FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Frontend importu a exportu dat skladového systému
Student:	Samuel Švalich
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem této práce, je tvorba uživatelského rozhraní a následná implementace frontendu skladového systému za účelem snadného importu, popřípadě exportu dat. Důraz je kladen na univerzálnost a snadné použití pro budoucí uživatele.

Postupujte v těchto krocích:
1. Řádně prostudujte jednotlivé procesy stávajícího řešení s ohledem na téma této práce.
2. Proveďte analýzu požadavků zahrnující proces importu a exportu dat ze skladového systému.
3. Na základě analýzy skladového systému a studia již existujícího backendového návrhu proveďte vhodný návrh budoucího frontendu.
4. Dle návrhu implementujte první verzi funkčního prototypu.
5. Otestujte první verzi prototypu a navrhněte vylepšení a úpravy.
6. Na základě testů a poznatků implementujte výsledný frontend importu / exportu.
7. Opět otestujte.
8. Shrňte dosažené výsledky, navrhněte další budoucí úpravy.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 6. února 2023 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Frontend importu a exportu dat skladového systému

Samuel Švalich

Katedra softwarového inženýrství
Vedoucí práce: Jiří Hunka

10. května 2023

Poděkování

Tímto bych chtěl poděkovat všem co mi při tvorbě této práce pomohli. Největší dík patří vedoucímu práce Ing. Jiřímu Hunkovi za veškerou zpětnou vazbu, kterou mi při tvorbě práce neustále poskytoval. Dále bych také rád poděkoval Ing. Oldřichu Malcovi za jeho trpělivost při opakovaném code review, jež mi během vývoje poskytoval. Také bych rád poděkoval Ing. Markovi Erbenovi, který vytvářel jak BE pro nový celek, tak i aktualizovanou dokumentaci, díky které jsem mohl jednoduše a bez problému pracovat. V neposlední řadě bych rád poděkoval svým přátelům Daně Suchomelové, Janu Trojákovi a Danielu Hromádkovi, kteří mi při práci dodávali psychickou podporu. Nakonec bych rád poděkoval své rodině, jež mi poskytla podmínky ke studiu a dodala finální kontrolu celé práce.

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. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

dne 10. května 2023 . . . . . .

Odkaz na tuto práci

Švalich, Samuel. *Frontend importu a exportu dat skladového systému.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Práce se zabývá vývojem frontendu pro skladový systém Atlantis, konkrétně frontendem pro import a export dat systému. Nejprve proběhla analýza aktuálního stavu systému Atlantis a jeho řešení pro import a export dat. Poté byl vytvořen návrh nového uživatelského rozhraní a tento návrh realizován a otestován. Na základě testování vytvořeného řešení byla navržena druhá verze a proces vývoje se opakoval. Na konci vývoje druhé verze bylo řešení opět otestováno a připraveno k nasazení do systému Atlantis, kde rozšíří možnosti exportu a přidá novou možnost importu dat systému přímo pomocí jeho rozhraní. Uživateli tak ušetří čas při importu dat oproti aktuálnímu řešení.

Klíčová slova webová aplikace, frontend, skladový systém, import dat, export dat, Vue.js, Vuetify

Abstract

The thesis deals with the development of a frontend for the Atlantis warehouse system, specifically a frontend for importing and exporting system data. First, the current state of the Atlantis system and its data import and export solution was analyzed. Then a new user interface design was created and this design was implemented and tested. Based on the testing of the developed solution, a second version was designed and the development process was repeated. At the end of the development of the second version, the solution was again tested and is ready to be deployed into Atlantis, where it will extend the export capabilities and add a new option to import system data directly using its interface. This will save the user time when importing data compared to the current solution.

Keywords web application, frontend, storage system, import, export, Vue.js, Vuetify

Obsah

Seznam obrázků

Úvod

Webové aplikace jsou v posledních letech ve světě informačních technologií stále více využívány jako hlavní způsob distribuce programů pro jejich uživatele. Aplikace se neustále posouvají dále a to zejména co se uživatelského rozhraní týče. V této práci se budu zabývat vývojem části uživatelského rozhraní právě jedné z těchto webových aplikací. Jedná se o aplikaci skladového systému Atlantis.

Webová aplikace Atlantis v aktuální podobě postrádá uživatelské rozhraní pro možnosti importování dat do systému a rozhraní pro export poskytuje pouze velice základní možnosti. Zákazníci aplikace vznesli požadavky na vytvoření rozhraní pro importování dat, společně s požadavky na rozšíření možností aktuálního exportu dat. Nové rozhraní pro obě tyto funkcionality by nepochybně zjednodušilo práci nejen uživatelů aplikace, ale i týmu, jež aplikaci vyvíjí. Je tedy zapotřebí toto uživatelské rozhraní vytvořit.

Vytvořením nového rozhraní pro import a export dat ze systému bude ušetřen uživatelům čas a náklady, které jsou zbytečně využívány při aktuálním řešení problému. Také se tím zjednoduší práce vývojářů aplikace, kteří jsou zbytečně vytíženi aktuálním průběhem importu dat do systému.

Motivací této práce je tedy ušetřit čas a náklady na straně uživatelů aplikace, stejně tak jako na straně vývojářů. Má osobní motivace pro zvolení právě tohoto projektu je možnost osobního rozvoje při práci s technologiemi projektu Atlantis, společně se zkušenostmi nabytými při práci v reálném komerčním projektu.

Cílem této práce je nastudovat současný stav skladového systému a jeho aktuální řešení pro import a export dat. Poté nasbírat a analyzovat požadavky na uživatelské rozhraní těchto funkcionalit. Na základě nasbíraných požadavků vytvořit návrh tohoto rozhraní a na závěr tento návrh

realizovat a otestovat. Cílem práce bude také na základě výstupu testování navrhnout druhou verzi uživatelského rozhraní, která bude opět realizována a otestována.

Práce se bude tedy ze začátku zabývat analýzou aplikace Atlantis a aktuálním řešením importu a exportu dat. Po analýze se bude zabývat vytvořením první verze řešení, které se bude skládat ze sběru požadavků, návrhu rozhraní, implementace a na závěr testování a zhodnocení této verze řešení. Tento postup se poté bude celý opakovat pro verzi druhou.

Práce bude zapojená do aplikace Atlantis, ve které aktuálně existuje velice základní rozhraní pro export dat a žádné rozhraní pro import většího množství dat. Při vývoji budu spolupracovat se zaměstnancem aplikace Atlantis Ing. Markem Erbenem, který bude vyvíjet BE (backend) nových funkcionalit.

Kapitola 1

Analýza

V této kapitole popíši modely softwarového vývoje, které jsem zvažoval a pak přiblížím můj vlastní potup při vývoji. Dále představím skladový systém Atlantis a přiblížím jeho aktuální stav řešení s ohledem na import a export dat. Také popíši technologie, které systém používá a se kterými budu také pracovat.

1.1 Modely softwarového vývoje

Model softwarového vývoje popisuje způsob, jakým lze organizovat a řídit vývoj softwaru. Tyto modely mají za úkol zlepšit kvalitu finálního produktu, maximalizovat čas strávený vývojem a snížit náklady na průběh vývoje. Některé z používanějších modelů softwarového vývoje jsou například Vodopád, Iterační model, nebo spirálový model. Dále se budu zabývat modelem iteračním a vodopádem, jelikož byl můj vlastní postup těmito modely inspirován [1].

1.1.1 Vodopád

Vodopád je jedním z prvních uznávaných vývojových modelů zaměřující se na sekvenční vývoj. Sekvenční vývoj probíhá v jednotlivých krocích, které by se navzájem neměly překrývat, aby se tedy mohla začít další fáze musí být předchozí zcela dokončena. Jednotlivé fáze vývoje se v tomto pořad dělí na :

- Analýza požadavků - V této fázi se sbírají a analyzují požadavky na software. Tato fáze by měla přiblížit přesně jaké chování je od softwaru očekáváno, ale stejně tak jaké chování už očekáváno není.

- Návrh - Tato fáze je zaměřena na vytvoření návrhu na základě našich nasbíraných požadavků. Výstupem by mě být co nejpřesnější návrh, abychom se při implementaci vyhnuli budoucím nejasnostem.

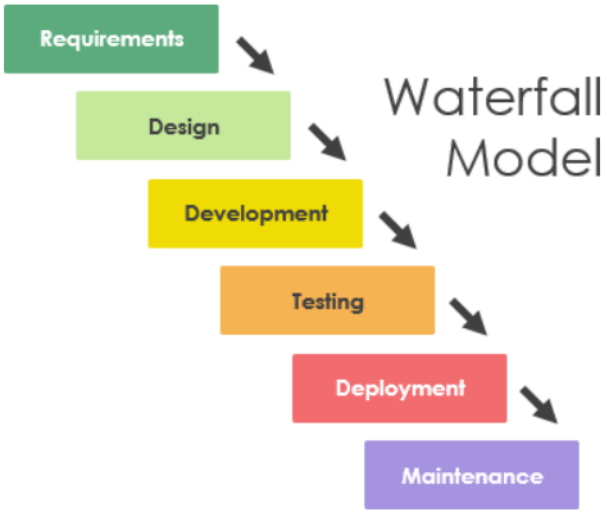
- Implementace - V této fázi se na základě návrhu realizuje náš software. Výsledkem by měl být funkční celek, který bude splňovat všechny požadavky zadavatele.

- Integrace a testování - Tato fáze slouží k integraci celku, pokud je to potřeba a následnému otestování. Na konci bychom tedy měli mít plně funkční a především otestovaný celek.

- Nasazení - Nejkratší fáze je nasazení, jedná se pouze o nasazení do zákazníkova prostředí, nebo vydání na trh. Výstupem je tedy plně funkční vydaný celek.

- Udržování - Poslední a nejdelší fází je pak údržba našeho softwaru, která může zahrnovat vydávání nových verzí, aby byl celek v souladu se zákonem, opravu různých chyb, nebo integraci do nového prostředí.

Hlavní výhodou této metody je přehledný a definovaný průběh vývoje, naopak nevýhodou je těžká adaptace na změny [2]. Graficky znázorněný průběh vývoje můžete vidět zde 1.1



Obrázek 1.1: Model Vodopádu [3]

1.1.2 Iterační metoda

Iterační model probíhá na rozdíl od vodopádu v cyklech. Hlavní myšlenka je vyvinout celek po menších funkčních částech a vyhnout se tak mohutné jednorázové implementaci. Průběh začíná stejně jako u vodopádu a to sběrem a analýzou požadavků. Poté se však přesuneme do prvního cyklu, který probíhá v následujících sekvenčních krocích :

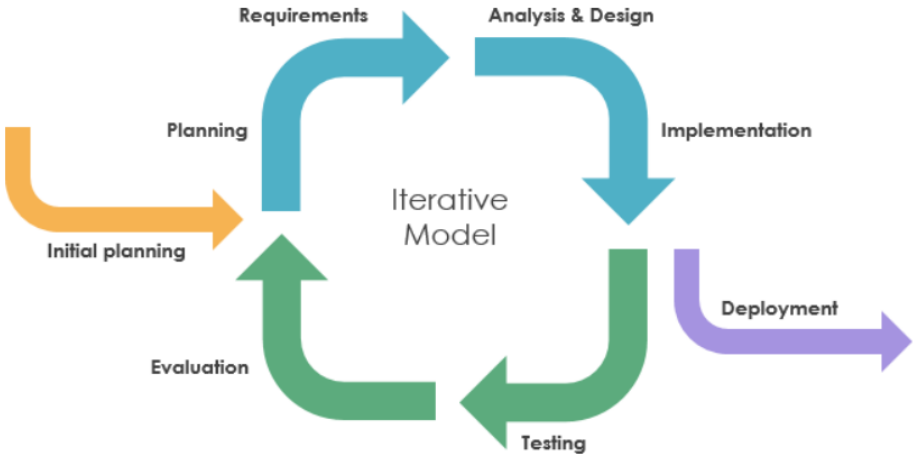
- Plánování a Analýza - Součástí tohoto kroku je především definovat jaká část celku se bude v tomto cyklu přidávat a udělat její analýzu. Měli bychom tedy přesně vědět jaké požadavky budeme v tomto cyklu dodávat.
- Design - V tomto kroku vytvoříme pomocí definovaných požadavků návrh našeho naplánovaného celku. Výstupem je tedy detailní návrh části, která se bude v této iteraci přidávat.
- Implementace - Tento krok slouží k implementaci návrhu a zasazení do zbytku projektu. Výstupem by tedy měl být celek, který splňuje všechny dosud naplánované požadavky společně s aktuální iterací.
- Testování - Tento krok se zaměřuje na testování, jak nových funkcionalit, tak jejich integraci do zbytku projektu. Výsledkem je tedy funkční a otestovaný celek.
- Zhodnocení - Poslední krok v iteraci je zhodnocení průběhu iterace, její silné a slabé stránky a možnosti na zlepšení do dalšího cyklu.

Poté co dokončíme náš poslední cyklus a máme celek, který splňuje veškeré požadavky od zadavatele, pokračujeme poslední dva kroky stejně jako u vodopádu. Nasadíme náš celek do prostředí klienta, nebo vydáme na trh a poté už jen zajišťujeme údržbu.

Na rozdíl od vodopádu je v iterační metodě výhodou možnost adaptace při vývoji, nevýhoda je pak to, že byla metoda určena k vývoji velkých celků a tak se složitě aplikuje na malé špatně dělitelné projekty [4]. Grafické znázornění iterační metody můžete vidět zde1.2

1.1.3 Zvolená metoda

Mnou zvolená metoda vývoje byla kombinací dvou výše zmíněných metod. Postup probíhal v časovém rozložení iterační metody, bylo tedy více opakujících se cyklů, které proběhly před vytvořením finální verze. Každý cyklus však probíhal jako menší vodopád a to v tom smyslu, že jsem vytvářel



Obrázek 1.2: Model Iterativní Metody [3]

celek, jež splňoval všechny zadané požadavky a ne jen jejich část, jako v metodě iterační. Na konci každého průchodu cyklem tak byly splněny všechny požadavky a proběhlo testování a na závěr zhodnocení iterace a jejího průběhu. Můj zvolený průběh vývoje se také odráží přímo ve struktuře textu práce, který je stejně jako vývoj též rozdělen podle vyvíjených iterací.

1.2 Současný stav skladového systému

Skladový systém Atlantis je aplikace, která spravuje systém skladů a jejich zaměstnance. Součástí aplikace je digitalizace položek napříč všemi sklady umožňující přehled nad stavy jednotlivých skladů. Systém také spravuje zásilky skladů, dodavatelé, zákazníky, výrobce, dopravce a pracovníky. Jednotlivým pracovníkům skladu jsou v aplikaci přiřazovány úkoly podle jejich rolí [5]. Pracovníci jsou v systému děleni na následující role :

- Vedoucí - Spravuje položky ve skladu, zásilky, jednotlivé sklady a jejich pobočky a umístění, Dále se stará o správu vlastníků pobočky, výrobců, dodavatelů, dopravců a odběratelů.

- Skladník - Provádí hlavně fyzickou realizaci zadaných úkolů. Ty jsou mu přiřazeny vedoucím, nebo si je sám vybere a přiřadí.

- Balič - Balič je nejmenší rolí v Atlantis. Uživatelé s touto rolí mohou balit zásilky.

1.2.1 Architektura aplikace

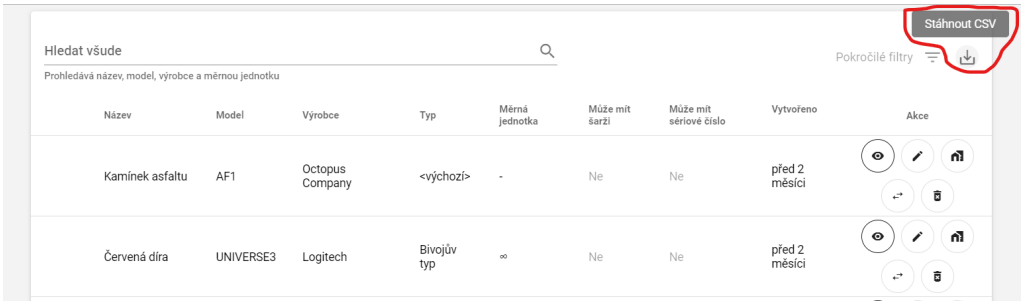
Aplikace využívá velice populární architekturu REST API, jež dělí aplikaci na dvě hlavní části klient a server. Klient posílá na server požadavky ve formě HTTP requestů. Server tyto požadavky zpracovává a odpovídá na ně. V případě skladového systému Atlantis se celek dělí na dva projekty, projekt Octopus (server) a projekt Swordfish (klient). Tato práce se bude zabývat převážně projektem Swordfish, vzhledem k tomu, že do tohoto projektu zapadá celek, jež bude vytvářen.

1.2.2 Aktuální export dat

V aktuálním stavu aplikace existuje jednoduché řešení exportu dat. Toto řešení dává uživatelům možnost exportovat data z tabulek aplikace do formátu CSV. Aktuální export však využívá dat určených k zobrazování tabulky, nad kterou export provádíme. Nejedná se tedy o data nijak připravená k exportu, ale pouze o přetransformovaný JSON, jež nám vrátí API, do CSV formátu. Zde můžete vidět aktuální rozhraní exportu 1.3

Hlavními problémy stávajícího řešení jsou tedy :

- Možnost exportovat pouze do formátu CSV.
- Uživatel nemá možnost export nijak konfigurovat (podle data nebo konkrétního skladu)
- Neexistuje přehled nad již exportovanými daty.



Obrázek 1.3: Aktuální export

1.2.3 Aktuální import dat

Import většího množství dat není pomocí aktuálního rozhraní aplikace nijak pohodlně možný. Pokud chceme importovat například nové zákazníky,

jedna z možností je použít script, jenž pomocí endpointů na vytvoření nebo, úpravu zákazníka po jednom provede požadovanou akci. Tento proces je potřeba začít manuálně za pomoci vývojářů aplikace. Je to tedy jeden z hlavních nedostatků stávajícího řešení.

1.3 Technologie projektu

Nyní přiblížím hlavní technologie projektu Swordfish, zabývající se FE aplikace Atlantis. Na těchto technologiích je projekt založen a musel jsem s nimi tedy pracovat i já při vývoji nových funkcionalit. Nebudou zde zmíněny veškeré knihovny, které se v projektu využívají, ale jen ty podstatné.

1.3.1 Javascript

Aplikace Swordfish je aktuálně napsaná v jazyce Javascript, a i když se v blízké budoucnosti plánuje přepsat do jazyku Typescript, mnou vytvořený celek byl zatím také napsán v jazyce Javascript.

Javascript je dynamický programovací jazyk nejčastěji využívaný pro tvorbu webových stránek. Konkrétněji se především využívá pro tvorbu klientské strany webových aplikací. Jedná se o interpretovaný programovací jazyk s objektově orientovanými prvky. [6]

1.3.2 Vue.js

Celý projekt FE (Swordfish) pro skladový systém ja napsaný ve Vue2 (verze Vue.js). Vue.js je javascriptový famework pro vytváření UI, jedná se o nadstavbu nad klasickým html, css a javascriptem. Jeho hlavní výhodou je tvorba komplexních celků pomocí předvytvořených znovu použitelných komponent, dává nám tak možnost dělit velké složité celky na menší jednodušší problémy. Další výhodou je jednoduše docílitelná reaktivita elementů, které se samy překreslují a reagují na uživatele [7]. Vue zjednodušuje vývoj hlavně díky těmto prvkům :

- Computed property - Jedná se o speciální funkci navracející hodnotu, tuto funkce se však nikdy nevolá ručně, naopak se zavolá pokaždé když se změní jakákoliv proměnná potřebná k jejímu výpočtu. Tímto je zajištěno, že jsou computed proměnné vždy aktualizovány podle stavu komponenty a díky tomu jsou skvělými prvky pro získání snadné responzivity.

- Watcher - Funkce typu watcher jsou manuální způsob jak reagovat na změnu dat v komponentě. Pokud vytvoříme funkci watcher s názvem nějaké proměnné, funkce se zavolá při každé změně hodnoty této proměnné.
- Props a Emit - Jednou z nejzákladnějších funkcionalit Vue jsou props a emits, dávají nám možnost předávat informace mezi jednotlivými komponentami. Props předávají informace z rodičovských komponent do potomků a emits vysílají eventy z potomků do rodičů.

1.3.3 Vuetify

Projekt je stejně tak jako na samotném Vue.js postavený na knihovně Vuetify. Vuetify je knihovna s předem vytvořenými komponentami pro Vue. Tyto komponenty, pokud jsou použity dobře a globálně po celém projektu, spolu velice dobře fungují a řeší nepříjemné problémy jako reaktivní změnu velikosti elementů v závislosti na velikosti okna prohlížeče. Jednoduše se pomocí nich dodržuje konzistence napříč celým projektem a díky možnosti je na míru upravovat slouží jako solidní základ pro komplexní a dynamické komponenty [8].

Kapitola 2

1. Verze

2.1 Sběr a analýza požadavků

Analýza a sběr požadavků je důležitá část procesu vývoje softwaru. V této části je určeno co přesně se od vyvíjeného softwaru očekává. Jedná se tedy o proces zjištění požadavků kladených na vyvíjený software a následné kategorizování těchto požadavků do různých skupin podle jejich charakteru.

Požadavky se dají dělit podle různých vlastností například na funkční a nefunkční požadavky, kde funkční požadavky označují požadavky, které vyžadují co přesně má aplikace umět (například exportovat data, nebo zobrazit tabulku exportů). Nefunkční požadavky jsou pak požadavky na to, jak má aplikace pracovat (Využité technologie, integrace do určitého prostředí) [9].

2.1.1 FURPS

Jednou z možností kategorizace funkčních a nefunkčních požadavků je na základě typu přínosu, jež jejich splnění přinese programu. Toto dělení se nazývá FURPS a jeho název je akronymem pro kategorie do kterých se požadavky dělí funkcionality, usability, reliability, performance, upportability.

- Funkcionality - Popisuje požadavky na hlavní funkce programu, tedy co bude program dělat pro uživatele.

- Usability - Řeší požadavky na použivatelnost celku, tedy jak budou ostatní programy, nebo přímo uživatelé interagovat s programem.

