FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Mapa skladu – frontend
Student:	Vojtěch Kopecký
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem této práce je návrh a realizace vhodného frontendu pro tvorbu mapy skladu s důrazem na použitelnost uživatelského rozhraní. Při realizaci navažte na výstup z týmu předmětu SP1 a SP2, kterého jste se účastnil.

Postupujte dle těchto kroků.
1. Proveďte důkladnou analýzu dané problematiky. Zároveň analyzujte frontendové rozhraní aktuálního skladového systému Atlantis, který by měl v budoucnu Vaši práci integrovat.
2. Na základě analýzy proveďte vhodný návrh uživatelského rozhraní, za použití vhodných postupů / metodik.
3. Návrh řádně konzultujte minimálně se svým vedoucím a projektovým manažerem skladového systému Atlantis, případně i s budoucími uživateli.
4. Na základě návrhu implementujte vaše uživatelské rozhraní, neopomeňte nutnou komunikaci s backendovou částí.
5. Výsledné řešení řádně otestujte. Navrhněte a realizujte vhodné testy, minimálně proveďte usability testování.
6. Shrňte dosažené výsledky, navrhněte možná budoucí vylepšení dle získaných zkušeností.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 23. ledna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Mapa skladu – frontend

Vojtěch Kopecký

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

10. května 2022

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu práce Ing. Jiřímu Hunkovi, který mi po celou dobu projektu radil, vyřizoval mnohé organizační záležitosti a pomáhal mi projekt dotáhnout do konce. Dále děkuji Ing. Oldřichovi Malcovi, který mi pomáhal s technickou stránkou věci a poskytl mi nespočet neocenitelných rad z praxe. Děkuji všem testerům za obětovaný čas a také zaměstnancům skladu, kteří mi poskytli možnost zúčastnit se terénního průzkumu. V neposlední řadě děkuji své rodině a přátelům za neustálou podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při 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, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2022 . . . . . .

Odkaz na tuto práci

Kopecký, Vojtěch. *Mapa skladu – frontend*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Práce se zaměřuje na proces analýzy, návrhu, vývoje a testování vhodného uživatelského rozhraní pro tvorbu mapy skladu, s důrazem na použitelnost a uživatelskou přívětivost. Vzniklý frontend mapového editoru využívá technologie Vue.js a Vuetify a je zakomponován do webové aplikace skladového systému, zároveň je zajištěna komunikace s backendem. V návrhové fázi se práce zabývá problematikou zkreslení a znázornění skladových objektů do virtuální podoby, která je nejen pochopitelná pro běžného uživatele, ale zároveň užitečná a dostatečně přesná pro optimalizační algoritmy, které data z mapy využijí pro zvýšení efektivity nejčastějších skladových procesů. Výsledná mapa má velký potenciál jak pro reálné uživatele, tak pro celý skladový systém, práce tedy prozkoumá další možnosti rozšíření a uplatnění mapy.

Klíčová slova skladový systém, návrh uživatelského rozhraní, použitelnost, frontend, Vue.js, Vuetify, mapa skladu

Abstract

This thesis focuses on the process of analysis, design, development and testing of a fitting user interface for creating a map of a warehouse, with emphasis on usability and user comfort. The newly created frontend was made using Vue.js and Vuetify and is embedded into a web application of a warehouse management system, while communication with the backend has also been established. In the design phase, the thesis deals with the issue of distorting and representing warehouse objects in a virtual form, that is both understandable by the average user and accurate enough for optimization algorithms, which make use of the data from the map to increase effectivity of the most frequent warehouse processes. The resulting map has great potential for both real users and the entire warehouse management system as a whole, the thesis therefore explores further extension and application of the map.

Keywords warehouse management system, UI design, usability, frontend, Vue.js, Vuetify, warehouse map

Obsah

Seznam obrázků

Seznam ukázek kódu

Úvod

V dnešní době se v logistické sféře vyskytuje čím dál větší snaha vše automatizovat a vypustit lidský faktor – samořídící auta, doručování balíčků dronem, sklad, kde veškerou práci vykonávají roboti... I když tyto úvahy zní lákavě, a dokonce se už takovéto koncepty zkouší v praxi, je zřejmé, že ještě dlouhou dobu budeme v logistice spoléhat na lidské zdroje. Proto je potřeba vyvíjet informační systémy, které jsou nejen dostatečně přehledné a jednoduché na použití pro obyčejného uživatele, ale zároveň se snaží lidské práci vyjít vstříc a ideálně ji nejen zpříjemnit, ale i zefektivnit, a ušetřit tak drahocenný čas zaměstnanců.

Tomuto úkolu jsem se rozhodl věnovat nejdříve ve dvou předmětech vyučovaných na FIT ČVUT (SP1 a SP2), a později na výstupy navázat bakalářskou prací. Začal jsem spolupracovat na rozvoji skladového systému od společnosti Jagu s.r.o., který cílil na malé a středně velké sklady. Skladový systém byl v té době již plně nasazený v několika reálných skladech a disponoval funkcemi, které plně umožňovaly elektronicky evidovat většinu informací jak o zboží, tak i o jednotlivých skladových procesech. Chybělo však rozhraní pro podporu práce skladníka přímo v terénu, které by uživateli nabízelo efektivní postup práce, a tím pádem šetřilo čas zaměstnanců.

Možné řešení skladových optimalizací spočívalo v mapě skladu, kterou bylo potřeba pro každý sklad vytvořit. Nad touto problematikou jsem provedl kompletní proces tvorby uživatelského rozhraní, který je důkladně popsán a zdokumentován v této bakalářské práci. Nejdříve odůvodním, proč je mapa skladu potřeba, a přesně popíšu, jaké skladové úlohy podporuje. Následně prozkoumám různé možnosti, jak mapu znázornit, tvořit a jak s ní interagovat, přičemž bude kladen důraz na budoucí uživatele, kteří by měli být schopni znázornění mapy lehce porozumět a efektivně s ním zacházet. Vytvořím prototyp mapového editoru, který podrobím uživatelskému testování, a získané poznatky využiji při finální implementaci. Na závěr popíšu další možnosti rozšíření a využití finálního produktu.

Jedním z hlavních důvodů pro výběr tohoto tématu byla možnost podílet se na opravdovém projektu, jehož výsledky už jsou používány v praxi. Zároveň jsem měl příležitost seznámit se s novými technologiemi, které skladový systém využívá. Při práci jsem nabyl spoustu nových vědomostí a na druhou stranu jsem vytvořil něco, co má potenciál ulehčit a zpříjemnit práci skladovým zaměstnancům.

Kapitola 1

Definice pojmů

V této sekci zadefinuji nejdůležitější pojmy z oblasti návrhu uživatelského rozhraní. Vysvětlím, na které konkrétní vlastnosti je při návrhovém procesu třeba brát ohled a jaké jsou preferované metody pro jejich dosažení.

1.1 Uživatelské rozhraní

Uživatelské rozhraní, UI (z anglického user interface) nebo jednoduše „rozhraní“ je prostředek, kterým osoba ovládá softwarovou aplikaci nebo hardwarové zařízení [1].

Návrh uživatelského rozhraní, zvláště počítačového, je už dlouhou dobu zkoumán. Počátky těchto výzkumů sahají až do 80. letech 20. století, kdy vznikla nová disciplína zabývající se interakcí mezi člověkem a počítačem – **Human-computer interaction** (zkráceně HCI) [2]. V této době totiž osobní počítače začaly být dodávány s novými ovládacími prvky, jako např. myš, a programy na počítači se najednou daly ovládat pomocí grafických uživatelských rozhraní, které uživatelům podávaly informace intuitivnějším způsobem než jen pomocí příkazové řádky. S tolika novými možnostmi bylo potřeba přijít s určitými zásadami, kterých by se nová UI měla držet, aby se v nich uživatel neztratil a aby je mohl správně používat.

HCI spojuje mnoho různých oborů, od technických (počítačová grafika, softwarové inženýrství, ergonomika) až po humanitní (psychologie, sociologie, estetika). Při tak širokém rozsahu HCI nepředstavuje tedy přesnou kuchařku, podle které vytvoříme to nejlepší uživatelské rozhraní, nýbrž nám dává k dispozici kolekci metodik a principů, kterých bychom se měli při návrhu držet, nebo je alespoň vzít v potaz. [2][3]

1.2 User-centered design

User-centered design (zkráceně UCD) je metodika HCI, která vyjadřuje jednu z nejdůležitějších myšlenek uživatelského rozhraní – tvoříme jej pro reálné uživatele, ne pouze pro sebe. Proto se musíme dostatečně vcítit do role cílového uživatele, zjistit, čeho přesně chce daný uživatel při interakci s počítačem dosáhnout, jak by toho chtěl dosáhnout a za jakých podmínek bude provádět samotnou interakci. [4]

1.2.1 Persony

Persony jsou fiktivní postavy, které jsou založené na vlastnostech reálných uživatelů [5]. Cílem stanovení person je lepší porozumění cílovým uživatelům, a tedy zlepšení finálního produktu. Často se totiž stává, že nemůžeme uvažovat cílové uživatele jako jednu skupinu, jelikož mají různé dovednosti, očekávání a motivace. Tím, že cílové uživatele kategorizujeme do skupin, můžeme jednodušeji vyjít každé skupině vstříc a navrhnout tak rozhraní, které bude lépe použitelné.

Kromě cílových uživatelů do person můžeme zahrnout i uživatele, kteří naše rozhraní naopak používat nebudou. Cooper [5] identifikoval celkem 3 typy person:

- **Primární:** Reprezentuje hlavního cílového uživatele.
- **Sekundární:** Cílový uživatel, který má však jiné požadavky/vlastnosti, nebo nebude rozhraní používat tak často jako primární. Sekundární personou dále může být někdo, kdo rozhraní sice nepoužívá, ale je ovlivněn jeho používáním (např. zákazník v obchodě).
- **Negativní (také anti-persony):** Skupina uživatelů, o které jsme si jisti, že naše rozhraní používat nebudou a žádným způsobem na ně necílíme.

1.2.2 Artefakt

Pojem *artefakt* označuje způsob, kterým reprezentujeme reálné informace a procesy v našem rozhraní. Artefakt tedy např. definuje, jak v našem rozhraní vypadají různé objekty nebo co v rozhraní udělat (např. na jaké položky v menu kliknout), aby uživatel provedl určitý proces. Jedná se tedy o určitý způsob zkreslení reality. U artefaktu je zásadní, aby cílový uživatel v rozhraní jednoduše identifikoval, co jaký prvek v rozhraní reprezentuje a jak s těmito prvky zacházet, aby dosáhl svého cíle. Artefakt by měl tedy uživateli vycházet vstříc – jednotlivé prvky a akce v rozhraní by měly korespondovat s představami a profesní zkušeností uživatele. Dobrý artefakt je ten, ve kterém se cílový uživatel ihned zorientuje a dokáže v něm udělat přesně to, co chce. [6]

1.3 Použitelnost

Použitelnost uživatelského rozhraní určuje, jak jednoduché je jej používat. Dle Nielsena [7] je použitelnost definovaná pěti komponentami:

- **Snadnost učení:** Jak jednoduché je pro uživatele plnit základní úkoly, když rozhraní vidí poprvé?
- **Efektivita:** Jak rychle jsou uživatelé schopni plnit i pokročilejší úkoly poté, co se s rozhraním dostatečně seznámili?
- **Zapamatovatelnost:** Pokud uživatelé rozhraní již delší dobu neviděli, jak jednoduše se dokáží dostat na předešlou úroveň zručnosti v daném rozhraní?
- **Chyby:** Kolik chyb uživatelé dělají, jak moc závažné tyto chyby jsou a jak jednoduché je se z těmito chybami vypořádat?
- **Spokojenost:** Jak příjemné je rozhraní používat?

Nielsen dále poukazuje na další zásadní vlastnost uživatelského rozhraní, kterou je **utilita**[1]. Utilita určuje, jestli rozhraní opravdu dělá to, k čemu bylo vytvořeno. Utilita a použitelnost jsou důležité vlastnosti UI a dohromady určují **užitečnost** daného rozhraní: fakt, že je rozhraní jednoduché na použití, nám k ničemu není, pokud výstupy daného rozhraní nesplňují naše požadavky. Zároveň není žádoucí, když teoreticky lze dosáhnout kýžených výsledků v daném rozhraní, ale prakticky jich nelze dosáhnout kvůli přílišné složitosti rozhraní. UI proto musí mít jak vysokou použitelnost, tak utilitu, aby bylo užitečné [7].

1.4 Testování použitelnosti

Při testování použitelnosti výzkumník (nebo také moderátor) zadá účastníkům nějaké úlohy, které mají být vykonány v daném rozhraní [8]. Důvody, proč testovat, se dají shrnout do tří důležitých bodů:

- **Identifikovat problémy** ve stávajícím rozhraní
- **Odhalit příležitosti** ke zlepšení rozhraní
- **Porozumět** chování a přednostem cílových uživatelů

[1] Z anglického *utility*. Nejbližším překladem do češtiny je sice *užitečnost*, jelikož však dále budu definovat anglický pojem *usefulness*, který má v češtině identický překlad, budu dále používat pojem *utilita*.

Před zahájením testování je potřeba sestavit si plán: [9][10]

- **Úlohy**: Které části rozhraní chceme testovat? Které úkony by mohly být nejproblematičtější? Úlohy by měly být sestaveny tak, abychom měli příležitost odhalit co nejvíce problémů a ověřit si naše hypotézy o rozhraní. Zároveň by měly být realistické, přesně definované a uživateli by měly dát možnost užívat rozhraní přirozeně, bez nutnosti přílišných zásahů ze strany moderátora.
- **Prostředí**: Při výběru prostředí je třeba dbát na to, abychom v něm měli dostatečnou kontrolu – měli bychom být schopni test dobře moderovat a s účastníkem pohodlně komunikovat, zároveň musíme zajistit, abychom dokázali po celou dobu testu sbírat data nutná k finálnímu vyhodnocení.

  Testování lze provádět prezenčně či distančně – distanční testování sice vyžaduje více přípravy, ale stále poskytuje užitečné výsledky a je méně časově náročné pro účastníky.
- **Účastníci**: Účastníci by měli korespondovat se stanovenými personami – měli by reprezentovat cílové uživatele. Účastníků najmeme tolik, abychom měli dostatečné množství dat k přesnému vyhodnocení rozhraní – Nielsen a Norman udávají 3-5.
- **Sběr dat**: Je třeba určit, co všechno si budeme v průběhu testování zaznamenávat a jakým způsobem informace získáme.
  - **Úvodní rozhovor**: Se sběrem dat můžeme začít hned na začátku testu, kdy účastníka seznamujeme s celkovým průběhem. Můžeme položit pár jednoduchých otázek, abychom se dozvěděli více o účastníkovi a o jeho zkušenosti s daným typem rozhraní.
  - **Pozorování**: Probíhá při samotném testu – buď moderátor, nebo nějaká pověřená osoba pozoruje, jaké kroky uživatel v rozhraní koná a jaké emoce uživatel dává najevo. Je dobré si tyto informace zaznamenávat do tzv. **logu**, kde jeden záznam sestává z časového údaje a samotného poznatku (např. „20:31: Uživatel se snažil kliknout na tlačítko *Uložit* a byl frustrovaný z toho, že nefunguje“)
  - **Audiovizuální záznam**: Očekává se, že se nepodaří všechny důležité poznatky zachytit pouze pozorováním – proto se často pořizují audiovizuální záznamy, které se zpětně podrobně procházejí pro identifikaci dalších možných problémů, které jsme předtím nezachytili. Nahráváme jak zvukovou stopu (komunikaci mezi účastníkem a moderátorem), tak např. záběr obličeje účastníka, zároveň je vhodné nahrávat i obrazovku testovaného zařízení, pokud to situace umožňuje. Pro pořízení je nutný **informovaný souhlas** účastníka, jelikož zpracováváme osobní údaje nad rámec očekávání účastníka.

  - **Konečný rozhovor**: Po splnění všech úloh s účastníkem vedeme diskuzi, kde si poslechneme jeho názory a snažíme se porozumět klíčovým problémům, které při testování nastaly. Necháme účastníka vysvětlit, proč jednotlivé kroky konal a co očekával, že se v rozhraní stane.

- **Způsob vyhodnocení**: Jak ze sesbíraných dat zjistíme, jestli je rozhraní použitelné?

  - **Kvantitativní**: míra úspěšnosti jednotlivých úloh, čas strávený nad úlohou, počet kliknutí ke splnění úlohy...
  - **Kvalitativní**: reakce účastníka (výraz obličeje, tón hlasu), spokojenost účastníka, zkušenost účastníka, jak obtížně se s rozhraním zacházelo...

Během testu je vhodné držet se určitých zásad: [8][9][10]

- Účastníka nejdříve s testem a jednotlivými úlohami seznámíme. Důležité je zmínit, že **netestujeme jeho, ale dané rozhraní** – proto by se neměl zdráhat jakkoliv rozhraní kritizovat. Kritika naopak zvýší šanci na zlepšení použitelnosti rozhraní.

- Účastníka dále požádáme, aby při testu **přemýšlel nahlas** a své kroky v rozhraní stručně popisoval. Získáme tím více dat při pozorování a v audiovizuálním záznamu.

- Pokud se účastník při konání nějaké úlohy zadrhne, doporučuje se nijak nezasahovat a účastníkovi nepomáhat – chceme, aby test simuloval reálnou situaci, ve které uživatel většinou žádnou pomoc nemá k dispozici. Navíc je užitečné pozorovat, jak si účastník počíná v krizových situacích a jakým způsobem se z nich snaží dostat.

1.5 Prototypování

Prototyp je experimentální model navrhovaného řešení buď části, nebo celého rozhraní. Používá se pro testování nápadů a ověřování různých domněnek, aby návrháři mohli provést vhodná vylepšení či změnit směr, kterým se v návrhu vydali [11].

Prototypy se typicky dělí na lo-fi (low fidelity) a hi-fi (high fidelity). Lo-fi prototypy bývají jednodušší a slouží hlavně k ověření základních myšlenek a funkcí rozhraní – nezabývají se tedy finálním vzhledem rozhraní. Lze je sestrojit rychle a levně, bývají však méně přesné. Mezi populární techniky lo-fi prototypů patří např. papírové modely, ve kterých rozhraní nakreslíme na papír, nebo wireframy, kde se většinou využívá digitální software pro sestavení

jednoduchých interaktivních aplikací, ve kterých již uživatel může klikat na jednotlivé prvky.

Hi-fi prototypy bývají komplexnější – často se velmi podobají finálnímu produktu a je v nich kladen důraz i na vzhled rozhraní. Umožňují přesnější a hlubší testování, bývá však těžší je vytvořit – je potřeba použít komplexnější prototypovací software nebo prototyp nakódovat. [12]

1.6 Vývojové metodiky softwaru

Metodiky představují určitou strukturu, které by se měli vývojáři držet. Bez struktury mohou vývojáři selhat tím, že zákazníci si budou neustále měnit podmínky nebo vývojáři ztratí příliš mnoho času a prostředků vývojem něčeho, co zákazníci vůbec nechtěli. Výsledkem je ztráta peněz, frustrovaní vývojáři a nekvalitní či skoro nepoužitelný software. Metodiky pomáhají vývojářům softwaru pracovat efektivněji a s menšími riziky.

1.6.1 Vodopád

Navzdory desetiletím od doby, kdy byla poprvé použita, je metodologie vodopádu v některých projektech stále relevantní i dnes. Je to jednoduchá, lineární metoda, kde jsou vývojové fáze uspořádány do sekvenčních, kaskádových procesů.

V tomto přístupu není cesty zpět, stejně jako u reálného vodopádu – voda v něm proudí pouze jedním směrem. Každá fáze vodopádu musí být dokončena před přechodem na další – např. před zahájením návrhu musí být stanoveny všechny požadavky.

Princip vodopádu je snadno pochopitelný, díky čemuž je oblíbený mezi týmy s menšími zkušenostmi s návrhem. Kvůli nemožnosti vracet se do předchozích fází vývoje je však tato metodika neflexibilní a je třeba se jí vyhnout u projektů s rychle se měnícími požadavky.

1.6.2 Iterativní vývoj

Při iterativním vývoji se software vyvíjí pomocí opakovatelných cyklů – iterací. Při každé iteraci je cílem do software přidat nebo změnit funkcionalitu – může tedy docházet k vytyčení nových požadavků, návrhu, testování a vývoji nové funkcionality. Software se tedy s každou další iterací stává funkčnější a použitelnější. Iterativní vývoj je vhodný pro projekty, kde nejsou předem jasné požadavky a jednotlivé fáze vývoje (návrh, testování…) nelze provést sekvenčně. [13]

Kapitola 2

Popis skladové domény

Tato kapitola popisuje počáteční proces průzkumu skladového prostředí. Zadefinuji skladové pojmy, procesy a role, které jsou nutné pro pochopení dané problematiky. Cílem kapitoly je přesně popsat problém, který vyřeším, nebo alespoň podpořím přidáním nové funkcionality, a dále vysvětlit, proč se jako nejvhodnější řešení jeví právě mapa skladu.

Na jaře 2021 jsem se stal součástí vývojového týmu v rámci předmětu SP1, který dostal za úkol rozšířit skladový systém *Atlantis* o nové funkcionality. Naše zadání bylo implementovat funkcionalitu **shlukování úkolů** (anglicky *order batching*). V našem týmu jsme ihned zjistili, že pro pochopení zadání, a tedy i pro správné určení postupu práce na projektu bude potřeba se s prostředím skladu dostatečně seznámit.

Skladový systém nejdříve vznikl jako praktická část dvojice diplomových prací, psaných Ing. Oldřichem Malcem a Ing. Pavlem Kovářem v roce 2020. Od té doby se software vyvíjí komerčně, studentům FIT je však dále nabízena možnost podílet se na vývoji nových funkcionalit a rozhraní, např. v rámci vyučovaných předmětů či akademických prací. V jedné takové práci, psané Bc. Denisem Talárem, již byl proveden důkladný rozbor skladové domény, spolu se slovníkem nejčastějších skladových pojmů a popisem skladových procesů. Díky informacím z práce pana Talára [14] a následnému ověřování a diskuzi s experty na skladovou doménu[2] jsme byli schopni identifikovat a seznámit se s těmi nejdůležitějšími pojmy a procesy, které jsou součástí dané problematiky.

Je důležité poznamenat, že daný skladový systém cílí převážně na e-shopové a distribuční sklady. Systém je vyvíjen tak, aby mohl být použit na mezinárodní úrovni, avšak výzkum pana Talára byl prováděn na skladech v Česku. Použitá terminologie se tedy může lišit v jiných typech skladů či ve skladech v jiných zemích.

[2] vedoucí práce Ing. Jiří Hunka a Ing. Oldřich Malec

2.1 Doménové pojmy

Sklad je prostor pro ukládání a manipulaci se zbožím. Zboží se ukládá na tzv. skladových umístěních. Rozlišujeme více druhů skladů podle toho, jaké služby sklad nabízí.

Regál je několikapatrová struktura pro ukládání zboží. Regály jsou většinou ve skladu organizovaně rozmístěné a pojmenované tak, aby se ve skladu dalo lépe zorientovat a tím i efektivněji pracovat. Položky v regálu jsou dosažitelné buď ručně, nebo pomocí manipulační techniky – v případě vysokých regálů nebo těžkých položek se k přístupu a následné manipulaci položek musí použít tzv. retrak.

Retrak je vysokozdvižný vozík, který používáme v případě, že chceme zacházet s těžkým nebo příliš vysoce umístěným zbožím. Mají omezenou baterii, která však většinou vydrží celý den práce. Po konci pracovní doby se všechny retraky zaparkují do určeného prostoru ve skladu, kde se nechají přes noc nabíjet. Při užití retraku musí skladník dodržovat zásady BOZP.

