Multiplatformní mobilní aplikace se skenerem skladových zásob

Jaroslav Bělák

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jaroslav Bělák**
Osobní číslo:	**A21045**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Multiplatformní mobilní aplikace se skenerem skladových zásob**
Téma práce anglicky:	**Multiplatform Mobile App with Stock Scanner**

Zásady pro vypracování

1. Představte framework Ionic a jeho klíčové vlastnosti pro tvorbu multiplatformních mobilních aplikací.
2. Stručně popište problematiku aplikací pro vedení skladu a uveďte běžně využívaná řešení.
3. Popište klíčové prvky a funkcionality vlastního CRM systému zadavatele a navrhněte, jak do něj bude aplikace integrována.
4. V praktické části navrhněte a implementujte aplikaci v souladu s potřebami a specifikacemi CRM systému, a zdokumentujte postup tvorby aplikace včetně procesu skenování a sledování dostupnosti produktů.
5. Popište proces návrhu uživatelského rozhraní aplikace včetně použití komponent frameworku Ionic pro zajištění intuitivního a uživatelsky přívětivého prostředí.

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

Seznam doporučené literatury:

1. Introduction | Vue.js. Vue.js – The Progressive JavaScript Framework | Vue.js [online]. 2023 [cit. 2023-11-06]. Dostupné z: https://vuejs.org/guide/introduction.html
2. Introduction to Ionic. Ionic Framework – The Cross-Platform App Development Leader [online]. 2023 [cit. 2023-11--06]. Dostupné z: https://ionicframework.com/docs
3. TypeScript: JavaScript With Syntax For Types. TypeScript Documentation [online]. 2023 [cit. 2023-11-06]. Dostupné z: https://www.typescriptlang.org/docs/
4. AU-YEUNG, John. Vue.js 3 By Example: Blueprints to learn Vue web development, full-stack development, and cross--platform development quickly. Packt Publishing, 2021. ISBN 978-1838826345.
5. FREEMAN, Adam. Essential TypeScript 5. Third Edition. Manning, 2023. ISBN 978-1633437319.
6. GOLDBERG, Josh. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript. O'Reilly Media, 2022. ISBN 978-1633437319.

Vedoucí bakalářské práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

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

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.
- že při tvorbě této práce jsem použil/a nástroj generativního modelu AI ChatGPT, Gemini a Bing AI za účelem účelem korektury, vylepšení formulace a rešerše. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne .....

podpis studenta

ABSTRAKT

Bakalářská práce se zabývá tvorbou multiplatformní aplikace na skenování položek ve skladu za pomocí frameworku Ionic. Aplikace je vytvořena na míru k vlastnímu CRM systému zadavatele tak, aby společnosti pomáhala s kontrolou dostupností produktů, která pak následovně povede k větší spolehlivosti procesu objednávky. V teoretické části jsou popsány technologie Vue a Ionic Framework, které byly použity k vytvoření aplikace. V praktické části je řešen návrh a implementace aplikace s ohledem na CRM systém a návrh uživatelského rozhraní aplikace pomocí komponent ve frameworku Ionic.

Klíčová slova: multiplatformní aplikace, Vue, Ionic Framework, Ionic, skladové systémy, skenování

ABSTRACT

The bachelor thesis deals with the development of a multiplatform mobile app for scanning items in the warehouse using Ionic Framework. The application is custom-built for the client's CRM system to assist in monitoring product availability, thereby leading to greater reliability in the ordering process. The theoretical part describes the Vue and Ionic Framework technologies used in the application development. The practical part focuses on the design and implementation of the application with regard to the CRM system and designing the application's user interface using components within the Ionic framework.

Keywords: multiplatform application, Vue, Ionic Framework, Ionic, inventory systems, scanning

Chtěl bych poděkovat panu Ing. Radku Valovi Ph.D., za jeho vedení této práce a za cenné úpravy, které mi poskytl. Dále bych chtěl poděkovat své rodině a blízkým za jejich podporu, a také bych rád poděkoval Michalu Štramberskému za případné další návrhy k této práci.

OBSAH

ÚVOD

V dnešní době se procesy správy skladů stále více automatizují pomocí technologie skenerů. Vzhledem k tomu, že zadavatel dosud nevyužíval tuto technologii, požádal o implementaci nového řešení, jehož cílem je zefektivnit kontrolu dostupnosti produktů pomocí mobilní aplikace s funkcí skenování. Díky tomu je pak proces objednání rychlejší, což vede ke zvýšení efektivity skladového hospodářství.

V teoretické části práce jsou nejprve uvedeny základní informace o frameworcích Ionic a Vue. Ionic Framework umožňuje vývojáři vytvářet multiplatformní mobilní aplikace přes webové technologie a framework Vue umožňuje jednoduše vytvářet webové rozhraní pomocí šablonového syntaxu a znovupoužitelných komponent. V třetí kapitole teoretické části je stručně popsaná teorie skladových systémů a je vyjmenováno několik běžných řešení.

V úvodu praktické části je představen CRM systém zadavatele včetně jeho funkcionalit, což umožňuje definovat způsob integrace stávajícího systému s plánovanou mobilní aplikací. Poté je detailně popsán samotný koncept této integrace. Dále je shrnuta zadavatelem stanovená požadovaná funkcionalita mobilní aplikace a z ní vyvozené funkcionální a nefunkcionální požadavky. Následuje popis důležitých implementačních kroků, jako je implementace způsobu komunikace s REST API, tvorba uživatelského rozhraní a realizace zejména funkcionálních požadavků na aplikaci. Pro lepší vizualizaci je text doplněn ukázkami kódu a obrazovek aplikace.

V závěrečném shrnutí je uvedeno zhodnocení splnění požadavků zadavatele a také vyjádření k projektu mobilní aplikace od zadavatele.

I. TEORETICKÁ ČÁST

1 IONIC FRAMEWORK

Ionic Framework je toolkit s otevřeným zdrojovým kódem, díky kterému lze stavět mobilní aplikace pomocí webových technologií (na rozdíl od Swift v iOS, Kotlin či Java v Androidu či přes platformu Xamarin v C#), a který umožňuje vytvoření aplikace do platforem Android a iOS. Prostřednictvím Ionic Framework lze dále vytvářet desktopové aplikace či progresivní webové aplikace. [1]

1.1 Komponenty k uživatelskému rozhraní

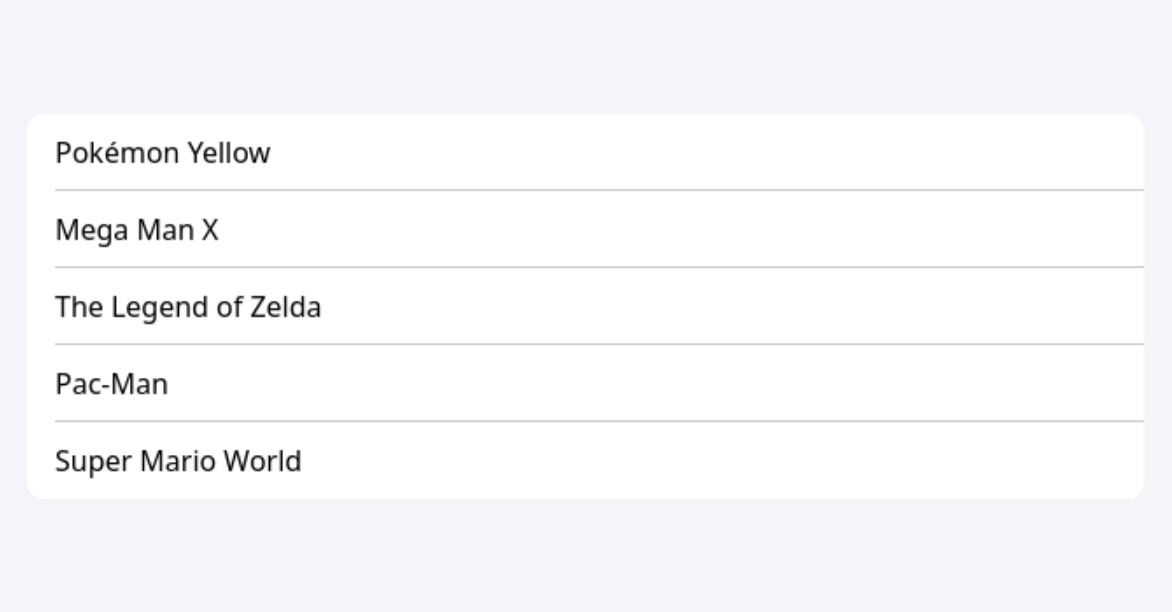
Ionic Framework nabízí různé komponenty k uživatelskému rozhraní, které jsou implementovány pomocí technologie StencilJS, která pak používá sadu technologií Web Components. Tato sada technologií umožňuje do HTML dokumentu implementovat vlastní prvky, jenž je následně možno používat a konfigurovat jako běžné HTML prvky. Motiv komponentů lze dále nastavovat pomocí proměnných v CSS. [2]

Prvek „Content“ slouží jako jeden ze tří prvků nejvyšší úrovně, které by měly být v šabloně pouze jednou, ostatní jsou Header a Footer. Obsahuje metody pro jemný přesun na vrch stránky nebo na spodek stránky. U tohoto prvku lze modifikovat padding, margin nebo barva pozadí. [3]

Prvek „IonPage“ u navigace ve Vue umožňuje funkcionalitu přechodů mezi obrazovkami a funkcionalitu navigační haldy přes Ionic Framework. Je nutné mít tento prvek v šabloně pro každou obrazovku, pro níž má být tato funkcionalita dostupná. [4]

Prvek „Grid“ implementuje mřížkový systém založený na systému flexbox v CSS. [5] Sloupce se rozšíří, aby vyplnily řádky a zakládá se na 12 sloupcovém layoutu, který je podobný jako u CSS frameworku Bootstrap. Počet sloupců však lze modifikovat. [6]

Prvek „List“ a „Item“ jsou prvky, které se používají společně. List je používán pro seznamy. Obsah prvku List představují řádky prvků Item. [7] Item by se měl používat pouze v případě toho, že je prvek v prvku List. Itemy mohou být modifikovány, mazány, přetříděny. [8]



Obrázek 1.1 Příklad prvku „List“ s vnořenými prvky „Item“ [7]

1.2 Adaptivní motiv aplikace

Ionic Framework automaticky přizpůsobí vzhled svých komponentů podle platformy, na níž je aplikace spuštěna. V případě Androidu nebo jiné platformy je použit Material Design, v případě iOS je použit design iOS. Motiv aplikace lze nastavit. [9]

1.3 Ionic Capacitor

Ionic Framework se nejčastěji používá k vývoji mobilních aplikací, a tudíž je žádoucí mít i přístup k nativním funkcím jako fotoaparát, akcelerometr, GPS, notifikace apod. Toto je umožněno prostřednictvím běhového prostředí Ionic Capacitator, které představuje vrstvu, na níž je Ionic Framework postaven. Ionic Capacitor je kompatibilní s různými webovými frameworky, např. React, Angular, Svelte nebo Vue, který byl použit v dalších částech této práce. [10] [11]

1.4 Ionic CLI

Aplikace vytvořené přes Ionic Framework se jsou primárně vyvíjeny přes nástroj v příkazové řádce Ionic, tedy Ionic CLI. Pomocí tohoto nástroje můžeme např. vytvořit projekt (ionic start) nebo spustit dočasný webový vývojový server (ionic serve). Přes Ionic CLI taky lze interagovat s Ionic Capacitorem, např. příkazem „ionic capacitor run“ je mobilní aplikace sestavena a nasazena do emulátoru či přípojného telefonu. [12]

1.4.1 Příkaz „ionic start“

Prostřednictvím tohoto příkazu lze vytvořit projekt s Ionic Frameworkem. Pro tento příkaz jsou dostupné 3 šablony: blank (prázdný projekt s jednou stránkou), sidemenu (jedna stránka s bočním menu) a tabs (jedna stránka se 3 kartami). [13] Lze taky vybrat framework, se kterým se má pracovat, a z toho jsou na vybranou „vue“, „angular“, „angular-standalone“ a „react“ (angular-standalone je pro projekty v Angular využívající nové Standalone API). [14]

1.4.2 Příkaz „ionic serve“

Tento příkaz umožňuje spustit vývojářský server, na který se pak otevře okno prohlížeče, aby se k serveru připojil. Server sleduje změny v souborech a při každé změně automaticky znovu načte a sestaví soubory pro vývojářský server. Lze specifikovat port, specifikovat, jestli se mají generovat zdrojové mapy, jestli se má vypisovat výstup z konzole, nebo jestli zviditelnit vývojářský server pro LAN. Je možné použít SSL, ale je experimentální. [15]

2 VUE

Vue je webový framework pro tvorbu uživatelských rozhraní. Konkuruje s frameworky jako Svelte, React nebo Angular.

