bakalářské práce: Mgr. Radim Remeš
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **15. ledna 2019**
Termín odevzdání bakalářské práce: **12. dubna 2020**

V Českých Budějovicích dne 14. března 2019

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA

L.S.

doc. Dr. Ing. Dagmar Škodová Parmová

doc. RNDr. Jana Klicnarová, Ph.D.

Abstrakt

Cílem této bakalářské práce je vytvořit aplikaci pro správu skladu. Aplikace je vyvíjena pomocí JavaScriptového frameworku React Native, jehož základní myšlenkou je chytré propojení mezi platformami a jednou z nich je právě Windows pro kterou je tato aplikace vyvíjena. Praktická část práce se zabývá popisem jednotlivých technologií použitých při implementaci aplikace a jsou zde vysvětleny základní použité techniky objektově orientovaného programování a jejich návaznost na samotnou ukázkovou aplikaci. Praktická část se věnuje analýze, návrhu a samotnému procesu implementaci aplikace. Výstupem celé bakalářské práce je betaverze desktopové aplikace, která je otestována v podmínkách simulovaného prostředí skladu, tak aby zde byly použity veškeré základní funkcionality aplikace. Nedílnou součástí práce je také analýza toho, jak se aplikaci podařilo obstát a jaké byly případně nalezeny chyby.

Klíčová slova: React Native, Universal Windows Platform, JavaScript, PHP, REST API, operační systém Windows

Abstract

This bachelor thesis aims to create a desktop application for stock management. The application will be developed on JavaScript framework React Native. The fundamental idea of this library is a smart connection between two dominating platforms – iOS, Android, and Windows where this application is tested. The practical part describes a way of the framework's implementation in the sample application and explains linkages, processes, and techniques used for object-orientated programming. Data obtained from the application are stored on the MySQL server and then returned to the application in JSON format. The output of the thesis is a desktop application that is tested in simulated conditions for stock management in elementary operations. An integral part is a subsequent analysis of how the application stands the test or which errors occur.

Keywords: React Native, Universal Windows Platform, JavaScript, PHP, REST API, OS Windows

Čestné prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne ……...

Zdeněk Dolák

Tímto bych chtěl poděkovat svému vedoucímu práce Mgr. Radimu Remešovi za cenné připomínky a odborné rady při vypracování bakalářské práce. V době koronaviru bylo vypracování bakalářské díky omezenému přístupu k literatuře značně ztížené, děkuji tak všem, kteří se i přesto zdarma ve prospěch studentské veřejnosti podělili o své literární zdroje. Za velkou trpělivost a ochotu děkuji své přítelkyni. Poděkovat bych chtěl také svým kolegům a spolužákům za vzájemnou podporu během celého studia. Velké poděkování patří také mé rodině za jejich podporu při studiu.

Obsah

1 Úvod

Moderní technologie jsou pro podnikatele a manažery v dnešním byznysu velmi důležitou oblastí pro investice. Snaží se tak držet krok s nepřehlednou dobou plnou inovací. A právě v odvětví logistiky je kladen enormní důraz na efektivitu a kvalitu služeb. Logistické společnosti se tedy velmi snaží nacházet způsoby, jakými lze inovovat a zefektivnit stávající procesy. Cloudové služby, bezpečné modely, mobilní zařízení i aplikace předznamenávají éru skutečně otevřené ekonomiky a z dat se stává komodita, kterou člověk nedokáže bez pomoci softwaru analyzovat. (Logistika budoucnosti: Éra zákazníků, 2019)

Cílem práce je vytvořit desktopovou aplikaci pro správu skladu. Aplikace bude umožňovat provádět operace nad položkami skladu a o každé položce bude uchovávat rozšiřující informace. Jednotlivé položky budou uchovány v databázi. Aplikace bude umožňovat zobrazení celkových údajů, jako stav skladu nebo dostupný počet vybrané položky. Pro splnění tohoto cíle bude nutná analýza požadavků pro nový informační systém. Tyto poznatky následně poslouží k návrhu architektury, datového modelu a diagramu tříd. Pro implementaci na základě návrhu bude použit framework React Native. Aplikace bude otestována v simulovaném prostředí skladu a na závěr bude provedena analýza tohoto testu.

Motivací pro tuto práci je osobní zkušenost s podnikový informačním systémem, který po dlouhodobém používání vykazoval chyby, kterým by se dalo předejít. Tento informační systém nebyl v příliš připraven pro moderní technologie a pro uživatele často přestavoval spíše obtíže. Velkou nevýhodou zmiňovaného systému byla nemožnost sdílení dat mezi jednotlivými uživateli v terénu mimo síť firmy.

„Business Inteligence a Data Mining otevírají nové obzory, limitem je ovšem naše současná snaha porozumět datům, respektive nedostatek datových inženýrů. Dá se předpokládat, že jej s časem dokážeme překlenout, stejně jako dokážeme změnit zažité stereotypy. Nutně se se všemi těmito změnami totiž změní pohled na to, co je veřejné a co soukromé, naučíme se sdílením vytěžit z dat nové hodnoty a snižovat tak riziko své i ostatních, protože ostatní budou dělat totéž.“

„Sdílení dat snižuje rizika.“

(Logistika budoucnosti: Éra zákazníků, 2019)

2 Teoretická část

V teoretické části je cílem popsat použité technologie. Nejprve je popsán proces výběru technologií a následně jsou ve dvou kapitolách popsány jednotlivé technologie. V první podkapitole jsou popsány technologie použité na viditelné části aplikace – frontend a druhá podkapitola se zaměřuje na technologie v backendové části aplikace.

2.1 Výběr technologií

Možností pro vývoj aplikace na platformě desktop Windows existuje mnoho. Většina z nich ale pochází z dob, kdy byly desktopová zařízení hlavní platformou pro tvorbu aplikací. Dnes však v tomto ohledu dominují především mobilní zařízení a vývojové možnosti desktopových zařízení již nereflektují nejnovější trendy v oblasti vývoje aplikací. React Native splňuje požadavky možnosti komplexnější práce s daty a zároveň obsahuje díky častým aktualizacím nejmodernější trendy, a to zejména díky tomu, že se jedná o původně webové nebo mobilní prostředí pro vývoj aplikací. Prostředníkem mezi React Native a zařízením Windows byla přirozeně zvolena Universal Windows Platform, jež je právě pro tyto případy navržena. Výběr dalších technologií byl podle jednoduchého klíče nejoblíbenější technologie ve své oblasti (např. MySQL).

Dalším důležitým důvodem k volbě klíčové technologie React Native je potenciální budoucí rozšíření aplikace. React Native je primárně multiplatformní technologie, tzn. že kód který bude v průběhu vypracování praktické části této práce vytvořen bude moci v budoucnu být využit k vývoji na dalších mobilních platformách (Android nebo iOS). Vzhledem k povaze aplikace je to výhodné, protože mobilní aplikaci mohou využívat přímo skladníci na skladě nebo manažeři mimo kancelářské prostory.

2.2 Frontend (React Native)

Tato kapitola je zaměřena na konkrétní technologie použité na frontendu a také na React Native v kontextu vývoje desktopové aplikace pro Windows. Nejprve je vysvětlen pojem React Native, krátce jeho historie a obecně pojem hybridní aplikace. Dále jsou v této kapitole podrobněji vysvětleny techniky a nástroje používané během vývoje desktopové aplikace pro Windows na platformě React Native.

2.2.1 React Native

React Native je javascriptový framework pro vytváření nativních aplikací pro operační systémy jako je iOS nebo Android a s pomocí externích knihoven lze vyvíjet i na platformách jako je Windows (Universal Windows Platform). Je založen na Reactu, což je facebooková javascriptová knihovna pro vytváření uživatelského rozhraní upřednostňujícího mobilní aplikace před webovými prohlížeči. Jinými slovy: programátor může napsat aplikaci, která je pak totožná na všech platformách včetně PC. S nativním vývojem je zaručena minimálně dvojnásobná rychlost vývoje multiplatformní aplikace. Jedná se o velmi dynamicky se rozvíjející způsob vývoje aplikací a s každou chvílí přichází nové použitelné komponenty pro ještě efektivnější vývoj. (Eisenman, 2018)

Za vývojem tohoto frameworku stojí americká technologická firma Facebook. Představen byl v roce 2015 na konferenci F8. Facebook ji původně chtěl použít pouze pro vývoj svých vlastních aplikací, ale poté, co se setkal s velkým nadšením programátorů uvnitř Facebooku, tak se společnost rozhodla framework kompletně zveřejnit a přejít tak do open-source. V poslední době vznikají knihovny podporující desktopový vývoj pro Windows. Tato metoda bude pro práci využita. (Eisenman, 2018)

2.2.2 Hybridní aplikace

Aplikace se v zásadě dělí na 3 hlavní kategorie, a to aplikace webové, nativní a hybridní. Webová aplikace, jak již z názvu vyplývá je aplikace, která je umístěna na webu – je tak přístupná z jakéhokoliv zařízení ve webovém prohlížeči. Naopak nativní aplikace je dostupná pouze přímo na zařízení a je vyvíjena pouze pro určitou platformu. Kombinací těchto dvou přístupů k vývoji aplikací je hybridní aplikace, pro něž je určen i framework React Native. (Semaphore, 2020)

Vývoj hybridní aplikace má celou řadu výhod. Hlavní a již zmíněnou výhodou je možnost vývoje "nativní" aplikace pro více platforem zároveň. Syntax je založená na obecně známých principech webových technologií HTML, CSS a JavaScript, což umožňuje i rychlý přechod zkušených programátorů z oblasti vývoje webů. Výhodou také může být off-line dostupnost některých funkcí v aplikaci. Nevýhodou může být například určitá rozdílnost přístupu k nativním funkcím jednotlivých platforem, například funkce fotoaparátu nebo mikrofonu musí být pro jednotlivé platformy programovány odděleně. Hybridní aplikace také není vhodná pro všechny typy aplikací – dobře se určitě budou vyvíjet aplikace podobné webovému rozhraní,

naopak pro graficky náročné aplikace nebo aplikace využívající AR[1] je jakýkoliv typ hybridní aplikace naprosto nevhodný a zbytečný. (Semaphore, 2020)

Za zmínku stojí také společnosti, jež se rozhodly hybridní aplikace nebo konkrétně React Native využívat ve svých projektech. Přirozeně to jsou projekty od Facebooku jako – Facebook App, Instagram nebo Oculus, dále to jsou projekty od společností jako je Coinbase[2], Shopify, Bloomberg, Salesforce[3] nebo Tesla. (Semaphore, 2020)

2.2.3 ReactJS

ReactJS je do velké míry "mateřskou" technologií pro React Native. Jedná se o taktéž javascriptovou knihovnu, ale s tím rozdílem že je výhradně určená pro vývoj webových stránek a jejich uživatelského rozhraní. Veškeré základní principy z ReactJS jsou následně obsaženy i v React Native. (React JS: Getting Started, 2021)

Základním principem, pro který je celá rodina frameworku React typická je druh dynamicky uložených dat *state* (viz 3.1.7), který stále uchovává aktuální stav. Využití tak nalezne v aplikacích, kde potřebujeme uživateli nabízet aktuální data a dynamicky reagovat na jakoukoliv změnu stavu. (React Native: Learn once, write anywhere., 2021)

2.2.4 Universal Windows Platform

Klíčovou technologií pro vývoj nativní desktopové aplikace pro Windows je Universal Windows Platform (UWP) od Microsoftu, která umožňuje vyvíjet nativní aplikace na bázi operačních systémů Windows, včetně platforem kromě té hlavní desktopové, kterou je Windows 10, je možné na vyvíjet aplikace také pro Microsoft Xbox, Windows Phone nebo Microsoft HoloLens[4]. (Universal Windows Platform Documentation, 2021)

2.2.5 Javascript ES6

Javascript ECMAScript 6 nebo ECMAScript 2015 je v pořadí 6. edice implementace skriptovacího jazyka JavaScript, kterou konkrétně používá právě React Native. Programovací jazyk není nijak upraven pro použití v React Native, a naopak React Native dokáže využít

[1] AR – argumented reality – rozšířená realita
[2] Coinbase – internetová burza s kryptoměnami sídlící v USA
[3] Salesforce – americká společnost zabývající se cloudovými službami
[4] Microsoft HoloLens – projekt od Microsoftu zabývající se virtuální realitou

většiny předností tohoto skriptovacího jazyka. (ECMAScript® 2021 Language Specification, 2021)

2.2.6 JSX

Název JSX vyplývá z názvu JS (JavaScript) a "X" z anglického slova extension – rozšíření. Používá se v syntaxi Reactu a zejména v částech, kde je implementována UI část. JSX určitě v mnohém záměrně připomíná HTML, avšak jedná o čistokrevný JavaScript. HTML je značkovací jazyk, který je obecně srozumitelný a je tak pro nové vývojáře mnohem snadnější React Native co nejdříve pochopit. (Introducing JSX, 2021)

Pokud vytvoříme komponentu pomocí JSX (viz ukázka kódu 1) je při renderování této části kódu přeložena do čistého JavaScriptu (viz ukázka kódu 2). Jak z ukázky vyplývá, pro psaní kódu v React Native není tedy nutné používat JSX, je zde alternativa v podobě čistého JavaScriptu a využití funkce *React.createElement*. Syntax je ale s použitím JSX mnohem přehlednější a srozumitelnější. (Introducing JSX, 2021)

Zde je příklad toho, jak může vypadat JSX *element* v praxi.

Ukázka kódu 1 - JSX element

```
1
2  const element = (
3     <Text className="pozdrav">
4        Hello, world!
5     </Text>
6  );
7
```

Zdroj: převzato z Introducing JSX, 2021, Visual Studio Code

Tato syntaxa se následně zkompiluje jako čistý JavaScript.

Ukázka kódu 2 - JavaScript element, vlastní tvorba

```
1
2  const element = React.createElement(
3     'Text',
4     {className: 'pozdrav'},
5     'Hello, world!'
6  );
```

Zdroj: převzato z Introducing JSX, 2021, Visual Studio Code

V rámci JSX můžeme také používat další techniky jako například *Embedding Expressions* – díky které lze používat proměnné a funkce. V ukázka kódu 2 můžeme vidět použití proměnné, která se bude renderovat jako string v komponentě *Text*. Na jednotlivé komponenty se také lze pomocí ES6 funkce *import* odkazovat z jiných souborů, což může znovu velmi pomoci přehlednosti celého kódu. JSX obecně obsahuje velké množství možností, jako například podmíněné renderování, dědičnost, komponenta jako funkce a další. (Introducing JSX, 2021)

2.2.7 Komponenty a Props

Komponenty umožní rozdělit UI do nezávislých a znovupoužitelných kusů a přimějí uvažovat o každém kusu samostatně. V konceptu jsou komponenty v podstatě javascriptové funkce, které přijímají libovolné vstupy tzv. *props* a vracejí funkci, jež nám říká, co a jak se má na obrazovce uživateli po kompilaci objevit. V ukázka kódu 4 můžeme vidět definici jedné takové komponenty. *Props* nám udávají jaké vlastnosti budou jednotlivé komponenty mít a téměř každá komponenta je tak jejich prostřednictvím přizpůsobitelná. (Components and Props, 2021)

2.2.8 State a životní cyklus

Existují dva typy dat, jakými lze ovládat komponenty – jsou jimi *props* a *state*. *State* narozdíl od *props*, které jsou pouze pro čtení a neměly by být nijak modifikovány, dovoluje komponentě měnit její výstup v reálném čase na základě interakcí uživatele, odpovědi ze sítě nebo čehokoliv jiného. *State* se tak hodí pro data, která jsou proměnlivá a je žádoucí, aby se změny v reálném čase projevovaly. *State* je obecně inicializován v konstruktoru a pro změnu je volána funkce *setState*. V ukázka kódu 3 je inicializován *state greet* jako string “Ahoj“ v rámci objektu *Welcome*. *State* může být pak také velmi jednoduše přečten jako string jako například v ukázka kódu 4. (React Native, 2021)

Ukázka kódu 3 - Inicializace state v konstruktoru

```
constructor(props){
   super(props);
   this.state ={
      greet: "Ahoj",
   };
}
```

Zdroj: převzato z React Native, 2021, Visual Studio Code

2.2.9 Higher-Order component

Higher-Order Component (HOC) je pokročilá technika v React Native pro znovupoužití již vytvořených komponent. Své uplatnění najde například při využívání principu DRY (*Don't Repeat Yourself, česky – Neopakujte se*). V zásadě se jedná o funkci, která vrací novou komponentu. Může být užitečná zejména v situaci, kdy potřebujeme vypsat větší množství dat, které mají vypadat stejně, ale s jinými údaji. Každá komponenta má specifikovanou svoji vlastnost prostřednictvím *props* (z anglického slova "property"). (React Native, 2021)

Ukázka kódu 4 - React Native Komponenta s proměnou "state"

```
1
2  class Welcome extends React.Component {
3     render() {
4        return <Text>{this.state.greet}, {this.props.name}</Text>;
5     }
6  }
7
```

Zdroj: převzato z React Native, 2021, Visual Studio Code

Příklad komponenty, která vypisuje uživateli pozdrav, ale kdo je pozdraven je specifikováno až vlastností *this.props.name* a vlastnost *this.state.greet* je inicializována již v konstruktoru komponenty *Welcome* (viz ukázka kódu 3). Tato komponenta může být následně volána jednoduše následujícím způsobem v ukázka kódu 5. Výsledkem budou postupně jednotlivé textové výstupy *Ahoj Honzo*, *Ahoj Petře* a *Ahoj Emile*. (React Native, 2021)

Ukázka kódu 5 - Použití HOC

```
1
2  function App() {
3     return (
4        <View>
5           <Welcome name="Honzo" />
6           <Welcome name="Petře" />
7           <Welcome name="Emile" />
8        </View>
9     );
10 }
11
```

Zdroj: převzato z React Native, 2021, Visual Studio Code

2.2.10 NPM

NPM (původně Node Package Manager) je nástroj pro správu balíčků pro programovací jazyk JavaScript. Balíček je dostupný pro systémy na bázi Windows i Unix. Byl vyvinut společností GitHub, používá se velmi často právě pro získávání javascriptových repozitářů z této webové služby. (NPM / CLI: the package manager for JavaScript, 2021)

2.2.11 CLI

Command-line interface je ve své podstatě počítačový program, který přebírá textový vstup, vyhodnocuje ho a na jeho základě spouští požadované funkce operačního systému. V 60.letech to byl jediný způsob, jak ovládat počítače, v 70.letech přišlo na systém Unix první grafické rozhraní (GUI). Dnes se CLI stále využívá při vývoji aplikací, např. pomocí příkazu *npm* (viz 2.2.10) lze do svého projektu stáhnout z internetu jakýkoliv nabízený javascriptový balíček. CLI může také dobře posloužit při konfiguraci důležitých systémových souborů, změnách v adresářích nebo udělení práv jednotlivým uživatelům nebo skupinám. (NPM / CLI: the package manager for JavaScript, 2021)

2.3 Backend (REST API a MySQL Server)

Tato kapitola se zabývá obecně backendovou částí aplikace. A to zejména technickým popisem komunikace mezi interpretem REST API a databází.