Umístění označuje nějaký prostor ve skladu, na kterém lze ukládat položky. Nejčastěji se za umístění považuje místo v patře nějakého regálu, ale může také být jinde – paleta na zemi, krabice, místo ve skříni... Stejně jako regály jsou i umístění pojmenována – časté je použití kombinace jména regálu, čísla patra a dalších označení. Ve větších skladech je velmi důležité používat takový pojmenovávací systém, který umožní skladníkům jednotlivá umístění co nejrychleji najít.

Položka (též **produkt**, **zboží**) je cokoliv, co je uchováváno na skladových umístěních. Každá položka je zaevidována v systému a náleží jí nějaký čárový kód. Některé položky mohou mít šarži a na ní uvedené datum spotřeby, což znamená, že nemusí být trvanlivé a časem se mohou znehodnotit. U těchto položek je snaha expedovat ty, které mají datum spotřeby co nejbližší.

Objednávka nejčastěji pochází z e-shopu, jedná se však o obecnou žádost o doručení kusů položek na nějakou adresu. Objednávky jsou tedy základní jednotkou, která má za důsledek většinu skladových procesů – vyskladnění, balení, expedice... Položky v rámci jedné objednávky se balí většinou do jedné krabice.

Úkol je jakýkoliv proces nebo skupina procesů, které je potřeba ve skladu provést skladníky – mohou jimi být naskladnění, vyskladnění či např. balení. Úkoly jsou do systému zadávány vedoucím, který může úkoly rovnou přiřazovat daným skladníkům. Úkoly mohou být též označeny jako volné, což znamená, že jakýkoliv skladník si úkol může vzít na starost a přiřadit si jej sobě.

2.2 Procesy

Naskladnění je uložení položek na skladové umístění, za účelem jejich dlouhodobého uchování.

Vyskladnění je vyjmutí určitého množství položek ze skladového umístění za účelem další manipulace. Tou může být např. balení či zpětné naskladnění na jiné umístění.

Balení je proces, kdy jsou vyskladněné kusy zboží zabaleny do vhodně velké krabice, která je zabalená do fólie a následně expedována.

Expedice je proces, při kterém jsou kusy zboží předány dopravci, který zboží dále doručí zákazníkům nebo jiným příjemcům.

Přeskladnění je přesun jednoho či více kusů produktů z jednoho skladového umístění na druhé. Jedná se tedy o vyskladnění a následné naskladnění. Nejčastějším důvodem pro přeskladnění je optimalizace – např. přemístění vysoce žádaného produktu na dostupnější místo.

Příjem je proces, při kterém je zboží zaevidováno do systému a následně uloženo na nějaké skladové umístění.

2.3 Role

Skladník je člověk manipulující se zbožím. Provádí naskladnění, vyskladnění či přeskladnění. Při práci si pomáhá retrakem.

Balič má na starost balení kusů zboží a kontroluje, jestli v rámci každé objednávky zabalil správné množství položek.

Vedoucí má na starost správný chod práce ve skladu. Vytváří skladníkům úlohy a schvaluje jejich splnění.

2.4 Shlukování úkolů

Po identifikaci pojmů, procesů a rolí jsme měli dostatečné podklady k tomu, abychom si zadefinovali pojem shlukování úkolů.

Shluknutí úkolů (anglicky *order batching*) je proces, při kterém se dva a více úkolů sloučí do jednoho úkolu. Shluknutí se může provést za různých podmínek:

- shluknutí na základě vlastností objednávek
- shluknutí na základě polohy
- shluknutí na základě obou faktorů

2.4.1 Shluknutí na základě vlastností objednávek

Při tomto způsobu shluknutí se využívá faktu, že úkoly jsou součástí přípravy objednávek, které si jsou nějak podobné. Touto problematikou se zabýval Bc. Jan Cvrček, který ve své práci [15, Chapter 2.2.4.1] popsal tyto strategie shlukování:

- Jednoprvkové objednávky se stejným produktem
- Jednoprvkové objednávky s různými produkty
- Identické objednávky
- FIFO – chronologicky, dle času vytvoření

Velký smysl dává např. první strategie – *Jednoprvkové objednávky se stejným produktem*. Pokud se vyskladní jeden stejný produkt dvakrát a následně se oba kusy přesunou do balicí zóny, celý proces zabere přibližně polovinu času, než kdyby se vyskladnil a přesunoval produkt po jednom. Samozřejmě se musí vzít v potaz to, že všechny kusy produktu, které by se mohly hromadně vyskladnit, musí být dostupné na ideálně jednom skladovém umístění.

Menší smysl už dává např. poslední strategie – *FIFO – chronologicky, dle času vytvoření*. Fakt, že objednávky byly vytvořeny v malém časovém úseku, nic neříká o tom, jestli jsou dané položky ve skladu umístěné blízko sebe. Stále však tato strategie může být užitečná, a to už jenom z důvodu, že skladník většinou unese více položek najednou. Pokud se tedy shluknou objednávky, jejichž všechny položky skladník dokáže unést, opět může být rychlejší vyskladnit první objednávku, pak druhou a obě poté přesunout do balící zóny, než podniknout dvě cesty.

Nehledě na zvolenou strategii je tento typ shlukování založen na jednoduchých podobnostech informací, které lze všechny jednoduše ze skladového systému získat. Na druhou stranu nemusí všechna tato shluknutí být pro skladový chod přínosná – zvláště u 2. a 4. strategie se může stát, že nový průchod shluknutým úkolem bude trvat stejně dlouho, ne-li déle.

2.4.2 Shluknutí na základě polohy

Úkoly dále můžeme shluknout, pokud víme, že jsou položky ve skladu umístěny v dostatečné blízkosti. Nemusíme se tedy omezovat na objednávky, které jsou identické nebo obsahují pouze jeden stejný produkt.

Tento způsob shlukování má potenciál zvýšit pravděpodobnost, že dojde k ušetření času. Do velké míry však záleží na přesnosti dat, a to zejména vzdáleností mezi dvěma umístěními. Pro určení dobrého odhadu času trasy však potřebujeme nejen čistou vzdálenost, ale také údaje o rychlostech – skladník může např. pracovat s retrakem, kterému nějakou dobu trvá, než se zdvihne do požadované výšky a pak opět sjede dolů.

Vzhledem k objemu a rozmanitosti potřebných informací je zavedení automatizace tohoto způsobu shlukování poměrně náročné. Pokud však možnost automatického shlukování ve skladovém systému není, musí se provádět nějakým zaměstnancem – většinou vedoucím skladu –, což vede k navýšení objemu práce; tato práce navíc vyžaduje dobré kombinační schopnosti a je komplikována vyšší chybovostí. I přes tyto zápory se však shlukování na základě polohy pravidelně provádí, jelikož úspora času je znatelná [14, Page 11].

Pokud by se do stávajícího systému přidalo rozhraní pro zadání vzdáleností mezi jednotlivými umístěními, polohové shlukování by se dalo provádět automaticky. Tento návrh je však uživatelsky velmi nepřívětivý, jelikož by se pro každé umístění musela změřit vzdálenost ke všem ostatním – pro větší sklad o 5000 umístěních by toto bylo nemožné. Lepší řešení spočívá v tom, že uživateli dáme možnost načrtnout mapu skladu, do které zadá informace o poloze a rozměrech skladových umístění. Algoritmus v backendové části aplikace pak tato data může využít pro výpočet vzdáleností a následnou detekci objednávek vhodných ke shluknutí.

2.4.3 Shluknutí na základě obou faktorů

Největší smysl dává shluknout úkoly jak na základě vlastností objednávek, tak na základě polohy. Musíme totiž dbát např. na to, jaké množství zboží skladník unese – pokud navíc používá retrak, dokáže převážet pouze jednu paletu se zbožím. Shluknutí na základě polohy tak nedává smysl v situacích, kdy by skladník nedokázal unést všechno zboží. Dalším případem, kdy by bylo potřeba brát v potaz vlastnost objednávky, je např. doba jejího objednání – obecně je prioritou nejdříve expedovat nejstarší objednávky. Proto by např. nedávalo smysl shluknout velmi starou objednávku s velmi novou. Jako nejpraktičtější se jeví takové řešení shlukování, při kterém se dbá jak na vlastnosti objednávek, tak na polohu zboží ve skladu.

2.5 Mapa skladu

V předchozích kapitolách jsem došel k závěru, že největší smysl dává shlukovat úkoly jak na základě vlastností objednávek, tak na základě polohy zboží. Pro zjištění poloh zboží ve skladu je však potřeba znát polohu umístění, kde je uloženo. Proto bylo potřeba navrhnout a implementovat nástroj (neboli editor) na **tvorbu mapy skladu** jakožto rozšíření stávajícího skladového systému. Mým úkolem bylo vyřešit frontendovou část nástroje, tedy navrhnout nové uživatelské rozhraní. Pro další postup bylo potřeba provést důkladnou analýzu, která sestávala z těchto hlavních bodů:

- Důkladně prozkoumat sklady, na které cílí daný skladový systém, a podrobně prostudovat, co všechno se v nich nachází
- Přesně určit informace, které optimalizační algoritmy potřebují na shlukování úkolů
- Prozkoumat a určit cílové uživatele – stanovit persony
- Určit způsob, jak mapu znázornit a jak s ní interagovat – prozkoumat podobné aplikace a zvolit vhodný artefakt

Kapitola 3

Analýza

Po stanovení toho, že je potřeba navrhnout a implementovat nástroj pro tvorbu mapy skladu, bylo nutné provést důkladnou analýzu dané problematiky – co přesně se v mapě má zaznamenat a jakým způsobem ji vizualizovat. Při analýze jsem postupoval podle několika standardních metodik UCD (persony, výběr artefaktu), ale zároveň jsem musel doplnit další specifické analytické kroky kvůli unikátnosti zadání.

Analýzu jsem prováděl především **kvalitativně**. V případě takto unikátního problému by totiž kvantitativní průzkum nepřinesl mnoho výsledků[3]. Při analýze jsem použil těchto kvalitativních metod:

- **Rozhovory** s experty na skladovou doménu. Experti byli dostatečně obeznámení s oblastí skladové domény, díky své pracovní zkušenosti také znali charakteristiky skladových zaměstnanců a prostředí skladového systému *Atlantis*. Spolupracoval jsem se dvěma experty – Jiřím Hunkou a Oldřichem Malcem.
- **Terénní výzkum** v e-shopovém skladu v oblasti Ruzyně, na podzim 2021. Při výzkumu jsem měl možnost pozorovat skladové zaměstnance v praxi a uskutečnit rozhovor se skladovým vedoucím. Zároveň jsem si prohlédl strukturu skladu a dozvěděl se více o objektech, které se ve skladu obvykle nachází.

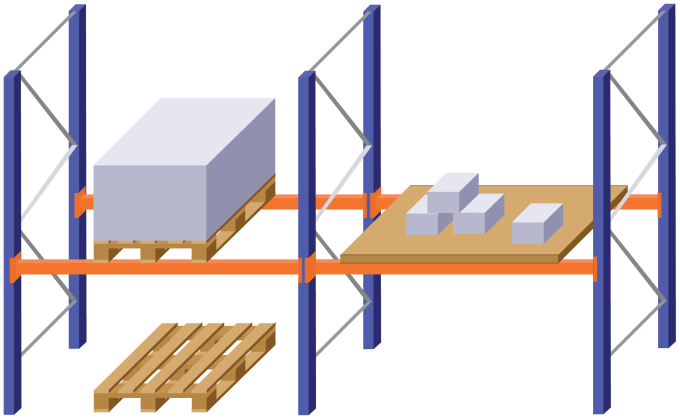
3.1 Bližší pohled na sklad

V kapitole 2 jsem identifikoval skladové pojmy a nejčastější procesy. To samotné však nestačilo k nabytí dostatečných vědomostí o tom, jak takový sklad vypadá a jaké jsou jeho důležité části. Stejně jako většina lidí jsem sice nějakou základní představu o vnitřním rozložení[4] skladu měl, tato představa však

[3] Tento závěr opodstatňuje např. absence podobných řešení při průzkumu v sekci 3.4

[4] z anglického *layout* – způsob, jakým jsou části něčeho uspořádány a umístěny

Obrázek 3.1: Paletový regál s paletami a policí



nebyla příliš užitečná, jelikož nebyla dostatečně podrobná a ověřená v praxi. Bylo tedy potřeba podrobněji analyzovat rozložení skladu a získat dostatek pravdivých, praxí ověřených informací, které by mi posloužily k další analýze a návrhu. Při sběru informací jsem použil poznatky jak z **rozhovorů s experty**, tak z **terénního výzkumu**.

3.1.1 Regály

Ve skladech, na které se náš systém zaměřuje, je regál jedním z nejčastějších skladových objektů. Existuje mnoho druhů regálů, zaměřím se však na dva nejčastější druhy:

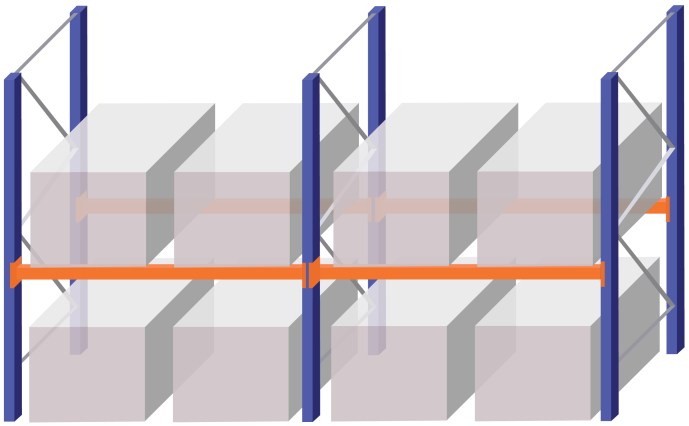
- **paletové**
- **policové**

U obou těchto druhů platí:

- nejsou pojízdné – stojí na místě[5]
- nejsou nijak automatizované, produkty se na ně ukládají manuálně (ať už ručně, či s použitím retraku)
- nevyužívají žádný speciální způsob ukládání – produkty jsou volně vodorovně položeny v patrech regálu

[5]pojízdné regály však existují a slouží ke snažšímu vyskladňování – jejich struktura však bývá často jednoduchá, a proto jsem se při analýze zaměřil na nepojízdné regály.

Obrázek 3.2: Paletový regál se dvěma patry a čtyřmi sloupci



Paletové regály se od policových liší tím, že jsou primárně určeny ke skladování palet zboží – obvykle většího množství zboží naskládaného a připevněného k paletě, většinou s využitím fólie a popruhů. Jelikož je paleta dostatečně pevná podložka, paletové regály na rozdíl od policových nedisponují policemi, ale pouze dvěma horizontálními nosníky, na které se paleta položí (obrázek 3.1).

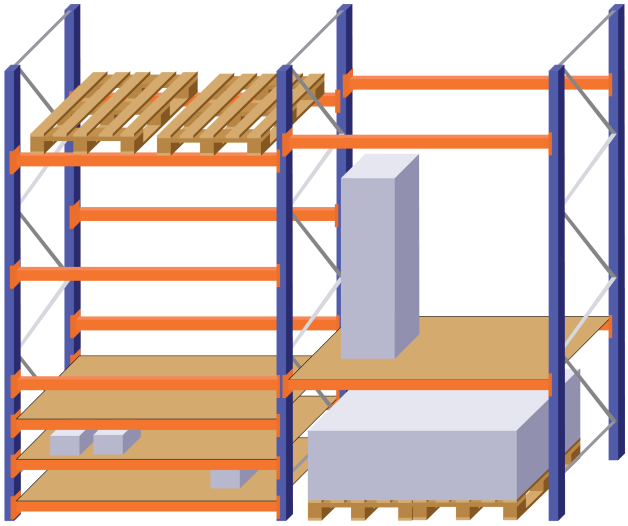
Co názvosloví těchto dvou typů regálů ještě více znejasní, je fakt, že patro paletového regálu můžeme jednoduše konvertovat na polici tím, že na nosníky položíme nějakou pevnou neděravou podložku (obrázek 3.1). Nemá proto dále smysl uvažovat o dvou rozdílných typech a budu dále uvažovat pouze o paletových regálech, přičemž policové jsou podmnožinou regálů paletových.

V paletovém regálu jsou rozložena **umístění** tak, aby se na každé umístění vešla paleta se zbožím nebo aby v daném místě byla police, kam je možné nějaké zboží položit.

Jelikož je vždy snaha sklad navrhnout co nejsymetričtější, jsou umístění po celém regálu rovnoměrně rozmístěna tak, aby vytvořila **patra** a **sloupce**. Všechna umístění v jednom patře se nacházejí vedle sebe a všechna umístění v jednom sloupci se nacházejí nad sebou (obrázek 3.2).

Ideální případ na obrázku 3.2 se praktikuje velmi často. Jednotlivá patra a sloupce se očíslují a umístění se pojmenují tak, aby šlo lehce poznat, v jakém sloupci a v jakém patře se nacházejí. Skladník může při vyhledávání nějakého umístění nejprve přijít ke správnému sloupci, a pak umístění najít ve správném patře. Tento jednoduchý způsob vyhledávání výrazně zrychluje většinu skladových procesů, zejména pokud je potřeba zacházet s retrakem, kterému nějakou dobu trvá, aby se vyzdvihl do určité výšky.

Obrázek 3.3: Konfigurovaný paletový regál se čtyřmi patry a čtyřmi sloupci



3.1.2 Konfigurace paletového regálu

Ideální případ rozložení umístění však může zkomplikovat fakt, že paletové regály jsou konfigurovatelné. Zatímco šířka oranžových horizontálních nosníků zůstává většinou v celém regálu stejná[6], nic nám nebrání v tom, abychom si tyto nosníky různě v regálu neodebírali, nepřidávali a nenastavovali do různých výšek. Tento fakt jsem si uvědomil při terénním výzkumu ve skladu v Ruzyni, kde jsem spatřil několik regálů, které se kvůli konfiguraci odklonily od ideálního případu.

Tento problém konfigurovatelnosti bylo potřeba roztřídit do jednotlivých podproblémů, abych každý z nich mohl vyřešit v budoucím návrhu. Celkem jsem identifikoval čtyři hlavní podproblémy:

3.1.2.1 Rozdělení patra

Při terénním výzkumu jsem si všiml, že v několika regálech bylo první (nejspodnější) patro rozděleno na tři patra, ve kterých se neskladovaly celé palety, ale už rozbalené kusy zboží (obrázek 3.3, vlevo dole). Tento sklad byl e-shopovým skladem, což znamenalo, že nejen skladoval zboží, ale také jednotlivé kusy balil a expedoval zákazníkům. Proto se v tomto skladu ukládaly nejen celé palety s ještě nerozbaleným zbožím, ale také již rozbalené kusy, které skladníci mohli ihned vyzvednout a zabalit.

[6] což určuje i vzdálenost mezi dvěma modrými rámy

3.1.2.2 Sloučení patra

Ve zkoumaném skladu jsem dále narazil na případ, kdy bylo potřeba skladovat krabice s velkými deštníky. Tyto krabice byly příliš dlouhé na to, aby se na nějakou polici položily vodorovně, bylo potřeba je tedy ukládat svisle – jelikož se ale svisle nevešly do jednoho patra, skladníci museli odebrat nosníky patra nad ním, tedy sloučit dvě patra do jednoho (obrázek 3.3, vpravo).

3.1.2.3 Sloučení umístění

Někdy je potřeba ukládat velké zboží, které se na šířku nevejde do jediného umístění, ale zabere např. tři. Pokud by se toto praktikovalo často, bylo by výhodné všechna tato umístění sloučit do jediného, jelikož při evidování tří rozdílných umístění bychom museli řešit, do kterého z nich bychom ve skladovém systému zaznamenali, že v něm ukládáme daný velký produkt, a pak bychom museli dávat pozor na to, že ostatní dvě umístění jsou sice v systému volná, ale ve skutečnosti jsou již zabraná velkým produktem (obrázek 3.3, vpravo dole).

3.1.2.4 Rozdílná výška pater vedle sebe

Na tento případ jsem ve zkoumaném skladu nenarazil, je však dobré si uvědomit, že dvě patra vedle sebe nemusí nutně být ve stejné výšce (obrázek 3.3, vpravo nahoře).

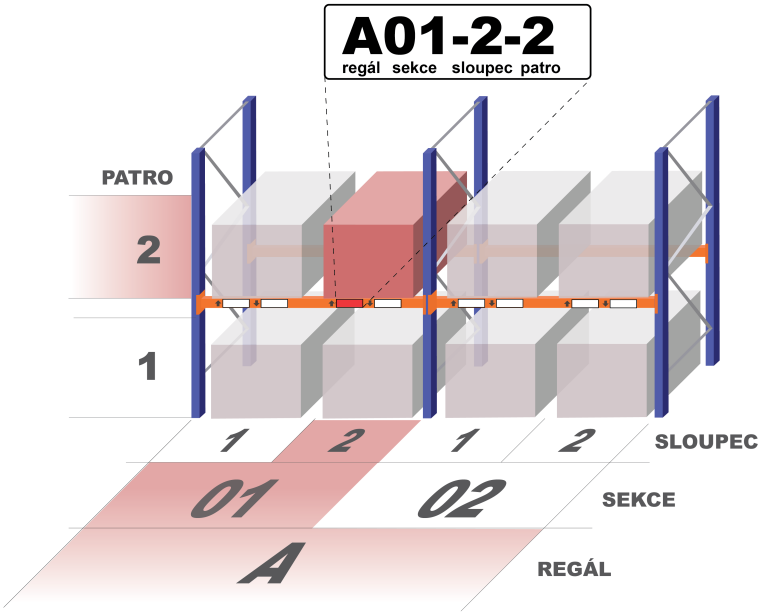
3.1.3 Jmenovací systémy

V předchozí sekci jsem již zmínil, že se umístění pojmenovávají podle regálu, sloupce a patra, ve kterém se nacházejí, aby byla snadno vyhledatelná. Každý sklad může využívat libovolný formát podle vlastního uvážení – všechny tyto formáty mají však společné to, že název umístění sestává z informací, které postupně využijeme k hlubšímu a hlubšímu prohledávání skladu, až se dostaneme ke konečnému umístění. Na začátku názvů tedy většinou bývají názvy regálů nebo nějakých velkých zón ve skladu, a pak se teprve v názvu uvede daný sloupec a patro regálu či jiný údaj, který skladníkovi pomůže dané umístění najít.

Jako příklad jednoho jmenovacího systému uvedu ten, který se běžně používá ve skladech, na které cílí náš skladový systém. Kromě názvu regálu, čísla sloupce a patra také používá číslo **sekce**, což je část regálu mezi dvěma modrými rámy. Ukázka systému je znázorněna na obrázku 3.4.

Otázkou je, jak se jmenovací systémy vypořádají s regály, které nemají ideální rozvržení a byly nějak nakonfigurovány, jako např. v sekci 3.1.2. Opět je třeba zdůraznit, že každý sklad může mít své vlastní řešení, zásadou však je, zachovat ve jménech nějaký pořádek – pokud jedno patro v jedné sekci rozdělíme na tři, většinou evidujeme v regálu stejný počet pater jako předtím,

Obrázek 3.4: Příklad pojmenování umístění dle jeho polohy ve skladu



jelikož v ostatních sekcích jsme žádná patra nepřidali. Místo zaznamenání nových pater spíše uvedeme, že se v daném patře nachází tři umístění nad sebou místo jednoho, a ve jménu umístění budeme nově evidovat **číslo umístění**. Cílem je nenarušit jmenovací systém tak, aby se dalo stále efektivně ve skladu vyhledávat.