- Reliability - Zabývá se požadavky na spolehlivost programu, tedy jaké chyby by se mohli při použití objevit a jak na ně bude program reagovat.
- Performance - Popisuje požadavky na výkonnost programu, jak rychle musí program reagovat na požadavky.
- Supportability - Řeší požadavky na podporovatelnost programu po jeho dokončení, jak jednoduché je ho vylepšovat nebo opravovat.

2.1.2 MoSCoW

Další možnost dělení požadavků je na základě důležitosti jejich splnění. Na toto dělení se používá MoSCoW model, jehož název je opět akronymem pro kategorie, do kterých požadavky dělí, must have, should have, could have, won't have.

- Must have - Požadavky, které musí být přednostně splněny.
- Could have - Požadavky, které mají vysokou prioritu a měli by být pokud možno splněny.
- Must have - Požadavky s nižší prioritou, které by mohli být splněny.
- Must have - Požadavky, které nebudou splněny.

2.1.3 Průběh sběru a analýzy požadavků

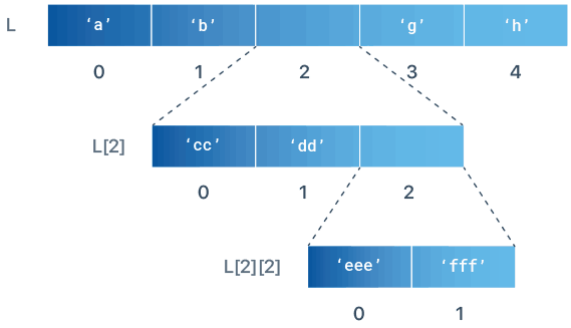
Požadavky jsem získal formou interview s Ing. Markem Erbenem, který se tímto problémem již začal zabývat ze strany BE a měl tak i připravený návrh API. Během této schůzky mi byl přiblížen stav BE řešení a požadavky především na funkcionalitu FE. Také jsem byl informován o datových strukturách, se kterými jsem měl při tvoření návrhů pracovat.

Pomocí již zmíněných způsobů kategorizace (FURPS, MoSCoW) budou nyní představeny požadavky, jež budou v první verzi splněny. V případě této práce bude do kategorizace přidána speciální kategorie a to pro rozdělení požadavků pro export, import, nebo pro obě funkcionality, abychom se vyhnuli zbytečnému opakování společných požadavků.

2.1.4 Požadavky

Jak jsem již zmínil cílem práce je vytvořit celek pro import a export dat skladového systému. Informace se pro účel FE nachází v doménách, které

si můžeme představit jako entity v relační databázi, mají tedy atributy, unikátní atributy a primární klíče. Máme tedy entitu a její atributy, pro přehlednost se však atributy dále dělí na podskupiny atributů. Každý atribut tedy může být skalár (samostatný atribut) nebo skupina dalších atributů. Jedná se o zanořený list atributů, jehož grafické znázornění můžete vidět zde 2.1



Obrázek 2.1: Nested List [10]

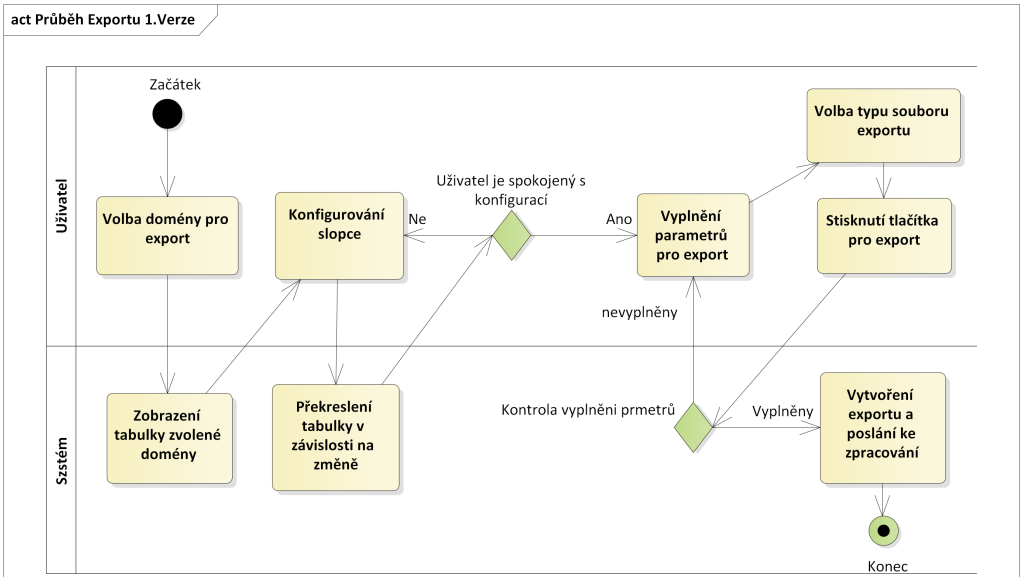
Každá doména také může mít list parametrů, které jsou použity jako omezení pro import a export. Představme si, že chceme exportovat uživatele naší aplikace, kteří se připojili poslední rok, parametrem by pak byl rok připojení uživatele do aplikace.

2.1.5 Export

Průběh exportu by měl probíhat následovně, uživatel si zvolí doménu k exportu a typ souboru, do kterého bude chtít exportovat. Na základě domény se mu zobrazí tabulka s falešnými daty, kde si bude moci zvolit, které sloupce bude chtít exportovat a které bude chtít vynechat. Nakonec pokud má doména parametry pro export tak je uživatel vyplní a poté si nechá data vyexportovat.2.2

2.1.6 Import

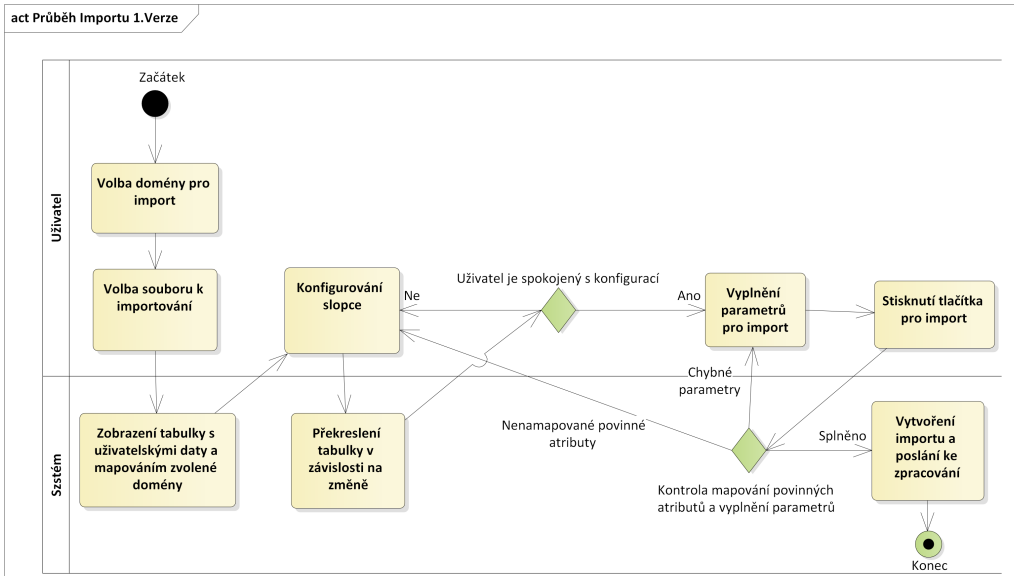
Průběh importu by měl být následovný. Uživatel si zvolí doménu, do které se chystá importovat a zvolí soubor, který chce importovat. V dalším kroku se mu zobrazí jeho soubor v tabulce s informacemi o namapování jeho dat na doménu. Také se uživateli zobrazí požadavky pro mapování dat, tedy povinné atributy domény, které musí být namapovány, aby mohl import



Obrázek 2.2: Activity diagram: Export

proběhnout. Tehdy uživatel domapuje nenamapované, nebo špatně namapované sloupce podle jeho dat.

Dále pokud bude mít doména parametry pro import bude je muset uživatel opět vyplnit jako u exportu. Pokud jsou parametry vyplněny a všechny povinné atributy namapovány, může být soubor importován.2.3



Obrázek 2.3: Activity diagram: Import

2.1.7 Funkční požadavky

FP1: Zobrazení existujících Importů/Exportů

Uživatel musí mít možnost si přehledně zobrazit již existující importy/exporty a hlavní informace o nich.

Priorita: Must have
Modul: Import/Export

FP2: Mazání Importů/Exportů

Uživatel by měl mít možnost smazat již nepotřebné importy/exporty.

Priorita: Should have
Modul: Import/Export

FP3: Možnost zvolení domény

Při vykonávání obou akcí importu a exportu je potřeba, aby si uživatel mohl zvolit, nad jakou z možných domén chce akci vykonávat.

Priorita: Must have
Modul: Import/Export

FP4: Vyplnění parametrů

Akce importu a exportu je možné nadále upravovat pomocí parametrů, je tedy potřeba dát uživateli možnost tyto parametry během obou procesů vyplnit.

Priorita: Must have
Modul: Import/Export

FP5: Volba typu souboru

Při exportu je potřeba dát uživateli možnost zvolit do jakého typu souboru si přeje export provést. Na výběr byli zvoleny typy xlsx, csv, ods. Během exportu musí být tedy možnost zvolit jeden z těchto typů pro export.

Priorita: Must have
Modul: Export

FP6: Modifikace exportu

Při exportu je zapotřebí dát uživateli možnost upravit si exportovaná data, a to ve formě zvolení, jaké sloupce tabulky se budou exportovat a jaké budou vynechané. Při exportu tedy chceme dát uživateli možnost zvolit sloupce k exportu a exportovat pouze tyto sloupce.

Priorita: Must have
Modul: Export

FP7: Nahrání Souboru

Pro akci importu je potřeba dát uživateli možnost nahrát soubor, který chce importovat.

Priorita: Must have
Modul: Import

FP8: Kontrola nahraného souboru

Po nahrání souboru uživatelem je potřeba zkontrolovat, jestli soubor splňuje určité požadavky na import. Především správný datový typ souboru.

Priorita: Should have
Modul: Import

FP9: Zobrazení uživatelského souboru

Při importu je potřeba zobrazit uživateli jeho zvolený soubor. Jelikož může uživatel nahrát jakýkoliv soubor správného datového typu, je potřeba tyto soubory nějak jednotně zobrazovat.

Priorita: Must have
Modul: Import

FP10: Zarovnání tabulky

Jelikož uživatelský soubor může obsahovat informace i mimo data k importu je potřeba dát uživateli možnost nám sdělit, který je první řádek obsahující data k importu stejně tak, jako počet řádků v hlavičce importovaných dat.

Priorita: Must have
Modul: Import

FP11: Zobrazení mapování souboru na doménu

Hlavní částí importu je mapování uživatelského souboru na doménu, tedy propojení uživatelského souboru se strukturou domény, do které se snažíme importovat. Je tedy potřeba u uživatelského souboru, zobrazeného v tabulce, přehledně zobrazit jaké sloupce tabulky jsou namapovány na jaké atributy domény.

Priorita: Must have
Modul: Import

FP12: Mapování tabulky na doménu

Pokud máme tedy zobrazen uživatelský soubor ve formě tabulky, společně s mapováním sloupců je potřeba dát uživateli nástroj k přemapování těhto sloupců. Uživatel musí mít tedy možnost u každého sloupce zvolit na jaký atribut domény ho chce namapovat.

Priorita: Must have
Modul: Import

FP13: Primární klíče

Při procesu mapování sloupců tabulky je také potřeba zvolit jaké sloupce budu použity jako primární klíče. Je tedy potřeba dát uživateli možnost u každého sloupce zvolit jedná-li se o primární klíč.

Priorita: Must have
Modul: Import

2.1.8 Nefunkční požadavky

NP1: Zapojení do projektu Swordfish

Nové funkcionality budou zapojeny do již existujícího projektu Swordfish odkud budou později používány.

NP2: Technologie Vue.js

Nový celek bude stejně jako zbytek projektu psaný v jazyce javascript s nadstavbou Vue.js

NP3: Konzistence se zbytkem projektu

Nové funkcionality budou vytvořeny pomocí Vuetify, nebo již existujících komponent tak, aby byly konzistentní se zbytkem projektu.

NP4: Reaktivita

Uživatelské rozhraní musí být reaktivní a počítat s dynamickými rozměry zařízení, na kterých bude používáno.

NP5: API

Nové funkcionality budou využívat API projektu Octopus stejně jako zbytek projketu.

2.2 Vytvoření Návrhů

Vytváření návrhů v první verzi bylo provedeno velice minimalisticky a sloužilo spíše jako proof of concept. Byli tak navrženy jen zajímavější a složitější části celku, spíše než návrh celého rozhraní, za předpokladu, že se co nejrychleji vytvoří první funkční verze.

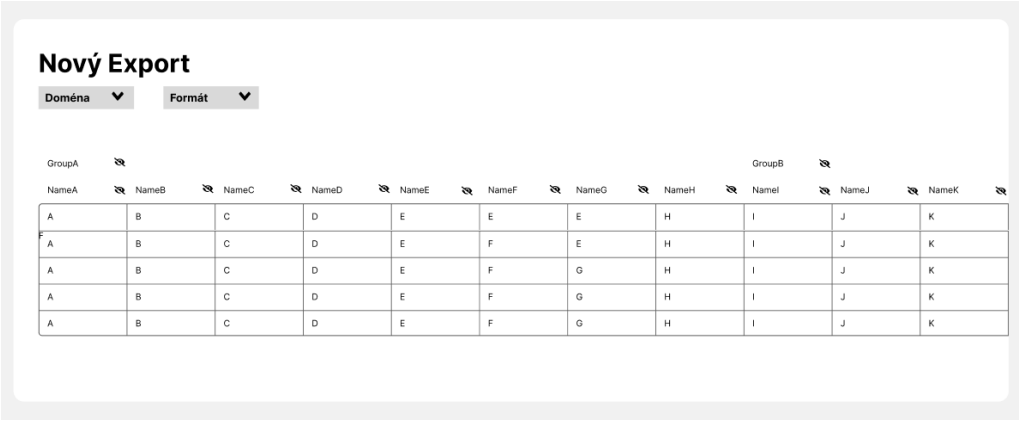
2.2.1 Figma

Pro návrh designu nového UI jsem si zvolil nástroj Figma. Figma je nástroj na vytváření komplexních a reaktivních designů UI přímo ve webovém prohlížeči. Nabízí mnohem lépe vypadající návrhy než například Balsamic i přes to, že je také Open Source. Zvolil jsem jí především z důvodu doporučení jak od vedoucího projektu Ing. Jiřího Hunky, tak od Ing. Marka Erbena. Hlavní výhodou figmy je možnost vytvářet znovu použitelné komponenty, které vedou k rychlému vytvoření návrhů pro robustní webové aplikace. Velikou výhodou je také možnost týmové spolupráce na návrhu a také on-line prezentování sledováním obrazu účastníka [11].

2.2.2 Export

Mým prvním návrhem byl proof of koncept konfigurace exportu dat ze skladového systému. Hlavním problémem bylo dát uživateli možnost upravit každý sloupec tabulky tak, aby si mohl zvolit, zda ho chce exportovat či nikoliv. Vytvořil jsem tedy dva návrhy. Oba návrhy sdílí zobrazení náhledu tabulky ve stylu v jakém bude exportována. Data v tabulce jsou falešná a slouží pro lepší představu.

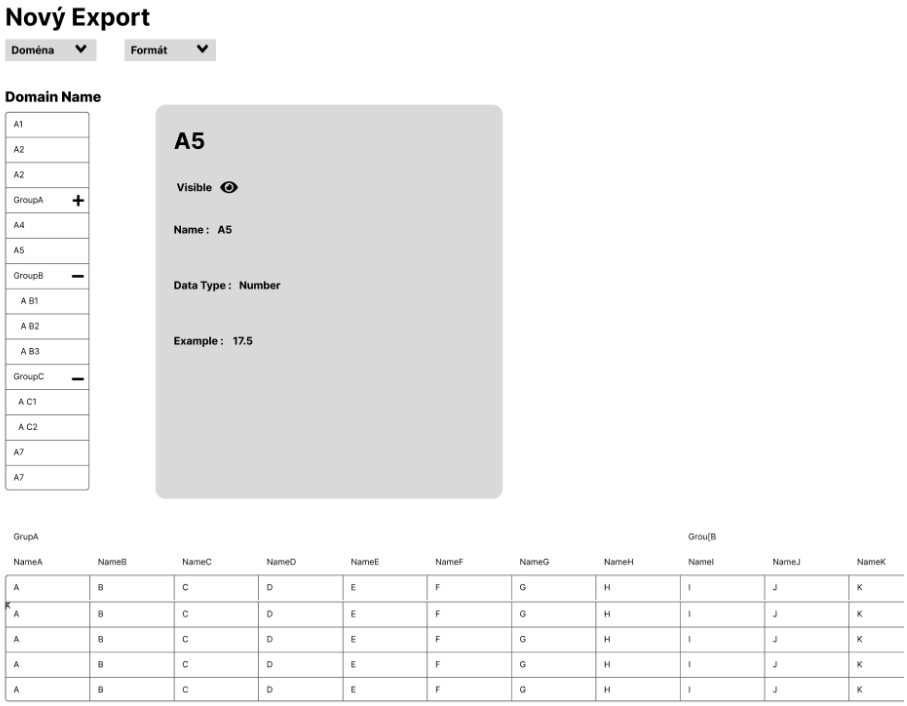
V prvním návrhu je problém řešen tím, že uživatel přímo ovládá tabulku pomocí tlačítek u jmen atributů. Nevýhodou je, že pokud uživatel nějaký atribut schová tak ho nemůžeme úplně odstranit, ale musíme ho v tabulce nějak zachovat, aby byla možnost atribut vrátit zpět. Je to jednoduché řešení, tabulka s odebranými atributy pak ale nezobrazuje příliš přesně jak bude vypadat vyexportovaná verze, ale pouze to přibližuje. Tento návrh můžete vidět zde 2.4



Obrázek 2.4: První návrh konfigurace exportu

Druhý návrh řeší problém poněkud robustněji. Dáme uživateli nástroje na manipulaci s tabulkou mimo a poté již zobrazujeme přesný náhled tabulky jak bude vypadat po exportu. Jak jsem již zmínil toto řešení zabírá mnohem více místa, ale dává nám do budoucna prostor pokud bychom chtěli měnit více věcí, než pouze viditelnost atributu. Návrh tohoto řešení můžete vidět zde 2.5

Po zvážení obou možností byl zvolen první návrh, především kvůli jeho jednoduchosti na ovládání. Jelikož se jedná pouze o zvolení viditelnosti sloupců tabulky, mohlo by druhé řešení působit více složitě a uživatele zbytečně zmást. První řešení sice nenabízí přesný náhled exportované tabulky, ale jeho jednoduchost na ovládání je pro tento návrh důležitější.



Obrázek 2.5: Druhý návrh konfigurace exportu

2.2.3 Import

Pro import jsem měl více konkrétní představu pro návrh řešení a zobrazení všech možností uživateli. Je potřeba uživateli zobrazit jeho nahraný soubor, na který se při průchodu BE snažíme namapovat atributy zvolené domény. Po zobrazení musíme uživateli dát možnost namapovat zbytek tabulky co nejjednodušeji a zároveň přehledně.

Důležité je, aby bylo jasně viditelné jaké, požadavky musí uživatel splnit (Především namapovaní povinných atributů). Také musí být vidět skupiny jednotlivých atributů, aby měl uživatel přehled při rozhodování, které sloupce budou použity jako primární klíče a budou rozhodovat o unikátnosti při vkladu do databáze.

I přes velké množství akcí, které musí být uživateli k dispozici, jsme se rozhodli opět pro minimální design náhledu tabulky, která bude na konci mapování importována, tentokrát zobrazující reálná data nahraná uživatelem. K ovládání budou použity hlavičky jednotlivých sloupců a po rozbalení této hlavičky bude zobrazen list atributů možných k namapování. Atributy budou seřazeny podle skupin v zanořeném listu a budou označeny červeně pokud jsou povinné a nenamapované, šedě pokud jsou již nama-

povány a zeleně pokud se jedná o atribut namapován na námi právě rozbalený sloupec. U zvoleného atributu sloupce pak bude ikona klíče, která bude rozhodovat a znázorňovat jedná-li se o primární klíč skupiny. Obrázek návrhu najdete zde 2.6

Obrázek 2.6: Návrh 1. iterace Import

2.3 Realizace

Po vytvoření návrhů obou hlavních částí nových funkcionalit přešel vývoj do stavu realizace. Jelikož se jedná o přidání funkcionality do ji běžícího projektu, bylo potřeba se seznámit s jeho fungováním a stavem. Dokumentace projektu Swordfish pokrývá bohužel jen velice základní informace a tak bylo potřeba dívat se přímo do kódu a seznamovat se s projektem tímto způsobem. Projekt je naštěstí udržován velice konzistentní a tak byl i tento způsob poměrně efektivní.

2.3.1 Export

Jako první krok jsem zvolil tvorbu základního pohledu pro export, který se skládal z tabulky zobrazující již vytvořené exporty. Nejprve však bylo potřeba vytvořit si záložku v projektu pro exporty. Tato akce se skládala z několika kroků vedoucích k přidání exportů mezi ostatní části projektu. Nejprve bylo potřeba přidat záložku pro exporty do komponenty hlavního menu aplikace, poté jsem vytvořil soubor pro **router** aplikace s adresou hlavní stránky exportů. Tyto dva kroky vedly k funkční prázdné stránce pro exporty, ze které jsem mohl dále vycházet.

Jako další krok jsem zvolil vytvoření tabulky pro zobrazení již vykonaných exportů a základních informací o nich. Tabulky tohoto typu jsou v

projektu velice časté a je pro ně vytvořená upravená komponenta Vuetify **x-data-table** vytvořená Ing. Oldřichem Malcem. Použití této tabulky je velice jednoduché a tak jsem poměrně rychle zvládl komponentu zapojit do svého celku.