2.3.1 Apache server a XAMPP

Apache http server je platforma pro software webového serveru. Apache server je součástí drtivé většiny Linux distribucí, existují samozřejmě i verze pro Windows a další operační systémy. Poslední verze projektu je ze srpna roku 2020 (2.4.46). (Apache: HTTP Server Documentation, 2020)

XAMPP je balíček systémů pro více platforem (Windows, MacOS, Linux), nejdůležitější jeho součástí jsou Apache HTTP Server, MariaDB databáze a interpret[5] pro skripty napsané v PHP nebo Perl. Název XAMPP je akronym slov X – jako ideografické slovo pro více platforem, A

[5] interpret – počítačový program, který interpretuje jazyk zápisu jednoho kódu do dalšího

– Apache Server, M – jako MariaDB (nebo MySQL), P – PHP a P – Perl. (Abbreviations.com: STANDS4 LLC, 2021)

2.3.2 REST API

REST API je rozhraní pro programování aplikací, které umožňuje interakci s webovými službami. REST API může využívat jakýkoliv webový protokol, nejčastěji využívá výhod tzv. HTTP protokolu. Vývojář tak nemusí instalovat žádnou knihovnu nebo přidaný software, aby REST API mohl používat. Poprvé se o této technologii zmiňuje počítačový vědec Dr. Roy Fielding v jeho disertační práci v roce 2000, ve které popisuje tehdejší alternativy k podobným webovým službám a následně navrhuje tuto novou technologii, jež má řešit problémy s delší časovou odezvou bezpečností. (Fielding, 2000)

Obrázek 1 - Komunikace mezi uživatelem, interpretem a databází

Zdroj: převzato z Fielding, 2000, Draw.io: Diagram Software and Flowchart Maker, 2021

Data, která lze z REST API získat nejsou nijak přímo svázána s určitou metodou a zdrojem, ale jsou naopak flexibilní a získávat je lze na základě parametrizace dotazu do tohoto webového rozhraní. Technologie je velmi rozmanitá a dovoluje tak uživateli vytvořit aplikace podle přísných požadavků. Odpověď z rozhraní lze získat v libovolném formátu, nejčastěji jsou používané *XML*, *JSON* (viz 2.3.6) nebo *YAML*. (What is a REST API?, 2021)

2.3.3 SQL

SQL, Structured Query Language je jazyk používaný pro vytváření dotazů do relačních databází, navržen je tak aby bylo možné získat z databáze jakoukoliv specifickou informaci. Pomocí SQL lze do databáze data také zapisovat, upravovat je nebo případně i mazat. Syntax jazyku SQL je velmi intuitivní. (M. Kroenke et al., 2015)

Zde je malá ukázka toho, jak může SQL dotaz vypadat. *SELECT* nám říká, co vše chceme z tabulky *item_table* zobrazit. *WHERE* je podmínka, kterou zobrazené položky musí splňovat a konečně *ORDER BY* nám říká, podle čeho se má tento výsledek seřadit.

Ukázka kódu 6 - SQL dotaz

```
SELECT * FROM item_table i WHERE active ORDER BY i.arrived
```

Zdroj:převzato z M. Kroenke et al., 2015, Visual Studio Code

2.3.4 MySQL

MySQL je systém pro řízení relačních databází s otevřeným kódem. Zajímavý je vznik názvu technologie, hlavním autorem první verze z roku 1995 je finský počítačový vědec Ulf Michael Widenius, který technologii pojmenoval po své dceři My – překlad “Moje“ SQL je tak mylný. Za zmínku stojí i to že Widenius stejný princip pojmenování použil i u MaxDB a MariaDB. (MySQL 8.0 Reference Manual: History of MySQL, 2021)

Svoji oblíbenost systém získal zejména pro jednoduchost, kterou nabízí, jednak jazyk SQL je obecně dobře uchopitelný a struktura relační databáze je také velmi intuitivní. MySQL také svým uživatelům garantuje rychlost, spolehlivost a škálovatelnost. Vlastnosti, které jsou pro databáze v dnešní době klíčové. MySQL také velmi dobře spolupracuje s jinými systémy, ať už to jsou různé knihovny, administrativní nástroje nebo široký výčet API prostředí. MySQL bez administrativního nástroje pro správu dat je pouze v CLI (viz 2.2.11) a pro použití databáze je nutné, aby MySQL modul byl spuštěn na serveru. V prostředí operačního systému Windows se o běh systému může postarat například XAMPP (viz 2.3.1), ve kterém je přímo modul MySQL obsažen a typicky je dostupný na adrese *http://localhost:3306*. (MySQL 8.0 Reference Manual: Including MySQL NDB Cluster 8.0, 2021)

2.3.5 PhpMyAdmin

PhpMyAdmin je jedním z mnoha administrativních systémů pro správu databází, který systému MySQL dodávají navíc ještě grafické rozhraní, ze kterého lze celou databázi ovládat. Jak napovídá název – celá aplikace je napsána primárně v jazyku PHP. PhpMyAdmin nabízí relativně přehledné prostředí se širokou škálou možností. (Welcome to phpMyAdmin's documentation!, 2021)

2.3.6 JSON

JSON (JavaScript Object Notation) je odlehčený formát pro výměnu dat. Je jednoduše čitelný i zapisovatelný člověkem a snadno analyzovatelný i generovatelný strojově. Je založen na podmnožině programovacího jazyka JavaScript (Standard ECMA-262 3rd Edition – December 1999). JSON je textový, na jazyce zcela nezávislý formát, využívající však konvence dobře známé programátorům jazyků rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších). Díky tomu je JSON pro výměnu dat opravdu ideálním jazykem.

(Introducing JSON, 2017)

Ukázka kódu 7 - JSON objekt

```
1
2  [
3     {
4        "id":"0",
5        "name":"Antonín",
6        "age": 34
7     },
8     {
9        "id":"1",
10       "name":"Marek",
11       "age": 26
12    },
13    ...
14 ]
15
```

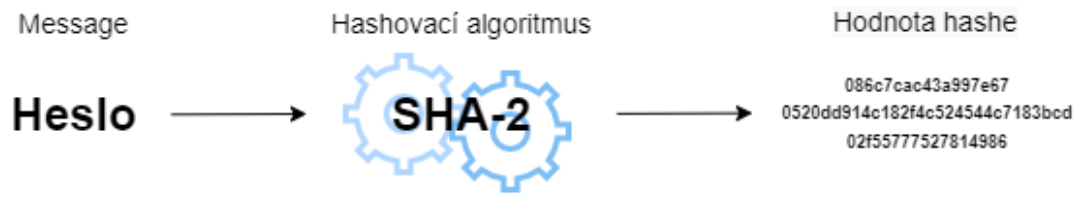
Zdroj: převzato z Introducing JSON, 2017, Visual Studio Code

2.3.7 Hashování hesel v SHA-2

Secure Hash Algorithm (SHA) je soubor kryptografických hashovacích funkcí vytvořen Národní Bezpečnostní Agenturou (NSA) Spojených Států Amerických a poprvé zveřejněn a patentován v roce 2001 a narozdíl od svého předchůdce SHA-1 je ještě bezpečnější. Hashovací funkce přijme data o jakékoliv délce a jednosměrně vytvoří tzv. hash o fixní délce, u SHA-256 je to 256 bitů což odpovídá 32 znakům a existuje 10^{77} kombinací, pro lidskou mysl nepředstavitelné číslo. Ve vesmíru je podle odhadů vědců přibližně 10^{80} atomů (Marie Helmenstine, Ph.D., 2019). Pro potenciálního útočníka je tak téměř nemožné tuto konkrétní překážku překonat, což z hashování SHA-2 dělá opravdu velmi efektivní nástroj v oblasti bezpečnosti informačních systémů. (US National Service Agency & Glenn)

Při vývoji aplikací má hashování mnoho využití, tím nejpoužívanějším je uchovávání citlivých údajů a zejména hesel nebo zpětné ověření nějakého dokumentu nebo souboru o kterém máme nějaký elektronický podpis. Hash by měla být při ověřování se stejnými vstupními daty vždy totožná se zahashovaným údajem uloženým v databázi. (Decoded: Examples of How Hashing Algorithms Work, 2020)

Obrázek 2 - Proces hashování dat



Zdroj: převzato z SHA1 vs SHA256, 2018, Draw.io: Diagram Software and Flowchart Maker, 2021

3 Analýza

3.1 Analýza trhu s aplikacemi pro správu skladu

Téměř veškeré firmy na českém trhu do nějaké míry operují s výrobky či materiálem, v tom případě je pro firmu klíčové mít svůj systém, jak s těmito položkami nakládat, jak je evidovat a mít o nich přehled.

Nejpopulárnější takovou službou je SAP, která konkrétně nabízí v rámci řešení SAP Business One program "Řešení pro malé a středně velké podniky", jehož jednou z vlastností je i skladové hospodářství. Tato část řešení je napojena na ostatní část celého řešení a dokáže na úrovni zásob spravovat položky skladu, ceníky i speciální cenové nabídky nebo převody mezi sklady. ("Řešení pro malé a středně velké podniky: Přehled", 2015)

„*Funkce pro správu skladových zásob řešení SAP Business One umožňují uživatelům spravovat kmenové informace o položkách a vést výrobní čísla, čísla šarží a ceníky. Uživatelé také definují alternativní položky, provádějí příjmy a výdaje pro na-výšení či snížení skladové zásoby, přeceňují skladové zásoby na základě aktuálních tržních hodnot, provádějí pravidelné inventury a generují seznamy zboží k vyzvednutí ze skladu pro nesplněné prodejní objednávky.*"

("Řešení pro malé a středně velké podniky: Přehled", 2015)

Jedním z největších českých ekonomických programů je Pohoda a její aktuální verze Pohoda 2021 od softwarové firmy Stormware, jejíž součástí je i řešení pro řízení zásob a skladu. Platforma je integrována jako součást účetního systému, takže nechybí funkce jako vystavování položkových dokladů pro potřeby účetnictví. Za zmínku také stojí pokročilý program pro náročnější obchodník POHODA E1, jenž zákazníkům umožňuje sklad ovládat i pomocí SQL dotazů a nabídne uživateli mnohem podrobnější informace o položkách na skladě. (Stormware Software: Sklady, 2021)

Vyvíjené řešení si klade za cíl být jednoduché a vzhledem k tomu, že není součástí žádného většího ekonomického softwaru, bylo by vhodné, aby do budoucna bylo teoreticky možné se s jiným softwarem propojit. Tuto možnost budou mít budoucí zájemci o integraci vyvíjeného softwaru prostřednictvím interpreta REST API, který je přímo propojený s databází a je schopný vyřešit základní dotazy i z jiných aplikací.

3.2 Cílová skupina

Aplikace je navrhována, tak aby byla použitelná pro management firmy pro přehled aktuálního stavu materiálu či výrobků na skladě. A zároveň, aby s aplikací mohli operovat i zaměstnanci firmy při operativním přesouvání materiálu, interně v rámci firmy nebo při expedici zboží ze skladu. Cílovou skupinou jsou tak firmy, které aplikaci mohou nasadit do svého provozu.

3.3 Analýza požadavků na aplikaci

3.3.1 Funkční požadavky

- Přístup do aplikace

Každý uživatel bude mít přiděleny své přístupové údaje uložené bezpečně v databázi na základě nichž se bude do aplikace přihlašovat.

- Přidání uživatele

Administrátor aplikace bude mít možnost přidat nové uživatele do aplikace, aby měli přístup do aplikace. Například v případě že firma přijme nového zaměstnance.

- Odebrání uživatelů

Administrátor bude mít také možnost odebrat jednotlivým uživatelům přístupová práva. Připraveno pro situaci, kdy například firma ukončí se zaměstnancem pracovním poměr a bude tak chtít zabránit tomu, aby již neměl přístup k citlivým informacím.

- Operace s položkami

Každý uživatel bude mít možnost s jednotlivými položkami provádět základní operace. Jako je expedice zboží, naskladnění nebo označení zboží jako poškozeného.

- Přehled položek ve skladu

Uživatel bude mít přehled o základních souhrnných údajích o položkách ve skladu. Například, jaké je celkové množství položek ve skladu nebo celkový počet položek od konkrétního dodavatele apod.

- Změna množství

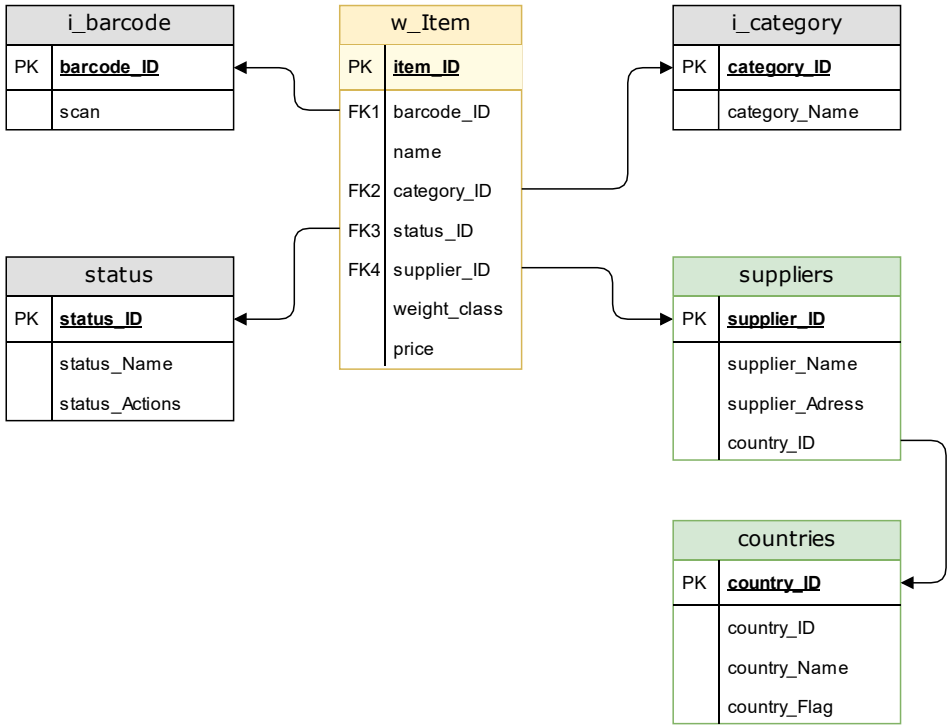
Uživatel bude mít možnost přidat a odebrat množství u jednotlivých položek, a to podle toho kolik dané položky v tu chvíli například skladník napočítá.

3.3.2 Funkční požadavky na databázi

Databáze je klíčová součást aplikace, protože aplikace samotná je postavená na datech z databáze. Aplikace je platforma, na které bude možné s jednotlivými položkami na skladě provádět změny, které se budou následně projevovat v reálném čase v databázi.

Cílem návrhu databáze je uložení údajů o jednotlivých položkách. Každá položka má své atributy, o nichž jsou další informace vedeny v dalších tabulkách. Jednotlivé atributy jsou s položkou provázány díky cizím klíčům přes ID jednotlivých atribut.

Obrázek 3 - Návrh databáze pro aplikaci StockManager



Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

3.3.3 Nefunkční požadavky

- Bezpečnost

Aplikace bude obsahovat potenciálně velmi citlivá data a je tak žádoucí, aby data byly bezpečně uložena. K datům by měli mít přístup pouze autorizovaní uživatelé.

- Rychlost aplikace

S aplikací budou pracovat operátoři na skladě během provozu, potřebují tak mít v reálném čase co nejrychleji správná data.

- Škálovatelnost

Aplikace může za delší dobu být naplněna velkým množstvím dat, musí tak být na takovou situaci připravena.

- Rozšiřitelnost

V situaci, kdy by byla nasazena v reálném provozu bude muset být možné aplikaci navázat na další systémy v rámci firmy. Poté by museli být další funkce do aplikace doprogramovány. Musí tedy být rozšiřitelná.

- Integrace do jiných systémů

Část aplikace bude integrovatelná do jiných systémů, aby o položkách měli přehled i další oddělení ve firmě.

3.4 Doménový model

V této kapitole jsou popsány základní entity, které se v aplikaci objevují.

3.4.1 Uživatel

Uživatel je základním článkem pro aplikaci, tato entita položky může sledovat. Aby mohl uživatel nějakým způsobem operovat s položka nebo sledovat jejich stav nebo množství, tak musí být přihlášen.

3.4.2 Položka ve skladu

Entita položky ve skladu je velmi důležitou součástí aplikace. Každá položka má své atributy a každá je jedinečná, rozlišitelná pouze podle primárního klíče ID v databázi.

3.4.3 Kategorie položky

Každá položka ve skladu je zařazena hned do několika kategorií, a to podle váhové kategorie, nebezpečnosti nákladu nebo interního rozdělení zboží ve výrobních procesech.

3.4.4 Stav položky ve skladu

Stav se eviduje také u každé u každé položky, jedná se o rozdělení podle jednotlivých procesů na skladu. Jiný stav je ihned po přijetí na sklad a jiný před noční inventarizací a jiný, pokud je položka připravena do výrobního procesu.

3.4.5 Dodavatelé

Každá položka, která pochází z importu (není výrobkem) má zaznamenána svého dodavatele. O každém dodavateli jsou uložené základní informace jako adresa nebo stát ve kterém sídlí.

3.4.6 Státy

Databáze státu je navázána na dodavatele, aby se pak v budoucnu dalo rozlišovat z jakého státu bylo zboží dovezeno. Důvody jsou různé legislativní podmínky států nebo různé daňové povinnosti vůči různým zemím.