2.1 Deklarativní renderování

Framework Vue umožňuje renderování dat do DOMu pomocí šablonového syntaxu. Jako jedna ze základních funkcionalit jsou dvojté složené závorky, do kterých se vloží název proměnné. Například, pokud máme proměnnou „test“ s hodnotou 1, a vložíme do šablony `{{test}}`, tak v šabloně se pak načtením Vue promění proměnná s dvojtými závorkami v současný stav proměnné „test“, a vzniká tak vazba s DOM a proměnnou ve Vue. [16]

2.1.1 Binding atributů

Jelikož dvojté závorky nejdou použít v atributech prvků v HTML, tak se pro utvoření vazby mezi hodnotou atributu v HTML a proměnnou ve Vue (binding) používá direktiva „v-bind“. Pokud si vezmeme stejnou proměnnou, a chtěli bychom ji vložit do atributu „count“ v prvku div, tak bychom napsali `<div v-bind:count="test"></div>`

Zkratka pro binding atribut Abychom nemuseli pořád psát „v-bind“, tak můžeme z minulého příkladu napsat šablonu jako `<div :count="test"></div>`.

Zkratka pro binding atribut, kde se shoduje název proměnné s názvem atributu Pokud je název atributy stejný jako název proměnné, se kterou se má vytvořit binding, tak můžeme z napsat šablonu jako `<div :nazev-promenne></div>`.

2.2 Reaktivita

Vue sleduje změny stavu v JavaScriptu a aktualizuje DOM ve chvíli, kdy nastanou změny. Toho je dosaženo ve Vue 3 za pomocí JavaScriptového objektu „Proxy“. Poté se spouští Watcher objekty ve Vue, které hlídají změnu dat spuštěnou přes Proxy objekt. Vnitřně se pak používá fronta pro efektivnější aktualizaci DOMu. [17]

2.3 Single-File Components

Vue disponuje funkcionalitou, do které lze v jednom souboru definovat šablonu, logiku a stylování. Je s nimi jednodušší obsluhovat Composition API, umožňuje podporu Hot Reload, nabízí podporu s IDE a pomocí `<style scoped>` lze dokonce mít i CSS pouze

v rámci komponenty. Single-File Components však vyžadují kompilaci přes Vite či Vue CLI. [18]

2.4 Styly API

Vue má 2 styly API: Options API a Composition API, přičemž se spíše preferuje Composition API.

2.4.1 Options API

Pomocí Options API se definuje komponenta pomocí polí objektu, a k datům definovaným ve funkci „data“ se přistupuje pomocí „this“. [19]

```
export default {
        data() {
                return {
                        promenna: 0
                }
        },
        methods: {
                zmenitPromennou() {
                        this.promenna++;
                }
        },
}
```

Níže je sofistikovanější příklad využití Options API s možnostmi „watch“ a „computed“. Ve „watch“ lze napsat funkci s názvem proměnné, pro kterou se mají sledovat změny, a ve „computed“ lze definovat výpočty proměnných, které závisí na jiných proměnných a automaticky se aktualizují při jejich změně.

```
export default {
        data() {
                return {
                        promenna: 0
                };
        },
        methods: {
                zmenitPromennou() {
                        this.promenna++;
                }
        },
        computed: {
                dvojnasobek() {
                        return this.promenna * 2;
                }
        },
        watch: {
                promenna(newValue, oldValue) {
                        console.log(`Změna proměnné z ${oldValue} na ${newValue}`);
                }
        }
};
```

2.4.2 Composition API

S Composition API se definuje logika komponenty pomocí importované funkce. Typicky se používá se Single-File Componenty pomocí syntaxe `<script setup>`. [19]

```
<script setup>
import { ref } from 'vue';
```

```
        const promenna = ref(0);
        function zmenPromennou() {
                promenna.value++;
        }
        </script>
```

Composition API lze použít i bez potřeby nástrojů na sestavení, a tedy i bez Single-File Componentů.

```
const { ref } = Vue; //předpokládá globální Vue objekt
export default {
        setup() {
                const promenna = ref(0);
                const zmenPromennou = () => {
                        promenna.value++;
                };
                return {
                        promenna,
                        zmenPromennou
                };
        },
        template: `<div><button @click="zmenPromennou">Změň
↪  proměnnou</button>{{promenna}}</div>`
};
```

3 PROBLEMATIKA APLIKACÍ PRO VEDENÍ SKLADU A BĚŽNÁ ŘEŠENÍ

V této části se popisuje problematika aplikací pro vedení skladu a několik běžně užívaných řešení pro srovnání se systémem zadavatele. Následující podkapitoly popisují definici problematiky aplikací a různé typy aplikací.

3.1 Základní úkoly skladových aplikací

Skladové aplikace (či skladové systémy) jsou aplikace, které umožňují hospodařit se skladem. Skladové aplikace by měly řešit následující problematiku:

- Evidence stavu zásob: systém nabízí přehled o zásobách a dokáže zobrazit atributy zboží (např. název, množství, aktuální objem zásob, datum příjmu, datum expirace zboží apod.). [20]
- Sledování pohybů na skladě: skladový systém dokáže sledovat pohyby na skladě a na základě vyhodnocení těchto pohybů můžeprovést požadované přednastavené automatické akce (např. upozornění na nedostatek zboží.)
- Vychystávání a balení objednávek: skladový systém navádí skladníka nejkratší trasou při vychystávání objednávek a kontroluje správnost a množství odebíraného zboží. [21]
- Integrace s dalšími systémy: skladový systém by měl zvládnout synchronizovat data se systémy např. e-shopu nebo ERP. [21]
- Bezpečnost dat: skladový systém by měl zajistit, aby včasně opravoval bezpečnostní trhliny a aby byl software napsaný s nadhledem pro bezpečnost.
- Reporting: skladový systém automaticky generuje reporty o provozu a zásobách v skladu. Tyto reporty mohou obsahovat požadované informace o skladu a pohybech v něm, např. údaje o rychlosti a množství vychystaných objednávek za zvolený čas, procentuální přesnosti vychystávání i balení nebo produktivitě personálu. [21]

3.1.1 Typy aplikací

Skladové aplikace je možno dělit podle ceny a vlastnictví a podle implementace. Podle ceny lze rozdělit skladové aplikace na: [22]

- skladové aplikace s otevřeným zdrojem,
- skladové aplikace, které jsou zdarma (freeware),

- placené skladové aplikace,

Podle implementace lze rozdělit skladové aplikace na:

- samostatná skladová aplikace provozovaná v prostorách firmy,
- cloudová skladová aplikace provozovaná na serverech u tvůrců skladové aplikace,
- skladová aplikace integrovaná s ERP

Skladové aplikace s otevřeným zdrojovým kódem
Skladové aplikace s otevřeným zdrojovým kódem mají přístupný zdrojový kód, kterým je možno modifikovat program podle svých potřeb. Tyto aplikace bývají udržovány organizací nebo skupinou dobrovolníků. Zisk pak mají vývojáři obvykle z placené zákaznické podpory. Z toho pak plynou výhody se symbiotickým vztahem, a to sice takovým, že vývojář je zaplacen za přidávání nových funkcí, které pak mohou všichni využít, a zákazník pak může tyto funkce využít.

Freeware skladové aplikace
Freeware skladové aplikace aplikace, na rozdíl od tabulkových procesorů, dokáží automatizovat procesy, vyznačují se přehlednějším rozhraním a obsahují více funkcí. Nevýhoda využívání freeware skladových aplikací však spočívá ve vyloučení odpovědnosti, a to ať už ukončením podpory pro program, nebo nedodržováním legislativy. [22] Nevýhodou oproti programům s otevřeným zdrojem je nemožnost přizpůsobovat software pro své potřeby.

Placené skladové aplikace
"Placené aplikace mají uzavřený zdrojový kód, což vylučuje možnost modifikovat program podle potřeb uživatele přímo uživatelem. Bývá ovšem možné požádat vývojáře aplikace o přidání dalších funkcí. Mezi programy tohoto typu patří SAP, MoneyERP, POHODA (mimo licenci Start) nebo K2.

Samostatná skladová aplikace
Samostatné skladové aplikace bývají obvykle nasazovány ve vlastních prostorách firmy s vlastním hardwarem. Obecně mohou podporovat větší přizpůsobení (i když mohou být nákladné) a organizace může udržovat přísnější kontrolu nad svými daty a softwarem. [23] Je vhodný pro menší a střední organizace, které hledají samostatný a specializovaný systém pro správu svých skladových operací. [24]

Cloudová skladová aplikace
Cloudové skladové aplikace lze rychle nasadit s nižšími počátečními náklady. [23] Hlavní výhodou cloudových skladových aplikací je absence požadavku vlastního hardwaru. Nevýhodou však je, že když zanikne společnost poskytující službu, tak zanikne spolu s ním i skladová aplikace. V rámci státu je nutno ještě zvážit, jestli lze cloudová aplikace používat s legislativou danou státem. Takovým příkladem je nutnost zvážit, jestli vyhovuje cloudová aplikace regulaci GDPR.

Integrovaná skladová aplikace
Některé systémy řízení skladu jsou sestaveny jako moduly nebo aplikace, které se integrují s platformami ERP. [23] Umožňuje synchronizaci dat a procesů mezi správou skladu a ostatními oblastmi podniku. [24]

3.2 Běžná řešení

V této podkapitole se vyjmenuje několik skladových systémů.

POHODA je integrovaný skladový systém. POHODA usnadňuje práci se skladovými zásobami pomocí náhledů zboží a čárových kódů, které se využívají také při inventuře. Dále poskytuje dokonalý přehled o zásobách doplněný o informace o zárukách, šaržích a výrobních číslech. [25] Zásoby se oceňují pořizovacími cenami, tedy součtem ceny pořízení a nákladů, které souvisejí s jejich pořízením. Na skladě se zásoby oceňují průměrnou pořizovací cenou, která byla zjištěna jako vážený aritmetický průměr z pořizovacích cen nebo z vlastních nákladů a množství zásob na skladě. [26]

AutoERP je, mimo jiné, skladový systém, který je open-source. Vyznačuje se tím, že je zdrojový kód majetkem zákazníka, který jej pak může použít k účelům podle licence GPLv3. Umožňuje skenování přes zařízení Zebra, který se pak aplikací propojí s ERP systémem. AutoERP má možnost vlastní customizace bez potřeby znalosti kódu, jedná se tedy o nocode platformu, tedy platforma, kde se dá vytvářet software bez nutnosti psát kód. [27] Podle uživatelů není vzhled moc zajímavý, a minimální cena je 8000 Kč měsíčně, aby bylo možné použít modul skladového systému. [28] [29]

K2 je integrovaný skladový systém, který umožňuje řízení obchodních případů pomocí evidence dokladů a jejich stavů, které znázorňují jejich průběh. [30] Na tzv. kartě zboží jsou základní údaje, jako jsou cena, jednotka, rozměry či hmotnost, ale i podrobnější evidence dle šarží. [31] K2 podporuje řízení skladu pomocí čteček čárových kódů a má integrované základní principy známé z WMS systémů. [32] Detailní evidence

skladových pohybů o položce pomáhá skladovou evidenci rozdělit na příslušné doklady, které danou položku obsahují (tj. příjemky, výdejky, rezervační listy atp.)

