FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Frontend skladového systému
Student:	Bc. Oldřich Malec
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je kompletní tvorba frontendu skladového systému. Návrh nového frontendu bude zohledňovat (a rozšiřovat) funkcionalitu starého skladového systému Sysel od společnosti Jagu s.r.o. Při návrhu spolupracujte s Bc. Pavlem Kovářem, který bude připravovat serverovou část systému. Zpracujte kompletně minimálně roli skladníka a manažera skladu.

Postupujte dle následujících kroků:
1. Analyzujte všechny případy užití současného řešení a zjistěte, jaké nové funkce by měl systém také podporovat. Při řešení neopomeňte zhodnotit konkurenční řešení.
2. Analyzujte možnosti evidence logistiky - správy zboží připraveného k vyskladnění.
3. Na základě analýzy proveďte vhodný návrh, zaměřte se na efektivitu práce jednotlivých uživatelských rolí.
4. Návrh zrealizujte v podobě funkčního prototypu.
5. Prototyp podrobte vhodnými Vámi navrženými testy.
6. Na základě testování prototyp upravte.
7. Společně s Bc. Pavlem Kovářem zajistěte vydání alfa verze.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 13. února 2019

České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra softwarového inženýrství

Diplomová práce

Frontend skladového systému

Bc. Oldřich Malec

Vedoucí práce: Ing. Jiří Hunka

4. března 2019

Poděkování

TODO

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel licenční smlouvu o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 4. března 2019 .....

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

MALEC, Oldřich. *Frontend skladového systému.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019. Dostupný také z WWW: ⟨`https://gitlab.fit.cvut.cz/malecold/master-thesis`⟩.

Abstrakt

TODO

Klíčová slova TODO

Abstract

TODO

Keywords TODO

Obsah

Seznam ukázek kódu

Seznam tabulek

Úvod

TODO

Kapitola 1

Analýza požadavků

Kapitola 2

Návrh uživatelského rozhraní

Kapitola 3

Volba technologie

Jelikož bude aplikace rozdělena na backend, kterým se zabývá můj kolega Bc. Pavel Kovář, a frontend, který je předmětem této práce, je vhodné věnovat jistou část textu volbě vhodné technologie.

Cílová platforma Aplikace je navrhována s ohledem na hardwarové vybavení skladu, ve kterém bude poprvé nasazována: zdejší skladníci jsou vybaveni mobilními telefony *Zebra TC25BJ*, které disponují OS Android 7.1 a vestavěnou čtečkou čárových kódů. Kromě skladníků by měla být aplikace použitelná také z tabletu či stolního počítače pro účely vedoucích pracovníků. Z důvodu jednoduchosti vývoje, testování, možností aktualizací a obecně dobré zkušenosti z jiných projektů bylo hned při úvodním návrhu určeno, že aplikace bude tvořena formou webové služby, která bude na klientských zařízeních zobrazována ve WebView v jednoduchém kontejneru chovajícím se jako nativní aplikace. Řídící pracovníci budou naopak moci využít přístupu odkudkoliv, kde budou připojeni k internetu pouze pomocí běžného webového prohlížeče.
Z tohoto důvodu jsou v následující rešerši zhodnocovány frameworky či knihovny, které usnadňují vývoj *webových aplikací*.

3.1 Frameworky a knihovny

V době psaní této práce patří mezi nejpopulárnější [12] [18] front-endové frameworky či knihovny Angular [3], React [37], Vue.js [49], Ember.js [15] a Backbone.js [8].

Názvosloví Pro účely tohoto textu budu na následujících řádcích používat slovo *framework*, kterým budu označovat jak frameworky, tak knihovny, z důvodu snížení opakování textu.

3.1.1 Datum vydání

Zatímco v současnosti nejčastěji porovnávanými frameworky jsou první dva zmíněné, Vue.js je z této pětice vybraných nejmladší, nabírá ale velké obliby. Ember.js a Backbone.js jsou poté lehce upozaděny z důvodu jejich stáří. Přehled prvního vydání jednotlivých frameworků je v tabulce 3.1

	Angular	React	Vue.js	Ember.js	Backbone.js
Vydání první verze	2010/2016[a]	2013	2014	2011	2010

Tabulka 3.1: Volba frameworku: Datum vydání

[a]V roce 2010 byl vydán AngularJS, který byl v roce 2016 kompletně přepsán do TypeScriptu a vydán jako Angular 2, či jednoduše *Angular*.

Datum vydání ovšem nelze objektivně ohodnotit bodovým ziskem. Na jedné straně stojí fakt, že starší framework může být vyspělejší a tudíž stabilnější atp., na straně druhé nové frameworky se často učí z chyb provedených jejich předchůdci a vyberou z nich pouze to nejlepší. Tato tabulka tedy zůstane čistě přehledová.

3.1.2 Zázemí

Zatímco Angular a React jsou vyvíjeny velkými společnostmi: Googlem, respektive Facebookem, které zná každý, Ember.js je vyvíjen společností Tilde Inc. [42], která také není žádným startupem. Vue.js a Backbone.js by se naopak daly nazvat *komunitními projekty*, neboť jsou vytvořeny převážně jedním autorem (Evan You, respektive Jeremy Ashkenas) a rozvíjeny a udržovány komunitou vývojářů.
Na první pohled by se mohlo zdát, že z tohoto hodnocení budou vycházet lépe ty frameworky, které mají za sebou stabilní firmy, neboť je tím zajištěn jejich kontinuální vývoj. Ve skutečnosti ale velké firmy *zabíjejí* své projekty poměrně často, stačí se podívat například na seznam projektů, které ukončil Google [34].