3.5 Případy užití

3.5.1 Role

- Uživatel

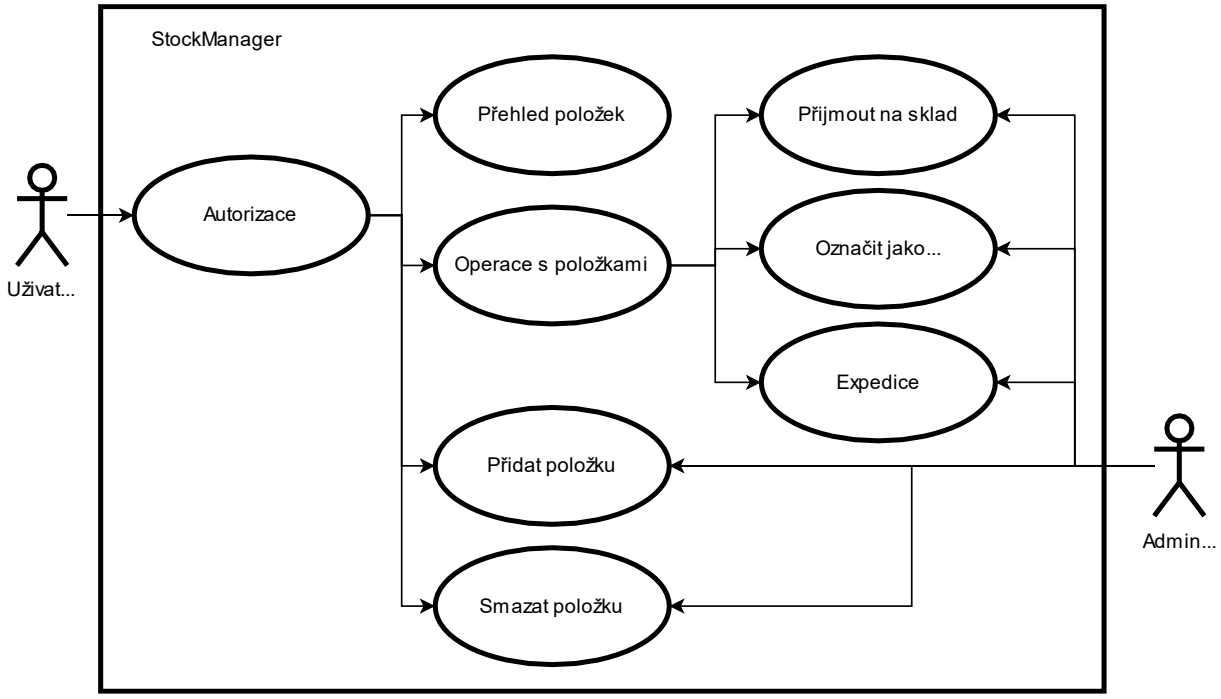
Uživatel je osoba, která k užívání aplikace dostala od administrátora přístupové údaje. Administrátor uživateli vytvoří údaje do databáze na základě jeho požadavku. Uživatelem může být manažer společnosti, která funkcí aplikace využívá nebo skladník který s položkami operuje fyzicky přímo na skladě. Uživatel se k používání aplikace musí nejprve vždy autorizovat.

- Administrátor systému

Administrátor systému je osoba, která má veškeré přístupové údaje k systému, včetně přístupu do databáze. Administrátor je oprávněn provádět jakékoliv změny v databázi a udělovat, případně odebírat přístupové údaje uživatelům. Administrátor systému je typicky zaměstnanec v IT, který v případě potřeby může do systému zasáhnout. Administrátor by svých práv neměl zneužívat.

3.5.2 Diagram případu užití

Obrázek 4 - Diagram případu užití aplikace



Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

Uživatel je oprávněn jakoukoliv změnu provést až po úspěšném přihlášení do aplikace. Pokud je mu přístup udělen, může nahlédnout do přehledu položek, kde může vidět základní informace o položkách, které se na skladě v danou chvíli nacházejí (přehled o dodavatelích, stavech a kategoriích jednotlivých položek). Dále může uživatel s položkami provádět základní změny jejich stavu. Položka, která je přijata na sklad může být dále změněna na stav, kdy je převzata výrobním oddělením ke zpracování. V případě, kdyby u položky došlo k poškození – může být označena jako poškozená. Dále může nabývat stavů neaktivní, univerzální nebo čekající. Uživatel může také přidat novou položku nebo položku smazat.

Administrátor systému je oprávněn k jakékoliv změně na úrovni databáze, může tak provádět stejné změny jako uživatel, a to bez jakéhokoliv omezení. Důležitou rolí administrátora je také udělování přístupu novým uživatelům, případně obnovení hesla v případě že by uživatel heslo zapomněl.

4 Návrh a implementace aplikace

Tato kapitola popisuje postupně proces instalace celého vývojového prostředí, nástrojů a frameworků potřebných k implementaci samotné aplikace. Následně jsou do kontextu vývoje aplikace uvedeny jednotlivé technologie popisované v teoretické části. Graficky jsou znázorněny případy užití aplikace, architektura, a nakonec i diagram tříd, který je následně podrobněji rozebrán.

Technicky jsou čtenáři představeny základní funkcionality a jejich vzájemná komunikace na frontendu a backendu aplikace. V další části zdrojový kód jsou popsány i vybrané části kódu, které jsou pro aplikaci a její základní funkcionality důležité. V rámci bezpečnosti je uživateli také technicky popsán proces autentizace uživatele do aplikace a politika vytváření a skladování citlivých údajů.

Pro aplikaci byl zvolen název StockManager, tedy manažer pro správu skladových zásob.

4.1 Vývojové prostředí

K implementaci aplikace bude použito univerzální IDE[6] prostředí pro vývoj v různých jazycích Visual Studio Community 2019, a to hlavně, jak díky dostupnosti a rozšířenosti platformy, tak i díky možnosti aplikaci debuggovat[7] v reálném čase přímo na stejném zařízení, na kterém probíhá úprava kódu. K základnímu vybavení prostředí je zapotřebí i C++ knihovna pro "překlad" kódu z JSX do prostředí platformy Windows. Ve Visual Studio Community je při vývoji aplikace pro debugging aplikace spuštěno UWP (viz 2.2.4) prostředí a aplikace StockManager.

Samotný kód, ve kterém je prováděna většina úprav je JSX (viz 2.2.6). Pro úpravu JSX byla zvolena sesterská aplikace Visual Code, která je o poznání jednoduší a primárně slouží pro úpravu kódu. Současně pro lepší orientaci a doplňování kódu byly použity rozšiřující knihovny React Native Snippet, React Native Tools, Remote – WSL pro CLI příkazy a konečně C/C++ knihovna pro malé úpravy v kódu, který překládá JSX pro platformu UWP.

[6] IDE – Integrated Development Enviroment, prostředí určené pro vývoj počítačových programů
[7] debuggování – ladění a odstraňování chyb během vývoje aplikace

4.2 Instalace vývojového prostředí

4.2.1 Požadavky

Pro vývoj aplikace jsou zapotřebí následující požadavky na software a hardware.

- Windows 10
- Vývojové prostředí (Visual Studio)
- Prohlížeč s Chrome jádrem
- Microsoft .NET Framework 4.6
- Procesor x86 nebo x64
- 1 GB RAM minimálně
- Minimálně 2.5 GB volného místa na disku
- MSVC v141 – VS 2017 C++ x64/x86 build tools (v14.16)

4.2.2 Universal Windows Platform

Instalace celého základní vývojového prostředí UWP byla provedena na základě oficiálního návodu Getting started with React Native for Windows. Kdy je na začátku nutné si v rámci Visual Studio získat knihovny, compilery a nástroje pro sestavení potřebné k vývoj a těmi jsou:

- Universal Windows Platform development
- MSVC v141 – VS 2017 C++ x64/x86 build tools (v14.16)
- MSVC v141 – VS 2017 C++ ARM build tools (v14.16)
- Desktop development with C++

Další nástroje jsou potřeba nainstalovat na úrovni samotného systému – tzv. webové nástroje, jsou jimi:

- Node JS – nástroj, který dohlíží na běh aplikace a je postaven na javascriptovém jádře v prohlížeči Chrome
- NPM – nástroj pro získávání JavaScriptových balíčků z internetu
- Yarn (nepovinné) – správce balíčků pro Node

(Get Started with Windows, 2020)

4.2.3 Framework React Native

Pokud jsou splněny všechny výše zmíněné požadavky, může být inicializovaný v požadovaném adresáři React Native projekt. A to jednoduchým příkazem v CLI, se kterými si nástroje pro správu internetových balíčků umí poradit. (“React Native“, 2021)

Ukázka kódu 8 – CLI příkaz pro inicializaci React Native projektu

```
react-native init StockManager
```

Zdroj: “React Native“, 2021

Aby mohla aplikace běžet a bylo možné ji v reálném čase upravovat, je potřeba aby byl spuštěn balíček *react-native* s parametrem *run-windows*.

Ukázka kódu 9 – CLI příkaz pro spuštění React Native Windows aplikace

```
npm react-native run-windows
```

Zdroj: “React Native“, 2021

4.2.4 Rozhraní serveru

Pro server, na kterém je spuštěna REST API není potřeba žádných speciálních požadavků, pro Windows je to aplikace XAMPP (viz 2.3.1) a správně nakonfigurovaný PHP server, který se odkazuje na konkrétní adresář v systému Windows, kde jsou uloženy PHP soubory REST API. Rozhraní je následně pro potřeby dotazů dostupné na lokálním prostředí typicky na adrese *http://localhost/stockmanagerapi/*.

4.2.5 MySQL

Podobně jako rozhraní serveru je i pro MySQL server předpřipravená konfigurace v prostředí XAMPP. V tomto prostředí je také již od počátku dostupný administrátorský nástroj pro správu databází PhpMyAdmin (viz 2.3.5). Který je dostupný na lokálním prostředí na adrese *http://localhost/phpmyadmin/*, odkud je možné prostřednictvím MySQL databázového klienta provádět v databázi změny.

4.3 Architektura aplikace

4.3.1 Technologický stack

Technologický stack je označení pro soubor nástrojů, programovacích jazyků a technologií, které dohromady utváří digitální produkt jako webovou stránku, mobilní nebo webovou aplikaci. Technologický stack obsahuje většinou technologie použité na frontendu a na backendu aplikace, některé složitější pak obsahují i databázi. Nejpopulárnějšími technologický stacky jsou například MEAN, MERN nebo LAMP. (Tech Stacks, 2021)

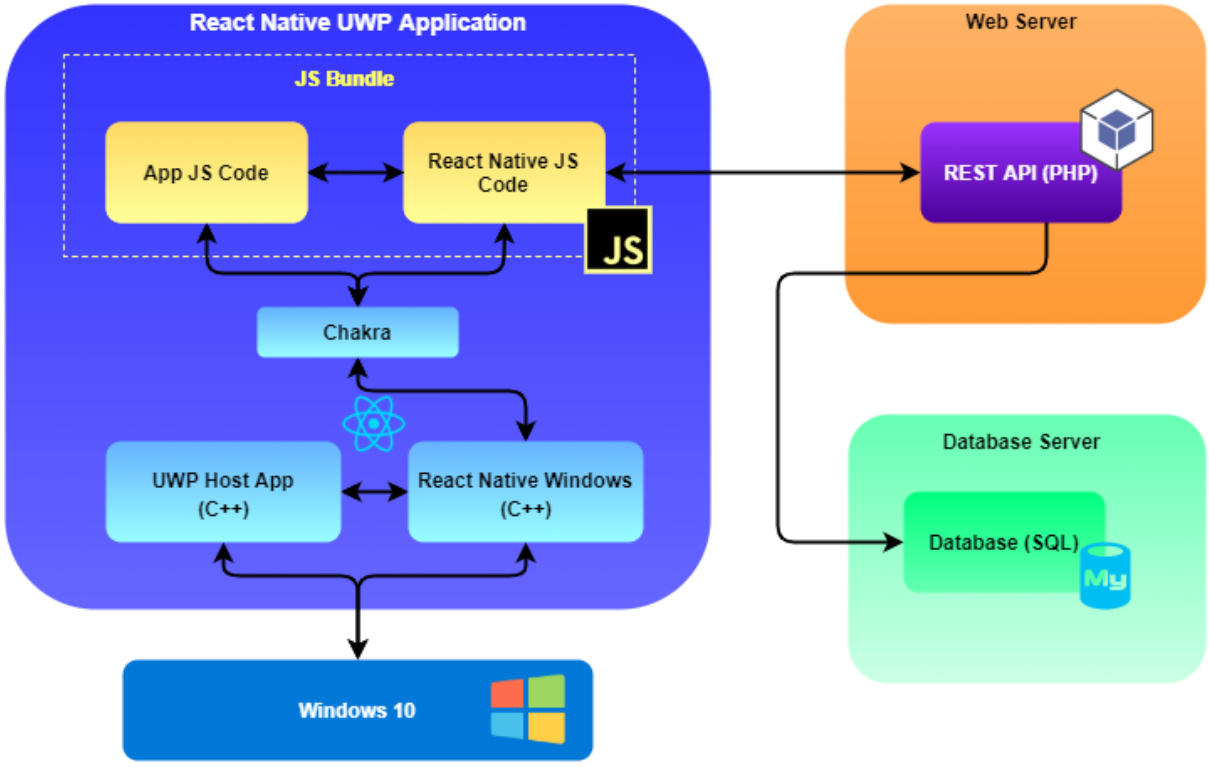
V tomto případě je použit technologický stack, který nemá své jednoznačné označení. Pro ukládání dat je to databáze MySQL, dále framework UWP od Microsoftu, jenž je postaven na programovacích jazycích C++ a C#, PHP REST API jako interpret mezi frontendem a backendem a React Native jako frontend aplikace. Výsledným akronymem tohoto technologického stacku je tak CUMPR.

CUMPR (C – C# a C++, U – UWP, M – MySQL, P – PHP a R – React Native)

4.3.2 Diagram architektury aplikace

V tomto diagramu je naznačeno, jakým způsobem spolu jednotlivé části aplikace komunikují. Nejdůležitější částí je celý element React Native UWP Application, kde celá aplikace běží. Frontendová část aplikace a přímo kód je v části JS Bundle. Aby mohl být kód z JS renderování uživateli musí být veškeré komponenty a data "přeloženy" do formy čitelné pro UWP, tedy Windows aplikaci. Tento transfer dat zajišťuje komponenta *Chakra*. Proto aby *Chakra* kódu z JS porozuměla, tak komunikuje s knihovnou *React Native Windows*. *UWP Host App* je výstupem celého procesu. (Universal Windows Platform documentation, 2021) Webový server je prostředí pro REST API a je interpretem mezi databází a samotnou aplikací (viz 4.2.4) a v databázi jsou uložena všechna důležitá data pro aplikaci (viz 4.2.5).

Obrázek 5 - Návrh architektury celé aplikace StockManager

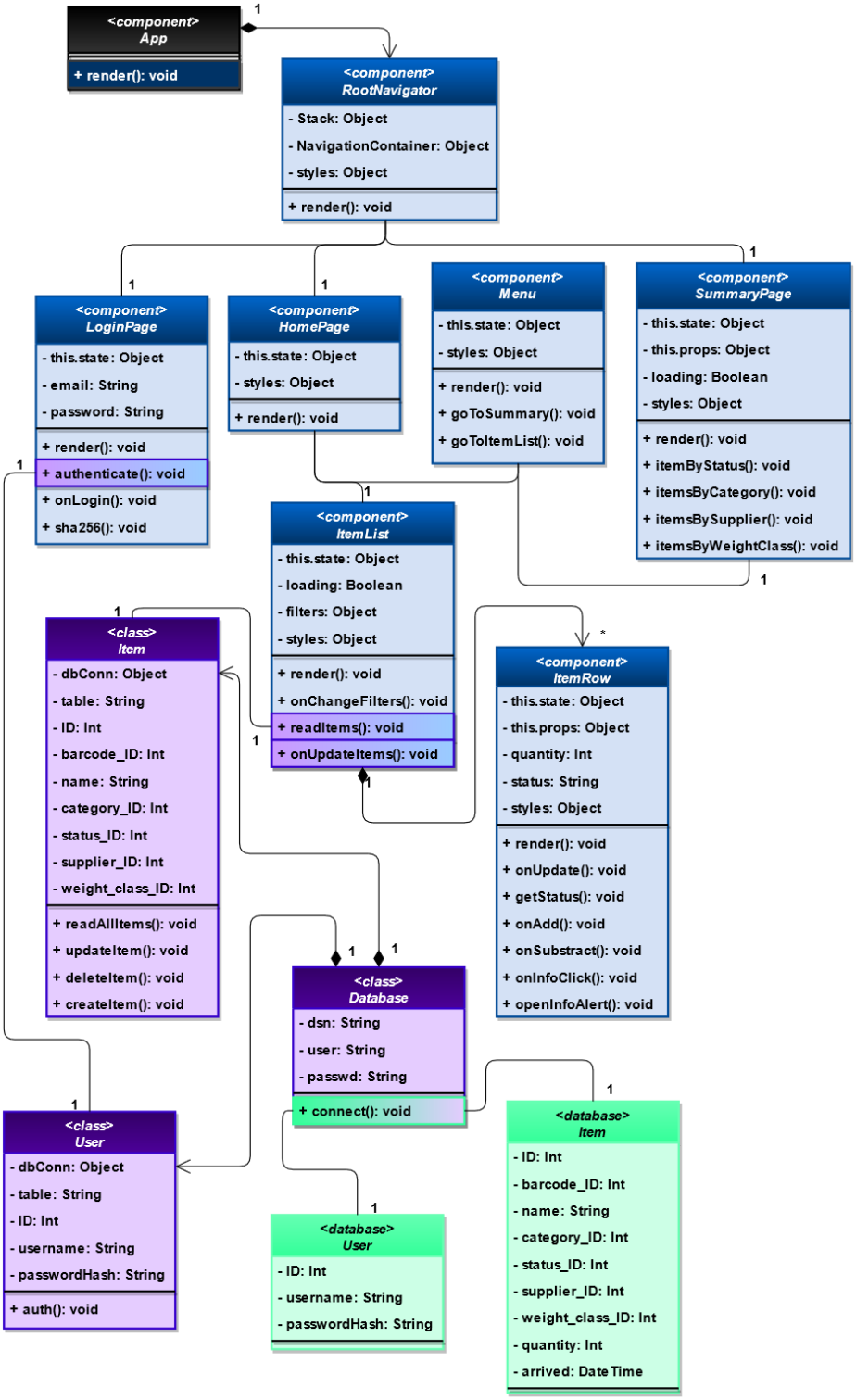


Zdroj: převzato z Universal Windows Platform documentation: The Universal Windows Platform (UWP) lets you build apps for any Windows device—PCs, Xbox One, HoloLens, and more—and publish them to the Store., 2021, vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

4.4 Diagram tříd

V této kapitole je popsán diagram tříd jako celek a nejzásadnější komponenty a třídy rozebrány do podrobnějších souvislostí.

Obrázek 6 – Diagram tříd aplikace StockManager



Zdroj: vlastní tvorba Draw.io: Diagram Software and Flowchart Maker, 2021

4.4.1 Základní popis

Diagram tříd ukazuje základní přehled tříd a atributů použitých při návrhu a implementaci aplikace. Jednotlivé části jsou barevně odlišeny, přičemž modrá je část frontendu (React Native), fialová je backend (REST API) a databáze je zvýrazněna zelenou barvou. Dále jsou v jednotlivých třídách vyznačeny dvojbarevně funkce, které danou třídu propojují s další třídou a jsou tak pro aplikaci a přenos dat klíčové.

4.4.2 Komponenta App

Klíčová komponenta, která renderuje celou vlastní aplikaci. App je výstupem pro celý projekt. Obsahuje pouze jednu funkce – *render*, která je v celém stromu všech ostatních funkcí na nejvyšším místě a každá další je jejím potomkem. Na tuto funkci navazuje přímou pouze další velmi důležitá komponenta *RootNavigator*, která je zodpovědná za přesuny mezi jednotlivými komponentami v UI.