SAP EWM je oproti ostatním uvedeným systémům používaný celosvětově. SAP Extended Warehouse Management (SAP EWM) je modul/systém pro správu skladů, který je součástí balíčku řešení SAP Supply Chain Management (SAP SCM). [33] SAP Extended Warehouse Management (EWM) se používá k efektivní správě zásob ve skladu a k podpoře zpracování pohybu zboží. Umožňuje společnosti kontrolovat příchozí a odchozí procesy ve skladu a pohyb zboží ve skladu. Hlavním procesem ve skladu je příjem a výdej materiálu, příjem a výdej zboží, plnění objednávek zákazníků a distribuce zboží. Když společnost neskladuje žádné zboží, není potřeba, aby skladové hospodářství spravovalo zboží. [34]

3.3 Shrnutí problematiky

Běžně používaná řešení si jsou funkcionalitou velmi podobné. Hlavním rozdílem je však cena, vlastnictví a implementace. Pokud by si uživatel přál mít systém, který je mu otevřený, může zvolit AutoERP. Pokud chce uživatel mít jistotu nad českou legislativou, tak je vhodnější spíše zvolit český skladový systém. Pokud někdo nechce spoléhat na ostatní společnosti, může si vytvořit systém na míru, což si právě vybral zadavatel.

II. PRAKTICKÁ ČÁST

4 CRM SYSTÉM ZADAVATELE

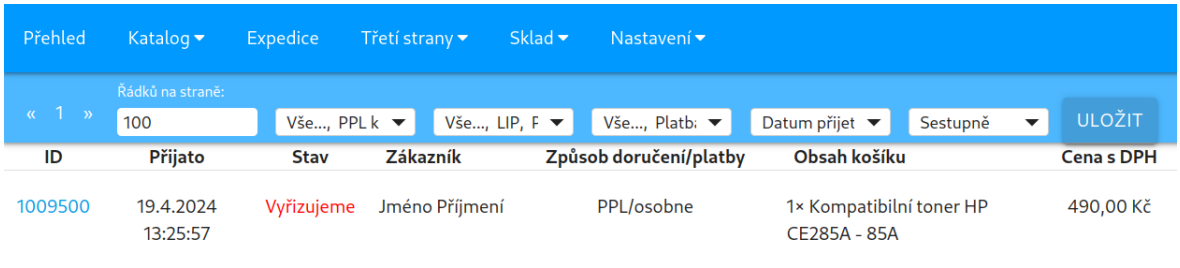
CRM systém je zabudován do stejného systému jako e-shop samotný.

4.1 Frontend CRM systému

Frontend je běžný HTML a CSS frontend s několika JS funkcionalitami, které řeší asynchronní komunikaci se serverem.

4.1.1 Objednávky

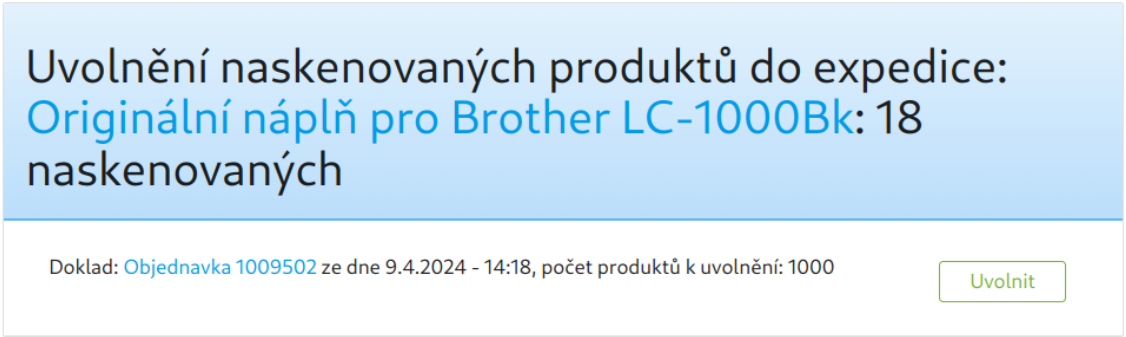
Objednávky mají obvyklé údaje, jako ID, čas přijetí, stav objednávky, jméno zákazníka a doručení. Dopravy se však evidují v systému tak, že je pro každou platbu a „způsob dopravy“ jedna doprava v databázi.



Obrázek 4.1 Stránka „Objednávky“

4.1.2 Řešení úkolů

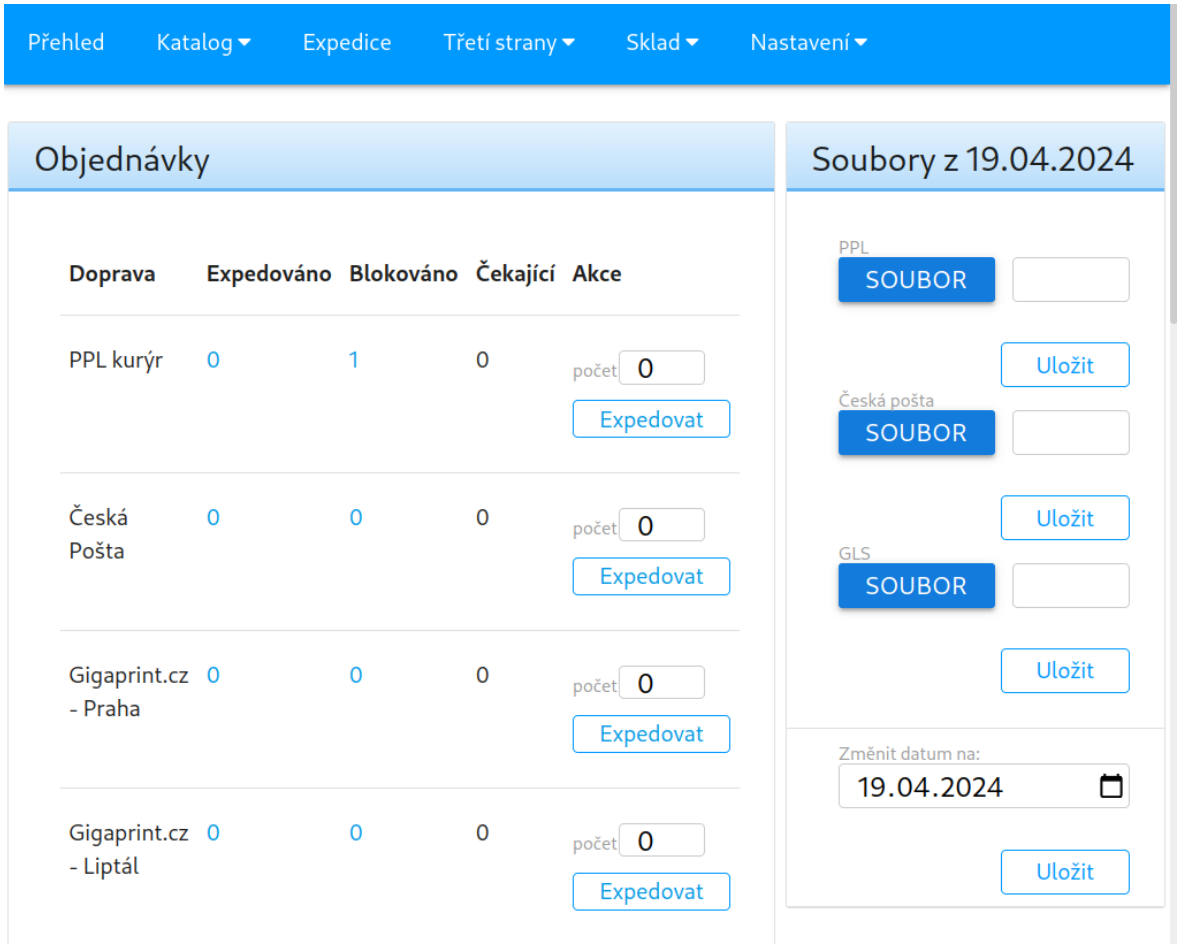
Pro zaměstnance je nutné, aby řešili úkoly, které blokují objednávky, druhy expedicí nebo produkty. Níže je příklad úkolu, který se vyskytne, když je více naskenovaných produktů než u objednávek, které jsou zamknuty zámkem pro originální produkty.



Obrázek 4.2 Úkol „Uvolnění naskenovaných produktů do expedice“

4.1.3 Expedice

U expedice se expedují objednávky, které však mohou být blokovány. V případě této práce je jeden z blokovacích důvodů ten, že se nenaskenovaly všechny produkty.



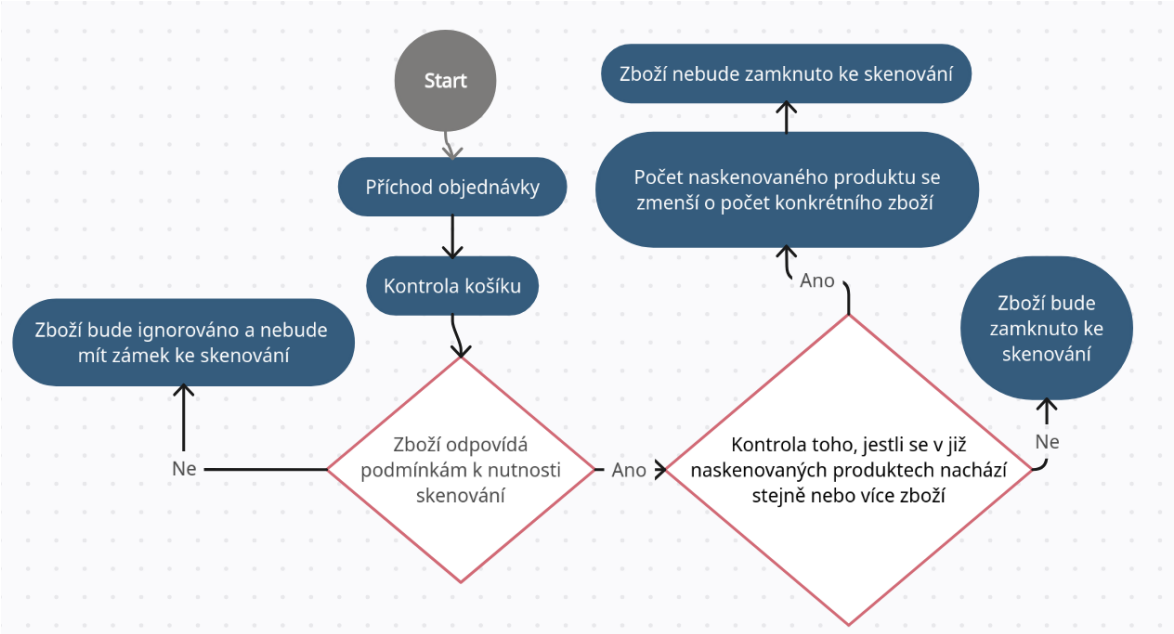
Obrázek 4.3 Stránka „Expedice“

4.2 Backend

CRM systém zadavatele je postaven pomocí PHP frameworku Laravel 6.

4.2.1 Zámky

Zámky jsou funkcionalita, která slouží v případě expedice k blokování objednávek, a obecněji slouží jako úkol pro zaměstnance. Příkladem může být zámek, že zbývá málo volných čísel v číselné řadě čísel balíků v PPL, nebo jak bylo zmíněno v podkapitole Expedice, tak v rámci této práce může být zámek úkolem pro naskenování všech produktů v přijmovém listu. Níže je aktivitní diagram, který ukazuje proces vytváření zámku pro skenování.



Obrázek 4.4 Aktivitní diagram vytváření zámku pro skenování

4.2.2 Objednaný předmět v košíku (neboli OrderedBasketItem)

OrderedBasketItem značí předmět v objednávce. Každý OrderedBasketItem je přiřazený ke konkrétnímu daňovému dokladu, a příkladem takového předmětu je běžně produkt, ale může to být i dárek v podobě čokolády, pokud zákazník nakoupí nad určitou cenovou hranici.