Oproti tomu komunitní projekty mohou žít dále i v případě, že jejich hlavní autor už na projektu nechce, nebo nemůže pracovat. Z toho důvodu nelze jednoznačně určit, které zázemí je pro budoucnost frameworku výhodnější, a u tabulky 3.2 se tedy opět zdržuji udělování bodů.

	Angular	React	Vue.js	Ember.js	Backbone.js
Zázemí velké společnosti	ano	ano	ne	částečně	ne

Tabulka 3.2: Volba frameworku: Zázemí

3.1.3 Licence

Licence k použití frameworku je důležitá položka při rozhodování. Naštěstí všech 5 porovnávaných frameworků je v době psaní této licencováno pod MIT licencí, která povoluje jakékoliv použití i v komerční sféře, úpravy, distribuce i použití v ne-opensource projektech. Nevýhodou této licence je nulová záruka funkčnosti či zodpovědnost autorů za potenciální spáchané škody tímto softwarem.

Licencování Reactu Facebook původně vydal svůj React pod BSD licencí spolu s dalšími patenty, avšak 24. září 2017 byl React převeden pod MIT licenci [1, 47].
Jelikož jsou všechny frameworky licencovány stejně, neprobíhá v tabulce 3.3 žádné bodování.

	Angular	React	Vue.js	Ember.js	Backbone.js
Licence	MIT	MIT	MIT	MIT	MIT

Tabulka 3.3: Volba frameworku: Licence

3.1.4 Křivka učení

Složitost frameworku je důležitá metrika, neboť má dopady zejména na ekonomickou stránku projektu. Jednoduché prvotní vniknutí do problematiky frameworku ovšem také nemusí být nutně výhodou, pokud v něm je později problémové provést některé pokročilé věci, nebo i v pokročilém stádiu zdržuje svým

nízkoúrovňovým přístupem k problémům, které jiné frameworky řeší automaticky.

Angular, React a Vue.js Přehled obtížnosti tří v současnosti nejčastěji sklo-ňovaných frameworků přehledně shrnul Rajdeep Chandra ve své prezentaci *My experience with Angular 2 , React and Vue* [19], ze které vychází hodnocení v tabulce 3.4.

Ember.js Tento framework je dle V. Lascika [31] vhodný spíše pro projekty, na kterých pracuje velké množství vývojářů, a to z důvodu své komplexnosti. Proto jej pro použití v mé vznikající aplikaci hodnotím nula body.

Backbone.js U této knihovny je důležité zmínit, že umožňuje vývojáři vytvořit si strukturu aplikace kompletně dle svého uvážení [2]. To ssebou může nést jak výhody pro zkušeného, tak nevýhody pro nezkušeného vývojáře, který v pokročilém stádiu vývoje může zjistit, že některou ze základních struktur navrhl špatně. Samotná obtížnost práce s touto knihovnou je ale poměrně nízká.

	Angular	React	Vue.js	Ember.js	Backbone.js
Obtížnost	vysoká	vyšší	nízká	velmi vysoká[a]	nízká
bodový zisk	*1*	*2*	*4*	*1*	*4*

Tabulka 3.4: Volba frameworku: Obtížnost

[a]za předpokladu, že na projektu bude pracovat pouze velmi malé množství vývojářů

3.1.5 Oficiální dokumentace

Hlavním zdrojem ke studiu frameworku by měla být jeho oficiální dokumentace, v této sekci tedy budu hodnotit kvalitu a obsáhlost oficiálního manuálu k jednotlivým frameworkům.

Angular Jedná se o velmi obsáhlou a dobře rozdělenou dokumentaci [5], která obsahuje i řadu příkladů a ve srozumitelné stromové struktuře vývojář jednoduše najde, co potřebuje.

React Dokumentace Reactu [17] je o poznání jednodušší než ta Angularu, avšak to je způsobeno tím, že React je pouze knihovna, kdežto Angular je plnohodnotný framework. Dokumentace je rozdělena na jednodušší úvod a pokročilejší techniky, je tedy snadné s ní pracovat.

Vue.js Nejmladší z frameworků má také velmi přátelskou dokumentaci [22], která je podobně jako u Angularu velmi bohatá a stromově strukturovaná.

Ember.js Oficiální manuál Ember.js [16] je taktéž poměrně obsáhlý a strukturou připomíná dokumentaci Angularu a Vue.js. Obsahuje velké množství ukázek kódu a je logicky strukturován.

Backbone.js Poslední ze zkoumaných frameworků má oficiální dokumentaci [9] na první pohled méně atraktivní a pro nováčka může být matoucí. Oproti ostatním dokumentacím chybí například barevné zvýraznění důležitých bodů a další grafické strukturování textu.

	Angular	React	Vue.js	Ember.js	Backbone.js
Kvalita oficiální dokumentace	velmi vysoká	vysoká	velmi vysoká	velmi vysoká	střední
bodový zisk	*3*	*2*	*3*	*3*	*1*

Tabulka 3.5: Volba frameworku: Dokumentace

3.1.6 Testování

Testování je při vývoji softwaru velkým tématem. Mnoho vývojářů nerado testuje, jiní se naopak specializují pouze na testování. V moderních aplikacích je testování z větší části řešeno automatizovanými testy, které se spouští v rámci CI. Přestože se tato práce zabývá pouze frontendem, i ten je možné testovat již od úrovně Unit testů, vyvíjej jej pomocí TDD a nakonec samozřejmě vše zkontrolovat pomocí E2E testů.

Angular Nejkomplexější z frameworků nabízí přehlednou dokumentaci [4], jak by měl být testován. Popisuje jak používat Unit testy i E2E testy a pro druhý zmíněný typ nabízí i samotný testovací framework: Protractor [36]. Jeho testovatelnost tedy hodnotím jako velmi dobrou.