Druhý krok se projevil mnohem složitější, jelikož je komponenta konfigurace exportu zcela dynamická, s čím se zbytek aplikace moc nepotýká, bylo potřeba komplexní návrh zrealizovat od základů. Konfigurace exportu se skládá ze tří informací, domény, typu souboru do kterého budu exportovat a samotné konfigurace sloupců. Na začátek jsem vytvořil *select menu* pomocí Vuetify komponenty **v-select** pro výběr domény, podle které jsem mohl zobrazit tabulku. Díky responzivitě Vue bylo s využitím *computed promněnné* jednoduché ihned překreslit tabulku na základě zvolené domény v *select menu*.

Samotná tabulka byla vytvořena ze základní Vuetify komponenty tabulky **v-simple-table** bez žádných přidaných funkcionalit, tu jsem upravil aby hlavička obsahovala tlačítka ovládající viditelnost sloupce. Hlavička tabulky musela odpovídat šabloně zvolené domény, tyto šablony jsem pomocí funkce *created* (Funkce Vue, volaná při vytvoření instance komponenty) získal z API a pak mezi nimi jen přecházel při změně domény v *select menu*. Data tabulky jsem generoval pomocí velice triviálních funkcí využívajících náhodnost.

Po dokončení této hlavní části exportu jsme se na schůzce s kolegou Ing. Markem Erbenem dohodli, že můj návrh nebyl ideální a výsledek není dostatečně přehledný pro použití. Po krátké diskuzi jsme návrh pozměnili tak, aby hlavička tabulky zobrazovala všechny vrstvy zanořeného seznamu atributů.

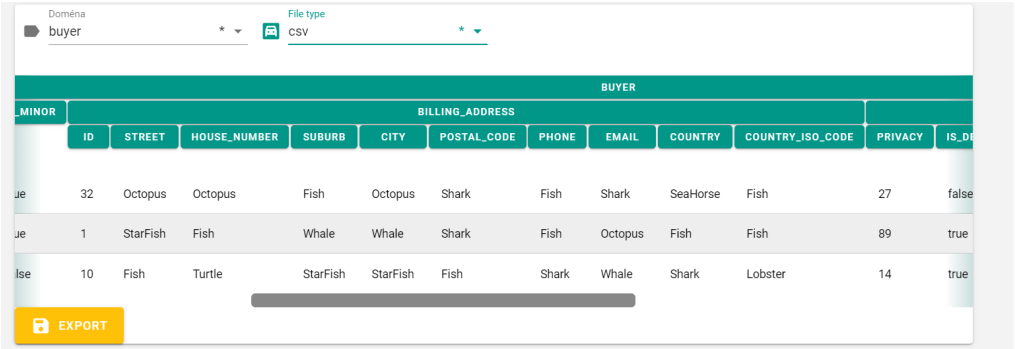
Při implementaci tohoto návrhu jsem řešil především problém dynamické zanořené hlavičky tabulky, netřeba zmiňovat, že Vuetify na tento problém připravené řešení nemá. Jako první jsem se pokusil vytvořit rekurzivní komponentu, která by zobrazovala jednotlivé řádky. Toto řešení se sice dostalo poměrně blízko, ale selhalo díky faktu, že HTML vykresluje tabulky po řádcích, tedy pokud byla ve skupině další skupina, nebyla jednoduchá možnost vykreslit jí celou. Řešení jsem tedy zavrhl a bohužel se musel smířit s primitivním a pracným řešením napočítání si tvaru tabulky a kontroly všech úprav ručně.

Při skrytí skalárního atributu bylo zapotřebí pouze nezobrazovat jeho hodnoty v tabulce dat, ale při skrytí celé skupiny bylo potřeba vynechat celé sloupce a změnit šířku tabulky. Manuální kontrola tvaru tabulky zahrnovala počítání odsazení každého řádku tabulky, pro každý řádek bylo tedy potřeba napočítat kolik skalárních atributů obsahuje a podle toho odsadit následující řádek. Toto číslo se také v průběhu konfigurace může měnit,

pokud uživatel schová celou skupinu. Při schování celé skupiny se totiž její skaláry přestanou zobrazovat a do odsazení dalšího řádku se už nepočítají.

Dále bylo potřeba manuálně kontrolovat jaký sloupce se uživatel chystá schovat. Jelikož hlavička tabulky a data v ní nejsou nijak provázána, je při kliknutí na konkrétní hlavičku potřeba zjistit její pozici a využít ji na schování patřičného sloupce dat.

Toto řešení bylo sice velice pracné na realizaci, bylo však funkční a poskytovalo vysokou kontrolu nad všemi aspekty zobrazované tabulky a bylo tedy pi realizaci využito. Konečný stav realizovaného řešení konfigurace exportu můžete vidět zde 2.7



Obrázek 2.7: 1. iterace Export

2.3.2 Import

Implementace Importu začala stejně jako u exportu, a to vytvořením funkční záložky pro importy společně s tabulkou pro zobrazení již vytvořených importů. Tohoto bylo znovu docíleno pomocí již existující komponenty **x-data-table**.

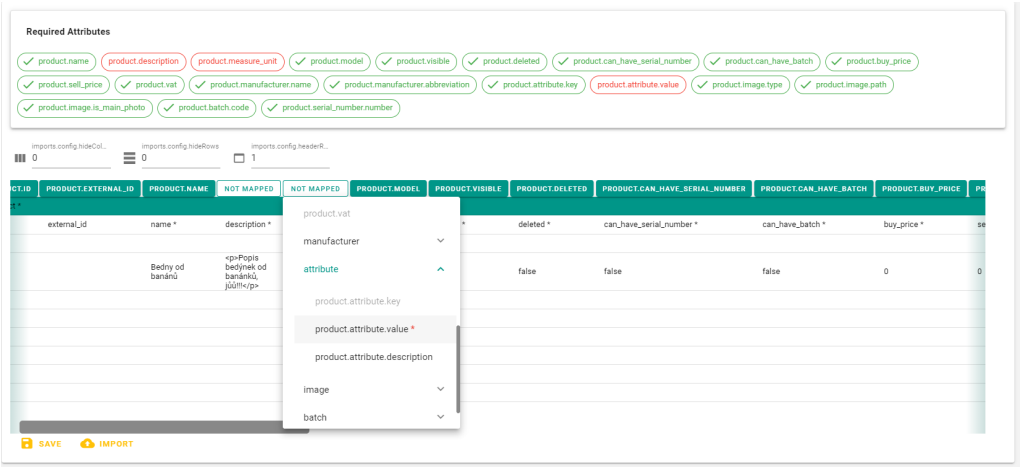
Další krok se bohužel opět opakoval a narazil jsem na podobné problémy jako u exportu. Jako první bylo opět potřeba zobrazit uživateli všechny skupiny atributů a jejich skaláry, tentokrát ve formě vertikálního menu, tak aby poskytovalo možnost vybrat konkrétní skalární atribut. Na tento problém jsem naštěstí díky Vuetify komponentě v-menu a v-list mohl využít řešení s rekurzivní komponentou skupiny. Toto řešení fungovalo podle představ, jediný problém nastal při využití *emits* z nižších komponent bylo potřeba propsat *emit* až do hlavní rodičovské komponenty.

Další zajímavý problém bylo zobrazení uživatelovi nahrané tabulky v aplikaci. Na tento problém jsem využil knihovnu SheetJS, která pracuje s XLSX soubory a soubory podobného typu na jejich zpracování, vytváření

a zobrazování. Rozhodl jsem se knihovnu požít pouze na získání dat ze souboru, jelikož jsem chtěl mít více kontroly na tím, jak bude tabulka zobrazena. Po získání dat jsem byl, opět pomocí Vuetify komponenty **simple-table**, schopný vytvořit náhledovou tabulku, obsahující výše zmíněné rekurzivní menu jako hlavičku každého sloupce.

Díky knihovně sheetJs jsem měl sice k dispozici zpracovaná data uživatelem nahraného souboru, neměl jsem však možnost kontrolovat jestli soubor neobsahoval i nějaké zápisy mimo požadovanou tabulku k importu (například popisky před nebo nad tabulkou), které bychom mohli chybně interpretovat jako součást tabulky. Bylo tedy potřeba dát uživateli možnost nám sdělit kde přesně v jeho souboru se tabulka k importu nachází. Toho jsem dosáhl pomocí přidání tří přepínačů pro první řádek tabulky, počet řádků v hlavičce a nakonec první sloupec tabulky.

Poslední důležitá část byla zobrazení povinných atributů, které musí uživatel namapovat. Pro zobrazení jsem použil box, který v sobě obsahuje pole povinných atributů aktuální domény. Tyto atributy jsou zobrazeny červeně a pokud je uživatel namapuje změní barvu na zelenou, bylo tedy jen potřeba hlídat aktuální stav namapovaných povinných atributů pomocí *watcher* funkcí. Konečný stav realizované konfigurace importu můžete vidět zde 2.8



Obrázek 2.8: 1. iterace Import

2.3.3 Shrnutí problémů

V této části textu shrnu a krátce popíši hlavní problémy při realizaci první iterace a přiřadím jim obtížnost od 1 do 5, kde 5 značí problém nejsložitější a 1 problém triviální.

- Zasazení nové funkcionality do projektu Swordfish
  Náročnost: 1
- Tabulka pro zobrazování dat
  Náročnost: 1
- Vykreslení hlaviček tabulky pro konfiguraci exportu
  Náročnost: 5
- Udržení reaktivity tabulky pro konfiguraci exportu
  Náročnost: 4
- Načtení uživatelského souboru pro import
  Náročnost: 4
- Zobrazení uživatelského souboru
  Náročnost: 3
- Mapování uživatelského souboru na doménu
  Náročnost: 4
- Validace uživatelských vstupů
  Náročnost: 2

2.4 Testování

Testování první verze nového uživatelského rozhraní mělo proběhnout pomocí uživatelských testů, při kterých se tetuje funkčnost celku, ale především jak je intuitivní na ovládání a pokud je pro uživatele dostatečně přehledný pro práci. Do tohoto stádia testování jsem se však s první verzí nedostal, kvůli vážným nedostatkům návrhu, na které poukázal vedoucí práce Ing. Jiří Hunka při mojí prezentaci řešení. Námitky byli vedeny především na absenci informací o datech ovlivněných provedením importu „Uživatel takhle importuje naslepo“. Další námitky byli vedeny nad způsobem uchování jednotlivých provedených importů a exportů a vynechání možnosti pro uživatele, aby mohl znovu použít již existující konfiguraci.

Bylo tedy rozhodnuto, že tato verze není dostatečně bezpečná k nasazení a bude potřeba přepracovat řešení a uživatelsky otestovat až iteraci další.

2.5 Zhodnocení

V této sekci se zaměřím na hodnocení jednotlivých částí 1. iterace, jak z pohledu jejich průběhu tak i finálního výsledku. Popíši, jaké měla fáze silné

stránky a jaké praktiky si přenesu do dalších iterací, stejně tak jako jaké byly nedostatky, ze kterých se v dalších iterací ponaučím.

2.5.1 Sběr požadavků

Sběr požadavků byla v první iteraci část vývoje, jež byla nepochybně podceněna. Schůzka s kolegou Ing. Markem Erbenem mi sice předala základní požadavky očekávané od nové funkcionality z pohledu funkčnosti, nedozvěděl jsem se však požadavky kladeny z pohledu UI a UX. Právě tyto chybějící požadavky pak vedly k nepoužitelnému finálnímu celku. Hlavní chybou tedy byla chybějící schůzka přímo s vedoucím projektu Ing. Jiřím Hunkou, který jak jsem se později dozvěděl, měl velice konkrétní požadavky právě na UI a UX.

Při další iteraci tak začnu podrobnou schůzkou s vedoucím projektu, kde zjistím jeho požadavky z obou pohledů, jak funkčnosti, tak UI a UX a tím snad zajistím použitelnost další iterace.

2.5.2 Návrh

Návrh byl v první iteraci nepochybně nejslabší stránkou vývoje a jeho nedostatky měli za důsledek zpomalení průběhu implementace, díky spoustě práce navíc vykonané jako kompenzaci. Prvotní návrh fungoval, jak jsem již zmínil, spíše jako proof of concept než návrh, kterým bych se mohl řídit při vývoji. Toto vedlo k velice častému implementování částí, které se později několikrát měnili a to velice zpomalilo celý proces. Za jedinou dobrou stránku průběhu návrhu bych označil zvolení technologie Figma, která se ukázala být na práci ideální.

Při další iteraci budu tedy nadále používat nástroj Figma, ale před implementací se soustředím na komplexní návrh celku importů a exportů, který budu v průběhu vývoje aktualizovat pokud by se objevili nějaké důvody ke změnám.

2.5.3 Realizace

Realizace, i přes nějaké nedostatky, byla během 1. iterace část vývoje s již jsem byl nejvíce spokojen. Realizace byla nepochybě zdržována prozkoumáváním slepých uliček, kterým by správně předcházel propracovanější návrh. Tato nadbytečná práce mi však sloužila jako prostor pro rychlejší zlepšení dovedností s technologiemi Vue a Vuetify.

Jako nedostatek bych označil nekonzistentní kód, jež byl důsledkem opomíjeného refaktoringu. Kód psaný ke konci iterace byl psaný jiným

způsobem než ten psaný na začátku, díky zkušenostem nasbíraným během vývoje.

V Příští verzi se při implementaci soustředím na refaktoring kódu stejně tak jako na psaní kódu samotné, abych zajistil konzistenci a dobrou čitelnost pro ostatní vývojáře.

Kapitola 3

2. Verze

3.1 Sběr požadavků

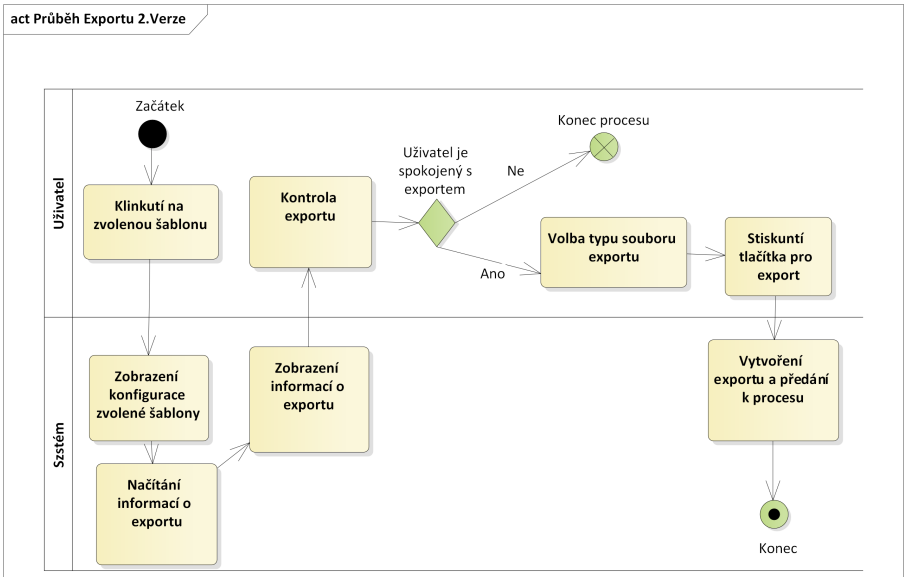
Sběr požadavků byl v druhé iteraci proveden s ponaučením z iterace první, důsledněji a přímo od vedoucího projektu Ing. Jiřího Hunky. Požadavky byly sbírány během schůzky s vedoucím, kde jsme společně hodnotili 1.iteraci a její nedostatky ze stran funkčnosti, ale především se strany UI a UX. Zde se projevila výhoda iteračního postupu a její možnosti pro zlepšení iterací na základě nedostatků předchozích, jak už u lepšího celkového výsledku tak i průběhů samotných. Během schůzky byly tedy vytvořeny následující požadavky ke splnění během 2. iterace.

3.1.1 Export

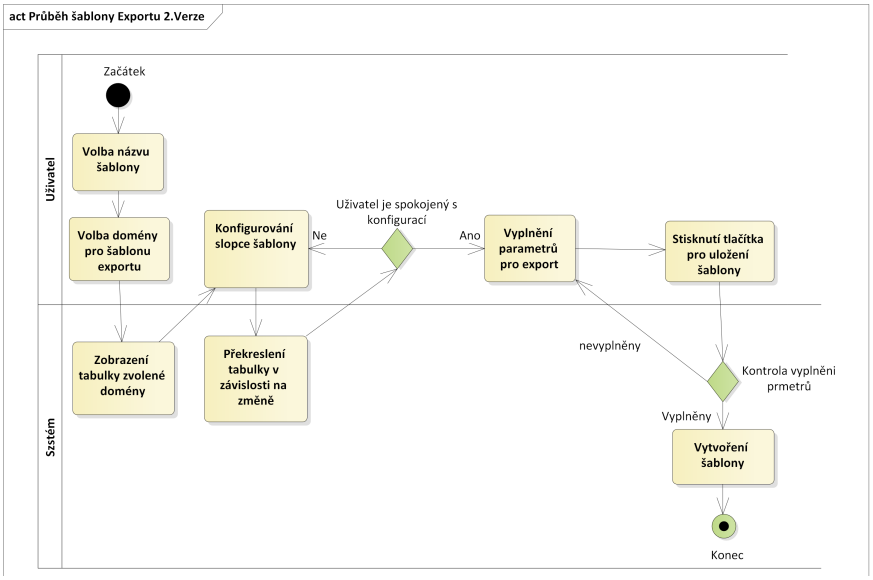
Změny v sekci exportu by měli zahrnovat rozdělení exportů na šablony exportu a samotné exporty. Bude tedy existovat samotná jednotka šablony, jež si bude pouze uchovávat konfiguraci vytvořenou uživatelem. Pomocí těchto šablon se pak budou vytvářet jednotlivé exporty. Dále je potřeba aby uživatel před vykonáním exportu věděl, kolik dat se bude exportovat a dokázal tak rozeznat, jestli někde v procesu neudělal chybu. 3.13.2

3.1.2 Import

Změny v části importu byli velice podobné exportu, a to vytvořit prvek šablony pro import, která bude uchovávat namapování nastavené uživatelem. Importy se potom budou zobrazovat přehledně pomocí těchto šablon. Stejně jako u exportu je pak potřeba uživateli sdělit informace o následcích importu ještě před jeho provedením.3.33.4



Obrázek 3.1: Activity Diagram: Export 2.Verze

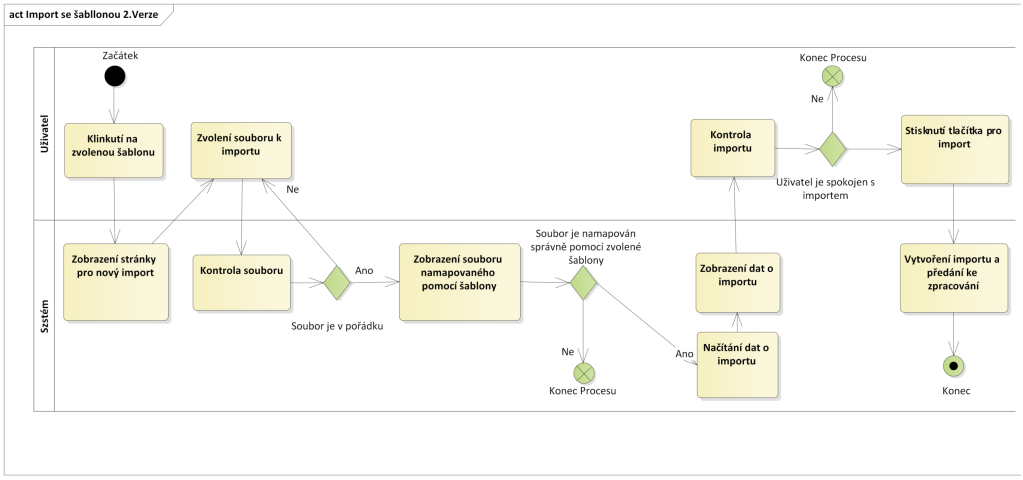


Obrázek 3.2: Activity Diagram: Vytvoření šablony 2.Verze

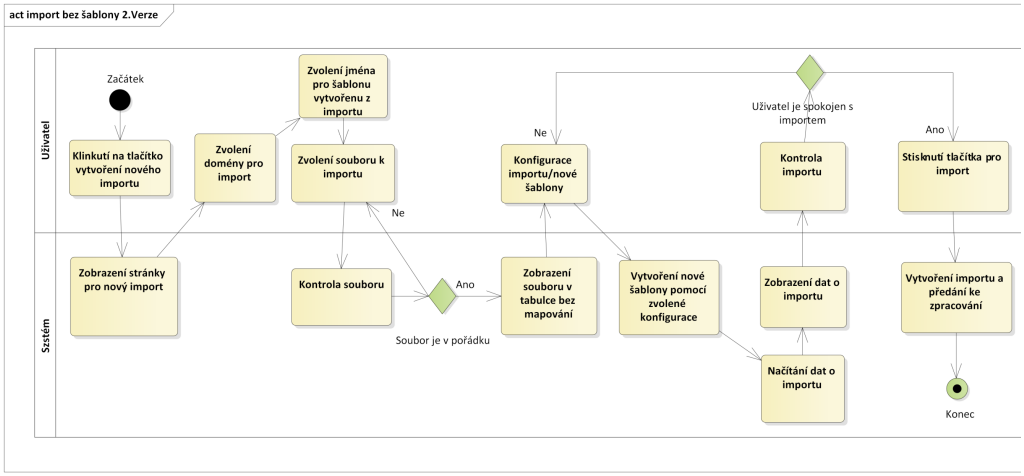
3.1.3 Funkční požadavky

FP1: Vytvoření nové šablony

Jako první krok musí mít uživatel možnost vytvořit nové šablony jak pro import, tak export.



Obrázek 3.3: Activity Diagram: Import se šablonou 2.Verze



Obrázek 3.4: Activity Diagram: Import bez šablony 2.Verze

Priorita: Must have
Modul: Import/Export

FP2: Zobrazení Šablon

Stejně jako byl v první verzi požadavek na zobrazení všech importů/exportů, je potřeba stejně zobrazit všechny šablony importů a exportů společně se základními informacemi o nich.

Priorita: Must have
Modul: Import/Export

FP3: Mazání vytvořených šablon

Uživatel by měl mít možnost smazat vytvořené šablony importů a exportů společně se všemi instancemi vytvořenými pomocí těchto šablon.

Priorita: Should have
Modul: Import/Export

FP4: Úprava vytvořených šablon

Uživatel musí mít možnost upravovat již vytvořené šablony, ať už se jedná pouze o jméno šablony, nebo přímo o její nastavení.