4.2.3 Produkt

Produkt je konkrétní předmět, který nabízí na stránkách k prodeji. Nepočítá se do toho předem zmíněná čokoláda. Produkt má mnoho kategorií, například to, jestli je produkt

(v tomto případě toner) kompatibilní s tiskárnou, nebo jestli je přímo originálně vytvořený pro tiskárnu, nebo to, jakou barvu má toner.

4.2.4 Produkt třetí strany (neboli ThirdProduct)

Produkt třetí strany v podstatě slouží jako vazba na produkt, kde je vazba s konkrétním dodavatelem (proto produkt třetí strany) a produktem.

4.2.5 Daňový doklad (neboli TaxDocument)

Daňový doklad je doklad, který má k sobě přiřazené předměty v košíku, a obsahuje, kdo je příjemcem a dodavatelem v rámci daňového dokladu.

4.2.6 Skenovaný produkt (neboli ScannedProduct)

Skenovaný produkt představuje záznam o produktu, který slouží jako jednoduchá mezipaměť udržující aktuální množství volných produktů na skladě. Jak je ukázáno funkcionalitou zámku a příkladem skenování, tento záznam umožňuje automatické odemykání zámku pro skenování odebráním počtu již zaznamenaných produktů.

5 NÁVRH INTEGRACE MOBILNÍ APLIKACE S CRM SYSTÉMEM

Implementace vychází z funkcionálních požadavků, které budou v následující podkapitole.

5.1 Funkcionální požadavky zadavatele

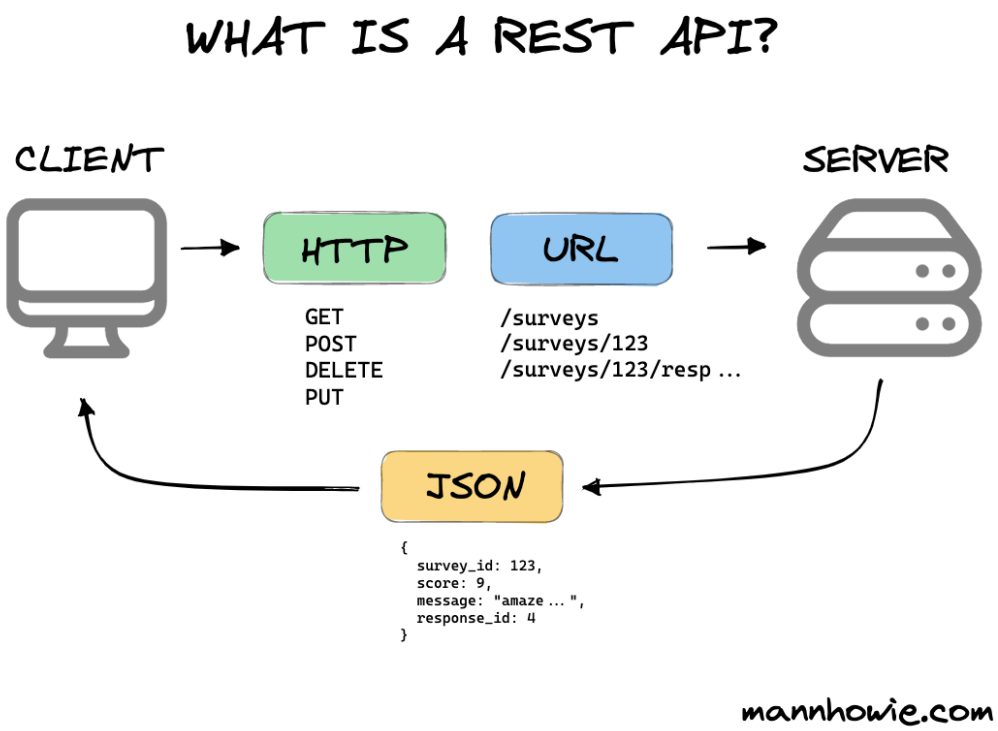
Funkcionální požadavky zadavatele jsou shrnuty v Tabulce 5.1.

Tabulka 5.1 Funkcionální požadavky k CRM systému

ID	Požadavek
FUN01	Webová aplikace se rozšíří o RESTFUL API, která bude interagovat s mobilní aplikací.
FUN02	Na serverové straně bude přidán nový typ úkolu pro zaměstnance: skenování konkrétního dokladu. Pokud je v objednávce produkt, který vyžaduje skenování, zaměstnanec bude požádán o skenování pomocí skeneru. Po naskenování zboží databáze zaznamená, kdo provedl úkol, a to podle přiděleného uživatelského ID.
FUN03	Po naskenování produktů se bude odemykat zámek u produktů, pokud je dostatek skladových položek pro zamknuté produkty. Bude taky vytvořen nový typ úkolu pro zaměstnance, pro případ, že není dostatek skladových položek, kde si zaměstanec manuálně povolí zámek u prioritních objednávek.
FUN04	V mobilní aplikaci bude schopen zaměstnanec vytvořit výdejový doklad k dodavateli ze zboží. Tato možnost se uplatní v situaci, kdy zákazník vrátí zboží, které původně zakoupil od dodavatele. Výběrem dodavatele v aplikaci bude moct zaměstanec vytvořit výdejový doklad, kam naskenuje produkty vrácené od zákazníka. Systém automaticky najde příjmový doklad obsahující právě ten produkt, který byl naskenován. V interním komentáři v objektu OrderedBasketItem bude specifikováno, z jakého příjmového dokladu byl produkt původně zakoupen. Pro výběr produktů z příjmových dokladů bude použit algoritmus LIFO.

5.2 Funkcionální požadavek č. 1 - Implementace REST API

Funkcionalita prvního požadavku je znázorněna v následujícím obrázku.



Obrázek 5.1 Diagram „REST API“ [35]

Do CRM systému bylo potřeba pro základní funkci skenování přidat koncové body: getStockNotes, getStockNoteProducts, getStockNoteProductEans asendScannedProgress, které jsou implementovány ve třídě StockNoteController. Princip aplikace je pak takový, že si uživatel vybere ze příjemek na skenování díky getStockNotes, získá seznam produktů ke skenování pomocí getStockNoteProducts, obnovení EANů produktu pomocí getStockNoteProductEans a odeslání naskenovaných produktů za pomocí sendScannedProgress (která je však součástí funkcionálního požadavku č. 4).

Aplikace také obsahuje možnost skenovat produkty k vrácení k dodavateli. Obsahuje funkce getStockNotes a getStockNoteProducts, které mají podobnou funkcionalitu jako u StockNoteController, a dále jsou implementovány funkce createWarehouseReleaseNoteWithSupplier, která vytváří výdejový list s konkrétním dodavatelem jako příjemce, a scanItemFromSupplier, která implementuje LIFO algoritmus, který vybírá objednané zboží z aktivních přijimových dokladů. Funkce createWarehouseReleaseNoteWithSupplier a scanItemFromSupplier jsou však i součástí funkcionálníího požadavku č. 5.

5.2.1 Koncové body StockNoteController

Tato podkapitola pojednává o koncových bodech StockNoteController.

Koncový bod „getStockNotes“ Pomocí tohoto koncového bodu lze získat seznam příjemek či výdejek. Lze filtrovat výsledky s výběrem dodavatele, časového období, ID v systému (jsou různé řady viditelného ID pro různé doklady, takže příjemka může mít stejné „ID“ jako jiná příjemka, ale v rámci databáze se evidují s jiným neviditelným ID), a výběrem vyloučení interním ID, který slouží pro zpřehlednění výběru při skenování více příjemek.

Tato funkce je taky volána v případě, že se ověřují výdejky či skenuje vrácené zboží. Níže je příklad JSON dat, které vrací koncový bod.


```
{
        "current_page": 1,
        "data": [
        {
                "id": 1,
                "type": "P\u0159\u00edjemka",
                "fulfillment": "2023-08-02",
                "supplier": "Dodavatel"
        },
        {
                "id": 2,
                "type": "P\u0159\u00edjemka",
                "fulfillment": "2023-08-02",
                "supplier": "Dodavatel"
        }
        ],
        "first_page_url": "https:\/\/domena.com\/stock-notes?page=1",
        "from": 1,
        "last_page": 1,
        "last_page_url": "https:\/\/domena.com\/stock-notes?page=1",
        "next_page_url": null,
        "path": "https:\/\/domena.com\/stock-notes",
        "per_page": 15,
        "prev_page_url": null,
        "to": 2,
        "total": 2
}
```


Koncový bod „getStockNoteProducts“ Tato funkce přijímá pole čísel nebo číslo a vrací seznam produktů v daňových dokladech.


```
        {
                "items": [
                {
                        "id": 1,
                        "tax_document_id": 1,
                        "price": "1.00",
                        "amount": 10,
                        "product_id": 1,
                        "name": "Produkt 1",
                        "eans": [
                                "1234567890123"
                        ],
                        "scannedAmount": 0,
                        "isSolved": 0
                },
                {
                        "id": 2,
                        "tax_document_id": 1,
                        "price": "2.00",
                        "amount": 10,
                        "product_id": 2,
                        "name": "Produkt 2",
                        "eans": [
```

```
                        "1234567890124"
                    ],
                    "scannedAmount": 0,
                    "isSolved": 0
                }
                ]
        }
```


Koncový bod „getStockNoteProductEans“ Tato funkce přijímá ID produktů a vrátí aktuální EANy produktů. Tahle funkcionalita je zapotřebí, protože aplikace pouze přijímá seznam EANů produktů, které jsou v daňovém dokladu, ve chvíli, kdy si uživatel otevře daňový doklad. Jinými slovy, seznam EANů produktů se nesynchronizuje v reálném čase. Zde je příklad zdrojového kódu, který vrací JSON odpověď:

```
private function _getStockNoteProductEans(array $itemIds): array {
    $items = \App\Product::withoutGlobalScopes()->whereIn('id',
↪   $itemIds)->with('eans')->get();
    $template = [];
    $items->each(function ($n, $key) use (&$template) {
        $newArray = [];
        $newArray['product_id'] = $n->id;
        $newArray['eans'] = $n->eans->pluck("ean")->toArray();
        $template['items'][$key] = $newArray;
    });
    return $template;
}
```

5.2.2 Koncové body WarehouseReleaseController

Tato podkapitola pojednává o koncových bodech WarehouseReleaseController.

Koncový bod „getStockNotes“ Tento koncový bod vrací skladové výdejky, které čekají na expedici. Kromě toho jsou si s koncovým bodem stejného názvu ve StockNoteController téměř totožné.


```
{
        "current_page": 1,
        "data": [
        {
                "id": 1,
                "type": "V\u00fddejka",
                "fulfillment": "2023-08-02",
                "supplier": "Dodavatel"
        },
        {
                "id": 2,
                "type": "V\u00fddejka",
                "fulfillment": "2023-08-02",
                "supplier": "Dodavatel"
        }
        ],
        "first_page_url": "https:\/\/domena.com\/warehouse-release?page=1",
        "from": 1,
        "last_page": 1,
        "last_page_url": "https:\/\/domena.com\/warehouse-release?page=1",
        "next_page_url": null,
        "path": "https:\/\/domena.com\/warehouse-release",
        "per_page": 15,
        "prev_page_url": null,
        "to": 2,
        "total": 2
}
```

Koncový bod „getStockNoteProducts“ Tento koncový bod vrací seznam produktů v jedné skladové výdejce.


```
{
        "items": [
        {
                "id": 3,
                "amount": 1,
                "product_id": 1,
                "name": "Produkt 1",
                "eans": [
                "1234567890123"
                ]
        }
        ]
}
```


5.3 Funkcionální požadavek č. 2 - Vytvoření úkolu skenování pro zaměstnance

Tento funkcionální požadavek byl implementován pomocí událostí v Laravel 6. Když se přidá OrderedBasketItem do objednávky, tak se zkontroluje, jestli vyhovuje zadaným požadavkům ve funkcionálním požadavku FUN02.

```
class AddScannerLock
{
        public function handle(Event $event): void
        {
                $basketItem = $event->getBasketItem();
                if ($this->isHandleable($basketItem)) {
                        $basketItem->lock(Scanner::class);
                }
        }
        private function isHandleable(OrderedBasketItem $basketItem): bool
        {
                return ($basketItem->object_type == ThirdProduct::class ?
    $basketItem->_3rdProduct->locks->where('route', WaitForPair::class)->count() ==
↪   0 : true)
                && ($basketItem->taxDocument instanceof StockReceiptNote)
                && ($basketItem->taxDocument->stock_id == \App\Branch::LIPTAL_ID)
                && (
                $basketItem->object instanceof Product
                && $basketItem->object->getIsScannerableProductAttribute()
                || $basketItem->object instanceof ThirdProduct
                && $basketItem->object->product instanceof Product
                && $basketItem->object->product->getIsScannerableProductAttribute()
                );
        }
}
```

Bylo taky potřeba přidat nový model: ScannedProduct, jenž obsahuje údaje ID daňového dokladu, ID produktu a počet naskenovaných kusů.