React Tato knihovna ve své dokumentaci na první pohled testování příliš neprosazuje, avšak stránku dedikovanou této kratochvíli [41] lze nakonec také nalézt. Narozdíl od Angularu zde není poskytován dedikovaný framework určený přímo pro testování této knihovny. Stránka popisuje testovací frameworky, které používají různé společnosti: autoři Reactu - Facebook - zmiňují svůj Jest [28], také je ale zmiňován Enzyme [20] od Airbnb. Konkrétní příklady použití bychom tedy dále hledali v dokumentacích těchto frameworků. Celkově je pro React k dispozici více různých testovacích frameworků, které opět pokrývají jak Unit, tak E2E testování.

Vue.js Ani Vue.js se neztratí co se Unit a E2E testů týká. Přímo ve své dokumentaci [43] popisuje spouštění Unit testů pomoci vestavěných příkazů, které interně používají buďto již zmiňovaný Jest nebo Mocha [32], a dále odkazuje na vlastní kompletní testovací knihovnu [24]. Také je možné použít velké množství různých testových frameworků a aplikace ve Vue.js lze vyvíjet i pomocí TDD [30].

Ember.js Tento framework se ve své dokumentaci věnuje testování poměrně intenzivně a zasvětil mu rovnou několik stránek [21]. Jako výchozí testovací framework představuje vlastní QUnit, avšak informuje, že je možné použít i frameworky třetích stran. Testování přehledně rozděluje na *Unit*, *Container*, *Render*, a *Application* testy a také popisuje jak testovat jednotlivé komponenty. Testování Ember.js tedy také hodnotím jako velmi dobré.

Backbone.js U posledního zkoumaného frameworku na první pohled není testování moc jasné. Oficiální stránka odkazuje na *test suite* [7], což je ovšem stránka, která testy *spouští* a ne popisuje. Zdá se, že na této stránce je možné otestovat svůj prohlížeč, zda podporuje veškeré funkcionality Backbone.js. Co se psaní testů týká, rozumněji již vypadá externí stránka *Backbone.js Testing* [38], která již zmiňuje i zde dříve probírané frameworky jako *Mocha*. Ve výsledku tak bude pravděpodobně testování Backbone vypadat obdobně jako u ostatních frameworků, avšak přístup oficiální dokumentace snižuje jeho pochopení.

	Angular	React	Vue.js	Ember.js	Backbone.js
Možnosti testování a jejich popis v dokumentaci	velmi kvalitní	velmi kvalitní	velmi kvalitní	velmi kvalitní	matoucí
bodový zisk	*2*	*2*	*2*	*2*	*1*

Tabulka 3.6: Volba frameworku: Testování

3.1.7 Vývojářské nástroje

Dalším důležitým nástrojem při práci s frameworkem je možnost jeho debuggování. Framework by měl nabízet vlastní řešení, které vývojáři usnadní nalézt chybu, zjistit, jak se jeho kód chová či odladil rychlostní problémy.
Všechny ze zde porovnávaných frameworků nabízejí tyto nástroje formou doplňku do prohlížeče, konkrétně se dále budeme bavit o doplňcích pro Google Chrome.

	Angular	React	Vue.js	Ember.js	Backbone.js
Název	Augury	React Developer Tools	Vue.js devtools	Ember inspector	Backbone Debugger
Počet stažení[a]	230k	1.351k	706k	57k	9,5k
bodový zisk	*2*	*4*	*3*	*1*	*0*
Hodnocení (z 5)	3.9	4.2	4.7	4.8	4,5
bodový zisk	*2*	*1*	*3*	*4*	*2*

Tabulka 3.7: Volba frameworku: Devtools

[a] Stav k 24. 12. 2018

3.1.8 Počet hvězdiček na GitHubu

Počet hvězdiček na GitHubu lze velmi volně interpretovat jako oblíbenost frameworku mezi vývojáři. Z tohoto důvodu již v tabulce 3.8 hodnotím frameworky dle počtu získaných hvězdiček.

	Angular	React	Vue.js	Ember.js	Backbone.js
Počet hvězdiček na GitHubu[a]	43,6k	117,7k	122,3k	20,3k	27,3k
bodový zisk[b]	*2*	*4*	*5*	*0*	*1*

Tabulka 3.8: Volba frameworku: Počet hvězdiček na GitHubu

[a] Stav k 17. 12. 2018

[b] Hodnocení přeskakuje bodový zisk 3, aby bylo zhodnoceno i absolutní množství hvězdiček, nejen pořadí.

3.1.9 Počet npm balíků

Npm [10] je repozitář javascriptových komponent, na kterém jsou sdíleny jednak kompletní řešení (jako například Angular, Rect, Vue.js a další), ale především různé rozšiřující pluginy do těchto frameworků. Z toho důvodu budu v následující metrice hodnotit, kolik balíků npm nabízí pro jednotlivé porovnávané frameworky.
Bodové zisky hrubě odpovídají relativnímu počtu nalezených balíků.

	Angular	React	Vue.js	Ember.js	Backbone.js
Počet npm balíků	26,6k	73,4k	20,6k	6,4k	1,5k
bodový zisk	*2*	*3*	*2*	*1*	*0*

Tabulka 3.9: Volba frameworku: Počet npm balíků

3.1.10 Otázky na Stack Overflow

Stack Overflow je jedním z portálů sítě Stack Exchange, který zná prakticky každý vývojář. Kdokoliv zde může položit otázku a komunita poté odpovídá, zatímco hlasuje o kvalitě odpovědí, aby byla vybrána ta nejlepší.
Z pohledu volby frameworku může být na jednu stranu vhodné, aby bylo na této stránce hodně otázek týkajících se dané technologie, na druhou stranu to ale může znamenat i nekvalitní dokumentaci. Jelikož ale v předchozí sekci nebyla žádná dokumentace vyhodnocena jako vysloveně špatná, budu dále usuzovat, že větší množství otázek je lepší.