Priorita: Must have
Modul: Import/Export

FP5: Import/Export pomocí šablon

Uživatel musí mít nově možnost vytvořit nový import nebo export pomocí již vytvořených šablon. Díky tomu nebude muset pokaždé nastavovat parametry importu nebo exportu.

Priorita: Must have
Modul: Import/Export

FP6: Zobrazení statistiky

Hlavním nedostatkem předchozí verze byla absence informací o importu a exporu před jejich provedením. Je tedy nutno uživateli zobrazit informace o importu/exportu, který se chystá provést.

Priorita: Must have
Modul: Import/Export

3.2 Návrh

Po nedostatcích z 1. iterace bude vytvořen nový propracovaný návrh opět v technologii Figma. Tento nově vytvořený návrh tentokrát pokryje každou stránku a část importů a exportů tak, aby se na něj při implementaci mohl kdykoliv obrátit a bude prezentován vedoucímu projektu před tím, než se přesunu k jeho realizaci.

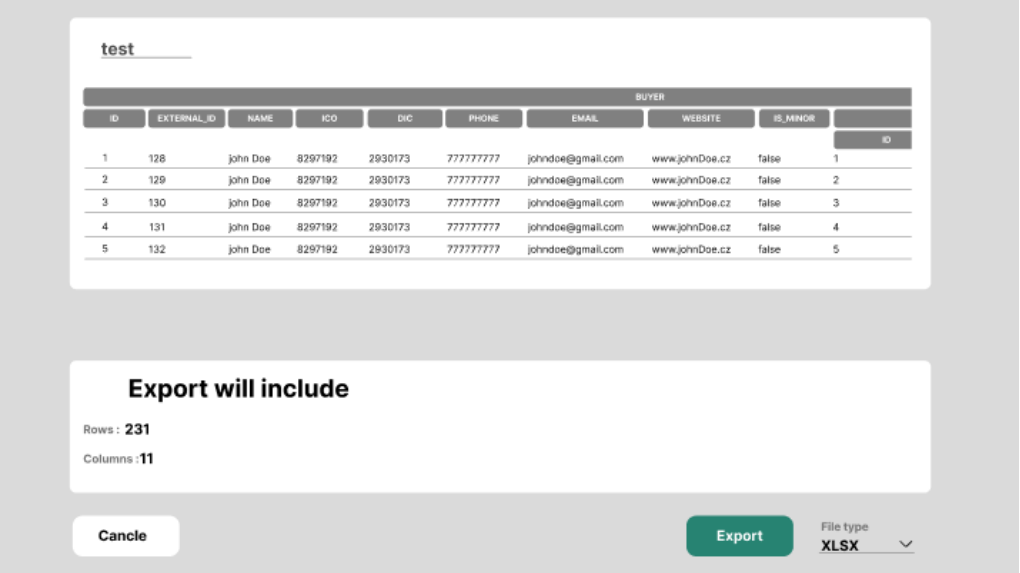
3.2.1 Návrh Exportů

Jelikož se exporty nově dělí podle jejich šablon, je potřeba změnit způsob zobrazení již vytvořených exportů tak, aby bylo jasné pod jakou šablonu patří. Pro tento účel byla zvolená rozevírací tabulka šablon, kde se po rozevření šablony zobrazí všechny její exporty. Tento návrh nám také dává možnost zobrazit možné akce k vykonání jak u šablon, tak i u samotných exportů jak můžete vidět zde 3.5

Obrázek 3.5: Vnořená tabulka exportů

Jako další bylo zapotřebí vytvořit návrh pro vytvoření šablony samotné, tato akce je vlastně téměř totožná s předchozí akcí pro vytvoření exportu s rozdílem toho, že se finální konfigurace uloží jako šablona a neprovádí se export.

Další změna přichází v návrhu akce exportování, kde muselo být nově zakomponováno zobrazení informací o exportovaných datech ještě před provedením exportu. Byla tedy vytvořena kompletně nová stránka, která bude zobrazovat informace o zvolené šabloně a exportu, který je právě vytvářen. Výběr typu souboru pro export byl přesunut vedle tlačítka exportu, kde ji uživatel spíše nepřehlédne. Obrázek návrhu můžete vidět zde 3.6



Obrázek 3.6: Návrh exportu 2. iterace

Zbytek vytvořeného návrhu, který zde není zobrazen, pouze přesně zaznamenává aktuální stav řešení v aplikaci a neobsahuje žádně změny.

3.2.2 Návrh Importů

Návrh pro importy se opět hodně podobal exportům. Bylo tedy zapotřebí vytvořit návrh pro tabulku šablon importů obsahující i tabulky jejich importů. Tabulka má stejné rozložení jako u exportů a obsahuje opět možnost vykonávat akce nad šablonami a importy samotnými, což můžete vidět na obrázku zde 3.7

U importů bylo potřeba důkladněji navrhnout část průběhu importování samotného, jedná se totiž o mnohem komplexnější akci než je export. Při importu je problémem provázanost šablony s uživatelským souborem. Nastavení šablony je totiž přímo provázáno se souborem, použitým při jejím nastavení. Z toho vyplývá, že není možné vytvořit šablonu bez uživatelského souboru.

Samozřejmě se nabízela možnost nechat uživatele při tvorbě šablony nahrát soubor, sloužící pouze jako nástroj ke konfigurování. Toto řešení je však neintuitivní a přidává zbytečné kroky a tak bylo zavrženo.

Po několika nápadech bylo schváleno řešení, jež upřednostňuje jednoduchost na úkor uživatelovi možnosti vytvořit šablonu manuálně. Tvoření šablony bude řešeno jako vedlejší produkt tvoření importu, pokud se tedy uživatel rozhodne vytvořit import a nezvolí si k tomu žádnou z již vy-

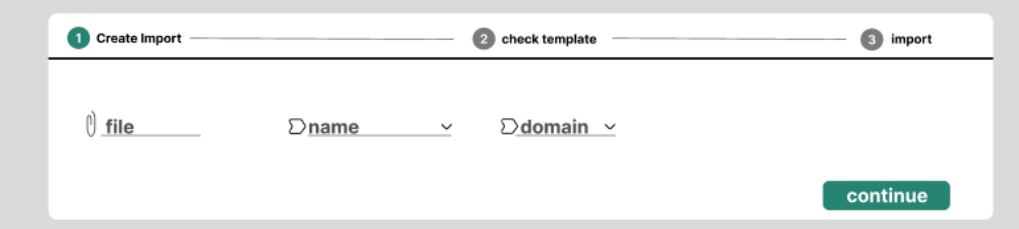
Obrázek 3.7: Vnořená tabulka importů

tvořených šablon, bude konfigurace právě tohoto importu použita k vytvoření nové šablony. Toto řešení navíc dává větší důraz na importování dat, spíše než na tvoření šablon, většina uživatelů bude totiž spíše chtít primárně importovat data a nebude se chtít zdržovat s vytvářením šablon.

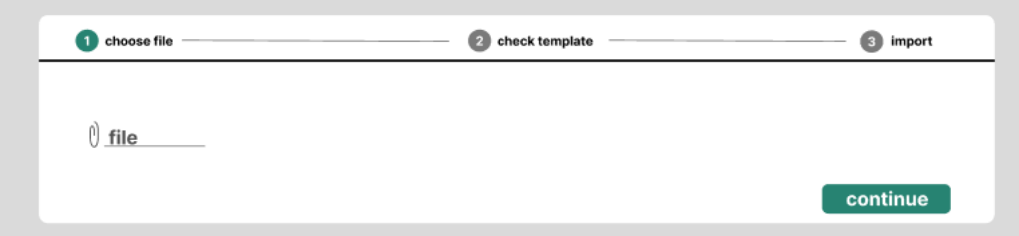
Po vyřešení šablon v novém návrhu bylo zapotřebí navrhnout rozhraní, ve kterém se bude samotný import odehrávat. Jelikož se akce importu nyní odehrávala ve třech krocích, nahrání souboru, namapování tabulky a zobrazení dat o importu, rozhodl jsem se pro znázornění tohoto faktu a přidal *stepper* nad jednotlivé kroky, jež jasně zobrazoval uživateli v jaké fázi importování se právě nachází a co je od něj očekáváno.

Stepper je, jak jsem již zmínil rozdělen na tři kroky, prvním z nichž je nahrání souboru pro import. Tento krok jsem se později rozhodl rozdělit na dvě možnosti a spojit tak dva případy užití v jeden průchod skrz rozhraní. To by mělo napomoci k zjednodušení rozhraní a snadnější orientaci uživatele díky menšímu množství unikátních stránek. Uživatel totiž bude tyto tři kroky importu procházet v obou případech jak importování pomocí šablony, tak vytváření importu společně se šablonou. V případě použití již existující šablony bude uživatel v prvním kroku pouze nahrávat soubor k importu. Pokud se však rozhodne pro import bez šablony, bude od něho v prvním kroku také požadováno, aby zadal jméno pro šablonu, jež bude vytvořena a také doménu, kterou se chystá importovat. Zobrazení obou návehů najdete zde 3.8a

Druhý krok importu se bude opět, díky propojení dvou případů užití, dělit na dvě možnosti. Tyto možnosti jsou si velice podobné jak z pohledu návrhu, tak implementace. Při importování bez předem vybrané šablony si uživatel v druhém kroku bude muset namapovat svou tabulku na zvole-



(a) import bez šablony



(b) Import se šablonou

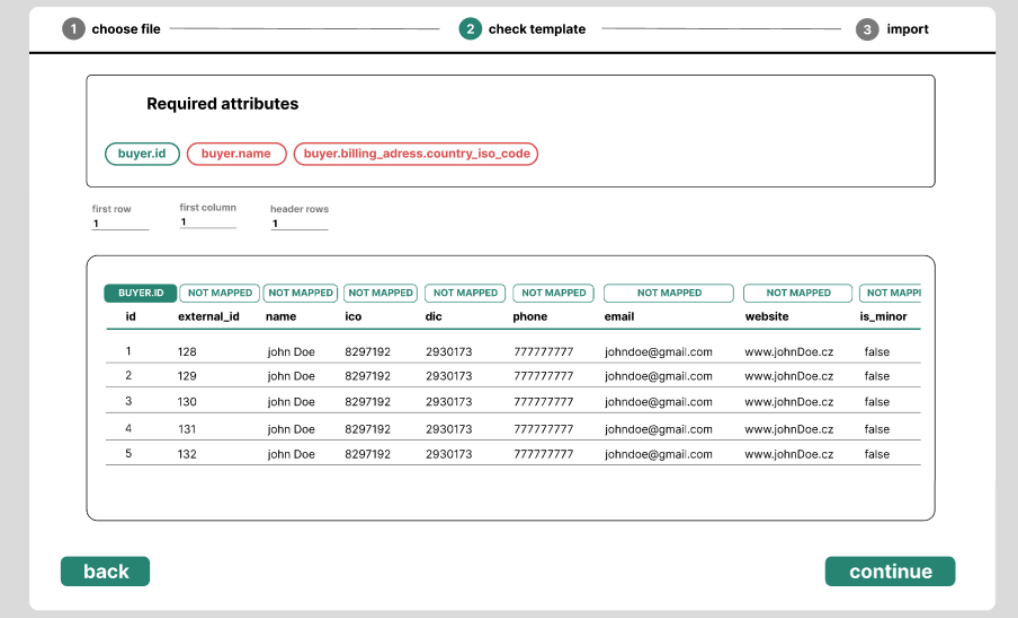
Obrázek 3.8: Vyplnění informací pro import

nou doménu jako v 1.iteraci importu. Tedy opět mu zobrazím náhled jeho souboru v tabulce, kde je nad každým sloupcem možnost zvolit atribut domény k namapování. Také mu budou vypsány povinné atributy k namapování a parametry importu k vyplnění (bude-li je doména obsahovat). Po splnění všech požadavků bude tato konfigurace uložena jako nová šablona pod názvem zvoleným v prvním kroku.

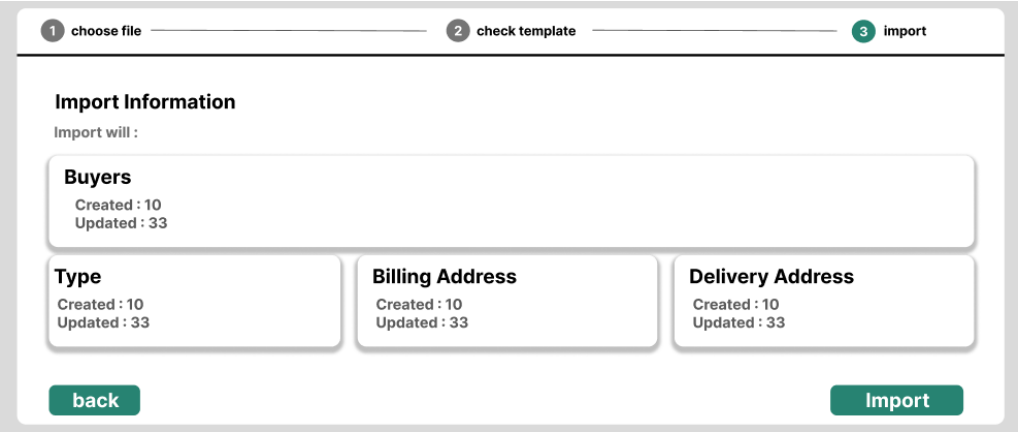
V případě toho, že se nacházíme v druhém kroku importu a uživatel importuje pomocí již existující šablony, bude mu v druhém kroku zobrazena zvolená konfigurace pouze pro čtení, tedy bez možnosti úpravy. Vybraná šablona však bude použita na právě nahraný soubor, mělo by tedy být na první pohled jasné pokud uživatel nehodou zvolil špatnou šablonu a nebo soubor k importu. Obrázek návrhu najdete zde 3.9

Posledním krokem je pak už jen samotný import dat. V tomto kroku budou uživateli zobrazeny informace ohledně důsledků připraveného importu. Ty by měli uživateli dát možnost se ujistit, že chce opravdu připravený import provést. Toto je způsob, jak ochránit systém před velkým množstvím požadavků na navrácení skladů do stavů před chybnými importy. Uživateli bude k dispozici náhled na množství přidaných záznamů stejně tak jako počet záznamů upravených, jak je vidět na obrázku zde 3.10

Návrh také zahrnuje možnost uživatele odejít od již rozdělaného importu. Toto je zajištěno přidáním rozdělaného importu mezi ostatní importy šablony i před jeho dokončení. Uživatel se tak může kdykoliv k importu vrátit a dokončit ho.



Obrázek 3.9: Konfigurace importu



Obrázek 3.10: Statistiky Provedení importu

3.3 Implementace

Implementace druhé iterace byla obsahově méně rozsáhlá něž iterace první především co se nových požadavků týče. Jednalo se hlavně o zakomponování šablon z návrhů do projektu. Větší důraz jsem kladl na refaktoring kódu a zachování konzistence jak jsem již zmínil v hodnocení 1. iterace.

Při refaktoringu jsem po sobě opravoval větší část kódu první iterace. Jednalo se o správné využití *enum* a *service* souborů, které nebylo v první

iteraci ideální. Dále bylo potřeba přepsat části kódu, jež byli při code review trefně označeny jako „céčko v Javascriptu“, tak aby více využíval možností Javascriptu, hlavně jeho funkcí nad poli. Jako poslední větší část refaktoringu kódu pak bylo dodržování stylu kódu se zbytkem projektu.

Další problém, který byl v první iteraci zanedbán byli překlady. Překlady v první iteraci byli přítomny a pokrývali většinu textu v aplikaci, byli však opomenuty překlady proměnných přicházejících z API, které byli v anglickém jazyce. V druhé iteraci jsem se tedy zaměřil na překlad těchto proměnných, jež se v některých místech projevilo jako poměrně obtížné. Například při překladech skupin statistik 3.10 je třeba využít kódy, jež jsou přiděleny skupinám na BE, například skupina Billing Addres má kód *buyers.billing_address*. Jako nejrozumnější řešení se ukázalo vytvořit soubor, který propojí poslední slovo kódu (v tomto případě billing_addres) s jeho překlady. Bylo však potřeba pro každé unikátní slovo vytvořit nový záznam a nalézt k němu patřičný překlad v překladových souborech.

3.3.1 Export

Implementace změn pro exporty byla poměrně přímočará. Jako první bylo potřeba vytvořit rozevírací tabulku šablon a jejich exportů. Tato funkcionalita tabulky byla naštěstí již zahrnuta v dříve zmíněné komponentě **x-datatable** existující v projektu. Stačilo tedy tuto komponentu správně využít a získal jsem požadovaný výsledek.

Jako další jsem z celku konfigurace exportu vytvořil komponentu pro konfiguraci šablony s cílem pro jednoduché znovupoužití. Této komponentě jsem pak přidal pomocí *props* možnost zobrazit konfiguraci již vytvořené šablony. Komponenta uměla konfigurovat jednotlivé části šablony, stejně tak jako vyplnit tyto informace z již existující šablony a tím ji zobrazit uživateli. Další přidaná funkcionalita opět s použitím *props* bylo zobrazení komponenty pouze pro čtení. S těmito funkcionalitami jsem byl schopen použít tuto komponentu jak pro konfiguraci nové šablony, tak pro zobrazení již existující.

Převedením konfigurace šablony do samostatné komponenty se však vytvořil nový problém, a to jak nejlépe vyřešit uložení konfigurace šablony. Informace o konfiguraci se totiž nachází uvnitř komponenty pro konfigurace (potomek), tlačítko na uložení je však v komponentě rodičovské (rodič). Je tedy potřeba nějakým způsobem propojit tyto komponenty, aby společně uložili data pomocí API. Tento problém by se dal vyřešit třemi způsoby.

První by byl přesunout tlačítko na uložení do komponenty pro konfiguraci (potomka), toto řešení by bylo nejjednodušší z důvodu eliminace potřeby pro komunikaci mezi dvěma komponentami. Řešením by se však

přidalo další nové tlačítko pro uložení, na které by musel uživatel kliknout před pokračováním a to bylo v rozporu s návrhem.

Další možnost by bylo použít *emit* a vyslat informace o konfiguraci z potomka do rodiče, kde se nachází tlačítko na uložení. Toto řešení by sice sedělo podle návrhu, nedávalo mi však smysl ukládat konfiguraci šablony mimo její komponentu.

rozhodl jsem se tedy pro možnost třetí, použití kombinace *props* a *watcheru*. Pomocí těchto prvků jsem mohl v rodičovské komponentě změnit hodnotu předanou pomocí *props*, ta se tím změnila i v komponentě potomka. V potomkovi pak stačilo pomocí *watchru* při změně této proměnné aktuální nastavení uložit. Řešení je ideální z důvodu jednoduchého znovupoužití aniž by kladlo požadavky na rodičovskou komponentu.

Novou problematikou z pohledu UX, jež bylo potřeba vyřešit se stalo automatické rozbalování šablon, majících export v procesu. Tedy pokud si uživatel vytvoří nový export, *router* ho automaticky přesune zpět na stránku se šablonami exportů a ztratí tak přehled o svém exportu. Bylo tedy potřeba, aby se takovýto nově vytvořený export nalezl a jeho šablona se rozbalila, čímž zajistíme lepší přehlednost systému. Tento problém byl vyřešen průchodem všemi šablonami a hledáním exportů, jež jsou aktuálně ve stavu *ve frontě* nebo *zpracováván*. Tyto exporty se pak seřadí od nejnovějšího a šablona nejnovějšího z nich se automaticky rozbalí. Tento problém byl obdobně vyřešen u importů.

3.3.2 Import

Realizace importu byla poněkud složitější jak později přiblížím, započala však stejně a to vytvořením rozevírací tabulky pro šablony a jejich importy pomocí komponeny **x-data-table**. Další krok byl opět stejný jako u exportů, přenesení konfigurace importu do komponenty pro nastavení šablony. Tato komponenta byla vytvořená pomocí stejných principů jako u exportů a sdílela tak i funkcionalitu pro zobrazení existujících šablon a použití komponenty pouze pro účel čtení.

Zabalení do komponenty konfigurace šablony přineslo stejné problémy jako u exportů a to ukládání konfigurace uvnitř komponenty. Tento problém byl vyřešen stejně, v případě importů však bylo zapotřebí přidat další krok. U importu je totiž před uložením komponenty potřeba splnit požadavky ve formě namapování všech povinných atributů. Je tedy možné, že se uživatel pokusí o pokračování aniž by tyto požadavky byli splněny. Je tedy potřeba přidat kontrolu, zda je vůbec možné šablonu uložit a to pomocí *emitu* z komponenty šablony, držící informace o mapování povinných atributů, do rodiče.

Řešení tedy ve finále funguje tak, že potom co uživatel klikne na tlačítko vedoucí k uložení šablony se změní hodnota *props* předaná do komponenty šablony, ve které je tato akce zachycena *watcherem* a ten zkontroluje, jestli jsou splněny požadavky pro uložení. Po kontrole pomocí *emitu* pošle rodiči informaci o tom, jestli je vše v pořádku a rodič poté na základě této informace buď uloží šablonu, nebo informuje uživatele o chybě šablony.

U importů byla navíc přidána *stepper* komponenta, provádějící uživatele importem. Tato komponenta již ve vuetify existuje jako **v-stepper**, kterou jsem mohl využít. Jediná komplikace při implementaci této části bylo využití jedné stránky pro tři různé případy užití. Do *steppru* se totiž dá jak jsem již zmínil za různých předpokladů. Jelikož se v projektu mezi jednotlivými stránkami přechází pomocí *roteru* není možné použít *props*, protože komponenta, ze které se přechází, nemusí být rodičem komponenty, do které se přechází. Při přechodu přes *router* se využívá adresa pro identifikaci stránky. Je však možné použít pro různé adresy stejnou výchozí stránku. Rozhodl jsem se tedy využít tohoto faktu a rozlišovat případ užití na základě adresy, která byla použita při přechodu do komponenty.

Prvním případem užití je vytvoření nového importu bez využití šablony. V tomto případě je tedy potřeba, aby měl uživatel možnost zvolit jméno šablony, jež bude vytvořena společně s importem, stejně tak jako doménu tohoto importu a soubor, který bude importován. V druhém kroku je mu pak potřeba zobrazit komponentu konfigurace šablony tak, aby mohl namapovat svou tabulku na zvolenou doménu a při přechodu do třetího kroku pak tuto konfiguraci uložíme jako novou šablonu.