Naskenování řídí koncový bod sendScannedProgress, který ukládá rozskenované daňové doklady a který je součástí funkcionálního požadavku č. 4. U úkolu se pomocí metody „setSolved“ připáruje uživatel.

5.4 Funkcionální požadavek č. 3 - Odemykání zámků u produktů se skenovaným produktem

Implementace tohoto požadavku vyžaduje nejdřív dříve zmíněný koncový bod sendScannedProgress.

Při novém naskenování produktu se vytvoří nový záznam pro model ScannedProduct. Tento záznam spustí událost v Laravelu, která vytvoří úkol na řešení produktů, které jsou blokovány, protože splňují podmínky pro úkol ke skenování.

5.5 Funkcionální požadavek č. 4 - Skenování výdejových dokladů

Na tento požadavek bylo nutno implementovat 2 koncové body, createWarehouseReleaseNoteWithSupplier a scanItemFromSupplier.

Bylo také nutno vytvořit nový model: ScannedReleaseItem, který zaznamenává, kolik položek z aktivního dokladu bylo naskenováno pro vrácení dodavateli.

6 NÁVRH A IMPLEMENTACE MOBILNÍ APLIKACE

Mobilní aplikace byla implementována pomocí frameworku Ionic ve webovém frameworku Vue. Pro implementaci byly použity především knihovny použity knihovny Pinia, která pomáhá se správou stavu ve Vue a Axios, který slouží k HTTP požadavkům.

6.1 Funkcionální a nefunkcionální požadavky

Tabulka 6.1 Funkcionální požadavky mobilní aplikace

ID	Požadavek
FUN01	Mobilní aplikace umožní přihlásit se přes údaje, přes které se přihlašuje na webovou aplikaci.
FUN02	Uživatel bude moct skenovat za pomocí zařízení Zebra.
FUN03	Před skenováním si uživatel bude moct vybrat skupinu příjmových dokladů. Často totiž dodavatel posílá více účetních dokladů v jedné dodávce, a tak uživatel nebude muset ručně rozlišovat jednotlivé doklady.
FUN04	V mobilní aplikaci bude možné aktualizovat EANy produktů. Každý produkt může mít více různých EAN kódů, ale každý z nich musí být jedinečný pro daný produkt.
FUN05	Uživatel musí během skenování dostat notifikaci, že skenuje produkt, který není ve zvoleném příjmovém dokladu.
FUN06	Uživateli bude umožněno skenovat více kusů najednou tím, že před začátkem skenování určí, kolik položek chce skenovat v jedné dávce.
FUN07	Během skenování musí být barevně odlišeno naskenované zboží a zboží, které chybí.
FUN08	Aplikace musí mít pouze světlý režim.
FUN09	Mobilní aplikace umožní skenování i pomocí fotoaparátu.

Tabulka 6.2 Nefunkcionální požadavky mobilní aplikace

ID	Požadavek
NEF01	Bezpečnost: Všechny požadavky by měly být zabezpečeny pomocí ověření pomocí Bearer tokenu a data by měla být přenášena šifrovaně pomocí protokolu HTTPS.
NEF02	Rozšiřitelnost: Softwarová architektura bude navržena pro další škálovatelnost, aby mohla být v budoucnu jednoduše rozšířena o nové funkcionality.
NEF03	Bude navrhnuto intuitivní grafické rozhraní mobilní aplikace.

6.2 Store

Tato kapitola pojedná o tzv. „storech“, což je objekt, co ukládá stav aplikace.

6.2.1 TokenStore

TokenStore se zabývá přihlašovacími údaji a je nutný k tomu, aby mohla aplikace snadněji vytvářet requesty přes Axios. Přes interceptor v Axios lze nastavit, že bude použit token z TokenStore.

```
axios.interceptors.request.use(
(config) => {
        const token = tokenStore.token;
        if (token) {
                config.headers['Authorization'] = `Bearer ${token}`;
        }
        return config;
},
(error) => {
        return Promise.reject(error);
}
);
```

TokenStore bývá používán rovněž pro pro ukládání pobočky ve výběru přidávání fotek ke kase, která však není popsána v této bakalářské práci.

6.2.2 BasketItemStore

Tento store se používá pro ukládání stavu košíku v rozskenovaném daňovém dokladu. BasketItemStore hlídá samotné produkty, ID daňových dokladů, jestli daňový doklad nějak pokročil ve skenování (changed), jestli se ověřuje výdejka a to, jestli se skenuje vrácené zboží. Příkladem použití je například při obsluze kliku při výběru daňového dokladu k tomu, aby se přidalo ID daňového dokladu.

```
const onStockNoteSelect = (id: number) => {
        basketItemStore.addStockItemId(id);
        if(props.returnItem) basketItemStore.setContactInfo(data.items.find(item =>
↪  item.id == id)?.contact_info || "");
        basketItemStore.isScanningMultiple ? router.back() :
↪  router.push("/receipt/scanning");
}
```

6.2.3 NotificationStore

Tento store slouží pro ukládání načítací obrazovky a vyskakovacích oken. Příkladem využití je například u HTTP požadavků.

```
const loadStockNotes = ($event: InfiniteScrollCustomEvent | null) => {
        if ($event) {
                data.page++;
        } else {
                data.page = 1;
                notificationStore.showLoadScreen();
        }
        let callArray = (data.suppliers instanceof Object &&
    Object.keys(data.suppliers).length) ? [stockNotesCall(data.page)] :
⇆   [stockNotesCall(data.page), axios.get("/api/pub/suppliers")];
        Promise.all(callArray)
        .then((responses: AxiosResponse[]) => {
                data.items = data.items.concat(responses[0].data.data as
↪  ReceiptStockNoteResponse[]) as ReceiptStockNoteResponse[];
                if (responses[0].data.data && responses[0].data.data.length == 0) {
                        data.noMoreItems = true;
                }
```

```
                if (callArray.length == 2) {
                        data.suppliers = responses[1].data as SupplierResponse;
                }
                if ($event) {
                        ($event.target as HTMLIonInfiniteScrollElement).complete();
                } else {
                        notificationStore.hideLoadScreen();
                }
        });
}
```

6.3 Komunikace s REST API

Komunikace s REST API je umožněna pomocí knihovny Axios. Axios disponuje možností „baseUrl“, díky které není potřeba u HTTP požadavku opisovat URL webové stránky. V kódu je zápis následný:

```
axios.defaults.baseURL = 'https://domena.com/';
```

Příklad může být kód v následujícím řádku.

```
const loadProducts = () => {
        if (basketItemStore.changed && !basketItemStore.isScanningMultiple) return;
        notificationStore.showLoadScreen();
        axios
        .get("/api/pub/warehouse-release/" + basketItemStore.stockItemIds +
↪   "/products")
        .then((response) => {
                // handle success
                if (response.data.message) {
                        data.message = response.data.message;
                } else {
                        data.message = false;
                        basketItemStore.setBasketItems(response.data.items);
                }
                notificationStore.hideLoadScreen();
                console.log(response);
        });
}
```

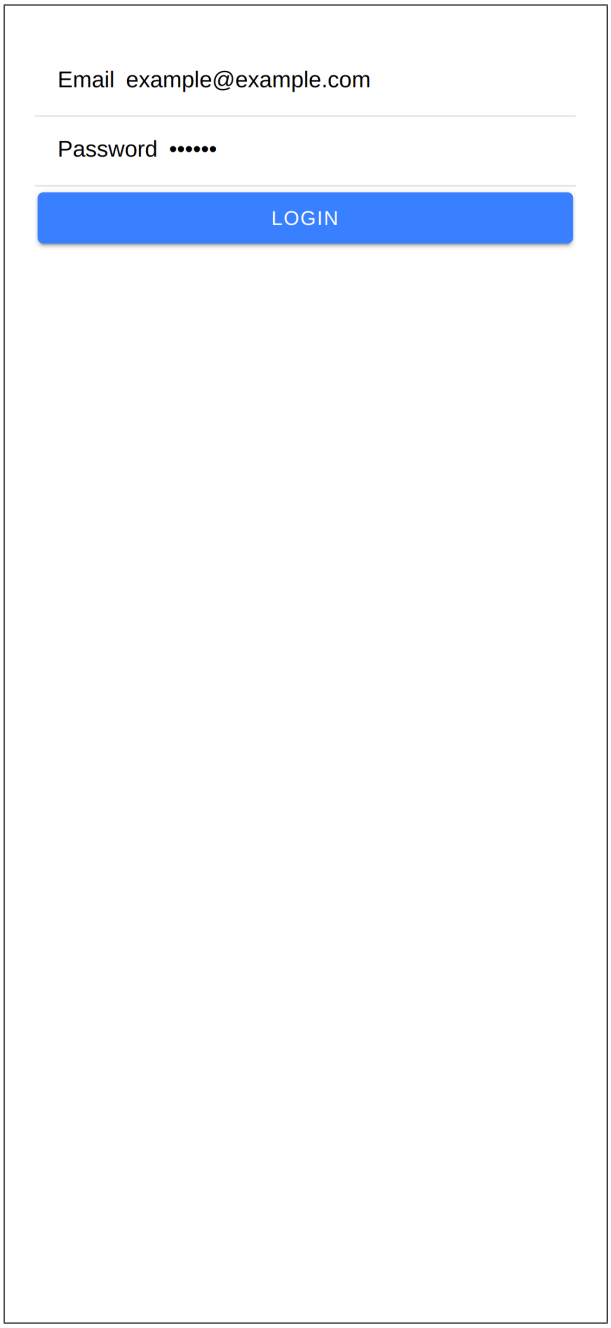
6.4 Uživatelské rozhraní

Tato část pojednává o popisu rozhraní.