3.1.4 Shrnutí

Poznatky z této sekce, které budou důležité v další analýze a návrhu, zde pro přehlednost shrnu:

- Regál je jedním z nejčastějších prvků ve skladu. Zaměřuji se na regál **paletový**, přičemž paletovým regálem lze jednoduše vyjádřit i regál **policový**.
- Paletové regály se dají nejprve rozdělit na **sekce**, což jsou prostory mezi dvěma rámy. Sekce se dále dají rozdělit na **patra** a **sloupce**, přičemž v jednom patře a v jednom sloupci se obvykle nachází jediné **skladové umístění.**
- Paletové regály se dají konfigurovat tím, že lze různě manipulovat s nosníky. Tím lze **dělit nebo shlukovat patra** v rámci jedné sekce nebo

měnit výšku patra. Zároveň se dají slučovat i samotná umístění, pokud na nich chceme skladovat velké produkty.

- Umístění se pojmenovávají podle jejich polohy ve skladu – používá se např. název regálu a čísla sloupce, patra nebo sekce.

3.2 Požadavky algoritmů

Potřeba editoru mapy skladu vyplynula z požadavku zaznamenávat informace, které optimalizační algoritmy využijí na návrh skladových optimalizací. Backendem optimalizací se v této době zabývali kolegové z týmu v rámci předmětu SP1, požadavky algoritmů proto byly vytyčeny po společných týmových schůzkách, jejichž náplní byla **diskuze a brainstorming**[7].

3.2.1 Informace o umístění

Skladová umístění byla těmi nejdůležitějšími objekty, které bylo potřeba do mapy zaznamenat. Algoritmus potřeboval určit vzdálenosti mezi jednotlivými umístěními a podle toho rozhodovat, jestli se vyplatí shluknout nějaké úkoly nebo provést nějaký přesun zboží.

Pro určení vzdáleností mezi dvěma umístěními jsme potřebovali znát **polohu** obou umístění, tzn. jejich souřadnice v mapě skladu. Tyto souřadnice by měly obsahovat i informaci o tom, v jaké výšce se umístění nachází, abychom mohli např. aproximovat dobu, kterou retrak potřebuje k dosažení umístění.

Došli jsme k závěru, že polohy nemusejí být zadávány s až tak velkou přesností a algoritmus zároveň nepotřebuje informaci o přesné reálné vzdálenosti v jednotkách SI – místo toho lze nejen polohu umístění, ale i výslednou vzdálenost mezi dvěma umístěními vyjádřit nějakou **fiktivní jednotkou**.

Druhou informací, kterou jsem potřeboval poskytnout backendu, byly **rozměry** jednotlivých umístění. Chtěli jsme také totiž implementovat optimalizaci, která by skladovým zaměstnancům nabízela přesuny zboží na lépe dostupnější místa. Při těchto optimalizacích by např. algoritmus pracoval s tím, jak moc je nějaké umístění zaplněné a jestli se do něj nedá umístit nějaký produkt navíc – tím by se ve skladu šetřilo místo. Rozměry jsem však na rozdíl od polohy potřeboval znát s **velkou přesností a ideálně v jednotkách SI**.

3.2.2 Průchozí prostor

Pro určení vzdáleností mezi dvěma body ve skladu bylo potřeba vědět, které prostory jsou ve skladu průchozí a které neprůchozí, aby algoritmus dokázal nalézt správnou trasu, která v realitě nevedla přes žádné překážky. Za první neprůchozí prostor se daly považovat **regály a umístění**. Druhým neprůchozím prostorem jsou zdi místnosti, ve které je sklad provozován – bylo potřeba

[7] Technika, při které se nápady vymýšlejí a probírají ve skupině.

určit nějaké **hranice skladu**. Samotná místnost může kromě obvodových zdí obsahovat i nějaké další zdi a další objekty, které nejsou průchozí a bylo by vhodné je do mapy zaznamenat – sklad může být také provozován ve více místnostech.

3.2.3 Zóny

Sklad je obvykle rozdělen na určité oblasti podle procesů, které v nich probíhají. Typickými příklady jsou balicí a expediční zóna. Vzdálenost skladového umístění od těchto zón se hodí při zjišťování jeho dostupnosti – je zřejmé, že objednávka bude připravena rychleji, když pro zboží, které chceme ihned zabalit, nemusíme chodit daleko. Díky údajům o poloze zón budou algoritmy schopné upřesnit dostupnosti umístění a následně navrhovat přesuny vysoce frekventovaného zboží na dostupnější umístění.

3.3 Persony

Při určování cílových uživatelů bylo otázkou, jestli se nám podaří vyvinout nástroj, který bude v rukou samotných skladových zaměstnanců produkovat dostatečně přesné a použitelné výsledky pro optimalizační algoritmy. Pokud by to nebylo možné, musel by se na návrhu mapy skladu podílet nějaký expert ze strany naší firmy, což by byl samozřejmě méně ideální případ. Proto bylo určení person jednoznačné – primární personou byl skladový zaměstnanec, který byl méně technologicky zručný, a „nouzovou“ sekundární personou byl firemní expert.

Při určování person jsem využil bohaté zkušenosti skladových expertů se skladovými zaměstnanci. Shledal jsem, že znalost expertů je dostačující ke správnému určení person, a nemusel jsem proto provádět kvantitativní průzkum pomocí dotazníků.

3.3.1 Primární persona: Standa

Je mu 44 let, pracuje jako skladový vedoucí ve skladu využívající náš skladový systém. V práci pro většinu administrativních úkonů používá levnější, méně výkonný notebook, s ne příliš kvalitním displayem. Každý den musí objednávky shlukovat ručně a ocenil by automatizaci procesu. Není si však jistý, jestli by mapu skladu dokázal správně navrhnout – v používání technologií totiž není nejzručnější a nejlépe mu jdou jednoduché technologické úkony, jako vyplňování tabulek a seznamů.

3.3.2 Sekundární persona: Petr

Je mu 28 let, pracuje jako vývojář na skladovém systému. Proto je také vysoce technologicky zručný a rád si zkouší různá nová uživatelská rozhraní. Je také

velmi pečlivý a vynalézavý, vždy si s nějakým rozhraním/nástrojem vyhraje, aby byl výsledek co nejlepší. Také se již vyzná v našem skladovém systému a zná jeho požadavky. Skladový systém je však stále ve vývoji, a proto by Petr radši trávil čas vývojem dalších důležitých funkcí, než někomu pomáhal s tvorbou mapy skladu.

3.3.3 Vysvětlení person

V naší aplikaci se zaměřím na dva odlišné póly z hlediska technologické zručnosti uživatelů. Chci vyvinout nástroj, který bude jednoduše koncipován – funkce by měly být přímočaré a v nejlepším případě založené na chování ostatních hojně používaných programů. Tím vyjdu vstříc primární personě – Standovi, který nepatří mezi ty nejzručnější a nechce používat komplikovaný nástroj. Vyhovění primární personě by mělo být samozřejmě prioritou. Pokud se to však nezdaří, existuje zde i sekundární persona – Petr, která je více technologicky zručná a vyskytuje se u ní větší šance na to, že správně navrhne mapu skladu, jelikož ví, jak skladový systém funguje i zevnitř.

3.4 Průzkum podobných aplikací

Před začátkem jakýchkoliv návrhů bylo vhodné podívat se po již existujících aplikacích, které danou problematiku do jisté míry řeší. Prvním logickým krokem bylo hledat aplikace, které byly přímo určené pro tvorbu mapy skladu – přínosné by bylo nejen vidět, jak se daná aplikace s problémem vypořádala a jak ho vyřešila, ale také, kolik takových aplikací vůbec existuje a jestli již neexistuje nějaká silná konkurence. Použil jsem vyhledávač Google a výrazy *warehouse map maker* či *warehouse map creator*.

Průzkum těchto specializovaných aplikací skončil neúspěchem – nepodařilo se mi najít jedinou volně dostupnou aplikaci, která splňovala daná kritéria. Tento výsledek byl však očekávaný, a to kvůli faktu, že většina skladových systémů je proprietární a neumožňuje přístup ke zkušebním verzím. Bylo potřeba rozšířit rámec vyhledávání. Zaměřil jsem se na grafické (2D i 3D) či tabulkové editory, které sice nebyly primárně určeny pro tvorbu mapy skladu, ale nabízely dostatek funkcionalit k znázornění většiny skladových objektů[8].

Pro výběr aplikací jsem využil osobní zkušenost, zároveň jsem vedl diskuzi s vývojovým týmem a skladovými experty. Použil jsem také vyhledávač Google a vyhledával jsem pod výrazy jako *diagram app* či *map maker*. Stanovil jsem si několik základních kritérií, které měly aplikace splňovat:

- Aplikace by měla disponovat rozhraním, které umožňuje nějak znázornit mapu skladu, do níž lze zadat alespoň některé informace ze sekce 3.2.

[8] Nehledě na intuitivitu celého procesu tvorby mapy

- Aplikace nemusí disponovat možností data mapy exportovat a získat v nějakém rozumném formátu – zajímal mě převážně způsob znázornění mapy v dané aplikaci.

Po zvětšení rozsahu vyhledávání jsem identifikoval větší počet aplikací, jejichž použitelnost bylo potřeba prozkoumat. Nielsen [16] identifikoval několik metod inspekce použitelnosti:

Heuristický průchod: neformální metoda, při které zkoumáme každý prvek rozhraní na základě předem stanovených heuristik.

Kognitivní průchod: formálnější metoda, při které kontrolujeme uživatelův průchod rozhraním. Zkoumáme, jestli jsou v rozhraní k dispozici možnosti, aby uživatel dosáhl svých cílů a jak pravděpodobné je, že si uživatel zvolí správnou možnost.

Pluralistický průchod: metoda, při které uživatelé, návrháři a vývojáři prochází určitými scénáři v rozhraní a diskutují o správnosti svých řešení.

Vzhledem k tomu, že jsem měl prozkoumat aplikace, které nebyly primárně určeny k tvorbě mapy skladu, rozhodl jsem se pro neformální metodu **heuristického průchodu**, kde jsem si stanovil tyto heuristiky:

- Možnost záznamu nutných dat – kolik informací potřebných algoritmy ze sekce 3.2 šlo do mapy zaznamenat?
- Efektivita návrhového procesu – jak dlouho trval návrh mapy, byly v rozhraní k dispozici nějaké funkce, které proces zefektivnily?
- Příjemnost návrhového procesu – používalo se rozhraní jednoduše a hladce?
- Pochopitelnost znázornění – jak pochopitelná byla výsledná mapa, bylo jasné, co jednotlivé prvky představují?

Každou heuristiku jsem ohodnotil na škále 0–10 bodů, kde 0 bodů znamenalo, že rozhraní danou heuristiku vůbec nesplnilo, a 10 bodů znamenalo, že rozhraní heuristiku splnilo úplně.

3.4.1 Diagrams.net

Tuto aplikaci jsem našel prostřednictvím vyhledávače Google. Diagrams.net je volně dostupná webová aplikace na tvorbu různých diagramů. Disponuje plátnem, na které můžeme pokládat různé útvary, které se vyskytují např. v UML diagramech. Napadlo mě, že bych mapu znázornil pohledem shora, a jednotlivé regály vyjádřil nějakým prvkem pro tvorbu diagramů – např. nějakým obdélníkem. Co mě zaujalo, bylo to, že aplikace si zakládala na přesnosti –

okolo plátna bylo k dispozici pravítko a při jakékoliv manipulaci s prvky diagramu se objevila mřížka, kterou jsem mohl využít k zarovnání prvku přesně na požadované místo. Aplikace však dále postrádala mnohé funkce pro tvorbu mapy, jako např. definice hranic.

Výsledné skóre: 13/40

- Možnost záznamu nutných dat: 2. Jelikož se jednalo o nástroj na tvorbu diagramů, rozhraní nedisponovalo dostatečnými možnostmi ani pro rozlišení základních skladových objektů.
- Příjemnost návrhového procesu: 5. Rozhraní poskytovalo některé funkce, které proces zpříjemňovaly – především funkce pravítka a mřížky.
- Efektivita návrhového procesu: 4. V rozhraní se objekty daly duplikovat a tím proces částečně zrychlit.
- Pochopitelnost znázornění: 2. Výsledek spíše připomíná nějaký diagram než mapu.

3.4.2 Microsoft Excel

Tato aplikace byla zvolena na základě diskuze vývojového týmu. Excel je tabulkový editor, typicky sloužící pro záznam různých administrativních dat, od výdajů až po různé informace o zboží. Tabulkové prostředí se však také může použít pro náčrt různých diagramů díky možnosti stylizace jednotlivých buněk, jako např. vybarvení či zvýraznění jejího okraje. Podobným způsobem by se mohla načrtnout i samotná mapa skladu.

Jelikož Excel pracoval s dvourozměrnou tabulkou, jediným možným praktickým způsobem, jak mapu znázornit, byl opět pohled seshora. Následně jsem využil různých možností stylování buněk tak, aby se v plánku rozlišily jednotlivé objekty ve skladu. Dokázal jsem vyjádřit hranice skladu, regály a různé zóny. Jakékoliv další informace, jako např. polohy umístění v rámci jednoho regálu a rozměry umístění, jsem však již zaznamenat nedokázal. Pro demonstraci jsem vytvořil mapu fiktivního skladu, zobrazenou na obrázku 3.5.

Stejně jako u aplikace Diagrams.net se jednalo o aplikaci, která nebyla primárně určena pro tvorbu map a plánků. Disponovala však mnoha možnostmi stylizace jednotlivých buněk, díky čemuž šlo vyjádřit více prvků skladu než jen samotný regál. Zároveň jsem při návrhu identifikoval pár intuitivních funkcí, které nějakým způsobem ulehčovaly návrh mapy:

- Kopírovat formát: Po označení jedné buňky lze celý její formát (tedy např. barvu a ohraničení) jedním kliknutím přenést na další buňku. Mohl jsem si tedy např. načrtnout jeden regál a touto funkcí rychle přidat několik identických regálů do mapy.

- Automatické vyplnění dat v buňkách: Po zvolení několika buněk lze přetáhnutím úchytu automaticky doplnit hodnoty do ostatních buněk. Pokud se tedy zvolí buňky s hodnotami 1, 2, 3, tak se do dalších buněk automaticky doplní hodnoty 4, 5, 6... Podobným způsobem by se mohly automaticky přiřazovat názvy regálům či umístěním – jmenný systém je často založen na podobném principu.

Výsledné skóre: 26/40

- Možnost záznamu nutných dat: 6. Dala se rozlišit většina skladových objektů. Šlo znázornit hranice skladu. Nešlo však dále zaznamenat informace o umístěních nad sebou, kvůli dvoudimenzionální tabulce.
- Příjemnost návrhového procesu: 5. Některé funkce tvorbě mapy vycházely vstříc, jiné se používaly těžkopádně – např. znázornění hranic skladu pomocí okrajů buněk.
- Efektivita návrhového procesu: 7. Rozhraní poskytovalo intuitivní funkce, které byly popsány výše. Proces návrhu se dal značně zrychlit. Některé funkce však návrh zpomalovaly (hranice skladu).
- Pochopitelnost znázornění: 8. Díky možnostem stylizace buněk mi mapa přišla přehledná a pochopitelná.

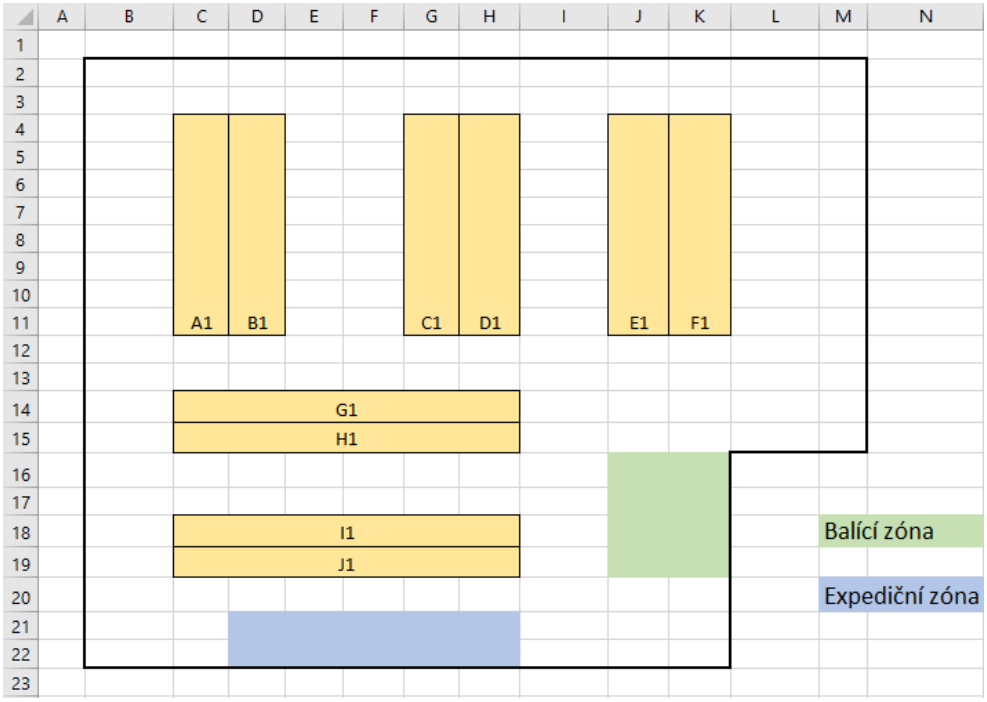
3.4.3 Minecraft

Tuto aplikaci jsem si zvolil na základě osobní zkušenosti. Minecraft je 3D počítačová hra na bázi sandbox (česky pískoviště) – hráč není omezován specifickými úkoly a má velkou míru volnosti při průzkumu, interakci a modifikaci herního světa. Svět je v Minecraftu složen z kostiček (bloků), které hráč může rozbíjet či pokládat. Hráč si tak může postavit v podstatě cokoliv – od jednoduchého domku až po např. napodobeninu Eiffelovy věže. Napadlo mě tedy, že by bylo zajímavým experimentem zkusit postavit alespoň část nějakého vymyšleného skladu v Minecraftu a celou zkušenost analyzovat. Část postaveného skladu je na obrázku 3.6.

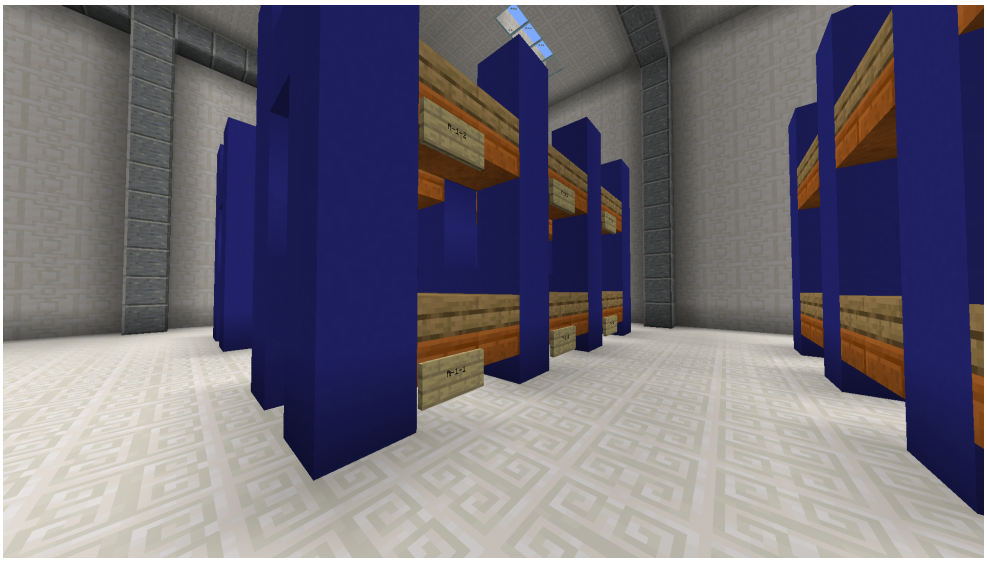
Výsledné skóre: 24/40

- Možnost záznamu nutných dat: 10. Ve světě jsem mohl vyjádřit celou skladovou místnost, a to nejen hranice místnosti (zdi), ale také podlahu a střechu. Znázornil jsem regály a rozlišil v nich jednotlivá patra. Umístění jsem v regálech vyznačil cedulemi, na které jsem napsal název umístění a jeho rozměr. Jednotlivé zóny ve skladu jsem vyjádřil změnou barvy podlahy. Daly se vyjádřit všechny informace pro algoritmy, a mnoho informací dokonce přesahovalo rámec požadavků.

Obrázek 3.5: Mapa fiktivního skladu vytvořená v programu Excel



Obrázek 3.6: Mapa fiktivního skladu vytvořená ve hře Minecraft



- Příjemnost návrhového procesu: 4. Ve hře jsem použil tzv. kreativní mód, ve kterém hráč může kromě chůze i létat a má k dispozici neomezený počet všech dostupných druhů bloků. Stavba byla z počátku příjemná a zábavná, velmi nízká efektivita procesu stavby však zkušenost znepříjemnila tím, jak dlouho trvalo postavit některé prvky (např. velkou zeď).

- Efektivita návrhového procesu: 0. Celý sklad jsem musel postavit opravdu blok po bloku. Žádný objekt (např. již postavený regál) nešlo duplikovat a každé umístění jsem musel označit novou cedulí. Dokázal jsem postavit jen malou část skladu (obrázek 3.6), i tak celý proces návrhu zabral okolo 30 minut.

- Pochopitelnost znázornění: 10. Díky 3D vizualizaci a pohledu z první osoby jsem si sklad mohl projít a prozkoumat jeho každou část jako ve skutečnosti. Každý objekt ve hře dostatečně připomínal jeho skutečnou podobu; dokonce jsem v regálu dokázal rozlišit jednotlivé oranžové nosníky a modré rámy.

3.4.4 Výsledky průzkumu

Dvě aplikace v průzkumu dosáhly dobrého bodového ohodnocení z hlediska splnění stanovených heuristik – jednalo se o aplikace Excel a Minecraft. Každá z těchto aplikací měla velmi odlišný artefakt – Excel pracoval s dvoudimenzionální tabulkou, zatímco Minecraft pracoval s trojdimenzionálním prostorem. Mapa znázorněná v Minecraftu byla daleko pochopitelnější a dalo se do ní zaznamenat více informací, proces návrhu však byl velmi těžkopádný. V aplikaci Excel nastal opačný případ – nedaly se zaznamenat všechny potřebné informace a mapa byla méně (avšak stále dostatečně) pochopitelná, navrhovala se však daleko rychleji a jednodušeji.

Toto byla dobrá ukázka dvou různých úrovní granularity – Excel měl granularitu příliš malou a dvoudimenzionální tabulka nestačila na záznam všech informací; Minecraft měl granularitu až moc velkou a informací šlo zadat tolik, že celý návrhový proces byl velmi pomalý. Díky průzkumu jsem si uvědomil, že bude potřeba nastavit granularitu informací v mé aplikaci tak, abych od uživatele získal všechny potřebné informace, ale zároveň ho nezatížil dalšími zbytečnými požadavky.

V aplikacích jsem dále identifikoval intuitivní funkce, které zrychlovaly či zpříjemňovaly proces návrhu mapy. Tyto funkce zohledním při návrhu.

3.5 Výběr artefaktu

Posledním krokem analytické části bylo vybrat vhodný typ artefaktu. Musel jsem vzít v potaz všechny předchozí části analýzy – vědomosti o rozložení

skladu, požadavky algoritmů, vlastnosti budoucích uživatelů a již používaná řešení.

Při výběru jsem přihlédl k artefaktům aplikací, které při heuristickém průchodu skončily s dobrým bodovým ohodnocením – nejlepší bodové ohodnocení získala aplikace Excel, jejíž artefakt mě zaujal tím, že přes velmi minimalistické znázornění mapy (pomocí dvoudimenzionální tabulky) se dala do mapy zadat většina potřebných informací, zároveň byl proces návrhu mapy poměrně rychlý a výsledná mapa byla dostatečně pochopitelná.