Tento průchod s sebou však nese značný problém, a to že uživatel může proces nového importu kdykoli opustit. Toto se bude jevit jako problém v případě, že uživatel bude chtít akci opustit během druhého kroku, kdy FE již vytvořil novou instanci importu a sním i novou šablonu, ale uživatel o tom ještě neví. Pokud se rozhodne akci v tomto kroku opustit a vrátit se na stránku se šablonami, uvidí tam novou šablonu s vygenerovaným jménem, což by mohlo vést ke zmatení a musíme tomu tedy zabránit. Využijeme k tomu tedy metodu transakcí, jež zajistí navrácení do původního stavu při nedokončení procesu uložení nové šablony. Právě k těmto akcím je náš *router* vybaven funkcemi typu *route guard*, které umožňují vykonání funkce před jakýmkoliv routováním. S využitím funkce *beforeRoute* tedy dokážeme provést kontrolu jestli se nacházíme ve stavu, kdy je potřeba před provedením *routu* navrátit stav aplikace zpět do původního stavu před započetím transakce. Pokud ano tak se zobrazí upozornění uživateli pomocí nově vytvořené komponenty **InformDialog**, která ho informuje o tom, že pokud se rozhodne pokračovat a opustit stránku, tak bude jeho neuložená šablona smazána.

Dalším možným průchodem touto komponentou je vytvoření importu na základě již existující šablony, v tomto případě stačí v prvním kroku dát uživateli možnost zvolit soubor na nahrání a v kroku druhém mu zobrazíme jeho zvolenou šablonu, použitou na zvolený soubor, pouze ke čtení. Při přechodu do tohoto kroku se tedy nic neukládá, jelikož nemá uživatel možnost šablonu nijak měnit.

Poslední možnost využití této komponenty nastane, chce-li se uživatel vrátit k již započatému, ale nedokončenému importu. V tomto případě se komponenta chová velice podobě jako v případě s použitím šablony s rozdílem, že se vykreslí hned na druhém kroku *steppru* bez možnosti se vrátit ke kroku prvnímu. Není totiž potřeba vybírat soubor k nahrání vzhledem k tomu byl již vybrán. Druhý a třetí krok jsou pak stejné.

Poslední krok *stepper* komponenty zobrazuje statistiky ohledně importu. Statistiky se však mohou načítat delší dobu a nemáme žádný jednoduchý zůsob jak by nás BE informoval ve chvíli, kdy budou statistiky hotové. Informace o statistikách se nachází přímo v endpoitu s informacemi o importu, dokud však statistiky nejsou hotové má tento atribut hodnotu *null*. Tento problém se tedy řeší pomocí aktivního *pollingu*, kdy v intervalech dotazujeme API na informace ohledně importu a pokaždé zkontrolujeme, jestli je hodnota statistik stále *null*, pokud ano tak pokračujeme a ptáme se dál do té doby, než budou statistiky nabývat jiné hodnoty. V této chvíli se pak přestaneme dotazovat a zobrazíme komponentu prezentující statistiky importu. Toto řešení se stejně využito pro statistiky exportů.

3.3.3 Shrnutí problémů

Nyní opět shrnu a ohodnotím hlavní problémy, které jsem řešil při implementaci druhé iterace.

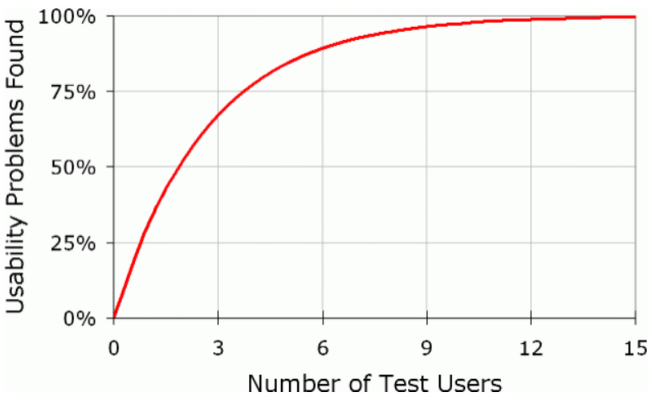
- Refaktoring kódu první iterace
  Náročnost: 4
- Překlady API dat
  Náročnost: 4
- Ukládání šablony
  Náročnost: 3
- Rozdělení průchodů vytváření importu
  Obtížnost: 2
- Zajištění navrácení stavu při nedokončené importu
  Obtížnost: 4

- Aktivní polling statistik importu a exportu
  Obtížnost: 3

3.4 Testování

Testování druhé iterace bude provedeno pomocí uživatelských testů. Struktura testu a počet testovaných uživatelů byli zvoleny podle výzkumu NN/g (Nielsen Norman Group).

Zde můžete najít jejich článek o výzkumu ideálního počtu testovaných uživatelů [12]. Tento výzkum dochází k závěru, že poměr nalezených chyb v rozhraní a testovaných uživatelů následuje tuto křivku 3.11. Na tomto základě bylo tedy rozhodnuto využít právě 5 testovacích uživatelů. V článku je také popsána důležitost opakovaného testování, z toho důvodu bylo rozhodnuto nejprve provést čtyři z pěti testů a na základě jejich výsledků opravit největší nedostatky a až poté provést test poslední.



Obrázek 3.11: Poměr nalezených chyb a testovacíh uživatelů podle NNG

3.4.1 Struktura uživatelského testu

Zde bude popsána struktura uživatelského testu, jež bude používána při uživatelském testování nového rozhraní. Struktura testů je navržena na základě tohoto článku [13] opět založeném na výzkumu NN/g.

Testování bude začínat krátkým dotazníkem zjišťující základní informace o testovém uživateli, společně s jeho zkušenostmi se skladovým systémem Atlantis. Pokud bude mít uživatel zkušenosti se systémem, bude dotazován přímo na zkušenosti s importováním a exportováním dat.

Poté proběhne samotný test dále přiblížených případů užití, při kterém budou sděleny jen potřebné informace. Pokud uživatel nebude schopen případ užití bez pomoci dokončit, bude mu poskytnuta rada a test bude dál pokračovat.

Na konci testování bude uživateli předložen další dotazník zjišťující nedostatky, které na rozhraní zaznamenal. Také bude dotazován, zda by nové rozhraní používal, kdyby bylo k dispozici.

3.4.2 Dotazy před testováním

1. Jak se jmenujete?
2. Kolik vám je let?
3. Jaké je vaše spojení s aplikací Atlantis?
4. Pracoval/a jste již v portálu Atlantis? Jak dlouho?
5. Pokud ano setkal/a jste se s importem nebo exportem dat?
6. Jak aktuálně řešíte Export dat?
7. Jak aktuálně řešíte Import dat?

3.4.3 Testované případy užití

V této části představím každý úkol, jež bude při testování uživateli zadán společně s krátkým popisem přibližujícím hlavní cíle testu. Následně popíši ideální potup, který nové řešení nabízí.

Exportuj Skladové položky.

Tento test bude jako první, s cílem otestovat přehlednost hlavní stránky exportů a zjistit, zda se v ní nový uživatel orientuje a bude schopný vykonat jednoduchou akci jako použít šablonu k vytvoření exportu.

Očekávaný postup uživatele pak bude následovný. V tabulce s přehledem šablon exportu si uživatel najde požadovanou šablonu a pomocí akce **Vytvořit export** vytvoří nový export této šablony. Následně počká na načtení statistik o exportu a když uvidí, že je vše v pořádku nechá export zpracovat.

Exportuj základní informace o odběratelích a jejich doručovací adresy.

Tento test má za úkol otestovat jak přehledné je konfigurování šablon pro export. Uživatel tak bude muset vytvořit novou šablonu podle zadání, které mu bude sděleno. Po jejím vytvoření ji pak využije k vytvoření nového exportu.

Ideální postup pro řešení je následující. V tabulce s přehledy šablon exportu uživatel zvolí možnost **Vytvořit šablonu exportu**, na stránce vytvoření nové šablony vyplní její jméno pomocí textového pole a doménu pomocí select pole. Nakonec využije tabulku pro konfiguraci šablony pro zvolení exportovaných sloupců dle požadavků. Tuto šablonu uloží a v tabulce přehledu šablon ji využije, jako v předchozím případu užití, na vytvoření nového exportu.

Importuj testovací soubor *Buyers*.

Tento test zadá uživateli na vytvoření nový import, pro který již existuje šablona a bude testovat jak přehlednost tabulky se šablonami importů, tak i stepperu, který provádí uživatele tvorbou nového importu.

Očekávaný postup uživatele je následovný. Uživatel si po prohlédnutí zadaného souboru v tabulce se šablonami importu najde požadovanou šablonu, u které zvolí akci **Vytvořit Import**. Na stránce nového importu pomocí okna vložení souboru vloží soubor *Buyers* a stiskne tlačítko **pokračovat**. Poté uvidí testovací soubor namapován na zvolenou šablonu a po kontrole stiskne tlačítko **pokračovat**. V posledním kroku počká na načtení statistik nového importu a po načtení dat stiskne tlačítko **Zpracovat Import** a odešle tak import ke zpracování.

Importuj testovací soubor *ProductsBasicInfo*.

Tento test bude zjišťovat použitelnost konfigurace nových šablon, především tedy mapování sloupců na zvolenou doménu. Tento krok je hlavním krokem vytváření nových šablon importu a je tak důležité, aby jej uživatele dokázali provádět.

Ideální postup tohoto případu užití je následovný. Uživatel, po zobrazení zadaného souboru, v tabulce šablon exportu zvolí akci **Vytvořit Import bez Šablony**. Po zobrazení stránky na vytvoření importu vloží soubor *ProductsBasicInfo* pomocí pole pro vložení souboru, poté pomocí textového pole vyplní jméno šablony, která bude vytvořena pomocí nového importu. Nakonec pomocí select pole vyplní doménu, do které se chystá importovat. Po dokončení těchto akcí stiskne tlačítko **pokračovat**. V dalším

kroku si pomocí tabulky pro mapování namapuje všechny sloupce testového souboru na zvolenou doménu a zvolí primární klíče. Uživatel po dokončení těchto kroků opět stiskne tlačítko **pokračovat**. V poslední části uživatel opět počká na načtení statistik importu a po jejich načtení stiskne **Zpracovat Import** a tím odešle import ke zpracování.

3.4.4 Dotazy po testování

1. Jaké věci vám v novém rozhraní chyběli, nebo přišli nejasné?
2. Jaké věci jste považoval/a za přínosné?
3. Je nové řešení lepší než váš aktuální postup Importu nebo exportu?
4. Používál/a byste nové řešení, pokud by bylo v portálu k dispozici?

3.4.5 0. Testovací skupina

V nulté testovací skupině se nacházel pouze vedoucí práce Ing. Jiří Hunka. Toto testování mělo za účel odhalit nedostatky v testovacím procesu samotném. Vedoucí práce po testování poskytl výhrady k testovacímu procesu, na jejichž základě byl testovací proces pro další skupiny upraven. Z těchto důvodů zde nebude výsledek testování zobrazen.

Jediná výhrada vedoucího práce k průběhu testu bylo příliš návodné zadávání úkolů. Na tomto základě byli upraveny zadání jednotlivých úkolů na ty, jež jsou viditelné ve struktuře uživatelského testu.

3.4.6 1. Testovací skupina

Tato skupina byla největší a tedy i hlavní skupinou testování. Testování proběhlo ve dnech 19-20.4.2023. Jednalo se o čtyři testovací uživatele, kteří se zásadně lišili ve zkušenostech se systémem Atlantis. Ve skupině byl jak uživatel, který v systému nikdy nepracoval, tak i hlavní vývojář FE. Nedostatky rozhraní odhaleny touto skupinou tedy pokrývali jak chyby v návrhu rozhraní samotném, tak i pouhé nedokonalosti v realizaci. Výstupy z jednotlivých testování najdete v příloze práce

Následně budou přiblíženy výsledky testování. Výsledky budou zobrazeny jako výpis všech nedostatků nalezených při testování společně s jejich důležitostí na opravu (Vysoká, Střední, Nízká), Náročností na opravu (1-5, kde 5 je nejnáročnější) a zda bude jejich oprava stále spadat pod druhou iteraci, nebo se přesune jako požadavek na iteraci třetí.

1. Ve všech tabulkách nového rozhraní nefunguje správně filtrování.
   Důležitost: Vysoká **Náročnost:** 5 **Iterace:** Druhá
2. Při konfiguraci šablony importu není jasné, že je potřeba tabulku zarovnat.
   Důležitost: Vysoká **Náročnost:** 5 **Iterace:** Druhá
3. Při konfiguraci exportu/importu nejsou přeloženy všechny informace, se kterými uživatel pracuje.
   Důležitost: Vysoká **Náročnost:** 5 **Iterace:** Druhá
4. Při konfiguraci šablony importu nejsou primární klíče pro uživatele srozumitelné.
   Důležitost: Vysoká **Náročnost:** 4 **Iterace:** Druhá
5. Při vytváření šablon je povoleno jméno pouze s délkou 20 znaků, to by mohlo být nedostačující.
   Důležitost: Vysoká **Náročnost:** 1 **Iterace:** Druhá
6. Není jasné, že se při načítání statistik nevykonává samotná akce importu/exportu.
   Důležitost: Střední **Náročnost:** 3 **Iterace:** Druhá
7. Při mapování atributů šablony importu není jasné jak odmapovat již namapovaný sloupec.
   Důležitost: Střední **Náročnost:** 3 **Iterace:** Druhá
8. Přidat možnost na přejmenování vytvořené šablony pro export.
   Důležitost: Střední **Náročnost:** 2 **Iterace:** Druhá
9. Na stránce se šablonami importu/exportu chybí v pravém dolním rohu kulaté tlačítko na přidání nového záznamu, jako je to ve zbytku aplikace.
   Důležitost: Střední **Náročnost:** 1 **Iterace:** Druhá
10. Chybí možnost náhledu na statistiky již zpracovaného importu/exportu.
    Důležitost: Střední **Náročnost:** 1 **Iterace:** Druhá
11. Po zpracování exportu nebo importu by mohlo uživateli přijít oznámení.
    Důležitost: Střední **Náročnost:** 5 **Iterace:** Třetí

12. Do nového rozhraní exportu se nedá nijak dostat pomocí tabulky, kterou chce uživatel exportovat, jako to funguje v aktuálním řešení.

    Důležitost: Střední **Náročnost:** 4 **Iterace:** Třetí

13. Při mapování, v konfiguraci šablony importu, tabulka pro výběr atributu zakrývá mapovaný sloupec a uživatel nevidí co mapuje.

    Důležitost: Nízká **Náročnost:** 1 **Iterace:** Druhá

14. Barvy tlačítek při průchodu importem jsou zelená, což není konzistentní se zbytkem aplikace.

    Důležitost: Nízká **Náročnost:** 1 **Iterace:** Druhá

15. Při konfiguraci importu vypadá kurzor při najetí na povinné atributy, jako by se na ně dalo klikat.

    Důležitost: Nízká **Náročnost:** 1 **Iterace:** Druhá

16. Při mapování, v konfiguraci šablony importu, chybí možnost vyhledávat atribut podle jména.

    Důležitost: Nízká **Náročnost:** 5 **Iterace:** Třetí

17. Při schovávání a zobrazování sloupců šablony exportu se uživateli mění pohled na tabulku.

    Důležitost: Nízká **Náročnost:** 4 **Iterace:** Třetí

Výsledky testování 1.skupiny odhalili nemalé množství problémů nového rozhraní, které bylo potřeba vyřešit. Následně proběhla oprava nalezených nedostatků před pokračováním uživatelského testování s 2. testovací skupinou. Analýza a oprava nedostatků byly bohužel alokovány pouze na dny 20-25.4.2023, a to z důvodu nedostatku času. Mezi testováními byly tedy opraveny jen tyto nedostatky 2, 3, 5, 7, 9, 10, 13, 14, 15. Opravené nedostatky byly zvoleny na základě jejich důležitosti a náročnosti.

3.4.7 2. Testovací skupina

Druhá testovací skupina, testována dne 25.4.2023 se, jak jsem již zmínil, skládala pouze z jednoho testovacího uživatele. Příčiny tohoto faktu byly především nedostatek času a jednoduše dosažitelných testovacích uživatelů. Uživatel v této skupině i tak dokázal upozornit na nedostatky, jež nebyly dosud odhaleny.

Nyní budou opět shrnuty všechny nedostatky rozhraní odhaleny touto skupinou. Výsledky budou zahrnovat i nedostatky již zmíněné druhou testovací skupinou, pokud mezi testováním došlo k pokusu o jejich opravu, v opačném případě budou vynechány.

1. Uživateli stále není jasné, že je potřeba zarovnat tabulku.

   Důležitost: Vysoká **Náročnost:** 5 **Iterace:** Druhá

2. Šířka tabulek pro import a export není konzistentní se zbytkem aplikace.

   Důležitost: Nízká **Náročnost:** 1 **Iterace:** Druhá

3. Při konfiguraci exportu by měl mít uživatel možnost schovat všechny sloupce a poté si vybrat pouze sloupce, které chce v exportu.

   Důležitost: Střední **Náročnost:** 3 **Iterace:** Třetí

Po analýze výsledků testování se opět přešlo k opravě nalezených chyb a nedostatků. Tentokrát bylo cílem opravit všechny zbylé nedostatky z prvního testu, jež nebyly přesunuty na další iteraci, společně s nově nalezenými nedostatky.

3.5 Zhodnocení

V následující části zhodnotím průběh jednotlivých fází vývoje druhé iterace. Také se zaměřím na budoucí vývoj tohoto rozhraní mimo tuto BP.

3.5.1 Sběr a analýza požadavků

Sběr požadavků byl v druhé iteraci proveden pomocí interviw přímo se zadavatelem práce Ing. Jiřím Hunkou, což se projevilo jako mnohem lepší volba. Zadavatel tak mohl přímo sdělit jeho požadavky na funkcionalitu celku, ale stejně tak jeho požadavky na UI / UX. Díky tomuto interview byly tak nasbírány klíčové požadavky, jež v první iteraci chyběly. Sběr požadavků této iterace tak hodnotím jako úspěšný postup.

3.5.2 Návrh

Tento proces v druhé iteraci proběhl s důrazem právě na detailnost jednotlivých návrhů, jelikož právě tento nedostatek byl příčinou nemalého množství problémů v iteraci první. Návrhy byly tedy vytvářeny tak, aby

se mohlo při realizaci postupovat přesně podle nich a nebylo zapotřebí domýšlet nový design. Návrhy byly také, na rozdíl od první iterace, při vývoji aktualizovány a díky tomu se mohou rovnou využít k návrhu pro 3. iteraci. Postup vytváření návrhů tedy hodnotím opět kladně.

3.5.3 Realizace

Realizace v druhé iteraci probíhala velmi podobně jako v iteraci první pouze s rozdílem většího důrazu na kvalitu kódu. Díky lépe zpracovanému návrhu jsem při realizaci mohl postupovat přesně podle něj. Nastali samozřejmě i chvíle, kdy bylo zapotřebí se od návrhu odklonit, ale bylo jich podstatně méně.

Největší překážka realizace se ukázala být kvalita psaného kódu. Jelikož se k nasazení dostala až tato iterace, bylo zapotřebí, aby proběhla revize kódu, kterou provedl Ing. Oldřich Malec jako hlavní FE vývojář projektu. Při této revizi se ukázalo, že můj kód nesplňuje zásadní požadavky na nasazení. Jelikož se jedná o mou první práci v komerčním projektu, neuvědomoval jsem si, jak podstatné je udržovat v aplikaci kvalitní a konzistentní kód. Z tohoto důvodu byla polovina realizace v druhé iteraci zaměřena na refaktoring kódu.

Po dokončení realizace druhé iterace rozhraní je tedy systém schopný provádět komplexní exporty, u kterých si uživatel může vybrat přesně jaké informace chce a nechce exportovat. Jelikož jsou tyto konfigurace uloženy v šablonách, umožňují tak uživateli je znovu a rychle využívat, bez potřeby častého konfigurování nových exportů. Nové exporty také podporují export do více typů souborů, konkrétně xlsx, csv, ods. Rozhraní je dále schopné importovat uživatelské soubory do databáze systému. Konfigurace těchto importů jsou opět uloženy v šablonách a umožňují tak konfigurace znovu použít a vyhnout se tak opakovanému konfigurování. V poslední řadě systém informuje uživatele o množství exportovaných dat před samotným exportem, stejně tak informuje uživatele o následcích importu před jeho provedením.

3.5.4 Testování

Testování iterace druhé se s první moc porovnat nedá. V druhé iteraci proběhlo uživatelské testování dvou skupin. Nejprve bylo rozhraní testováno první skupinou uživatelů a vážnější nedostatky nalezeny touto skupinou byli opraveny. Poté proběhlo testování s druhou skupinou. Výstupy testování obou skupin se poté analyzovali a rozhodlo se, zda se opraví před nasazením, nebo se jejich oprava přesune do třetí iterace.

Testování proběhlo bez větších problému a odhalilo velké množství chyb a nedostatků nového rozhraní. Struktura uživatelských testů se tak projevila jako správně navržená. K testování proto nemám větší výhrady, pouze snad malé množství času, jež jsem si vyhradil na opravy chyb a nedostatků mezi jednotlivými testovacími skupinami.

3.5.5 Budoucí vývoj

Jak je zřejmé z uživatelského testování vývoj tohoto rozhraní není ještě úplně dokončen. Během testování byli určité požadavky přesunuty do třetí iterace, která však již není součástí této BP. Se zadavatelem práce bylo dohodnuto že ve vývoji FE budu osobně dále pokračovat mimo rozsah této práce a budu ho tak schopen rozšířit o nové požadavky nasbírané v testování.

Závěr

Motivací této práce bylo ušetření času jak uživatelů, tak i vývojářů skladového systému Atlantis, díky nové funkcionalitě importování dat přímo pomocí rozhraní aplikace, která nebyla v předchozím řešení možná.