4.4.3 Komponenta LoginPage

LoginPage je hlavní komponenta, která se uživateli zobrazuje po spuštění aplikace a slouží k získání autentizačních údajů od uživatele. Tato komponenta udržuje tedy 2 základní dynamicky se měnící stavy – *email* a *password*. Za spuštění celého procesu autentizace je zodpovědná funkce *onLogin* a následné šifrování do SHA-256 zajišťuje funkce *sha256* (viz 4.8.2). Dotaz do databáze poté provádí funkce *authenticate*, jenž spustí celý proces prověřování údajů. Odpověď z REST API může nabýt dvou stavů – autorizace udělena a autorizace neudělena. Pokud je uživatel autorizován je přesměrován přes komponentu *RootNavigator* na domovskou stránku, tedy *HomePage*.

4.4.4 Komponenty ItemList a ItemRow

Komponenta *ItemList* se uživateli zobrazí po úspěšném přihlášení a zobrazuje všechny položky, které jsou aktuálně dostupné v databázi. Uživatel má také možnost si položky vyfiltrovat nebo nalézt položku, pokud zná její ID – tedy jedinečné označení pro každou položku. Základním stavebním prvkem seznamu položek je *ItemRow*, která uživateli ukazuje v řádce podrobnosti o každé položce. Řádky jsou implementovány za použití *HOC* a zobrazuje se tolikrát, kolik je položek v databázi.

Řádka funguje zároveň jako tlačítko, při jejímž stisknutí se uživateli nabídnou základní možnosti, které může s položkou provést. V posledním sloupci jsou další méně častěji používané možnosti změny stavu položky.

4.4.5 Třída Item

Třída Item v PHP interpretu (REST API) je stěžejní třídou pro provádění všech dotazů do databáze. V této třídě jsou definovány všechny možné požadavky ze strany uživatele a veškeré atributy se kterými se může při odpovědi z databáze aplikace setkat. Hlavními funkcemi třídy je zpracování požadavků z frontendu aplikace. Přečtení veškerých položek ve skladu pomocí funkce *readAllItems*, přidání nové položky *createItem* nebo aktualizace nějakého stávajícího zápisu v databázi – *updateItem*. Třída také obsahuje funkci *response*, která odpověď z databáze dokáže naformátovat do JSON formátu pro zjednodušení zpracování dat na straně frontendu.

4.4.6 Třída Database

Třída *Database* v PHP interpretu je odpovědná za spojení s databází jako takovou. Obsahuje DSN (database source name) tedy adresu, na které je databáze spuštěna, parametr *user* – klientské jméno pro přihlášení do MySQL a pod parametrem *passwd* heslo. Samotné připojení na databázi následně provádí funkce *connect*. Návratovou hodnotou je úspěšné připojení nebo chybová hláška s podrobnostmi.

4.5 Frontend

Frontend softwarové aplikace nebo webové stránky je všechno s čím uživatel může interagovat. Z pohledu uživatele je frontend synonymum pro uživatelské rozhraní. Avšak z pohledu vývojáře je frontend uživatelské prostředí a vše co ho dělá funkčním. (“TechTerms“, 2021)

V případě aplikace StockManager se jedná o část, kde je Universal Windows Platform, na kterém běží React Native JS Bundle.

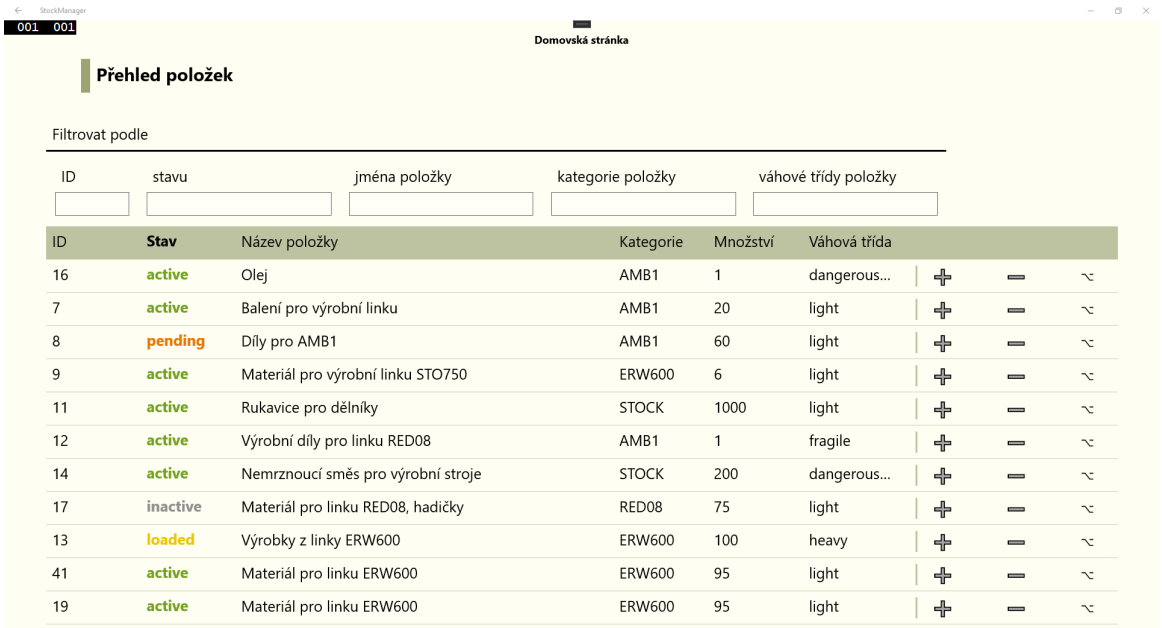
4.5.1 Uživatelské prostředí

Cílem návrhu uživatelské prostředí je poskytnout uživateli maximálně přehledné a informačně obsáhlé údaje o požadovaných položkách. Pro přehlednost byl zvoleny příjemné barvy, které nejsou nijak výrazné, tak aby všechny informace byly dobře čitelné.

- Domovská obrazovka aplikace

Uživatel ihned po přihlášení vidí, jaké položky jsou aktuálně na skladě dostupné a se kterými může provádět nějaké změny. Uživatel má možnost přejít na další stránku přehledu položek pomocí tlačítka *Přehled položek* v horní části obrazovky. Níže má pak uživatel možnost si vyfiltrovat určitou položku podle pěti různých atributů (ID, stav, jméno položky, kategorie a váhová třídy). V samotné tabulce jsou jednotlivé řádky obsahující základní informace o položkách, označení položky, její stav, název, kategorie, množství, váhová třída. Na konci řádky má odděleně uživatel k dispozici 3 tlačítka, přidat pomocí tlačítka *"+"* a odebrat pomocí *"-"* množství dané komodity. Posledním tlačítkem na řádce jsou *Další možnosti*, které uživatel využije v případě, že potřebuje použít méně běžnou změnu stavu, například označení položky jako poškozené.

Obrázek 7 - Domovská obrazovka aplikace s přehledem položek



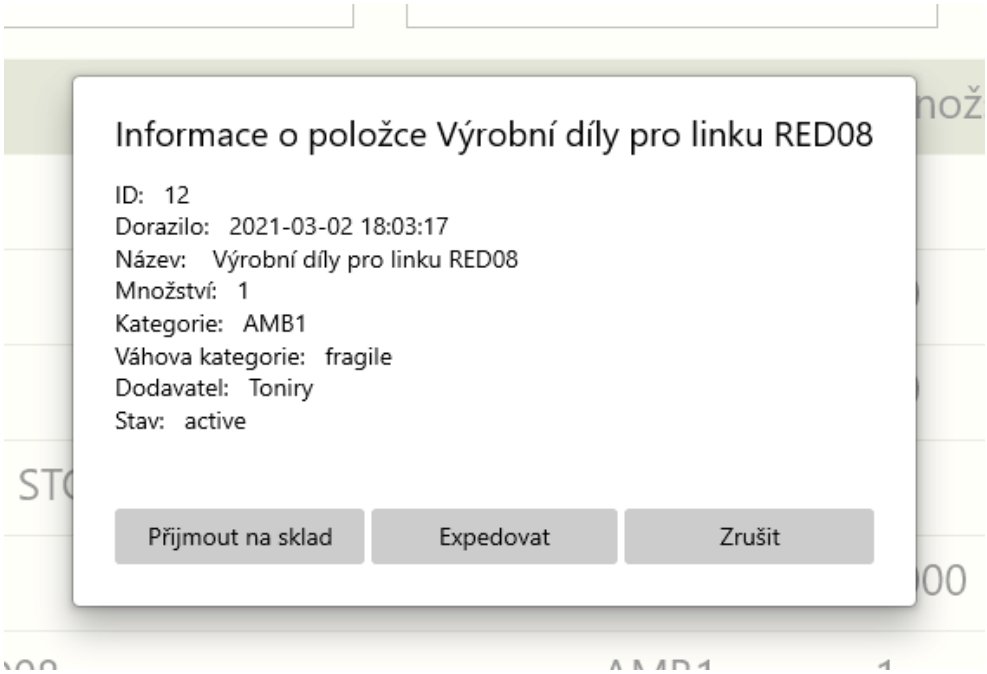
ID	Stav	Název položky	Kategorie	Množství	Váhová třída
16	active	Olej	AMB1	1	dangerous...
7	active	Balení pro výrobní linku	AMB1	20	light
8	pending	Díly pro AMB1	AMB1	60	light
9	active	Materiál pro výrobní linku STO750	ERW600	6	light
11	active	Rukavice pro dělníky	STOCK	1000	light
12	active	Výrobní díly pro linku RED08	AMB1	1	fragile
14	active	Nemrznoucí směs pro výrobní stroje	STOCK	200	dangerous...
17	inactive	Materiál pro linku RED08, hadičky	RED08	75	light
13	loaded	Výrobky z linky ERW600	ERW600	100	heavy
41	active	Materiál pro linku ERW600	ERW600	95	light
19	active	Materiál pro linku ERW600	ERW600	95	light

Zdroj: Aplikace StockManager, vlastní tvorba

- Informace o položce a možnosti uživatele

Na obrázek 8 je okénko, které se uživateli zobrazí poté co si vybere a stiskne na některou ze zobrazovaných položek. O položce se uživateli zobrazí doplňující informace, které v základní přehledu dostupné nejsou. Uživatel má 2 základní možností, které změní stav položky. *Přijmout na sklad* v případě, že zboží právě dorazilo na sklad a je potřeba ho zpracovat nebo *Expedovat* v případě že je zboží připraveno pro odeslání ze skladu. Případně může celou akci ukončit stisknutím tlačítka *Zrušit*. Jiné možnosti má uživatel po stisknutí tlačítka *Další možnosti* na konci řádku, kde může například zboží označit jako poškozené nebo převést do univerzální stavu.

Obrázek 8 - Možnosti uživatele s položkou



Zdroj: Aplikace StockManager, vlastní tvorba

4.6 Backend

V počítačovém světě backend referuje na jakoukoliv část aplikace nebo webu, která není vidět a je v přímém kontrastu s frontendem. V terminologii programátorů backend znamená jakousi vrstvu, která přistupuje k datům. Nejmodernější aplikace jsou dynamické, to znamená že jsou generovány on-the-fly[8] pomocí skriptů v pozadí. Každý takový skript je zákonitě součástí nějakého backendu. (“TechTerms“, 2021)

V aplikaci StockManager je za backend považována REST API, která komunikuje s databází a generuje pro aplikaci pomocí SQL dotazů potřebná data.

4.6.1 Databáze

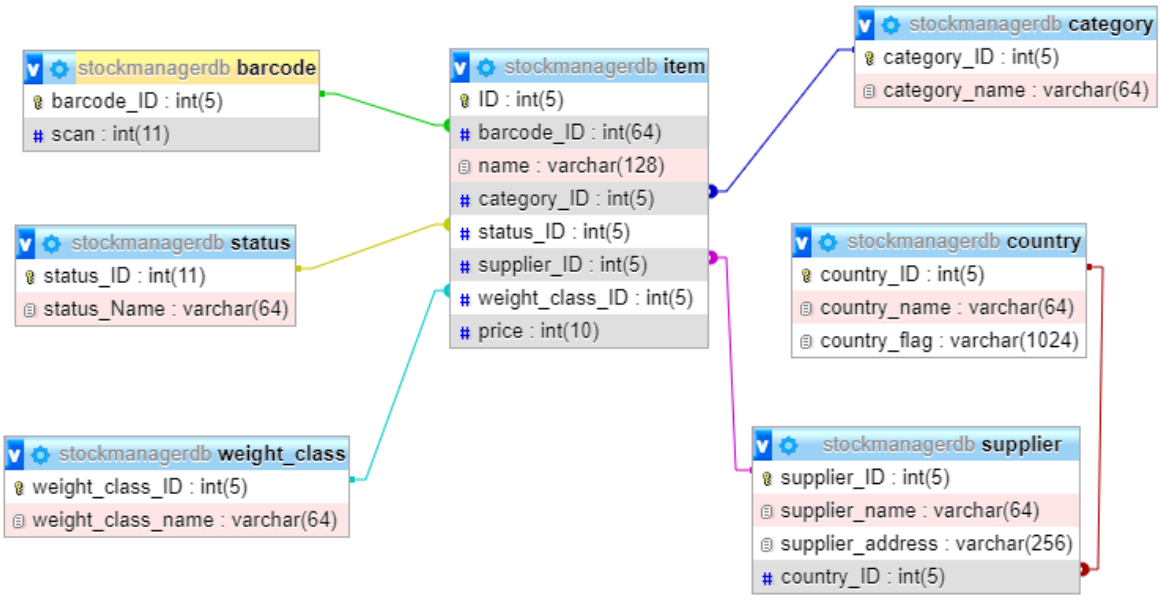
Databáze aplikace byla navržena na základě analýzy funkčních požadavků na aplikaci (viz 3.3.2). Je potřeba si určit základní strukturu uchovávání dat, která vychází z hlavní premisy, že středobodem datové struktury bude jednoznačně položka skladu – *item*. Každý *item* má pak své atributy v podobě cizích klíčů.

Databáze byla implementovaná na základě návrhu databáze (viz 3.3.2). Pro implementaci databáze byla použita open-source technologie phpMyAdmin, která byla vybrána zejména pro jednoduché UI prostředí a také proto, že pracuje s oblíbeným strukturovacím dotazovacím jazykem SQL (viz 2.3.3)

V průběhu implementace a debuggingu aplikace bylo vyvozeno, že bude potřeba databázi pozměnit. Došlo k přejmenování většiny atributů, tak aby splňovaly určité standardy. Doplněné byly také na základě potřeb aplikace datové typy k jednotlivým atributům.

[8] on-the-fly – za chodu

Obrázek 9 - Implementace databáze v phpMyAdmin



Zdroj: převzato z Welcome to phpMyAdmin's documentation!, 2021, vlastní tvorba

Jednotlivými cizími klíči v tabulce *item* jsou:

- *FK1 – barcode_ID*
- *FK2 – category_ID*
- *FK3 – status_ID*
- *FK4 – suppliers_ID*

4.7 Zdrojový kód

Tato kapitola popisuje jednotlivé části kódu. Postupně je v této kapitole popsán proces získání všech položek, které jsou následně vypsány uživateli na *Domovské stránce*. V části frontendu dojde k zavolání funkce *readItems*, která spustí proces přes REST API, na jehož konci je dotaz do databáze kde jsou požadovaná data uložena.

4.7.1 Frontend

V ukázka kódu 10 je návratový JSX kód, který ukazuje, jak vypadá domovská obrazovka aplikace (viz obrázek 7), na které jsou aktuálně naskladněné položky. Důležitou součástí této

komponenty je *Flatlist*, což je určitý seznam položek definovaných v atributu *renderItem* (řádka kódu 23), kde jsou za použití techniky HOC (viz 2.2.9) zobrazeny jednotlivé komponenty *Item*, každý s unikátním označením pod atributem *id* (řádka 25), převzatého z databáze (*item.ID*). Komponenta *Item* je zároveň vlastní komponenta designovaná, tak aby dokázala vypsat data, která jsou získána z databáze.

V komponentě jsou také vypisována filtrovaná data z dynamicky se měnícího zdroje *this.state.filteredItems* (řádka 20). Ve chvíli, kdy uživatel na cokoliv komponentě *Item* změní je volána funkce *this.onUpdateItems*, která následně změnu provede zápisem do databáze.

Ukázka kódu 10 – Návratová hodnota funkce render komponenty ItemsList

```
1  return (
2     <SafeAreaView style={styles.itemListView}>
3        <View style={styles.filtersView}>
4           <View style={styles.filter}>
5              <Text style={styles.tex}>Filtrovat podle ID</Text>
6              <FilterView
7                 value={this.state.filters}
8                 onChangeText={this.onChangeFilters.bind(this)}
9              />
10          </View>
11    </View>
12    <Item id={-1} item={firstRow}/>
13       {this.state.loading &&
14          <ActivityIndicator
15             size="large"
16             animating={this.state.loading}
17             style={styles.loading}
18          />}
19          <FlatList
20             data={this.state.filteredItems}
21             style={styles.flatListView}
22             keyExtractor={(item) => item.ID.toString()}
23             renderItem={({item}) =>
24                <Item
25                   id={item.ID}
26                   item={item}
27                   onItemCallback={this.onUpdateItems}
28                />} />
29    </SafeAreaView>
30 );
```

Zdroj: Aplikace StockManager, 2021, Visual Studio Code

Každá komponenta v aplikaci má vloženou komponentu *ActivityIndicator* (viz ukázka kódu 10 řádka 14), která před nahráním dat z databáze zobrazí uživateli indikátor načítání a vzhledem k tomu, že se jedná o nativní funkci, tak se uživateli zobrazí indikátor, který může znát z jiných Windows aplikací. Kdyby došlo k dalšímu vývoji a aplikace byla spuštěna na platformě Android, tak se stejná funkce uživateli bude renderovat jako indikátor načítání nativní na platformě Android. Zobrazení této komponenty je zároveň podmíněno dynamickou proměnnou *this.state.loading*, která je ve stavu *false,* pokud jsou všechny data načteny, a naopak ve stavu *true*, pokud je potřeba data ještě načítat, a tak uživateli zobrazit pouze indikátor načítání.

Ukázka kódu 11 - Komunikace s REST API, funkce readItems

```
1
2  export function readItems(){
3     return fetch(apiUrl+'item/read.php')
4     .then(response => response.json())
5     .then(data => {
6        return data
7     });
8  };
9
```

Zdroj: Aplikace StockManager, 2021, Visual Studio Code

V ukázka kódu 11 je funkce, která na frontendu komunikuje s REST API, jejíž cílem je získat data z databáze. Z takto získaných dat aplikace následně na frontendu pracuje, jsou vypisovány uživateli nebo jsou z nich vytvářeny přehledy. Návratovou hodnotou této funkce je hodnota funkce fetch, která zasílá požadavek na REST API a následně formátuje odpověď do čitelnějšího objektového formátu JSON (řádka 4).

