FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Mobilní a tabletové zobrazení aplikace Optilynx
Student:	Martin Huber
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem této práce je realizace mobilního a tabletového zobrazení aplikace pro optiky Optilynx.
Postupujte v těchto krocích:
1. Analyzujte možnosti využití aplikace Optilynx na mobilních zařízeních.
2. Na základě analýzy navrhněte vhodné řešení, neopomeňte potřeby tisku účtenek a zakázkových karet.
3. Implementujte prototyp frontendu, napojený na již funkční a hotové API backendu.
4. Ověřte funkcionalitu frontendu na reálných uživatelích desktopové verze.
5. Získané poznatky z testování využijte k realizaci finálního frontendu.
6. Zhodnoťte finální řešení a navrhněte případné rozšíření do budoucna.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Mobilní a tabletové rozhraní Optilynx

Martin Huber

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

13. května 2019

Poděkování

Především bych chtěl poděkovat vedoucímu této práce, Ing. Jiřímu Hunkovi, za poctivé vedení práce a poskytnuté rady. Také bych chtěl poděkovat Ing. Filipu Glazarovi a Bc. Oldřichu Malcovi za průběžnou spolupráci. Dále i všem, kteří se zúčastnili testování. V neposlední řadě patří dík i rodině a přátelům za poskytnutou podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2019 .....

Odkaz na tuto práci

Huber, Martin. *Mobilní a tabletové rozhraní Optilynx*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Cílem této bakalářské práce je realizovat mobilní zobrazení informačního systému pro optiky. Je provedena analýza stávajícího mobilního zobrazení a analýza možností použití informačního systému na mobilních zařízeních. Pro realizaci vhodného řešení jsou popsány použité technologie. Poté je řešení implentováno formou úpravy webové aplikace ve frameworku Angular. Implementovaný prototyp je otestován na reálných uživatelích desktopové verze a výsledky testování jsou vyhodnoceny a zapracovány do finální verze zobrazení.

Klíčová slova responzivní web design, uživatelské rozhraní, frontend, webová aplikace, Angular, Material Design, TypeScript

Abstract

The purpose of this bachelor's thesis is to realize a responsive user interface for an information system used by an eye optics company. The analyses of the current user interface and of the usage possibilites on mobile devices are performed. The technologies used to realize the appropriate solution are described. A prototype is implemented by expanding upon the original web application realized with the Angular framework. The prototype is then tested

on real users of the dektop version. The test results are evaluated and implemented into the final responsive interface.

Keywords responsive web design, user interface, frontend, web application, Angular, Material Design, Typescript

Obsah

Seznam obrázků

Seznam zdrojových kódů

Seznam tabulek

Úvod

Mobilní technologie hýbou dnešním světem. Dá se prohlásit, že v posledních letech značně ovlivnily podobu komunikace a používání aplikací. Spousta společností či firem nějakým způsobem podporují své uživatele se zaměřením na použití mobilních technologií. Na druhou stranu se tento trend netýká výhradně jen koncových uživatelů. Firma se může rozhodnout vyvinout mobilní aplikaci i pro své zaměstnance, kteří ji pak mohou využívat mj. ke zvýšení své flexibility a časové efektivity.

Právě z těchto důvodů jsem si vybral tuto bakalářskou práci. Tato práce navazuje na systém Optilynx – webový informační systém pro středně velké optiky. Klade si za cíl vytvořit tzv. responzivní zobrazení jakožto součást uživatelského rozhraní, tj. zajistit, aby se tato aplikace korektně zobrazovala nejen na velkých počítačových obrazovkách, ale i na obrazovkách menších, mobilních zařízení – na tabletech a mobilních telefonech – spolu se zachováním plné funkcionality původní aplikace.

K dosažení tohoto cíle si ve své bakalářské práci kladu dílčí cíle. Prvním z nich bude analýza využitelnosti aplikace Optilynx na mobilních zařízeních. Následuje analýza stávajícího uživatelského rozhraní včetně odhalení chyb a nežádoucích vlastností jeho responzivity. Dalším cílem bude návrh odpovídajícího řešení a jeho následná implementace.

Poté bude aplikace otestována na reálných uživatelích a poznatky získané z tohoto testování bude třeba promítnout na realizaci finálního řešení. Konečným cílem bude zhodnocení finálního řešení a návrh potenciálních rozšíření a zlepšení do budoucna.

Kapitola 1

Seznámení s oční optikou a stávajícím rozhraním aplikace Optilynx

Za účelem navržení a realizace korektního řešení problému této bakalářské práce je třeba porozumět dané problematice a popsat ji. Právě tomuto se věnuje první kapitola. Nejprve popíše společnost optik a systém Optilynx, který společnost používá, včetně nejdůležitějších funkcionalit systému a popisu jednotlivých obrazovek aplikace. Následovat bude analýza možností využití aplikace Optilynx na mobilních zařízeních. Další sekce kapitoly se zaměřují na obecnou definici uživatelského rozhraní a vyčlenění nedostatků mobilního rozhraní stávající aplikace Optilynx včetně několika ilustrací. V poslední části kapitoly se čtenář seznámí s analýzou konkurenčních aplikací.

1.1 Popis společnosti optik

Společnost, jejíž pobočky očních optik využívají informační systém Optilynx, je např. firma OČNÍ OPTIKA BENEŠOV s.r.o. [1], která zároveň i úzce spolupracuje na rozšíření systému Optilynx. Firma nabízí všechny standardní služby a produkty z oblasti očních optik, včetně poradenství s potřebami optických korekčních pomůcek, vyměřování pomůcek či prodeje brýlových čoček, obrub a kontaktních čoček.

V současné době společnost zaměstnává dohromady 15 zaměstanců ve svých 5 pobočkách v Benešově u Prahy, Vlašimi, Sedlčanech, Týnci nad Sázavou a ve Voticích. Informační systém Optilynx využívá ve svých pobočkách od roku 2018.

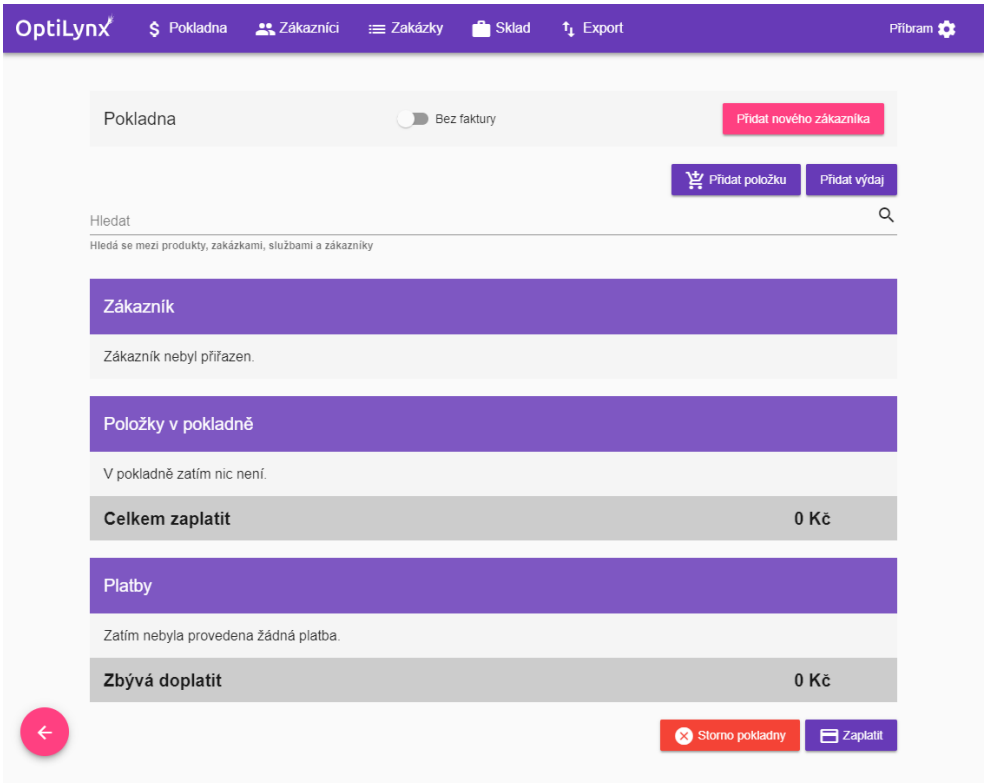
1.2 Informační systém Optilynx

Informační systém Optilynx je výsledkem diplomových prací dvou studentů FIT ČVUT, tehdejších bakalářů Bc. Jaroslava Hrácha [2] a Bc. Filipa Glazara [3]. Pan Hrách vytvořil pro systém Optilynx jeho frontendovou část (uživatelská prezentace, sběr dat od uživatele), pan Glazar se pak postaral o backend (zpracovávání a ukládání uživatelských dat). Ačkoliv je tato bakalářská práce blíže spjata s diplomovou prací pana Hrácha vzhledem k souvislosti s frontendovým návrhem (a jeho rozšířením), bylo pro mne důležité prostudovat si obě diplomové práce za účelem pochopení fungování informačního systému Optilynx jako celku.

Frontendová část systému Optilynx je webová aplikace implementovaná ve frameworku Angular s využitím frontendové knihovny Angular Material. Backendová část systému je pak realizována pomocí frameworku Laravel. Detailnímu rozboru technologií frontendové části systému Optilynx se věnuji v sekci 2.4.

Informační systém Optilynx uživatelům z oblasti prodeje optických pomůcek umožňuje spravovat každodenní procesy na pobočce oční optiky. Uživatel v rámci systému mj. může:

- obsluhovat zákazníka skrze pokladnu:
  - přidat zboží na pokladnu
  - přidat zakázku na pokladnu
  - provést platbu a vytisknout účtenku
  - provést prodej na fakturu
- spravovat zákazníky
- vytvářet zakázky
- spravovat skladové zboží
  - zadat nový produkt do systému
  - naskladnit či vyskladnit produkty
  - párovat faktury
  - spravovat přesuny produktů mezi sklady
  - spravovat dodavatele
- provádět export dat skrze tisk či stažení souboru



Obrázek 1.1: Aplikace Optilynx – desktopová verze, obrazovka pokladny

1.3 Popis obrazovek aplikace

Po otevření aplikace ve webovém prohlížeči a zadání správných přihlašovacích údajů si uživatel vybere danou pobočku, kterou chce spravovat. Po výběru přejde aplikace automaticky do obrazovky *Pokladna*, která představuje proces identfikace zboží či služby a následné zaplacení. Lze zde vyhledávat skrz produkty, služby i zakázky, přidávat je do stavu pokladny a následně vše zaplatit s možností volby a rozdělení způsobu platby. Lze zde také přidat nového zákazníka do systému.

V obrazovce *Zákazníci* může uživatel prohledávat záznamy zákazníků podle různých kritérií, zobrazit si detail konkrétního zákazníka, vytvořit pro něj novou zakázku či taktéž přidat nového zákazníka do systému. Obrazovka přidávání nového zákazníka poskytuje formulář pro zadání zákazníkova jména, rodného čísla, adresy a dalších informací. Detail zákazníka pak obsahuje informace zadané v tvorbě jeho záznamu spolu s přehledem zakázek zákazníka.

Pro správu samotných zakázek slouží obrazovka *Zakázky*. Umožňuje mezi nimi vyhledávat, vydat zakázku, zobrazit detail zakázky nebo vytvořit novou zakázku. Výdej zakázky přesune uživatele zpět do pokladny, kam se vloží daná zakázka. Tvorba nové zakázky vyvolá opět obrazovku zákazníků.

Při tvorbě nové zakázky je možné upravit informace o zákazníkovi. Následuje zadávání detailních informací o zakázce, jako jsou parametry čoček, obruby, doplňkové služby či produkty. Následně může uživatel zadat i slevové poukazy, zakázku uložit a případně nechat zaplatit zálohu.

Obrazovka *Sklad* skýtá možnosti vyhledávání ve skladových položkách, přidávání nových položek, přidávání položek na pokladnu, prohlédnutí detailu položky, naskladnění, párování faktur či správu dodavatelů. Lze si prohlédnout historii přesunů mezi sklady i historii naskladnění. Také si uživatel může otevřít stav skladu či obrazovku s inventurou.

Jednotlivé obrazovky aplikace doprovází všudypřítomná horní lišta s hlavní navigací mezi důležitými obrazovkami. Horní lišta obsahuje i tlačítko s rozbalovacím menu, které obsahuje odkazy na další obrazovky aplikace. Zde lze přejít na uzavření pokladny, zobrazit přehledy pokladny, její historii, pokladní deník či upravit informace o organizaci a jejích pobočkách.

Dalším stálým prvkem aplikace je tzv. FAB - Floating Action Button, které představuje tlačítko „zpět“ fungující jako způsob přechodu k předešlému stavu aplikace.

1.4 Analýza využitelnosti aplikace Optilynx na mobilních zařízeních

Mobilní rozhraní aplikace Optilynx najde využití především v rozšíření flexibility pracovníků optik. Pracovník používající mobilní verzi aplikace bude moci být blíže zákazníkovi, což bude výhodné zejména při tvorbě zakázky, kdy pracovník jednak vyplňuje detaily čoček, ale zároveň i nabízí zákazníkovi další produkty z nabídky v prodejně a případně vyhledává informace o produktech, jako je cena nebo dostupnost na skladě.

Tento způsob využití mobilního rozhraní aplikace Optilynx pak může zvýšit časovou efektivitu interakce se zákazníkem. Další možnost využití spočívá v dostupnosti aplikace odkudkoliv s internetovým připojením. Pokud např. nastane nějaký problém v procesu optiky mimo pracovní dobu, může pracovník bez přístupu ke stolnímu počítači řešit problém skrz mobilní rozhraní.

Ne každý uživatelský proces je efektivnější za použití mobilního rozhraní než za použití rozhraní pro stolní počítač. Je třeba vzít v potaz omezující okolnosti mobilních zařízení, např. velikost obrazovky nebo způsob zadávání dat. Vždy budou existovat uživatelské procesy, které budou pohodlněji a časově efektivněji proveditelné skrz rozhraní pro stolní počítač. V návrhu mobilního zobrazení je pak toto třeba zohlednit a uzpůsobit návrh slabým stránkám mobilních zařízení.

1.5 Uživatelské rozhraní

Návrh uživatelského rozhraní je součást oblasti studie interakce mezi člověkem a počítačem [4]. Uživatelské rozhraní je ta část systému (počítače a jeho programového vybavení), kterou uživatel dokáže vnímat a ovládat. Základními dvěma součástmi uživatelského rozhraní jsou vstup a výstup. Vstup představuje způsob, jakým uživatel zadává své požadavky nebo vyjadřuje své potřeby systému. Mezi moderní zařízení sloužící k zadávání vstupu počítači patří např. počítačová myš, klávesnice nebo dotyková obrazovka, jež je pro tuto bakalářskou práci nejrelevantnější.

Výstupem se pak rozumí způsob předání zpracovaných výsledků a požadavků počítače uživateli. Výstupní zařízení obvykle používaná jakožto součást uživatelského rozhraní jsou především obrazovka a případně i reproduktory či jiná zvuková zařízení.

Správný návrh uživatelského rozhraní systému vyvažuje dobře navržené vstupní a výstupní mechanismy, které uspokojují a pokrývají požadavky, schopnosti a omezení uživatele co nejefektivněji. Nejlepší rozhraní je takové, jež umožní uživateli soustředit se na svou práci a odstínit uživatele od mechanismů v pozadí, které jsou použity pro zprostředkování a zpracování informací.

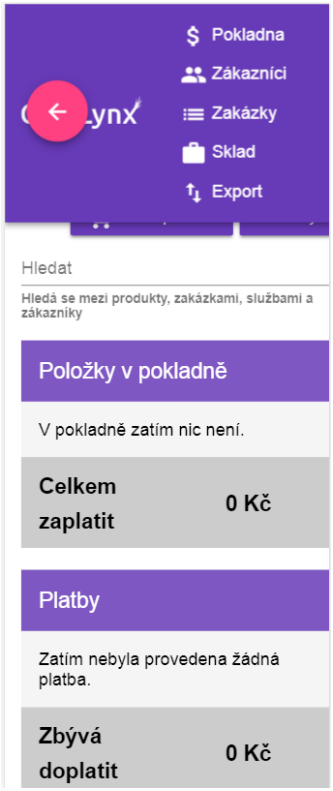
Návrh a vývoj uživatelského rozhraní je dnes nedílnou součástí vývoje počítačových aplikací. Tvorba uživatelského rozhraní webové aplikace pro mobilní zařízení je náplní této bakalářské práce. Z tohoto důvodu je mobilnímu uživatelskému rozhraní a tzv. *responzivnímu web designu* věnována samostatná sekce 2.2.

1.6 Nedostatky mobilního rozhraní aplikace Optilynx

Responzivita v původní frontendové aplikaci Optilynx je přítomna jen v malém rozsahu. Nebyla původně vyvíjena pro mobilní zařízení, nýbrž pro stolní počítače. Autor původní aplikace pro dosažení alespoň základní responzivity aplikace používá pár jednoduchých CSS pravidel. Jsou to tzv. *media query* aneb CSS pravidla, jež dokáží reagovat na mediální vlastnosti přístroje, ve kterém se aplikace zobrazuje.

Použitá *media query* pravidla v původní aplikaci Optilynx ovlivňují především šířku kontejneru obsahu zobrazovaného na obrazovce. Na počítačovém zobrazení je šířka tohoto kontejneru fixně zadefinována a zabírá jen prostřední část obrazovky. Pro menší obrazovky je zadefinováno několik *media query* pravidel, jež kontejner zužují do daných fixních šířek, obsah pak opět zabírá jen určitou šíři obrazovky.

Tato *media query* pravidla jsou vždy definovaná pro obrazovky širší než daný počet pixelů. Pokud pro obrazovku neplatí ani *media query* pravidlo



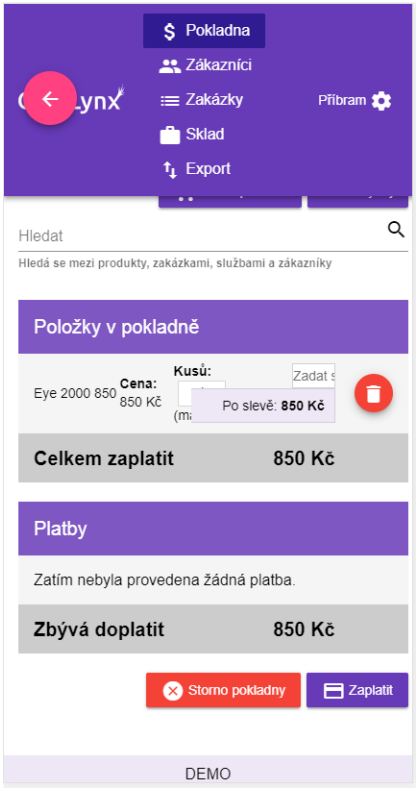
Obrázek 1.2: Původní mobilní zobrazení Optilynx – přesah horní lišty

s nejmenší šířkou, pak se obsah škáluje úměrně k šířce obrazovky. Škálování obsahu je nedostatkem z důvodu zhoršení čitelnosti textu.