Cílem mé práce bylo navržení a vytvoření funkčního uživatelského rozhraní pro importování a exportovaní dat skladového systému Atlantis. Tohoto cíle jsem dosáhl pomocí analýzy předchozího řešení importu a exportu skladového systému. Na základě nedostatků předchozího řešení společně s pohovorem s BE vývojářem projektu jsem byl schopen nasbírat a analyzovat požadavky na nové rozhraní. Pomocí nasbíraných požadavků jsem pak vytvořil návrh první verze tohoto rozhraní, jež jsem poté realizoval. Tato první funkční verze rozhraní byla následně podrobena testování, při kterém byly odhaleny kritické nedostatky v návrhu a nebyla tak nasazena do produkčního prostředí.

Na základě testování první verze a její nedostatků jsem opět nasbíral a analyzoval požadavky na rozhraní pro import a export dat systému. Požadavky byly poté využity k vytvoření detailního návrhu druhé verze uživatelského rozhraní. Návrh byl poté realizován a podroben testování. Testování bylo tentokrát provedeno stylem uživatelských testů a po jeho provedení a analýze bylo rozhodnuto nasadit nové rozhraní do produkčního prostředí. Všechny cíle práce byli tedy splněny.

Aktuálně je tedy FE připravený k nasazení do produkčního prostředí aplikace a po jeho nasazení přinese možnost provádět import dat přímo v aplikaci Atlantis. Díky novému návrhu se šablonami importu mohou importy bezpečně provádět přímo uživatele aplikace a ušetřit tak práci vývojářům.

Díky provedeným uživatelským testům jsou k dispozici informace pro budoucí rozvoj tohoto rozhraní, na kterém se budu osobně podílet. Bude se

jednat především o rozšíření rozhraní o nasbírané požadavky.

Bibliografie

1. TUTORIALSPOINT. *Overview* [online]. tutorialspoint.com, 2006. [cit. 2023-04-17]. Dostupné z: https://www.tutorialspoint.com/sdlc/sdlc_overview.htm.
2. POINT, Tutorials. *Waterfall Model* [online]. tutorialspoint.com, 2019. [cit. 2023-04-17]. Dostupné z: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.
3. PARADIGM, Visual. *What is a Software Process Model?* [online]. visual-paradigm.com, 2020. [cit. 2023-04-23]. Dostupné z: https://www.visual-paradigm.com/guide/software-development-process/what-is-a-software-process-model/.
4. TUTORIALSPOINT. *Iterative Model* [online]. tutorialspoint.com, 2019. [cit. 2023-04-17]. Dostupné z: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm.
5. MALEC, Oldřilch. *Frontend skladového systému* [online]. 2020. [cit. 2023-04-17]. Dostupné z: http://hdl.handle.net/10467/86593. Dis. pr.
6. CONTRIBUTORS, MDN. *JavaScript* [online]. MDN Web Docs, 2019. [cit. 2023-04-18]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/javascript.
7. . *Introduction — Vue.js* [online]. vuejs.org, 2020. [cit. 2023-04-17]. Dostupné z: https://vuejs.org/guide/introduction.html#what-is-vue.
8. VUETIFYJS. *Why Vuetify — Vuetify.js* [online]. https, 2020. [cit. 2023-04-24]. Dostupné z: https://vuetifyjs.com/en/introduction/why-vuetify/.

9. LIKEDIN. *Conjoining FURPS and MoSCoW to Analyse and Prioritise Requirements* [online]. linkedin.com, 2003. [cit. 2023-04-17]. Dostupné z: `https://www.linkedin.com/pulse/conjoining-furps-moscow-analyse-prioritise-jonathan-dyson`.
10. LEARNBYEXAMPLE.ORG. *Python Nested List* [online]. Learn By Example, 2019. [cit. 2023-04-24]. Dostupné z: `https://www.learnbyexample.org/python-nested-list/`.
11. KOPF, Ben. *The Power of Figma as a Design Tool* [online]. Toptal Design Blog, 2018. [cit. 2023-04-23]. Dostupné z: `https://www.toptal.com/designers/ui/figma-design-tool`.
12. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users*. Nielsen Norman Group, 2000. Dostupné také z: `https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`.
13. MORAN, Kate. *Usability Testing 101*. Nielsen Norman Group, Nielsen Norman Group, 2019. Dostupné také z: `https://www.nngroup.com/articles/usability-testing-101/`.
14. MALEC, Oldřich. *Oldřich Malec Uživatelský Test*. 2023.
15. HEJDA, Max. *Max Hejda Uživatelský Test*. 2023.
16. DVOŘÁK, Matrin. *Matrin Dvořák Uživatelský Test*. 2023.
17. KUDRNA, Libor. *Libor Kudrna Uživatelský Test*. 2023.
18. PLYSKACH, Andrii. *Andrii Plyskach Uživatelský Test*. 2023.

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

BP Bakalářská práce

CSV Comma-separated values

FURPS Funkcionality, Usability, Reliability, Performance, Supportability

HTTP Hypertext Transfer Protoco

JSON Javascript Object Notation

MoSCoW Must have, Should have, Could have, Won't have

NN/g Nielsen Norman Group

ODS OpenDocument Spreadsheet

REST Representational State Transfer

UI User interface / Uživatelské rozhraní

URL Uniform Resource Locator

UX User experience / Uživetleký prožitek

Příloha B

Testování

B.1 0. Uživatelský test

Video + Audiozáznam z testování `https://youtu.be/FNDFoJuWA_8`

B.2 1. Uživatelský test

[14]

Dotazník před testem

1. **Jméno:** Oldřich Malec
2. **Věk:** 28
3. **Pracovní pozice:** Hlavní vývojář FE a Projektový manažer aplikace Atlantis
4. **Zkušenosti s aplikací Atlantis:** Zkušenosti s aplikací jsou velmi pokročilé, v aplikaci se také zabývá správou a jako support pro zákazníky.
5. **Jak aktuálně řešíte Export dat:** Export dat řeší primárně pomocí aktuálního řešení exportu celé tabulky. Pokud je však třeba propojit více tabulek, nebo jinak komplexní export, data sbírá přímo s databáze bez využití FE.
6. **Jak aktuálně řešíte Import dat:** Import dat aktuálně řeší vytvořením scriptu, jenž využije API a přidává nebo modifikuje záznamy po jednom. Toto řešení zabere něco přes dvě hodiny.

Uživatelův postup při zadání: Exportuj skladové položky

Uživatel jako první využil již existující řešení exportu. Po výzvě, aby stávající řešení ignoroval, dokázal najít šablonu pro export skladových položek, kterou poté využil k samotnému exportu. Uživatel si však myslel, že proces načítání statistik exportu byl export samotný a nebylo mu tak jasné proč se poté export objevil ve stavu *ve frontě*. To byl však jediný problém tohoto testu.

Uživatelův postup při zadání: Exportuj základní informace o odběratelích a jejich doručovací adresy

Uživatel tentokrát rovnou požil nové řešení, jelikož aktuální řešení neposkytuje možnost upravovat exportovaná data. Po projití možných šablon se správně rozhodl si vytvořit šablonu vlastní, jež vytvořil bez problému a velice rychle pochopil jak funguje konfigurace šablony a šablonu správně nakonfiguroval. Poté šablonu využil na import jako při prvním testu.

Uživatelův postup při zadání: Importuj testovací soubor *Buyers*

Uživatel se po prohlédnutí souboru k importu ihned přesunul na stránku importů, kde opět začal procházet šablony dokud nenašel šablonu, u které si sice nebyl jist jestli bude fungovat, ale rozhodl se ji použít. Pomocí této šablony vytvořil nový import, kde byl trochu zmaten na stránce konfigurace, protože ji nemohl nijak upravovat a nebyl si jist proč. Jelikož si ale zvolil šablonu správnou tak mohl bez problému pokračovat na poslední krok importu, kde opět čekal na statistiky, u kterých už pochopil, že neprobíhá import samotný. Po načtení statistik odeslal import k procesu.

Uživatelův postup při zadání: Importuj testovací soubor *TestFile*

Uživatel opět započal prohlédnutím souboru, jež měl za úkol naimportovat a pokračoval hledáním správné šablony. Poté co nenašel žádnou odpovídající šablonu se zasekl, protože nemohl najít možnost tvorby nové šablony. Musel jsem ho informovat, že šablona samotná se nedá vytvořit a že musí použít tlačítko **vytvořit import bez šablony**. Uživatel pokračoval a po vyplnění informací pro nový import se dostal ke konfiguraci šablony, mapování sloupců pochopil velice rychle a správně namapoval soubor. Nebyl si vůbec jist co znamenají primární klíče a rozhodl se je tedy ignorovat. Také byl schopen bez problému nastavit hlavičku tabulky společně s prvním řádkem a sloupcem. Po konfiguraci pokračoval k poslednímu kroku kde si po načtení prohlédl statistiky a import poslal k procesu.

Dotazník po Testu

1. **Jaké věci v novém rozhraní chyběli, nebo nebyli nejasné:**

   Uživateli chybělo při vytváření nové šablony exportu klasické tlačítko, které se nachází ve zbytku aplikace. Také mu nebylo hned jasné, že načítání statistiky nebylo proces exportu. Nebylo mu hned jasné, že při vytváření importu pomocí šablony je konfigurace pouze na náhled pro kontrolu. Také by chtěl, aby si po dokončení importu mohl opět zobrazit statistiky, což není aktuálně možné. Dále mu chyběla možnost při mapování atributů vyhledávání v listu atributů k mapování. Jako poslední si nebyl hned jist, jak vytvořit novou šablonu importu.

2. **Jaké věci se vám líbili:**

   Uživateli se líbilo jak funguje mapování atributů při konfiguraci importu, stejně tak se mu i líbilo jednoduché ovládání konfigurace viditelnosti sloupců při tvorbě šablony exportu.

3. **Je nové řešení lepší než váš aktuální postup:**

   Uživatel si uvědomuje možnosti, které nový export nabízí, ale celkově mu přišel postup složitější a pomalejší. Vzhledem k absenci aktuálního řešení importu mu nové řešení přišlo jistě lepší.

4. **Používal/a byste nové řešení, pokud by bylo v portálu k dispozici:**

   Uživatel by používal nové řešení exportu pokud by to bylo nutné, tedy pokud by potřeboval vytvořit komplexnější export, jinak by se držel aktuálního řešení kvůli jednoduchosti a rychlosti. U importu by nové řešení jistě využil.

B.3 2. Uživatelský test

[15] Video záznam z testování `https://youtu.be/MtqjC9Gweto`

Dotazník před testem

1. **Jméno:** Max Hejda
2. **Věk:** 25
3. **Aplikace Atlantis:** Uživatel není zaměstnán jako vývojář aplikace, ani není jejím uživatelem
4. **Zkušenosti s aplikací Atlantis:** Zkušenosti s aplikací nejsou žádné, ale uživatel má zkušenosti s importem a exportem v jiných aplikacích.

Uživatelův postup při zadání: Exportuj skladové položky

Uživatel jako první využil staré řešení. Na výzvu, aby opakoval úkol jinak, našel stránku exportů, kde použil šablonu *products* bez její kontroly. Nepočkal na statistiky o exportu a export rovnou provedl. Tento postup vedl ke správnému řešení, kdyby však šablona *products* neobsahovala chtěnou konfiguraci uživatel by se to nedozvěděl.

Uživatelův postup při zadání: Exportuj základní informace o odběratelích a jejich doručovací adresy

Uživatel pochopil, že potřebuje novou šablonu, při jejím vytváření si ale nejprve nevšiml, jak šablonu konfigurovat. Vytvořil tedy neupravenou šablonu, kterou se poté snažil upravit, což aktuálně není možné. Poté co zjistil jak šablonu konfigurovat vytvořil novou a tu podle zadání nastavil a použil.

Uživatelův postup při zadání: Importuj testovací soubor *Buyers*

Uživatel přešel na stránku importů, kde nevyužil žádnou existující šablonu a použil možnost **Nový import bez Šablony**. Na další stránce porozuměl automatickému namapování souboru a pokračoval na stránku importu, kde opět nepočkal na statistiky a import odeslal ke zpracování.

Uživatelův postup při zadání: Importuj testovací soubor *TestFile*

Uživatel opět vytvořil nový export bez zvážení použití šablony, při volbě importované oblasti dat musel uživatel hádat, protože se nepodíval na zadaný

soubor, po testování mi sdělil, že očekával náhled na soubor. Po tom zjistil, že byl jeho odhad špatný se vrátil a svou chybu opravil. Při konfiguraci šablony uživatel ignoroval informace o povinných atributech a rovnou začal mapovat tabulku. Uživatel také nepochopil, jak odmapovat namapovaný sloupec. Uživatel si vůbec nevšiml zarovnání tabulky. Poté, co domapoval všechny sloupce pokračoval na poslední krok, kde tentokrát počkal na statistiky, kterým prý porozuměl.

Dotazník po Testu

1. **Jaké věci v novém rozhraní chyběli, nebo nebyli nejasné:**

   Nebylo dostatečně jasné, jak odmapovat sloupce při konfiguraci importu. Také mu chvilku trvalo, něž odhalil, jak konfigurovat šablonu exportu. Při testování vůbec nevyužil možnost zarovnání tabulky při importu.

2. **Jaké věci se vám líbili:**

   Uživateli se líbilo, jak funguje konfigurace šablony exportu.

3. **Jaké je toto řešení v porovnání s ostatními aplikacemi:**

   Uživatel hodnotí toto řešení jako přehledné pro danou aplikaci, kde se pracuje s komplexními datovými strukturami.

B.4 3. Uživatelský test

[16]

Dotazník před testem

1. **Jméno:** Martin Dvořák
2. **Věk:** 22
3. **Aplikace Atlantis:** FE vývojář aplikace, někdy dělá malé úpravy na BE
4. **Zkušenosti s aplikací Atlantis:** S aplikací je velmi dobře seznámen, při vývoji FE ji testuje.
5. **Jak aktuálně řešíte Export dat:** Při potřebě exportu využívá staré řešení, někdy použije rovnou databázi.
6. **Jak aktuálně řešíte Import dat:** Využívá databázi.

Uživatelův postup při zadání: Exportuj skladové položky

Uživatel rovnou přešel na stránku s exporty, kde se nepokusil využít existující šablonu, ale rovnou si vytvořil novou. Rychle zjistil jak ji nakonfigurovat, ale nechal ji v základní podobě. Po vytvoření mu chyběla možnost šablonu upravit, ale poté ji využil na nový export, počkal na statistiky exportu a po jejich načtení export nechal provést.

Uživatelův postup při zadání: Exportuj základní informace o odběratelích a jejich doručovací adresy

Vzhledem k předchozímu testu postupoval uživatel obdobně, jen nakonfiguroval šablonu podle zadání a tentokrát se rozhodl přeskočit čekání na statistiky.

Uživatelův postup při zadání: Importuj testovací soubor *Buyers*

Uživatel přešel na stránku s importy, kde opět nevyužil existující šablony ale rozhodl se vytvořit nový import bez šablony. Při zadání informací zvolil oblast dat správně podle názvu souboru. Při konfiguraci si četl všechny poskytnuté popisky. Jelikož se soubor namapoval sám, nebylo mu jasné jestli musí něco v tomto kroku dělat. Po chvíli se rozhodl pokračovat na poslední krok, kde počkal na statistiky a po jejich načtení import dokončil.

Uživatelův postup při zadání: Importuj testovací soubor *TestFile*

Uživatel opět vytvořil nový import bez šablony, odhadoval však oblast dat importu jelikož si předem neprohlédl soubor. Na další stránce pak očekával možnost se opravit, aniž by se musel vracet k předchozímu kroku. Soubor byl schopen správně namapovat společně s zarovnáním tabulky. Import poté úspěšně provedl.

Dotazník po Testu

1. **Jaké věci v novém rozhraní chyběli, nebo nebyli nejasné:**

   Při mapování souboru k importu se uživateli nelíbilo, že nejsou jednotlivé možnosti v menu přeložené. Vysvětlení primárních klíčů mu také přišlo velmi nejasné.

2. **Jaké věci se vám líbili:**

   Uživateli se líbil průchod importu pomocí stepperu, přišlo mu to přehledné a dobře se pomocí toho mohl orientovat. Také se mu líbilo zobrazení povinných atributů.

3. **Je nové řešení lepší než váš aktuální postup:**

   Uživateli přijde nové řešení přístupnější, jelikož je přímo v rozhraní aplikace.

4. **Používal/a byjste nové řešení, pokud by bylo v portálu k dispozici:**

   Uživatel by nové řešení využil při testování rozhraní, díky rychlé manipulaci s daty pomocí šablon.

B.5 4. Uživatelský test

[17]

Dotazník před testem

1. **Jméno:** Libor Kudrna
2. **Věk:** 44
3. **Aplikace Atlantis:** Uživatel aplikace, zastává roli skladníka a vedoucího skladu. Někdy působí jako tester nasazené verze aplikace.
4. **Zkušenosti s aplikací Atlantis:** S aplikací je velmi dobře seznámen jako její uživatel a tester.
5. **Jak aktuálně řešíte Export dat:** K exportování využívá aktuální řešení.
6. **Jak aktuálně řešíte Import dat:** K importu využívá přímo API.

Uživatelův postup při zadání: Exportuj skladové položky

Uživatel nejdříve využil stávající řešení a po výzvě k opakování exportu bez stávajícího řešení otevřel stránku s exporty a bez kontroly využil šablonu *Products*, která vedla ke správnému řešení. Uživatel se na šablonu ale nepodíval, takže to nemohl vědět. Při načítání statistik nebylo uživateli jasné na co čeká, ale export dokončil až po jejich načtení.

Uživatelův postup při zadání: Exportuj základní informace o odběratelích a jejich doručovací adresy

Uživatel pochopil, že bude muset vytvořit novou šablonu exportu. Konfigurace šablony mu přišla velice intuitivní a po jejím dokončení byl schopen ji uložit a využít pro nový export.

Uživatelův postup při zadání: Importuj testovací soubor *Buyers*

Uživatel si nejprve zobrazil zadaný soubor a poté zvolil cestu nového importu bez šablony. Při vyplňování oblasti dat importu správně zvolil doménu Odběratelé podle zadaného souboru, v dalším kroku však nepochopil automatické mapování a nebyl si jist jestli má něco dělat. Po chvíli se rozhodl pokračovat na další krok, kde počkal na statistiky importu a po jejich načtení import dokončil.

Uživatelův postup při zadání: Importuj testovací soubor *TestFile*

Uživatel si opět nejprve zobrazil soubor a poté se rozhodl využít import bez šablony. Informace vyplnil bez problému a při konfiguraci pochopil mapování sloupců bez problému, stejně tak jako zarovnání tabulky s hlavičkou. Primární klíče mu přišli nejasné a tak se je rozhodl ignorovat. V posledním kroku opět počkal na statistiky, po jejichž načtení import dokončil.

Dotazník po Testu

1. **Jaké věci v novém rozhraní chyběli, nebo nebyli nejasné:**

   Uživatel byl především nespokojený s nepřeloženými možnostmi v tabulce mapování. Také čekal nějaký typ oznámení po zpracování rozdělaného importu respektive exportu. Nové řešení exportu by také chtěl moci použít přímo z tabulky exportovaných dat, jako je to u stávajícího řešení.

2. **Jaké věci se vám líbili:**

   Uživateli přišlo nové řešení vzhledem k jeho komplexitě překvapivě málo složité a také mu vyhovovalo konfigurování šablony exportu.

3. **Je nové řešení lepší než váš aktuální postup:**

   Uživateli přijde nové řešení importu lepší než jeho stávající a řešení exportu je prý sice komplexnější, ale vidí výhody, které s sebou nese.

4. **Používal/a by jste nové řešení, pokud by bylo v portálu k dispozici:**

   Uživatel by při nasazení těchto nových rozhraní do aplikace využíval jak nový import, tak i export.

B.6 5. Uživatelský test

[18] Video + Audio záznam z testování `https://youtu.be/J5wPWSYKAjo`

Dotazník před testem

1. **Jméno:** Andrii Plyskach
2. **Věk:** 26
3. **Aplikace Atlantis:** hlavní BE vývojář
4. **Zkušenosti s aplikací Atlantis:** S aplikací je velmi dobře seznámen jako její vývojář a tester.
5. **Jak aktuálně řešíte Export dat:** Pro testování používá aktuální řešení exportu.
6. **Jak aktuálně řešíte Import dat:** Import v aplikaci nijak neřeší.

Uživatelův postup při zadání: Exportuj skladové položky

Uživatel rovnou přešel na novou stránku pro export, kde využil šablonu *products*, aniž by se na ní podíval. Tato šablona byla sice správně použita, ale to uživatel nemohl vědět. Dále na stránce vytvoření exportu počkal na statistiky a po jejich načtení export dokončil.

Uživatelův postup při zadání: Exportuj základní informace o odběratelích a jejich doručovací adresy

Uživatel rychle pochopil, že bude muset vytvořit novou šablonu pro export. Konfigurace šablony mu přišla velice intuitivní a po jejím dokončení byl schopen ji uložit a využít pro nový export.

Uživatelův postup při zadání: Importuj testovací soubor *Buyers*

Uživatel si nejprve zobrazil zadaný soubor a poté správně využil šablonu *buyers*. Nebylo mu hned jasné, že konfigurace šablony je pouze read only, poté ale pokračoval na poslední krok, kde nepočkal na statistiky a import odeslal ke zpracování.

Uživatelův postup při zadání: Importuj testovací soubor *TestFile*

Uživatel si opět nejprve zobrazil soubor a poté se rozhodl využít import bez šablony. Informace vyplnil bez problému a při konfiguraci se nejprve snažil na mapování využít zobrazení povinných atributů, poté ale pochopil mapování sloupců bez problému. Zarovnání hlavičky uživatel naprosto přehlédl a ignoroval. Primární klíče mu přišli nejasné, a tak se je rozhodl ignorovat. V posledním kroku tentokrát počkal na statistiky a po jejich načtení import dokončil.

Dotazník po Testu

1. **Jaké věci v novém rozhraní chyběli, nebo nebyli nejasné:**

   Uživatel by při konfiguraci šablony exportu spíše očekával, že kliknutí na sloupec ho do exportu přidá spíše, než odebere.

2. **Jaké věci se vám líbili:**

   Uživateli se líbilo mapování při konfiguraci importu.