4.7.2 REST API

Zatímco v ukázka kódu 10 je funkce, která získává data z databáze. Funkce *read* v další ukázka kódu 11 je PHP funkce v REST API, která tyto data pomocí SQL dotazu z databáze získává. *SELECT* (řádka 3) vybírá veškeré vlastnosti, které položka má. Kromě hlavní tabulky *items* jsou do dotazu pomocí *LEFT JOIN* zahrnuty také další tabulky, které jsou na ni přímo cizími klíči navázány. Výsledkem pak je že místo například ID kategorie, které je v databázi v tabulce *items* obsaženo, je název dané kategorie položky z tabulky *category*. Data z databáze jsou následně díky funkci *ORDER BY* seřazena vzestupně podle toho, kdy na sklad dorazila, tedy podle atributu položky *arrived*.

Ukázka kódu 12 - SQL dotaz pro získání všech položek skladu

```
1
2  public function read() {
3     $query = 'SELECT
4        i.ID,
5        i.name,
6        c.category_name AS category_name,
7        b.scan AS scan,
8        s.status_name AS status_name,
9        u.supplier_name AS supplier_name,
10       w.weight_class_name AS weight_class_name,
11       i.quantity,
12       i.arrived
13    FROM
14       '. $this->table . ' i
15    LEFT JOIN
16       category c ON i.category_ID = c.category_ID
17    LEFT JOIN
18       barcode b ON i.barcode_ID = b.barcode_ID
19    LEFT JOIN
20       status s ON i.status_ID = s.status_ID
21    LEFT JOIN
22       supplier u ON i.supplier_ID = u.supplier_ID
23    LEFT JOIN
24       weight_class w ON i.weight_class_ID = w.weight_class_ID
25    ORDER BY
26       i.arrived ASC';
```

Zdroj: Aplikace StockManager, 2021, Visual Studio Code

4.8 Bezpečnost

Bezpečnost aplikace je zajištěna především bezpečným uložením dat v aplikaci a zejména pak přihlašovacích údajů.

4.8.1 Hesla

Hesla pro přístup na server nebo přímo do aplikace byla volena, tak aby vyhovovala všem přísným standardům pro tvorbu hesel. Například heslo pro přístup do databáze je vytvořeno softwarovým generátorem Bitwarden[9], obsahuje 128 znaků a zároveň je složeno z malých, velkých písmen, číslic a speciálních znaků.

[9] Bitwarden – spolehlivý open-source trezor pro hesla

4.8.2 Autentizace

Poté co uživatel zadá své heslo do přihlašovacího formuláře na úvodní stránce je tento řetězec předán do funkce *authorize* na frontendu aplikace, která heslo zahashuje do 256bitového řetězce (viz 2.3.7), který je pak následně poslán k autentizaci do databáze, která již obsahuje zahashované heslo. Porovná tyto dva údaje a pokud se neodlišují je uživatel autorizován a je přesměrován na domovskou stránku aplikace.

Tato metoda je využívána zejména díky velké bezpečnosti hashování. Je prakticky nemožné, aby se případnému útočníkovi podařilo do aplikace nabourat i po tom co by nějakým způsobem získal přístup k zahashovaným údajům z databáze. Hash je také nemožné zpětně rozšifrovat. (Decoded: Examples of How Hashing Algorithms Work, 2020)

5 Testování aplikace

V této kapitole jsou popsány způsoby otestování aplikace, jejíž výsledky jsou popsány v části Vyhodnocení testů (viz 5.3). Testování aplikace je velmi důležitá část celého procesu vývoje. Při testování je možné nalézt chyby, které mohly být v průběhu implementace aplikace přehlédnuty. Úspěšné testování může být jedna z posledních činností před zveřejněním projektu. Pro relevantní výsledky testů by testeři neměli být příliš zaujatí a měla by jim být umožněna přirozená interakce s aplikací. (Omniconvert, 2020)

5.1 Testované zařízení

Celá aplikace byla vyvíjena, tak aby fungovala správně na zařízeních PC a na operačním systému Windows 10, tudíž i testování probíhá na tom stejném zařízení. Zařízení splňuje veškeré minimální parametry pro vývoj na platformě Universal Windows Platform od Microsoftu (viz 4.2.1).

Základní parametry testovaného zařízení:

- OS: Windows 10 © Microsoft Corporation, All rights reserved.
- Procesor: Intel® Core™ i5-5200 CPU @ 2.20GHz 2.20GHz
- RAM paměť: 16 GB
- Typ systému: 64-bit OS a procesor na bázi x64

5.2 Uživatelské testování

V této kapitole jsou popsány základní metodiky použité při testování aplikace uživatelem. Následně jsou jednotlivé testy aplikovány.

5.2.1 Testování použitelnosti

Koncept testování použitelnosti můžeme definovat jako efektivitu a snadné použití rozhraní aplikace uživatelem, abychom uspokojili jeho potřeby. V praxi to znamená, že pokud uživatel aplikaci otevře, musí mu být ihned jasné co a jak ovládat. Může mu k tomu dopomoci při první návštěvě například nějaká nápověda. Použitelnost vyvíjené aplikace je klíčová, protože může pro zákazníky nebo uživatele naprosto klíčová při rozhodnutí, zda ji budou nadále používat či ne. Pokud vývojáři aplikace v tomto bodu neuspějí může to předznamenat jejich neúspěch. (Omniconvert, 2020)

5.2.2 A/B testování

A/B testování je nejefektivnějším způsob, jak zjistit, zda aplikace plní svůj účel. V teorii je A/B testování založeno na tom, že např. novou funkci aplikace poskytneme jen části uživatelů, co nejvíce podobné druhé testovací skupině a sledujeme, jak na novou funkcionalitu uživatelé reagují. Jinými slovy se jedná o způsob založený na zkoušení nových funkcí metodou pokus a omyl. Pokud je skupina s novou funkcí nespokojena od takové vývojové větve se běžně odstupuje, pokud pro přidání nové funkce nejsou jiné pragmatičtější důvody. (Omniconvert, 2020)

5.2.3 Beta testování

Za beta verzi aplikace můžeme považovat takovou verzi aplikace, která zatím neprošla ostrým provozem. Beta verze může být otevřena pro veřejnost s upozorněním, že se jedná pouze o nedokončenou verzi aplikace nebo pro úzkou skupinu lidí, která se na testování a hledání chyb chce podílet. Cílem beta testování je odhalit problémy, které mohou způsobovat chyby. Data pro získání nastalých chyb lze získat automatickým odesíláním dat z uživatelského používání, ale také zpětnou vazbou od beta testerů, která může být velmi cenná. (Omniconvert, 2020)

5.3 Vyhodnocení testů

5.3.1 Testování použitelnosti

První, s čím se uživatel po spuštění aplikace setká je přihlašovací obrazovka do aplikace. Zde vidí jen tři UI objekty – políčko pro vyplnění přihlašovacího jména s placeholderem[10] *"email"*, z něhož je patrné, co po uživateli aplikace požaduje a políčko pro vyplnění hesla s placeholderem *"heslo"* a tlačítko pro přihlášení. Po vyplnění emailu kurzor automaticky přeskakuje na heslo a po následném stisknutí tlačítka *Enter* se spustí proces autentizace. Pro uživatele je tento proces velmi intuitivní a příjemný.

Po autorizaci se uživatel dostane na domovskou stránku aplikace, kde je přehled všech položek na skladě. Uživatel si může vyhledat specifickou položku ze skladu pomocí políček pro filtrování, a to podle ID, stavu, jména položky, kategorie a váhové třídy. V horní části obrazovky je možnost přejít na přehled skladu. Rozšířené možnosti k položce dostane uživatel stisknutím na tu položku, se kterou má záměr změnu provést. Otevře se mu vyskakovací okno s dalšími podrobnostmi o položce a s několika tlačítky, kterými s položkou dle jeho požadavků může operovat. Pokud uživatel nějakou možnost vybere položka změní svůj stav, jenž se změní i v databázi. Uživatel může také celou akci ukončit bez změny pomocí tlačítka *Zrušit*. V každé řádce položky má uživatel také možnost, díky tlačítkům *"+"* a *"-"*, u každé položky změnit její množství podle jeho požadavků. Veškeré možné akce na domovské stránce jsou pro uživatele na první pohled uživateli jasné a přehledné.

Z pohledu použitelnosti je relativně přehledná i stránka *Přehled položek* (viz Příloha B9). Na levé straně uživatel vidí obecný přehled. Kolik je celkově datových záznamů v celé aplikaci a kolik položek je aktuálně ve skladě. V pravé části obrazovky pak uživatel může vidět již podrobnější data. V horní části jsou počty položek v jednotlivých výrobních procesech, prostřední tabulka je věnována tomu, z jakých dodavatelských řetězců položky na skladě pocházejí a konečně poslední tabulka uživateli ukazuje rozdělení položek mezi váhové kategorie. Prostředí přehledu je podobně jako ostatní stránky aplikace navrhováno, tak aby uživatel nebyl zahlcen příliš velkým množstvím informací a zároveň aby získal informace, které požaduje.

[10] placeholder – ukázková hodnota do formulářového pole

Zamýšlená použitelnost jednotlivých funkcionalit funguje, tak jak byla navrhována a uživatel by při používání neměl mít potíže s porozuměním jaká data právě sleduje.

5.3.2 A/B testování

A/B testování bylo do jisté míry prováděno během celého průběhu návrhu a implementace aplikace. Některé původně zamýšlené funkcionality byly rozpracovány, avšak bylo od nich odstoupeno, protože by byly příliš časově náročné v porovnání s tím, jak moc důležitou funkcionalitu by to uživatelům přinášelo. Jednalo se například o přidání grafických přehledů do komponenty *SummaryPage*, kde je přehled položek. Pro UWP platformu React Native neexistuje příliš mnoho knihoven, které by byly jednoduché pro implementaci.

Některé funkce byly naopak přidány v průběhu, protože se jednalo o jednoduchou implementaci. Jednalo se například o filtrování položek. Tato funkcionalita je velmi jednoduše rozšiřitelná o další parametry, podle kterých může uživatel položky filtrovat.

5.3.3 Beta testování

Aplikace je v beta verzi. Pro testování byl připraven testovací scénář, který má za cíl aplikaci prověřit, zda by obstála v reálném prostředí skladu.

- Testovací scénář

Testovací scénář by designován tak, aby byl otestován standardní proces logistického procesu od materiálu po výrobek. Před odesláním hotového výrobku však došlo k poškození balení, aby mohly být vyzkoušeny i méně používané funkce aplikace.

1. Přichází nový materiál ze skladu v Polsku a operátor se přihlašuje do aplikace.
2. Operátor skladu si materiál vyfiltruje a přijímá jej sklad.
3. Druhý den operátor posílá materiál na výrobní linku *RED08*.
4. O 3 dny později je materiál zpracovaný a operátor výroby vytváří novou položku *Výrobek RED08*.
5. Operátor na skladě si zboží vyfiltruje podle kategorie a přijímá výrobky na sklad.
6. Operátor zjišťuje, že balení zboží je poškozeno a převádí položku skladu na stav *damaged* – poškozeno.
7. Probíhá oprava balení zboží.
8. Operátor dostává informaci, že se pro zboží našel kupec, zboží si vyfiltruje podle *ID* v databázi aplikace a vyexpeduje ho.
9. Zboží je odesláno.

- Výsledek testu

Celý testovací scénář technicky proběhl podle předpokladů správně. Avšak objevilo se několik nejasností ohledně životního cyklu položek. V momentě, kdy byl materiál již zpracován, tak stále v přehledu databáze stále zůstává, po nějaké době používání aplikace by tak uživatel ztratil o položkách přehled. Jednoznačným řešení by bylo omezení přehledu časovým filtrem podle atributu *arrived*. Avšak uživatel má již v této chvíli mnoho možností, podle kterých si data může vyfiltrovat a nikdy neoperuje s celými daty.

Pozn. Celý proces průchodu testovacím scénářem je zdokumentován se screenshoty v přílohách na straně 66.

6 Závěr

Hlavním cílem této bakalářské práce bylo čtenáři přiblížit proces vývoje desktopové aplikace. Pro takový projekt je nejprve důležité nashromáždit dostatek informací, zanalyzovat požadavky a následně se správně rozhodnout, jaký způsob implementace zvolit. Proces výběru technologií je popsán v první částí teoretické části. Následně jsou jednotlivé technologie podrobněji popsány za použití odborné literatury, především dokumentací. V praktické části jsou zanalyzovány podobné aplikace pro správu skladu a v tomto kontextu je posléze provedena analýza požadavků pro aplikaci, doménový model a případy užití. Další kapitola se věnuje samotné implementaci aplikace. Postupně je čtenář seznámen s autorovým vývojovým prostředím, instalací frameworků, hardwarovými požadavky a návrhem samotné aplikace. V další části kapitoly implementace je popis zdrojového kódu, kde je názorně ukázán příběh procesu získání dat z databáze do aplikace prostřednictvím REST API. Na závěr vlastní práce je aplikace podrobena několika testům, ve kterých aplikace obstála. V přílohách je dostupný i s obrázky průběh beta testování aplikace.

Výstupem celé práce je tedy desktopová aplikace pro správu skladu, jejíž zdrojový kód je veřejně dostupný na adrese https://github.com/dolakzdenek/StockManager_BP. Podle analýzy GitHubu je 39 % kódu JavaScript, 27 % PHP, 15 % Java a 11 % Objective-C. V tomto veřejném repozitáři nejsou nahrané soubory UWP, protože jsou příliš velké a potenciální zájemce o spuštění vývojového prostředí si může jedním CLI příkazem potřebné knihovny stáhnout. (Aplikace StockManager, 2021)

Pokud by se autor k vývoji aplikace vrátil, existuje určitě množství funkcionalit, které by bylo nutné před publikováním aplikace přidat, avšak během celého procesu vývoje bylo k aplikaci přistupováno tak aby byla v budoucnu rozšiřitelná o další funkce. Množství vytvořených komponent je také podle principu DRY v budoucnu znovupoužitelných s jinými parametry, čímž je právě další rozšiřitelnost aplikace zajištěna.

7 Summary

The main goal of this bachelor's thesis was to introduce the reader to the process of developing a desktop application. For such a project, it is first important to gather enough information, analyze the requirements and then decide correctly which method of implementation to choose. The process of technology selection is described in the first part of the theoretical part. Subsequently, the individual technologies are described in more detail using the literature, especially documentation. In the practical part, similar applications for warehouse management are analyzed, and in this context, an analysis of the requirements for the application, domain model and use cases is then performed. The next chapter deals with the implementation of the application itself. Gradually, the reader is introduced to the author's development environment, installation of frameworks, hardware requirements and design of the application itself. The next part of the implementation chapter is a description of the source code, which clearly shows the story of the process of obtaining data from the database into the application through the REST API. At the end of the work, the application is subjected to several tests, in which the application passed. The process of beta testing of the application is also available with pictures in the attachments.

According to GitHub's analysis, 39% is JavaScript code, 27% PHP, 15% Java and 11% Objective-C. UWP files are not uploaded in this public repository because of files size and potential applicants for running the development environment can download the required libraries with a single CLI command. (Aplikace StockManager, 2021)

If the author returned to the development of the application, there are certainly several functionalities that would need to be added before publishing the application, but during the entire development process, the application was approached so that it could be extended with other functions in the future. The number of created components is also reusable in the future according to the DRY principle with other parameters, which ensures further extensibility of the application.

I Seznam použité literatury

- *Logistika budoucnosti: Éra zákazníků [Online]*. (2019) (Vol. 20). Retrieved from https://www.komora.cz/files/uploads/2019/02/komora_0219_web.pdf
- *Eisenman, B. (2018). Learning React native: Building native mobile apps with JavaScript (Second edition)*. Boston, MA: O'Reilly.
- *Kout, P. (2004). Praktický JavaScript*. Zoner Press.
- *JavaScript Enviroment [Online]. (2019).* Retrieved January 09, 2020, from https://facebook.github.io/react-native/docs/javascript-environment
- *What Is React Native? The Rise of Hybrid Mobile Apps. (2020).* Semaphore, 1. https://semaphoreci.com/blog/what-is-react-native
- *Get Started with Windows. (2020). React Native For Windows + macOS.* Retrieved October 26, 2020, from https://microsoft.github.io/react-native-windows/docs/getting-started
- *Universal Windows Platform documentation: The Universal Windows Platform (UWP) lets you build apps for any Windows device—PCs, Xbox One, HoloLens, and more—and publish them to the Store. (2021).* Retrieved February 17, 2021, from https://docs.microsoft.com/en-us/windows/uwp/React Native: Learn once, write anywhere. (2021). Retrieved February 13, 2021, from https://reactnative.dev/
- *Lets you build apps for any Windows device—PCs, Xbox One, HoloLens, and more—and publish them to the Store.* (2021). Retrieved February 17, 2021, from https://docs.microsoft.com/en-us/windows/uwp/
- *M. Kroenke, D., J. Auer, D., & Goner, J. (2015). Databáze*. Albatros Media.
- *React JS: Getting Started. (2021).* Retrieved March 26, 2021, from https://reactjs.org/docs/getting-started.html
- *Higher-Order Components. React: A JavaScript library for building user interfaces [online]*. Palo Alto, CA: Facebook, 2021 [cit. 2021-02-12]. Dostupné z: https://reactjs.org/docs/higher-order-components.html
- *Welcome to phpMyAdmin's documentation!. (2021). PhpMyAdmin's documentation.* Retrieved February 13, 2021, from https://docs.phpmyadmin.net/en/latest/
- *NPM / CLI: the package manager for JavaScript. (2021).* Retrieved February 20, 2021, from https://github.com/npm/cli
- *ECMAScript® 2021 Language Specification: Draft ECMA-262 / February 19, 2021. (2021).* Retrieved February 20, 2021, from https://tc39.es/ecma262/

- *Introducing JSX. (2021).* Retrieved February 20, 2021, from https://reactjs.org/docs/introducing-jsx.html
- *Components and Props. (2021).* Retrieved February 20, 2021, from Components and Props
- *Apache: HTTP Server Documentation. (2020).* Retrieved February 27, 2021, from https://httpd.apache.org/docs/
- *Abbreviations.com: STANDS4 LLC. (2021). XAMPP*. Retrieved March 13, 2021, from https://www.abbreviations.com/term/1399280
- *What is a REST API?. (2021). Mulesoft.com.* Retrieved March 05, 2021, from https://www.mulesoft.com/resources/api/what-is-rest-api-design
- *Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures [Disseration]*. University of California, Irvine.
- *MySQL 8.0 Reference Manual: Including MySQL NDB Cluster 8.0. (2021).* Retrieved March 06, 2021, from https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-23.html
- *MySQL 8.0 Reference Manual: History of MySQL. (2021). MYSQL.COM.* Retrieved March 26, 2021, from https://dev.mysql.com/doc/refman/8.0/en/history.html
- *Introducing JSON. (2017).* Retrieved March 06, 2021, from https://www.json.org/json-en.html
- *US National Service Agency, & Glenn, L. (2001).* Device for and method of one-way cryptographic hashing.
- *Decoded: Examples of How Hashing Algorithms Work. (2020). Savyy Security.* Retrieved March 26, 2021, from https://cheapsslsecurity.com/blog/decoded-examples-of-how-hashing-algorithms-work/
- *Marie Helmenstine, Ph.D., A. (2019). How Many Atoms Exist in the Universe?,* https://www.thoughtco.com/number-of-atoms-in-the-universe-603795
- *SHA1 vs SHA256. (2018). Keycdn. Retrieved* March 28, 2021, from https://www.keycdn.com/support/sha1-vs-sha256
- *TechTerms: The Computer Dictionary. (2021).* Retrieved February 13, 2021, from https://techterms.com/
- *Řešení pro malé a středně velké podniky: Přehled. (2015).* SAP: Business One, 28. https://sapb1.sabris.com/FS/0633-sbo/doc/SAP_BusinessOne_Brochure_Czech_2015_final.pdf