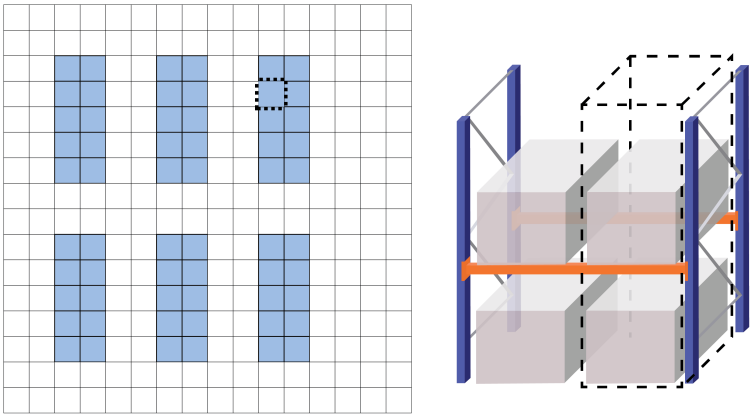
Druhou výhodou aplikace Excel a jejího artefaktu bylo to, že dle skladových expertů byla ve skladové doméně často používána a mnoho cílových uživatelů již bylo v nějaké míře seznámeno s jejím rozhraním. Pokud bych tedy artefakt aplikace Excel vylepšil tak, aby umožňoval záznam všech potřebných informací, mohl bych vytvořit použitelné rozhraní.

Dle dosavadních poznatků jsem určil artefakt s těmito vlastnostmi:

- Mapa bude reprezentována dvoudimenzionální tabulkou zobrazující pohled seshora
- Každé buňce půjde přiřadit typ – buňka je buď průchozí, nebo se v ní vyskytuje nějaký počet skladových umístění. Jedna buňka by tedy mohla reprezentovat např. jeden sloupec regálu (obrázek 3.7)
- Jednotlivé buňky, které by symbolizovaly sloupec regálu, by se daly upravovat – uživatel by si mohl navolit počet umístění ve sloupci, jejich rozměry, názvy a výšku, ve které se umístění nacházelo. Tím by se dala vyjádřit i většina způsobů konfigurace paletového regálu, které jsem identifikoval v sekci 3.1.2.

Tento artefakt se z počátku jevil jako výborné a jednoduché řešení, které by nemělo být implementačně náročné a zároveň mně i skladovým expertům připadalo jednoduché na pochopení. Přešel jsem tedy do fáze prototypování, ve které bylo třeba ověřit, jestli jsem si artefakt zvolil dobře, a na základě toho pokračovat v dalším návrhu a implementaci.

Obrázek 3.7: První artefakt



Kapitola 4

První prototyp

S dostatečnými teoretickými i praktickými podklady a vybraným artefaktem jsem byl připraven přistoupit k návrhu uživatelského rozhraní. Tato kapitola popisuje proces postupného vývoje prvního prototypu, který je na závěr kapitoly zhodnocen z hlediska použitelnosti.

4.1 Sběr požadavků

Před začátkem práce na návrhu a následném prototypování bylo důležité shrnout a kategorizovat požadavky kladené na nové rozhraní. Z hlediska softwarového inženýrství se požadavky většinou kategorizují na funkční a nefunkční. Jelikož jsem se zabýval také použitelností daného software, nabízelo se požadavky kategorizovat jemněji pomocí metody FURPS, která definuje tyto kategorie požadavků [17]:

- **F (functionality)** – funkčnost
- **U (usability)** – použitelnost
- **R (reliability)** – spolehlivost
- **P (performance)** – výkon
- **S (supportability)** – podporovatelnost/rozšiřitelnost

V prvotních fázích návrhu a vývoje jsem si nebyl jist všemi funkcemi, které jsem do rozhraní chtěl přidat. Proto je tento prvotní seznam požadavků koncipován tak, že se nezabývá již zvoleným artefaktem a všechny požadavky by měly být splněny při užití jakéhokoliv artefaktu.

F1 – Zadávání dat pro optimalizační algoritmy

V novém rozhraní by mělo být umožněno do mapy zadat všechny informace, které jsem zmínil v sekci Požadavky algoritmů (sekce 3.2), tedy:

- trojdimenzionální souřadnice umístění – tzn. x, y, z
- rozměry a názvy umístění
- informace o průchozím a neprůchozím prostoru
- hranice skladu
- důležité zóny – alespoň expediční a balicí zóna

Priorita: vysoká

Složitost: vysoká

U1 – Podpora hromadného zadávání dat

Ve skladech, na které skladový systém *Atlantis* cílí, se může vyskytovat řádově tisíce umístění v několika desítkách regálů. Nové rozhraní proto musí podporovat hromadné zaznamenání dat do mapy pomocí předem určených kritérií. Primárně se jedná o názvy a rozměry umístění, u názvů by bylo vhodné využít daný jmenovací systém skladu.

Priorita: vysoká

Složitost: vysoká

U2 – Přesné znázornění reálného skladu

V rozhraní by mělo být umožněno pojmenovat každé umístění přesně podle skutečnosti (možnost využít stejný jmenovací systém jako ve skladu) a zaznamenat všechny konfigurace regálu zmíněné v sekci 3.1.2.

Priorita: vysoká

Složitost: vysoká

U3 – Konzistence vzhledu se skladovým systémem *Atlantis*

Vyplývá z požadavku S1 – jelikož nové rozhraní bude vloženo do rozhraní skladového systému *Atlantis*, rozložení a vzhled nového rozhraní by nemělo kolidovat s rozhraním zbytku aplikace.

Priorita: nízká

Složitost: střední

P1 – Dostatečný výkon pro vykreslování a interakci

Rozhraní by mělo zvládat vykreslovat celou mapu skladu a umožnit uživatelskou interakci s dostatečně malou dobou odezvy. Pro menší úkony, kde by mělo rozhraní reagovat okamžitě, by tato doba neměla překročit 0,1s (doporučený limit dle Nielsena [18]).

Priorita: vysoká

S1 – Vložení do frontendu skladového systému *Atlantis*

Nové rozhraní bude vloženo do frontendu skladového systému *Atlantis*. To znamená, že bude potřeba pracovat s již existujícím kódem a zvolit kompatibilní technologie.

Důvody pro tento požadavek byly dva:

- Pro zákazníky je lepší mít všechnu funkcionalitu přístupnou pod jednou aplikací.
- Se samostatnou aplikací by se muselo vyřešit několik problémů, které již jsou ve frontendu skladového systému vyřešeny, jako např. zabezpečení, přihlašování a ověřování uživatele.

S2 – Webové rozhraní

Tento požadavek vychází z předchozího požadavku. Jelikož bylo rozhraní skladového systému webovou aplikací, muselo rozšíření tohoto rozhraní být také webové.

S3 – Cílové zařízení

Vzhledem k očekávanému rozsahu funkcí nového rozhraní byl hlavním cílovým zařízením určen **osobní počítač**. Většina osobních počítačů disponují dostatečně velkou obrazovku na to, aby zobrazila rozhraní s mnoha prvky. Počítače jsou zároveň vybaveny zaměřovacími zařízeními (myš, trackpad), která umožní dobrou úroveň interakce s mapou.

S vedoucím práce jsem vedl diskuzi o tom, jestli by neměl být nástroj přístupný i na mobilních a tabletových zařízeních. Shodli jsme se na tom, že by to v této fázi nadměrně zkomplikovalo práci; uznali jsme však, že do budoucna by bylo výhodné mapu na mobilních zařízeních alespoň zobrazit (např. pro účel orientace ve skladu).

4.2 Volba pracovního postupu

Při standardním postupu prototypování je vhodné řídit se určitými kroky – nejdříve vytvořit lo-fi prototyp, na kterém lze postupně vyladit základní funkcionalitu, a dále přejít na hi-fi prototyp, kde se začneme zabývat i vzhledem rozhraní.

Jelikož bylo zadané rozhraní z mého pohledu komplexní, přešel jsem přímo k vývoji hi-fi prototypu. Rozhraní jsem potřeboval mít na vysoké úrovni funkčnosti, abych mohl testovat koordinaci všech možných nástrojů a panelů rozhraní.

Došel jsem k závěru, že nejlepším krokem by bylo prototyp nakódovat, abych dokázal přesně vyjádřit všechny funkcionality. Nic mi nebránilo prototyp již rovnou integrovat do frontendu skladového systému *Atlantis*, čímž jsem dané rozhraní zároveň implementoval. Přešel jsem tedy do fáze, kdy jsem spojil návrh, prototypování a implementaci do jedné.

Tato strategie byla velice riskantní, z mého pohledu však návrh nešlo vést jiným způsobem – pro vyjádření všech funkcí v prototypu jsem jej musel vyvíjet na vysoké úrovni funkčnosti a nejjednodušším řešením se jevilo jej rovnou implementovat.

Při výběru vývojové metodiky se přirozeně nabízel **iterativní vývoj**, při kterém jsem se při každé iteraci vždy snažil dodat funkční prototyp, pokaždé s přidanými či změněnými funkcionalitami a komponentami. Metodika Vodopád by v tomto konkrétním případě vůbec nefungovala kvůli nestabilitě požadavků.

4.3 Volba technologie a grafického vzhledu

Jelikož jsem se rozhodl prototyp rovnou integrovat do frontendu skladového systému *Atlantis*, bylo potřeba stávající frontend analyzovat a zajistit, abych použil kompatibilní technologie a ideálně nové rozhraní navrhl ve stejném grafickém stylu.

4.3.1 Volba frameworku

Frontend skladového systému *Atlantis* se jmenuje *Swordfish* a původně byl vytvořen Ing. Oldřichem Malcem v rámci jeho diplomové práce [19]. Aplikace frontendu používala framework Vue.js. Jelikož jsem implementoval rozšíření frontendu, bylo vhodné použít stejný Javascriptový framework, neboť se nedoporučuje v rámci jednoho projektu používat více frameworků [20]. Po inspekci mi framework přišel dostačující pro účely tvorby nového rozhraní. Zvolil jsem tedy framework **Vue.js**.

Vue.js

Vue.js (dále také Vue) je Javascriptový framework pro tvorbu uživatelských rozhraní. Staví na standardním HTML, CSS a Javascriptu a poskytuje deklarativní programovací model založený na komponentách, kterým lze efektivně vyvíjet uživatelská rozhraní, ať už jednoduchá, nebo složitá [21].

Vue staví na reaktivitě – automaticky sleduje změny stavu Javascriptu a efektivně aktualizuje DOM, když dojde ke změnám. Dále využívá již zmíněného *component-based* modelu, což znamená, že jednotlivé části rozhraní se logicky člení do samostatných prvků – komponent. Obvykle je tedy rozhraní reprezentováno stromem vnořených komponent, kde kořenová komponenta celou aplikaci zapouzdřuje.

Principu komponent jsem využil při implementaci prototypu, kde byl nástroj na tvorbu mapy skladu zapouzdřen jednou kořenovou komponentou, která se poté dala vložit kamkoliv do aplikace. Nové rozhraní tím bylo přenosné a také jsem na něm mohl pracovat odděleně od ostatních částí aplikace.

V prvním prototypu jsem použil čistý Vue bez dalších podpůrných knihoven (kromě grafických), které ve frontendu použil pan Malec – např. Vuex či Router. Z počátku jsem se totiž zabýval znázorněním mapy a jednotlivých prvků rozhraní. V pozdější fázi projektu – při vývoji na druhém prototypu – se nové rozhraní stávalo čím dál více komplexní a k použití podpůrných knihoven jsem nakonec přistoupil. Proto budou popsány až v kapitole Druhý prototyp.

4.3.2 Volba grafických knihoven

Při volbě grafických knihoven jsem uvažoval dvě různá užití knihovny:

- Základní prvky rozhraní (tlačítka, panely, ovládací prvky...)
- Okno zobrazující mapu

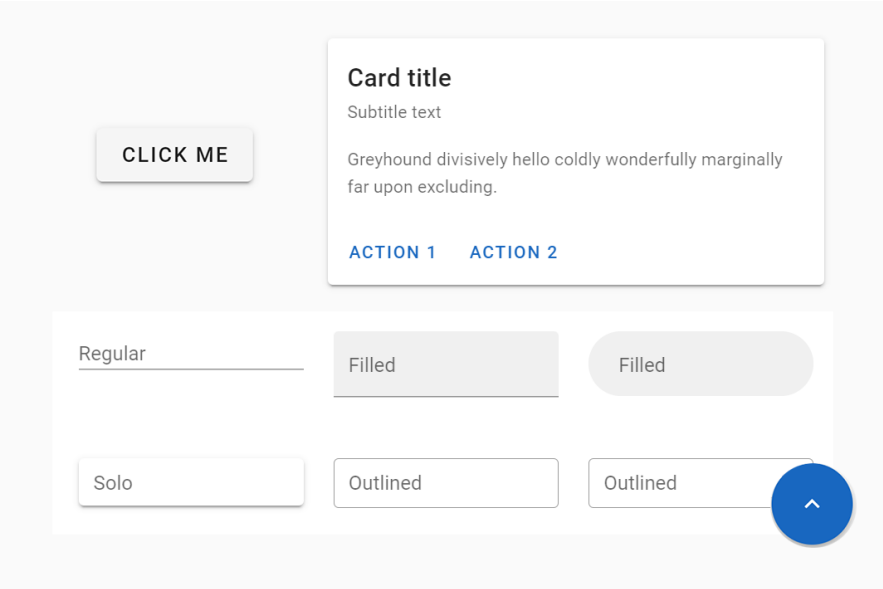
Aplikace Swordfish používala grafickou knihovnu Vuetify a většina rozhraní aplikace byla sestavena z prvků, které knihovna poskytuje. Proto jsem se pro základní prvky nového rozhraní rozhodl použít také knihovnu Vuetify, aby grafický styl nového rozhraní odpovídal stylu celé aplikace.

Při volbě grafické knihovny pro okno zobrazující mapu záleželo na komplexitě zvoleného artefaktu. Jelikož byl artefakt realizovatelný pomocí dvoudimenzionální tabulky, usoudil jsem, že není třeba využít grafické knihovny a postačil jsem si s čistou HTML tabulkou a CSS stylováním.

4.3.3 Grafický design

Spolu s výběrem grafické knihovny Vuetify jsem zvolil i styl grafického designu, jelikož knihovna Vuetify fungovala na principu *Material designu* [22]. Material design je sada designových pravidel vytvořená společností Google v roce 2014

Obrázek 4.1: Ukázka různých UI prvků knihovny Vuetify, založené na principu Material designu



[23]. Od té doby jsou pravidla postupně revidována a upravována a stala se velmi populární volbou v mobilních Android aplikacích [24].

Ač je Material Design jednoduchý na užití tím, že řeší mnoho problémů v UI a návrhář se nemusí starat např. o konzistenci vzhledu jednotlivých prvků, dostává se mu i kritiky [25]. Jednou z nevýhod je např. to, že výsledné aplikace příliš připomínají aplikace Google, které užití Material Designu preferují – rozhraní tak ztrácí na jedinečnosti. Dalším negativem je např. kontroverzní *floating button* – tlačítko nacházející se v dolním pravém rohu obrazovky, které při scrollování nezmizí a zůstává na místě, a tím může překrývat některé prvky UI (ukázka na obrázku 4.1).

Díky volbě knihovny Vuetify jsem tedy nemusel vymýšlet nový grafický design, jelikož jsem již pracoval s Material Designem. Ve Vuetify jsem nakonec našel většinu požadovaných prvků, které jsem do rozhraní chtěl vložit. Práce s knihovnou však nebyla ideální – mnoho prvků knihovny dávalo smysl použít jen na mobilních zařízeních, která se neřadila mezi cílové platformy mého nástroje. Také bych ocenil větší komplexitu knihovny – často jsem pro různé prvky potřeboval další funkcionalitu navíc.

Vzhled rozhraní se v průběhu prototypování dramaticky měnil, proto budu o rozhodnutích z hlediska designu pojednávat v kapitolách popisujících jednotlivé iterace. Po celou dobu jsem se však držel principů Material Designu.

Obrázek 4.2: První iterace prototypu, vytvořená ve spolupráci s kolegou Malcem

4.4 Rozhraní první iterace

Přišlo na řadu navrhnout a implementovat první iteraci prototypu. Jelikož toto byla moje první zkušenost s komplexnějším Javascriptovým frameworkem a mé znalosti jazyků webového frontendu (HTML, CSS, Javascript) nebyly na nejvyšší úrovni, pomohl mi Ing. Oldřich Malec s vytvořením jednoduchého prototypu založeného na předem zvoleném artefaktu. Díky první iteraci jsem mohl analyzovat již existující kód a spolu s ním se naučit správný workflow s frameworkem Vue.

Na obrázku 4.2 je vidět celé rozhraní první iterace. Disponovalo již mnoha ovládacími prvky a koncepty, které je nutné vysvětlit:

4.4.1 Okno s mapou

Okno v dolní části obrázku 4.2 vyobrazuje mapu skladu formou tabulky. Každá buňka měla určitý **typ**. Bylo na výběr ze 3 typů:

- **Regál** (červená barva): Na tomto místě se nachází nějaký počet skladových umístění nad sebou. Může se tedy jednat o sloupec regálu, skříň či pouze umístění na zemi.
- **Průchozí** (modrá barva): Tento prostor je volně průchozí a optimalizační algoritmy jej vezmou v potaz při hledání a měření tras.
- **Překážka** (šedá barva): Tato místa jsou neprůchozí.

4.4.2 Jmenovací systém

V buňkách typu „regál“ se umístěním automaticky přiřazovala jména podle počátečních symbolů, které si uživatel zvolil v horní části rozhraní. Formát jména umístění byl ve tvaru *řádek tabulky – sloupec tabulky – patro*. Ve všech buňkách typu „regál“ byl dále automaticky přiřazen stejný počet pater, který uživatel také mohl ovládat v horní části rozhraní.

4.4.3 Další ovládací prvky

Dále se v rozhraní vyskytovala dvě tlačítka na přidání nových řádků a sloupců do tabulky a paleta s barvami náležícími jednotlivým typům buněk. Po zvolení určitého typu jej pak uživatel mohl přiřazovat buňkám na mapě pomocí levého kliknutí myši.

Po pravém kliknutí myši na buňku s umístěním se zobrazilo vyskakovací okno (obrázek 4.3), kde uživatel mohl měnit typ a název buňky a dále názvy všech umístění v buňce. Tyto názvy se však automaticky přepsaly, pokud uživatel změnil počáteční symboly jmenovacího systému nebo počet pater.

4.5 Hodnocení první iterace

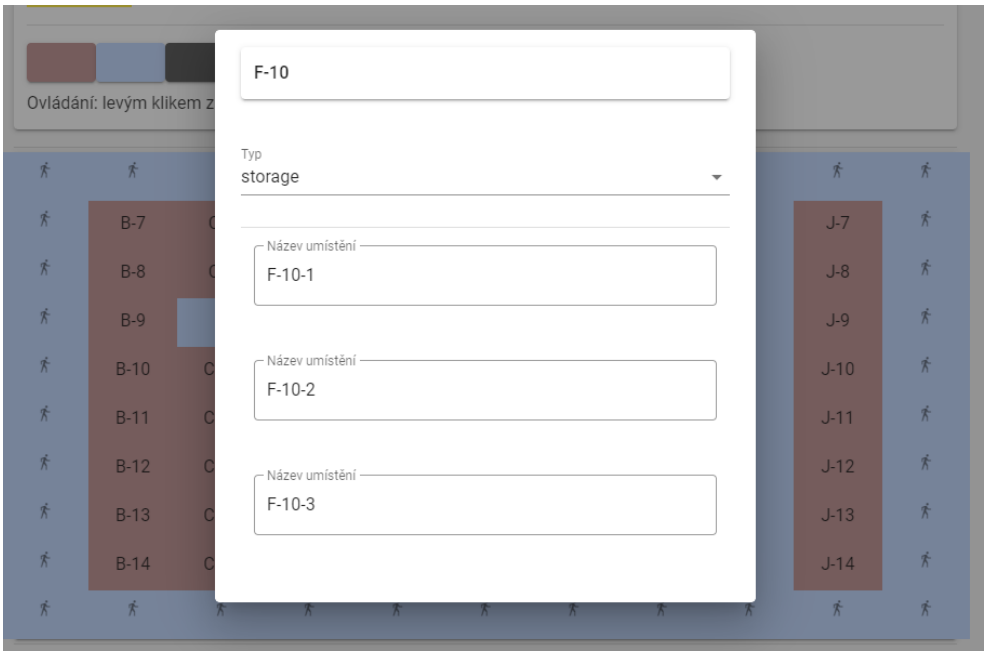
První iterace sice měla být ukázkou toho, jak by rozhraní mělo vypadat, přesto mi však dala dostatečné podklady pro to, abych si uvědomil, které další funkce budu muset implementovat či změnit, aby rozhraní bylo použitelnější. V první iteraci jsem identifikoval tyto problémy a nutné úpravy:

4.5.1 Ideální případ

Vzhledem k tomu, že se počet pater nastavoval globálně, tzn. všem buňkám najednou, dal se v tomto rozhraní vyjádřit pouze perfektně symetrický sklad, který měl v každém sloupci každého regálu stejný počet pater. Vůbec se nedaly vyjadřovat konfigurace paletového regálu, jelikož nešlo přidávat/odebírat patra jednotlivým buňkám.

- **Řešení:** Umožnit každé buňce typu „regál“ mít svůj vlastní počet pater, který by nebyl globálně nastavený. Dále by bylo vhodné umožnit přidat

Obrázek 4.3: Vyskakovací okno po pravém kliknutí myši



do jednoho patra více než jedno umístění – tím by šlo vyjádřit některé konfigurace, jako např. rozdělení jednoho patra.

4.5.2 Nestandardní jmenovací systém

Použitý jmenovací systém nereflektoval praxi – každé umístění pojmenoval podle symbolu řádku a sloupce tabulky. Uživatel by proto musel každé umístění ručně přejmenovat. Při změně jednoho počátečního symbolu jmenovacího systému se všechna umístění přejmenovala a uživatel by tak ztratil již zadané hodnoty.

- **Řešení:** Umožnit lepší nastavení jmenovacího systému a dát uživateli větší kontrolu nad tím, kdy a kde se spustí automatické přiřazení názvů, aby se uživateli nepřepsaly již zadané hodnoty.

4.5.3 Ovládání tabulky

Chyběly funkce pro ovládání tabulky, jako mazání sloupců/řádků nebo výběr více buněk najednou.

- **Řešení:** Přidání těchto chybějících prvků.

4.5.4 Chybějící informace

Do mapy nešlo zadat informace o rozměrech jednotlivých umístění a jejich vzdáleností od země.

- **Řešení:** Umožnit tyto informace zadat rozšířením rozhraní o nové ovládací prvky.

4.5.5 Pohled seshora

Tím, že mapa znázorňovala pohled na sklad přímo seshora, nedokázal uživatel na první pohled přijít na to, která umístění se pod danými buňkami skrývala. Uživatel si samozřejmě mohl zobrazit detail buňky pravým kliknutím, tato akce však fungovala pouze na jednu buňku – nešlo tak zobrazit celý regál ze strany, což byl pro skladové zaměstnance pohled, na který byli nejvíce zvyklí.

- **Řešení:** Umožnit pohled ze strany na regál alespoň tím, že by šlo zvolit více buněk najednou a zobrazit všechna umístění ve výběru. Tato umístění by se dále dala roztřídit, např. podle patra, ve kterém se nacházela.

4.5.6 Hranice skladu

Tabulka znázorňující mapu skladu byla vždy obdélníkového tvaru. Pokud bychom tedy okraje tabulky pokládali za hranice skladu, mohli bychom tak vyjádřit hranice pouze obdélníkových místností. Skladová místnost však také může být např. ve tvaru písmene L – takové hranice se v mapě již vyjádřit nedaly, což by mohlo mít negativní dopad na optimalizační algoritmy (obrázek 4.4).