	Angular	React	Vue.js	Ember.js	Backbone.js
Počet otázek na Stack Overflow	146k	118k	28k	23k	21k
Počet *zodpovězených* otázek	86k	72k	18k	17k	16k
bodový zisk	*3*	*3*	*1*	*1*	*1*

Tabulka 3.10: Volba frameworku: Otázky na Stack Overflow

3.1.11 Firemní stack

Další zvolenou metrikou je, jak daná technologie zapadá do firemní stacku firmy Jagu s.r.o., ve které bude tento nový projekt realizován. Firma se specializuje především na webové aplikace a middlewary na zakázku [25], a mezi nejpoužívanější technologie patří PHP (Nette, Laravel, Symfony), dále provozuje jeden informační systém postavený na Angularu a nově také menší aplikaci ve Vue.js. Tabulka 3.11 shrnuje, jak jsou jednotlivé frameworky blízko k tomuto stacku.

	Angular	React	Vue.js	Ember.js	Backbone.js
Shoda s firemním stackem	ano	ne	ano	ne	ne
bodový zisk	*2*	*0*	*2*	*0*	*0*

Tabulka 3.11: Volba frameworku: Shoda s firemním stackem

3.1.12 Dostupnost vývojářů

Metrikou, kterou z hlediska udržitelnosti projektu a jeho ekonomických nákladů nelze opomenout, je dostupnost a cena vývojářů se zájmem o danou technologii.
Tato data se ale obtížněji získávají, většina statistik hovoří o nabídkách práce v dané technologii, nikoliv o počtu lidí, kteří s ní pracují. Z toho důvodu jsem se rozhodl založit tuto metriku na výsledcích vyhledávání osob v profesní síti LinkedIn - tak dokážeme zjistit alespoň hrubý počet lidí, kteří o sobě sami tvrdí, že jsou vývojáři v daném frameworku.
Bodové zisky zde hrubě reflektují relativní počet nalezených profilů.

	Angular	React	Vue.js	Ember.js	Backbone.js
Počet výsledků na dotaz "<název> developer"	344k	333k	78k	21k	76k
bodový zisk	*4*	*4*	*2*	*1*	*2*

Tabulka 3.12: Volba frameworku: Počet vývojářů na LinkedIn

3.1.13 Integrace se Sentry

Sentry [40] je nástroj sloužící k automatickému i manuálnímu záznamu chyb v aplikacích. Ve firmě Jagu s.r.o. je využíván v řadě projektů a jeho nasazení bude vhodné i pro aplikaci řešenou v rámci této práce. Z toho důvodu je vhodné se podívat, jak hlubokou integraci je možné mezi jednotlivými frameworky a Sentry realizovat.
Při pohledu na přehled toho, jaké technologie Sentry podporuje v JavaScriptu [27] rychle zjišťujeme, že všech pět zde zkoumaných frameworků je oficiálně podporováno, včetně rychlého návodu na zprovoznění. Z toho důvodu neprobíhá v tabulce 3.13 žádné bodování.

	Angular	React	Vue.js	Ember.js	Backbone.js
Oficiální integrace se Sentry	ano	ano	ano	ano	ano

Tabulka 3.13: Volba frameworku: Integrace se Sentry

3.1.14 Souhrn průzkumu

V tabulce 3.14 jsou sečteny body z předchozích dílčích hodnocení.

	Angular	React	Vue.js	Ember.js	Backbone.js
bodový zisk celkem	23	25	27	14	12

Tabulka 3.14: Volba frameworku: Výsledky

Výsledky rozdělují frameworky na dvě skupiny. V té vedoucí je trojice Angular, React a Vue.js, v pozadí poté zůstávají Ember.js a Backbone.js. Na tomto místě je také vhodné znovu zmínit, že hodnocení frameworků probíhalo ve vztahu ke konkrétnímu projektu, který je cílem této práce, a také ve vztahu k firmě

Jagu s.r.o., která vývoj této aplikace zaštiťuje.
První tři frameworky jsou seřazeny poměrně těsně za sebou, avšak nejlépe vyšel ze srovnání nejmladší Vue.js, který tímto volím jako framework, ve kterém budu na práci dále pracovat.

Kapitola 4

Implementace

4.1 Příprava kostry aplikace

Pro instalaci základu projektu jsem vycházel z dokumentace [22] a využil NPM [10]. Tím jsem získal základní kostru aplikace, kterou je možné spouštět ve vývojářském režimu (ten podporuje například hot-swapping, ale pro produkční prostředí se nehodí, neboť není rychlostně optimalizován). Nástroje pro tvorbu optimalizované verze jsou také samozřejmě dostupné - zatím ale nejsou pro mé použití potřeba.

Jelikož se zabývám aplikací středních rozměrů, je vhodné k tomuto základu přidat některé pokročilé funkcionality, které popíšu v následujících sekcích.

4.1.1 Router

Jednou z prvních komponent, kterou jsem k čistému Vue.js přidal, byl oficiální Vue Router [23]. Jelikož Vue.js je zaměřeno na tvorbu SPA - Single Page Application, bez konfigurace routeru by se všechny obrazovky aplikace zobrazovaly pouze na URL `/` - pokročilý uživatel by tak nemohl zadat například adresu `/task/2345`, což je vhodné i při ladění chyb - tester může jednoduše poslat zprávu *„na této adrese je špatně dodavatel“*.