- *Stormware Software: Sklady. (2021).* Retrieved February 22, 2021, from https://www.stormware.cz/pohoda/sklady/
- *Tech Stacks. (2021). Full Scale.* Retrieved April 02, 2021, from https://fullscale.io/blog/top-5-tech-stacks/
- *What is... user testing. (2020). Omniconvert.* Retrieved March 27, 2021, from https://www.omniconvert.com/what-is/user-testing/
- *Visual Studio Code: Code editing. Redefined. (2021).* Retrieved April 09, 2021, from https://code.visualstudio.com/
- *Draw.io: Diagram Software and Flowchart Maker. (2021).* Retrieved April 10, 2021, from https://app.diagrams.net/
- *Aplikace StockManager. (2021).* Retrieved April 09, 2021, from https://github.com/dolakzdenek/StockManager_BP

II Seznam obrázků

III Seznam ukázek zdrojových kódů

IV Seznam příloh

A Zdrojové kódy

Celý zdrojový kód k aplikaci je veřejně přístupný na stránce:

https://github.com/dolakzdenek/StockManager_BP

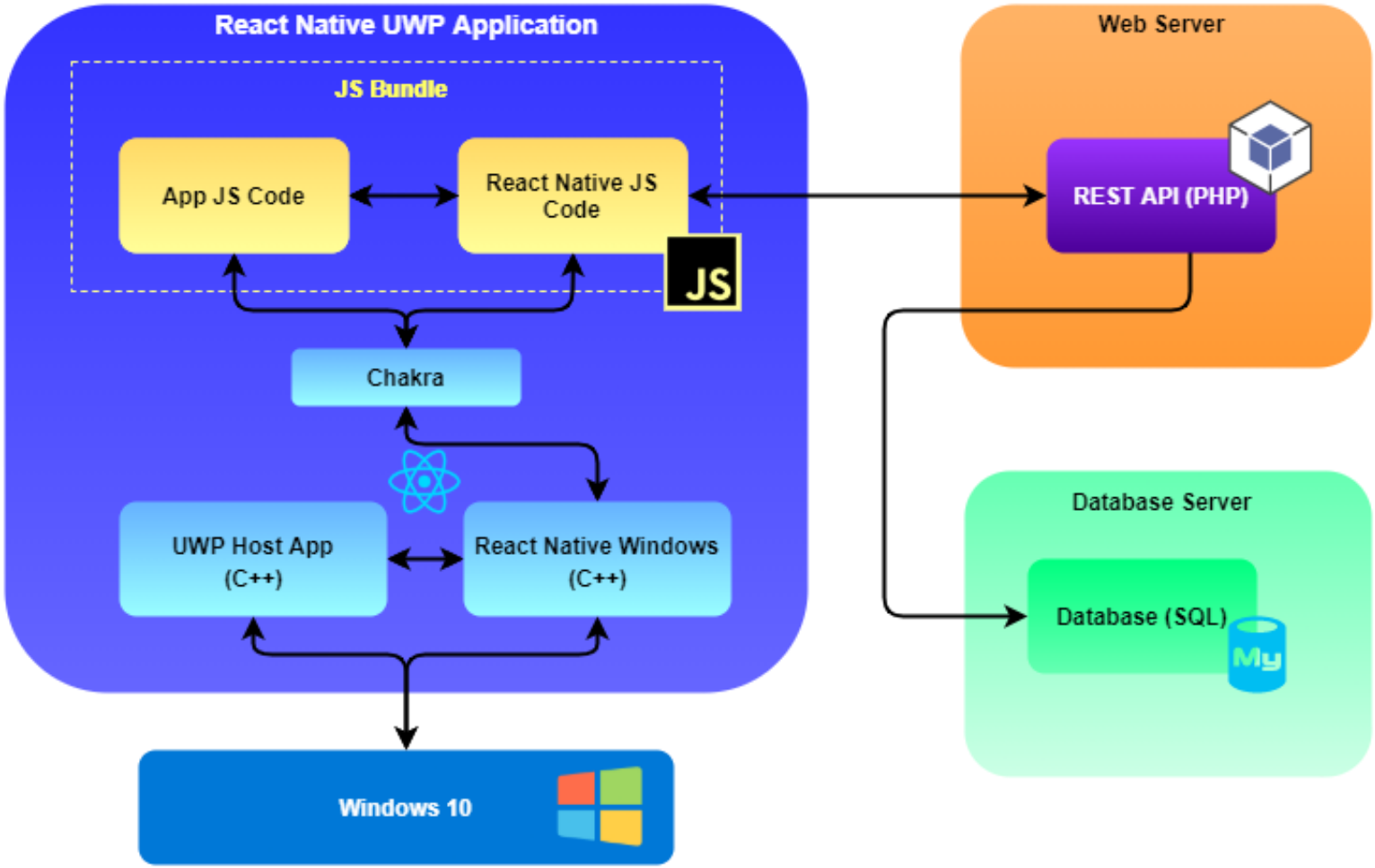
Příloha A1 – Zdrojový kód aplikace StockManager

B Obrázky v příloze

Příloha B1 - Komunikace uživatele, REST API a databáze

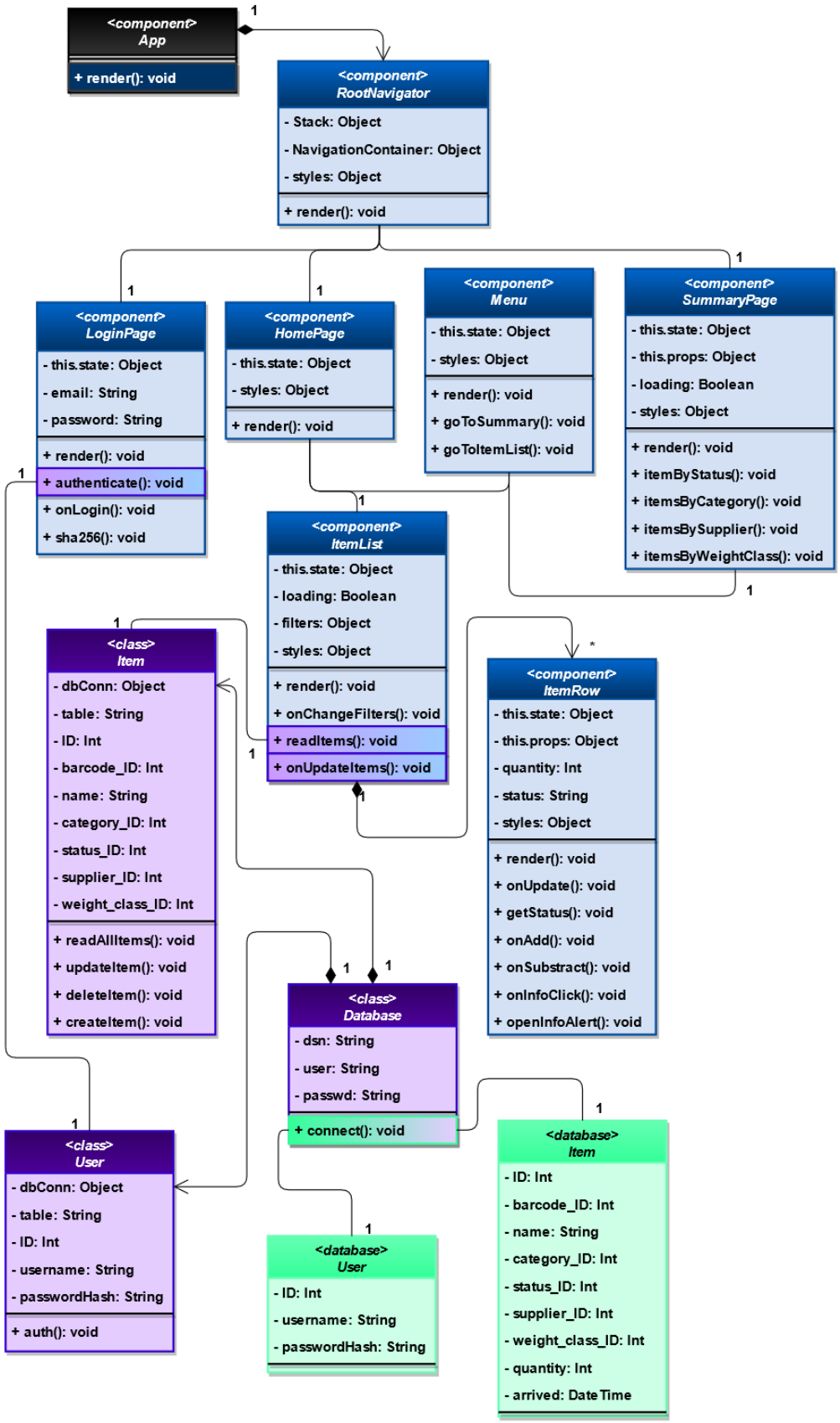
Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

Příloha B2 - Diagram architektury



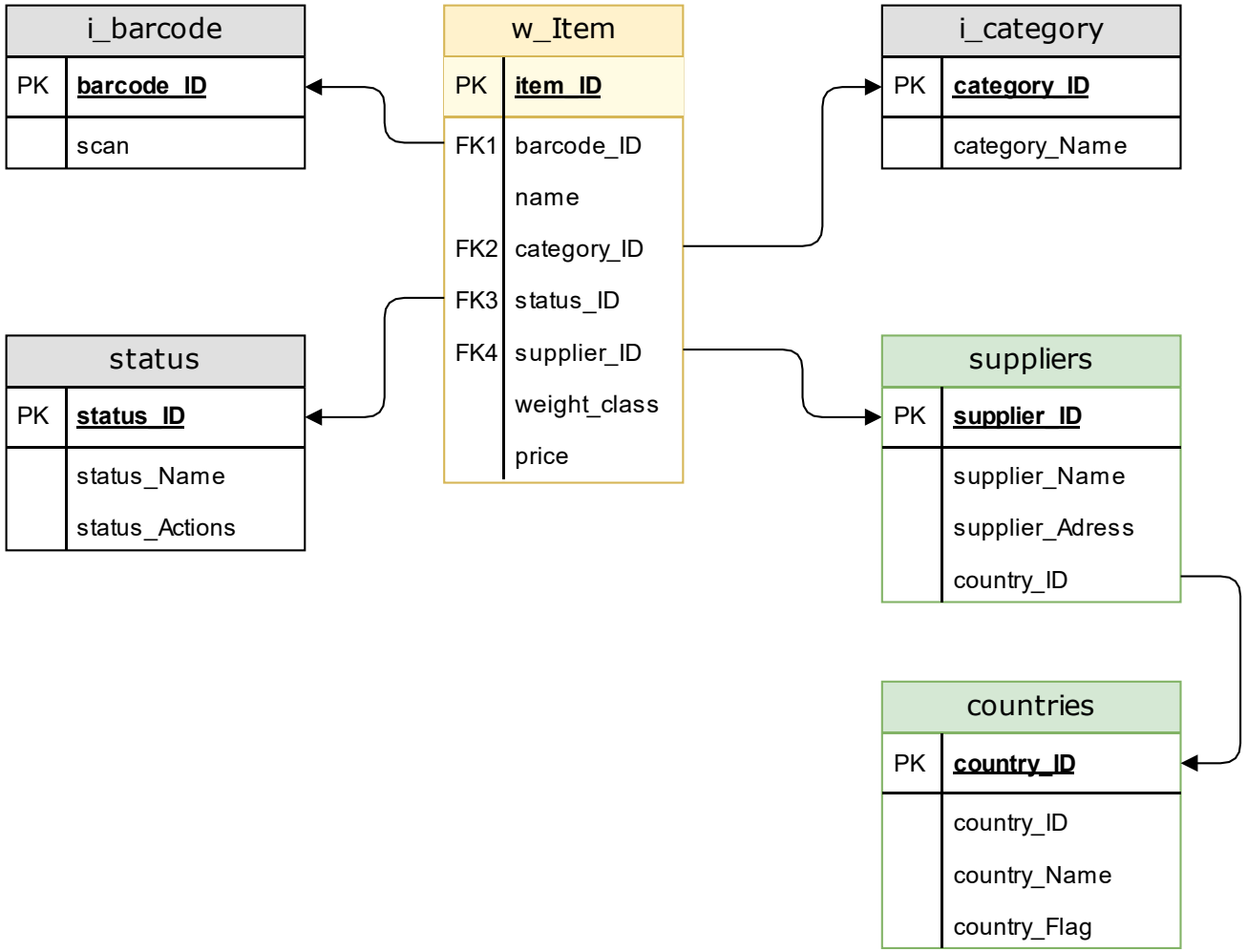
Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

Příloha B3 - Diagram tříd



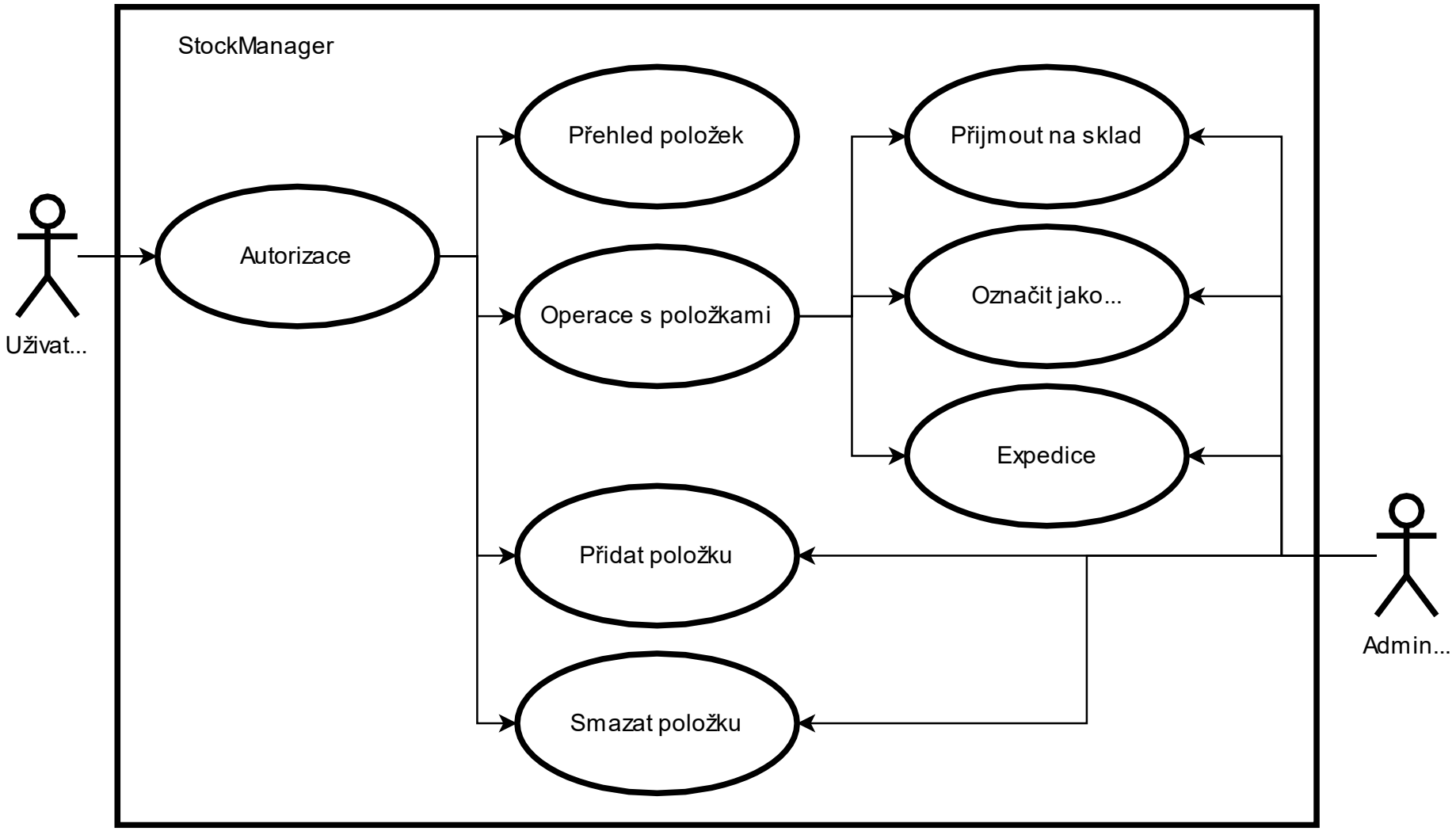
Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

Příloha B4 - Návrh databáze pro aplikaci StockManager



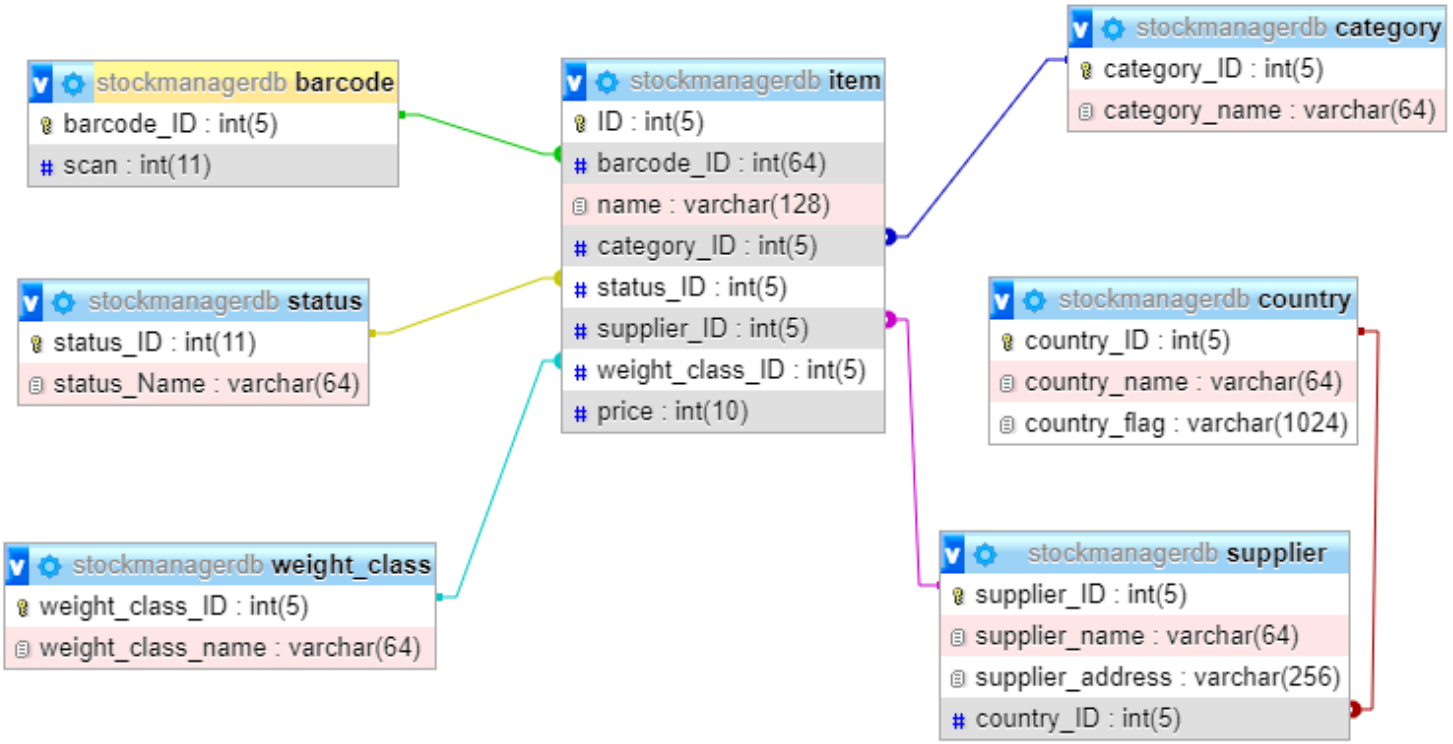
Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