Dalším nedostatkem aplikace je neresponzivní horní nabídka. Při zužování obrazovky se kontejner, který menu obsahuje, snaží nabídku vtěsnat do šířky obrazovky uspořádáním do více řad, případně do sloupce. Pokud to ovšem šířka obrazovky už nedovolí, pak horní lišta s nabídkou začne přetékat přes obrazovku, čímž se stránka stává vertikálně posuvnou a tlačítko nastavení se posouvá mimo obrazovku. Zároveň v této situaci horní lišta překrývá začátek vlastního obsahu stránky (obr. 1.2).

Většina obsahu při zužování obrazovky začne přetékat mimo kontejner obsahu, a tedy i samotnou obrazovku. Tento problém se týká především tabulek, které obsahují velké množství sloupců. Jiné tabulky v úzkých obrazovkách zase přetékající obsah skryjí, proto se k němu uživatel nemůže dostat ani posouváním stránky. Podobně i nadbytečné položky skladové podnabídky se nezobrazí vůbec a také se k nim nelze dostat ani posouváním.

Obsah hlaviček zakázek na stránce Zakázky se na malé obrazovce začne vzájemně překrývat. Podobně tak tomu je i při vyhledávání produktů nebo služeb v obrazovce pokladny či v obrazovkách skladu. Informace *po slevě* u pro-



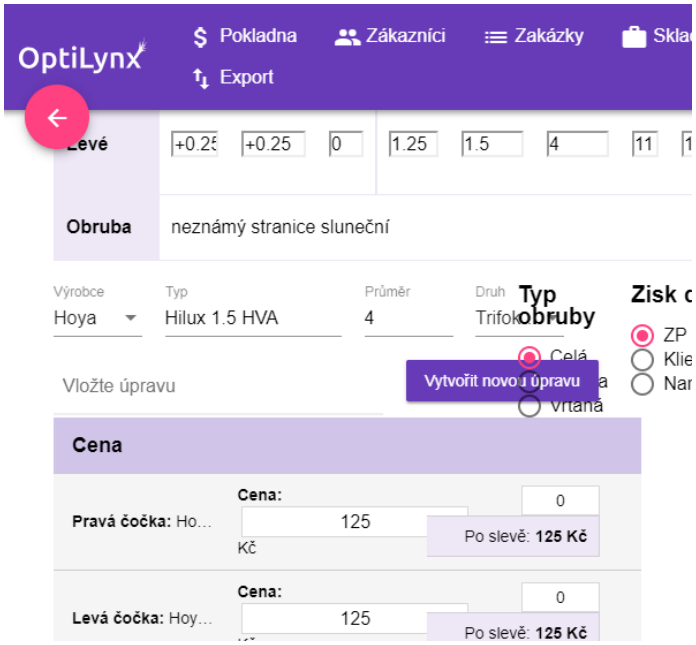
Obrázek 1.3: Původní mobilní zobrazení Optilynx – překryv pole pro zadání počtu kusů

duktu v pokladně překrývá pole pro udání počtu kusů (obr. 1.3). Překrývání obsahu se dále objevuje např. při tvorbě zakázky u výběru typu obruby (obr. 1.4).

Souhrnem všech problémů, jež se budu snažit ve své bakalářské práci vyřešit, se zabývá tabulka 1.1. V tabulce je k jednotlivým problémům uvedeno i předpokládané řešení problému a číslo závažnosti, které vychází z mé škály závažnosti, již jsem si pro tento účel vytvořil. Škála obsahuje čísla od 1 (s významem nejzávažnější problém) do 5 (významově marginální problém).

1.7 Konkurenční aplikace

Analýza konkurence je dobrým nástrojem k získání obecného přehledu o úrovni současných řešení dané problematiky. Lze tak identifikovat prvky konkurenčních řešení, jež mohou buď sloužit jako inspirace k tvorbě vlastního řešení, nebo mohou také poskytovat náhled na části, které by bylo vhodné vyřešit lepším způsobem a vyvarovat se tak konkurenčních chyb.



Obrázek 1.4: Původní mobilní zobrazení Optilynx – překryv výběru typu obruby u tvorby zakázky

1.7.1 Přímá konkurence

Systémů pro správu optik pro stolní počítače je na českém trhu hned několik (např. **OpticEvidence** či **Newton**) [2], ale žádný takový systém neposkytuje mobilní aplikaci či webové zobrazení pro mobilní zařízení. Podrobné analýze konkurence se v minulém roce věnoval Ing. Jaroslav Hrách ve své diplomové práci. Inspirace a poznatky vycházející z této analýzy přispěly k návrhu a tvorbě původního systému Optilynx.

OpticEvidence je software pro evidenci zákazníků a zakázek v rámci optiky. Mezi funkce tohoto systému dále patří i správa faktur či různé možnosti exportu dat. Tento systém je ovšem nabízen pouze ve formě aplikace pro desktopové systémy Microsoft Windows.

Informační systém **Newton** je dílem společnosti MAXOFT, s.r.o. Systém nabízí podobné funkce jako systém **OpticEvidence**, včetně evidence zakázkových karet a vyšetření. Opět je tento systém k dispozici pouze jako aplikace pro Windows.

1.7.2 Nepřímá konkurence

Vzhledem k tomu, že žádný dostupný systém pro správu optik nenabízí mobilní rozhraní, lze alespoň analyzovat jiné systémy, které se zaměřením a funkcionalitami podobají systémům pro správu optik a disponují i mo-

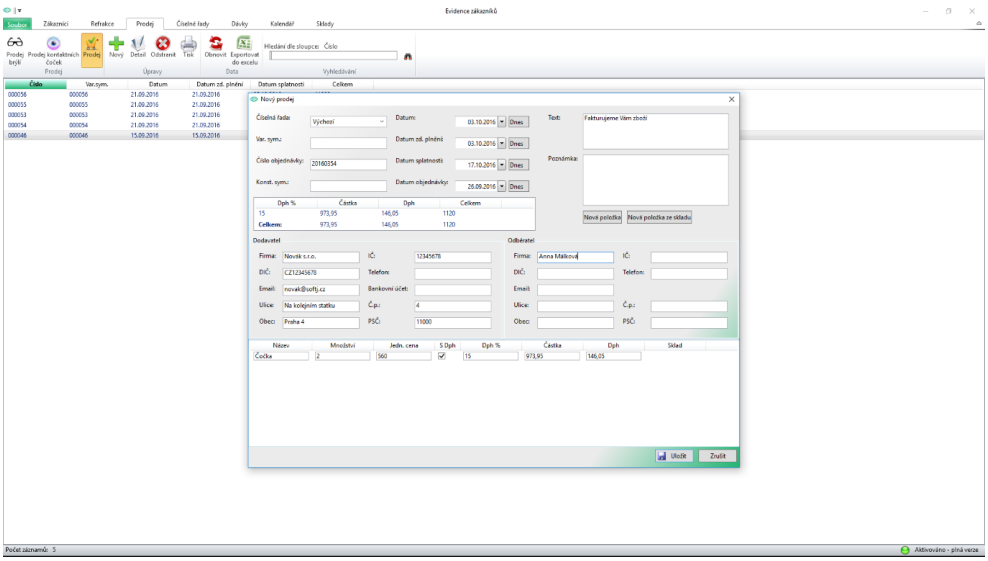
Tabulka 1.1: Souhrn problémů mobilního zobrazení a předpokládaných řešení

Problém	**Předpoklad řešení**	**Závažnost**
zobrazení kontejnerů obsahu	zavedení jednotné relativní mobilní šířky	1
hlavní nabídka, skladová podnabídka	sbalení do postranní vysouvací nabídky	1
tlačítka specifická pro jednotlivé stránky	sbalení do podnabídek	3
vyhledávací nabídky v obrazovkách napříč aplikací	kompaktnější zobrazovací pravidla, záměna popisů a tlačítek za ikony	2
produktové přehledy v pokladně a ve skladu	sbalení zadávacích polí pod samostatná tlačítka, zavedení ikon	2
tabulky napříč aplikací	redukce dohledatelných informací, přeuspořádání prvků	3
modální okna	případné přeuspořádání prvků	5
zakázkové karty v přehledu zakázek	redukce informací, sloupcové zobrazení tabulek, sbalovací kontejnery	3
detail zakázkové karty	úprava tabulky čoček na sloupcové zobrazení, přeuspořádání prvků	2
formulářové karty produktů, zákazníků a dodavatelů	přeuspořádání formulářových prvků	3
patička aplikace	přeuspořádání, vynechání nepodstatných informací	5

bilním uživatelským rozhraním. K analýze jsem si vybral tři takové systémy: **Fakturoid**, **iDoklad** a **KASA FIK**. První dva systémy poskytují správu daňových dokladů, což je i jedna ze složek systému Optilynx, zároveň obsahují i různé typy přehledů, jež v aplikaci Optilynx také tvoří velkou část obrazovek. Třetí systém je pak zaměřen na prodej a EET evidenci. Prodej je také jednou z dílčích částí systému Optilynx.

1.7.2.1 Fakturoid

Fakturační program Fakturoid vytvořili Lukáš Konarovský a Jan Korbel [7]. Jedná se o webovou aplikaci uzpůsobenou k zobrazení na velkých obrazovkách stolních počítačů i na malých obrazovkách přenosných zařízení. Mezi funkcionality Fakturoidu, obsažené v různých zdarma i placených tarifech, patří



Obrázek 1.5: Aplikace OpticEvidence [5]

mj. evidence faktur a nákladů, statistiky příjmů a výdajů, vystavování pravidelných faktur či export faktur do účetních programů.

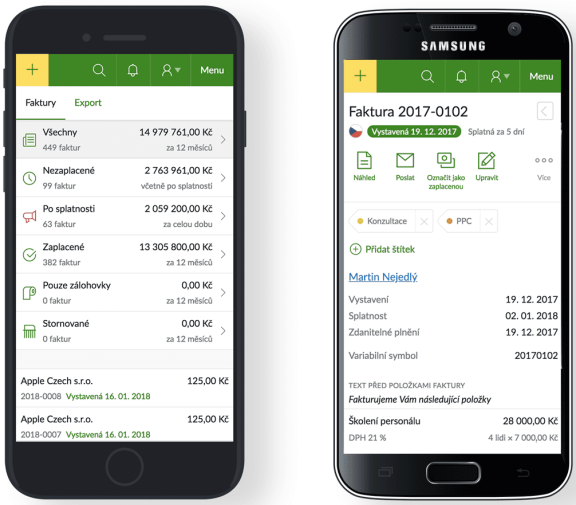
Návrh mobilního uživatelského rozhraní aplikace působí moderním dojmem. Prvky uvnitř aplikace jsou uspořádány přehledně a uživatel se intuitivně dokáže s rozhraním seznámit během několika minut. Vzhled mobilního rozhraní aplikace je také konzistentní s rozhraním pro velké obrazovky.

Mobilní rozhraní správně řeší tabulky zobrazením nejpodstatnějších informací a použitím kombinace velikosti písma a jeho barvy ke zvýšení přehlednosti a rozdělení důležitosti zobrazovaných informací. Konkrétní úkony faktur jsou schované v detailu faktury, kdy je naopak využita možnost vertikálního posouvání a jsou zde uvedeny všechny informace o faktuře, seřazené dle důležitosti. Specificky návrh tabulek v mobilním rozhraní této aplikace mi sloužil jako inspirace k mojí bakalářské práci.

1.7.2.2 iDoklad

Služba iDoklad je dílem společnosti Solitea [10]. Kromě webové aplikace jsou ke stažení dostupné i samostatné aplikace pro mobilní operační systémy Android a iOS. Služba na všech platformách umožňuje mj. vystavování faktur, správu kontaktů a ceníku či export faktur do formátu PDF nebo několika informačních systémů.

Kvalita realizovaného mobilního rozhraní se liší mezi platformami. Aplikace pro Android vzhledově působí zastarale a neobsahuje všechny funkce své webové verze. Varianta pro systém iOS naopak vypadá velice moderně, avšak



Obrázek 1.6: Fakturoid – mobilní rozhraní [6]

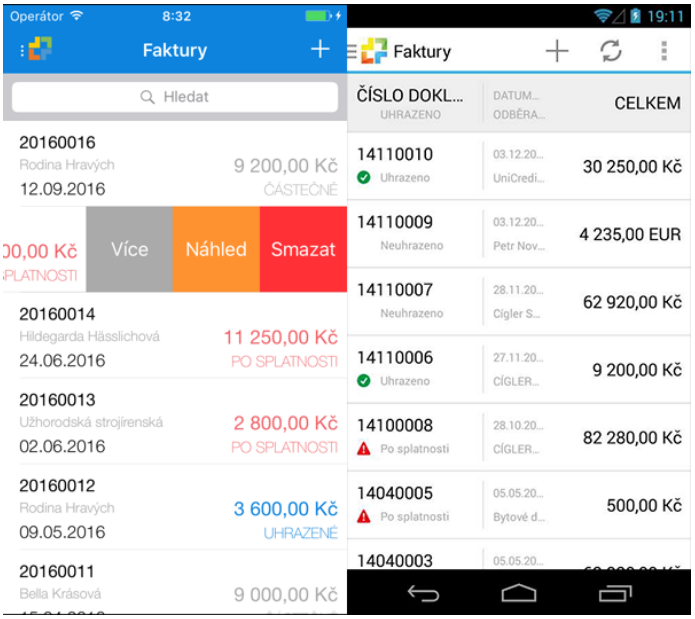
funkcemi se podobá své Android verzi. Uživatel se v obou variantách také zorientuje poměrně rychle.

Podobně jako Fakturoid i tato služba ve svých mobilních aplikacích k zobrazení přehledů používá velikost písma a barvu např. k označení čísla faktury a jejího stavu v přehledu faktur. V přehledech je ovšem snaha zobrazit co nejvíce informací, což mnohdy vede ke zkrácení dlouhých textů v tabulce přehledu. Podobným problémem trpí i tabletová zobrazení aplikací této služby, kdy není využita širší obrazovka ke správnému zobrazení přehledů, nýbrž ke zobrazení přehledového seznamu i detailu faktury zároveň, což je přístup typický např. pro desktopové emailové klienty. Zde je ovšem tento přístup zbytečný.

1.7.2.3 KASA FIK

Systém KASA FIK je mobilní aplikace pro Android, provozovaná společností KASA FIK s.r.o. [11]. Kromě mobilní aplikace společnost poskytuje i samostatné pokladní systémy s dotykovými obrazovkami v nepřenosné i přenosné variantě. Aplikace pak poskytuje funkce položkového prodeje, výdej a ukládání účtenek, skladové hospodářství či denní přehledy prodeje.

Podobně jako v systému Optilynx se i v aplikaci KASA FIK nacházejí formuláře pro tvorbu nových zákazníků či produktů. Formulářové prvky jsou v oknech těchto formulářů zobrazeny do sloupce, což hodnotím jako kladný krok pro zobrazení na užších obrazovkách. Tento způsob zobrazení formulářů by se tedy dal použít i v mobilním rozhraní aplikace Optilynx.



Obrázek 1.7: iDoklad – mobilní rozhraní pro systémy iOS a Android [8] [9]

Prodej je v aplikaci řešen dvěma způsoby: kalkulačkovým a položkovým. Kalkulačový umožňuje zadat jednu nebo více částek a následně zvolit možnost platby. Položkový způsob předpokládá, že v ceníku jsou vloženy potřebné položky. Samotný prodej probíhá vyhledáváním položek v ceníku a přidáváním do pokladny. Oba tyto způsoby určitě najdou využití např. v hospodách nebo restauracích, ovšem prodej v optice je založen na komplexnějším vyhledávání produktů a zakázek, k čemuž lze využít např. vyhledávacího pole.

Obrázek 1.8: KASA FIK – mobilní rozhraní aplikace pro Android

Kapitola 2

Návrh mobilního zobrazení

Tato kapitola se zabývá hlubším teoretickým rozborem návrhu uživatelského rozhraní a následnou přípravou na jeho realizaci. První část kapitoly rozebírá různé typy aplikací a návrhových přístupů a přibližuje čtenáři problematiku responzivního webového rozhraní. Zbytek kapitoly detailně popisuje technologie použité k realizaci řešení této bakalářské práce.

2.1 Volba typu aplikace

Při úvaze o volbě typu aplikace, která bude řešit zadaný problém, jsem bral v potaz dvě možnosti: tvorbu nativní mobilní aplikace či úpravu stávající webové aplikace. Původní aplikace Optilynx je navržena jako webová aplikace s ohledem na požadavky na navrhovaný systém. Autoři původní aplikace při výběru postupu v návrhu brali v potaz dva hlavní požadavky: nezávislost informačního systému na platformě a jeho dostupnost odkudkoliv.

Pro zachování a splnění těchto požadavků v návrhu a realizaci svého řešení se úprava stávající aplikace jeví jako vhodnější volba. Cesta tvorby mobilní aplikace by ve skutečnosti znamenala buď tvorbu dvou samostatných aplikací pro systémy iOS a Android, aby se pokryla většina zařízení, na kterých by mohli uživatelé informační systém používat, anebo by se vytvořila jen jedna aplikace, což ovšem snižuje časovou náročnost realizace řešení na úkor množství použitelných zařízení.

Tvorba byť i jen jedné mobilní aplikace by ale představovala návrh a realizaci plnohodnotné frontendové aplikace, což by se časovou náročností podobalo tvorbě původní frontendové aplikace pana Hrácha. Z těchto důvodů jsem se tedy rozhodl pro úpravu stávající webové aplikace.

Rozšíření aplikace Optilynx o mobilní rozhraní neovlivní splnitelnost dvou uvedených požadavků, naopak obohatí množství použitelných zařízení o mobilní zařízení, čímž zvýší uspokojení požadavku nezávislosti na platformě. Jedním z cílů této bakalářské práce je i to, aby navržené mobilní rozhraní obsahovalo všechny funkce stávající aplikace, tudíž úprava webové aplikace

by pořád měla splňovat požadavek, aby se mohl uživatel přihlásit do systému odkudkoliv a měl přístup ke stejným a aktuálním datům.