3. **Je nové řešení lepší než váš aktuální postup:**

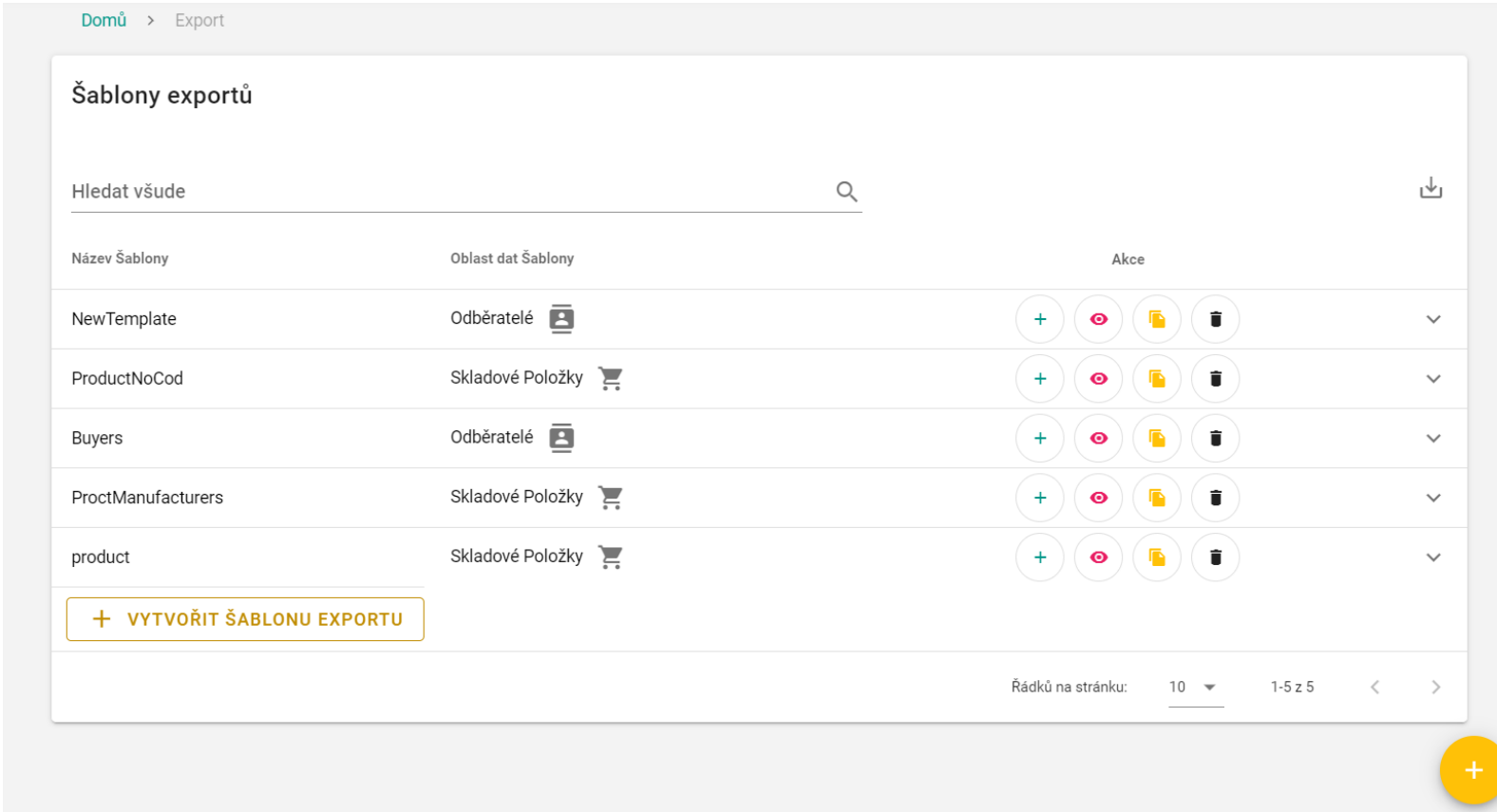
   Uživateli přijde nový export díky jeho konfigurovatelnosti, mnohem lepší než aktuální řešení.

4. **Používal/a by jste nové řešení, pokud by bylo v portálu k dispozici:**

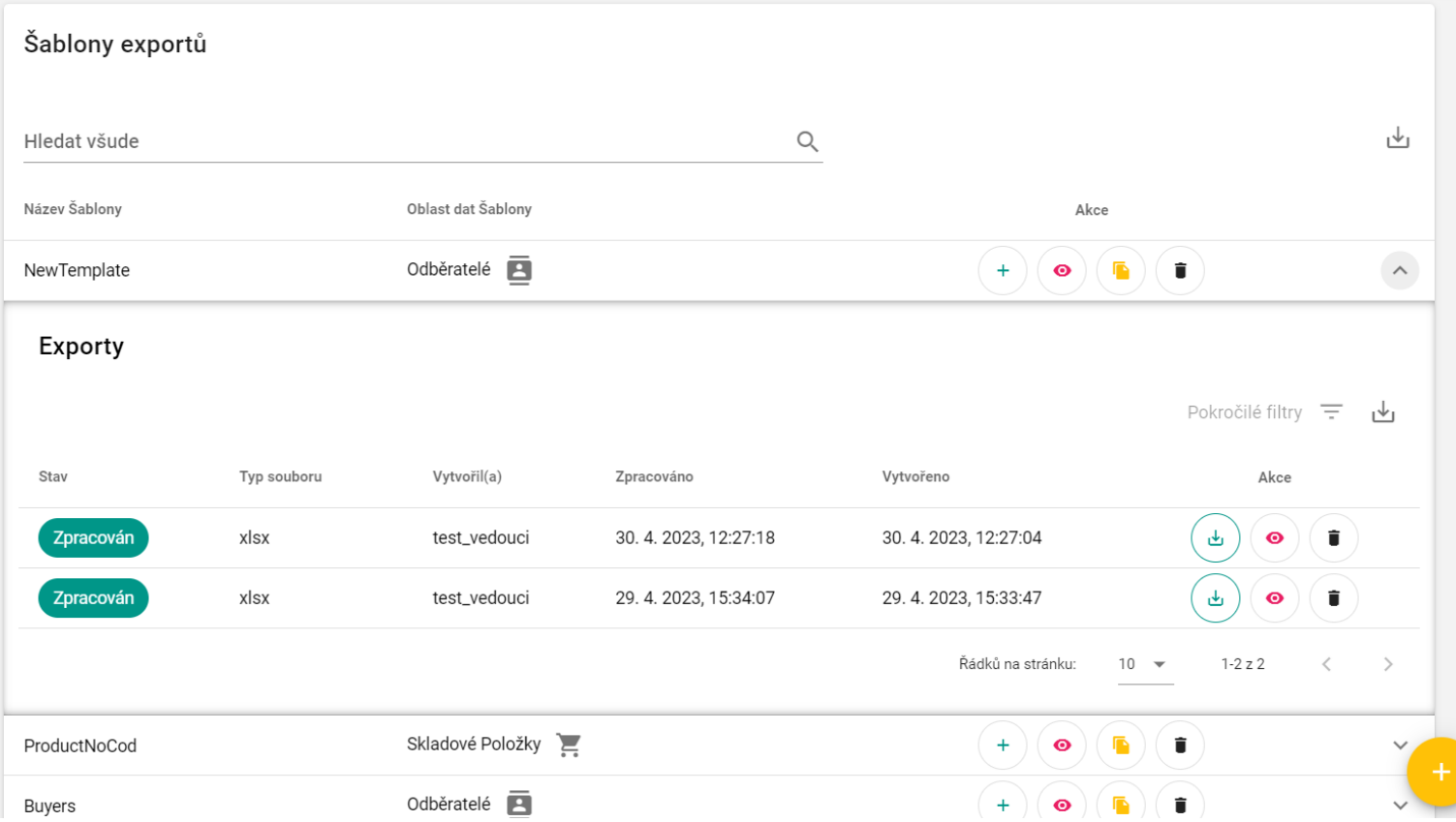
   Uživatel by nový import a export osobně nevyužil, jelikož se při vývoji pohybuje spíše na BE, ale u nových řešení vidí velké využití pro uživatele a ostatní vývojáře.

Příloha C

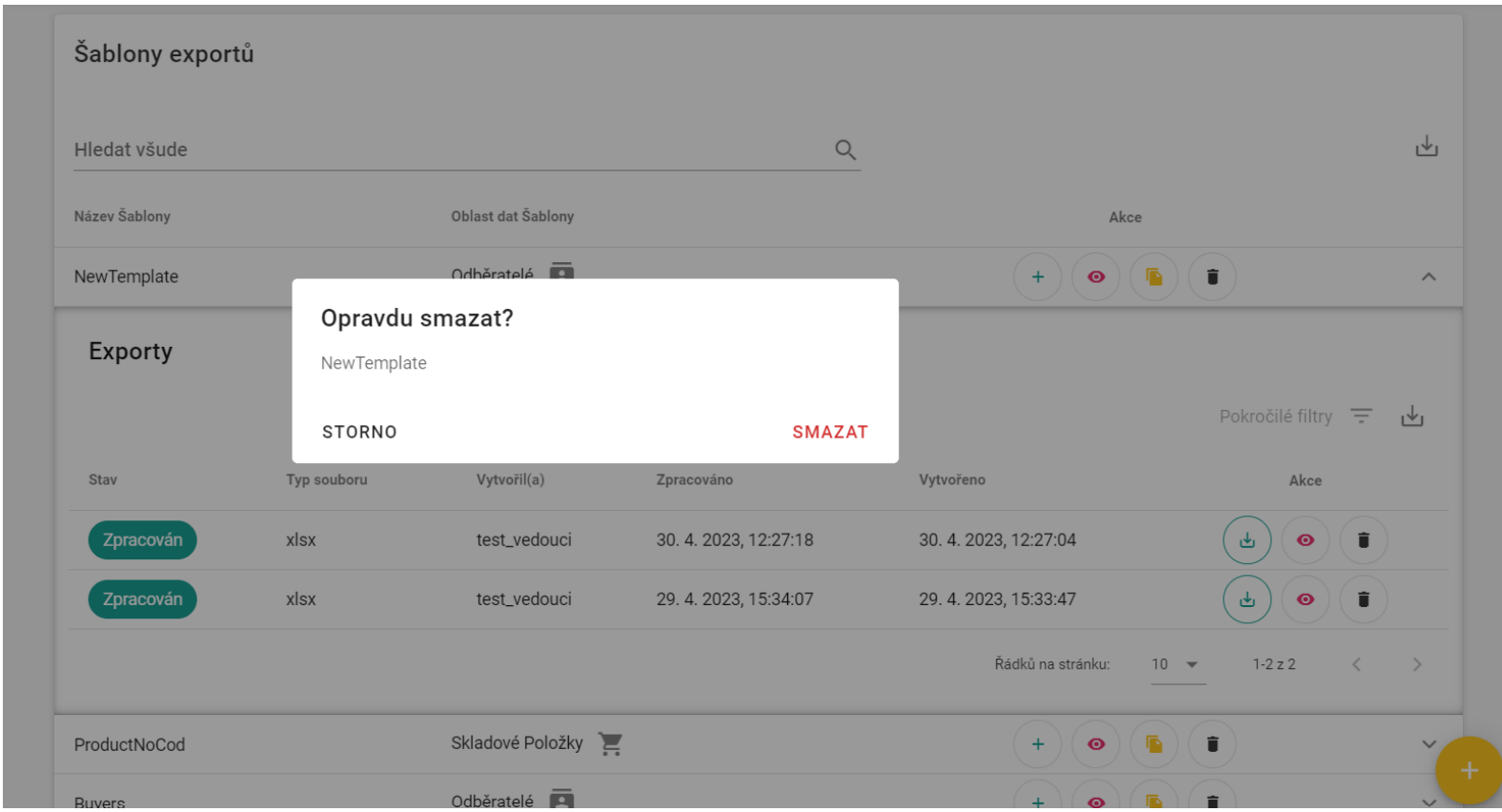
Finální vzhled rozhraní



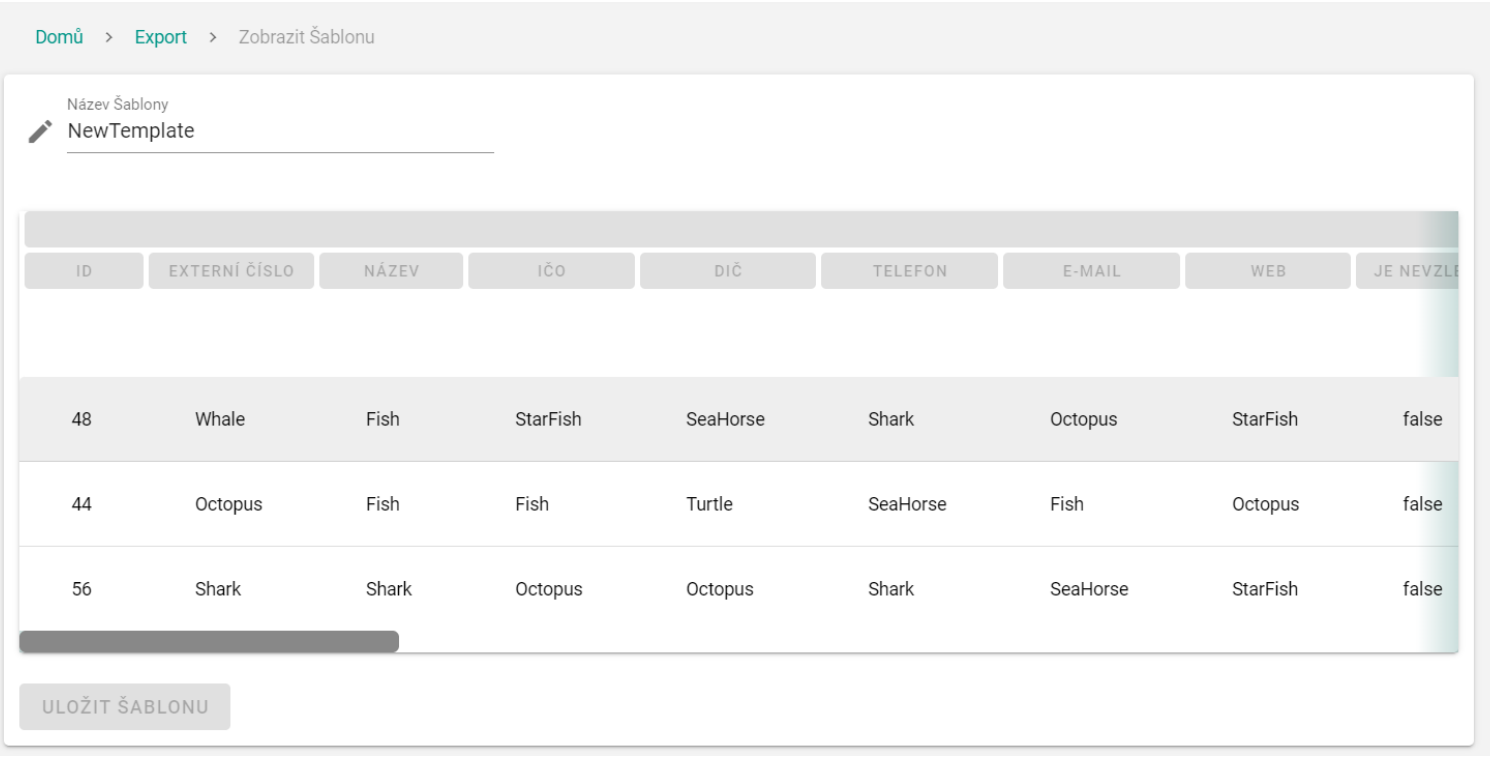
Obrázek C.1: Tabulka Šablon Exportu Zabalená



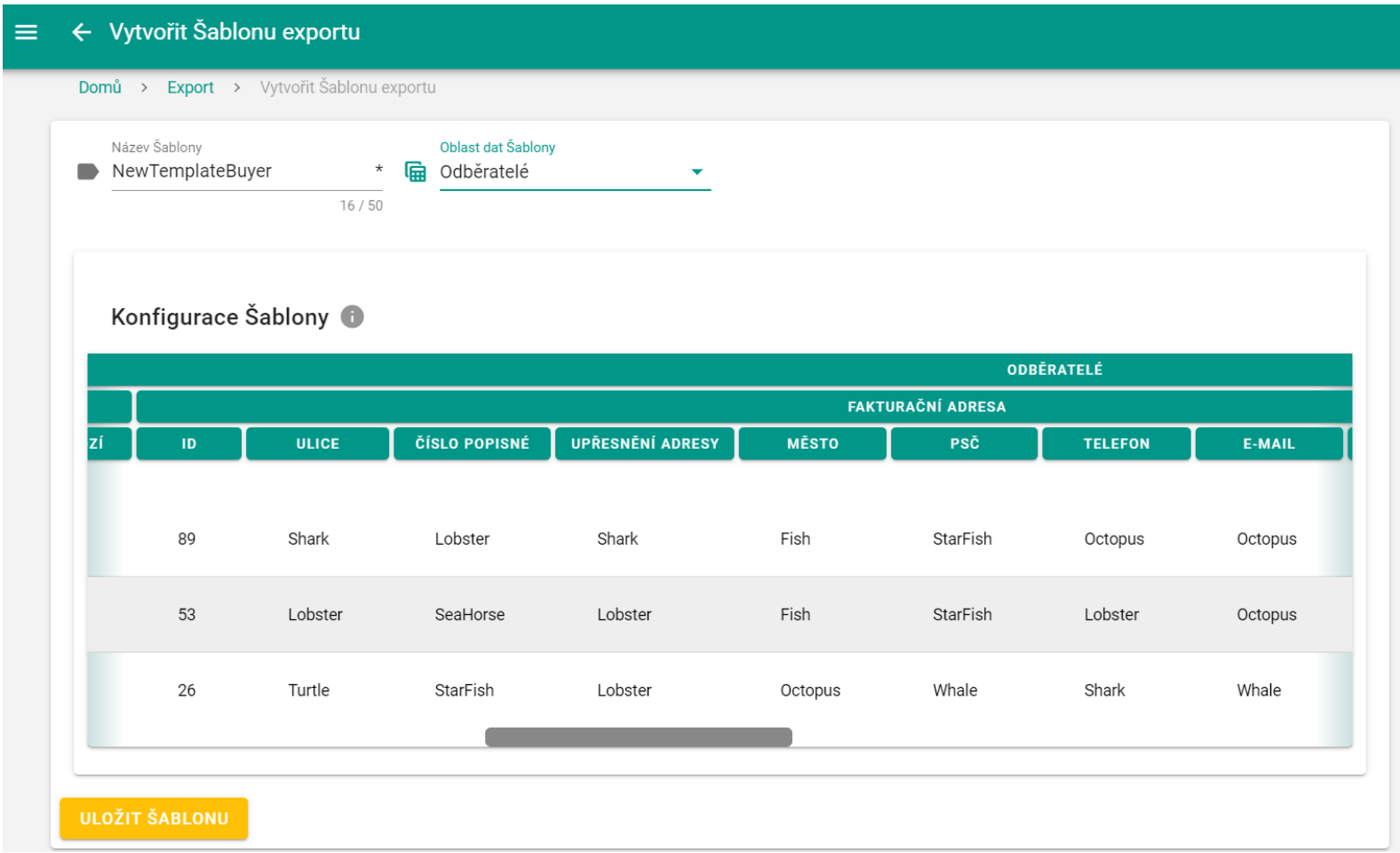
Obrázek C.2: Tavulka Šablon Exportu Rozbalená