Příloha B5 - Diagram užití



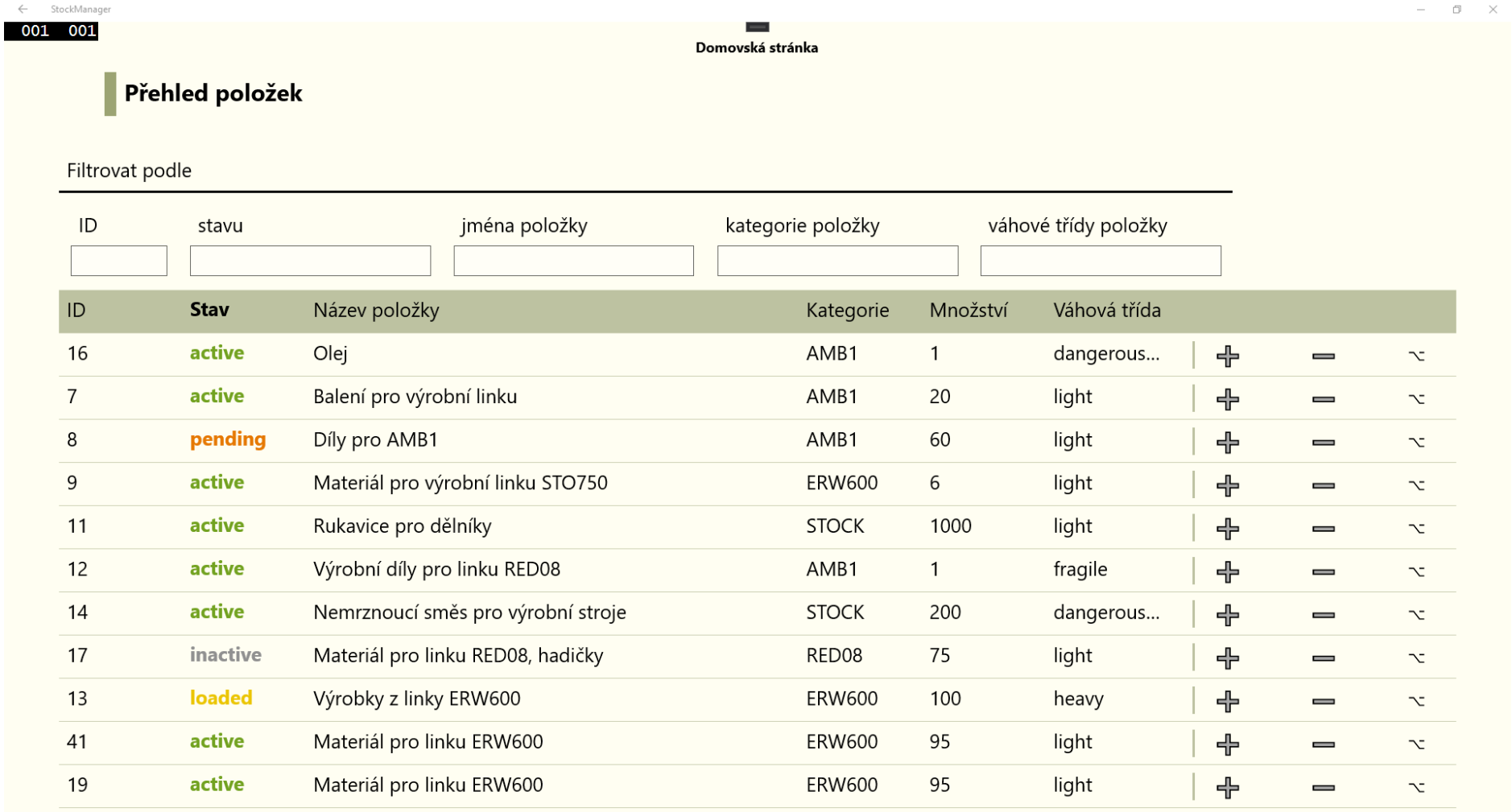
Zdroj: vlastní tvorba, Draw.io: Diagram Software and Flowchart Maker, 2021

Příloha B6 - Implementace databáze



Zdroj: převzato z Welcome to phpMyAdmin's documentation!, 2021, vlastní tvorba

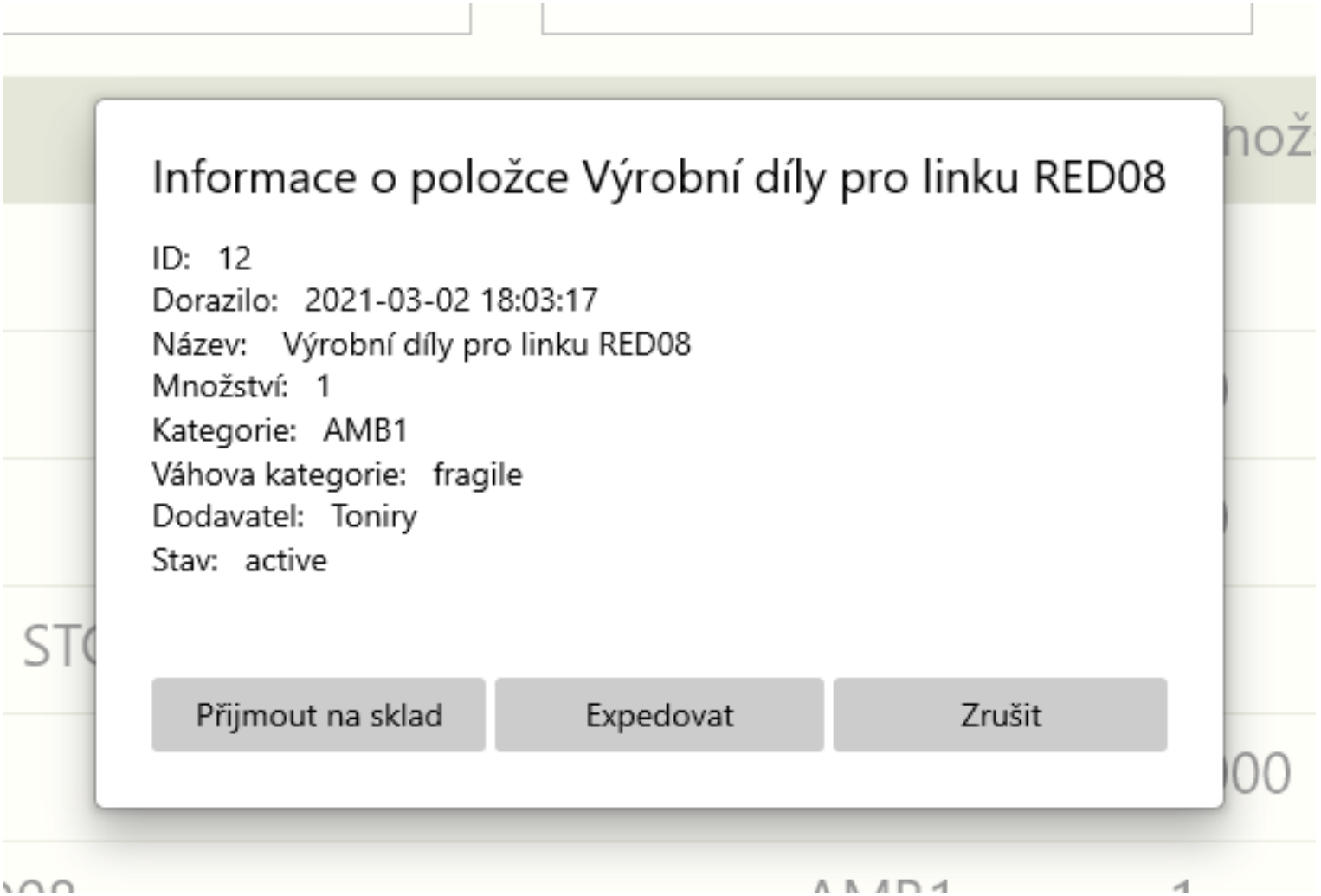
Příloha B7 - Domovská obrazovka aplikace, přehled všech dostupných položek



ID	Stav	Název položky	Kategorie	Množství	Váhová třída
16	active	Olej	AMB1	1	dangerous...
7	active	Balení pro výrobní linku	AMB1	20	light
8	pending	Díly pro AMB1	AMB1	60	light
9	active	Materiál pro výrobní linku STO750	ERW600	6	light
11	active	Rukavice pro dělníky	STOCK	1000	light
12	active	Výrobní díly pro linku RED08	AMB1	1	fragile
14	active	Nemrznoucí směs pro výrobní stroje	STOCK	200	dangerous...
17	inactive	Materiál pro linku RED08, hadičky	RED08	75	light
13	loaded	Výrobky z linky ERW600	ERW600	100	heavy
41	active	Materiál pro linku ERW600	ERW600	95	light
19	active	Materiál pro linku ERW600	ERW600	95	light

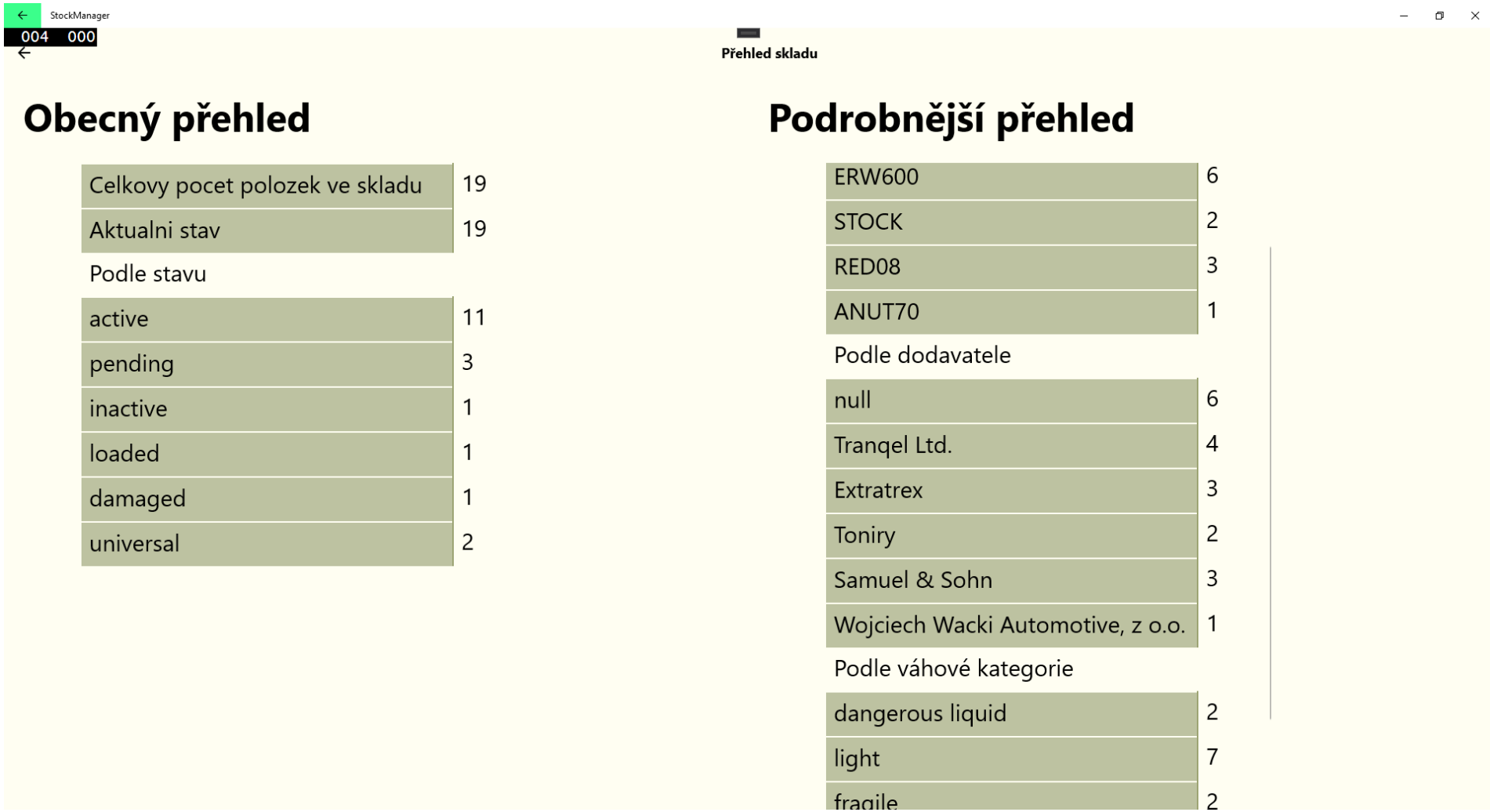
Zdroj: Aplikace StockManager, 2021, vlastní tvorba