- **Řešení:** Uživatel by mohl prostor za hranicemi skladu označit jako neprůchozí. Tímto by se daly definovat jakékoliv hranice skladu. Bylo však otázkou, jestli uživatel na takové řešení přijde sám a jestli mu tuto informaci není potřeba nějakým způsobem sdělit.

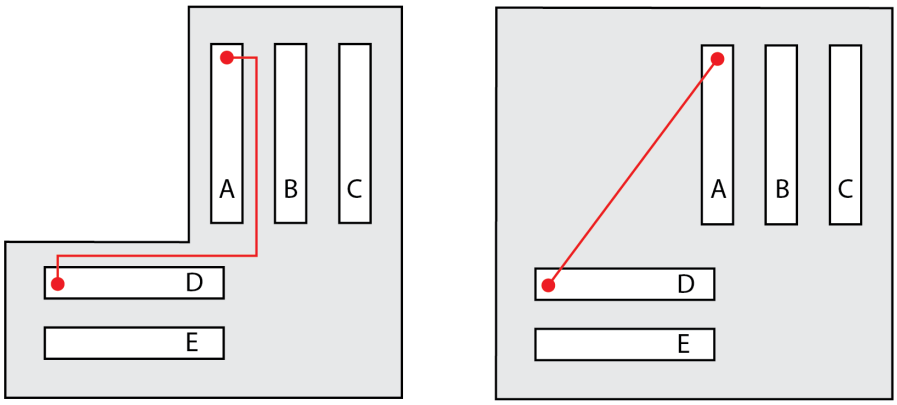
4.5.7 Závěr

V rozhraní se sice vyskytovala řada problémů, bylo však pravděpodobné, že je dokážu vyřešit ve druhé iteraci a rozhraní by se tak stalo dostatečně použitelným. Přesunul jsem se tedy do druhé iterace.

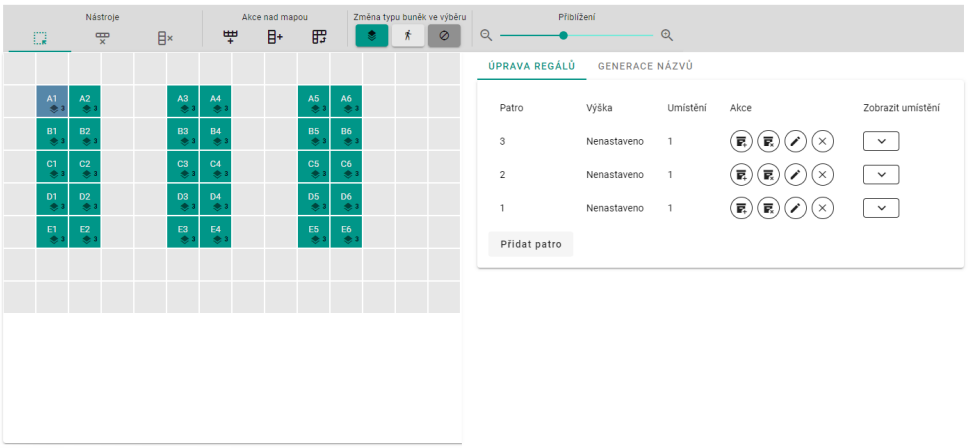
4.6 Rozhraní druhé iterace

V této sekci popíšu nový vzhled rozhraní a shrnu všechny nové funkcionality, které jsem do rozhraní přidal. Cílem bylo vyřešit většinu problémů v první

Obrázek 4.4: Příklad, kdy špatně nadefinované hranice mohou zkreslit informace o vzdálenostech trasy mezi dvěma body ve skladu



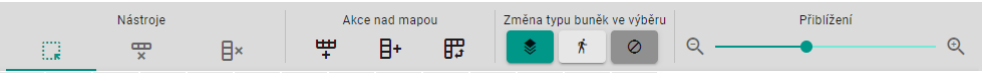
Obrázek 4.5: Celé rozhraní druhé iterace



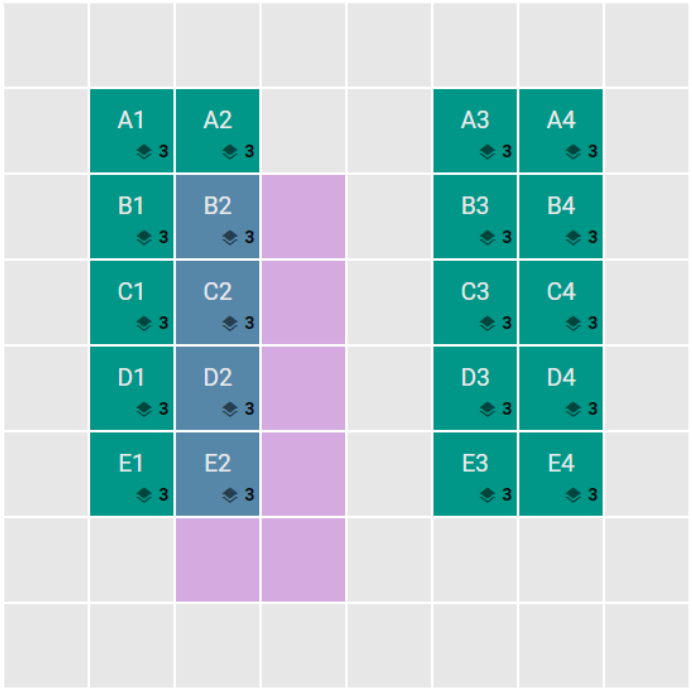
iteraci, zároveň jsem však chtěl zlepšit rozhraní i po grafické stránce a roztřídit určité funkce do různých panelů.

Nejprve jsem se rozhodl semknout všechny ovládací prvky k sobě a vytvořit tak rozhraní, které se vešlo na celou obrazovku – uživatel tím pádem nemusel na webové stránce scrollovat, čímž by se lehce ztratil. Další výhodou kompaktního rozhraní bylo, že se zkrátily vzdálenosti mezi jednotlivými ovládacími prvky, čímž jsem přispěl k lepší ergonomii. Celé rozhraní je vidět na obrázku 4.5.

Obrázek 4.6: Panel nástrojů druhé iterace



Obrázek 4.7: Ukázka zvýrazněných buněk po selekci



4.6.1 Panel nástrojů

Při druhé iteraci jsem se snažil udělat z nástroje funkční tabulkový editor. Všechny dosavadní nástroje pro ovládání tabulky jsem přesunul do jediného panelu, který se nacházel v záhlaví celého rozhraní. Opět zde byla snaha, aby se tento panel nacházel co nejblíž oknu s mapou, aby uživatel nemusel dlouho cestovat myší při manipulaci s tabulkou. Oproti první iteraci přibylo více prvků pro ovládání tabulky. Všechny tyto prvky a funkce nyní popíši.

Selekce Pro vyřešení problému 4.5.5 bylo potřeba implementovat nástroj, kterým by uživatel mohl zvolit více buněk najednou. Tento nástroj jsem pojmenoval *Selekce*. Fungoval na bázi **click and drag** – neboli klikni a táhni. To znamená, že při prvním kliknutí myši si uživatel vybere jeden roh obdélníku, pak myš přesune k druhému rohu obdélníku a poté myš pustí. Zajistil jsem, aby se vybrané buňky po celou dobu procesu selekce zvýrazňovaly (obrázek 4.7).

Odstranění sloupců a řádků Tento nástroj jsem rozdělil na dva – odstra-

není sloupce a odstranění řádku. Po výběru jednoho z těchto nástrojů uživatel najede myší na část tabulky, kterou by chtěl smazat, a levým kliknutím ji může odstranit. Stejně jako u nástroje Selekce i zde byla snaha podávat informace uživateli zvýrazněním buněk – poté, co uživatel najel myší na určitý sloupec/řádek, zvýraznil se mu červeně, což symbolizuje to, že se po kliknutí odstraní.

Změna typu buněk Tuto funkci jsem implementoval podobně jako paletu barev v mnoha grafických editorech. Po kliknutí na jeden z typů v „typové paletě“ se daný typ přiřadí všem buňkám, které byly vybrány nástrojem Selekce.

Přiblížení Funkce přiblížení se dle mé zkušenosti vyskytuje prakticky v jakémkoliv grafickém editoru. Implementována je i v Excelu. Tento fakt však nebyl jediným důvodem, proč tuto funkci do rozhraní přidat – při větších velikostech mapy (např. o 50 řádcích a 50 sloupcích) bylo potřeba mapu dostatečně oddálit, aby se celá vešla do zobrazovacího okna. Funkci jsem implementoval jednoduchým posuvníkem s ikonkami (obrázek 4.6, na pravé straně).

4.6.2 Okno s mapou

Okno s mapou jsem upravil tak, aby reagovalo na výběr a použití nástrojů z horního panelu, např. změnou barvy vybraných/právě mazaných buněk. Také jsem se pokusil částečně vyřešit problém 4.5.5 tím, že jsem přidal ke každé buňce typu „umístění“ číslo znázorňující počet pater v buňce.

4.6.3 Postranní panel

Postranní panel jsem přidal do pravé části rozhraní. V horní části panelu jsem umístil menu, kde se daly překlikávat jednotlivé záložky poskytující různé funkcionality.

Úprava regálu Tato záložka zobrazovala přehled všech umístění ve všech zvolených buňkách seskupených podle patra (obrázek 4.8). Nad každým patrem šlo provést několik operací:

- **Přidat/odebrat umístění:** V každé buňce ve výběru v daném patře se přidá nebo odebere umístění.
- **Přiřadit rozměr všem umístěním v patře**
- **Změna výšky patra**
- **Odstranění patra**

Dále šlo v každém patře zobrazit všechna umístění, která se v něm nacházela, a upravovat jejich názvy a rozměry (obrázek 4.9).

Obrázek 4.8: Záložka *Úprava regálu*

Generace názvů Funkci automatického přiřazování názvů k umístěním jsem upravil tak, že se nyní spouštěla manuálně a provedla se jen nad buňkami ve výběru. Všechny ovládací prvky spojené s automatickým přiřazováním jsem přesunul do panelu *Generace názvů*. Daly se zde nastavovat počáteční symboly jmenného systému a směry číslování (obrázek 4.10).

4.7 Hodnocení druhé iterace

Jelikož druhá iterace obsahovala dostatek funkcionalit ke splnění většiny požadavků, bylo vhodné ji otestovat a zjistit, jestli byl použitý artefakt pochopitelný a jestli rozhraní bylo použitelné.

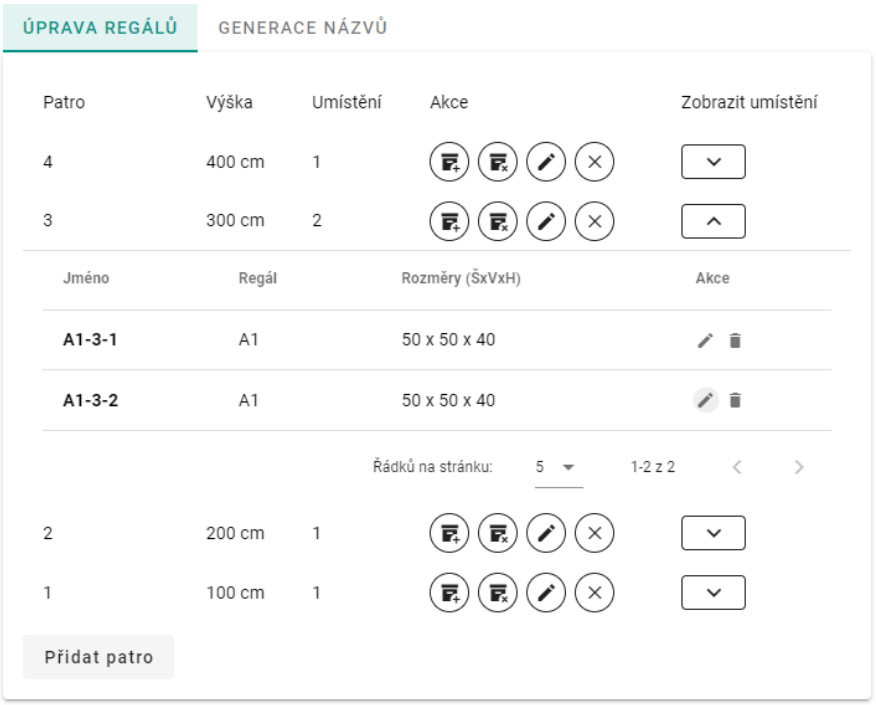
V této fázi jsem ještě nepodnikal důkladné testy použitelnosti, ale pouze menší testování, kterého se účastnili výhradně skladoví experti, jelikož se stále jednalo o ranou fázi projektu a nevěděl jsem, jestli byl tento styl rozhraní správnou volbou. Hlavním cílem bylo tedy potvrdit, jestli byla hlavní myšlenka artefaktu správná.

Testování proběhlo bez scénáře – testeři si pouze otevřeli dané rozhraní a zkusili v něm z hlavy namodelovat mapu buď fiktivního, nebo reálného skladu. Také jsem si nepořizoval žádný záznam.

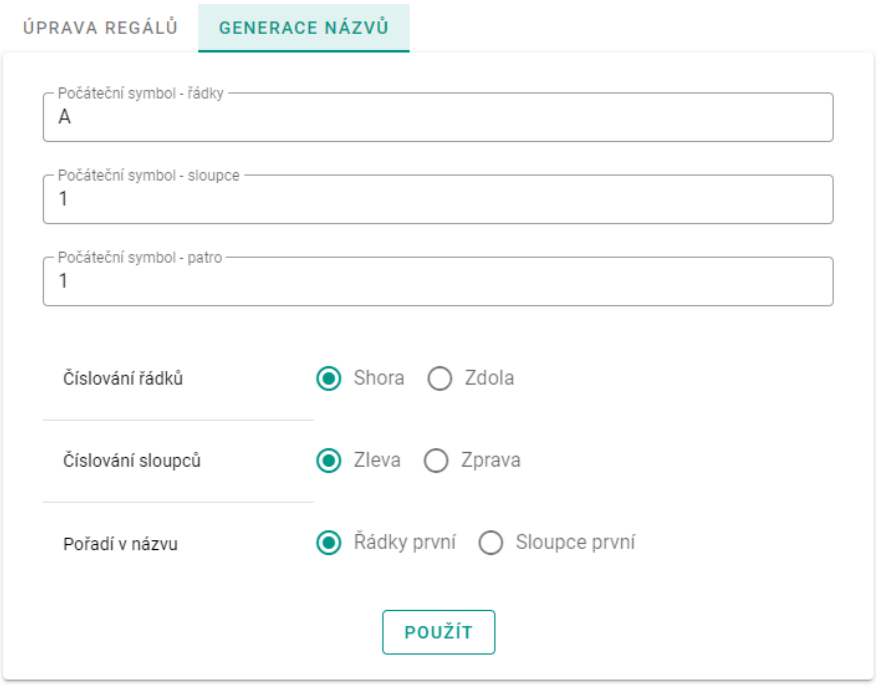
4.7.1 Výsledky testování

Ihned na začátku byl jeden účastník zaskočen tím, že typ buňky, která vyjadřovala jeden sloupec regálu, se jmenoval také *regál*. Účastníkovi se také nelíbilo, že jeden regál v mapě musel „rozkouskovat“ na jednotlivé sloupce

Obrázek 4.9: Záložka *Úprava regálu* – zobrazení všech umístění v daném patře



Obrázek 4.10: Záložka *Generace názvů*



regálu – připadalo mu přirozenější, kdyby se všechny tyto buňky spojily do jednoho obdélníku, který by jasně značil jeden regál.

Když však přišlo na řadu buňky s umístěními upravit pomocí panelu *Úprava regálů*, většina účastníků vůbec nevěděla, jak se v tomto panelu zorientovat. Funkce jednotlivých tlačítek jsem jim musel vysvětlovat, i to však nestačilo na to, aby si účastník dokázal představit regál ze strany a aby provedl správné úpravy.

Mnoha účastníkům také vadilo, že nešlo používat klávesové zkratky pro kopírování či mazání a s jednotlivými buňkami nešlo jakýmkoliv způsobem hýbat – všechny tyto funkcionality účastníci znali z programu Excel, kde byly dostupné.

4.7.2 Závěr

Po další diskuzi se skladovými experty jsem došel k závěru, že použitý artefakt nebyl vhodně zvolen. Rozhraní sice splňovalo požadavky algoritmů a daly se v něm vyjádřit i konfigurované paletové regály, bylo však příliš složité na použití.
V artefaktu jsem identifikoval tyto zásadní problémy:

- Absence pohledu ze strany – i když jsem vybraná umístění roztřídil podle pater (obr 4.9), většině uživatelů to nestačilo na to, představit si regál ze strany a provést vhodné úpravy.
- Funkci Generace názvů bylo příliš obtížné používat, jelikož názvy umístěním nepřiřazovala automaticky, ale musela se spouštět manuálně. Mnoho účastníků tuto funkci dokonce ani nenašlo. Zároveň bylo velmi složité po spuštění funkce ověřit, jestli byly názvy přiřazeny správně, jelikož uživatel na první pohled neviděl názvy všech nově pojmenovaných umístění a musel se k nim postupně dostávat rozkliknutím patra (obr 4.9).

Rozhraní tedy sice mělo dobrou utilitu – do mapy se dalo zaznamenat všechny informace, které požadovaly optimalizační algoritmy (požadavek F1), ale nebylo příliš použitelné. V této fázi projektu jsem měl dvě možnosti, jak pokračovat:

a) Spokojit se s dosavadním artefaktem (použít stávající znázornění mapy) a pouze doladit různé prvky rozhraní tak, aby rozhraní bylo použitelnější.

a) Zkusit upravit artefakt tak, aby jsem intuitivně vyřešil stávající problémy novým způsobem znázornění mapy.

Druhá možnost samozřejmě byla časově náročnější a vyžadovala kompletní přepracování stávajícího rozhraní. Nevýhodou bylo, že jsem rozhraní implementoval na plné úrovni funkčnosti kvůli tomu, že jsem nemohl použít prototypovací software. Došel jsem však k závěru, že úprava artefaktu je tou lepší,

ač náročnější volbou k tomu, abych vytvořil použitelné rozhraní, což bylo hlavním cílem této práce. Zároveň mi přišlo zajímavější prozkoumat nové možnosti artefaktu a tím získat více informací o problematice mapy skladu.

Kapitola 5

Druhý prototyp

Dle výsledků z testování prvního prototypu jsem určil, že bude potřeba provést změny v artefaktu a s tím i předělat celé uživatelské rozhraní. V této kapitole popíši provedené změny a funkcionality nového prototypu.

5.1 Návrh změn v artefaktu

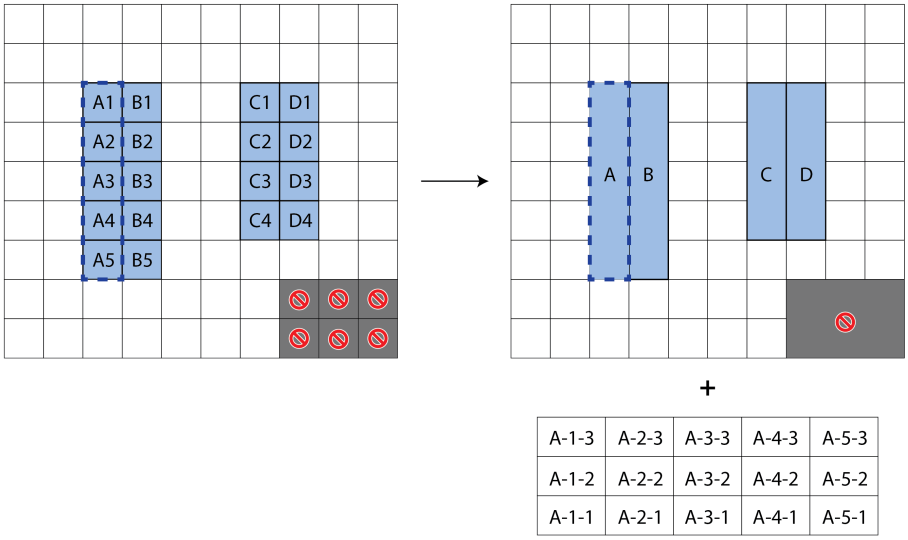
Použití 2D tabulky z předchozího artefaktu zůstalo, zásadní změnou však bylo vyjádření regálu – všechny buňky, které znázorňovaly sloupec nějakého regálu, se v novém artefaktu sloučily do jediného obdélníku (několik sloučených buněk dohromady). Tato změna přinesla mnoho výhod:

- Na první pohled bylo zřejmé, že obdélník **znázorňuje celý regál**.
- Regál lze jednoduše v mapě **přemístit** a **otáčet**.
- Regál by dále šlo snadno **označit**, **kopírovat** a **vložit** na ostatní místa ve skladu. Pro velmi symetrický sklad by tak návrh celé mapy zabral velmi krátkou dobu, jelikož by uživatel mohl navrhnout regál, který se ve skladu opakuje, a poté z něj celou mapu „poskládat“.
- V obdélníku znázorňující regál jsem nyní mohl zobrazit pouze jméno regálu, a nikoliv uvádět název pro každý sloupec regálu. Tím se mapa více zpřehlednila (obrázek 5.1)

Velmi zjednodušeně bych tedy změny v novém artefaktu popsal tak, že v předchozím artefaktu se „vybarvovaly kostičky“, kdežto v novém artefaktu tabulka představuje síť, na kterou pokládáme různé objekty, jako např. regál, či nějakou překážku, a pohybujeme s nimi.

Sloučením regálových kostiček do jediného obdélníku jsem však ztratil veškeré informace o umístěních. Přišel jsem tedy s novým nápadem, že by se samotný regál upravoval ze strany – uživateli by se při náhledu na regál otevřelo

Obrázek 5.1: Změna znázornění regálu mezi prvním a druhým artefaktem



nové okno s profilem regálu, kde by mohl jednodušeji upravovat jednotlivá umístění.

5.1.1 Problém orientace regálu

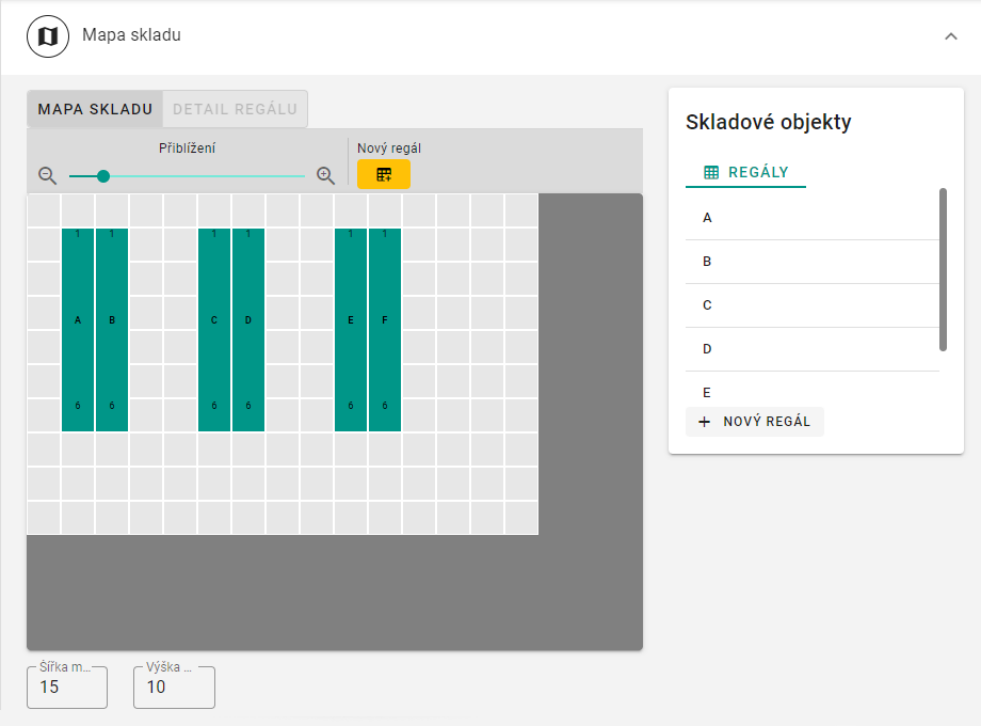
I přes řadu výhod s sebou nový artefakt přinesl také jeden zásadní problém, který bylo potřeba vyřešit. Tím, že se připojil pohled na regál ze strany, muselo se určit, z kterého směru se uživatel na regál dívá, zda zepředu, nebo zezadu. Dalším problémem bylo, že orientace regálu nebyla v mapě vůbec zaznamenána – regál byl pouze vyjádřen jako obdélník a jakékoliv informace o jeho orientaci chyběly. Měl jsem tedy před sebou dvě možnosti:

a) Přidat možnost zaznamenání přední strany regálu do mapy; uživatel by se poté vždy díval na předek regálu.

b) Nezaznamenávat přední stranu regálu; uživatel by se vždy díval na regál z jedné světové strany. To znamená, že by se mohl dívat i na zadní stranu regálu.