Podobně jako chceme mít hezké URL, pro uživatele, který bude aplikaci ovládat z prohlížeče (tedy ne z aplikace), je žádoucí nastavovat správné titulky stránek. K tomu sice ve Vue Routeru neexistuje nativní podpora, ale řešení je opravdu

snadné - je popsáno v jednom z *Issues* v repozitáři na GitHubu [48] a spočívá v doplnění titulků do meta atributů cest:

```
{
    path: '/',
    component: Homepage,
    meta: {
        title: 'Swordfish'
    }
}
```

Ukázka kódu 4.1: Nastavování titulků stránek pomocí Vue routeru - úprava definic

a dále úpravě samotné instance routeru:

```
router.beforeEach((to, from, next) => {
    document.title = to.meta.title;
    next();
})
```

Ukázka kódu 4.2: Nastavování titulků stránek pomocí Vue routeru - úprava instance routeru

4.1.2 Webpack

Webpack [45] je software, který zpracovává součásti webových aplikací a tvoří z nich balíčky vhodné pro webové prohlížeče. Primárně je zaměřen na Javascript, ale dokáže zpracovávat i řadu dalších formátů, přes styly v css či sass, obrázky v png, jpeg, svg či konfigurace v json, yaml a dalších.
Primárním důvodem, proč používat Webpack, je možnost rozdělování kódu do jednotlivých souborů. Z jiných programovacích jazyků jsme zvyklí mít oddělenou komponentu starající se o přihlašování, v dalším souboru mít podporu komunikace s externím API atp. Samotný Javascript sice samozřejmě umožňuje kód rozdělit do více souborů, avšak vše se spojuje pouze do jednoho kontextu prohlížeče, a tak může u větších projektů být matoucí, odkud se ta která závislost vlastně bere.

Použití loaderů Další příležitost k použití webpacku přichází ve chvíli, kdy chceme v projektu mít kód, který není v cílovém prohlížeči podporován. Tím může být například SASS, nebo třeba i moderní syntaxe Javascriptu řídící se standardem ES6 či novějším. S pomocí Webpacku lze nastavit, aby se při zpracování některých assetů použil *loader*, který například převede SASS na CSS, konstrukce ES6 na ES5 apod. [26].

Webpack ve spolupráci s Vue.js Základní instalace Vue.js využívá automaticky Babel [6] - to je kompilátor moderních verzí JS do starších, s kterými jsou kompatibilní prohlížeče. Pokud ale chceme využít pokročilé možnosti webpacku, je potřeba vytvořit soubor `webpack.config.js` kam zaneseme běženou konfiguraci webpacku - tedy seznam assetů, které budou zpracovávány a jednotlivé lodery, které je mají zpracovat.

4.1.3 Vue-CLI

TODO

4.1.4 Proměnné prostředí

TODO

4.1.5 Sentry

Sentry [40] je webová služba pro sledování chyb, které v aplikaci nastanou. Při použití v produkčním prostředí může vývojář díky Sentry o chybě vědět ještě dříve, než ji uživatel nahlásí, a to včetně všech detailů, jako například jaké kroky chybě předcházely, prostředí, ve kterém k chybě došlo a mnoho dalších.
Integrace s Vue.js je přímo podporována, a tak je integrace se Sentry záležitostí několika řádků kódu. V ukázce kódu 4.3 je vidět třída, kterou v projektu používám.

```
import Vue from 'vue'
import Raven from 'raven-js';
import RavenVue from 'raven-js/plugins/vue';
import Env from '@/service/Environment'

class SentryClient {

    #client = undefined;

    constructor() {
        if (Env.isProduction()) {
            // When this is enabled, there is **no console output**
            this.#client = Raven.config('<url>')
                .addPlugin(RavenVue, Vue)
                .install();
        }
    }

    captureMessage(msg, options) {
        if (Env.isProduction() && this.#client !== undefined) {
            this.#client.captureMessage(msg, options);
        }
    }
}

const Sentry = new SentryClient();

export default Sentry;
```

Ukázka kódu 4.3: Třída pro zasílání aplikačních chyb do Sentry

Používání `class` a `#` v JS kódu V ukázce kódu 4.3 je použito klíčové slovo `class`, což je konstrukce představena až ve standardu ECMAScript 2015 [11] a výsledný kód se chová totožně jako třídní objekty z jiných jazyků. Dále se uvnitř třídy vyskytuje proměnná začínající symbolem # - zde se jedná o *proposed feature* [14] do budoucí verze ECMAScript a jedná se o privátní proměnné - opět tak jak je známe z jiných jazyků.

Samotné zavedení třídy z ukázky kódu 4.3 zajistí, že veškeré chyby, které vzniknout v produkčním prostředí, jsou automaticky zaslání do Sentry včetně veškerých detailů. Občas je ale vhodné odeslat i uživatelskou zprávu, která nemusí nutně být chybová, může jít pouze o informativní zprávu. Sentry client toto umožňuje pomocí metody `captureMessage`, kterou výše zmíněné třída lehce obaluje. Použití v projektu je zaznamenáno v ukázce kódu 4.4.

```
import Sentry from "@/service/Sentry";

...

Sentry.captureMessage('Your message');
```

Ukázka kódu 4.4: Zasílání vlastních zpráv do Sentry

4.1.6 Podpora WebApp

V kapitole 3 jsem zmiňoval, že skladník bude aplikaci používat z nativní Android aplikace, která bude obalovat WebView, a vedoucí skladu si aplikaci otevře v běžném browseru - pro obě použití by konfigurace WebApp nebyla potřeba, avšak pokud by někdo nepotřeboval čtečku čárových kódu - což je jediný důvod, proč skladníci používají jako základ nativní Android aplikaci, je možné si otevřít na mobilu běžnou stránku a použít volbu ”Přidat na plochu”. Tím vznikne zástupce, který zobrazuje *favicon* webové stránky a po jehož otevření se opět otevře běžný webový prohlížeč.
Pokud je však na stránce definovaný `manifest.json` pro webové aplikace, může se po otevření tohoto zástupce otevřít stránka v režimu celé obrazovky, a to případně i se skrytými ovládacími prvky - vše tak vypadá, jako kdyby se jednalo o nativní aplikaci. Základní konfigurace tohoto manifestu je vidět v ukázce kódu 4.5.

```
{
  "short_name": "Swordfish",
  "name": "Swordfish - brána do skladového systému Atlantis",
  "icons": [
    {
      "src": "favicon/swordfish-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "start_url": "/",
  "display": "fullscreen",
  "orientation": "portrait"
}
```

Ukázka kódu 4.5: Manifest pro webové aplikace

TODO screenshot rozdílu?