6.4.1 Přihlášení

Tato část rozhraní se jako první zobrazí uživateli, pokud není přihlášený. Rozhraní je celkem jednoznačné, vyplňují se zde údaje na přihlášení, a to email a heslo. Po stisknutí tlačítka „Login“ se zobrazí bublina, která slouží jako zpětná vazba k uživateli, že se přihlašování provádí. Pokud bude přihlášení neúspěsné, tak vyskočí uživateli bublina, že přihlášení bylo neúspěšné a uživatel se může pokusit znovu přihlásit. Pokud se přihlášení podaří, tak se uloží token od serveru k pozdějšímu přihlášení, a bude také dostupné napříč aplikací tlačítko „Odhlásit se“. Zde je příklad rozhraní a obsluhy událostí ke kliku tlačítka, který je použit v této obrazovce:

```
<template>
<ion-page v-if="!tokenStore.token">
<ion-toolbar>
<ion-title>Header</ion-title>
</ion-toolbar>
<ion-content class="ion-padding">
<ion-list>
<ion-item>
<ion-label>Email</ion-label>
<ion-input type="text" v-model="data.email"></ion-input>
</ion-item>
<ion-item>
<ion-label>Password</ion-label>
<ion-input type="password" v-model="data.password"></ion-input>
</ion-item>
<ion-button @click="login" expand="block">Login</ion-button>
</ion-list>
</ion-content>
</ion-page>
<template>
...
<script setup>
const login = () => {
        notificationStore.showLoadScreen();
        axios.post('/api/pub/login', {
                email: data.email,
                password: data.password
        }).then(function (response: AxiosResponse) {
                console.log(response);
                notificationStore.hideLoadScreen();
                if (response.data.error) {
                        presentToast('top', response.data.error);
                        return;
                }
                tokenStore.setToken(response.data.token)
        }.bind(this));
}
</script>
```



Obrázek 6.1 Obrazovka „Přihlášení“

6.4.2 Výběr režimu činnosti

Tato část rozhraní se objeví po úspěšném přihlášení. Pokud se však uživatel přihlásil v minulém sezení (session) aplikace, tak se objeví tato část rozhraní jako první. Jsou v ní zobrazeny tlačítka pro výběr činnosti, kterou chce uživatel provádět, a to sice:

- skenovat příjemky, kde se ověřuje, že došly produkty z přijatého zboží v pořádku,
- skenovat vrácené zboží, kde se skenuje vrácené zboží od zákazníků,
- ověřovat výdejky, kde se ověřuje, že se produkty k expedici zabalily v pořádku,
- skenovat výdejky, kde se pomocí LIFO algoritmu vybírají minule naskenované produkty k vrácení k dodavateli
- přidávat fotky ke kase, což je funkcionalita mimo tuto práci

Zde je příklad z části rozhraní výběru režimu činnosti:

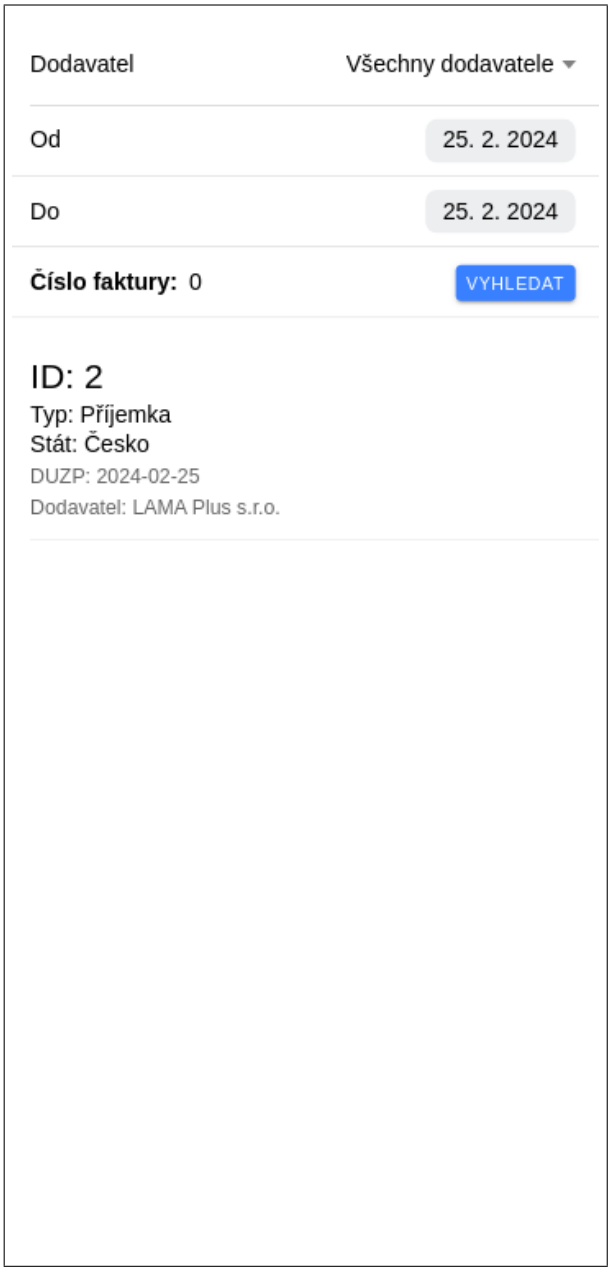
```
<ion-page v-else>
<ion-content class="ion-padding">
<div
style="display:flex; flex-direction: column; height: 100%;  margin: auto; flex: 1;
↪  padding: 2%; margin-top: 10%; align-items: center;">
<span>
<h5><b>Vyberte si, co chcete provádět</b></h5>
</span>
<router-link to="/receipt/stock-notes" v-slot="{ href, route, navigate }"
↪  class="selection" id="receiptScan">
<ion-button @click="navigate">
Skenovat příjemky
</ion-button>
</router-link>
<router-link to="/receipt/stock-notes-return" v-slot="{ href, route, navigate }"
↪  class="selection" id="receiptReturnScan">
<ion-button @click="navigate">
Skenovat vrácené zboží
</ion-button>
</router-link>
</div>
</ion-content>
</ion-page>
```

Obrázek 6.2 Obrazovka „Výběr činnosti“

6.4.3 Výběr daňových dokladů mimo skenování výdejek

Tato část rozhraní se objeví poté, co si uživatel vybere možnost skenovat příjemky, skenovat vrácené zboží nebo ověřovat výdejky. Co se týče Listu na zobrazení filtrovacích možností, tak jelikož je více dodavatelů, tak je v prvé řadě výběr dodavatelů, který zobrazí vyskakovací bublinu, kde je výběr dodavatelů. Dodavatel se však může vybrat pouze jeden. Poté jsou k filtraci možné vybrat data, od a do kdy má být v rozsahu datum uskutečnění zdanitelného plnění. Jako poslední možnost je filtrace pomocí interního čísla faktury, která se dá aplikovat pomocí tlačítka „Vyhledat“. Je však nutno dodat, že i přes zadání interní čísla faktury je možné, aby se zobrazilo více výsledků, protože jsou různá interní čísla například pro faktury a výdejky.

Do této části rozhraní se taky přistoupí poté, když uživatel, který skenuje příjemku, si přeje přidat další příjemku na skenování.



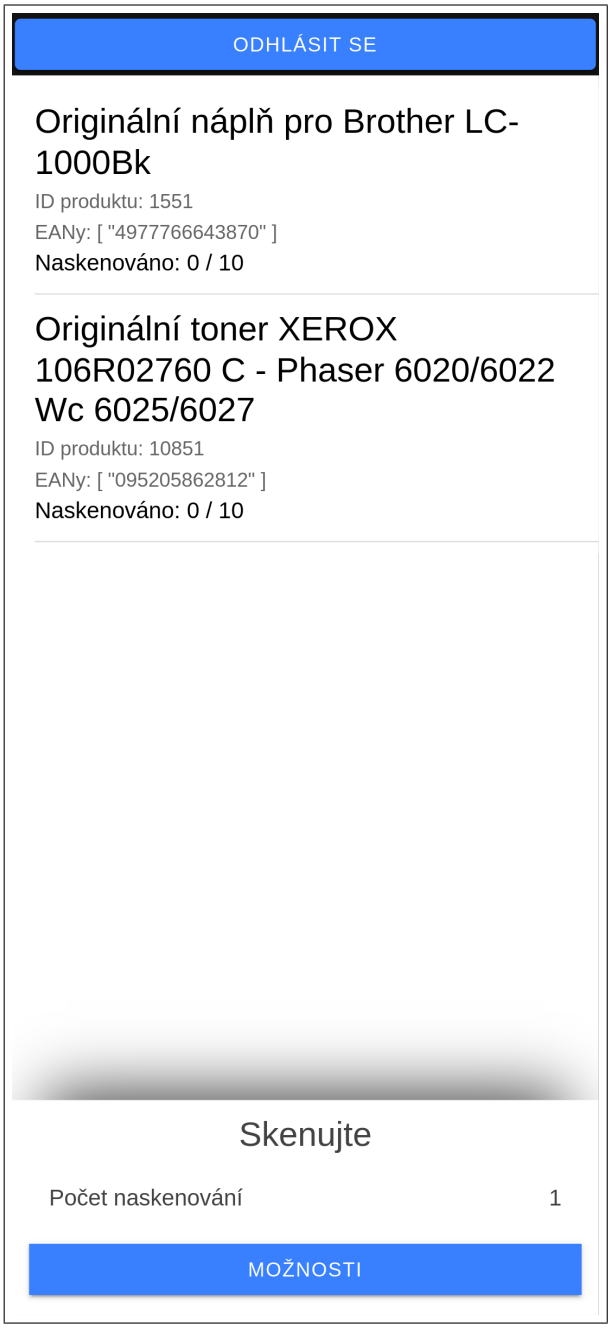
Obrázek 6.3 Obrazovka „Výběr příjemky“

6.4.4 Výběr výdejek na skenování

Tato část rozhraní se objeví poté, co si uživatel vybere možnost skenovat výdejky. Tato část rozhraní se moc neliší od „výběru příjemek“, ale rozdíly jsou takové, že není možné vybrat dodavatele a že je možnost vytvořit z tohoto menu výdejky na skenování.

6.4.5 Skenování příjemek (využívající API od StockNoteController)

Tato část rozhraní se objeví po zvolení možností skenovat příjemky, skenovat vrácené zboží nebo ověřovat výdejky, a vybere si konkrétní daňový doklad. V rozhraní se zobrazí List, ve kterém jsou Itemy, které odpovídají produktům v daňovém dokladu. Pozadí Itemů je zabarvené podle toho, jestli je produkt doskenovaný již z předešelého skenování (šedá), doskenovaný ze současného skenování (zelená), rozskenovaný (oranžová) nebo vůbec neskenovaný (bílá). V patičce se nachází stav skenování, popřípadě kontaktní informace, pokud se skenuje vrácené zboží, vstup počtu kusů při jednom naskenování a tlačítko pro možnosti. V možnostech se nachází možnost „Odeslat“, „Aktualizovat EANy“, „Přidat další příjemku“ nebo „Skenovat přes kameru“. Při odeslání se odešle přes API postup skenování přes sendScannedProgress. Při aktualizaci EANů se zavolá koncový bod getStockNoteProductEans. Při přidání další příjemky se skočí opět na menu výběru příjemek, kde se však již nezobrazí současně skenované příjemky. Při skenování kamerou se použije Ionic komunitní knihovna „BarcodeScanner“, díky které lze pomocí fotoaparátu v telefonu skenovat čárové kódy.



Obrázek 6.4 Obrazovka „Skenování“

6.4.6 Skenování výdejek (využívající API od WarehouseReleaseController)

Tato část rozhraní se objeví po zvolení možností skenovat výdejky a výběru konkrétního daňového dokladu. Stejně jako u skenování příjemek, také v tomto rozhraní se zobrazí List obsahující Itemy odpovídající produktům v daňovém dokladu. Pozadí Itemů však zabarvené není, protože se neověřuje, že přišly produkty v pořádku, ale skenují se jako v obchodě, kde díky skenování se přidávají produkty do výdejky. V patičce se nachází stav skenování, který se změní podle chyby, takže v případě, kdy není produkt dostupný s konkrétním EANem, nebo když není dostupný produkt z LIFO algoritmu na výběr.

6.5 Implementace mobilní aplikace

Tato podkapitola pojednává o implementaci funkcionálních a nefunkcionálních požadavků do mobilní aplikace.