Důvod, proč orientaci regálů řeším, je číslování sloupců regálu. Pokud uživatel zadá špatné očíslování sloupců regálu (např. sloupce očísluje z opačné strany, než jak jsou očíslované ve skutečnosti), tak vzniknou v mapě velké nepřesnosti, které mohou dramaticky ovlivnit optimalizační algoritmy. Bylo třeba tedy uživateli jasně sdělit, jakým směrem se na regál dívá, aby zadal očíslování sloupců správně.

Obrázek 5.2: Celkový vzhled rozhraní druhého prototypu



Oba dva zmíněné způsoby, jak se s tímto problémem vypořádat, měly své výhody a nevýhody.

- U způsobu **a)** by uživatel musel zadávat do mapy více informací. Díval by se však na regál vždy zepředu, což by více korespondovalo se skutečností.

- U způsobu **b)** by uživatel nemusel zadávat žádné další informace. Při pohledu ze strany by však musel dbát na to, že se dívá na regál z jedné světové strany, což by mohlo ztížit proces zadávání dat – občas by si uživatel v hlavě musel nějaký regál převrátit, tzn. zadávat data tak, jako kdyby se na regál díval zezadu.

Zde hlavně záleželo na cílových uživatelích, jak si pohled ze strany představovali. Jelikož jsem v této fázi projektu neměl dostatek času ověřovat si, který z těchto dvou způsobů by pro uživatele byl pochopitelnější, rozhodl jsem se pro jednodušší metodu **b)**, přičemž přechod na metodu **a)** by implementačně nebyl složitý.

5.2 Vzhled a funkce rozhraní

Rozhraní jsem musel vytvořit z velké části od začátku, aby bylo přizpůsobené novému artefaktu. Místo vybarvování jednotlivých políček uživatel nyní tvořil objekty (např. regál), které se poté umisťovaly do mapy. S objekty se po jejich označení levým tlačítkem myši dalo různě manipulovat, jako např. hýbat nebo rotovat. Celkový vzhled rozhraní je vidět na obrázku 5.2.

Cílem bylo také rozhraní zjednodušit a odstranit přebytečné prvky z předchozího artefaktu. Už nebyla potřeba masivní sada nástrojů na úpravu tabulky – proto jsem v horním panelu nástrojů nechal pouze lupu a tlačítko na tvorbu nového regálu. Nástroje na manipulaci s objekty se v panelu ukázaly pouze tehdy, pokud byl nějaký objekt zvolen (obr 5.3).

Do rozhraní jsem nově přidal postranní panel, který registroval všechny objekty umístěné v mapě – při velkých mapách by si tak uživatel mohl jednoduše najít objekt v seznamu, místo pátrání v mapě (obr 5.2, vpravo).

V mapě jsem rozlišil dva druhy objektů:

- **Regál:** zabíral $1 \times n$ nebo $n \times 1$ buněk v tabulce
- **Překážka**: zabírala $m \times n$ buněk v tabulce

Z časových důvodů jsem však stihl v prototypu implementovat pouze regály. Regály byly tím hlavním objektem, který měl být pro uživatele pochopitelný a zároveň byly tím nejdůležitějším prvkem pro optimalizační algoritmy, rozhraní proto stále dosahovalo vysoké úrovně funkčnosti.

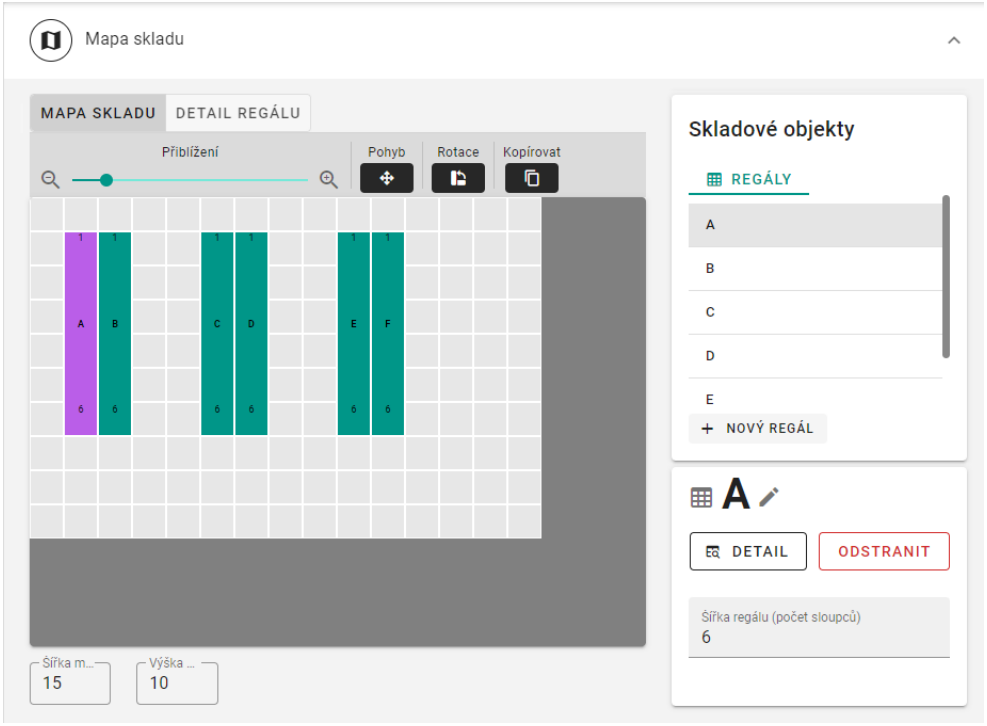
5.2.1 Regál

Regál byl na mapě zaznamenán jako obdélník, tzn. několik sloučených buněk tabulky dohromady. V obdélníku byly dále zobrazeny různé informační prvky (obrázek 5.4):

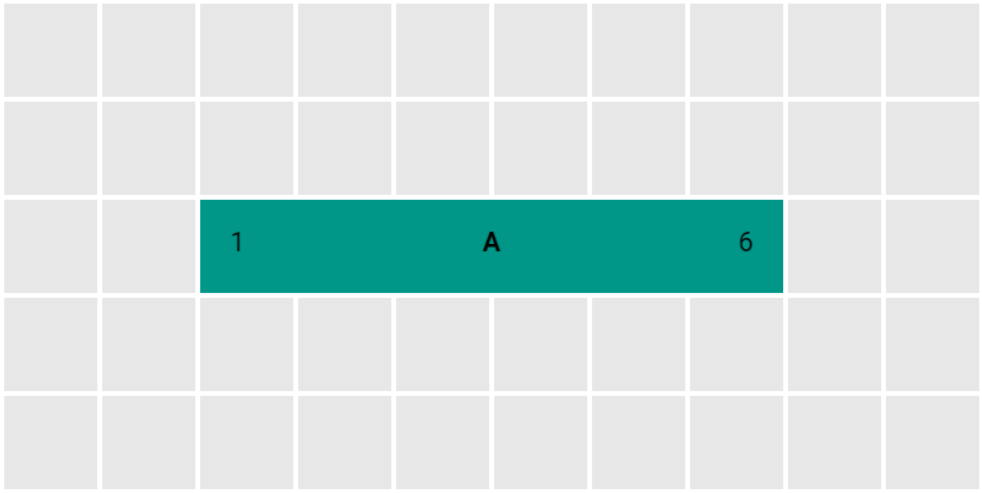
- **Jméno regálu:** uprostřed obdélníku.
- **Symboly prvního a posledního sloupce regálu:** na každém konci regálu. Tato čísla sloužila k tomu, aby uživatel věděl, jestli číslování sloupců zadal správně a odpovídají skutečnosti.

Levým kliknutím se dal regál, stejně jako každý jiný objekt, označit. Po označení regálu se rozhraní přizpůsobilo – daný regál se vybarvil jinou, výraznou barvou (v tomto případě světle fialovou), a uživatel měl nyní k dispozici různé funkce – regál přesunout na jiné místo, otočit jej o 90 stupňů, zduplikovat jej či změnit počet jeho sloupců. Dále bylo možné regál smazat či přejmenovat a také nahlédnout na něj ze strany, což byla nejdůležitější změna v novém prototypu.

Obrázek 5.3: Vzhled rozhraní po výběru regálu



Obrázek 5.4: Vzhled regálu



5.2.2 Detail regálu

Samotný pohled na regál ze strany jsem se rozhodl schovat pod záložku *Detail regálu*. Uživatel tedy měl možnost přepínat mezi pohledem na celou mapu seshora (záložka *Mapa skladu*) a pohledem ze strany na právě vybraný regál. Důvod, proč jsem tyto dva pohledy v rozhraní oddělil do odlišných záložek a nezobrazoval jsem je najednou, byl dán tím, že jsem nechtěl rozhraní příliš znepřehlednit – již jsem pracoval s faktem, že toto rozhraní je vložené do již stávajícího rozhraní aplikace Swordfish, a proto jsem se jej snažil dělat co nejjednodušší.

Regál jsem se rozhodl pojmout opět jako 2D tabulku, ve které každá buňka symbolizovala jedno umístění. Názvy se umístěním přiřazovaly automaticky dle jejich pozice v regálu – zatím jsem nepřidal možnost umístění přejmenovat na nějakou vlastní hodnotu[9]. Co se týkalo ovládání rozměrů celého regálu, mohl zde uživatel měnit pouze počet pater regálu, ne však počet sloupců, jelikož zatímco počet pater neměl na rozložení celkové mapy žádný vliv, počet sloupců měl – mohlo se např. stát, že uživatel regál rozšíří o nějaký počet sloupců, avšak v mapě by poté regál mohl kolidovat s jiným regálem. Tuto situaci bych uživateli nemohl lehce předat, pokud by mapu skladu neviděl. Proto se počet sloupců ovládal jen v záložce Mapa skladu.

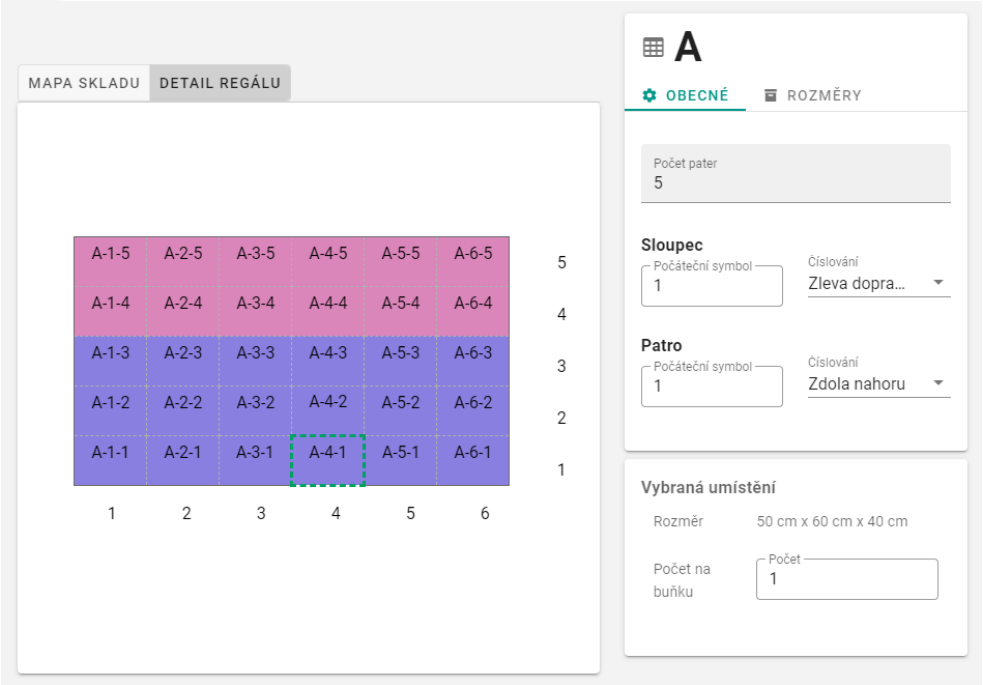
Uživatel dále mohl změnit počáteční symboly číslování sloupců a pater a směr těchto číslování. Jakákoliv změna číslování sloupců se projevila nejen v pohledu ze strany změnou názvů umístění, ale i v mapě – a to změnou symbolů na začátku a konci regálu (obr 5.4).

Přiřazování rozměrů umístěním jsem se rozhodl pojmout zajímavým konceptem. Všiml jsem si totiž, že rozměry umístění se ve skladu často opakují. Proto jsem v postranním panelu přidal záložku *Rozměry*, ve které uživatel mohl definovat jednotlivé rozměry, které se potom daly přiřazovat umístěním tím, že je uživatel vybral a klikl na tlačítko *Přiřadit*. Umístění poté změnila barvu na tu, kterou si uživatel vybral při tvorbě rozměru. Rozměry jsem pojal globálně – rozměry, které si uživatel nadefinoval v jednom regálu, byly dostupné i při definici dalších regálů, aby se uživatel nemusel opakovat.

Zajímavá otázka byla, jak vyjádřit barvu rozměru – zatím jsem toto rozhodnutí nechal na uživateli, který měl nad výběrem barvy úplnou kontrolu a mohl si regál „vybarvit“, jak chtěl. Napadlo mě však, že by se barva mohla odvozovat např. z celkového objemu umístění – tím by uživatel získal lepší představu o tom, která umístění mají menší, či větší rozměry. I když to byl zajímavý koncept, opět jsem se při implementaci tohoto rozhraní musel zabývat důležitějšími funkcemi, a proto jsem zatím barvu rozměrů nechal plně v rukou uživatelů.

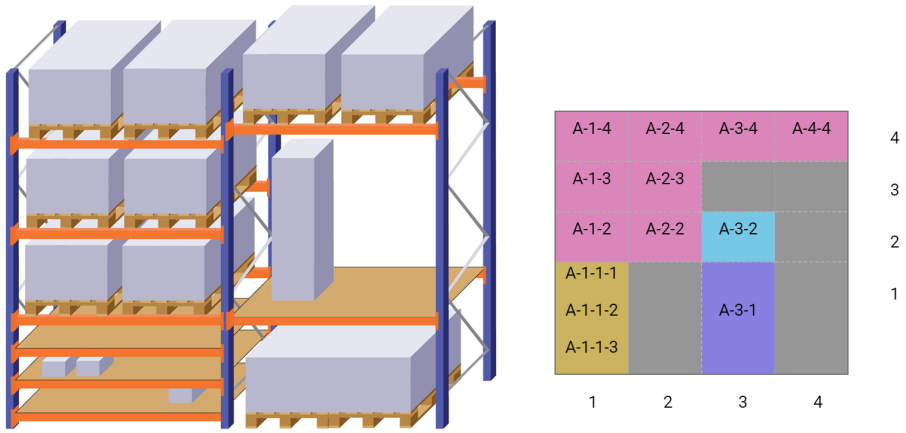
[9] Zaměřoval jsem se na hlavní funkcionalitu a v rozhraní se přirozeně nabízela spousta „nice-to-have“ funkcí, které jsem musel kvůli jejich četnosti a implementační náročnosti opomenout.

Obrázek 5.5: Záložka *Detail regálu*, představující pohled na regál ze strany



Obrázek 5.6: Záložka *Detail regálu* – definice rozměrů

Obrázek 5.7: Příklad, jak vyjádřit různé skladové konfigurace v novém artefaktu



Také jsem musel vyřešit problém konfigurací paletových regálů – do jedné sekce regálu by mělo jít vložit o patro navíc, nebo nějaké patro odebrat – sloučit dvě patra do jednoho. Nejradši bych tento problém vyřešil jednoduchým konceptem slučování a rozdělování buněk tabulky, avšak kvůli náročnosti implementace jsem přistoupil k alternativnímu řešení.

Místo slučování a rozdělování buněk tabulky jsem zavedl **počet umístění na buňku**. Buňka tedy standardně měla jedno umístění, nemusela však také mít žádné (počet umístění se rovnal 0), anebo mohla mít umístění několik. Tímto se dalo vyjádřit např. sloučení několika buněk takovým způsobem, že všechny sloučené buňky, kromě jedné, budou mít nulový počet umístění, a pouze jedna buňka ze všech sloučených by měla jedno umístění. Toto řešení bylo samozřejmě těžší na pochopení, dala se jím však vyjádřit většina konfigurací paletových regálů (příklad na obrázku 5.7).

5.3 Implementace

Základní technologický stack se od prvního prototypu nezměnil a stále jsem používal framework Vue.js s grafickou knihovnou Vuetify. Kód druhého prototypu byl však oproti prvnímu komplexnější a při implementaci jsem se potýkal s několika problémy. Také jsem musel zajistit komunikaci s backendovou částí aplikace. V této sekci detailněji popíšu implementační záležitosti.

5.3.1 Vuex

V sekci 4.3.1 jsem popsal, jak jsem využil systému komponent frameworku Vue k rozčlenění nového rozhraní na menší celky. V prvním prototypu nebyla členitost příliš velká a komponent jsem měl málo. Ve druhém prototypu však přibylo mnohem více funkcionality, a s tím počet komponent vzrostl (diagram 5.8).

Diagram 5.8: Strom komponent druhého prototypu

- `StockMap` ..... kořenová komponenta
  - `StockMapEdit` ..... rozhraní pro tvorbu mapy
    - `StockMapView` ..... okno s mapou v záložce *Mapa skladu*
    - `StockMapRackView` ..... okno s regálem v záložce *Detail regálu*
    - `StockMapSidePanel` ..... postranní panel
      - `StockMapExtendedPanel` ..... rozšíření panelu v záložce *Detail regálu*

Větší členitost s sebou přinesla problém. Každá Vue komponenta spravuje svůj vnitřní reaktivní stav, avšak každá komponenta v mém rozhraní zároveň pracovala s mapovými daty, které měla být konzistentní napříč všemi komponentami. Mapová data tedy bylo potřeba implementovat tak, aby existovala pouze jedna jejich instance, která byla mezi komponentami sdílená. Tato instance by také měla být reaktivní – jednotlivé komponenty by měly automaticky reagovat na změny v datech. Funkcionalitu sdíleného a zároveň reaktivního stavu poskytuje knihovna Vuex [26], která již byla v aplikaci Swordfish použita.

V aplikaci používající Vuex je sdílený stav spravován objektem s názvem *store*. Tento objekt je reaktivní, globálně dostupný a zároveň nelze jeho stav měnit přímo, ale pomocí *mutací*, které jednotlivé komponenty mohou volat. Ve storu jsou dále dostupné *gettery*, které vrací nějakou hodnotu vypočtenou ze stavu. Gettery jsou také reaktivní – pokud dojde ke změně stavu, getter se automaticky přepočítá, pokud byla ve stavu změněna nějaká hodnota, se kterou se v getteru pracovalo. Posledním důležitým prvkem Vuexu jsou *akce*, ve kterých lze zavolat více getterů a mutací – akce tedy představují komplexnější změny stavu. Akce lze také volat asynchronně, této možnosti jsem však nevyužil.

Všechny prvky storu – tedy stav, mutace, gettery a akce – lze rozdělit do jednotlivých modulů. V aplikaci Swordfish jsem tedy vytvořil vlastní Vuex modul, který spravoval sdílená mapová data a intuitivně jsem využil jednotlivých funkcí Vuexu pro usnadnění čtení a změny dat (ukázka kódu 5.1)

Ukázka kódu 5.1: Případy užití Vuex pro správu mapových dat

```
const storageMapModule = {
    state: {
        // Internal map structure
        map: {
            // ...
        },
        // ...
    },
    mutations: {
        // Create a new rack at coordinates 'topLeft' with name 'name'
        createNewRack(state, {topLeft, name}) {
            // ...
        },
        // Delete currently selected object
        deleteSelectedObject(state) {
            // ...
        },
        // ...
    },
    getters: {
        // Transform data from 'map' to easily display it in the map window
        render: (state) => {
            // ...
        },
        renderRack: (state) => {
            // ...
        }
        // ...
    },
    actions: {
        // Copy object to a new destination defined by 'newTopLeft'
        // Use getters to check for possible collisions
        // Commit createNewRack mutation if no collisions found
        copyObject({state, getters, commit}, {object, newTopLeft}) {
            // ...
        },
        // ...
    }
};
export default storageMapModule;
```

5.3.2 Komunikace s backendem

Při vývoji byl backend jen částečně implementován a v době psaní této práce je neaktuální. Architektura fungování celého rozhraní (tedy jak ukládání/načítání mapy, tak návrhu optimalizací) proto není přesně specifikována, ale je naplánována její základní struktura:

- Frontend pro tvorbu mapy bude posílat mapová data ve formátu JSON na endpointy REST API pro ukládání/načítání dat.
- API na ukládání/načítání mapy lze vytvořit novou, nebo lze přímo rozšířit stávající API skladového systému *Atlantis*. Tato API se jmenuje Octopus, je napsána v jazyce PHP a používá framework Symfony. I kdyby se však vytvořila nová API, musela by stále probíhat komunikace přes Octopus, např. pro ověření jednotlivých jmen umístění zaznamenaných v mapě.
- Poslední část backendu by navrhovala nové optimalizace. K tomu by potřebovala jak mapová data, tak data ze skladového systému. Musela by tedy komunikovat jak s novým API na ukládání/načítání mapy, tak s Octopus API.

Backend lze tedy implementovat mnoha způsoby – lze jej rozdělit na několik částí či vše implementovat jako rozšíření Octopus API. Způsob komunikace mezi frontendem a nějakými API endpointy je však přímočarý a pravděpodobně se nebude provádět jiným způsobem. Proto jsem ve frontendu zajistil generaci JSONu, který poté může být poslán na endpointy API pro další zpracování.

Data mapy jsem ve frontendu ukládal do jediného Javascriptového objektu (v ukázce kódu 5.1, řádek 4). Pro vygenerování JSONu a komunikaci s API jsem použil knihovnu Axios, která již byla ve frontendu skladového systému použita.

5.3.3 Zdrojové kódy

Zdrojové kódy jak prvního, tak druhého prototypu jsou dostupné na GitLabu firmy Jagu v projektu atlantis/Swordfish (merge request `WIP: Storage Map`, branch `storage_map`). Oba prototypy jsem již integroval do rozhraní skladového systému, a proto nemohou být dodány a spuštěny samostatně. Poslední verze prvního prototypu je dostupná pod commitem `8db711b5` a poslední verze druhého prototypu je dostupná pod nejnovějším commitem.

5.4 Shrnutí

Na základě chyb z prvního prototypu jsem navrhl a implementoval druhý prototyp na tak vysoké úrovni funkčnosti, že se už jednalo o samotnou imple-

mentaci. V novém prototypu jsem musel vyřešit řadu problémů, které vznikly změnou artefaktu. Při vývoji nového prototypu jsem více průběžně testoval – a to pomocí akceptačních testů, které měly za cíl dovést prototyp do stavu, kdy byl dostatečně stabilní a dávalo smysl jej podrobit důkladným testům použitelnosti, ve kterých jsem chtěl plně ověřit vhodnost a použitelnost nového artefaktu. Celý proces jak akceptačního testování, tak testování použitelnosti je popsán v další kapitole.