Úprava webové aplikace ještě poskytuje výhodu případného rozšíření do budoucna v rámci mobilní aplikace. Z webové aplikace lze totiž vytvořit nativní mobilní aplikaci vložením do tzv. obálky, která kromě instalace aplikace na mobilní systém zajistí mj. i přístup aplikace k nativním funkcím systému. Lze tak např. využít i možností tisku nebo čtení čárových kódů. Tuto funkcionalitu poskytuje např. framework Electron [12].

2.2 Responzivní webové rozhraní

Problém zobrazení webových stránek na mobilních zařízeních vznikl teprve v minulém desetiletí s příchodem telefonu iPhone v roce 2007 [13]. Do té doby byly webové stránky navrhovány pro počítačové obrazovky fixní šířky, většinou přizpůsobené menším obrazovkám, s prázdnou výplní po stranách (tzv. *whitespace*) pro větší obrazovky.

Z hlediska uživatelské použitelnosti ovšem nebylo vhodné zobrazovat stránky v mobilním zařízení stejným způsobem jako na počítači. Nejdříve tedy začaly pro tehdejší weby vznikat jejich mobilní varianty, obvykle dostupné na webové adrese s předponou „m.“. S příchodem většího množství dotykových zařízení podobných telefonu iPhone s různými proporcemi obrazovek vývojářská komunita postupně narážela na další problém – mobilní weby byly pro některé obrazovky moc malé, zatímco klasické weby moc velké.

Odpovědí na tento problém je tzv. *media query*. Je to pravidlo v rámci jazyka CSS, který se používá k popisu prezentace dat na webových stránkách [14]. Pravidlo *media query* bylo specifikováno v iteraci jazyka CSS3 a slouží k rozšíření možností přizpůsobení webového obsahu různým mediálním zařízením [15]. Mezi takové možnosti patří i přizpůsobení obsahu různě širokým obrazovkám.

Dalším prvkem, jenž přispívá k vhodnému zobrazení webových stránek na mobilních zařízeních, je koncept flexibility. Velikosti kontejnerů, ve kterých jsou strukturovány prvky webových stránek, mohou být kromě fixních pixelových hodnot definovány i v jednotkách relativních vůči nadřazeným prvkům. Takovéto jednotky v CSS zpravidla bývají procenta či tzv. *em* (jednotky délky relativní vůči velikosti písma elementu).

Právě kombinace *media query* pravidel a konceptu flexibility dala vzniknout tzv. *responzivnímu web designu*. Jedná se o způsob, jak použít výše zmíněné koncepty dohromady k návrhu webových stránek, jež patřičným způsobem reagují na obrazovky různých velikostí.

2.3 Volba přístupu

Náplní této bakalářské práce je návrh a realizace zobrazení webové aplikace pro různá mobilní zařízení. Tato zařízení se dají podle své velikosti rozdělit do dvou kategorií: mobilní telefony a tablety. Vzhledem k těmto dvěma kategoriím je třeba se zamyslet nad přístupem návrhu odpovídajícího rozhraní.

S ohledem na různé proporce zařízení těchto dvou kategorií se jako korektní řešení jeví zohlednit v návrhu silné a slabé stránky těchto kategorií a mobilní rozhraní upravit pro každou kategorii zvlášť. Mobilní telefony obecně disponují úzkou obrazovkou, kde je třeba upravit uspořádání obsahu tak, aby se odpovídajícím způsobem zobrazil pro uživatele nejdůležitější obsah, a ostatní prvky případně schovat či strukturovat způsobem, kdy uživatel těchto prvků lehce dosáhne. Tabletová zařízení se velikostí přibližují počítačovým obrazovkám, proto některé úpravy pro mobilní telefony není třeba u tabletů uplatňovat.

Po dohodě s vedoucím práce Ing. Jiřím Hunkou jsem ale došel k závěru, že kvůli důvodu časové efektivity zvolím přístup návrhu se zaměřením na mobilní telefony. Případné úpravy pro tabletové rozhraní se pak budou odrážet od mobilního návrhu.

2.4 Použité technologie

Tato sekce obsahuje popis technologií použitých k realizaci mobilního zobrazení. Vzhledem k tomu, že byl zvolen přístup úpravy stávající aplikace, se technologie původní aplikace a mého řešení kryjí.

2.4.1 Hypertext Markup Language

Základním stavebním blokem webových stránek je Hypertext Markup Language (HTML) [16]. Je to značkovací jazyk, který definuje strukturu webového obsahu. Používá k tomu tzv. *markup*, což představuje způsob, jakým označit strukturu obsahu dokumentu tak, aby syntaxe označení byla od samotného obsahu rozpoznatelná. HTML také tvoří vazby mezi označkovanými dokumenty pomocí hypertextu. Tento termín označuje odkazy přítomné v označkovaných dokumentech, jež je propojují s ostatními dokumenty v rámci jednoho webu i mezi různými weby.

Podoba jazyka HTML je definována ve specifikacích, které vydává a spravuje konsorcium World Wide Web Consortium (W3C). Specifikace kromě definice syntaxe obsahuje i sémantiku značek způsob jejích správného využití. Poslední specifikací jazyka HTML je HTML5. Právě tato verze je použita ve frontendové aplikaci systému Optilynx.

2.4.2 CSS

Vedle vymezení obsahu webové stránky je v dnešní době důležitá i definice vzhledu prezentace dat na webu. K tomu se používá stylovací jazyk Cascading Style Sheets (CSS) [14]. Tento jazyk představuje způsob popisu prezentace webového obsahu. Mezi možnosti CSS patří stanovení grafického rozložení, barev, vlastností písem či úprava webové prezentace pro mobilní zařízení.

Pro uzpůsobení vzhledu dokumentu jazyk CSS definuje stylovací pravidla, jež se uplatňují na příslušné elementy HTML dokumentu a patřičným způsobem upravují jejich vzhled. Vedle uplatnění CSS pravidel na konkrétní HTML značky (tagy) lze využít i vlastních specifikátorů (tříd), které lze v HTML dokumentu doplnit k tagům jako atribut a tím přiřadit danou CSS třídu HTML elementu. Stylovací pravidla definovaná v třídě se pak uplatňují na daný element, čímž lze dosáhnout abstrakce a obecnosti a předejít zbytečnému či neužitečnému stylování elementů v celém HTML dokumentu.

Jazyk CSS je definován ve specifikacích taktéž spravovaných konsorciem W3C. Poslední specifikace CSS3 kromě jiných změn přinesla i pravidla *media query*, možnost nastavit barvu s průhledným kanálem či rozšíření CSS selektorů atributů.

2.4.2.1 Media query

CSS pravidlo media query umožňuje zobrazit různá CSS pravidla v závislosti na vlastnostech zařízení, jež danou webovou stránku zobrazuje. Prostřednictvím pravidla media query lze zjistit mj. výšku a šířku oblasti pro zobrazení, orientaci zařízení (na výšku či na šířku), dostupnost JavaScriptu nebo počet bitů na barvu výstupního zařízení [17].

Pravidla media query neovlivňují HTML dokument jako takový, spíše ovlivňují CSS pravidla, jež jsou v rámci HTML dokumentu použita. Uvnitř takové media query se tedy mohou nacházet pravidla pro přeuspořádání obsahu, skrývání či zobrazování částí obsahu, změny velikostí kontejnerů obsahu či úpravy jakéhokoliv jiného stylu, který lze aplikovat pomocí CSS.

2.4.2.2 Rozložení Flexbox Layout

Flexbox Layout je modul jazyka CSS, jenž rozšiřuje možnosti rozložení obsahu na webové stránce [18]. Poskytuje rozložení (*layout*) pro obsah, jehož proporce jsou neznámé či dynamické. Lze tak pro obsah jednoduše definovat mnohé vlastnosti jeho uspořádání včetně pořadí prvků, směru jejich uspořádání či způsobu výplně volného místa mezi uspořádanými prvky.

Vedle CSS pravidel media query je flexbox layout další důležitou součástí responzivního návrhu webové aplikace. Oprošťuje vývojáře od nutnosti definovat nadbytečná CSS pravidla a používat fixní šířky pro prvky uvnitř kontejnerů obsahu.

2.4.2.3 SASS

Psaní stylů pro webové stránky či aplikace pomocí čistého CSS je sice obvyklé, ale existují varianty jazyka CSS, jež původní syntaxi a funkcionalitu CSS rozšiřují. Takovéto nástroje se nazývají CSS preprocesory. Mezi CSS preprocesory patří mj. **SASS**, **LESS** a **Stylus**.

Syntactically Awesome Style Sheets (SASS) je stylovací jazyk, jehož kód je ve výsledku překládaný do CSS. Mezi možnosti a vlastnosti SASS patří tvorba proměnných, zápis vnořených CSS pravidel, vylepšené řešení importu stylovacích souborů nebo zápis znovupoužitelných pravidel, tzv. *mixins* [19].

Ostatní CSS preprocesory se od SASS moc neliší. Původní frontendová aplikace Optilynx ke stylování používá právě SASS. Vzhledem k tomu, že jsem nenašel žádný důvod k přechodu na jiný preprocesor, pokračoval jsem v použití preprocesoru SASS i ve své bakalářské práci.

2.4.3 JavaScript

Třetí ze základních webových technologií je JavaScript. Jedná se o programovací jazyk většinou používaný k dynamickému skriptování webových stránek a aplikací na straně klienta (*client-side programming*) ve webovém prohlížeči [20]. Tento jazyk vznikl v roce 1995 původně jako jazyk pro programování na straně serveru (*server-side programming*). Brzy poté ale jazyk přešel pod vývoj společností Netscape, jenž JavaScript uvedla ve svém webovém prohlížeči Netscape Navigator 2.0.

V roce 1996 ve spolupráci s ECMA International vznikla standardizace JavaScriptu zvaná ECMAScript. Organizace ECMA International jazyk spravuje dodnes. Poslední verze ECMAScriptu **ES2018** je k dispozici od června minulého roku. Ačkoliv se oficiální standardizace jazyka jmenuje ECMAScript, je tento název s názvem JavaScript zaměnitelný.

Moderní webové stránky a aplikace používají JavaScript k tvorbě interaktivních prezentací, které dokáží rozšířit svou funkcionalitu o asynchronní komunikaci se serverem pomocí technologie **AJAX** (*Asynchronous JavaScript and XML*), animace nebo interakci s webovými službami či zařízeními pomocí různých **API** (*Application Programming Interface*). JavaScript také poskytuje možnosti manipulace s HTML dokumentem skrze model **DOM** (*Document Object Model*), který vytvoří webový prohlížeč na základě strutury HTML dokumentu. Díky modelu DOM lze dynamicky upravovat, vkládat či odstranit části obsahu webové stránky.

Jazyk JavaScript v posledních letech našel použití i mimo webový prohlížeč skrze úspěšnou platformu **Node.js**. Je to multiplatformní *runtime environment* (česky běhové prostředí) umožňující vývojářům pomocí JavaScriptu vytvářet plnohodnotné webové servery či automatizovat činnosti na počítači.

2.4.4 TypeScript

JavaScript i přes svůj úspěch není jazyk vhodný pro vývoj a správu rozsáhlých aplikací. Z toho důvodu vzniklo rozšíření jazyka zvané TypeScript [21]. Toto rozšíření, spravované společností Microsoft, umožňuje vytvářet JavaScriptové aplikace psaním kódu syntakticky obohaceného mj. o typovou kontrolu, systém modulů či pokročilý koncept tříd, objektů a rozhraní.

TypeScriptové soubory se před spuštěním překládají do čistého JavaScriptu. Překlad umožní kompatibilitu s webovými prohlížeči, jež používají JavaScript, a zároveň dokáže odhalit syntaktické chyby nebo např. špatně použité datové typy a typové konverze. Spolu s TypeScriptem jsou k dispozici i vývojářské nástroje, které lze integrovat do vývojových prostředí.

2.4.5 Webový framework

Frontendovou webovou aplikaci lze vyvinout i s využitím pouze základních technologií (např. HTML, CSS, JavaScript, TypeScript). Vývoj aplikací většího rozsahu by tak ovšem pro vývojáře představoval výzvu s ohledem na časovou efektivitu i organizaci kódu. K vývoji větších webových aplikací se tak obvykle využívají tzv. webové frameworky.

Webový framework je nástroj, jenž poskytuje způsob, jak vyvíjet webové aplikace rychleji a odstínit přitom vývojáře od nepodstatných nebo nízkoúrovňových částí vývoje [22]. Framework obsahuje strukturu, která slouží k poskytnutí určitého vývojového standardu a organizuje kód podle rozdělení částí aplikace do různých vrstev (např. uživatelská vrstva, datový model, vrstva pro zpracování dat). Webové frameworky se dle použití dělí do několika kategorií:

Server-side framework Slouží především ke zpracování a zprostředkování dat, ke komunikaci s databází a s uživatelskou částí aplikace. Také nazýván *backend framework*. Mezi populární backend frameworky patří:

- Django
- Laravel
- Zend

Client-side framework Realizuje uživatelskou část aplikace, jež je přítomna ve webovém prohlížeči. Poskytuje a sbírá data od uživatele, komunikuje se serverovou částí. Také nazýván *frontend framework*. Mezi používané frontend frameworky se řadí:

- Angular
- React
- Vue.js

Cross-functional framework Kombinuje dva výše uvedené typy frameworků v jeden. Poskytuje ucelené řešení pro vývoj backendu i frontendu. Také nazýván *full-stack framework* Příklady takovýchto frameworků:

- Symfony
- Meteor
- CodeIgniter

2.4.6 Angular

Původní frontendová aplikace Optilynx je realizovaná prostřednictvím frontendového webového frameworku Angular. Tento framework byl vytvořen společností Google a vydán v roce 2012 původně pod názvem AngularJS jako JavaScriptový framework architektury MVC (*model–view–controller*) [23]. Účelem frameworku bylo oddělit logiku aplikace od manipulace s modelem DOM a umožnit tvorbu stránek s dynamickou obnovou dat.

Takový přístup k tvorbě aplikací s dynamickou obnovou se nazývá *Single Page Application* (SPA). Je to typ webové aplikace, která většinou používá server pouze jako zdroj a úložiště dat [24]. Webový klient (prohlížeč) si při prvotní komunikaci se serverem stáhne celou aplikaci k sobě a jakákoliv další komunikace se serverem jen obstarává odesílání či přijímání samotného obsahu. Opakem tohoto přístupu je *Multiple Page Application* (MPA), což je tradičnější přístup řešení návrhu webových aplikací. Přechod mezi jednotlivými stránkami MPA aplikace pro klienta znamená znovunačtení celé aplikace.

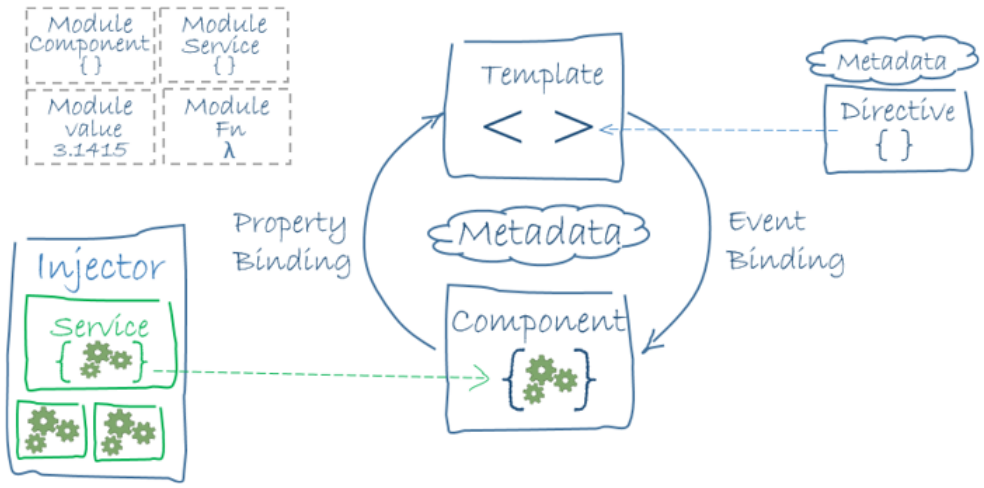
Framework AngularJS se po pár letech ukázal jako nevhodný pro budování aplikací většího rozsahu, proto došlo k jeho zásadnímu přepracování a vznikl Angular 2. Ten kromě dalších změn přinesl i jazyk TypeScript, jenž nahradil původní jazyk JavaScript jako primární programovací jazyk frameworku, a kompilování metodou AOT (*Ahead-of-Time*), které TypeScript a HTML v projektu před posláním stránky klientovi zkompiluje do JavaScriptu [25].

Nové verze frameworku Angular vycházejí pravidelně, poslední verze Angular 7 byla uvedena v roce 2018. Právě tato verze je použita ve frontendové aplikaci Optilynx.

2.4.6.1 Architektura frameworku

Základním stavebním prvkem každé Angular aplikace jsou moduly [26]. Modul sdružuje relevantní kód uvnitř komponent a služeb do jedné funkcionální kolekce, a poskytuje tak komponentám kontext pro kompilaci. Každá Angular aplikace se skládá alespoň z jednoho modulu (tzv. *root module*).

Komponenta je prvek, jenž definuje konkrétní část aplikace. Každá komponenta obsahuje definici své HTML části (pohled, anglicky *view*) vytvořením šablony (tzv. *template*), která představuje prezentaci dat a obsahu, a definici



Obrázek 2.1: Angular – ilustrace architektury aplikace [26]

své aplikační logiky ve formě TypeScriptové třídy. Součástí každé Angular aplikace je alespoň jedna komponenta (tzv. *root component*), jež propojuje hierarchii komponent s DOM modelem aplikace.

Komponenta může využívat služeb. Služba je třída, která slouží ke sdílení dat nebo aplikační logiky skrz více komponent. Využití služeb úzce souvisí s pojmem *dependency injection* (DI). DI je způsob, jakým vložit službu do komponenty jako závislost, a zpřehlednit tak kód oddělením použití logiky služeb od vlastní logiky komponenty.

2.4.7 Material Design

Při návrhu uživatelského rozhraní (UI) webové aplikace se často vyplatí využít nějaké externí knihovny či UI frameworku, jenž řeší základní problémy návrhu a poskytuje nástroje, které vývojáři umožní tvorbu pokročilejších řešení.

Material Design je návrhový jazyk vytvořený společností Google v roce 2014 [27]. Cílem Material Designu je dodat kvalitní řešení uživatelského rozhraní napříč různými platformami, které uživatelům dává kontrolu nad přehlednými a vzhledově příjemnými komponentami, jež se chováním podobají objektům ze skutečného světa. Přitom aplikuje základní a přirozená pravidla fyzického světa, především osvětlení a pohyb.