4.1.7 Překlady

Novou aplikaci je dnes vhodné hned od počátku psát jako *multijazyčnou* - jako základ tedy například v češtině a angličtině.
Pro překlady Vue.js aplikací je vhodné použít knihovnu `vue-i18n`[1]. [29]
Použití knihovny je pak obdobné jako známe z jiných jazyků. Máme definované klíče, které mohou být i vnořené (viz ukázka kódu 4.6) a ty následně používáme v šabloně (viz ukázka kódu 4.7)

```
{
    close: "Zavřít",
    home: "Domů",
    notFound: "Nenalezeno",
    lang: {
        switch: "Switch to English",
        switchDone: "Jazyk: čeština"
    }
}
```

Ukázka kódu 4.6: Definice překladů pro i18n

```
<div>
    <h1>{{ $t("base.notFound") }}</h1>
    <router-link to="/">{{ $t("base.home") }}</router-link>
</div>
```

Ukázka kódu 4.7: Použití překladů v i18n

Bude-li v budoucnu někdy potřeba přidat další jazyky, stačí vzít pouze definice překladů dle klíčů a přeložit vše do nového jazyku. Zde je vhodné ještě zmínit, že pouhé překládání podle klíčů je sice rychlé, ale i tak by po nasazení nového jazyka měly být překlady zkontrolovány překladatelem přímo v aplikaci, a to navíc takovým, který rozumí cílové doméně. Pokud se tak neučiní, nastává často situace, kterou vídáme u zahraničních služeb, které expandují do Česka:

[1] název je zkratka pro *internationalization* - číslovka 18 značí počet přeskočených znaků

překlady jsou dělané buďto strojově a nebo pouze dle klíčů, a v aplikaci se pak zobrazují slova, která bychom my jako Češi nikdy v daném kontextu nepoužili.

4.1.8 Material design

Nejrozšířenější grafickou knihovnou pro tvorbu webových a mobilních aplikací pro Android je bezesporu *Material Design* od Googlu.
První knihovna, na kterou jsem narazil, byla Vue Material [33]. Bohužel jsem záhy zjistil, že jelikož v době psaní práce (březen 2019) je teprve v beta verzi, a nemá implementované všechny komponenty, nakonec jsem se rozhodl pro její alternativu - Vuetify.
Jedná se o aktivně vyvíjenou knihovnu, která již podporuje veškeré základní prvky Material Designu a stále jsou přidávány i prvky nové [44].
Vuetify lze navíc jednoduše integrovat s již použitou knihovnou i18n.js popisovanou v předchozí sekci. TODO rozvést

4.1.9 State management pattern

Jako první se zde sluší říci, co to vlastně *State management pattern* je. Ve Vue.js aplikaci máme typicky velké množství komponent, a ty mezi sebou často potřebují komunikovat. Pěkný příklad je například *snackbar message*, někdy nazývaná také *toast message* či jednoduše *oznámení o provedení akce*. To je komponenta, která musí být dostupná z jakékoliv jiné komponenty systému a vždy se musí zobrazovat na stejném místě a stejně se chovat. Je tedy žádoucí zpřístupnit její *stav* a to tak, aby se při změně stavu něco automaticky stalo, a aby změna stavu byla řízena jistými pravidly.

Vuex Vuex [46] je knihovna, která implementuje *State management pattern* pro Vue.js. Zařizuje jednotný přístup ke stavům komponent a umožňuje jejich řízené změny. Jeho použití v aplikaci skladového systému je vidět na ukázce kódu 4.8, 4.9 a 4.10, kde řeší zobrazování *snackbar message*.

```
Vue.use(Vuex);

new Vue({
    store: new Vuex.Store({
        modules: {
            snackbar: {
                state: {
                    snack: ''
                },
                mutations: {
                    setSnack (state, snack) {
                        state.snack = snack
                    }
                },
            }
        }
    });
});
```

Ukázka kódu 4.8: Vuex pro snackbar-message: definice

```
...
export default {
    name: "Snackbar",
    data: ...
    created: function () {
        this.$store.watch(state => state.snackbar.snack, () => {
            const msg = this.$store.state.snackbar.snack;
            if (msg !== '') {
                this.show = true;
                this.text = msg;
                this.$store.commit('setSnack', '');
            }
        })
    }
}
```

Ukázka kódu 4.9: Vuex pro snackbar-message: Snackbar komponenta

```
// Any Vue component
...
this.$store.commit('setSnack', '<message to display>');
...
```