Kapitola 6

Testování

V této kapitole popíši celý proces testování druhého prototypu, od počátečních akceptačních testů až po finální testování použitelnosti, které mělo ověřit, zda byl artefakt dobře zvolen a jestli byl dostatečně pochopitelný pro reálné uživatele.

6.1 Akceptační testy

Akceptační testy jsou z hlediska vývoje softwaru testy, které přicházejí po vývoji a systémovém testování. [27] Testuje se tedy software, který je funkční a neobsahuje velké množství bugů, a ověřuje se, jestli je software připraven na produkci. Z hlediska standardního vývoje softwaru bych tedy akceptační testy konal až ve finálních fázích projektu, kdy už bych měl dávno provedené testování použitelnosti. Jelikož byl můj projekt nestandardní a současně s návrhem jsem již prováděl i implementaci, předcházely akceptační testy testům použitelnosti.

Druhým důvodem, proč jsem akceptační testy prováděl předem, bylo, že rozhraní mého prototypu bylo velmi komplexní – uživatel mohl přepínat mezi různými záložkami a manipulovat s mnoha prvky a musel jsem zajišťovat koordinaci těchto prvků tak, aby se rozhraní nijak nerozbilo a zároveň zůstalo dostatečně pochopitelné. Proto jsem před řádným testováním použitelnosti musel dostat prototyp do stavu, kdy jej uživatel nemohl snadno rozbít a kdy byl dostatečně stabilní na to, aby testování použitelnosti mělo vůbec smysl.

Akceptační testy jsem prováděl s vedoucím práce Ing. Jiřím Hunkou. Celkem jsem provedl dva větší akceptační testy, při kterých jsem přišel na opravdu hodně nedostatků v rozhraní, které jsem potom musel upravit. Velká část připomínek byla opravdu očividná a po jejich odhalení dávalo smysl je ihned opravit – jednalo se např. o špatně umístěné tlačítko, špatně zvolenou ikonku na tlačítku, nevhodné umístění tlačítek (např. tlačítko *Smazat* hned vedle tlačítka *Upravit*) nebo také fakt, že mnohé prvky šlo smazat bez jakéhokoliv

upozornění. Další připomínky však byly implementačně náročnější a týkaly se většinou komfortu uživatele – velmi intuitivní funkcí, která v rozhraní chyběla, bylo např. jednoduché zvolení objektu a následně jeho táhnutí po mapě za účelem jeho přesunutí. Jedná se o funkci, kterou každý uživatel v rozhraní očekává – z mé strany byla však implementačně náročná a bohužel jsem ji do rozhraní nezahrnul.

Díky akceptačním testům jsem prototyp „vyhladil“ a opravil zřejmé chyby, které by při testování použitelnosti vadily. Zároveň jsem s vedoucím práce vedl diskuzi o samotné použitelnosti nového rozhraní. Došli jsme k závěru, že nové rozhraní mělo šanci obstát a nějaké úrovně použitelnosti by rozhodně mohlo dosáhnout. Přišlo na řadu tyto testy provést a zjistit, jestli je rozhraní použitelné.

6.2 Příprava na testy použitelnosti

Před testováním bylo potřeba jej řádně naplánovat a zajistit, abych použitelnost správně ověřil a vyhodnotil.

6.2.1 Výběr účastníků

Nejdříve jsem musel stanovit, kolik účastníků najmu a jaké charakteristiky by měli mít. Hlavním faktorem při rozhodování byly persony, které jsem si předem určil v kapitole 3.3. Rozhraní bylo zamýšleno jak pro skladové zaměstnance (primární persona), tak i pro technicky zdatnější experty (sekundární persona). Bylo vhodné tedy přihlédnout k doporučenému počtu testerů podle Nielsena, který tvrdí, že 5 účastníků je optimálním počtem, ale na dostatečné otestování stačí 3–5 účastníků [28]. Plánoval jsem tedy testovat celkem s 4 účastníky, kde 2 byli zástupci primární persony a zbývající 2 představovali sekundární personu.

6.2.2 Scénáře

Při vymýšlení scénářů jsem musel brát v potaz odlišné charakteristiky person. Celkem jsem sestavil dva scénáře, každý zamýšlený pro jinou personu:

A) Tvorba mapy reálného skladu

Nejpřínosnějším způsobem, jak rozhraní otestovat, samozřejmě bylo nechat nějakého skladového zaměstnance vytvořit mapu reálného skladu. Tento scénář jsem pojal jako volný a silně improvizovaný – nedaly se totiž vytyčit pevné úlohy, jelikož každý reálný sklad je jinak strukturovaný a také jsem očekával, že úroveň porozumění rozhraní ze strany účastníků bude velmi proměnlivá. Rozhodl jsem se proto tento scénář silně moderovat a postupně vybudovat scénář za běhu podle toho, jak si daný účastník v rozhraní povede. Očekával

jsem také, že se účastníci budou mnohdy zasekávat, a proto jsem se pro snadnější průchod testem rozhodl, že účastníkům budu napovídat, pokud budou tápat příliš dlouho.

I přesto, že jsem se rozhodl improvizovat, jsem si vytyčil pár základních funkcionalit, které jsem chtěl otestovat. V průběhu testu jsem účastníka tedy postupně směroval tak, aby prošel všemi základními body:

- Tvorba nového regálu a základní manipulace s regálem (pohyb, rotace)
- Duplikace regálu: schopnost účastníka najít a využít tuto funkci pro zrychlení celého procesu tvorby mapy
- Schopnost účastníka upravovat šířku (počet sloupců) a výšku (počet pater) regálu – najít ovládací prvky pro provedení těchto kroků
- Schopnost účastníka najít v rozhraní záložku Detail regálu a pochopit, že se jedná o pohled ze strany na daný regál
- Přiřazování rozměrů a vyjadřování regálových konfigurací v záložce Detail regálu
- Dostatek funkcí v rozhraní na přesné popsání reality:
  - Dá se přesně vyjádřit daný jmenovací systém reálného skladu?
  - Dají se všechny regály a umístění do mapy zaznamenat a pojmenovat přesně podle skutečnosti?
  - Lze vyjádřit strukturu regálu v rozhraní tak, aby odpovídala skutečnosti?

B) Tvorba mapy fiktivního skladu

Jelikož jsem se chystal testovat také zástupce sekundárních person, kteří v žádném reálném skladu nepracovali, musel jsem vymyslet scénář, ve kterém se účastník pokusí vytvořit mapu nějakého fiktivního skladu. Bylo třeba tedy fiktivní sklad vymyslet. Při vymýšlení fiktivního skladu jsem se držel těchto bodů:

- Ve stávajícím prototypu ještě nebyly implementovány překážky – uvažoval jsem tedy o skladu bez překážek a s obdélníkovými hranicemi.
- Regály se v reálných skladech často opakují – měl bych tedy zajistit, aby se ve fiktivním skladu regály často opakovaly, a měl bych prověřit schopnost účastníka využít různých funkcionalit rozhraní k tomu, aby nemusel každý regál tvořit od začátku.

- Regály mohou být konfigurované – strukturu regálů bych měl tedy vymyslet tak, aby se v nich vyskytovala většina konfigurací popsaných v analýze. Prověřím schopnost uživatele nalézt a správně použít funkce při pohledu na regál ze strany tak, aby každou konfiguraci dokázal správně znázornit.

Dle předchozích bodů jsem vytvořil dokument, který nejdříve obsahoval plánek vyobrazující pohled seshora na obdélníkový sklad s několika regály (obr 6.1). Dále jsem do dokumentu vložil fotografie a ilustrace různých regálů ze strany, kde jsem vyznačil rozměry a názvy jednotlivých umístění. Vytvořil jsem celkem tři typy regálů:

1. Symetrický regál bez žádných konfigurací, s dvěma odlišnými rozměry umístění (v plánku regály **E**–**H**).

2. Méně symetrický regál, s třemi odlišnými rozměry umístění. V regálu na jednom místě umístění chybělo a v jedné sekci bylo jedno patro rozdělené na dva. (v plánku regály **A**–**D**).

3. Regál byl symetrický až na to, že v jedné sekci zabíralo jedno umístění 3 sloupce a 2 patra (obrázek 6.2, v plánku regál **I**).

Tento scénář byl méně improvizovaný než scénář A), opět jsem se jej však nesnažil členit na velmi specifické úlohy, abych účastníkovi ponechal jistou volnost a přiměl ho hledat správný pracovní postup samostatně. Úlohy jsem tedy zadal velmi vágně:

1. Vytvořte nějaký regál prvního typu a umístěte jej do mapy tak, aby poloha regálu odpovídala plánku.

2. Přidejte do mapy zbývající regály tohoto typu.

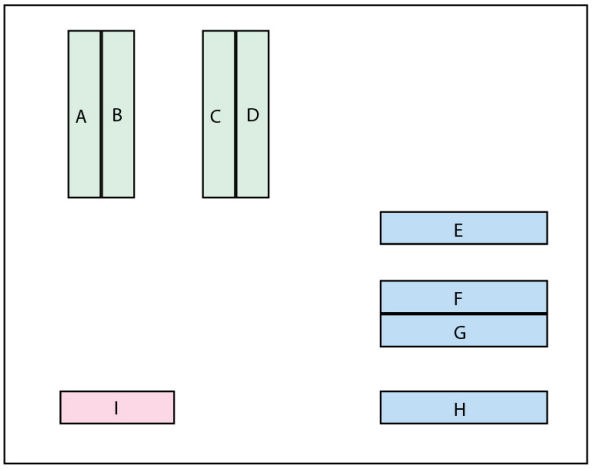
3. Vytvořte regály všech ostatních typů a umístěte je do mapy.

I v tomto scénáři jsem chtěl otestovat všechny hlavní funkce rozhraní zmíněné ve scénáři A). Zadání úloh v tomto scénáři však bylo konkrétnější a to, že jsem v dokumentu fiktivního skladu ukázal nějaké regály ze strany, už účastníky pobízelo k tomu, takový pohled v rozhraní vyhledat a použít. Proto jsem očekával, že v tomto scénáři nebudu příliš zasahovat a účastníkům pomáhat.

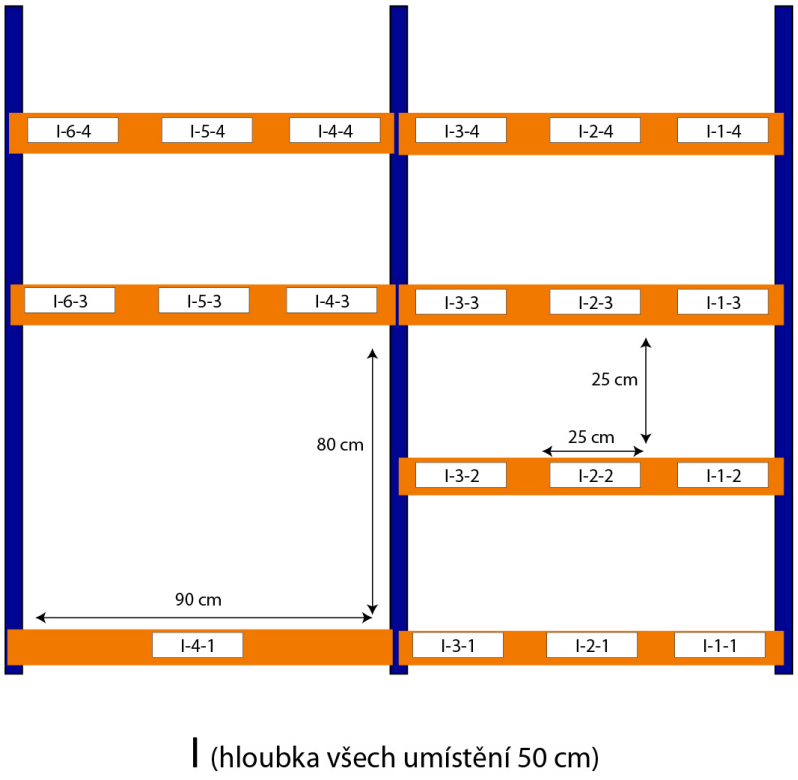
6.2.3 Prostředí

Testování jsem se rozhodl provést distančně, vzhledem k tomu, že bylo lehčí pořizovat audiovizuální záznam a také jsem mohl lépe pozorovat interakci

Obrázek 6.1: Plánek fiktivního skladu pro testování, s různými typy regálů



Obrázek 6.2: Fiktivní regál typu 3 z testovacího dokumentu



uživatele s rozhraním[10]. S účastníky jsem komunikoval pomocí služby Google Meet, která fungovala na principu online schůzky, kde jsem s účastníkem komunikoval mikrofonem a webkamerou. Interakci s prototypem jsem účastníkům umožnil pomocí programu Anydesk, který umožňuje vzdálené ovládání celé plochy. Pracoval jsem s dvěma počítači – na jednom jsem měl puštěnou schůzku, na druhém jsem měl nasdílený prototyp. Takto jsem mohl sledovat jak účastníka, tak testované rozhraní.

6.2.4 Sběr dat

Po získání informovaného souhlasu jsem po celou dobu testování pořizoval audiovizuální záznam ze schůzky ve službě Google Meet pomocí programu OBS.

Před testováním jsem s účastníky vedl kratší rozhovor. Nejdříve jsem je seznámil s testem a poté jsem účastníkům pokládal několik otázek, abych se dozvěděl více o jejich práci, skladu, ve kterém pracovali (pokud byli skladovými zaměstnanci), a jejich předchozích zkušenostech s mapami skladu a nějakým softwarem na tvorbu/úpravu mapy skladu.

Během testu jsem si zapisoval nejdůležitější události na papír. Tyto poznámky jsem poté využil ve finální diskuzi po konci testování, ve které jsem se snažil porozumět krokům, které účastníci provedli, a zároveň jsem účastníkům dal prostor pro jakékoliv připomínky, názory a možná vylepšení rozhraní.

6.2.5 Způsob vyhodnocení

Test jsem hodlal vyhodnotit především kvalitativně – hlavním účelem testu bylo zjistit, jestli účastníci pochopili daný artefakt a jestli ho byli schopni pohodlně a efektivně použít pro tvorbu mapy. Vzhledem k tomu, že jeden ze dvou scénářů byl improvizovaný a vyvíjel se za běhu, bylo těžké použitelnost hodnotit kvantitativně – zaměřil jsem se spíše na pocity a připomínky účastníků v rozhovorech a postupně si z těchto dat vybudoval výslednou představu o použitelnosti rozhraní.

Přesto jsem také použil kvantitativní metodiku pro vyhodnocení použitelnosti, a to System Usability Scale (SUS). Jedná se o soustavu deseti otázek, které účastníkovi položíme po konci testování:

1. Myslím si, že by se mi líbilo tento systém používat často.
2. Systém mi přišel zbytečně komplexní.
3. Myslím si, že systém byl jednoduchý na použití.
4. Myslím si, že bych potřeboval podporu technické osoby, abych systém mohl/a používat.

[10]Neviděl jsem sice, jak uživatel zachází s klávesnicí a myší, mohl jsem však podrobně sledovat obrazovku nasdíleného prototypu, což bylo pro mě daleko důležitější.

5. Myslím si, že různé funkce v tomto systému byly dobře integrovány.
6. Systém mi přišel příliš nekonzistentní.
7. Dokážu si představit, že většina lidí by se naučila tento systém používat velmi rychle.
8. Systém se mi používal těžkopádně.
9. Cítil/a jsem se sebejistě při používání tohoto systému.
10. Musel/a jsem se naučit spoustu věcí, než jsem systém mohl/a používat.

Každou otázku účastník ohodnotí 1–5 body, kde 1 bod znamená „silně nesouhlasím“ a 5 bodů znamená „silně souhlasím“. Celkový počet bodů se poté vypočítá tak, že u každé liché otázky 1 bod odečteme, u každé sudé otázky počet bodů odečteme od 5 a výsledné body všech otázek sečteme a vynásobíme 2,5. Výsledek bodů se pohybuje na škále od 0–100, kde se skóre větší než 68 považuje za nadprůměrné a skóre pod 68 za podprůměrné. [29]

I když byl SUS považován za validní způsob, jak zhodnotit použitelnost nějakého systému, tak se mi některé otázky nezdály příliš vhodné a samotný systém hodnocení mi připadal příliš komplikovaný. Rozhodl jsem se však, že tuto kvantitativní metodu použiji pouze pro zpřesnění celkového výsledku a pro zavedení alespoň nějakého ověřeného standardu pro vyhodnocení použitelnosti, aby byl výsledek testování více legitimní.

6.3 Průběh testování použitelnosti

Testování se nakonec zúčastnili celkem 3 účastníci – 2 zástupci primární persony a 1 zástupce sekundární persony, což dle Nielsena stačilo pro dostatečné otestování (3–5 účastníků [28]). Účastníci, kteří kreslili reálný sklad (scénář A)), se buď nacházeli fyzicky ve skladu, nebo měli k dispozici plánky a diagramy k tomu, aby si ověřili, že informace do mapy zadávali správně.

6.3.1 Olda

Olda reprezentoval sekundární personu, použil jsem tedy scénář B) Tvorba mapy fiktivního skladu. Funkci na vytvoření nového regálu účastník našel hned a regál správně umístil do mapy. Když chtěl regál přemístit, očekával, že regál přemístí pomocí táhnutí myši – taková funkce však chyběla. Další problém nastal při hledání funkce na prodloužení (přidání sloupců) regálu – účastník začal prozkoumávat celé rozhraní, a dokonce hledal i v záložce *Detail regálu*, stále se mu však nedařilo najít správný ovládací prvek. Po chvíli pátrání nakonec ovládací prvek našel a regál se mu podařilo prodloužit, takže jsem nemusel nijak zasahovat a radit.

Pro úpravu regálu ze strany účastník správně otevřel záložku *Detail regálu* a vyjádření regálu ze strany ihned pochopil. Při designu nejjednoduššího regálu bez žádných konfigurací účastník správně a rychle nastavil parametry – počet pater, počáteční symboly a směr číslování. Funkcionalita přiřazování rozměrů pomocí barev přišla účastníkovi intuitivní, ocenil by však více ovládacích prvků – při tvorbě nového rozměru chtěl účastník využít toho, že už si jeden velmi podobný rozměr předem vytvořil, takže jej chtěl zduplikovat. Zároveň účastníkovi přišlo matoucí, že nový rozměr neměl automaticky přiřazenou barvu a byl z počátku bílý, takže nešlo rozeznat umístění bez rozměru (které bylo také bílé) od umístění s již přiřazeným, avšak bílým rozměrem. Velmi nepříjemné bylo, když účastník použil tlačítko na smazání rozměru, neboť si myslel, že toto tlačítko jej pouze přesměruje do hlavního menu. Jinak však účastník úpravu ze strany zvládl velmi rychle a nijak zvlášť netápal. Regál prvního typu dokázal ze strany navrhnout přesně podle testovacího dokumentu. Účastník však provedl pár kroků navíc při přiřazování rozměrů, kdy pro každé patro vytvořil nový rozměr, i když ve většině pater měla umístění stejný rozměr. I když toto z informační stránky nebylo chybou a struktura regálu stále odpovídala plánku, tak si přebytečných, stejných rozměrů později všiml a duplikátní rozměry odstranil.

Po vytvoření prvního regálu jsem účastníkovi řekl, aby se pokusil do mapy přidat všechny ostatní regály tohoto typu. Během několika sekund účastník našel funkci *Kopírovat* a když zjistil, že se po první duplikaci regálu tento nástroj nevypnul a zůstal stále zapnutý, tak si s nástrojem chvíli pohrál a celou mapu zaplnil novými regály. I to byl dobrý příklad nouzové situace, jelikož nyní účastník musel všechny tyto přebytečné regály odstranit. Při odstraňování by účastník ocenil možnost zvolit více regálů najednou a všechny je naráz odstranit, což nešlo, a účastník proto musel každý regál zvolit a odstranit po jednom.

Při tvorbě druhého typu regálu, který už měl nějaké konfigurace, účastník nejdříve správně nastavil počet sloupců a pater. Poté pro vyjádření konfigurací hledal funkci na sloučení či rozdělení buněk. Po krátké době našel funkci *Počet umístění na buňku* a správně provedl potřebné úpravy – jedno umístění smazal nastavením hodnoty na 0 a jedné buňce správně nastavil počet umístění na 2. Jediný zádrhel nastal při přiřazování rozměru, kdy účastník nevěděl, jak se vypořádat s buňkou, ve které byly dvě umístění – rozměry se totiž přiřazovaly buňkám, a ne jednotlivým umístěním, a účastník nevěděl, jestli buňce přiřadit rozměr, který odpovídal jednomu menšímu umístění, nebo rozměr, který odpovídal součtu všech umístění v buňce. Účastník si správně vybral první možnost, tedy vytvořil nový rozměr, který odpovídal jednomu menšímu umístění, a buňce tento rozměr přiřadil.

Zbytek testování proběhl podobným způsobem a účastník se postupně v rozhraní zlepšoval a zrychloval. U posledního, třetího typu regálu účastník nejdříve místo 6 sloupců nastavil pouze 2, jelikož tím chtěl vyjádřit počet sekcí regálu. Později se však opravil a počet nastavil na 6. Účastník toto odůvodnil

tím, že jej spletly modré tyče na ilustraci regálu (obr 6.2), které naznačovaly, že regál rozdělí na 2 části. Účastník dále správně vyjádřil konfiguraci regálu, i když by opět ocenil možnost buňky sloučit, což by vizuálně bylo pochopitelnější.

Účastníkovi se nástroj líbil. Společně jsme došli k závěru, že rozhraní obsahovalo pár chyb, které by šlo lehce opravit – několik tlačítek mělo např. matoucí ikonku a některé ovládací prvky by mohly být výraznější a umístěné na lepších místech v rozhraní. Dále účastník chápal, že v rozhraní chyběly intuitivní funkce, jako např. tažení regálu pro jeho přemístění, jelikož sám pracoval s podobnými technologiemi a věděl, že tyto funkce byly implementačně náročné. Hlavní však bylo, že účastník porozuměl znázornění mapy a nakonec dokázal přesně vyjádřit všechny regály tak, jak byly zaznamenány v testovacím dokumentu. Účastník předčil má očekávání a jednotlivé funkce nacházel rychleji, než jsem předpokládal – celý proces tvorby zabral zhruba 45 minut; do času je započítána i průběžná diskuze nad různými funkcemi a vylepšeními. Během testu jsem nemusel nijak zásadně zasahovat a na všechno účastník přišel sám. Skóre SUS bylo 72,5, tedy nadprůměrné. Oblastí, kde došlo k největší ztrátě bodů, byla otázka 7 – *Dokážu si představit, že většina lidí by se naučila tento systém používat velmi rychle.* To bylo pochopitelné, jelikož rozhraní postrádalo pár intuitivních funkcí a samotné vyjádření mapy a regálů nemusel každý hned pochopit.

6.3.2 Libor

Libor reprezentoval primární personu. Pracoval nejen jako skladník, ale také jako vedoucí – v našem systému se tato role jmenovala „skladníkovedoucí“. Jeho sklad sestával ze 3 místností a celková plocha byla zhruba 150 m^2. Sklad měl v jedné místnosti 25 regálů po 5 policích (patrech), v druhé místnosti 10 regálů po 5 policích a ve třetí místnosti zhruba 30 umístění. V místnostech s regály byla za umístění považována jedna police v regálu.

S účastníkem jsem se domluvil, že zkusíme scénář A (tvorba mapy reálného skladu), a pokud testování nepovede dobrým směrem, zkusíme scénář fiktivní. Změna scénáře však nakonec nebyla nutná.