Google používá rozhraní Material Design ve všech svých aplikacích a zařízeních kvůli sjednocení uživatelského zážitku. Material Design je popsán v podrobné příručce, jež obsahuje vše od návrhových studií, přes výukové materiály, až po odkazy na komponenty připravené k použití při vývoji aplikací pro web, Android či iOS.

2.4.7.1 Angular Material

Google pro Angular vytvořil implementaci Angular Designu zvanou Angular Material [28]. Tato implementace obsahuje předpřipravené komponenty, které vývojáři mohou použít ve svých Angular projektech.

Tyto komponenty poskytují úpravy všech aspektů uživatelské interakce s webovou aplikací, včetně uspořádání obsahu, tvorby formulářových prvků, zobrazení tabulek či modálních oken. Téměř všechny interaktivní komponenty jsou také opatřeny animacemi.

Autor původní aplikace Optilynx nejdříve při návrhu uživatelského rozhraní aplikace používal knihovnu Bootstrap. V druhé fázi vývoje aplikace ovšem přešel na Angular Material. Poté, co jsem během implementace své práce měl možnost nahlédnout do fungování Angular Materialu a pochopit jej více do hloubky, jsem se i v návrhu responzivního rozhraní aplikace rozhodl použít Material Design.

Kapitola 3

Realizace mobilního a tabletového zobrazení

Náplní této kapitoly je popis samotné realizace výsledného řešení. První část kapitoly poskytuje přehled změněných částí projektu a dále se zabývá implementační částí nejdůležitějších prvků responzivního řešení aplikace. Popisuje konkrétní implementační detaily těchto prvků a nabízí i několik ukázek zdrojových kódů řešení.

Druhá část kapitoly pak představuje praktickou část této práce z hlediska uživatelského rozhraní. Popisuje změny uživatelského rozhraní v mobilním zobrazení aplikace Optilynx a vysvětluje důvody provedených kroků.

3.1 Implementace

Tato sekce se zabývá popisem implementovaných změn, jež vedly k realizaci mobilního rozhraní frontendové aplikace Optilynx.

3.1.1 Přehled implementovaných změn projektu

Během implementace svého řešení jsem přidával či upravoval komponenty, které jsou v Angular projektu uspořádány ve stromové struktuře. Takové uspořádání je spíše fyzické nežli logické – pomáhá vývojáři orientovat se v projektu a obecně ukazuje, jaké komponenty v sobě používají jiné komponenty. Může ale nastat i případ, kdy komponenta **A** použije komponentu **C** zanořenou v komponentě **B** v rámci stromové struktury.

Přehled upravených komponent vypadá následovně:

- **app** – Kořenová komponenta aplikace.
- **customers** – Komponenta obsahující seznam zákazníků včetně vyhledávání.

  - **customer-data** – Karta zákazníka s jeho údaji.
  - **customer-detail** – Komponenta obsahující kartu zákazníka a seznam zakázek zákazníka.

- **exports** – Komponenta sloužící k exportu dat.
  - **export-insurance** – Komponenta obsahující několik možností exportu dat pro pojišťovny.
  - **exports-modals** – Komponenta modálního okna pro zadání tisku faktury.

- **mobile-query** – Komponenta spravující mobilní zobrazení.
  - **mobile-query component** – Komponenta, která zjišťuje šířku obrazovky zařízení a komunikuje s *mobile-query* službou.
  - **mobile-query service** – Služba, která posílá informaci o šířce obrazovky ostatním komponentám, jež tuto informaci uplatňují k responzivnímu přizpůsobení svého obsahu.

- **navigation** – Komponenta zajišťující horní lištu aplikace a navigaci mezi obrazovkami aplikace.
  - **fab-back** – Komponenta tlačítka „zpět“.
  - **navlist** – Komponenta obsahující seznam položek hlavní nabídky.
  - **sidenav service** – Služba sloužící k předávání impulzů ke změnám stavu mobilní boční navigace.

- **orders** – Komponenta spravující seznam a přidávání zakázek včetně vyhledávání.
  - **order-detail** – Komponenta obsahující kartu zákazníka a formulář k zadání zakázky.
  - **order-vouchersModal** – Komponenta modálního okna pro přidávání či úpravu poukazů.
  - **order-products** – Komponenta spravující přidávání produktů a služeb do zakázky včetně vyhledávání.
  - **orders-list** – Komponenta, jež se stará o výpis seznamu všech zakázek.

- **organizations** – Komponenta obsahující kartu organizace s jejími údaji, a seznam poboček organizace.
  - **organizations-branch** – Komponenta karty s detaily pobočky organizace.

- **product** – Komponenta obsahující kartu produktu.
  - **product-history** – Komponenta se seznamem přesunů produktu.
- **terminal** – Komponenta pokladny sloužící k vyhledávání a přidávání produktů, služeb a zakázek do pokladny a jejich následnou platbu.
  - **terminal-cashregister** – Komponenta pokladního deníku obsahující souhrny všech platebních metod a měsíční souhrn finančních operací pro každý den v měsíci.
  - **terminal-history** – Komponenta pokladní historie se seznamem pokladních akcí a finančních informací ke každé akci.
  - **terminal-summary** – Komponenta shrnující transkace všech platebních metod mezi dvěma daty, včetně seznamu položek v každé transakci a možnosti stornování transkace.
- **warehouse** – Skladová komponenta obsahující seznam všech produktů v informačním systému včetně vyhledávání.
  - **warehouse-insert** – Komponenta naskladnění produktů včetně vyhledávání produktů i zakázkových položek.
  - **warehouse-insert-history** – Seznam historie naskladnění.
  - **warehouse-invoice** – Seznam faktur.
  - **warehouse-invoice-detail** – Komponenta detailu faktury.
  - **warehouse-invoice-pairing** – Komponenta párování faktur s dodacími listy produktů včetně vyhledávání produktů a dodacích listů.
  - **warehouse-level** – Komponenta, která obsahuje stav skladu ve formě součtu cen produktů ve skladu k danému datu a seznam takových produktů.
  - **warehouse-move** – Komponenta přesunu skladových položek mezi sklady včetně vyhledávání položek.
  - **warehouse-move-detail** – Komponenta detailu skladového přesunu obsahující seznam všech přesunutých položek.
  - **warehouse-move-history** – Komponenta historie všech skladových přesunů.
  - **warehouse-out** – Komponenta vyskladnění produktů včetně vyhledávání.
  - **warehouse-out-history** – Komponenta historie vyskladnění.
  - **warehouse-suppliers** – Komponenta seznamu všech dodavatelů v systému včetně vyhledávání.
  - **warehouse-suppliers-detail** – Karta dodavatele s jeho údaji.
  - **warehouse-taking** – Komponenta inventury skladu.

3.1.2 Media query ve frameworku

V podsekci 2.4.2.1 se zabývám aplikací CSS pravidel v závislosti na vlastnostech zařízení, jež webovou aplikaci zobrazuje, pomocí CSS pravidla media query. Při implementaci responzivních změn uživatelského rozhraní v Angular aplikaci by ovšem bylo výhodné k manipulaci s obsahem a DOM modelem, kterého běžná Angular aplikace hojně využívá, umět zjišťovat proporce daného zařízení přímo ve frameworku.

3.1.2.1 Komponenta mobile query

K účelu detekce proporcí zařízení jsem v projektu zavedl komponentu *mobile-query*. Tato komponenta má za úkol zjišťovat, zda je obrazovka zařízení užší než 900 pixelů. Toto konkrétní číslo jsem zvolil z toho důvodu, že obrazovky tabletů širší než 900 pixelů dokáží zobrazit aplikaci Optilynx v její desktopové verzi bez problémů, nebyl by tedy důvod tak široké obrazovky nutit ke zobrazení mobilní verze uživatelského rozhraní aplikace.

Komponenta naslouchá ke změnám šířky obrazovky pomocí tzv. *listeneru*. Jedná se o lambda výraz (TypeScriptovým výrazem také *fat-arrow function*), jenž obsahuje příkaz k detekci změn, a dále příkaz, který upozorní *mobile-query* službu o stavu obrazovky zařízení v případě změny. Tento stav má podobu objektu typu **MediaQueryList**, jehož členská proměnná *matches* udává, zda byla splněna požadovaná podmínka (v tomto případě, zda je šířka obrazovky menší než 900 pixelů) [29].

Aby mohla detekce změn šířky obrazovky fungovat, je nutné instanci komponenty mobile-query umístit někam do dokumentu. V případě aplikace Optilynx se tato instance nachází hned na začátku kořenové komponenty *app*. Tím dojde k instancování zjišťování šířky obrazovky před instancováním jiných komponent, a předejde se tak částečným či pozdním změnám na mobilní rozhraní v průběhu načítání aplikace.

```
1     mobileQuery: MediaQueryList;
2     private _mobileQueryListener: (e) => void;
3
4     constructor(
5         changeDetectorRef: ChangeDetectorRef,
6         media: MediaMatcher,
7         public mobileQueryService: MobileQueryService,
8     ) {
9         this.mobileQuery = media.matchMedia(
10            '(max-width: 900px)'
11        );
12        this._mobileQueryListener = (e) => {
13            changeDetectorRef.detectChanges();
14            mobileQueryService.mobileQueryChange(e);
```

```
15          };
16          this.mobileQuery.addListener(
17              this._mobileQueryListener
18          );
19      }
```

Zdrojový kód 3.1: Komponenta mobile query ke zjištění šířky obrazovky

3.1.2.2 Služba mobile query

Detekce změn šířky obrazovky by nebyla úplná, pokud by neexistoval způsob, jak o změnách notifikovat komponenty, jež tyto informace potřebují k responzivnímu uzpůsobení svého obsahu. Pro tento účel byla vytvořena služba *mobile-query service*.

Služba od komponenty mobile query dostává informace o změnách stavu šířky obrazovky. Toho je docíleno tak, že vždy při změně šířky obrazovky komponenta mobile query zavolá metodu **mobileQueryChange** služby. Tato metoda pak notifikuje o změně stavu všechny komponenty, jež tuto informaci potřebují.

Ke komunikaci s komponentami využívá služba objekt typu **Observable** [30]. Ten slouží jako notifikační zdroj, který služba obsahuje jako členskou proměnnou *mobileQueryObs$*. Ta vznikne zavoláním metody **asObservable** na objekt typu **ReplaySubject**. Je to takový typ zdroje, jenž dokáže dočasně ukládat notifikace [31]. Jednotlivé komponenty, které potřebují znát stav šířky obrazovky (konkrétně výše zmíněný parametr *matches*), naslouchají zdroji této služby přihlášením se k notifikacím zdroje skrze jeho metodu **subscribe**. Jako typ zdroje jsem zvolil ReplaySubject z toho důvodu, že komponenty se k notifikacím přihlašují postupně až v průběhu načítání aplikace.

Vždy, když se změní stav šířky obrazovky a komponenta media query zavolá metodu **mobileQueryChange** služby, vyšle ve volání metody službě i výše zmíněný stav ve formě objektu typu **MediaQueryList**. Služba pak při zavolání této metody vyšle všem komponentám (přihlášeným k notifikacím této služby) pomocí metody **next** notifikaci včetně změněného stavu.

Součástí příhlášení k naslouchání je opět lambda výraz, jenž definuje, co se stane v případě, že zdroj vydá novou notifikaci. V případě odběru notifikací od zdroje služby media query tento výraz obsahuje přiřazení zmíněného parametru *matches* objektu změněného stavu k členské proměnné dané komponenty. Komponenta pak může tento parametr dále používat jako obyčejnou proměnnou.

```
1   private mobileQuerySource = new ReplaySubject();
2
3   mobileQueryObs$ = this.mobileQuerySource
```

```
4                           .asObservable();
5
6     mobileQueryChange = (e) => {
7         this.mobileQuerySource.next(e);
8     };
```

Zdrojový kód 3.2: Služba mobile query pro notifikaci komponent

```
1     public mobileQueryMatches: boolean = false;
2     private mobileQuerySub;
3
4     this.mobileQuerySub = this.mobileQueryService
5                               .mobileQueryObs$
6                               .subscribe(
7         (e: MediaQueryList) => {
8             this.mobileQueryMatches = e.matches;
9         }
10    );
```

Zdrojový kód 3.3: Přihlášení komponenty k notifikacím služby mobile query

3.1.2.3 Použití mobile query v HTML

Pokud je komponenta přihlášena k odběru notifikací služby media query, pak její členská proměnná, do níž se přiřadí proměnná *matches* z notifikace, má vždy název *mobileQueryMatches* napříč všemi takovými komponentami v aplikaci Optilynx kvůli zachování konzistence a snažšího použití. Tato proměnná je typu **boolean** a nabývá tedy hodnot *true* nebo *false* podle toho, zda je splněna sledovaná podmínka šířky obrazovky.

Proměnnou mobileQueryMatches pak lze použít v HTML části komponenty v kombinaci s direktivami frameworku Angular, jež jsou součástí syntaxe zvané *template syntax* [32]. Lze tak např. docílit podmíněného zobrazení (nebo schování) určitého elementu pomocí direktivy *ngif*, která na vstup (v podobě hodnoty atributu) přijímá danou podmínku a volitelně i název bloku, jenž se zobrazí v případě, že uvedená podmínka nebyla splněna.

Tento *else* blok pak musí být definován uvnitř speciálního elementu **ng-template**, který musí jako svůj atribut obsahovat své jméno. V případě, že je v *ngif* direktivě jméno takového bloku v části *else* uvedeno a podmínka není splněna, na místo elementu s touto direktivou se dosadí právě vnitřek *else* bloku (elementu ng-template).

```
1     <div *ngIf="!mobileQueryMatches;
```

```
2                 else mobileProductDetail" ... >
3         <button mat-raised-button type="button" ... >
4             Detail
5         </button>
6     </div>
7
8     <ng-template #mobileProductDetail>
9         <div fxFlex="10">
10            <button mat-icon-button ... >
11                <mat-icon>info</mat-icon>
12            </button>
13        </div>
14    </ng-template>
```

Zdrojový kód 3.4: Podmíněné zobrazení HTML bloku direktivou *ngif*

Podobně pak lze podmíněně dynamicky měnit hodnoty atributů HTML elementů i jejich vnitřní obsah pomocí ternárního operátoru či dynamicky přiřazovat a odebírat CSS třídy HTML elementům pomocí speciálního atributu *ngClass*.

```
1     <div [fxFlex]="mobileQueryMatches ? 35 : 15">
2         {{mobileQueryMatches ? "" : "Zbyva doplatit:"}}
3         ...
4     </div>
5
6     <button ...
7         [ngClass]="{'full-width': mobileQueryMatches}">
8         Ulozit poukaz
9     </button>
```

Zdrojový kód 3.5: Další způsoby použití proměnné *mobileQueryMatches*

3.1.2.4 Media query jako pravidlo CSS

Vedle používání pravidla media query jakožto služby ve frameworku jsem se samozřejmě nevyhnul ani použití více „klasických“ media query pravidel v jazyce CSS. V souborech stylů aplikace se pak ve výsledku nachází několik desítek **@media** pravidel, jež porovnávají šířku obrazovky proti různým pixelovým hodnotám a patřičně aplikují různé CSS styly.

Tato pravidla nejsou definována jen pro hodnotu 900 pixelů, jak tomu je v případě pravidla v komponentě mobile query, ale pro různé hodnoty od 375 pixelů až po 900 pixelů z důvodu jemnější kontroly nad responzivním chováním různých prvků. Lze tak docílit úprav, které jsou specifické pro zařízení různých

šířek, a tedy i více detailního rozlišení prvků mobilního a tabletového zobrazení aplikace.

```
1     .card__edit {
2         position: absolute;
3         top: 20px;
4         right: 20px;
5         @media (max-width: 375px) {
6             top: -10px;
7             right: -10px;
8         }
9     }
10
11    .item-list__item--price {
12        @media (min-width: 901px) {
13            div:first-child {
14                text-overflow: ellipsis;
15                white-space: nowrap;
16                overflow: hidden;
17                padding-right: 20px;
18            }
19        }
20        @media (max-width: 900px) {
21            div:first-child {
22                word-break: break-word;
23            }
24        }
25        ...
26    }
```

Zdrojový kód 3.6: Příklady media query pravidel v jazyce preprocesoru LESS

3.1.3 Postranní navigace

Jedním z problémů, jež jsem si vytyčil vyřešit v sekci 1.6, byla hlavní nabídka aplikace a skladová podnabídka. Mou představou bylo sbalit tyto dvě nabídky do jedné postranní nabídky, která by se vysouvala stisknutím příslušného tlačítka. Za tímto účelem jsem použil předpřipravenou komponentu z Angular Materialu zvanou **sidenav**.

Struktura použití této komponenty v dokumentu je následující: HTML element **mat-sidenav-container** obklopuje dva *child* elementy – **mat-sidenav** a **mat-sidenav-content**. Element mat-sidenav představuje samotnou vysouvací nabídku a obsahuje element **mat-nav-list** se samotnými položkami nabídky. Hodnoty jeho atributů pak nastavují specifické nastavení jako polohu vysouvací nabídky či metody, jež se spustí při změně stavu vysouvací

nabídky. Element mat-sidenav-content pak obklopuje obsah, přes který se vysouvací nabídka zobrazí v případě, že bude otevřena.

```
1     <app-navigation></app-navigation>
2
3     <mat-sidenav-container>
4         <mat-sidenav #snav mode="over" fixedTopGap="56"
5                      fixedInViewport
6                      (openedStart)="snavOpenedChange()"
7                      (closedStart)="snavOpenedChange()">
8             <mat-nav-list>
9                 <app-navlist ...></app-navlist>
10            </mat-nav-list>
11        </mat-sidenav>
12        <mat-sidenav-content>
13            <div class="container" ...>
14                <div class="content">
15                    <main>
16                        <router-outlet></router-outlet>
17                    </main>
18                </div>
19            </div>
20        </mat-sidenav-content>
21    </mat-sidenav-container>
```

Zdrojový kód 3.7: HTML implementace postranní nabídky

Element mat-sidenav-container je v aplikaci Optilynx umístěn v kořenové komponentě *app*. K otevírání a zavírání nabídky ovšem slouží tlačítko umístěné v komponentě *navigation*. Vysílání impulzů k ovládání vysouvání nabídky je řízeno službou *sidenav*.

Tato služba obsahuje čtyři různé zdroje typu **Observable**. Tyto zdroje slouží k posílání notifikací k ovládání vysouvací nabídky, jeden zdroj pak notifikuje o tom, zda je nabídka otevřená, či nikoli. Při stisknutí tlačítka k ovládání vysouvací nabídky se zavolá metoda **toggleSnav** komponenty navigation. Metoda pak zavolá stejnojmennou metodu služby sidenav.