Ukázka kódu 4.10: Vuex pro snackbar-message: použití z jiné komponenty

Kapitola 5

Testování

Závěr

TODO

Zdroje

1. ALPERT, Sophie. *Change license and remove references to PATENTS* [online]. 2017 (cit. 2018-12-21). Dostupné z: `https://github.com/facebook/react/commit/b765fb25ebc6e53bb8de2496d2828d9d01c2774b#diff-9879d6db96fd29134fc802214163b95a`.
2. ANDRUSHKO, Sviatoslav. *The Best JS Frameworks for Front End* [online]. 2018 (cit. 2018-12-21). Dostupné z: `https://rubygarage.org/blog/best-javascript-frameworks-for-front-end`.
3. *Angular - One framework. Mobile & desktop.* [online]. 2018 (cit. 2018-12-16). Dostupné z: `https://angular.io/`.
4. *Angular - Testing* [online]. 2019 (cit. 2019-01-22). Dostupné z: `https://angular.io/guide/testing`.
5. *Angular - What is Angular?* [online]. 2018 (cit. 2018-12-28). Dostupné z: `https://angular.io/docs`.
6. *Babel · The compiler for next generation JavaScript* [online]. 2019 (cit. 2019-03-01). Dostupné z: `https://babeljs.io/`.
7. *Backbone Test Suite* [online] (cit. 2018-12-28). Dostupné z: `http://backbonejs.org/test/`.
8. *Backbone.js* [online]. 2016 (cit. 2018-12-17). Dostupné z: `http://backbonejs.org/`.
9. *Backbone.js* [online]. 2016 (cit. 2018-12-28). Dostupné z: `http://backbonejs.org/#Getting-started`.

10. *Build amazing things* [online]. 2018 (cit. 2018-12-21). Dostupné z: https://www.npmjs.com/.

11. *Classes - JavaScript | MDN* [online]. 2019 (cit. 2019-03-02). Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes.

12. *Collection: Front-end JavaScript frameworks* [online]. 2018 (cit. 2018-12-17). Dostupné z: https://github.com/collections/front-end-javascript-frameworks.

13. *Comparison with Other Frameworks - Vue.js* [online]. 2018 (cit. 2018-12-14). Dostupné z: https://vuejs.org/v2/guide/comparison.html.

14. EHRENBERG, Daniel. *Class field declarations for JavaScript* [online]. 2019 (cit. 2019-03-02). Dostupné z: https://github.com/tc39/proposal-class-fields#private-fields.

15. *Ember.js - A framework for ambitious web developers* [online]. 2018 (cit. 2018-12-17). Dostupné z: https://www.emberjs.com/.

16. *Ember.js Guides - Guides and Tutorials - Ember Guides* [online]. 2018 (cit. 2018-12-28). Dostupné z: https://guides.emberjs.com/release/.

17. *Getting Started - React* [online]. 2018 (cit. 2018-12-28). Dostupné z: https://reactjs.org/docs/getting-started.html.

18. GOEL, Aman. *Top 10 Web Development Frameworks in 2018* [online]. 2018 (cit. 2018-12-17). Dostupné z: https://hackr.io/blog/top-10-web-development-frameworks-in-2018.

19. CHANDRA, Rajdeep. *My experience with Angular 2 , React and Vue* [online]. 2018 (cit. 2018-12-21). Dostupné z: https://medium.com/@rajrock38/my-experience-with-angular-2-react-and-vue-fb654e3ecf33.

20. *Introduction - Enzyme* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://airbnb.io/enzyme/.

21. *Introduction - Testing - Ember Guides* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://guides.emberjs.com/release/testing/.

22. *Introduction - Vue.js* [online]. 2018 (cit. 2018-12-28). Dostupné z: https://vuejs.org/v2/guide/.

23. *Introduction | Vue Router* [online]. 2018 (cit. 2019-02-09). Dostupné z: https://router.vuejs.org/.

24. *Introduction | Vue Test Utils* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://vue-test-utils.vuejs.org/.

25. *Jagu - Software na míru, webové prezentace, grafika, webhosting* [online]. 2018 (cit. 2018-12-21). Dostupné z: https://www.jagu.cz/.

26. JANČA, Marek. *Webpack - moderní Web Development | Ackee blog* [online]. 2017 (cit. 2019-02-11). Dostupné z: https://www.ackee.cz/blog/moderni-web-development-webpack/.

27. *JavaScript Error Tracking* [online]. 2019 (cit. 2019-01-21). Dostupné z: https://sentry.io/for/javascript/.

28. *Jest - Delightful JavaScript Testing* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://jestjs.io/.

29. KAWAGUCHI, Kazuya. *Vue i18n* [online]. 2019 (cit. 2019-03-04). Dostupné z: https://kazupon.github.io/vue-i18n/.

30. KUROSKI, Daniel. *Working an application in Vue.js with TDD* [online]. 2018 (cit. 2019-01-22). Dostupné z: https://vue-test-utils.vuejs.org/.

31. LASCIK, V. *Honest look at Ember in the middle of 2018* [online]. 2018 (cit. 2018-12-20). Dostupné z: https://medium.com/@vlascik/honest-look-at-ember-in-the-middle-of-2018-a0dc2787e506.

32. *Mocha - the fun, simple, flexible JavaScript test framework* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://mochajs.org/.

33. MOURA, Marcos. *Vue Material - Material Design for Vue.js* [online]. 2019 (cit. 2019-03-02). Dostupné z: https://vuematerial.io/.

34. OGDEN, Cody. *Killed by Google - Google Graveyard - A Google Cemetery* [online]. 2018 (cit. 2018-12-23). Dostupné z: https://killedbygoogle.com/.

35. *Overview | Vue CLI 3* [online]. 2019 (cit. 2019-02-11). Dostupné z: https://cli.vuejs.org/guide/.

36. *Protractor - end-to-end testing for AngularJS* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://www.protractortest.org/.

37. *React - A JavaScript library for building user interfaces* [online]. 2018 (cit. 2018-12-16). Dostupné z: https://reactjs.org/.

38. ROEMER, Ryan. *Backbone.js Testing* [online]. 2018 (cit. 2018-12-28). Dostupné z: http://backbone-testing.com/.

39. *Routing* [online]. 2018 (cit. 2018-12-21). Dostupné z: https://vuejs.org/v2/guide/routing.html.