Příloha B8 – Možnosti uživatele s položkou



Zdroj: Aplikace StockManager, 2021, vlastní tvorba

Příloha B9 - Obrazovka přehledu položek

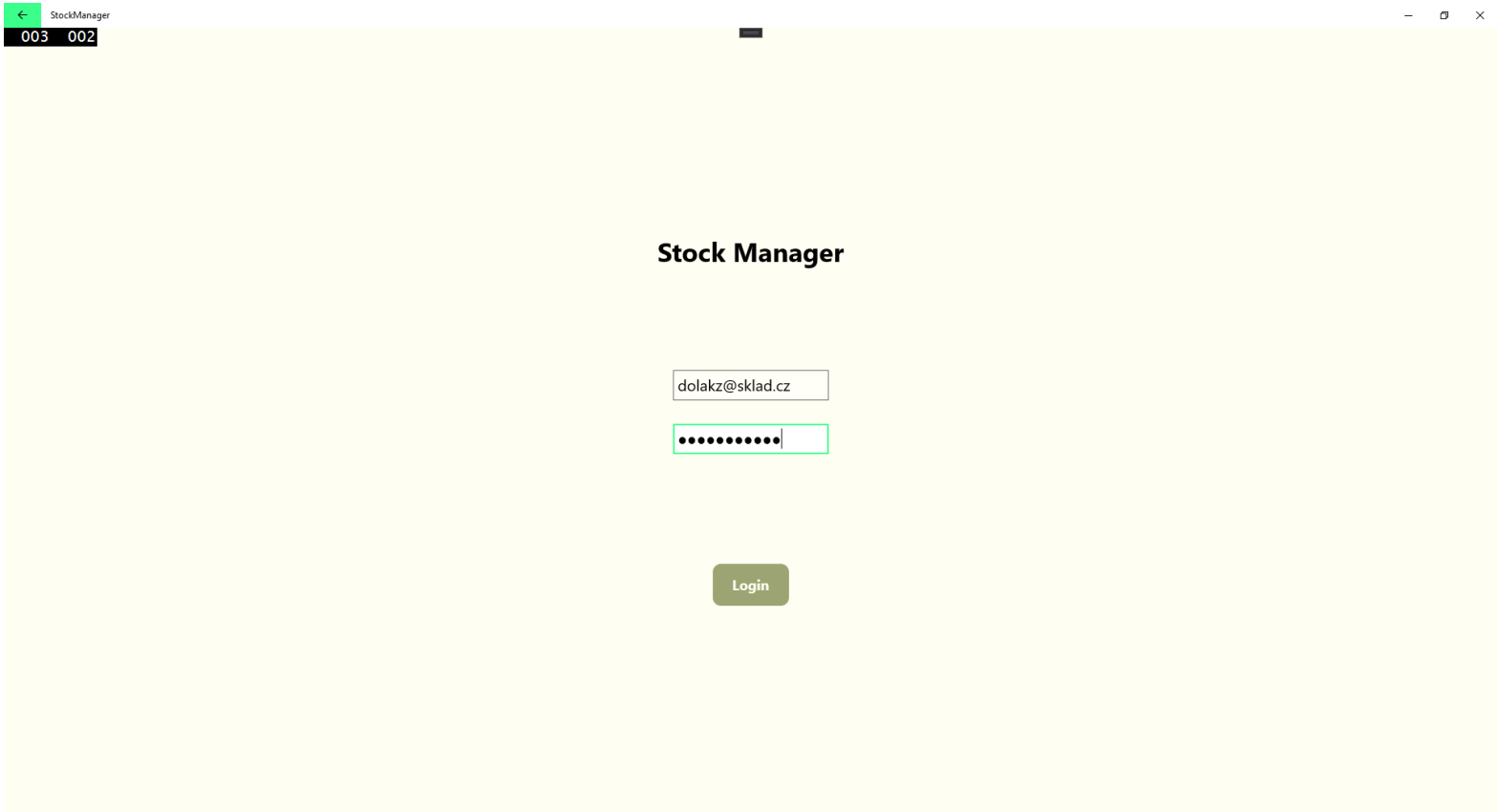


Zdroj: Aplikace StockManager,2021, vlastní tvorba

C Testovací scénář

1. Přichází nový materiál ze skladu v Polsku a operátor se přihlašuje do aplikace.

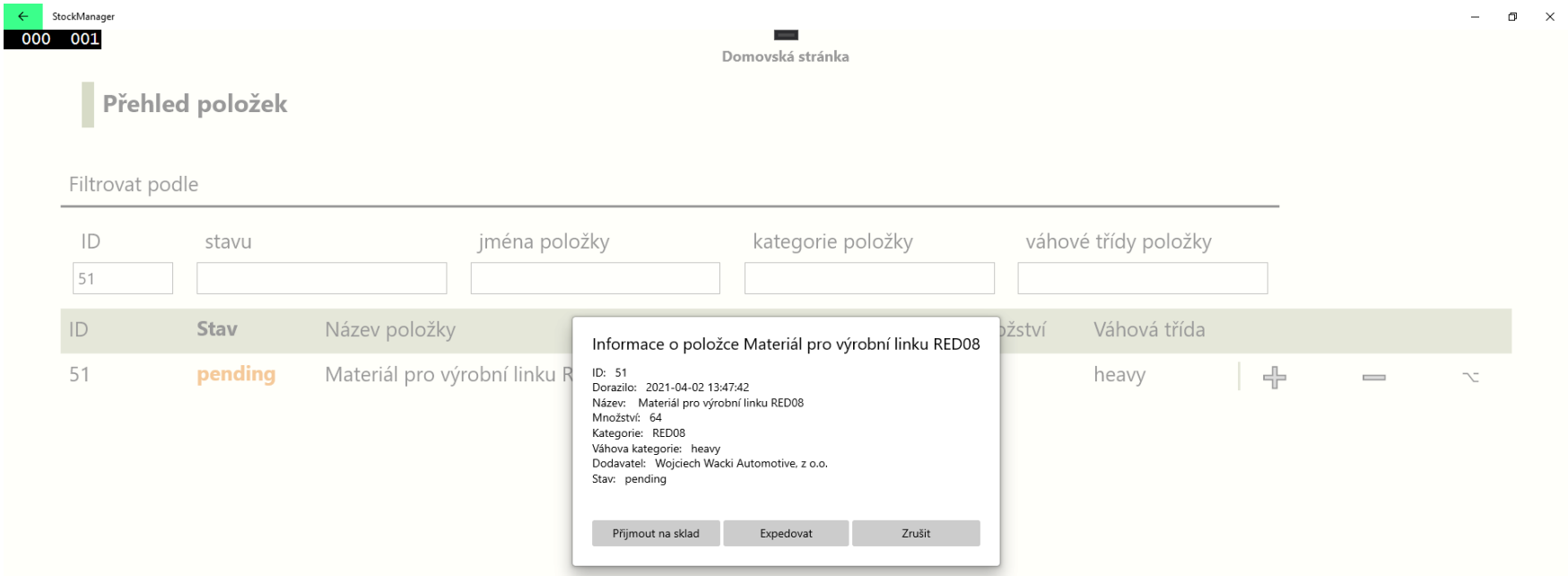
Příloha C1 - Testovací scénář krok č.1



Zdroj: Aplikace StockManager, 2021, vlastní tvorba

2. Operátor skladu si materiál vyfiltruje a přijímá jej sklad.

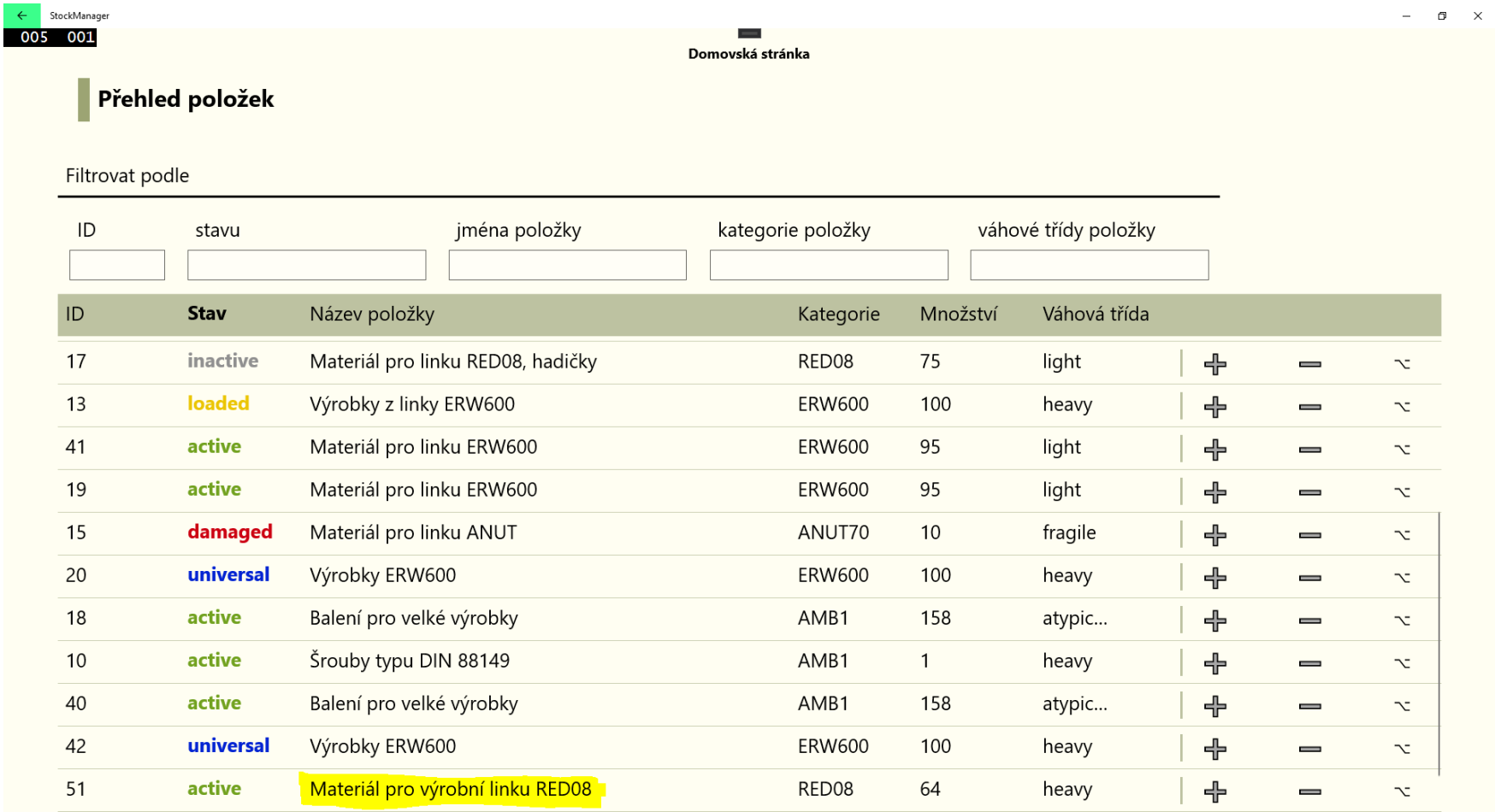
Příloha C2 - Testovací scénář krok č.2



Zdroj: Aplikace StockManager, 2021, vlastní tvorba

3. Druhý den operátor posílá materiál na výrobní linku *RED08*.

Příloha C3 - Testovací scénář krok č.3

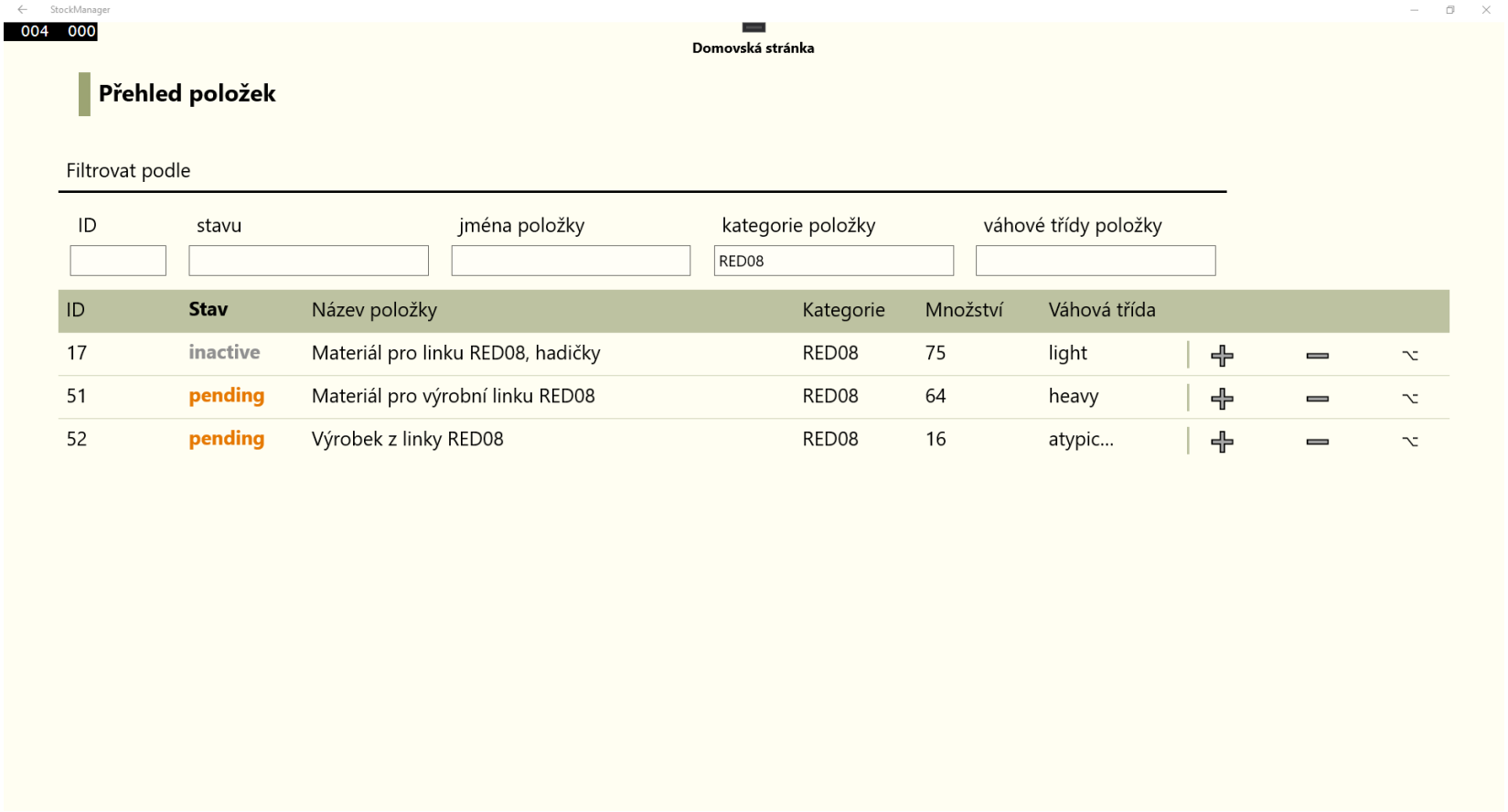


ID	Stav	Název položky	Kategorie	Množství	Váhová třída
17	inactive	Materiál pro linku RED08, hadičky	RED08	75	light
13	loaded	Výrobky z linky ERW600	ERW600	100	heavy
41	active	Materiál pro linku ERW600	ERW600	95	light
19	active	Materiál pro linku ERW600	ERW600	95	light
15	damaged	Materiál pro linku ANUT	ANUT70	10	fragile
20	universal	Výrobky ERW600	ERW600	100	heavy
18	active	Balení pro velké výrobky	AMB1	158	atypic...
10	active	Šrouby typu DIN 88149	AMB1	1	heavy
40	active	Balení pro velké výrobky	AMB1	158	atypic...
42	universal	Výrobky ERW600	ERW600	100	heavy
51	active	Materiál pro výrobní linku RED08	RED08	64	heavy

Zdroj: Aplikace StockManager, 2021, vlastní tvorba

4. O 3 dny později je materiál zpracovaný a operátor výroby vytváří novou položku *Výrobek RED08*.
5. Operátor na skladě si zboží vyfiltruje podle kategorie a přijímá výrobky na sklad.

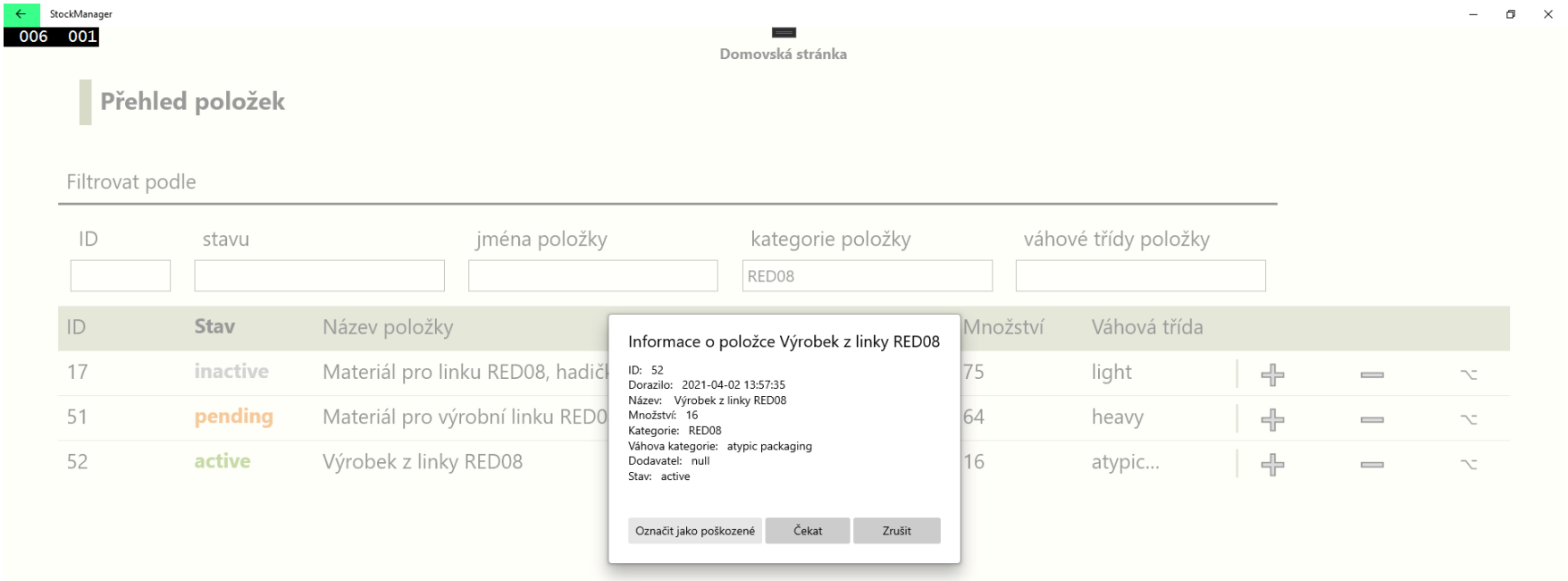
Příloha C4 - Testovací scénář krok č.5



Zdroj: Aplikace StockManager, 2021, vlastní tvorba

6. Operátor zjišťuje, že balení zboží je poškozeno a převádí položku skladu na stav *damaged* – poškozeno.

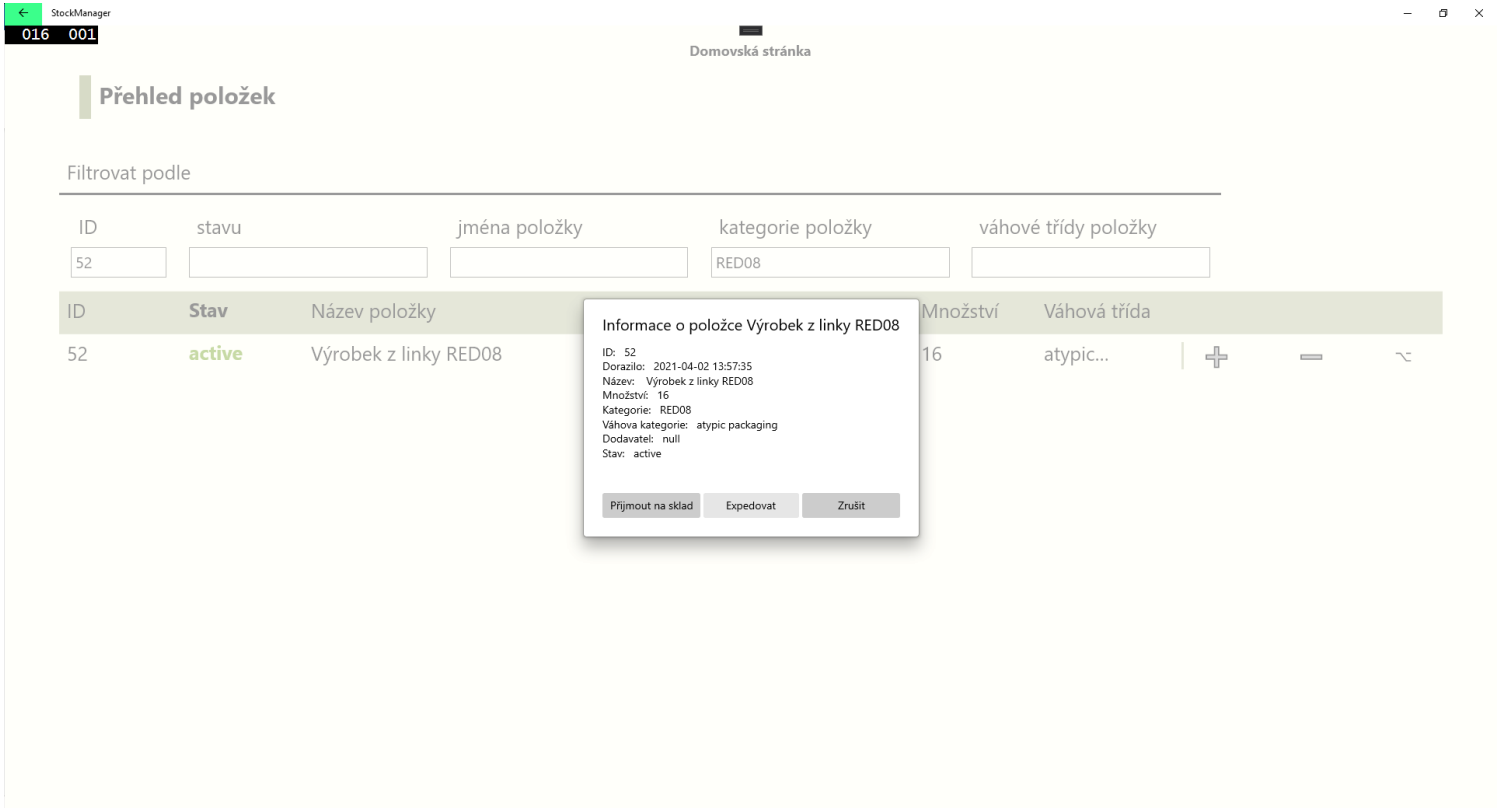
Příloha C5 - Testovací scénář krok č.6



Zdroj: Aplikace StockManager, 2021, vlastní tvorba

7. Probíhá oprava balení zboží.
8. Operátor dostává informaci, že se pro zboží našel kupec, zboží si vyfiltruje podle *ID* v databázi aplikace a vyexpeduje ho.

Příloha C6 - Testovací scénář krok č.8



Zdroj: Aplikace StockManager, 2021, vlastní tvorba

9. Zboží je odesláno.