Metoda služby vyšle notifikaci kořenové komponentě, jež je, podobně jako v případě služby mobile-query, přihlášena k odběru notifikací zdrojů (sloužících k obsluze vysouvací nabídky) služby sidenav. Komponenta pak dle příslušného lambda výrazu otevře či zavře vysouvací nabídku.

```
1     @ViewChild('snav') snav;
2     private snavToggleSub;
3
4     this.snavToggleSub = this.sidenavService.toggleSnav$
```

```
5                               .subscribe(
6         () => this.snav.toggle()
7     );
8
9     public snavOpenedChange() {
10        this.sidenavService.snavOpened(this.snav.opened);
11    }
```

Zdrojový kód 3.8: Ukázka obsluhy postranní nabídky v komponentě *app*

Současně se změnou stavu vysouvací nabídky se zavolá i metoda **snavOpenedChange**, což je definováno v atributech elementu mat-sidenav. Tato metoda obratem zavolá metodu **snavOpened** služby sidenav včetně předání parametru *snav.opened* postranní nabídky, který indikuje stav vysouvací nabídky. Služba v metodě vyšle notifikaci odběratelům a do notifikace opět předá parametr opened.

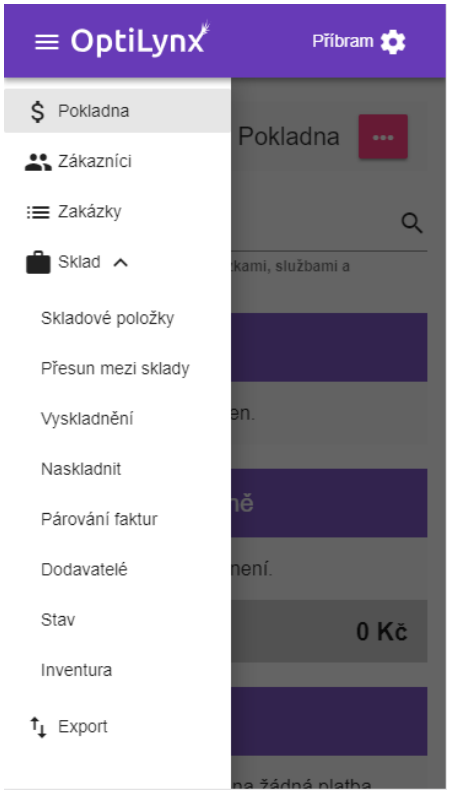
Konkrétním odběratelem v implementaci této práce je komponenta *fab-back*, jež obsahuje tlačítko „zpět“. Při změně stavu vysouvací nabídky se spustí animace, která tlačítko posouvá buď doleva, nebo doprava. Nutnost takto animovat tlačítko vznikla, když jsem zjistil, že otevřená postranní nabídka nepřekryje tlačítko fab-back, jelikož je toto tlačítko umístěno v komponentě navigation, jež není obklopena elementem mat-sidenav-content.

```
1     animations: [
2         trigger('animState', [
3             state('right', style({
4                 transform: 'translateX(0px)',
5             })),
6             state('left', style({
7                 transform: 'translateX(-300px)',
8             })),
9             transition('* => *', animate('350ms ease')),
10        ])
11    ]
```

Zdrojový kód 3.9: Zápis animace tlačítka *fab-back*

3.2 Popis mobilního uživatelského rozhraní

Obsahem této sekce je popis tvorby stěžejních prvků mobilního zobrazení frontendové aplikace Optilynx, doplněný o ukázky zobrazení ve formě ilustrací. Je důležité zmínit, že popisované změny se týkají pouze menších zařízení; původní zobrazení pro širší obrazovky bylo zachováno.



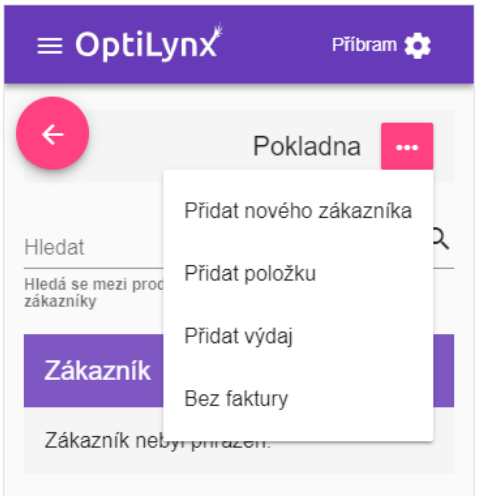
Obrázek 3.1: Mobilní zobrazení Optilynx – vysouvací nabídka

3.2.1 Vysouvací nabídka

Jak už bylo předestřeno, v mobilním zobrazení frontendové aplikace Optilynx figuruje postranní vysouvací nabídka. Ta nahrazuje jak hlavní nabídku umístěnou v horní liště, tak i skladovou podnabídku obsahující odkazy na obrazovky skladových operací.

Vysouvací nabídku lze otevřít tlačítkem v levé části horní lišty. Pro toto tlačítko jsem zvolil ikonu nabídky (viditelnou na obrázku 3.1), jež je v dnešní době typická pro účely indikace tlačítka hlavní nabídky. Postranní nabídku také lze otevřít přejetím prstu z levé části obrazovky.

Samotná vysouvací nabídka zachovává pořadí položek hlavní nabídky. Položky skladové podnabídky se nacházejí pod položkou **Sklad**. Po klepnutí na tuto položku se rozbalí skladová podnabídka doprovázená rozbalovací animací. V nabídce je světlešedou barvou zvýrazněna aktuální obrazovka. Vysouvací nabídku lze zavřít opětovným klepnutím na tlačítko nabídky či klepnutím do zašedivělého prostoru mimo nabídku.



Obrázek 3.2: Mobilní zobrazení Optilynx – rozbalovací nabídka akcí obrazovky

3.2.2 Akční tlačítka obrazovek

V původním zobrazení se na většině obrazovkách aplikace u hlavního nadpisu obrazovky nacházejí i tlačítka, jež představují nějaké akce specifické k dané obrazovce (např. v obrazovce pokladny tlačítka „Přidat položku“, „Přidat výdaj“ a „Přidat nového zákazníka“). Tato tlačítka jsou ovšem především pro obrazovky menších zařízení moc velká a obvykle zastřešují vedlejší akce, které uživatel běžně nevyužívá.

Proto jsem tato akční tlačítka napříč celou aplikací postupně nahradil vždy jedním tlačítkem. V takovém případě jsem řešil dvě situace, jež mohly nastat: stránka má jen jedno horní akční tlačítko nebo více tlačítek.

V případě jednoho tlačítka jsem ono tlačítko učinil kompaktnějším vynecháním textového popisu. Většina takových tlačítek v aplikaci totiž má kromě textového popisu i ikonu, která stručně indikuje akci, již dané tlačítko vykonává. Vynecháním textového popisu jsem tak vsázel na to, že jsou uvedené ikony dostatečně popisné, což dle mého názoru bylo v testování následně i potvrzeno.

Více takových akčních tlačítek jsem pak vždy sbalil pod jedno tlačítko, které po klepnutí otevře rozbalovací nabídku s položkami představujícími akce původních akčních tlačítek. Současně jsem zachoval i vizuální styl obou typů tlačítek k udržení konzistentního uživatelského požitku (*user experience*, UX).

3.2.3 Tabulky Angular Material

Na několika místech v aplikaci jsou k prezentaci dat použity tabulky z knihovny Angular Material. Jedná se např. o seznam zákazníků, historie skladových přesunů či seznam dodavatelů. Tyto tabulky byly vzhledem k počtu

sloupců obvykle moc široké pro menší obrazovky – na ty nejmenší obrazovky by se vedle sebe vešly nejvýše 3 až 4 sloupce – proto bylo nutno nějakým způsobem řešit jejich zobrazení na mobilních zařízeních.

Za tímto účelem jsem se nechal inspirovat řešením tabulek v mobilním rozhraní služby Fakturoid (viz. 1.7.2.1). K vyřešení tabulek jsem tak použil redukci údajů, kombinaci více sloupců dohromady a různé velikosti fontů k rozlišení informací. Každá tabulka přitom vyžadovala specifické řešení kvůli různých zobrazovaných dat.

Jako názornou tabulku k popisu lze zvolit např. seznam skladových přesunů v informačním systému. Tabulka klasického zobrazení obsahuje sloupce pro datum přesunu, zdrojový sklad, cílový sklad, počet kusů přesunutých produktů a název uživatele, který přesun zadal. Posledním sloupcem je speciální sloupec s tlačítkem **Detail**, jež uživateli zobrazí obrazovku detailu přesunu.

Pro mobilní zobrazení jsem následně vytvořil tabulku, která obsahuje pouze dva sloupce: **Z – do** | **datum** a **Ks** | **uživatel**. První sloupec kombinuje původní tři první sloupce, druhý sloupec kombinuje další dva a tlačítko detailu je nahrazeno funkcionalitou, kdy celý řádek slouží jako klikatelný odkaz na detail přesunu. Každá buňka sloupce pak efektivně obsahuje dva řádky, z nichž horní má větší velikost písma a spodní má menší velikost.

Původní tabulka skladových přesunů je, podobně jako i jiné tabulky v systému, ve výchozím stavu řazena podle sloupce data sestupně. Toho je docíleno pouhým nastavením atributu aktivního sloupce v definici tabulky v HTML části komponenty. Takové nastavení ale nefunguje pro sloupce, jejichž obsahem buňky není pouze samotná hodnota buňky, ale více hodnot či nějaký HTML zápis, jelikož pak není pro tabulku zřejmé, podle čeho řadit.

Správné řazení složených sloupců tabulky lze zajistit definicí způsobu, jakým má řazení tabulky přistupovat k datům sloupců. Toho lze dosáhnout úpravou vlastnosti **sortingDataAccessor** zdroje dat tabulky. Jedná se o funkci, jež definuje, k jakým datům sloupce má řadící funkce během řazení přistupovat [33]. Díky tomu jsem byl schopen dodat funkcionalitu řazení prvního sloupce dle data a druhého sloupce dle uživatele.

V několika tabulkách původního zobrazení informačního systému Optilynx se nachází sloupce, které obsahují více než jedno tlačítko akcí pro daný řádek, z nichž jedno je vždy tlačítko detailu. V takovém případě bylo třeba v rámci mobilního zobrazení zajistit, aby se všechna tlačítka (vyjma tlačítka deftailu) nacházela v obrazovce detailu. Toto je jeden z kompromisů návrhu mobilního zobrazení, jež podporují zachování plné funkcionality aplikace.

3.2.4 Výpis vyhledávání

Na obrazovce pokladny a na několika dalších obrazovkách se nacházejí varianty výpisu výsledků vyhledávání. V případě pokladny vyhledávací pole prohledává všechny produkty, služby, zakázky a zákazníky v informačním systému. Výpis

výsledků tohoto vyhledávání pak obsahuje nejdůležitější informace o vyhledané položce, případně i s tlačítkem s odkazem na detail dané položky nebo dalšími tlačítky. Tento výpis je pro menší obrazovky příliš robustní, a proto jej bylo třeba upravit.

Tlačítka na řádcích byla nahrazena odpovídajícími ikonami. Došlo k redukci nepodstatných informací (např. kód služby, informace o komisi produktu) či opakujících se výrazů (např. **Skladem**, **Cena**, **Zbývá doplatit**). Verbální stavy zakázek byly taktéž nahrazeny ikonami, nad touto změnou ale měli uživatelé rozpaky během uživatelského testování. Jsem však přesvědčen, že se jedna o sílu zvyku a že je toto řešení správné.

3.2.5 Posuvné tabulky souhrnů

Obrazovky přehledu pokladny a pokladního deníku obsahují tabulky souhrnů transakcí, ve kterých je nutné zachovat všechny sloupce. Proto nelze aplikovat způsob úpravy tabulek popsaný v podsekci 3.2.3 – redukci či kombinaci sloupců. Pro tento problém jsem tedy zvolil řešení vertikálně posuvné tabulky.

Pro takové tabulky jsou nastaveny pevné hodnoty minimální šířky. Pro menší obrazovky to tak znamená, že přetékající obsah tabulky je schován a tabulka se stává vertikálně posuvnou. V takové situaci by ovšem pro uživatele nemuselo být zřejmé, že tabulka je posuvná. Proto jsem k tabulce implementoval i indikátor posouvání. Tento indikátor v podobě šipky vpravo je ukázán, pokud je tabulka posuvná, a skryje se v případě, že uživatel posunul tabulku úplně vpravo.

Pro tento problém existuje více řešení. Vzhledem k malému počtu řádků původní tabulky by šlo např. tabulku transponovat, tedy zaměnit sloupce za řádky, čímž by se tabulka vešla celá i na nejmenší obrazovky. Provedením tohoto řešení pro každou takovou tabulku v obrazovce by ovšem vznikla dlouhá stránka, již by uživatel byl nucen přejíždět. Toto by se dalo optimalizovat schováním tabulek do rozbalovacích kontejnerů, které by ve výchozím stavu byly sbalené a uživatel by si je rozbalil jen v případě potřeby. Mým řešením jsem ale chtěl poskytnout uživateli okamžitý náhled alespoň na část dat tabulky.

3.2.6 Výpisy produktů

Na několika obrazovkách aplikace se vyskytují varianty seznamu produktů vložených uživatelem. Jedná se např. o produkty vložené do pokladny k prodeji či produkty určené ke skladovému přesunu. Takový seznam pak obvykle obsahuje název produktu, cenu a pole pro zadání počtu kusů, případně ceny nebo slevy. Jako poslední se na řádku produktu vždy nachází tlačítko k odebrání produktu z daného seznamu.

Na obrazovkách menších zařízení tyto prvky položek v seznamu začnou různě přetékat a vzájemně se překrývat (viz. 1.6). Cenu produktu (či jinou

informaci, např. kód EAN v případě skladového přesunu) jsem přeuspořádal pod název produktu. Zobrazení vícero zadávacích polí na řádku jsem vyřešil následujícím způsobem: ve výchozím stavu jsou tato pole skryta a nahrazena tlačítkem s ikonou. Po klepnutí na toto tlačítko se všechny prvky na řádku produktu nahradí zadávacími poli, které může uživatel vyplnit a vrátit se zpět do původního stavu řádku klepnutím na tlačítko s ikonou fajfky.

Ke zprovoznění této funkcionality bylo třeba upravit datové modely výpisu produktů. Datový model je objekt, který se používá k zachycení dat přijímaných skrze API a který lze použít ke zjednodušení ukládání přijímaných dat do kolekcí uvnitř frameworku [34]. Do těchto modelů jsem tedy přidal atribut **edit**, který slouží jako proměnná typu *boolean* a indikuje to, zda řádek daného produktu v seznamu je ve výchozím stavu či v alternativním stavu se zadávacími poli.

3.2.7 Seznam zakázek

Seznam zakázek v původním zobrazení obsahuje přehledové karty zakázek ve formě hlavičky zakázky se základními informacemi o zakázce, včetně jména zákazníka zakázky, stavu zakázky a ceny, spolu s tlačítky **Detail** a **Výdej**. Kromě slovy vypsaného stavu zakázky je hlavička karty podbarvená podle daného stavu. Také se v kartě vyskytuje tabulka čoček zakázky a případně i tabulky produktů a služeb v zakázce. Takové množství informací ovšem na menších obrazovkách také způsobuje problémy s přetékáním.

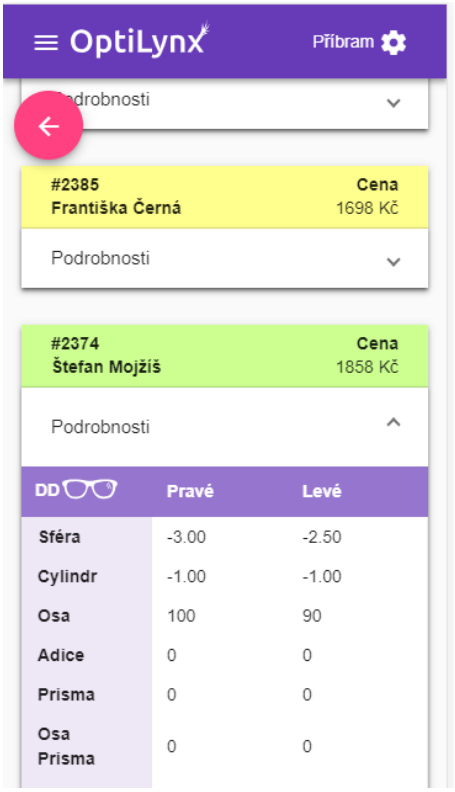
Většina informací v hlavičce zakázky se nachází i v jejím detailu a není potřebná k samotné identifikaci zakázky v seznamu zakázek. Proto v mobilním zobrazení byly tyto informace vynechány a naopak byly ponechány jen informace o čísle zakázky, jméně zákazníka a ceně zakázky. Podbarvení hlavičky samotné pak slouží k identifikaci stavu zakázky. V mobilním zobrazení slouží hlavička také jako odkaz na detail zakázky.

Tabulky čoček, produktů a služeb zakázky byly schovány do rozbalovacích kontejnerů, jež jsou ve výchozím stavu sbalené. Tabulka čoček pak byla navíc transponována, vleze se tedy šířkou i na nejmenší zařízení.

3.2.8 Detail zakázky

Tvorba či úprava samotné zakázky probíhá v jejím detailu. Tato obrazovka poskytuje možnosti úpravy čoček, obruby, přidávání dalších produktů a služeb, úpravu cen položek a slev a přidávání poukazů, spolu s rekapitulací celé zakázky.

Zadávání detailů čoček je v původním zobrazení představováno upravitelnou tabulkou, podobnou tabulce čoček u zakázky v seznamu zakázek, ovšem rozšířenou o další detaily čoček. V mobilním zobrazení také došlo k transponování této tabulky. Z tabulky jsem vyjmul formulářová pole pro upřesnění skel a obrubu, a to především z estetického hlediska. V případě tabulky a for-



Obrázek 3.3: Mobilní zobrazení Optilynx – seznam zakázek

mulářových prvků pro upřesnění došlo k jejich uspořádání pod rozbalovací kontejner, jenž je ve výchozím stavu rozbalený. Další formulářové prvky mimo tabulku čoček byly uspořádány do sloupce.

Výpisy položek v zakázce (ceny čoček, produkty, služby) dostály změn popsaných v podsekci 3.2.6. Formulářové prvky v okně přidávání nového poukazu byly taktéž uspořádány do sloupce. Jednotlivé prvky rekapitulace zakázky byly kvůli lepší přehlednosti přeuspořádány do dvou sloupců.