6.5.1 Funkcionální požadavek ID FUN01 - Přihlášení do aplikace

Bylo potřeba přidat API na přihlášení, a bylo potřeba přidat Laravel Passport na server.

```
Route::post('/login', function (Request $request) {
        $json = $request->json()->all();
        $user = \App\User::where('email', $json['email'])->first();
        if ($user == null) {
                return response()->json([
                "error" => "Unsuccessful login",
                ]);
        }
        if (Auth::attempt(['email' => $json['email'], 'password' =>
↪  $json['password']])) {
                $token = $user->createToken('Token Name')->accessToken;
                return response()->json([
                "token" => $token,
                ]);
        } else {
                return response()->json([
                "error" => "Unsuccessful login",
                ]);
        }
});
```

Následně bylo pomocí knihovny Axios implementováno přihlášení na mobilní aplikaci.

```
const login = () => {
  notificationStore.showLoadScreen();
  axios.post('/api/pub/login', {
    email: data.email,
    password: data.password
  }).then(function (response: AxiosResponse) {
    console.log(response);
    notificationStore.hideLoadScreen();
    if (response.data.error) {
      presentToast('top', response.data.error);
      return;
    }
    tokenStore.setToken(response.data.token)
  }.bind(this));
}
```

6.5.2 Funkcionální požadavek ID FUN02 - Skenování pomocí zařízení Zebra

Nejdříve se předpokládalo, že bude skenování pouze dosaženo pomocí telefonů Zebra, které mají v sobě zabudované skenery. Přes ně se dá komunikovat přes DataWedge API, která však vyžaduje komunikaci s Android Intenty. Této komunikace je dosaženo pomocí rozhraní „com-darryncampbell-cordova-plugin-intent“, díky kterému lze odesílat a přijímat Android Intenty. S pomocí tohoto rozhraní bylo následně vytvořeno Vue Composable, které ještě volá metody „onIonViewWillEnter“ a „onIonViewWillLeave“, které zajistí, že skener bude pouze aktivní v konkrétních rozhraních.

```
        onIonViewWillLeave(() => {
                (<any>window).plugins.intentShim.unregisterBroadcastReceiver();
                sendCommand("com.symbol.datawedge.api.SOFT_SCAN_TRIGGER",
↪   "STOP_SCANNING");
        });
```

6.5.3 Funkcionální požadavek ID FUN03 - Skenování více daňových dokladů najednou

Požadavek byl vyřešen Pinia Storem "BasketItemStore", jenž byl přidán do StockNotesScreen. Ve StockNotesScreen jsou z výsledků při vyhledávání vyloučena konkrétní ID, u nichž již probíhá skenování.

```
...
const basketItemStore = useBasketItemStore(true);
const stockNotesCall = (page = 1) => {
        let type = "";
        if (props.verifying) {
                switch (data.type) {
                        case "faktura":
                        type = "App\\Models\\TaxDocuments\\Invoice";
                        break;
                        case "vydejka":
                        type =
    "App\\Models\\TaxDocuments\\WarehouseReleaseNote,App\\Models\\TaxDocuments\\
↪   StockReleaseNote,App\\Models\\TaxDocuments\\CollectionStockReleaseNote";
                        break;
                        default:
                        type = "";
                        break;
                }
        } else if (props.returnItem) {
                type = "App\\Models\\TaxDocuments\\StockReceiptNote";
        } else {
                type = "App\\Models\\TaxDocuments\\WarehouseReceiptNote";
        }
        return axios.get("/api/pub/stock-notes",
        {
                params: {
                        page: page,
                        supplier: data.currentSupplier == 0 ? null :
↪   data.currentSupplier,
                        from: data.from ? data.from : null,
                        to: data.to ? data.to : null,
                        exclude: basketItemStore.stockItemIds.length ?
↪   basketItemStore.stockItemIds.join(",") : null,
                        type: type ? type : null,
                        id: data.desiredId ? data.desiredId : null,
                        verifying: props.verifying ? 1 : null,
                }
        });
}
```

6.5.4 Funkcionální požadavek ID FUN04 - Možnost aktualizace EANů při skenování

Jelikož jsme již implementovali API, tak stačí implementovat chování u aplikace.

```
const refreshEans = () => {
        notificationStore.showLoadScreen();
        axios.post('/api/pub/stock-notes/getProductEans', {
                itemList: basketItemStore.basketItemsAsArray.map((item) =>
↪  item.product_id)
        }
        ).then(function (response) {
                basketItemStore.setBasketItemEans(response.data.items);
                // handle success
                notificationStore.hideLoadScreen();
        });
}
```

6.5.5 Funkcionální požadavek ID FUN05 - Notifikace nedostupného produktu v dokladu při skenování

Požadavek je nastaven ve funkci vyvolané úspěšným skenem z fotoaparátu nebo skeneru.

```
const onScan = async (scannedEan: string) => {
        let foundElements = basketItemStore.basketItemsAsArray.filter(element =>
↪  element.eans.some((ean: string) => ean === scannedEan));
        if (foundElements.length) { // filter vrací prázdné pole, pokud nenajde
↪  žádný prvek
                onSuccessScan(foundElements);
        } else {
                navigator.vibrate(200);
                data.scanningStatus = "Naskenovanému EAN neodpovídá žádný z produktů
↪  v této výdejce";
                data.isVerified = undefined;
                data.isVerified = false;
        }
        data.attempts++;
}
```

6.5.6 Funkcionální požadavek ID FUN06 - Skenování více kusů najednou

V prvním kroku pro vytvoření tohoto požadavku bylo do ProductScreen přidáno pole Ionic Input navázané na data v komponentě ProductScreen.

```
<ion-input class="ion-text-right ion-padding-end ion-padding-start" type="number"
↪  v-model_number="data.scanAmount" id="scanAmount" name="scanAmount" min="1"
↪  label="Počet naskenování">
</ion-input>
```

Ve druhém kroku byla přidána informace o počtu kusů na skenování, která je vrácena v případě, kdy je sken úspěšný.

```
const onSuccessScan = (foundElements: Product[]) => {
        let foundElement = foundElements.find(foundElement =>
↪  !(foundElement.isSolved || foundElement.scannedAmount >= foundElement.amount));
        if (foundElement === undefined) {
                foundElement = foundElements[0]; // všechny položky by měly mít
↪  stejný ean, takže vybereme první
                navigator.vibrate(200);
                data.scanningStatus = "Produkt '" + foundElement.name + "' je již
↪  doskenovaný";
                data.isVerified = undefined;
                data.isVerified = false;
                return;
        }
```

```
        data.scanningStatus = foundElement.name;
        data.isVerified = true;
        data.successfulScans += data.scanAmount;
        basketItemStore.changed = true;
        foundElement.scannedAmount += data.scanAmount;
        basketItemStore.items.set(foundElement.id, foundElement);
}
```

6.5.7 Funkcionální požadavek ID FUN07 - Barevné odlišení zboží při skenování dokladů

Díky Vue je možné do prvku template napsat logiku, kdy se barevně odliší zboží, aniž bychom použili skript.

```
<ion-item v-for="item in basketItemStore.sortedProducts" :key="item.id"
↪  :id="'basketItem-' + item.id"
        class="collection-item productItem"
        :class="[item.isSolved ? 'solved' : (item.scannedAmount ?
            (item.scannedAmount >= item.amount ? 'unsolved-to-solved' :
        ↪  'partially-solved') : '')]">
        <ion-label>
                <h1 class="productName ion-text-wrap">{{ item.name }}</h1>
                <p>ID produktu: <span class="productId">{{ item.product_id
                ↪  }}</span></p>
                <p v-if="basketItemStore.stockItemId === null">ID příjemky: <span
                ↪  class="productTaxDocumentId">{{
                                item.tax_document_id }}</span></p>
                <p>EANy: <span class="productEans">{{ item.eans }}</span></p>
                <h2 v-if="item.isSolved"><b style="font-size: 20px;">Naskenováno:
                ↪  ✓</b></h2>
                <h2 v-else>Naskenováno: <span class="scannedAmount">{{
                ↪  item.scannedAmount }}</span> / <span
                                class="productAmount">{{ item.amount }}</span></h2>
        </ion-label>
</ion-item>


...

ion-item.unsolved-to-solved {
        --background: green;
}
ion-item.partially-solved {
        --background: orange;
}
ion-item.solved {
        --background: grey;
}
```

6.5.8 Funkcionální požadavek ID FUN08 - Vypnutí temného režimu

U souboru „src/themes/variables.css“ stačilo pouze zakomentovat definice u „@media (prefers-color-scheme: dark)“.

6.5.9 Funkcionální požadavek ID FUN09 - Skenování za pomocí fotoaparátu

Tento funkcionální požadavek byl implementován pomocí komunitní Ionic knihovny "BarcodeScanner", která umožňuje číst z fotoaparátu čárové kódy. Zde je příklad kódu ke skenování, který byl implementován do Vue Composable určen ke skenování.

```
const startScan = async () => {
        await didUserGrantPermission();
        data.isScanningUsingCamera = true;
        document.querySelector('body')!.classList.add('scanner-active');
```

```
        BarcodeScanner.hideBackground();
        const result = await BarcodeScanner.startScan();
        data.isScanningUsingCamera = false;
        document.querySelector('body')!.classList.remove('scanner-active');
        if (result.hasContent) {
                scannerCallback(result.content);
        }
};
```

6.5.10 Nefunkcionální požadavek ID NEF01 - Bezpečnost

Bezpečnost je zajištěna pomocí HTTPS pomocí baseUrl v Axios. V Axios interceptoru je dále nastavený Bearer token.

6.5.11 Nefunkcionální požadavek ID NEF02 - Rozšířitelnost

Rozšířitelnost je umožněna díky frameworku Vue. Pokud by někdo chtěl přidat další obrazovku, tak si stačí vytvořit Vue SFC a přidat jej do routeru v adresáři „src/router/index.ts“. Následující řádek ukazuje příklad pomocí „nova-obrazovka“:

```
import LoginScreen from '@/views/LoginScreen.vue';
import NovaObrazovka from '@/views/NovaObrazovka.vue';
...
const routes: Array<RouteRecordRaw> = [
{ path: '/', name: "Login", component: LoginScreen },
{ path: '/nova-obrazovka', name: "NovaObrazovka", component: NovaObrazovka },
...
];
```

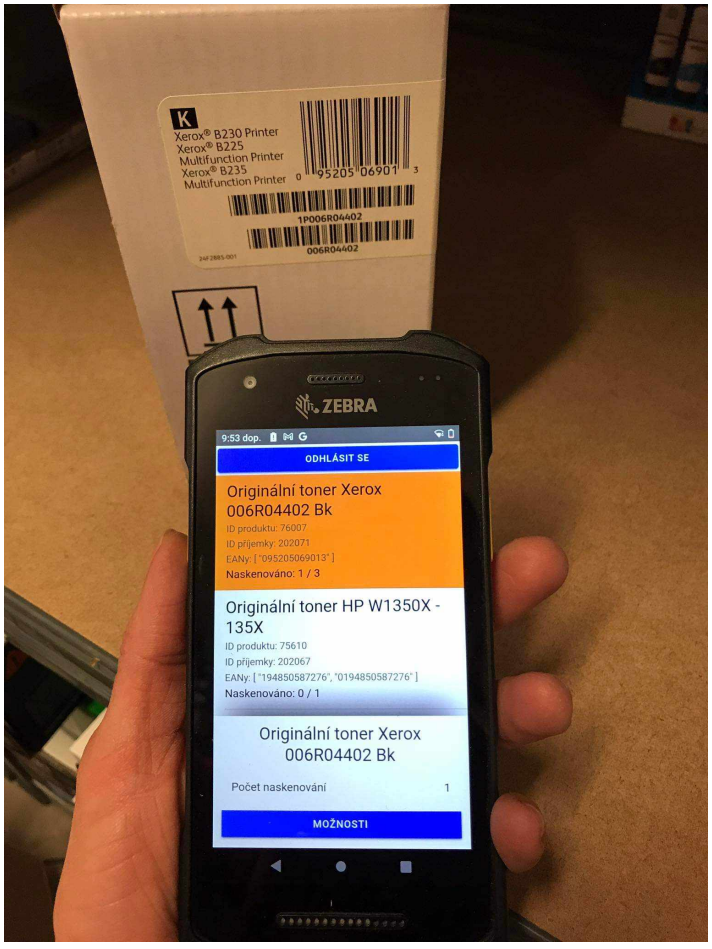
6.5.12 Nefunkcionální požadavek ID NEF03 - Rozhraní aplikace

Intuivní rozhraní aplikace bylo implementováno pomocí komponent z frameworku Ionic.