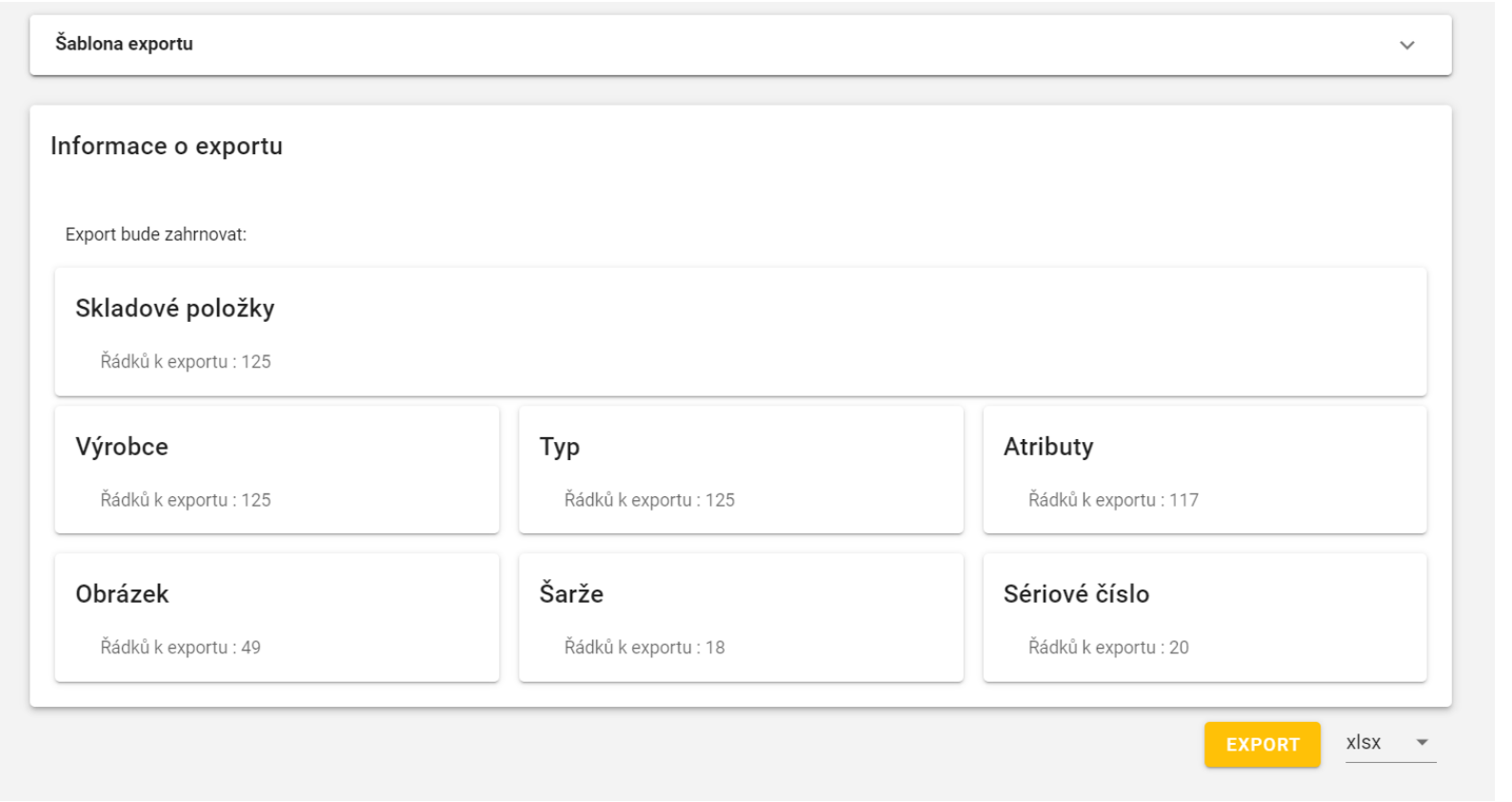
Obrázek C.3: Mazání Šablony exportu



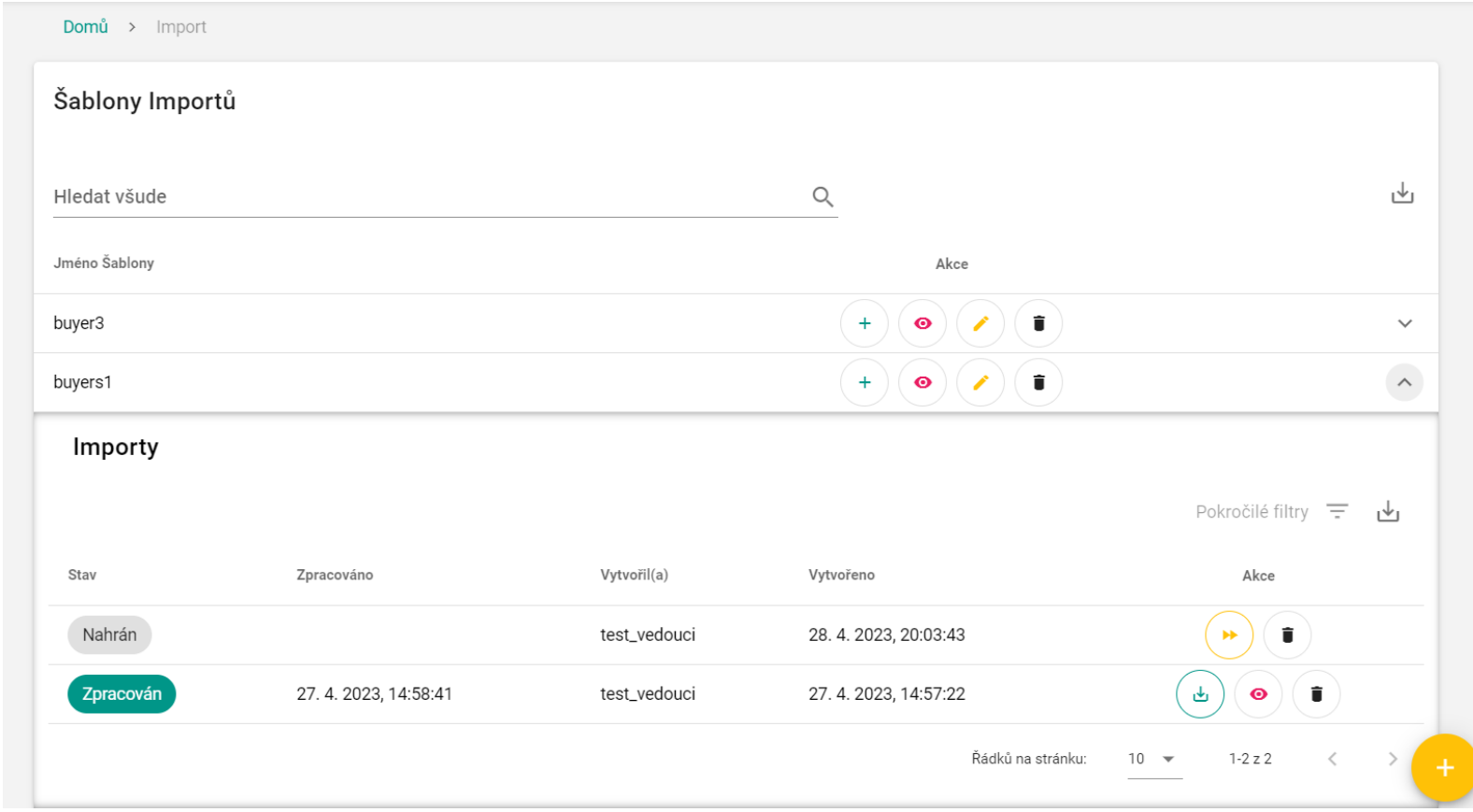
Obrázek C.4: Náhled na Šablonu Exportu



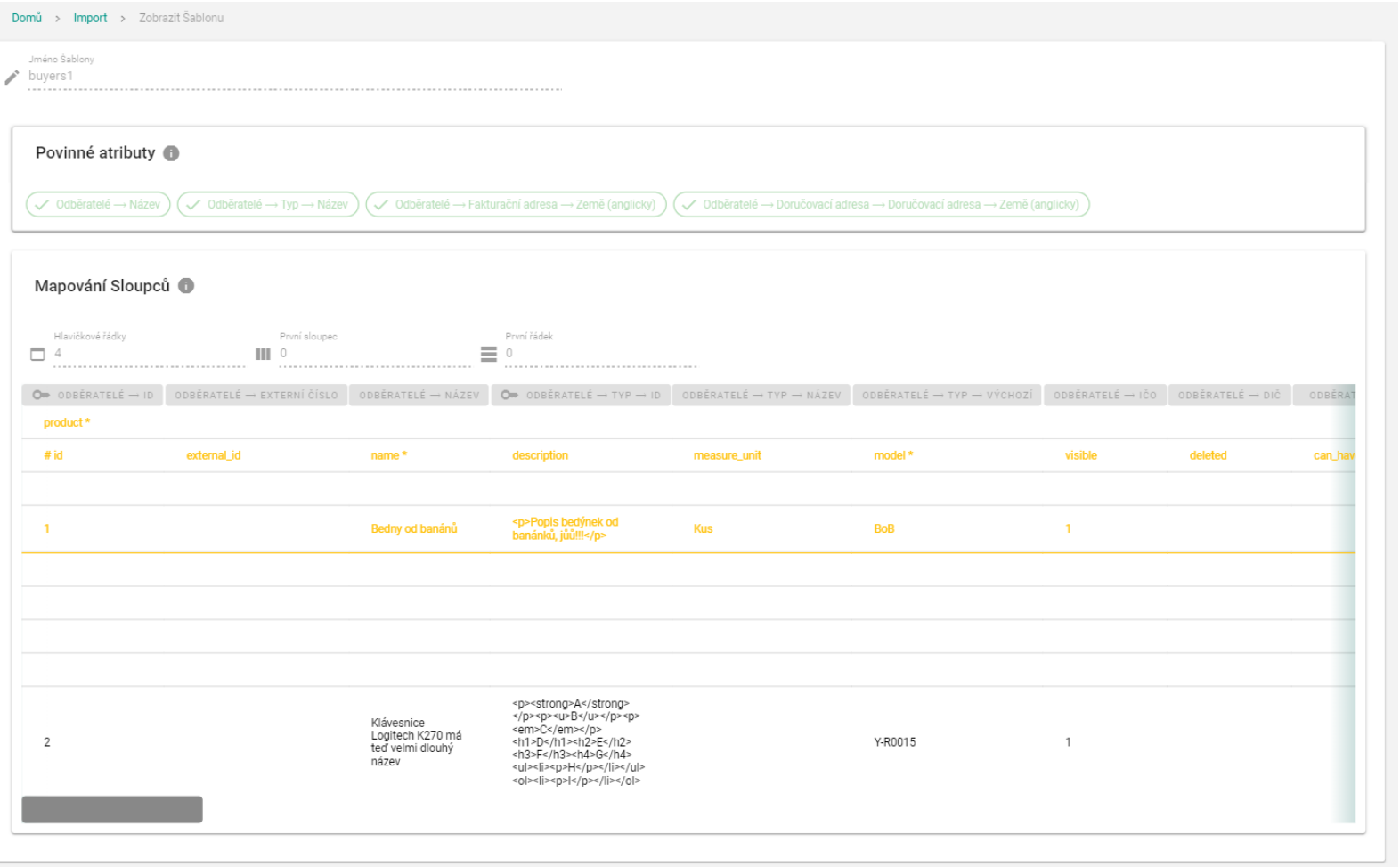
Obrázek C.5: Tvorba nové Šablony Exportu



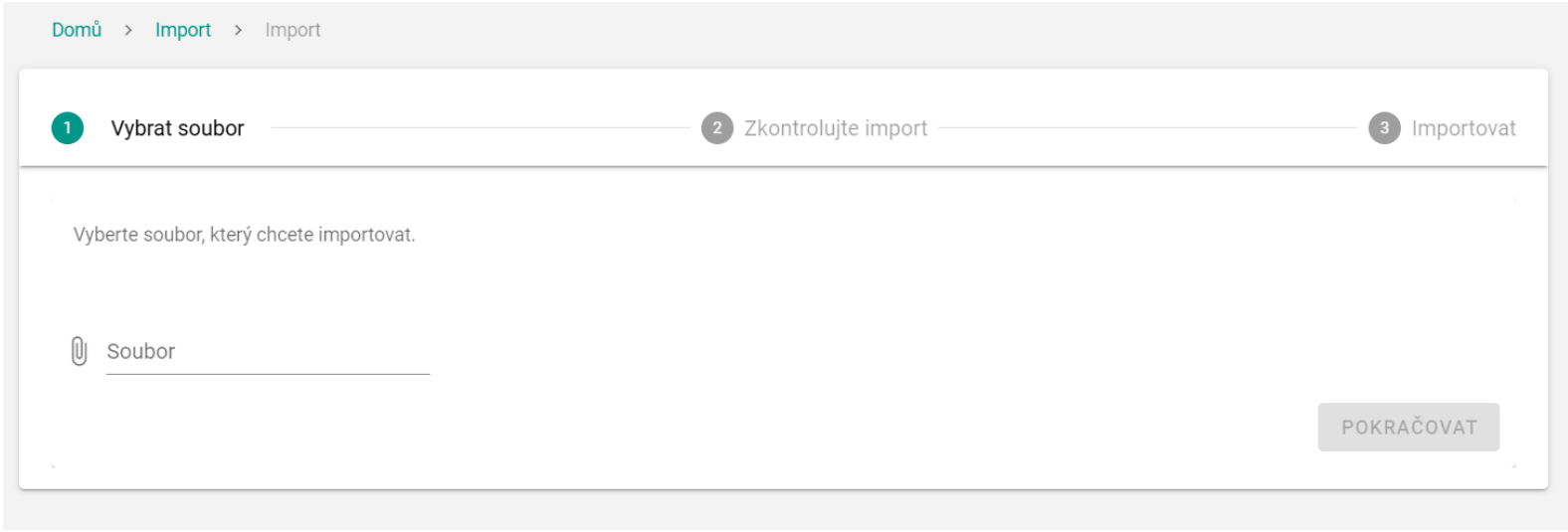
Obrázek C.6: Tvorba nového exportu



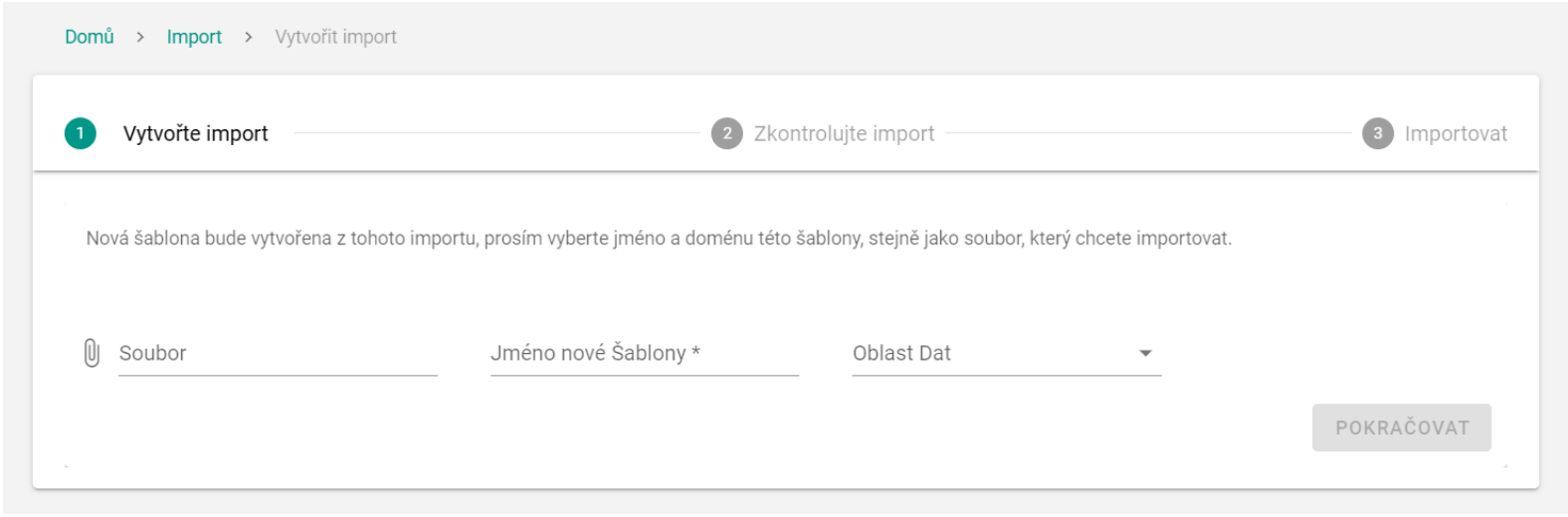
Obrázek C.7: Tabulka Šablon Importů rozbalená



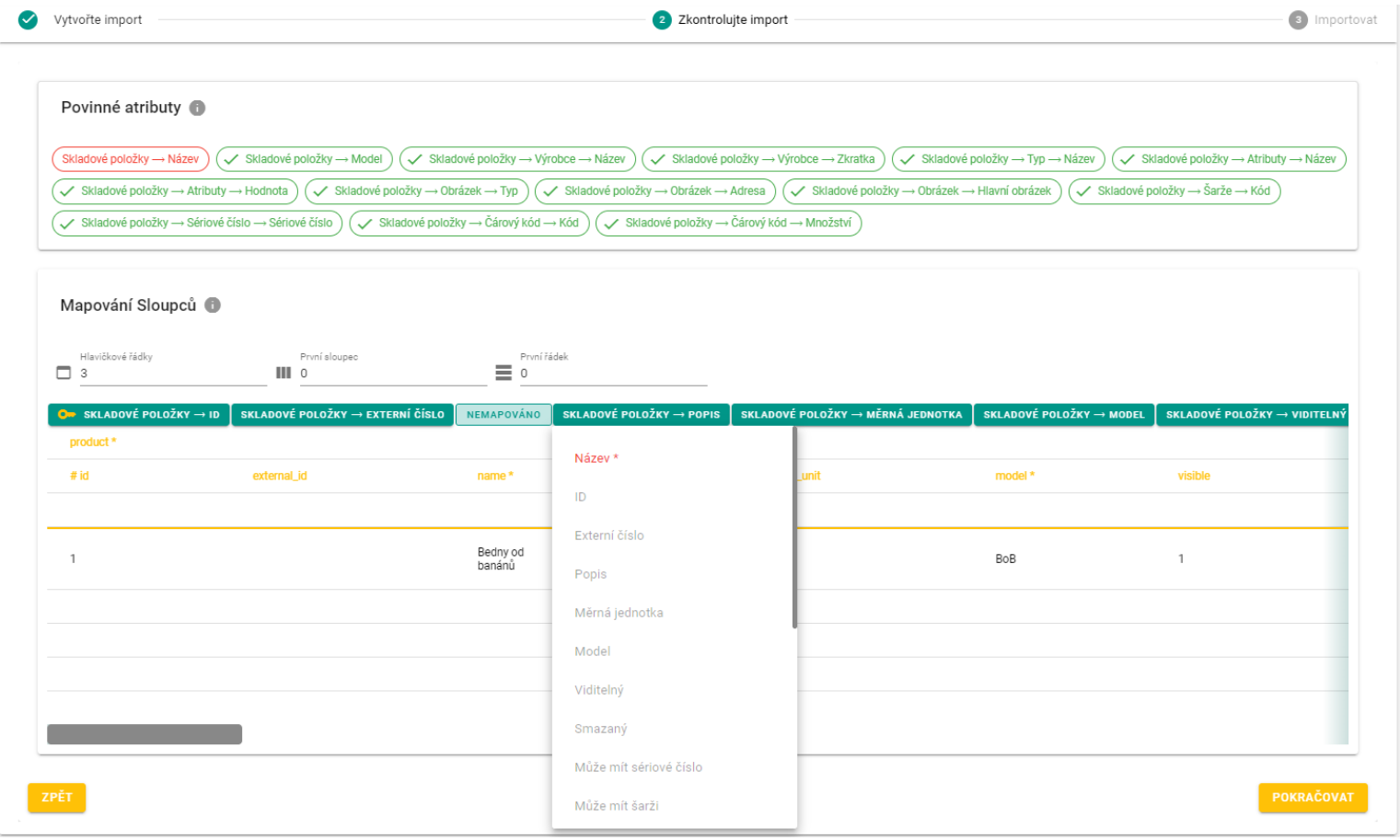
Obrázek C.8: Náhled Šablony importu



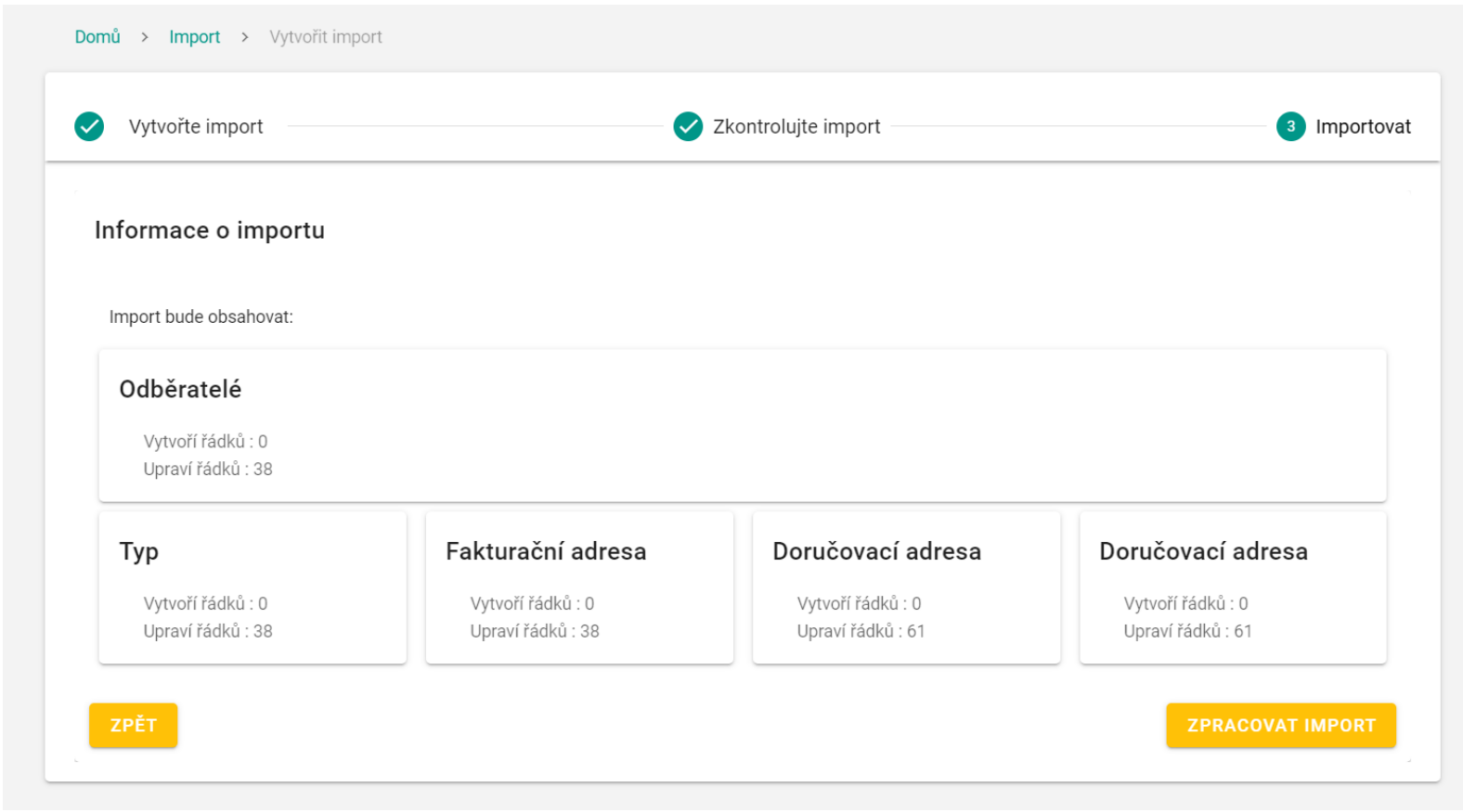
Obrázek C.9: První krok importu pomocí Šablony



Obrázek C.10: První krok importu bez Šablony



Obrázek C.11: Konfigurace Šablony importu



Obrázek C.12: Statistiky nového importu

Příloha D

Návrh v aplikaci Figma

Veškeré návrhy uživatelského rozhraní vytvořené v rámci této práce jsou k dispozici v piloženém médiu práce ve formátu pdf. Zde si tyto návrhy můžete také zobrazit v portálu Figma : `https://www.figma.com/file/LbSyDkrFDiBLyYQhgD9ZMs/Export-proof-of-concept?node-id=344%3A5127&t=hXcIR9qmqYw6PyX0-1`

Příloha E

Obsah přiloženého Média

- `README.md` ..... stručný popis obsahu média
- `src`
  - `latex/` ..... zdrojové soubory textu BP
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `UI_Navrh.pdf` ..... Návrhy uživatelského rozhraní ve formátu PDF