3.2.9 Údajové karty

Ve frontendové aplikaci informačního systému Optilynx se na několika obrazovkách vyskytují údajové karty. Tyto karty obsahují informace o dané položce (produktu, zákazníkovi, dodavateli) a zároveň jsou implementovány jako formuláře sloužící k úpravě těchto informací. Ve výsledku mají tedy prvky informací v kartě a prvky editačního formuláře karty stejné rozložení.

Toto rozložení v původním zobrazení strukturuje prvky do několika sloupců. Taková struktura na obrazovkách menších zařízení způsobuje přetékání prvků či jejich schovávání. Proto došlo k přeuspořádání prvků do jednoho či do dvou sloupců (v případě údajové karty produktu). Toto

uspořádání nijak zásadně neprodlužuje výslednou stránku a slouží i ke zpřehlednění informací v kartě.

Na nejmenších obrazovkách také došlo k přemístění tlačítka pro přepínání editačního a informativního stavu karty na pravý horní roh. Předchází se tak situaci, kdy je delší jméno na menší obrazovce překryto tímto tlačítkem.

Kapitola 4

Testování

Implementovaný prototyp responzivního řešení bylo třeba otestovat. Právě to je obsahem této kapitoly. První sekce kapitoly se zabývá předběžným uživatelským testováním realizovaného prototypu. Odhalí základní nedostatky mobilního zobrazení z pohledu uživatele nezasvěceného do problematiky prodeje optických pomůcek.

Po zpracování připomínek z předběžného testování do prototypu mobilního zobrazení se druhá sekce kapitoly zaměří na otestování prototypu na reálných uživatelích desktopové verze systému na dvou pobočkách optiky. Výsledky tohoto testování jsou zhodnoceny, a na závěr kapitoly jsou popsány případné změny mobilního zobrazení po tomto testování.

4.1 Předběžné uživatelské testování použitelnosti řešení

V půlce dubna byla hotova většina implementace prototypu frontendu. Ačkoliv jedním z cílů této bakalářské práce bylo otestovat prototyp na reálných uživatelích desktopové verze aplikace Optilynx, rozhodl jsem se před tímto testováním provést ještě předběžné testování použitelnosti svého řešení, abych odhalil základní nedostatky responzivního návrhu a otestoval jeho intuitivnost. Proto jsem si vybral dva studenty, kteří tyto vlastnosti otestují z pohledu uživatele, jenž nemá s prací v oblasti optiky žádné zkušenosti. Takový typ uživatele se pak zaměřuje na jiné problémy, než znalý pracovník optiky.

Pro účely předběžného testování jsem vytvořil několik testovacích případů, které měly vést k otestování většiny základních prvků mobilního zobrazení. Náplní těchto testovacích případů bylo:

- obsluha pokladny – přidání produktu do pokladny, prodej, aplikování slev, odebírání položek, výdej zakázky
- tvorba zákazníka

- operace s produkty – přidání do systému, naskladnění, přesun
- přidávání dodavatelů
- export dat pro pojišťovny

Tyto testovací případy sloužily jako inspirace k pozdější tvorbě plnohodnotného testovacího scénáře pro pracovníky optiky.

4.1.1 Vyhodnocení předběžného testování

Prototyp mobilního zobrazení se ukázal býti poměrně intuitivním pro neznalé uživatele. V systému se zorientovali rychle a zadané úkoly z testovacích případů řešili rychle a případné zpomalení v řešení úkolu ústilo spíše z neznalosti systému než z neintuitivního návrhu prototypu.

Testeři však našli několik připomínek ke konkrétním detailům návrhu. První připomínka byla ke vzhledu ikony tlačítka pro zadání slevy produktu v pokladně a souvisle s tím i k dalším tlačítkům pro změnu stavu řádků v produktových výpisech. Černobílá ikona se pro testery zdála být spíše jakýmsi indikátorem, a až po vyzkoušení ostatních prvků na obrazovce došli úspěšně k závěru, že ikona představuje tlačítko pro přepnutí stavu produktu k pokladně pro zadání slevy produktu. Po předběžném testování byla tato chyba napravena sjednocením stylu tlačítka s ostatními barevnými kruhovými tlačítky v systému.

Další připomínka se týkala polohy tlačítka pro přepínání informativního a editačního stavu u tvorby zákazníka, produktu a dodavatele. Padl návrh přemístit v mobilním zobrazení tlačítko na konec karty nebo mít dvě tlačítka na obou pozicích. Tento návrh jsem však zamítl s odůvodněním, že karty zákazníků, produktů a dodavatelů nejsou tak dlouhé, aby poloha přepínacího tlačítka nahoře ubírala na použitelnosti návrhu.

Také byla v předběžném testování objevena chyba v zobrazení tabulek produktů a služeb u zakázky v seznamu zakázek. Tato chyba byla opravena přeuspořádáním zmíněných tabulek pod sebe.

4.2 Testování na reálných uživatelích informačního systému

Koncem dubna jsem vyrazil na uživatelské testování svého prototypu mobilního rozhraní na dvě pobočky optiky do Benešova a Sedlčan. Cílem tohoto testování kromě ověření obecné použitelnosti mobilního zobrazení bylo i ověřit použitelnost zobrazení v rámci typických procesů optiky, pro něž je informační systém Optilynx využíván.

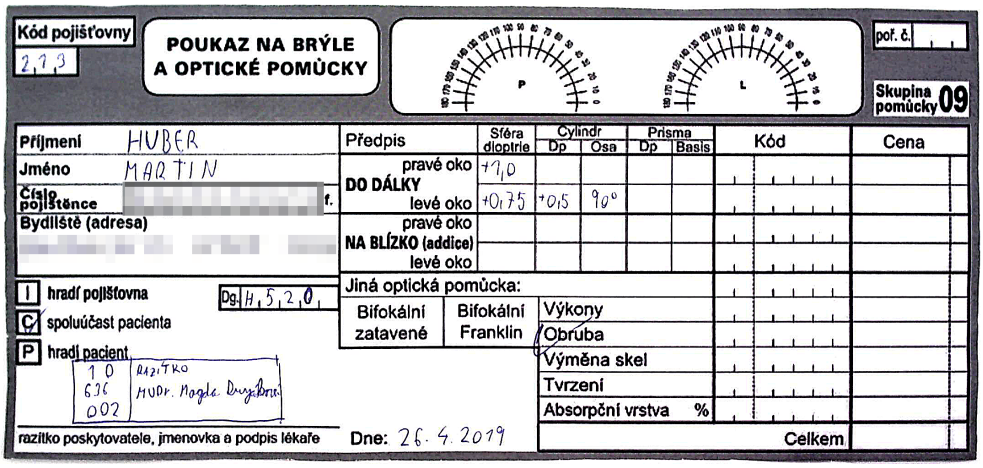
K otestování potřebných částí responzivního uživatelského rozhraní byl vytvořen testovací scénář, který má simulovat několik důležitých procesů v optice. Testovací případy tohoto scénáře, zkráceny o konkrétní zadaná testovací data, byly následující (vysvětlivka účelu testovacího případu je v závorce):

- **Případ 1** – Prodejte zákazníkovi zadané produkty. Na produkty přidejte zadané slevy.
- **Případ 2** – Prodejte zákazníkovi zadanou položku. Položka není na skladě a není třeba ji naskladňovat (test přístupu k možnosti v horní podnabídce pokladny).
- **Případ 3** – Prodejte 100 kusů zadaného produktu zadanému zákazníkovi na fakturu.
- **Případ 4** – Prodejte zákazníkovi zadaný produkt (test stornování pokladny pozdějším odmítnutím platby).
- **Případ 5** – Vytvořte novou zakázku podle poukazu zákazníka.
- **Případ 6** – Naskladněte položky vytvořené zakázky.
- **Případ 7** – Vydejte připravenou zakázku zákazníkovi.
- **Případ 8** – Naskladněte nový produkt a spárujte jeho dodací list s fakturou dle zadaných údajů.
- **Případ 9** – Zjistěte stav naskladnění zadaného produktu na skladě.
- **Případ 10** – Přesuňte zadaný produkt na zadaný sklad.
- **Případ 11** – Stornujte doklad zadané transakce.
- **Případ 12** – Přidejte do systému zadaného dodavatele.

Pro účely testovacího případu č. 5 jsem si navíc připravil i poukaz na brýle a optické pomůcky (obr. 4.1), jenž jsem vyplnil tak, aby údaje pro levou čočku byly odlišné od údajů pro pravou čočku. Součástí tohoto testovacího případu totiž bylo i vyplnění údajů čoček brýlí. Tabulka čoček v původním zobrazení obsahuje údaje pro pravou čočku nahoře a pro levou čočku dole (podle způsobu, jakým jsou informace o čočkách zapsány v poukazu).

Vzhledem k tomu, že jsem tabulku čoček pro mobilní zobrazení transponoval, vyskytly se údaje pro pravou čočku v levém sloupci a údaje pro levou čočku v pravém sloupci. Během testování s pracovníky optik jsem chtěl otestovat i způsob, jakým by vyplňovali údaje takto transponované tabulky. Proto jsem v rámci testování záměrně skryl pojmenování sloupců levé a pravé čočky.

K testování byl použit telefon LG G3. Během testování byla nahráván záznam obrazovky telefonu, což posloužilo jako materiál ke zpětnému vyhodnocení testování.



Obrázek 4.1: Poukaz použitý při testování

4.2.1 Oční optika Benešov

První testování s reálnými uživateli aplikace Optilynx proběhlo na pobočce optiky v Benešově. Testování se účastnily dvě pracovnice na pobočce: Šárka Martínková a Věra Chmúrová, odpovědná vedoucí pobočky. Z důvodu nedostatku času na dvě samostatná testování se obě zaměstnankyně střídaly v plnění úkolů jednoho testovacího scénáře.

4.2.1.1 Průběh testování

Prodej a zadávání slevy produktu v pokladně proběhl bez problémů. Věra Chmúrová během pár okamžiků zjistila, že barevné tlačítko u jména produktu jí dovolí zadat slevu. V případě přidání neskladové položky si vzpomněla, že na počítačové verzi k tomuto účelu je tlačítko. Poté ihned klepnula na horní tlačítko podnabídky a našla přidání položky.

Šárka Martínková řešila prodej kusů na fakturu přes tvorbu zakázky. Takový způsob mě zaskočil, jelikož mě vůbec při tvorbě testovacího scénáře nenapadl. Počítal jsem se způsobem prodeje na fakturu přes pokladnu, ale toto byla v nasazené desktopové verzi teprve týden stará funkcionalita. Po připomenutí kolegyní Věrou se ale Šárka úspěšně dobrala k zamýšlenému způsobu.

Věra Chmúrová se chopila tvorby zakázky dle poukazu. Nejdříve mě jako nového zákazníka přidala bez problémů do systému a následně začala vyplňovat detaily čoček zakázky. Zarazilo ji, že sloupce levé a pravé čočky jsou nepojmenované, a rozhodla se tedy vyplňovat je standardním způsobem – nejdříve pravou čočku, pak levou.

Po vyplnění čoček ovšem omylem klepnula na tlačítko zpět, jež jsem pro účely testování přemístil z levého horního rohu do levého dolního rohu. Na tento krok si ovšem paní Věra několikrát za testování stěžovala, protože jí

tlačítko zavázelo ve vyplňování zadávacích polí. Po opětovném vyplnění čoček a dalších detailů zakázky paní Věra následně přidala a vyplnila i nový poukaz. Problém jí nedělala ani platba zálohy zakázky. Úspěšně paní Věra i naskladnila položky zakázky a zakázku vydala.

Při testování naskladnění produktu se paní Věra nejdříve podívala, zda již produkt v systému existuje, a pak se pustila do tvorby nového produktu a naskladnění. Při vyplňování ceny za kus u naskladnění už automaticky použila tlačítko k přepnutí editačního stavu řádku produktu. Následné spárování dodacího listu s fakturou taky proběhlo hladce. V obrazovce vytvořené faktury jsem si ovšem všiml jednoho nedostatku: pokud jsou vyplněny údaje produktu, ale ne jeho název, vytvoří se ten název automaticky z vyplněných údajů. V tabulce produktů na faktuře pak toto jméno bylo na obrazovce menšího zařízení moc dlouhé, a tím docházelo k jeho zalamování a k rozšiřování řádku.

Poté, co standardním způsobem paní Věra splnila testovací případ skladového přesunu, jsem objevil chybu v automatickém řazení tabulky historie skladového přesunu – sloupec ikonou šipky ukazoval, že je seřazený, řazení ovšem proběhlo až po opětovném klepnutí na hlavičku sloupce. Testování paní Věra zakončila úspěšným splněním testovacích případů storna dokladu a zadání nového dodavatele.

4.2.1.2 Vyhodnocení testování

Toto testování dopadlo lépe, než jsem čekal. Obě zaměstnankyně se v mobilním uživatelském rozhraní systému orientovaly s přehledem, čemuž napomáhalo také to, že jsem se v návrhu mobilního zobrazení snažil co nejvíce zachovat procesy fungování informačního systému.

Hlavní výhrady k použitelnosti se našly především k poloze tlačítka „zpět“ a k chybě automatického řazení tabulky historie skladových přesunů. Obě tyto připomínky, spolu s nedostatkem zalamování jména produktu na faktuře, byly zapracovány do výsledné implementace.

4.2.2 Oční optika Sedlčany

Na pobočce optiky v Sedlčanech proběhlo druhé testování reálných uživatelů. Do testování se zapojily obě pracovnice na místě: Jana Blandová a Lenka Mašková, odpovědná vedoucí pobočky. Obě pracovnice se z časových důvodů také střídaly v plnění testovacích případů jednoho scénáře.

4.2.2.1 Průběh testování

Obsluhy pokladny se v testování ujala paní Blandová. Pro přidání slevy ihned použila tlačítka k přepínání stavu, zarazilo ji ovšem, že se cena po zadání slevy aktualizuje jen ve stavu zadávání ceny, ne ve výchozím stavu řádku. Lenka Mašková pak po chvilkovém zmatení využila tlačítka podnabídky pokladny k přidání neskladové položky a úspěšně ji prodala.

Testovací případy prodeje na fakturu a storna pokladny proběhly standardním způsobem. Při tvorbě zakázky podle poukazu se paní Blandová snažila dostat k přehledu zákazníků. Po chvilce snažení se jí úspěšně podařilo zvládnout ovládání boční nabídky, a nakonec se rozhodla najít zákazníka přes pokladnu. Zákazník byl již v systému vytvořen, proto se pustila rovnou do tvorby zakázky.

Při načítání údajů zákazníka do zakázky bylo ovšem přerušeno internetové spojení, tudíž aplikace musela být načtena znovu. U vyplňování čoček zakázky si pak paní Blandová také všimla, že v hlavičce tabulky čoček chybí rozlišení pravé a levé čočky. Začala tedy také vyplňovat levý sloupec jako pravou čočku a pravý sloupec jako levou čočku. Platba zálohy pak už probíhala podobným způsobem jako v prvním testování v Benešově.

Naskladňování zakázky působilo paní Maškové mírné obtíže. Zakázku v obrazovce naskladnění chtěla vyhledat podle jména zákazníka zakázky, ovšem ve výsledcích se zobrazovaly obě zakázky (včetně zakázky vytvořené a vydané v prvním testování). Paní Mašková se nemohla rozhodnout, kterou z nich zvolit, ani s pomocí ikon signalizujících stav zakázky. Nakonec si podle detailů zakázky zapsala její číslo a podle něj pak zvolila správnou zakázku v obrazovce naskladnění. Vydání zakázky nečinilo žádné potíže.

Ostatní testovací úkoly až po přidání nového dodavatele do systému proběhly bez problémů. Paní Mašková při přidávání dodavatele jej nejdříve chtěla vyhledat, poté si ale chvíli nevěděla rady, když zjistila, že v systému není, a nemohla najít tlačítko pro přidání dodavatele. Nakonec ale úspěšně zjistila, že zjednodušené tlačítko s ikonou slouží právě k tomuto účelu. Zbytek testovacího scénáře se už nepotkal se zádrhelem.

4.2.2.2 Vyhodnocení testování

Druhé uživatelské testování v Sedlčanech také hodnotím jako úspěšné. Jediné prvky mobilního zobrazení, jež pracovnice pobočky označily za problémové, byly taktéž poloha tlačítka „zpět“, dále neznámé ikony stavů zakázky a připomínka k ceně produku se slevou v pokladně. První připomínka se vyskytla i v testování v Benešově a byla, spolu s připomínkou k ceně produktu v pokladně, zpracována do finální verze mobilního zobrazení. Připomínku o ikonách stavů zakázek ovšem připisuji spíše neseznámením se s těmito ikonami.

4.2.3 Souhrn výsledků testování a implementovaných změn

Testování prototypu mobilního zobrazení aplikace Optilynx na reálných uživatelích na pobočkách optiky potvrdilo použitelnost prototypu. Uživatelé se ve většině případů dokázali seznámit s novými mobilními prvky systému v krátkém čase. Hlavními poznatky z těchto dvou testování byly detaily pro-

cesů, jež doprovází používání systému Optilynx uživateli obeznámenými s problematikou práce v optice.

Během těchto dvou testování se také potvrdila má hypotéza, že uživatel bude při tvorbě zakázky vyplňovat levý sloupec tabulky čoček údaji pravé čočky a naopak.

Po provedení a vyhodnocení obou testování byly připomínky z testování promítnuty do finální podoby mobilního zobrazení. Tlačítko „zpět“ bylo přesunuto zpět na svou původní pozici v levém horním rohu obrazovky.

Chyba v automatickém řazení tabulek byla způsobena špatným umístěním definice vlastnosti **sortingDataAccessor** datového zdroje pro mobilní tabulku (viz. 3.2.3) ve třídě komponenty tabulky. Pro mobilní tabulku bylo zároveň třeba dodefinovat automatické řazení mimo HTML tabulky i ve třídě.

V seznamu produktů na faktuře došlo k přemístění jména produktu na samostatný řádek vytvořením nového rozvržení jak hlavičky tabulky, tak i samotné buňky tabulky. Zobrazení ceny v pokladně u produktu bylo upraveno tak, aby odráželo cenu po zadané slevě.

Finální podoba mobilního zobrazení aplikace Optilynx po úpravách prototypu po uživatelském testování tak poskytuje robustní řešení uživatelského rozhraní aplikace nejen pro desktopové obrazovky, ale i pro moderní mobilní telefony a tablety.

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat mobilní a tabletové zobrazení frontendové aplikace informačního systému pro optiky zvaného Optilynx. Byla provedena analýza využitelnosti informačního systému na mobilních zařízeních a analýza nedostatků stávajícího mobilního zobrazení.