7 SHRNUTÍ

V rámci bakalářské práce byly pokryty všechny požadavky stanovené zadavatelem. Aplikaci lze použít nejen na mobilních zařízeních Zebra, ale i na většině zařízeních s aktuální verzí operačního systému Android, jelikož byla implementována i funkce skenování z fotoaparátu. Zadavatel je s výslednou aplikací spokojen, jak potvrzuje jedno vyjádření zaměstnance:

„Student prokázal významný přínos v oblasti pracovních procesů a softwarového vývoje v naší firmě od roku 2022 až do současnosti. Student dostával postupně UX zadání pro mobilní aplikaci, kterou rozšiřoval podle požadavků z naší strany. Aplikaci navíc rozšířil o možnost skenování 1D/2D kódů pomocí kamery, díky čemuž ji mohli používat i zaměstnanci na firemních telefonech bez snímače. Aktuální verze software, kterou student vytvořil, ušetří naší společnosti 2-3 hodiny lidské práce denně. Aktuálně s námi student spolupracuje na dalších rozšířeních, které nejsou součástí jeho bakalářské práce, a které povedou k ještě významnější časové úspoře a zlepšení přesnosti dat.“



Obrázek 7.1 Použití aplikace v praxi

ZÁVĚR

Cílem této práce bylo implementovat řešení pro zefektivnění kontroly dostupnosti produktů za pomocí skenování přes mobilní aplikaci. Byly použity frameworky Ionic, který slouží pro vývoj multiplatformních aplikací a Vue, který poskytuje strukturovaný přístup k tvorbě uživatelského rozhraní. Využitím těchto frameworků byl dosažen rychlý a efektivní vývoj multiplatformní aplikace. Díky řešení navrhnuté bakalářskou prací je proces objednání produktů ze skladu rychlejší a vede k efektivnějšímu skladovému hospodářství.

V teoretické části práce se čtenář seznámil s frameworky Ionic a Vue a s teorií skladových systémů. V praktické části práce byla popsána integrace mobilní aplikace s CRM systémem zadavatele, požadovaná funkcionalita aplikace s funkcionálními a nefunkcionálními požadavky, způsob komunikace s REST API, uživatelské rozhraní a samotná implementace aplikace.

Samotná implementace aplikace byla prezentována prostřednictvím praktických kódových ukázek, které ilustrovaly využití frameworků Ionic a Vue při vytváření mobilních aplikací.

Byla nakonec implementována aplikace podle požadavků zadavatele, který je s ní spokojen. Ukázalo se, že lze použít nejen mobilní zařízení Zebra, ale i běžný telefon s operačním systémem Android.

SEZNAM POUŽITÉ LITERATURY

[1] Ionic. *Cross Platform | Ionic Documentation*. Online. 2023. Dostupné z: https://ionicframework.com/docs/core-concepts/cross-platform. [cit. 2024-04-14].

[2] Ionic. *Stencil*. Online. 2024. Dostupné z: https://stenciljs.com/. [cit. 2024-04-14].

[3] Ionic. *ion-content: Scrollable Component for Ionic App Content*. Online. 2023. Dostupné z: https://ionicframework.com/docs/api/content. [cit. 2024-04-14].

[4] Ionic. *Vue Navigation: Use Ionic + Vue Router to Create Multi-Page Apps*. Online. 2023. Dostupné z: https://ionicframework.com/docs/vue/navigation#ionpage. [cit. 2024-04-14].

[5] KHANNA, Rahat. *Getting Started with Ionic*. Packt Publishing, první vydání, 2016, ISBN 978-1-78439-057-0.

[6] Ionic. *ion-grid: Display Grids for Mobile-First Custom App Layout*. Online. 2023. Dostupné z: https://ionicframework.com/docs/api/grid. [cit. 2024-04-14].

[7] Ionic. *ion-list: Item List View Component for iOS and Android Apps*. Online. 2023. Dostupné z: https://ionicframework.com/docs/api/list. [cit. 2024-04-14].

[8] Ionic. *ion-item: Input, Edit, or Delete iOS and Android Item Elements*. Online. 2023. Dostupné z: https://ionicframework.com/docs/api/item. [cit. 2024-04-14].

[9] Ionic. *Ionic Platform Styles | Platform-Specific Styles for Ionic Apps*. Online. 2023. Dostupné z: https://ionicframework.com/docs/theming/platform-styles. [cit. 2024-04-14].

[10] Ionic. *Capacitor by Ionic - Cross-platform apps with web technology*. Online. 2024. Dostupné z: https://capacitorjs.com/. [cit. 2024-04-14].

[11] Ionic. *App Development Core Concepts and Tools - Ionic Framework API*. Online. 2023. Dostupné z: https://ionicframework.com/docs/core-concepts/fundamentals. [cit. 2024-04-14].

[12] Ionic. *How to Install The Ionic Framework CLI to Build Mobile Apps*. Online. 2023. Dostupné z: https://ionicframework.com/docs/intro/cli. [cit. 2024-04-14].

[13] CHENG, Fu. *Build Mobile Apps with Ionic 4 and Firebase*. Apress, druhé vydání, 2018, ISBN 978-1-4842-3775-5.

[14] Ionic. *ionic start | Ionic Documentation*. Online. 2023. Dostupné z: https://ionicframework.com/docs/cli/commands/start. [cit. 2024-04-14].

[15] Ionic. *ionic serve | Ionic Documentation*. Online. 2023. Dostupné z: https://ionicframework.com/docs/cli/commands/serve. [cit. 2024-04-14].

[16] hunter__js. *Vue.js Declarative Rendering - GeeksforGeeks*. Online. Sanchhaya Education Private Limited. 2021. Dostupné z: https://www.geeksforgeeks.org/vue-js-declarative-rendering. [cit. 2024-04-14].

[17] SHAVIN, Maya. *Learning Vue: Core Concepts and Practical Patterns for Reusable, Composable, Scalable User Interfaces*. O'Reilly Media, Inc., první vydání, 2024, ISBN 978-1-4920-9882-9.

[18] Vue. *Single-File Components | Vue.js*. Online. 2023. Dostupné z: https://vuejs.org/guide/scaling-up/sfc.html. [cit. 2024-04-14].

[19] Vue. *Introduction | Vue.js*. Online. 2024. Dostupné z: https://vuejs.org/guide/introduction.html#api-styles. [cit. 2024-04-14].

[20] Seyfor, a.s. *Skladová evidence - přehledné vedení v ERP systému | Money ERP*. Online. 2022. Dostupné z: https://moneyerp.com/cs-cz/erp-je-skladova-evidence. [cit. 2024-04-14].

[21] KOHÚT, Tomáš. *eCommerce logistika: Co je to WMS a jak jej vybrat | Skladon*. Online. Skladon.cz. 2021. Dostupné z: https://skladon.com/cs/blog/ecommerce-logistika-co-je-to-wms-a-jak-jej-vybrat-seznam-vlastnosti-wms. [cit. 2024-04-14].

[22] Seyfor, a.s. *Skladové programy a systémy zdarma: kdy se vyplatí a na co dávat pozor - Money Blog*. Online. 2022. Dostupné z: https://money.cz/novinky-a-tipy/podnikani/skladove-programy-a-systemy-zdarma-kdy-se-vyplati-a-jaka-maji-omezeni. [cit. 2024-04-14].

[23] SAP SE. *Co je warehouse management system (WMS)? | SAP Insights*. Online. 2022. Dostupné z: https://www.sap.com/cz/products/scm/extended-warehouse-management/what-is-a-wms.html. [cit. 2024-04-14].

[24] Apertia Tech s.r.o. *O wms systémech v roce 2024 - Sklad*. Online. 2023. Dostupné z: https://skladsystem.cz/o-systemu-wms. [cit. 2024-04-14].

[25] STORMWARE s.r.o. *Sklady*. Online. 2020. Dostupné z: https://www.stormware.cz/pohoda/sklady. [cit. 2024-04-14].

[26] STORMWARE s.r.o. *Principy a metody*. Online. 2017. Dostupné z: https://www.stormware.cz/prirucka-pohoda-online/Sklady/Principy_a_metody. [cit. 2024-04-14].

[27] Green Fox Academy. *Co je no-code? Díky němu můžeš prorazit v IT i bez znalosti kódu*. Online. 2022. Dostupné z: https://www.greenfoxacademy.cz/post/co-je-no-code-diky-nemu-muzes-prorazit-v-it-i-bez-znalosti-kodu. [cit. 2024-04-14].

[28] Apertia Tech s.r.o. *Jaké CRM vám vyhovuje? Zjistíte v našem porovnání CRM*. Online. 2023. Dostupné z: https://autocrm.cz/porovnani-crm. [cit. 2024-04-14].

[29] Apertia Tech s.r.o. *Ceník - AutoERP*. Online. 2023. Dostupné z: https://autoerp.cz/cenik. [cit. 2024-04-14].

[30] K2 atmitec s.r.o. *Snadná správa obchodních záležitostí | Informační systém K2*. Online. 2016. Dostupné z: https://www.k2.cz/cs/obchod. [cit. 2024-04-14].

[31] K2 atmitec s.r.o. *Řízený sklad – WMS pro skladové hospodářství | Informační systém K2*. Online. 2017. Dostupné z: https://www.k2.cz/cs/rizeny-sklad-wms. [cit. 2024-04-14].

[32] REGALSISTEM s.r.o. *Top skladové systémy pro efektivní správu skladu | REGALSISTEM*. Online. 2023. Dostupné z: https://www.regalsistem.cz/aktuality/skladove-systemy-ktere-vam-pomohou-s-efektivni-spravou-sklad. [cit. 2024-04-14].

[33] YAVUZ, Baybars. *What is SAP Extended Warehouse Management (SAP EWM)? | MDP Group*. Online. MDP Group AG. 2022. Dostupné z: https://mdpgroup.com/en/blog/what-is-sap-extended-warehouse-management-ewm. [cit. 2024-04-14].

[34] Tutorials Point India Private Limited. *CST Studio Suite 3D EM simulation and analysis software*. Online. 2016. Dostupné z: https://www.tutorialspoint.com/sap_ewm/sap_ewm_overview.htm. [cit. 2024-04-14].

[35] MANN, Howie. *REST APIs Explained - 4 Components*. Online. 2023. Dostupné z: https://mannhowie.com/rest-api. [cit. 2024-04-18].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SFC	Single-File Component
CRM	Customer Relationship Management
CLI	Command-Line Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
LIFO	Last In, First Out
ERP	Enterprise Resource Planning
EAN	European Article Number
API	Application Programming Interface
REST	Representational State Transfer
URL	Uniform Resource Locator
PHP	PHP: Hypertext Preprocessor
ID	Identifikátor

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. HODNOCENÍ PRÁCE

Jméno studenta: Jaroslav Bělák
Název práce: Multiplatformní mobilní aplikace se skenerem skladových zásob

Student prokázal významný přínos v oblasti pracovních procesů a softwarového vývoje v naší firmě od roku 2022 až do současnosti. Zde je shnutí jeho přínosů a schopností:

Student se aktivně zapojoval svými postřehy během vývoje a zvládal agilně reagovat na změny v zadání z naší strany.
Nastudoval si dokumentaci pro námi dodaný hardware Zebra TC21, pro který vytvořil mobilní aplikaci, kde měl za úkol zprovoznit snímač 1D/2D kódů.
Student se aktivně podílel na úpravách CRM pro potřeby jeho aplikace. Měnil databázovou strukturu, modely, listenery, Nakonec přidal do CRM controller pro jeho API endpointy pomocí kterých komunikoval s mobilní aplikací.
Podílel se na tvorbě unit testů pro backend v PhpUnit, bez kterých bychom nebyli schopni bezpečně rozšiřovat kód.
Student dostával postupně UX zadání pro mobilní aplikaci, kterou rozšiřoval podle požadavků z naší strany. Aplikaci navíc rozšířil o možnost skenování 1D/2D kódů pomocí kamery, díky čemuž ji mohli používat i zaměstnanci na firemních telefonech bez snímače.
Aktuální verze software, kterou student vytvořil, ušetří naší společnosti 2-3h lidské práce denně. Aktuálně s námi student spolupracuje na dalších rozšířeních, které nejsou součástí jeho bakalářské práce, a které povedou k ještě významnější časové úspoře a zlepšení přesnosti dat.
Student pracoval samostatně a byl schopen si rychle problematiku nastudovat, má nadstandardní znalosti webových technologií oproti jeho vrstevníkům. V budoucnu by se měl zaměřit na větší preciznost a komunikaci v týmu.

Ve Zlíně 22.04.2024
v zastoupení Cetria s.ro.
Michal Štramberský