40. *Sentry | Error Tracking Software — JavaScript, Python, PHP, Ruby, more* [online]. 2019 (cit. 2019-01-21). Dostupné z: https://sentry.io/welcome/.

41. *Test Utilities - React* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://reactjs.org/docs/test-utils.html.

42. *Tilde Inc. - About Us* [online]. 2018 (cit. 2018-12-17). Dostupné z: https://www.tilde.io/about-us/.

43. *Unit Testing - Vue.js* [online]. 2019 (cit. 2019-01-22). Dostupné z: https://vuejs.org/v2/guide/unit-testing.html.

44. *Vue.js Material Component Framework — Vuetify.js* [online]. 2019 (cit. 2019-03-02). Dostupné z: https://vuetifyjs.com/.

45. *webpack* [online]. 2019 (cit. 2019-02-10). Dostupné z: https://webpack.js.org/.

46. *What is Vuex? | Vuex* [online]. 2019 (cit. 2019-03-04). Dostupné z: https://vuex.vuejs.org/.

47. WOLFF, Adam. *Relicensing React, Jest, Flow, and Immutable.js* [online]. 2017 (cit. 2018-12-21). Dostupné z: https://code.fb.com/web/relicensing-react-jest-flow-and-immutable-js/.

48. YAOLONG, Dengy. *how to change document.title in vue-router? · Issue #914 · vuejs/vue-router* [online]. 2016 (cit. 2019-02-09). Dostupné z: https://github.com/vuejs/vue-router/issues/914.

49. YOU, Evan. *Vue - The Progressive JavaScript Framework* [online]. 2018 (cit. 2018-12-16). Dostupné z: https://vuejs.org/.

Dodatek A

Seznam použitých zkratek

API	Application Programming Interface
BSD	Berkeley Software Distribution
CI	Continuous Integration
CLI	Command Line Interface
DI	Dependency Injection
E2E	End-to-end
ES	ECMAScript
FIT	Fakulta informačních technologií
GUI	Graphical User Interface
MIT	Massachusetts Institute of Technology
NPM	Node Package Manager
OOP	Objektově orientované programování
OS	Operační Systém
SASS	Syntactically Awesome Style Sheets
SPA	Single Page Application
TDD	Test-driven development
URL	Uniform Resource Locator
JS	Javascript

Dodatek B

Slovník pojmů

Backend	Část aplikace, která se stará o ukládání dat, jejich zpracování a bussiness logiku. Většinou není přímo přístupná koncovému uživateli, ten k ní přistupuje přes frontend.
Dependency Injection	Technika, která umožňuje ”vložení” instance objektu, který poskytuje nějakou službu, do jiného objektu, který pak může danou službu efektivně používat.
EcmaScript	Definice programovacího jazyka, kterou implementuje například Javascript.
Favicon	Ikonka webové stránky, která se ve většině prohlížečů zobrazuje v panelu v horní liště a na dalších místech, které odkazují na daný web.
Framework	Softwarová struktura, která slouží jako podpora pro pohodlnější programování. Může obsahovat podpůrné funkce, knihovny či nástroje pro efektivnější, bezpečnější a pohodlnější vývoj softwaru.
Frontent	Část aplikace, s kterou přímo interaguje koncový uživatel či administrátor, typicky pomocí GUI. Většinou komunikuje s druhou, serverovou částí - backendem.

GitHub	GitHub je webová služba podporující vývoj softwaru za pomoci verzovacího nástroje Git.
Hot Swapping	Jedná se o způsob zobrazení změn v aplikace při změně jejího kódu. Při hot-swappingu není potřeba spouštět žádné procesy sestavení aplikace a často ani načíst znovu obrazovku - vše je provedeno automaticky za běhu a změny jsou viditelné ihned.
Middleware	Software realizující integraci mezi dvěma jinými systémy, typicky pomocí API.
Sentry	Sentry je webová služba pro sledování chyb, které v aplikaci nastanou. Při použití v produkčním prostředí může vývojář díky Sentry o chybě vědět ještě dříve, než ji uživatel nahlásí, a to včetně všech detailů, jako například jaké kroky chybě předcházely, prostředí, ve kterém k chybě došlo a mnoho dalších.
Snackbar	Velmi podobný toast message, narozdíl od čistě informativního toast message umožňuje Snackbar uživateli spustit i programátorem definovanou akci. Zobrazuje se typicky ve spodní části aplikace a informuje o proběhlé akci.
Toast message	Krátká informativní zpráva, který se objevuje ve spodní části aplikace a obsahuje typicky informaci o potvrzení provedení akce.
TypeScript	Nadmnožina JavaScriptu, která jej rozšiřuje především o statické typování proměnných a další atributy z OOP.
Vuetify	Knihovna pro Vue.js, která implementuje Material Design a poskytuje základní stavební komponenty, ale i pokročilé bloky jako například datové tabulky.

Vuex	Knihovna pro Vue.js, která se stará o state management.
Webpack	Webpack je software, který zpracovává součásti webových aplikací a tvoří z nich balíčky vhodné pro webové prohlížeče. Primárně je zaměřen na Javascript, ale dokáže zpracovávat i řadu dalších formátů, přes styly v css či sass, obrázky v png, jpeg, svg či konfigurace v json, yaml a dalších.
WebView	Komponenta nativní Android aplikace, která zobrazuje stanovenou URL jako svůj obsah. Používá se zejména v místech, kde je žádoucí zobrazovat obsah z webu, ale je potřeba přístup k funkcím zařízení, ke kterým není možné přistupovat z běžného webového prohlížeče.

Dodatek C

TODO Přílohy