Následně byl vybrán vhodný přístup k návrhu řešení. Dále byly popsány technologie použité k realizaci řešení a poté byla provedena samotná implementace mobilního a tabletového zobrazení.

Po dokončení implementační části bylo provedeno předběžné testování a následné testování prototypu řešení na reálných uživatelích desktopové aplikace na dvou pobočkách optiky v Benešově a Sedlčanech. Připomínky z obou testování byly zpracovány do výsledného responzivního zobrazení.

Výhled do budoucna

Odkazy na obrazovky skladu v boční navigaci i v desktopové podnabídce by bylo dobré rozdělit do podkategorií, a to i v desktopové verzi, aby se předešlo dlouhému výpisu položek. Dále by bylo vhodné na mobilním zobrazení implementovat menší barevná tlačítka, jež v současném stavu na nejmenších zařízeních zabírají velkou část řádku v produktových výpisech.

Ke zvýšení výkonu aplikace na mobilních zařízeních by také bylo vhodné upravit načítání zdrojových dat tak, aby se načítaly s větší granularitou. Došlo by tak ke drobnějšímu rozprostření výkonu aplikace a zároveň by to neomezovalo uživatele v používání aplikace vzhledem k pomalejšímu použití na mobilním zařízení (ve srovnání s desktopovým zobrazením aplikace).

Literatura

[1] OČNÍ OPTIKA BENEŠOV s.r.o.: OČNÍ OPTIKA BENEŠOV s.r.o., [online], 2019, [cit. 2019-01-23]. Dostupné z: http://www.optikabenesov.cz/

[2] Hrách, J.: Optilynx frontend - Informační systém pro optiky, [online], 2018, [cit. 2019-01-23]. Dostupné z: https://dspace.cvut.cz/handle/10467/76252

[3] Glazar, F.: Optilynx - Informační systém pro optiky, [online], 2018, [cit. 2019-01-23]. Dostupné z: https://dspace.cvut.cz/handle/10467/76236

[4] Galitz, W. O.: *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons, druhé vydání, 2007, ISBN 0-471-084646.

[5] OptivEvidence: OpticEvidence - Systém pro komplexní správu optiky, [online], 2019, [cit. 2019-05-09]. Dostupné z: https://www.opticevidence.cz/

[6] Fakturoid s.r.o.: Mobilní aplikace — Fakturoid, [online], 2019, [cit. 2019-04-25]. Dostupné z: https://www.fakturoid.cz/mobilni-aplikace

[7] Fakturoid s.r.o.: Kdo mě stvořil? — Fakturoid, [online], 2019, [cit. 2019-04-25]. Dostupné z: https://www.fakturoid.cz/o-nas

[8] Google: iDoklad – Aplikace na Google Play, [online], 2019, [cit. 2019-04-25]. Dostupné z: https://play.google.com/store/apps/details?id=cz.ictsystem.idoklad

[9] Apple Inc.: iDoklad on the App Store, [online], 2019, [cit. 2019-04-25]. Dostupné z: https://itunes.apple.com/cz/app/idoklad/id550274216

[10] Solitea Česká republika, a.s.: iDoklad - účetnictví v cloudu - iDoklad online fakturace, [online], 2019, [cit. 2019-04-25]. Dostupné z: https://www.idoklad.cz/

[11] KASA FIK s.r.o.: KASA FIK, [online], 2019, [cit. 2019-04-25]. Dostupné z: https://www.kasafik.cz

[12] GitHub: Electron, [online], 2019, [cit. 2019-05-09]. Dostupné z: https://electronjs.org/

[13] Peterson, C.: *Learning responsive web design: a beginner's guide*. O'Reilly Media, Inc., první vydání, 2014, ISBN 9781449362942.

[14] Mozilla: CSS: Cascading Style Sheets — MDN, [online], 2019, [cit. 2019-04-29]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[15] W3C: Media Queries, [online], 2012, [cit. 2019-04-29]. Dostupné z: https://www.w3.org/TR/css3-mediaqueries/

[16] Mozilla: HTML: Hypertext Markup Language — MDN, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[17] Refsnes Data: CSS @media Rule, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://www.w3schools.com/csSref/css3_pr_mediaquery.asp

[18] Coyier, C.: A Complete Guide to Flexbox, [online], 2019, [cit. 2019-05-09]. Dostupné z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

[19] Sass: Sass Basics, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://sass-lang.com/guide

[20] Mozilla: JavaScript - MDN Web Docs Glossary: Definitions of Web-related terms — MDN, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript

[21] Maharry, D.: *TypeScript Revealed*. Apress, první vydání, 2013, ISBN 978-1-4302-5726-4.

[22] Ryabtsev, A.: Web Frameworks: How To Get Started, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://djangostars.com/blog/what-is-a-web-framework/

[23] Bodrov-Krukowski, I.: Angular Introduction: What It Is, and Why You Should Use It — SitePoint, [online], 2018, [cit. 2019-05-01]. Dostupné z: https://www.sitepoint.com/angular-introduction/

[24] Jahoda, B.: Single page application, [online], 2019, [cit. 2019-05-01]. Dostupné z: http://jecas.cz/spa

[25] Google: Angular - The Ahead-of-Time (AOT) compiler, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://angular.io/guide/aot-compiler

[26] Google: Angular - Architecture overview, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://angular.io/guide/architecture

[27] Interaction Design Foundation: Material Design, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://www.interaction-design.org/literature/topics/material-design

[28] Google: Angular Material, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://material.angular.io/

[29] Mozilla: MediaQueryList - Web APIs — MDN, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/MediaQueryList

[30] Google: Angular - Observables, [online], 2019, [cit. 2019-05-01]. Dostupné z: https://angular.io/guide/observables

[31] Stack Overflow: Subject vs BehaviorSubject vs ReplaySubject in Angular, [online], 2018, [cit. 2019-05-01]. Dostupné z: https://stackoverflow.com/questions/43118769/subject-vs-behaviorsubject-vs-replaysubject-in-angular

[32] Angular - Template Syntax, [online], 2019, [cit. 2019-05-04]. Dostupné z: https://angular.io/guide/template-syntax

[33] Google: Table — Angular Material, [online], 2019, [cit. 2019-05-06]. Dostupné z: https://material.angular.io/components/table/api

[34] Hirczy, K.: Interaction Design Foundation: Material Design, [online], 2017, [cit. 2019-05-07]. Dostupné z: https://nehalist.io/working-with-models-in-angular/

Příloha **A**

Seznam použitých zkratek

UI User Interface

SPA Single Page Application

AJAX Asynchronous JavaScript and XML

XML Extensible Markup Language

HTML HyperText Markup Language

MPA Multiple Page Application

FAB Floating Action Button

CSS Cascading Style Sheets

SASS Syntactically Awesome Style Sheets

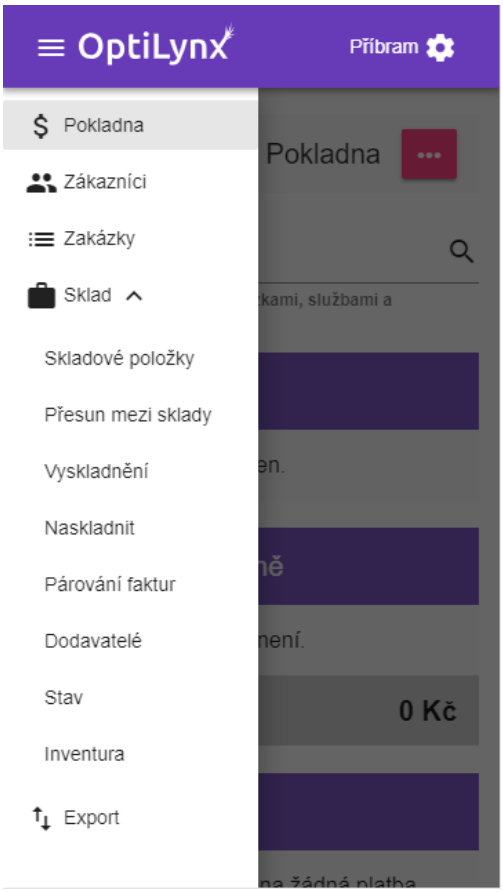
API Application Programming Interface

DOM Document Object Model

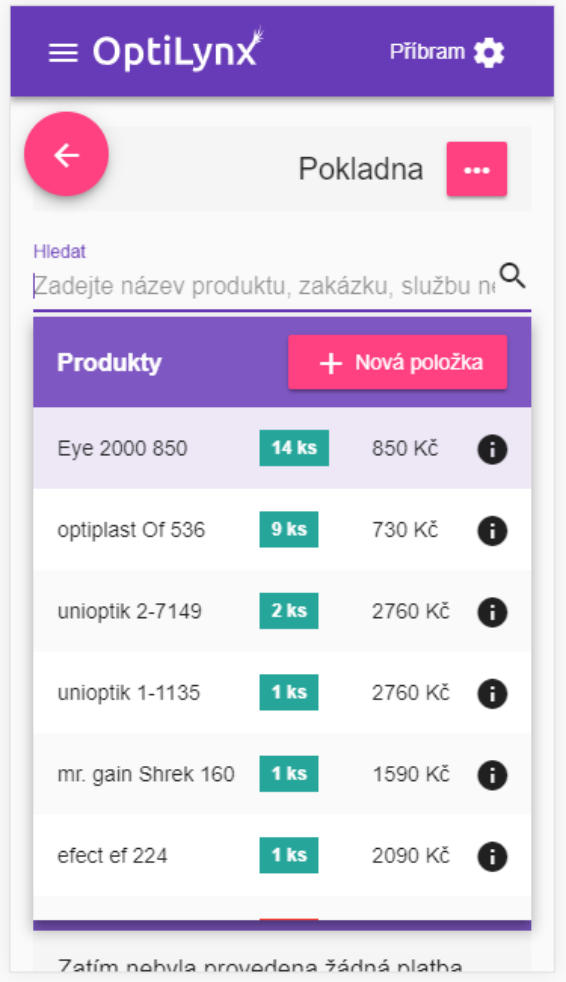
IDE Integrated Development Environment

Příloha B

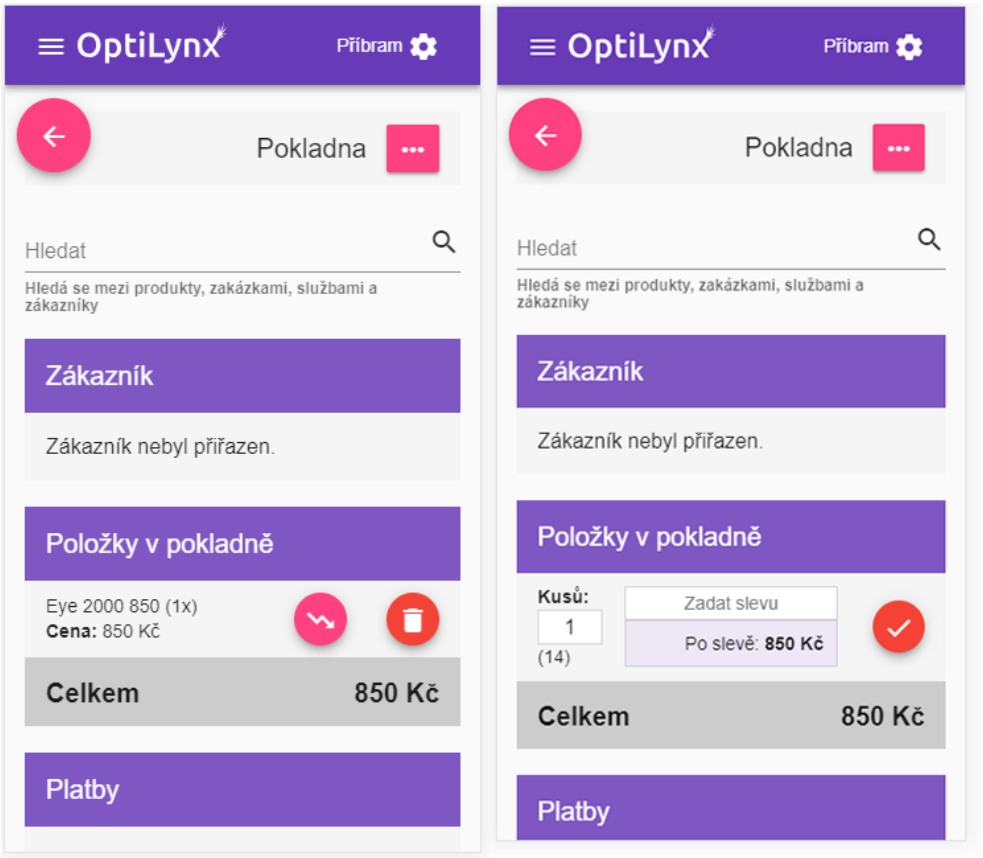
Finální mobilní zobrazení aplikace Optilynx



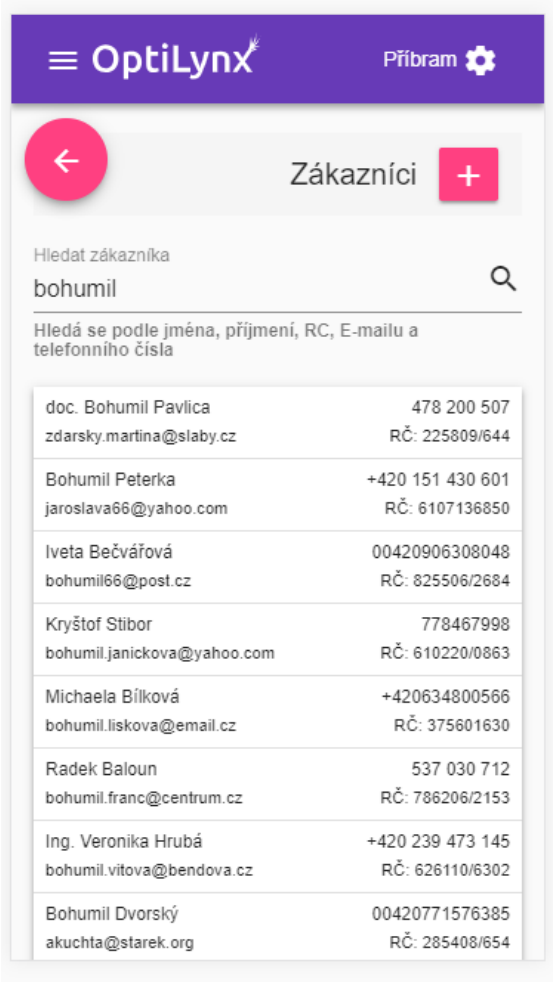
Obrázek B.1: Mobilní zobrazení Optilynx – Vysouvací nabídka



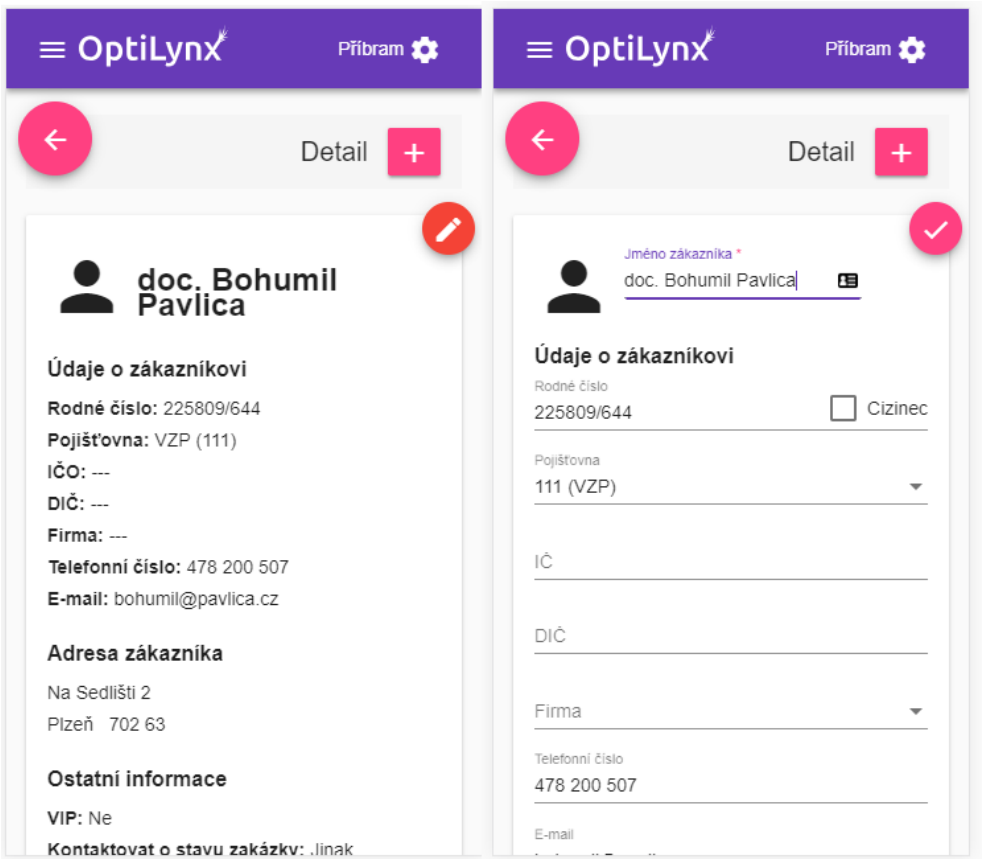
Obrázek B.2: Mobilní zobrazení Optilynx – Pokladna – Vyhledávací pole



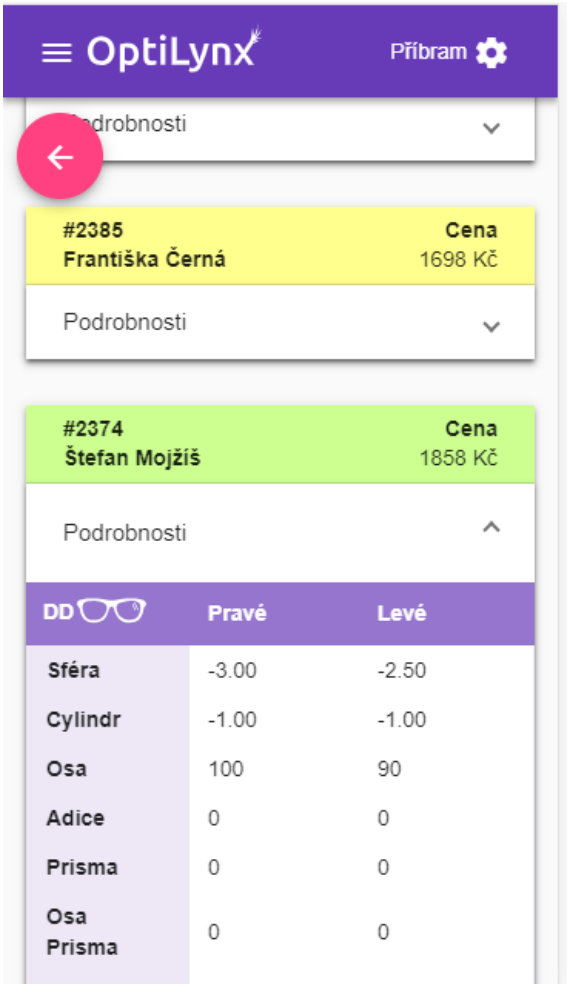
Obrázek B.3: Mobilní zobrazení Optilynx – Pokladna – Výpis produktů