Účastník začal postupně tvořit jednu místnost skladu s regály. Účastníkovi jsem musel částečně pomáhat a vysvětlovat, kde najde jednotlivé funkce. Po vysvětlení a provedení účastníka rozhraním však účastník zvládal rozhraní ovládat samostatně a pomáhal jsem mu daleko méně.

Místnost reálného skladu, kterou si účastník vybral, byla speciální tím, že názvy všech regálů začínaly písmenem „V“, což označovalo místnost, ve které regál byl. V rozhraní tedy okno, které registrovalo všechny regály na mapě, bylo v podstatě zbytečné, jelikož všechny regály se jmenovaly stejně.

Při náhledu na regál ze strany účastník sice našel a dokázal použít funkci přiřazování rozměrů, avšak rozměr přiřadil takovým způsobem, že si vybral všechny buňky v regálu a přiřadil jim rozměr celého regálu – ve skutečnosti

však do mapy zaznamenal, že každé umístění bylo stejně velké jako celý regál, což byla samozřejmě velká nepřesnost. Způsob, jakým účastník uvažoval při přiřazování rozměrů, mi však přišel pochopitelný, avšak mě doposud vůbec nenapadlo, že by někdo uvažoval takovýmto směrem. Po vysvětlení účastníkovi, že rozměr se přiřazuje každému umístění, a ne nějaké ploše, kterou si uživatel vybere v regálu, se účastník opravil a rozměry nastavil správně.

Dále účastník chtěl do jednoho sloupce regálu přidat o jedno patro navíc. Došel k závěru, že by musel regál v mapě rozdělit na více regálů a v jednom z nich zvýšit počet pater. I když by takový způsob stále odpovídal skutečnosti, vysvětlil jsem mu, že regál rozdělovat nemusel, a ukázal jsem mu funkci *Počet umístění na buňku*, kterou dosud nepoužil. V regálu jsem přidal o jedno patro navíc a ve všech ostatních sloupcích, kde bylo o jedno patro méně, jsem snížil počet umístění na 0 v nejvyšším patře. Účastník toto řešení pochopil, ale uznal, že by k takovému řešení sám nedošel.

Účastník dokázal použít funkci *Kopírování regálu* a nesnažil se každý regál tvořit úplně od začátku. Pochopil také, že jednotlivé symboly na každém konci regálu v mapě představovaly symbol sloupce, a s regály manipuloval tak, aby tyto symboly odpovídaly skutečnosti. Tyto symboly byly v konkrétním případě mapy velmi důležité, jelikož se každý regál jmenoval stejně.

Účastníka jsem nenechal vytvořit celou mapu, jelikož by to zabralo příliš mnoho času; zároveň jsme uznali, že by účastník dokázal mapu dotvořit sám a další pokračování by pro testování bylo méně přínosné, jelikož jsme prošli všemi základními body. S účastníkem jsem po celou dobu vedl diskuzi o různých funkcionalitách a věcech, které by v mapě rád měl nebo změnil. Pokusím se je shrnout do tohoto seznamu:

- Účastník by uvítal, kdyby nemusel každý regál v mapě rozklikávat, aby se dozvěděl, jaké jsou v něm rozměry. Radši by si na regál pouze najel myší nebo by se regál v mapě mohl přímo vybarvit nějakou barvou.

- Účastník by dále ocenil, kdyby se po přiřazení rozměrů v regálu automaticky počítala celková výška a šířka regálu. Pomohlo by mu to např. v situaci, kdyby rozměr umístění zadal špatně – pak by dle rozměrů celého regálu snadno zjistil, že zadal špatné hodnoty. Rozměry celého regálu by účastník rád viděl také v mapě, např. formou kót. Účastník by totiž mapu skladu rád používal k tomu, aby si mohl změřit nějakou vzdálenost, např. mezi dvěma regály, a zjistit, jestli se do mezery nevejde nový regál.

- Účastník chtěl také do mapy zaznamenat různé překážky a objekty, jako např. okno či sloup. Řekl, že by to určitě pomohlo s orientací ve skladu, jelikož by se v mapě nyní vyskytovaly nějaké záchytné body. Mapu skladu by tak účastník využil k tomu, aby např. nějakému novému zaměstnanci pomohl se ve skladu zorientovat. Proto by bylo vhodné také

mít možnost mapu exportovat do nějakého vytisknutelného formátu, jako např. PDF.

Obecně měl účastník z rozhraní dobrý pocit a dozvěděl jsem se, že by mapu rád používal nejen pro optimalizaci, ale také pro další osobní účely. Společně jsme uznali, že by bylo vhodné k mapě připravit nějaký jednoduchý manuál, aby nový uživatel věděl, co má dělat a jakým způsobem informace do mapy zadávat. Skóre SUS bylo překvapivě stejné jako u předchozího účastníka – 72,5. U otázky ohledně podpory technické osoby účastník dodal, že pokud by měl k dispozici manuál, podporu technické osoby by nejspíše nepotřeboval.

6.3.3 Verča

Verča byla také zástupcem primární persony. Pracovala v menším skladu o zhruba 12 regálech. Použil jsem scénář A.

Stejně jako u předchozího účastníka jsem i tomuto účastníkovi pomáhal a radil při úplných začátcích. Účastník zvládl vytvořit nový regál a zvládl základní manipulaci s ním – některé funkce účastník našel sám, na jiné jsem musel poukázat, že se v rozhraní nacházejí. Při pohledu na regál ze strany účastník rychle pochopil, jaké úpravy provést, aby správně seděly názvy umístění – názvy v rozhraní šlo vyjádřit přesně tak jako ve skutečnosti. Účastník tvořil regál o třech sloupcích, kde se za umístění pokládala celá police (patro) – opět chyběla funkce sloučení několika buněk, kterou by účastník rád použil, a místo toho jsem účastníkovi ukázal alternativní řešení, kdy jsem počet umístění ve všech ostatních sloupcích, kromě prvního, nastavil na 0.

Regály se ve skladu opakovaly a musel jsem poukázat na to, že lze již hotový regál zduplikovat – účastník se chystal tvořit regál od začátku a nenapadlo jej hledat nějaké jednodušší řešení. Vzhledem k celkovém počtu regálů ve skladu (12) by se pravděpodobně uživatel obešel bez nástroje na duplikaci, a zvládl by v rozumném čase všechny regály vytvořit – pro účely testování jsem však účastníka nechtěl příliš zatěžovat a radši poradil.

Opět jsme nedokončili mapu celou, ze stejných důvodů jako u Libora. Sice jsem musel opět účastníkovi pomáhat, ale později v testování jsem již příliš nezasahoval. Účastník do mapy dokázal zaznamenat umístění přesně tak, jak byla ve skutečnosti. Stejně jako u Libora účastníka mrzelo, že se v mapě nedaly zjistit další informace o nějakém regálu, aniž by si jej účastník rozkliknul a podíval se na něj ze strany. Účastník by dokonce preferoval mít detail regálu hned vedle okna s mapou – v případě daného skladu by to dávalo smysl, jelikož regály měly pouze 3 sloupce, a tak by se okno s detailem regálu a okno s mapou pravděpodobně vešly vedle sebe.

Účastník řekl, že v jejich skladu by nejspíš mapu nepoužívali – když jsem vysvětlil, že mapa by sloužila k optimalizacím, účastník tvrdil, že sklad je dostatečně malý na to, aby optimalizace konali skladníci sami. Zároveň bylo zboží skladováno takovým způsobem, kdy i záznam o rozměrech umístění ne-

byl příliš užitečný, jelikož zboží již bylo rozbalené a jednotlivé položky se různě poskládaly tak, aby se do police vešly. Systém by proto užitečné optimalizace ani nemohl navrhovat. Jelikož se jednalo o malý sklad, nebylo problémem se ve skladu zorientovat – mapa by proto neposloužila i pro tyto účely. Účastník však rozuměl účelu mapy ve větších skladech a dokázal si představit, že tam by se již mapa skladu velmi hodila.

Opět jsme došli k závěru, že by bylo vhodné vytvořit manuál, díky kterému bych účastníkovi nemusel pomáhat a pravděpodobně by práci zvládl sám. Účastníkovi se rozhraní poměrně líbilo a nepřipadalo mu příliš komplexní, skóre SUS bylo překvapivě opět 72,5.

Data z testování: Data z testování, jako např. načrtnuté mapy jak fiktivních, tak reálných skladů či emoční audiovizuální záznamy, naleznete v přiloženém médiu ve složce `testing`.

6.4 Výsledky testování použitelnosti

Z testování použitelnosti jsem měl mnoho poznatků. Pro zkrácení textu budu místo „zástupců primárních person“ používat jednoduše „primární persony“ apod. – nejedná se tedy o fiktivní postavy, ale jejich zástupce při testování:

Pochopitelnost: Sekundární persona artefakt pochopila dostatečně a dokázala správně vyjádřit konfigurace a rozměry přesně podle fiktivního plánku. Primární persony však z rozhraní nedokázaly vyčíst, jak funkce správně použít pro vyjádření konfigurací. Přiřazování rozměrů každá primární persona pochopila trochu jinak a snažily se funkci využít jiným způsobem. S primárními personami jsem se shodl na tom, že s manuálem nebo uživatelskou dokumentací by dokázali rozhraní správně používat, avšak správné řešení by mělo být komunikováno i přes samotné rozhraní.

Intuitivita: Všichni účastníci očekávali funkci táhnutí objektem po mapě a slučování/rozdělování buněk při pohledu na regál ze strany – obě tyto funkce chyběly. Na druhou stranu se účastníkům líbilo přiřazování rozměrů a vybarvování částí regálu, i když ze začátku nebyl přesný postup zjevný.

Konzistence ovládacích prvků: U některých tlačítek nebyla jasná jejich funkce. U 2 ze 3 účastníků se stalo, že omylem smazali rozměr, přičemž si mysleli, že tlačítkem pouze zavřou panel v rozhraní. Dále byly různé ovládací prvky v rozhraní špatně rozmístěny a všichni účastníci měli z počátku problém najít správný ovládací prvek – např. šířku regálu.

Popsání reality U reálných skladů šlo zaznamenat jmenovací systém přesně podle skutečnosti a všechny regály šlo v rozhraní znázornit, i včetně konfigurací.

Návrh nástrojů pro komfort: Libor by rád ocenil možnost zadat do mapy pojmenované překážky (okno, sloup), možnost automaticky počítat reálnou šířku regálu a tento rozměr zobrazovat jak při pohledu ze strany, tak i v mapě formou kót. Obě primární persony by také ocenily možnost inspekce regálu přímo v mapě bez nutnosti si daný detail regálu otevřít – žádanou funkcí bylo např. zobrazení rozměrů v daném regálu.

Přesnost dat pro optimalizační algoritmy: Artefakt uživatele směroval ke správnému řešení a při návrhu reálných skladů byl každý sklad zaznamenán tak, že optimalizační algoritmy měly k dispozici správná data.

Cílové sklady: Bylo dobré si ujasnit, že některé sklady mapu využijí a jiné zase ne – Libor by mapu skladu rozhodně využil, jelikož pracoval ve středně velkém skladu, a kromě optimalizací by mapu využil i na další účely, jako např. měření vzdáleností mezi regály ve skladu. Verča však pracovala v malém skladu, kde nebyly optimalizace potřeba a zároveň by mapa skladu neposloužila ani k jiným účelům, jako např. k orientaci ve skladu.

Použitelnost: Celkový průměr skóre SUS byl 72,5, což znamenalo, že z kvantitativního hlediska byla použitelnost rozhraní lehce nadprůměrná (hranice nadprůměrnosti/podprůměrnosti je 68, maximum bodů je 100). Artefakt byl zvolen dobře, avšak rozhraní je třeba zpřehlednit (přemístit a zvýraznit stávající prvky) a zvýšit jeho konzistenci (opravit chybné funkce tlačítek), zároveň měly primární persony problém s počátečním pochopením. Přes tyto nedostatky se však všichni účastníci během testování dokázali zorientovat a provádět změny poměrně rychle a jednoduše a z rozhraní všichni měli dobrý pocit. Zároveň šlo přesně popsat reálné sklady a jména všech umístění. Přihlédnutím ke kvantitativním (skóre SUS) a kvalitativním výsledkům bych rozhraní tedy prohlásil za dostatečně použitelné, přičemž jeho použitelnost lze ještě zásadně zvýšit.

6.5 Návrh změn v rozhraní

V této sekci shrnu navrhované změny v rozhraní, které vyplynuly z testování použitelnosti.

- Zhotovit manuál či uživatelskou dokumentaci dostatečně pochopitelnou pro primární personu
- Implementovat možnost záznamu překážek a zón do mapy (implementace těchto funkcí byla v plánu, ale nestihl jsem ji provést)
- Opravit nekonzistentní prvky v rozhraní (tlačítko s křížkem u okna s rozměrem, které ve skutečnosti smaže rozměr místo zavření panelu) a ně-

které prvky zvýraznit, aby se daly jednoduše nalézt (např. pole na ovládání šířky regálu)

- Přidat základní intuitivní funkce – pohyb objektů tažením a sloučení/rozdělení buněk při náhledu ze strany
- Přidat možnost dozvědět se některé informace o regálu před otevřením jeho detailu – např. všechny rozměry, které jsou v regálu zaznamenány
- Přidat komfortní funkce, jako např. průběžné počítání skutečné šířky regálu

Kapitola 7

Možnosti navázání a využití

Výsledný editor na tvorbu mapy skladu je dostatečně použitelný a frontendové řešení je již implementováno do rozhraní skladového systému. K plnému nasazení mapy však zbývá pár kroků:

Zvýšení použitelnosti: Lze dosáhnout vyšší použitelnosti nástroje implementováním navrhovaných změn (v sekci 6.5), které vzešly z testování použitelnosti.

Podpůrné rozhraní: V současné době bylo vytvořeno pouze rozhraní samotného editoru. Chybí však rozhraní na ukládání, tvoření či mazání map.

Backend: Backendové řešení vznikalo současně v rámci týmů SP1 a SP2, v době psaní této práce je však neaktuální. Chybí tedy backend jak na uložení a znovunačtení mapy, ale také backend s optimalizačními algoritmy, které z mapy budou vypočítávat vzdálenosti a navrhovat optimalizace.

Identifikoval jsem také další možnosti využití výstupů nástroje:

3D vizualizace: Výsledný JSON je formátován způsobem, kterým by šlo mapu jednoduše vizualizovat ve 3D. Vizualizace by mohla proběhnout např. pomocí pluginu do programu Blender nebo ji lze implementovat přímo do rozhraní skladového systému pomocí knihovny zaměřující se na trojdimenzionální prostor.

Heatmapy: Po zhotovení backendové části lze do UI aplikace Swordfish implementovat funkci heatmapy, což je vizualizace vytížení skladových umístění.

Import nových umístění do skladu: Pokud by se systém ve skladu zaváděl poprvé, jevilo by se dobrým řešením vytvořit mapu skladu a využít

možnosti automatické generace názvů umístění. Poté by se nová umístění dala do systému hromadně importovat.

Generace štítků: Každé umístění ve skladu je většinou označeno štítkem, na kterém je napsáno jméno umístění a další údaje. Uživatel by si do nové mapy mohl zaznamenat všechna umístění ve skladu, která by chtěl označit štítkem, a poté vygenerovat a vytisknout papírové archy se štítky. Lepším řešením se jevilo umožnit v mapě vybrat několik skladových objektů (např. regálů) najednou a poté nechat vygenerovat štítky pro všechny objekty ve výběru.

Export do různých formátů: Uživatelé by ocenili možnost vytisknout si mapu skladu např. ve formátu PDF. Užitečný by mohl být i export ve formátu CSV.

Závěr

Cílem tohoto projektu bylo nejdříve důkladně analyzovat problematiku mapy skladu a skladový systém *Atlantis* a poté navrhnout nové uživatelské rozhraní frontendu pro tvorbu mapy skladu s důrazem na použitelnost. Uživatelské rozhraní následně mělo být implementováno a otestováno, přičemž měly být navrženy vhodné testy, minimálně mělo být provedeno testování použitelnosti. Nakonec měly být shrnuty dosažené výsledky a navržena budoucí vylepšení.

Výsledkem práce je dostatečně použitelný a funkční frontendový nástroj pro tvorbu mapy skladu. Nejdříve jsem provedl důkladnou analýzu dané problematiky, při které jsem díky rozhovorům se skladovými experty a terénnímu průzkumu identifikoval skladové objekty a jejich vlastnosti. Poté jsem prozkoumal podobné aplikace pomocí heuristického průchodu a na základě poznatků stanovil prvotní artefakt.

V prototypové fázi jsem spojil návrh a implementaci a vytvořil jsem první prototyp, který jsem podrobil testování, zároveň jsem při tvorbě neustále komunikoval s vedoucím a projektovým manažerem. Na základě výsledků testování jsem stanovil potřebné úpravy artefaktu. Poté jsme se přesunul do fáze tvorby druhého prototypu, ve které jsem nástroj již implementoval do rozhraní skladového systému a zajistil jsem dostatečné prostředky pro komunikaci s backendem formou JSONu.

Řešení jsem v průběhu testoval akceptačními testy a ve finální fázi projektu jsem provedl testování použitelnosti, jehož se zúčastnili také reální skladoví zaměstnanci. Při testování jsem si ověřil, že rozhraní je dostatečně použitelné, zároveň jsem identifikoval problematické části rozhraní a navrhl jejich řešení.

Na práci lze navázat mnoha způsoby. Nejprve lze vyřešit problematické části rozhraní a zlepšit tak použitelnost. Dále lze zhotovit backend, který by zajistil ukládání a načítání mapy a využil data z mapy pro návrh vhodných optimalizací. Výstupy nástroje lze také využít na zhotovení různých vizualizací, jako např. heatmapy či 3D vizualizace.

Bibliografie

1. CHRISTENSSON, Per. User Interface Definition. In: *TechTerms* [online]. 2009 [cit. 2022-04-11]. Dostupné z: `https://techterms.com/definition/user_interface`.
2. MACKENZIE, I. Scott. *Human-computer interaction: An empirical research perspective*. San Francisco: Morgan Kaufmann Publishers Inc., c2013. ISBN 978-0-12-405865-1.
3. NORMAN, Donald A. *The design of everyday things: revised and expanded edition*. New York: Basic Book, c[2013]. ISBN 978-0-465-05065-9.
4. ABRAS, Chadia; MALONEY-KRICHMAR, Diane; PREECE, Jenny. User-Centered Design. In: *Encyclopedia of Human-Computer Interaction*. Thousand Oaks: Sage Publications, 2004, s. 445–456.
5. COOPER, Alan; REIMANN, Robert; CRONIN, Dave; NOESSEL, Christopher. *About face: the essentials of interaction design*. Fourth edition. Indianapolis: Wiley, [2014]. ISBN 978-1-118-76657-6.
6. SCHMIDT, Jan. *Průzkum kontextu* [online]. c2022 [cit. 2022-04-16]. Dostupné z: `https://courses.fit.cvut.cz/BI-TUR/lectures/files/TUR4NGkontext.pdf`.
7. NIELSEN, Jakob. Usability 101: Introduction to Usability. In: *Nielsen Norman Group* [online]. 2012 [cit. 2022-04-11]. Dostupné z: `https://www.nngroup.com/articles/usability-101-introduction-to-usability/`.
8. MORAN, Kate. Usability Testing 101. In: *Nielsen Norman Group* [online]. 2019 [cit. 2022-04-28]. Dostupné z: `https://www.nngroup.com/articles/usability-testing-101/`.
9. SCHMIDT, Jan. *Testování použitelnosti* [online]. c2022 [cit. 2022-04-28]. Dostupné z: `https://courses.fit.cvut.cz/BI-TUR/lectures/files/TUR8usability.pdf`.

10. Usability Testing. In: *Interaction Design Foundation* [online]. [B.r.] [cit. 2022-04-28]. Dostupné z: https://www.interaction-design.org/literature/topics/usability-testing.

11. DAM, Rikke Friis; SIANG, Teo Yu. Design Thinking: Get Started with Prototyping. In: *Interaction Design Foundation* [online]. 2021 [cit. 2022-05-02]. Dostupné z: https://www.interaction-design.org/literature/article/design-thinking-get-started-with-prototyping.

12. BABICH, Nick. Prototyping 101: The Difference between Low-Fidelity and High-Fidelity Prototypes and When to Use Each. In: *Adobe Blog* [online]. 2017 [cit. 2022-04-28]. Dostupné z: https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use.

13. NIKOLAIEVA, Aliona. 8 Best Software Development Methodologies. In: *Uptech* [online]. [B.r.] [cit. 2022-05-02]. Dostupné z: https://www.uptech.team/blog/software-development-methodologies.

14. TALÁR, Denis. *Optimalizace nejčastějších procesů ve skladovém systému*. Praha, 2020. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

15. CVRČEK, Jan. *Rozšíření skladového systému Atlantis*. Praha, 2021. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

16. NIELSEN, Jakob; MACK, Robert L. *Usability Inspection Methods*. USA: John Wiley & Sons, Inc., 1994. ISBN 0471018775.

17. MLEJNEK, Jiří. *Analýza a sběr požadavků* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/506241/mod_resource/content/7/03.prednaska.pdf.

18. NIELSEN, Jakob. Response Times: The 3 Important Limits. In: *Nielsen Norman Group* [online]. 1993 [cit. 2022-05-08]. Dostupné z: https://www.nngroup.com/articles/response-times-3-important-limits/.

19. MALEC, Oldřich. *Frontend skladového systému*. Praha, 2020. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

20. Using multiple Javascript frameworks in a project? In: *Stack Overflow* [online]. [B.r.] [cit. 2022-04-09]. Dostupné z: https://stackoverflow.com/questions/1963828/using-multiple-javascript-frameworks-in-a-project.

21. Vue 3. In: *Github* [online]. 2022 [cit. 2022-05-02]. Dostupné z: https://github.com/vuejs/docs.

22. Vuetify. In: *Github* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://github.com/vuetifyjs/vuetify.

23. Material Design. In: *Interaction Design Foundation* [online]. [B.r.] [cit. 2022-05-05]. Dostupné z: https://www.interaction-design.org/literature/topics/material-design.

24. PATIL, Abhijit. *Why Use Material Design for Your Next Web Design Project* [online]. 2020-09-18 [cit. 2022-05-05]. Dostupné z: https://blog.thedigitalgroup.com/why-use-material-design-for-your-next-web-design-project.

25. MOSS, Ben. *Why don't we just use Material Design?* [Online]. 2019-04-18 [cit. 2022-05-05]. Dostupné z: https://www.webdesignerdepot.com/2019/04/why-dont-we-just-use-material-design/.

26. Vuex. In: *Github* [online]. 2022 [cit. 2022-05-07]. Dostupné z: https://github.com/vuejs/vuex.

27. What Is Acceptance Testing (A Complete Guide). In: *Software Testing Help* [online]. 2022 [cit. 2022-04-30]. Dostupné z: https://www.softwaretestinghelp.com/what-is-acceptance-testing/.

28. NIELSEN, Jakob. Why You Only Need to Test with 5 Users. In: *Nielsen Norman Group* [online]. 2000 [cit. 2022-04-30]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

29. *System Usability Scale (SUS)* [online] [cit. 2022-05-01]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CSS Cascading Style Sheets

CSV Comma Separated Values

DOM Document Object Model

FIT Fakulta informačních technologií

HCI Human-computer interaction

HTML Hypertext Markup Language

JSON Javascript Object Notation

PDF Portable Document Format

SP1 Softwarový projekt 1

SP2 Softwarový projekt 2

SUS System Usability Scale

SVG Scalable Vector Graphics

UCD User-centered design

UI User interface

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `testing` ..... ukázka výsledků testování
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `BP_Kopecky_Vojtech_2022.tex` ..... hlavní soubor
- `BP_Kopecky_Vojtech_2022.pdf` ..... text práce ve formátu PDF