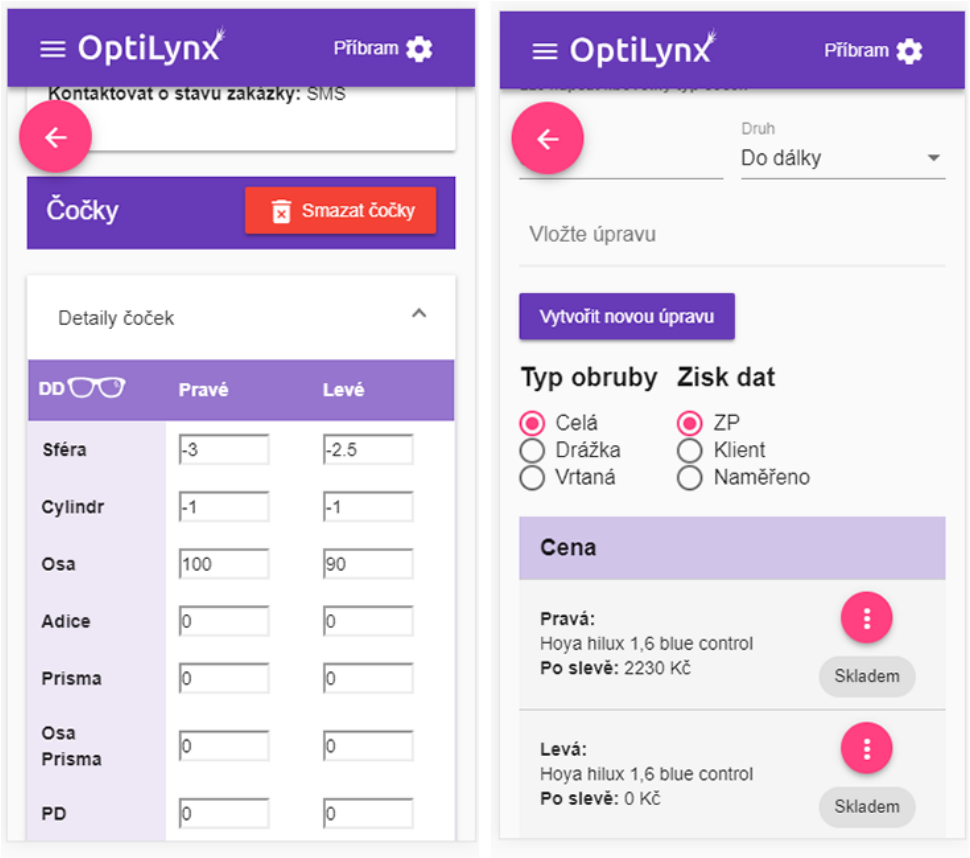
Obrázek B.4: Mobilní zobrazení Optilynx – Zákazníci



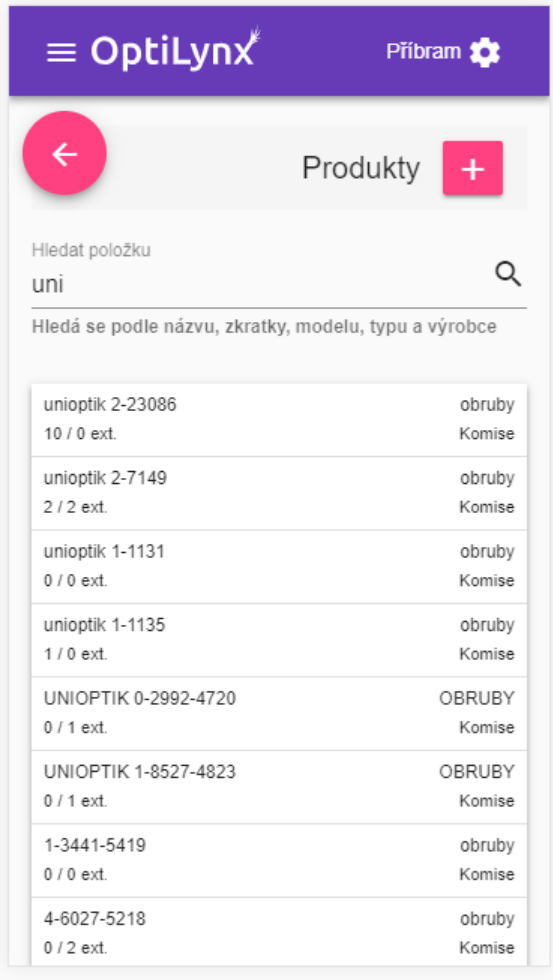
Obrázek B.5: Mobilní zobrazení Optilynx – Karta zákazníka



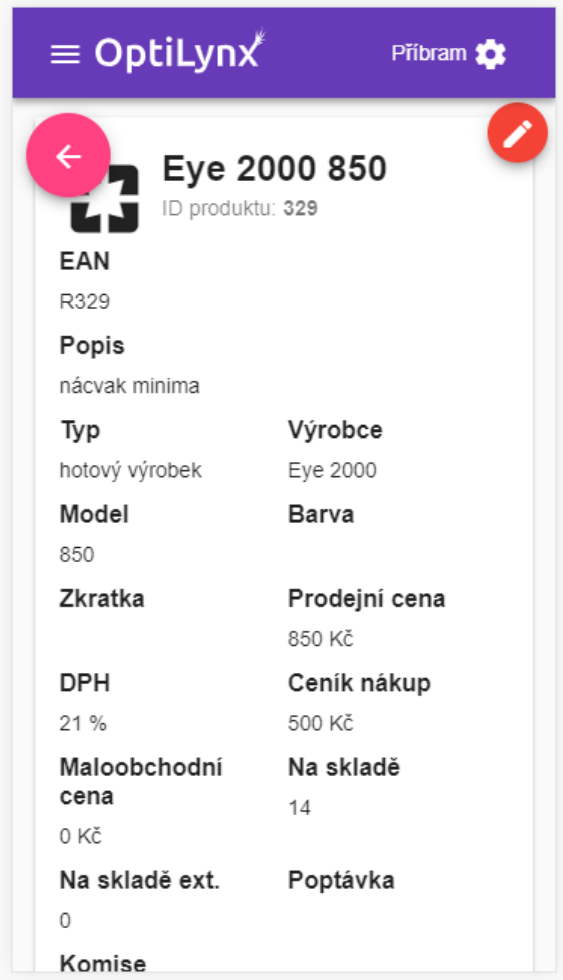
Obrázek B.6: Mobilní zobrazení Optilynx – Zakázky



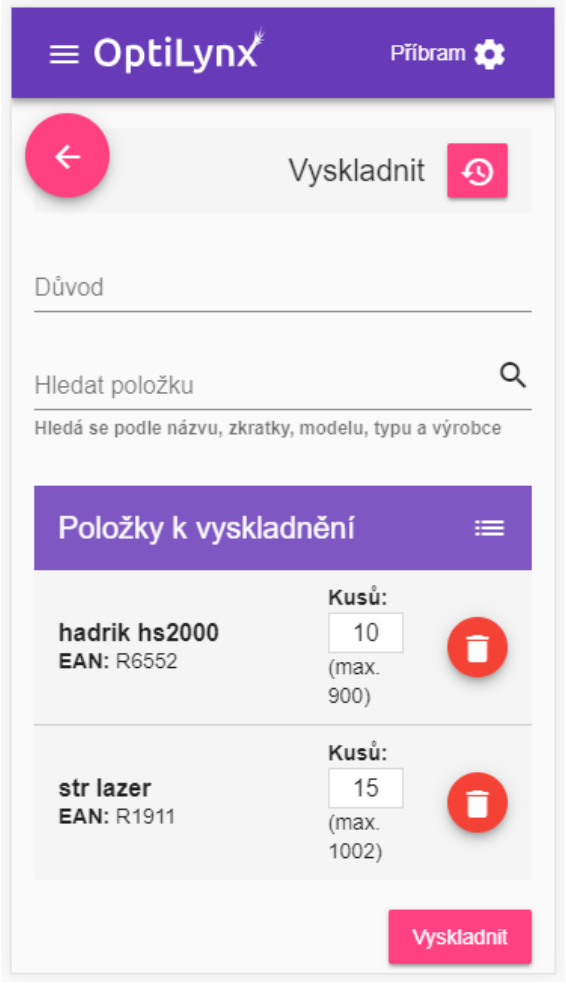
Obrázek B.7: Mobilní zobrazení Optilynx – Karta zakázky



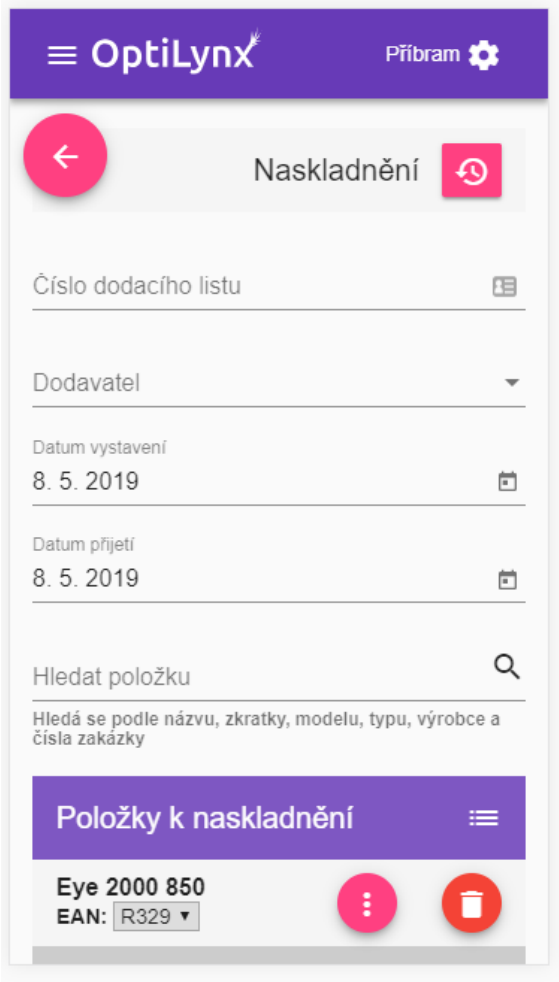
Obrázek B.8: Mobilní zobrazení Optilynx – Skladové položky



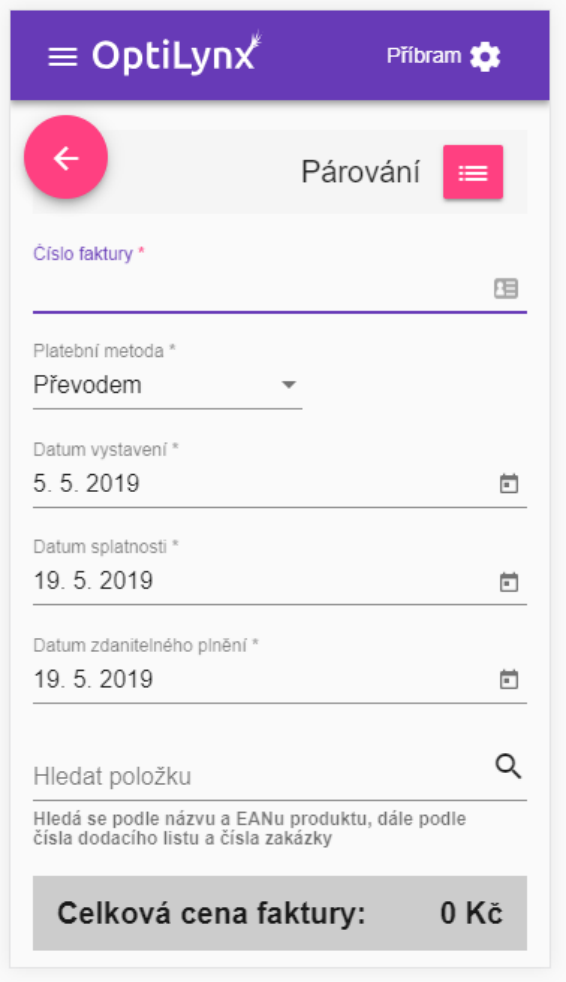
Obrázek B.9: Mobilní zobrazení Optilynx – Detail produktu



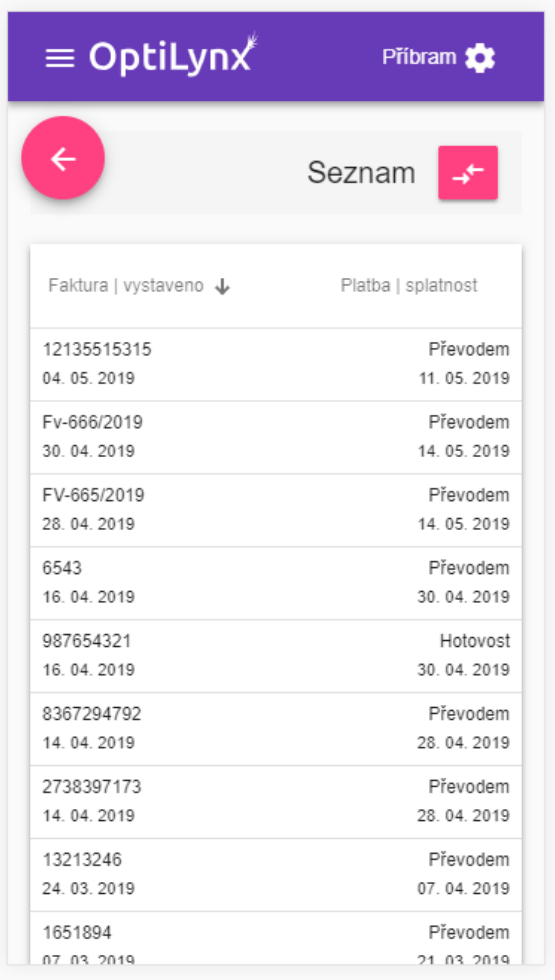
Obrázek B.10: Mobilní zobrazení Optilynx – Vyskladnění



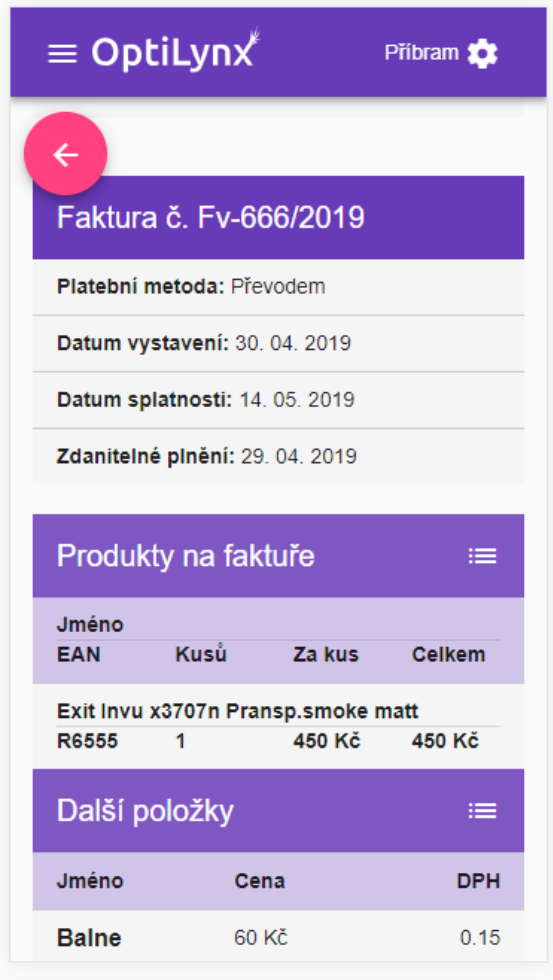
Obrázek B.11: Mobilní zobrazení Optilynx – Naskladnění



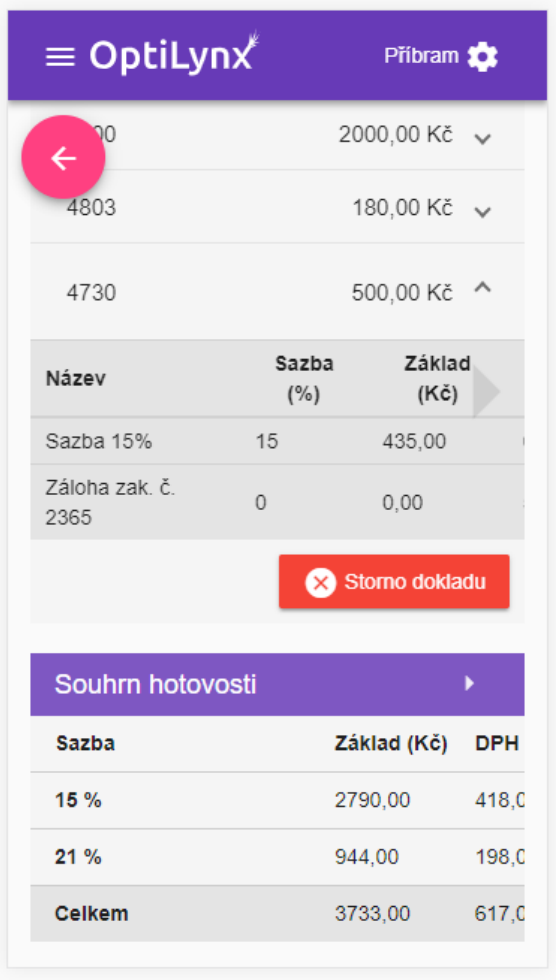
Obrázek B.12: Mobilní zobrazení Optilynx – Párování faktur



Obrázek B.13: Mobilní zobrazení Optilynx – Historie párování faktur



Obrázek B.14: Mobilní zobrazení Optilynx – Detail faktury



Obrázek B.15: Mobilní zobrazení Optilynx – Pokladní přehled

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `img` ..... obrázky finální podoby mobilního zobrazení aplikace Optilynx
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF