FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Analýza evropských cenových srovnávačů zboží při zalistování pomocí Google Merchant XML feedu

Miroslav Vdoviak

PLZEŇ

2023

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Analýza evropských cenových srovnávačů zboží při zalistování pomocí Google Merchant XML feedu

Miroslav Vdoviak

Vedoucí práce
Ing. Martin Dostal, Ph.D.

PLZEŇ **2023**

Citace v seznamu literatury:
VDOVIAK, Miroslav. *Analýza evropských cenových srovnávačů zboží při zalistování pomocí Google Merchant XML feedu*. Plzeň, 2023. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Martin Dostal, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miroslav VDOVIAK**
Osobní číslo:	**A20B0268P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Analýza evropských cenových srovnávačů zboží při zalistování pomocí Google Merchant XML feedu**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte problematiku evropských cenových srovnávačů zboží a zjistěte, jak fungují z pohledu eshopu, který tam chce prodávat zboží.
2. Identifikujte a analyzujte hlavní problémy s předáváním dat.
3. Navrhněte metodu poskytování dat pro vybrané cenové srovnávače zboží „na jeden průchod“. Navržená metoda by měla být vhodná pro zpracování velkého množství položek.
4. Implementujte nástroj, který umožní předávání dat do vybraných cenových srovnávačů. Zaměřte se na efektivitu předávání/přenosu velkého množství dat.
5. Vytvořený nástroj pečlivě otestujte a proveďte i validaci vygenerovaného XML feedu. Proveďte statickou analýzu kódu a případně další související kontroly kvality výsledného sw.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Ing. Martin Dostal, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2022

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Plzni dne 04. dubna 2023

. . . . . .
Miroslav Vdoviak

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Tato práce se zabývá analýzou evropských cenových srovnávačů zboží. Součástí zadání je vytvoření kompletní webové aplikace, jejíž cílem je analýza, transformace a validace vstupních dat pro potřeby jednotlivých srovnávačů zboží. Z těchto dat bude následně možné generovat výsledné feedy, kde každý bude mít unikátní formát, který bude odpovídat specifikaci definované v konkrétním evropském cenovém srovnávači zboží. V rámci práce byl proveden průzkum velkého množství cenových srovnávačů, ze kterých jich bylo pět vybráno a implementováno. Dále bylo potřebné prozkoumat techniky, jakými bude probíhat validace a následná transformace dat. Pro účel implementace bylo kritické seznámení s robustním aplikačním frameworkem Symfony. Hlavním obsahem práce je samotná realizace aplikace, jejímž výsledkem bude nástroj, který jednoznačně ušetří čas uživatelům, kteří by chtěli své zboží na některém z implementovaných cenových srovnávačů inzerovat.

Abstract

This thesis deals with the analysis of European price comparators of goods. Part of the assignment is the creation of a complete web application whose aim is to analyze, transform and validate input data for the needs of individual comparators of goods. From this data it will then be possible to generate resulting feeds, where each will have a unique format that corresponds to the specification defined in the specific European price comparator of goods. The thesis carried out a survey of a number of price comparators, from which five were selected and implemented. It was also necessary to examine the techniques by which the validation and subsequent transformation of the data will take place. For the purpose of the implementation was a critical acquaintance with the robust application framework Symfony. The main content of the thesis is the actual implementation of the application, which will result in a tool that clearly saves time for users who would like to advertise their goods on one of the implemented price comparators.

Klíčová slova

Cenové srovnávače • Symfony • Kelkoo • Shopmania • Glami • Compari • PriceRunner • React

Poděkování

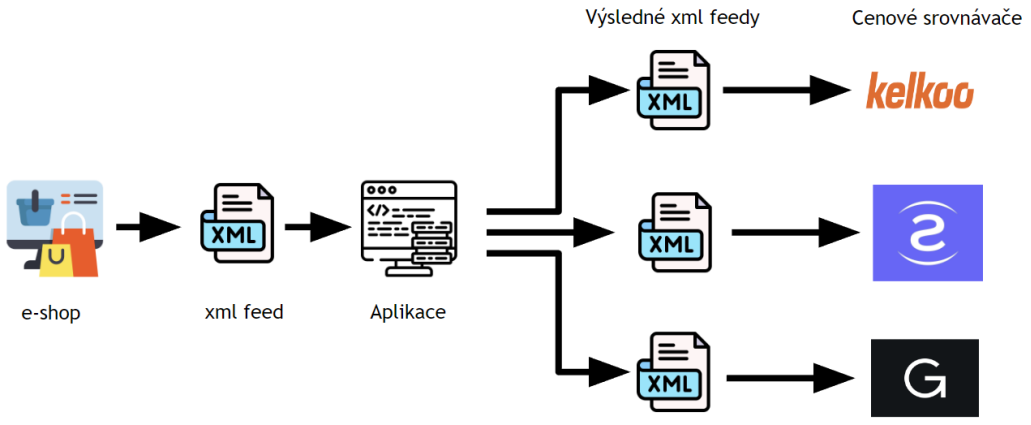
Na tomto místě bych rád poděkoval Ing. Martinu Dostalovi, Ph.D. za odborné vedení, korekci textu a veškerý věnovaný čas při vytváření této práce.

Obsah

Úvod 1

Hlavním problémem předávání dat jednotlivým cenovým srovnávačům je fakt, že neexistuje žádný standard, který by specifikoval obecný model, jenž by měli všichni dodržovat. V důsledku této skutečnosti si každý srovnávač zboží nadefinoval vlastní strukturu dat a vlastní omezení na jednotlivé parametry těchto dat. Tímto faktem vzniká potíž pro e-shopy[1], které by na těchto stránkách chtěli své zboží prodávat. Internetový obchod musí vzít všechny své produkty a převést je pro každý srovnávač do formátu, který odpovídá jeho specifikaci. Tímto vzniká problém, jelikož se může jednat o tisíce položek a tato operace bude časově, a tím pádem i finančně, velmi náročná.

Cílem této bakalářské práce je provést analýzu, identifikovat problémy a navrhnout metodu generování, která na jeden průchod dokáže vygenerovat správné feedy pro co největší množství evropských cenových srovnávačů zboží. Metoda generování je vizualizována na obrázku 1.1.



Obrázek 1.1: Základní funkcionalita aplikace

[1] Zkratka pro elektronický obchod.

Aby bylo docíleno požadovaného výsledku, bude zapotřebí vytvořit příjemné a intuitivní uživatelské rozhraní a server, který bude schopen v adekvátním čase zpracovat větší množství požadavků na generování. Dále bude potřeba, aby byla aplikace schopna zvalidovat vstupní soubor internetového obchodu a říci, zda je vůbec možné z poskytnutých dat cokoliv vygenerovat. Případné chyby souboru bude nutné uživateli zobrazit v přijatelném formátu, podle kterého je bude schopen efektivně opravit.

Po vygenerování výsledných souborů bude zapotřebí implementovat postup jakým budou nová data uživateli vrácena.

Hned v první kapitole bude vysvětlena problematika evropských cenových srovnávačů zboží a budou identifikovány a analyzovány hlavní problémy s předáváním dat. V následujících kapitolách budou popsány architektury a technologie, které se používají k tvorbě moderních webových aplikací. V kapitole 4.3 budou popsány komponenty, jež umožní na jeden průchod zpracovat data vstupního souboru a převést je do formátu, který je podporován vybranými cenovými srovnávači zboží. Nástroj, který umožní předávání dat do těchto srovnávačů, bude popsán v kapitole 5. V ní bude popsána samotná implementace uživatelského rozhraní a serveru aplikace. Problematika testování bude diskutovaná v kapitole 6. Zde budou probrány nástroje pro statickou analýzu kódu a postupy pro kontrolu vygenerovaných feedů. Dále zde bude ověřena funkcionalita použitím testovacích scénářů.

Cenové srovnávače — 2

V posledních letech došlo k velkému rozvoji online nakupování. Výrazně se zvýšila poptávka, ale také počet různých obchodů, kde můžeme chtěné zboží nakoupit. Problémem je, že již existuje velmi mnoho internetových obchodů a každý má různé vlastnosti. Jeden má lepší ceny a zákaznický servis a druhý zase nabízí dopravu zdarma či jiné slevy. Samozřejmě obyčejný člověk chce koupit zboží co nejlevněji. Zároveň si chce ale být jist, zda se nejedná o podvodný internetový obchod nebo jestli nemá špatné recenze. Například často odesílá poškozené zboží. Procházet a kontrolovat takto všechny možné stránky by bylo velmi nepraktické.

Právě k tomuto účelu existují cenové srovnávače zboží. Jedná se o velmi populární nástroje či webové platformy, pomocí kterých lze ušetřit čas i peníze. Umožnují jednoduše porovnávat veškeré zboží od různých prodejců na jednom místě [Laf08]. Dále nabízejí spoustu dodatečných funkcionalit. Mezi nejdůležitější z nich patří recenze prodejců a samotných produktů. Dále například kategorie, filtry či dostupnost zboží. Je tedy zřejmé, že kupující bude mít mnohem větší přehled, díky kterému se pravděpodobně lépe rozhodne, kde a co tedy nakoupit.

Asi nejznámějším cenovým srovnávačem v České republice je nákupní rádce Heureka.

V této kapitole se podíváme jakým způsobem lze produkty na srovnávačích inzerovat. Dále budou popsány implementované srovnávače a také již existující nástroje pro generování feedů.

2.1 Inzerce produktů na cenových srovnávačích

Postup se může lišit v závislosti na konkrétním srovnávači. V základu je ale velmi identický. V prvním kroku je nutné vybrat si cenový srovnávač zboží, na kterém chceme inzerovat. Jakmile je tento bod splněn, je potřebné se na této stránce zaregistrovat a vytvořit si účet. Dále přichází nejdůležitější krok a to je nahrání svých produktových dat na cílový srovnávač [ČSO21].

Tato data jsou většinou ve formátu xml a musí dodržet strukturu definovanou cílovým srovnávačem. Tou je například pojmenování elementů a vyplnění povinných polí. Mezi ně prakticky vždy patří jméno produktu, cena a popis. Soubor obsahující tato data je pravidelně načítán a přináší tedy aktuální informace.

2.2 Výsledky srovnávání

Jak již bylo řečeno v předchozí kapitole, pro inzerci je nutné mít produktová data ve vhodném formátu. To samozřejmě pro inzerci stačí, ale nemusí to být dostačující. Pro co nejlepší výsledky je nutné aby se e-shop zobrazoval mezi prvními nalezenými. Pokud se totiž bude stejný produkt prodávat v desítkách jiných obchodů, uživatel pravděpodobně nebude procházet všechny výsledky, ale jen první nalezené.

Pro posun v žebříčku vygenerovaných výsledků není potřeba mít nejnižší cenu, ale mnohem důležitější je mít ověřený e-shop a velké množství pozitivních recenzí.

2.3 Podporované formáty

Jak již bylo řečeno dříve, produktová data jsou nejčastěji ve formátu XML. Existují ale i jiné. V této části si představíme dva nejpoužívanější představitele.

2.3.1 XML

XML (Extensible Markup Language) je značkovací jazyk podobný jazyka HTML [Doc23b]. Narozdíl od HTML ale nemá pevně předdefinované elementy. Každý tvůrce xml souboru si tedy může podle potřeby vytvořit elementy vlastní. Byl vyvinut na počátku devadesátých let a velmi rychle se stal oblíbeným formátem pro popis dat a pro jejich výměnu mezi různými systémy či aplikacemi.

Mezi jeho výhody patří čitelnost, platformová nezávislost a možnost validace pomocí validačních schémat.

Na příkladu 2.1 můžeme vidět základní syntax. Tedy entitu student, která má definované jméno, příjmení a věk.

Zdrojový kód 2.1: Ukázka značkovacího jazyku XML

```
<student>
    <name>Jan</name>
    <surname>Novák</surname>
    <age>30</age>
</student>
```

2.3.2 JSON

JSON (Javascript object notation) je textově založený jednoduše čitelný formát. Je nezávislý na platformě. Jedná se o standard používaný pro výměnu dat na webu. K tomuto účelu podporuje i komplexnější datové struktury jakými jsou objekty či pole.

Styl zápisu je velmi jednoduchý. Pro tvorbu objektů jsou užívány složené závorky a pro pole jsou použity hranaté závorky. Celý JSON je reprezentován ve formátu klíč-hodnota [Dun11], kde za každou hodnotou následuje čárka.

Základní použití můžeme vidět na kódovém bloku 2.2. Stejně jako u příkladu v sekci o XML reprezentuje studenta, který má definované jméno, příjmení a věk.

Zdrojový kód 2.2: Ukázka formátu JSON

```
1  student = {
2    name: Jan,
3    surname: Novak,
4    age: 30,
5  }
```

2.4 Implementované cenové srovnávače zboží

V rámci bakalářské práce bylo kritické analyzovat velké množství evropských srovnávačů a z nich vybrat pět reprezentantů. Kritickým aspektem bylo, aby srovnávač pokrýval co nejvíce států.

V rámci analýzy bylo také důležité najít ke každému srovnávači specifikaci produktového feedu. To bohužel některé stránky nabízejí až po registraci a vytvoření účtu. Po nalezení příslušné specifikace bylo nezbytné zkontrolovat, zda je vůbec do ní možné vstupní soubor přetransformovat. Například některé srovnávače povinně požadují takové elementy, které se ve vstupním souboru nenachází, protože nejsou definovány ve specifikaci google merchant xml feedu. Tímto aspektem došlo k vyfiltrování několika srovnávačů. Zde jsou popsány finální vybrané srovnávače.

Shopmania

Mezinárodní srovnávač orientovaný na všechny kategorie produktů. Nabízí velkou variaci funkcí, mezi které patří vytváření seznamů oblíbených produktů nebo sledování historie vývoje ceny produktu a následné upozornění při jejím poklesu.

Kelkoo

Jeden z nejstarších srovnávačů v evropě. Nabízí téměř identické funkce jako

Shopmanie. Je využíván ke správě e-commerce[1] kampaní v 39 světových trzích [Kel22].

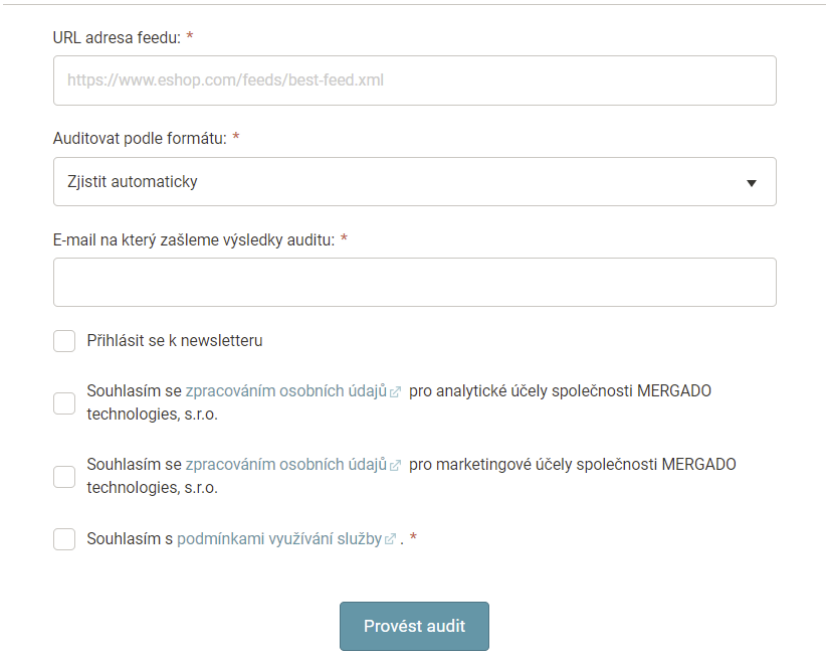
PriceRunner
Kompletně nezávislý srovnávač působící ve Velké Británii, Švédsku, Norsku a Dánsku. Z velké části zaměřen na techniku.

Glami
Orientován primárně na módu a kosmetiku. Užitečný pro uživatele, kteří hledají inspiraci a pravidelně sledují nové trendy z této oblasti.

Compari
Velmi oblíben v Rumunsku a České Republice. Poskytuje mobilní aplikaci pro uživatele s chytrými telefony.

2.5 Existující nástroje pro generování xml feedů

Jelikož se jedná o velmi moderní téma, není divu, že již existují aplikace, které obdobnou funkcionalitu nabízejí. V rámci vývoje aplikace došlo k seznámení se dvěma níže popsanými aplikacemi, kde jedna z nich byla použita k validaci vygenerovaných feedů.



Obrázek 2.1: Formulář pro audit feedu na stránce Mergado [Mer23]

[1] Forma obchodování využívající moderní elektronické komunikační prostředky.

Mergado

Marketingový nástroj pro správu a optimalizaci produktových feedů. Umožňuje transformaci, optimalizaci a editaci dat. Dále nabízí monitorování a analýzu výkonu na různých srovnávačích. Zdarma také nabízejí audit xml feedu, který byl použit pro validaci výsledných vygenerovaných feedů aplikací. Vzhled formuláře pro audit feedu můžeme vidět na obrázku 2.1

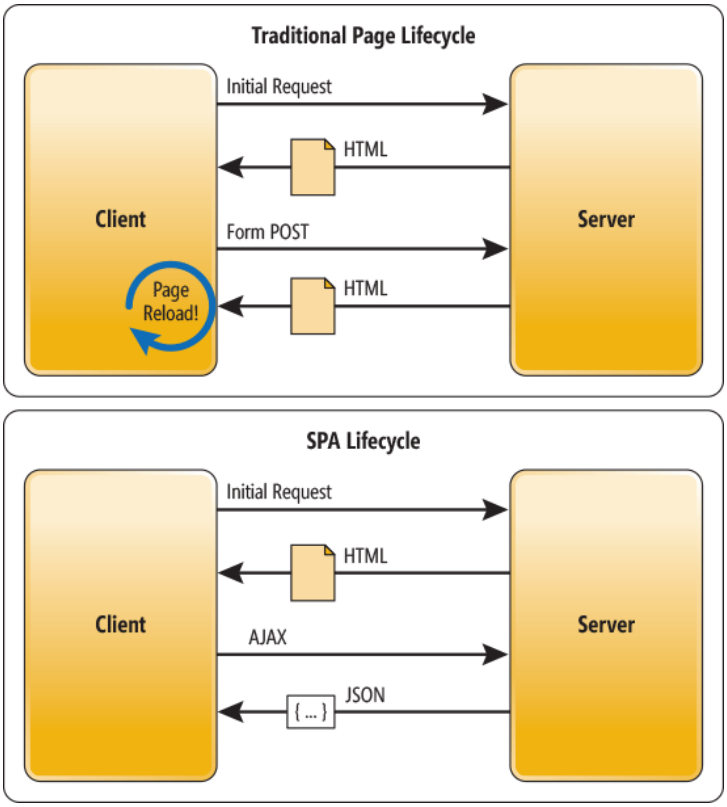
Conviu

Velmi podobný nástroji Mergado. Taktéž umožňuje editaci, monitorování a optimalizaci produktových feedů.

Druhy webových aplikací

3

V této kapitole budou popsány dvě možné populární architektury užívané k tvorbě webových aplikací. Jejich rozdíl, jenž bude popsán níže, můžeme vypozorovat na obrázku 3.1.



Obrázek 3.1: MPA vs SPA [kex15]

3.1 MPA

MPA (multi-page applications) neboli vícestránkové webové aplikace využívají tradiční a dlouhá léta používaný způsob fungování. Jedná se o aplikace, které jsou tvořeny několika HTML stránkami, jež se renderují na serveru. Při přechodu na jinou webovou stránku je odeslán požadavek na server, který novou HTML stránku navrátí. Důsledkem toho lze vypozorovat, že uživatel musí při každém takovém požadavku čekat na odpověď serveru. To může razantně snížit celkovou interakci s aplikací. Tento přístup je vývojově i údržbově složitější [Koď20]. Využívá se často pro aplikace, které nabízí velké množství kontentu. Příkladem může být Amazon. Jeho logo můžeme vidět na obrázku 3.2.

Obrázek 3.2: Amazon logo [Log23]

3.2 SPA

SPA (single-page applications) česky jednostránkové aplikace používají pro svou funkcionalitu pouze jednu HTML stránku. Veškerý běh aplikace a navigace mezi různými stránkami probíhá na straně klienta bez nutnosti použití serveru. Díky absenci neustálého dotazování serveru a načítání nových stránek dochází k razantnímu zvýšení rychlosti. Zároveň je aplikace schopná v okamžiku reagovat na uživatelské požadavky, čímž se mnohonásobně zvyšuje interakce s uživatelem a celkový požitek z užívání aplikace. SPA také dokáže mnohem lépe reagovat na stále se navyšující množství uživatelů. Mezi další výhody můžeme zařadit efektivní využívání

Obrázek 3.3: Loga frontendových frameworků [Chr20]

mezipaměti a možnost používání aplikace bez internetového připojení. Samozřejmě pouze v omezeném měřítku.

Jelikož nemůže být vše pouze pozitivní, SPA má i své nevýhody. Mezi známé problémy můžeme zařadit například špatnou optimalizaci pro SEO[1] [Koď20]. To je cekem problém, protože čím vyšší pozice ve výsledcích vyhledávání, tím pravděpodobněji uživatel otevře nalezenou stránku. Pro příklad zeptejme se sami sebe jak často se podíváme na druhý či další list nalezených výsledků vyhledávání. Další nevýhodou může být špatná funkcionalita stránek v důsledku vypnutého Javascriptu v prohlížeči.

Pro implementaci jednostránkových aplikací jsou často používány frameworky jako React, Angular nebo Vue [Hom21]. Jejich loga můžeme vidět na obrázku 3.3.

[1] Česky optimalizace pro vyhledávače jsou metody, jež zajišťují prioritní zobrazení určité URL ve výsledcích vyhledávačů.

Technologie pro tvorbu webových aplikací

4

V této kapitole budou představeny různé technologie, pomocí kterých lze sestavit funkční webovou aplikaci. Je zřejmé, že takových technologií existuje velké množství, a proto budou představeny pouze ty, které autor této bakalářské práce zamýšlel k využití.

4.1 PHP

PHP (Hypertext Preprocessor) je serverově orientovaný skriptovací programovací jazyk [Tut16]. Skripty jsou vykonávány na straně serveru a zpracované výsledky jsou následně odeslány zpět ve formátu HTML. Je primárně určen pro tvorbu dynamických webových stránek. Jeho syntaxe je inspirovaná programovacím jazykem C. Ačkoliv se často říká, že PHP je již zastaralé, stále se jedná o nejvíce používaný jazyk pro tvorbu webových aplikací. Důvodem je jak jeho jednoduchost, tak fakt, že většině firem se finančně nevyplatí přepsat veškerý zdrojový kód aplikace do jiného programovacího jazyka.

Mezi jeho výhody patří velké množství kvalitně zpracované dokumentace a komunita vývojářů, která neustále vyvíjí nové frameworky a knihovny. Mezi populární frameworky patří Laravel, Zendl či Symfony, které bude použito pro vývoj aplikace v této bakalářské práci.

Jedná se o synchronní jazyk, který je často využíván v oblasti e-commerce. Není zcela vhodný pro asynchronní zpracování či aplikace fungující v reálném čase. Pro tyto aplikace je užitečné použít softwarový systém Node.js, který používá asynchronní, událostmi řízený, model [Kai14].

4.2 Symfony

Open source framework pro PHP slouží k tvorbě webových aplikací. Zjednodušuje jejich tvorbu díky základní struktuře projektu a velkému množství předimplementovaných komponent a metod, jež razantně zlehčují vývojáři práci a urychlují vývoj. Jeho největší výhodou je dokumentace ve vysoké kvalitě. Každá metoda, komponenta a konfigurační soubory jsou podrobně popsány a jejich funkcionalita ukázána na několika příkladech. Díky této skutečnosti je pochopení a použití těchto entit velmi triviální.

Jedná se o jeden z nejpoužívanějších PHP frameworků v Evropě. Mezi jeho konkurenty patří například Laravel. V České republice je populární možností Nette.

Obrázek 4.1: Symfony logo [Sym23b]

4.3 Symfony komponenty

Symfony komponenty jsou skupina oddělitelných a znovupoužitelných knihoven pro tvorbu PHP aplikací [Sym23c]. Byly otestovány ve statisících projektech a jsou používány v projektech jako Phpstan, Wikimedia nebo Composer [Sym23a]. Aktuálně je jich vytvořeno již přes 30. Jejich využitím je možné několikanásobně zvýšit rychlost vývoje a celkovou kvalitu výsledného kódu. Hlavní výhodou je oddělitelnost. To znamená, že není nutné použít celý framework, ale pouze určité, pro aplikaci potřebné, části. Níže budou popsány komponenty, které byly po analýze vybrány jako ideální k použití pro vytvoření výsledné aplikace.

4.3.1 Messenger

Umožňuje synchronní a asynchronní zpracování zpráv na pozadí [Sym23g]. Je hojně používány pro zvýšení škálovatelnosti aplikací. Je vhodný pro manipulaci s velkými daty. Dále se také využívá pro aktivity, které není nutné ihned zpracovat. Příkladem může být odeslání emailu po registraci do aplikace.

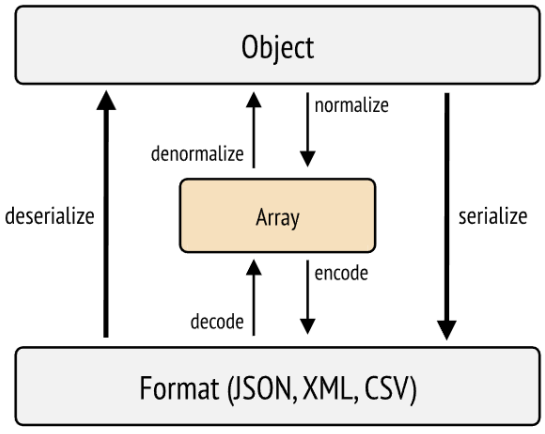
Pro zajištění správné funkcionality je nutné definovat 2 třídy. Jedna bude reprezentovat zprávu, kterou je nutné zpracovat. Druhá specifikuje jakým způsobem bude zpracována. Pro asynchronní zpracování zpráv je zapotřebí vytvoření transportu. Jedná se o mechanismus, který je schopen posílat a přijímat zprávy. Framework symfony nabízí nemalé množství transportů. Například AMQP, doctrine, redis a mnoho dalších. Pro přijímání, nebo také jinak řečeno konzumaci zpráv, zde existuje entita nazývající se worker. Program může mít více workerů. V případě většího počtu pracovníků je výhodné nadefinovat si supervisora. Jedná se o správce procesů, který se stará o pracovníky. Kontroluje zda běží a případně je restartuje.

4.3.2 HttpKernel

Úkolem této komponenty je zpracování HTTP[1] požadavku a navrácení příslušné odpovědi. Jedná se o jednu z nejdůležitějších částí celého frameworku. Velké využití má u routingu, jenž bude popsán v kapitole 4.3.6.

4.3.3 Serializer

Slouží k převádění objektů do různých formátů. Těmi mohou být XML, YAML, JSON nebo CSV [Sym23h]. Funguje oběma směry a transformovaná data lze tedy jednoduše převést zpět do původního formátu. V aplikaci bude využit pro zpracování vstupního souboru, který je ve formátu XML. Data z tohoto souboru budou převedena do pole objektů, se kterým bude dále možné jednoduše pracovat. Chování komponenty lze podrobně vidět na obrázku 4.2. Z něj je možné vypozorovat, že komponenta je schopná zpracovat i **pole** ruzných formátů.



Obrázek 4.2: Funcionalita serializer komponenty [Sym23h]

[1] Internetový protokol určený pro komunikaci s WWW servery.

4.3.4 Dependency injection

Jedná se o komponentu, pomocí které lze oddělit vytváření od použití objektů. Této vlastnosti je dosáhnuto s využitím Dependency injection kontejneru, ve kterém jsou registrovány všechny potřebné závislosti k vytvoření cílového objektu. Výhodou Dependency injection je fakt, že snižuje závislost mezi jednotlivými komponentami, redukuje množství programátorem psaného kódu a zjednodušuje tvorbu objektů a jejich předávání.

4.3.5 Mime

Nabízí jednoduchý a intuitivní způsob jakým vkládat text a binární data do e-mailových zpráv. Bude použit pro zaslání vygenerovaných výsledků na mail.

Na kódovém bloku 4.1 můžeme vidět jednoduché vytvoření email obejktu, ke kterému jsou postupně řetězově přidávány informace. Tedy informace o odesílateli, příjemci, subjekt, samotný text, html data a příloha.

Zdrojový kód 4.1: Ukázka použití mime komponenty

```
1  $email = (new Email())
2    ->from('sender@example.com')
3    ->to('reciever@example.com')
4    ->subject('Hello reciever!')
5    ->text('This is a test.')
6    ->html('<p>This is a <b>test email</b>.</p>')
7    ->attachFromPath('/path/to/file.jpg')
```

4.3.6 Routing

Její hlavní funkcionalitou je mapování URL adres na příslušné akce v aplikaci. Tedy například na akci definovanou v kontroleru programu. Umožňuje nadefinování všech možných typů požadavků jako jsou `GET`, `POST`, `PUT` či `DELETE`. Pro efektivní vyhledání cesty komponenta obsahuje routovací tabulku, která zahrnuje kolekci všech registrovaných cest.

4.3.7 Validator

Umožňuje validaci dat na základě definovaných pravidel. V základu obsahuje pravidla pro validaci e-mailových a url adres, hesel, délky řetězců, formátů jako ISBN nebo IBAN a mnoho dalšího [Sym23i]. Pokud by programátorovi nestačilo velké množství předdefinovaných pravidel, má možnost si nadefinovat vlastní. Validátor nabízí několik způsobů, jakými lze pravidla použít. V aplikaci budou implementována pomocí anotací.

Na příkladu 4.2 můžeme vidět jak lze v ukázkové třídě Author zkontrolovat, zda jméno není prázdné a obsahuje pouze 20 znaků. Také je možné specifikovat zprávu, která se zobrazí pokud k příslušné chybě dojde.

Zdrojový kód 4.2: Ukázka použití Validator komponenty

```
1  class Author
2  {
3      #[Assert\NotBlank(
4        message: 'Product type can not be blank!',
5    )]
6    #[Assert\Length(
7          max: 20,
8        maxMessage: 'First name can not be longer than 20
     characters!'
9    )]
10       private $name;
11 }
```

Pokud dojde k nálezu chyby, validátor vrátí pole všech chybových hlášení, které je následně možné libovolně formátovat.

Tato komponenta se vysloveně nabízí pro validaci vstupního souboru uživatele. Nalezené chyby budou zpracovány a odeslány zpět to uživatelského rozhraní, kde je bude možné zobrazit uživateli v přijatelném a čitelném formátu.

4.3.8 FileSystem

FileSystem komponenta přináší platformě nezávislý přístup pro operace nad souborovým systémem a pro operace se soubory a adresáři [Sym23f].

4.3.9 Cache

Nabízí programátorům jednoduchý způsob, jakým ukládat data do mezipaměti. Využitím cache paměti je možné zvýšit efektivitu aplikace důsledkem rychlejšího načítání dat. Komponenta poskytuje zvýšenou bezpečnost a několik různých adaptérů pro práci s odlišnými typy mezipaměti. Těmi mohou být například Redis, Apcu, Memcached a mnoho dalších [Sym23d]. V programu bude komponenta využita pro dočasné uložení uživatelovo nahraného souboru.

4.3.10 Config

Config je nástroj určený k nastavení konfiguračních parametrů aplikace. Pomáhá načíst, vyplnit a zvalidovat konfigurační hodnoty všech možných druhů. Například YAML nebo XML [Sym23e].

4.4 Symfony knihovny

V této části si představíme knihovny, které jsou esenciální k vytvoření uživatelského rozhraní s využitím Reactu.

4.4.1 Webpack Encore

Čistě javascriptová knihovna, která obaluje Webpack a poskytuje Api[2], pomocí kterého lze velmi jednoduše využívat všechny funkce Webpacku. Webpack bude popsán v kapitole 4.9.1.

4.4.2 Twig

Twig je užitečná šablonovací knihovna. Její velkou výhodou je oddělení logiky a zpracování kódu od samotného zobrazení dat. Důsledkem je strukturovaný a více čitelný kód. Jelikož bude uživatelské rozhraní psáno v Reactu, Twig bude využit pouze jako šablona, jež bude obsahovat veškerý Webpackem zabalený kód.

4.5 Javascript

Javascript je platformě nezávislý skriptovací programovací jazyk určený k tvorbě komplexních dynamických webových stránek. Vykonáva se přímo v prohlížeči uživatele. Díky tomuto faktu tedy není nutné každé zpracování události posílat na server, ale je možné jej zpracovat na straně klienta. Používá se pro animace, interaktivní mapy a pro spoustu dalších funkcí, které mnohonásobně zvyšují interakci s uživatelem a celkový požitek s užíváním aplikace [Doc23a]. Jedná se o klíčovou technologii nutnou k tvorbě jednostránkových aplikací.

4.6 React

React je Javascriptová knihovna určená k tvorbě uživatelských rozhraní. Byla vytvořená společností Facebook [Gac15]. V nynější době se jedná o nejpopulárnější technologii určenou k implementaci jednostránkových aplikací.

React je postaven na myšlence rozdělování aplikace do spousty znovupoužitelných komponent. Ty se dají vzájemně propojovat. Důležitým prvkem Reactu je virtuální document object model. Ten abstraktně představuje reálný DOM[3]. Využívá ho k efektivní aktualizaci a načítání uživatelského rozhraní jelikož minimalizuje počet aktualizací reálného DOMU.

[2] Programové rozhraní, které nabízí sbírku procedur, funkcí, tříd a protokolů.
[3] DOM je objektově orientovaná reprezentace HTML či XML dokumentu.

Dále poskytuje jednoduchou správu stavu. Ten je většinou uchován v rodičovských komponentách a díky jednosměrnému toku dat a možnosti propojování komponent je postupně předáván, jako vlastnost (`props`), do komponent nižších. Díky této vlastnosti je snadné sledování a aktualizace stavu. Také se redukuje počet možných chyb.

V příkladu 4.3 můžeme vidět jednoduchou komponentu reprezentující studenta. Ta jako `props` obdrží tři parametry představující studentovo jméno, příjmení a věk. Tyto data následně při provolání zobrazí pod sebou jako odstavce.

Zdrojový kód 4.3: Ukázka React komponenty reprezentující studenta

```
  const Student = (props) => {
    return (
      <p>{props.name}</p>
      <p>{props.surname}</p>
      <p>{props.age}</p>
    )
  }

const name = 'Jan';
const surname = 'Novak';
const age = '30';

const app = () => {
  return (
    <Student name={name} surname={surname} age={age} />
)
}
```

4.7 Použité js frameworky a knihovny

V této kapitole budou představeny frameworky a knihovny, které budou použity v rámci této bakalářské práce.

4.7.1 Design

Pro stylování se nabízí mnoho možností. Velmi populární a používaný je bootstrap[4]. Jedná se o sadu nástrojů s předdefinovanými šablonami, které jsou založené na technologiích HTML a CSS. Dále také například Tailwind CSS[5]. Ten na rozdíl od bootstrapu nepoužívá předdefinované komponenty, ale nabízí pouze jednoduché a univerzální CSS třídy, jenž obsahují všechny možné kombinace vlastností a hod-

[4] https://getbootstrap.com/.
[5] https://tailwindcss.com/.

not klasického CSS. Pro bakalářskou práci bude použita knihovna Material UI[6]. Ta obsahuje velké množství moderně vypadajících komponent. Tyto komponenty jsou jednoduché na použití v projektu a nabízí rozsáhle možnosti stylování a přizpůsobení.

4.7.2 Routing

O směrování mezi stránkami aplikace se postará framework React Router[7]. Ten nabízí možnost směrování na straně klienta. Při klasickém směrování si webový prohlížeč vyžádá nový dokument od serveru. Následně si stáhne a vyhodnotí JS a CSS a vyrendruje HTML. Směrování na straně klienta umožní aktualizaci stránky bez nutnosti vyžádání dokumentu od webového serveru. Z tohoto důvodu je aplikace mnohem rychlejší a nabízí příležitosti pro větší dynamičnost.

4.7.3 Správa dat

Ke správě dat bude využita knihovna React Query[8]. Přesněji jeden z hooků této knihovny nazývaný `useMutation`. Hooky budou popsány v kapitole 5.1.3. Jako klasická metoda `fetch` i `useMutations` slouží k zasílání požadavků a přijímání odpovědí ze serveru. Nabízí ale i další funkcionalitu. Automaticky se stará o aktualizaci lokálních datových cache a aktualizuje lokální stav ještě předtím, než dojde odpověď ze serveru. Dále umožňuje zpracovat chybové stavy a vrací informaci o tom, zda se stále čeká na odpověď od serveru, již se dá využít k vytvoření progress baru.

4.7.4 PropTypes

Jednoduchá knihovna, pomocí které lze definovat očekávané datové typy a strukturu dat přenášených mezi jednotlivými komponentami. Díky tomuto mechanismu dochází k zvýšení stability aplikace, jelikož jsou chyby odhaleny již v průběhu vývoje aplikace.

Na následujícím kódu 4.4 je možné vidět základní použití **propTypes**, které budou kontrolovat správně předané hodnoty z příkladu 4.3.

Zdrojový kód 4.4: Ukázka použití propTypes

```
1  Student.propTypes = {
2      name: PropTypes.string.isRequired,
3      surname: PropTypes.string.isRequired,
4      age: PropTypes.number.isRequired
5  };
```

[6] https://mui.com/.
[7] https://reactrouter.com/en/main.
[8] https://tanstack.com/query/v3/.

4.8 Redis

Moderní key-value paměťové úložiště, jenž nabízí spoustu různých funkcionalit. Může sloužit jako cache, databáze či dokonce message broker[9]. Její největší výhodou je vysoká rychlost. Té je docíleno právě faktem, že je uložena primárně v paměti. Zároveň je možné skrze konfiguraci nastavit ukládání na disk. Není omezen velikostí paměti, ale umí efektivně využít swapovací soubor, do kterého ukládá nepoužívaná data.

Poskytuje AOF (Append only file), do které si zapisuje všechny příkazy, jež manipulují s databází. Z tohoto souboru je následně schopen po výpadku znova zrekonstruovat databázi zpět [Štr10].

4.9 Nástroje

V této kapitole budou popsány nástroje, jež jsou esenciální pro správnou funkcionalitu implementované aplikace.

4.9.1 Webpack

Open-source[10] nástroj sloužící ke správě balíčků a modulů v Javascriptu. Kromě JS umí transformovat i frontendové prvky jako HTML, CSS či dokonce obrázky. Při použití Webpacku dochází k načtení všech závislostí, ze kterých se vytvoří graf závislostí, který mapuje moduly a generuje jeden či více balíčků. Tyto balíčky obsahují zkompilovaný kód, který je ve formátu, se kterým umí prohlížeč pracovat.

Webpack lze rozdělit do čtyř principů [Tho19].

Vstup
: Reprezentuje vstupní bod aplikace. Jedná se o první modul, ze kterého Webpack vychází a postupně zpracovává všechny importované soubory a přidává je do již zmíněného grafu závislostí. Takto postupně prochází veškerý kód, který je **potřebný** pro spuštění aplikace.

Výstup
: Místo, kam se uloží nově zabalené soubory.

Loaders
: V základu umí webpack pracovat pouze ze soubory ve formátu JS nebo JSON. Pomocí **Loaders** je schopný převádět další typy souborů.

[9] Modul, který umožňuje komunikaci mezi různými protokoly.
[10] Technicky dostupný kód.

Plugins

Ty se používají pro operace, jež nelze vykonat pomocí **Loaderů**. Příkladem může být nastavení globálních proměnných, extrakce stylů, minifikace[11] a mnoho dalšího.

Webpack je také klíčový pro React, jelikož dokáže převáděť moderní Javascriptové prvky, jako například ES6[12] moduly, jež nejsou podporovány všemi prohlížeči, do podoby, která je již pro ně akceptovatelná.

4.9.2 Npm

Npm (Node Package Manager) je správce Javascriptových balíčků. Používá se k jejich instalaci a následné údržbě. Všechny balíčky, jež aplikace používá, jsou definovány v souboru `package.json`. Součástí npm je program příkazového řádku s jehož pomocí lze jednoduše instalovat nové balíčky. Toho je docíleno příkazem `npm install [package]`, kde **package** je jméno balíčku, který chceme nainstalovat. Veškeré nainstalované balíčky jsou uloženy v adresáři `node_modules`.

4.9.3 Composer

Nástroj pro správu knihoven a zdrojů projektu. Umožňuje deklarovat, instalovat a spravovat závislé knihovny pro specifický projekt. Funkcinalitou je téměř identický s npm. Hlavní rozdíl je, že composer je určen primárně pro programovací jazyk PHP. Všechny použité knihovny jsou definovány v souboru `composer.json`. Data těchto knihoven jsou uložena v adresáři `vendor`.

[11] Proces odstraňování nepotřebných prvků.
[12] Prvky ECMAScript specifikace, jenž je standard pro Javascript.

Implementace 5

5.1 Uživatelské rozhraní

V této kapitole bude popsána implementace uživatelského rozhraní. Budou zde vyobrazeny všechny důležité struktury a komponenty, ze kterých je rozhraní vytvořeno. Veškerý kód a zdroje náležící vizuální části programu se nachází ve složce `assets/javascript/`. Celé uživatelské rozhraní bylo napsáno v programovacím jazyce Javascript s využitím populárních knihoven React a Material UI. Budou zde postupně popsány všechny podstatné komponenty a funkce, ze kterých je výsledná aplikace složena. Následně budou jednotlivě vyobrazeny všechny stránky aplikace, které vznikly spojením všech těchto komponent a funkcí do jednoho funkčního celku.

5.1.1 Komponenty

V této kapitole bude rozepsán význam jednotlivých komponent. Tyto znovupoužitelné entity jsou použity k vytvoření celého vzhledu aplikace. Všechny se nachází ve složce `components`.

Menu
: Zobrazuje hlavičku aplikace.

Footer
: Ukazuje patičku aplikace.

UploadForm
: Jedná se o komponentu reprezentující tlačítko, přes které má uživatel možnost nahrát svůj vstupní soubor z vlastního souborového systému. Také obsahuje progress bar, kterým uživateli předává informaci, že dochází ke zpracování daného souboru. Bez jeho přítomnosti by mohlo dojít k mylnému předpokladu, že aplikace zamrzla.

BaseLayout
Tato komponenta je využívaná každou stránkou. Jejím významem je vykreslit elementy, které mají všechny stránky společné. Těmi jsou hlavička, patička a tlačítko pro nahrání vstupního souboru.

AppTooltip
AppTooltip je určena k zobrazení nápovědy uživateli po najetí myší na dané tlačítko či text.

AppSnackbar
Komponenta je využita pro vyobrazení informační zprávy uživateli.

AppBackDrop
Slouží k vytvoření načítacího dialogu, který uživatele informuje, že právě dochází k načítání obsahu.

GoogleFeedTable
Jedná se o entitu představující tabulku, která zobrazuje všechny kritické chyby vstupního souboru.

PriceCompTable
PriceCompTable je komponenta, jejíž významem je vykreslení tabulky, která ukazuje jednotlivé evropské cenové srovnávače zboží. Každý řádek této tabulky reprezentuje jeden srovnávač, který je možné rozbalit. Po rozbalení se ukáže vnitřní tabulka obsahující všechny chyby daného srovnávače.

5.1.2 Kontext

Kontext je využit k umožnění přístupu a zpracování definovaného stavu odkudkoliv z aplikace. Jedná se o globální data, ke kterým mají výhradní přístup i hluboce zanořené komponenty. V aplikaci jsou definovány tyto kontexty:

fileNameContext
Je využit pro uchovávání a změnu informace o tom jaký soubor je aktuálně uživatelem zpracováván.

priceCompSitesContext
Jedná se o důležitý stav, jejž je nutný pro výběr srovnávačů, pro které chce uživatel vytvořit výsledné xml feedy. Pomocí tohoto stavu dochází k odblokování či zablokování tlačítka generování.

snackBarContext
Slouží k manipulaci se snackbar komponentou. Pomocí kontextu je možné danou komponentu zavolat a přidělit ji zpravů s různou závažností.

5.1.3 Hooks

Jedná se o speciální funkce Reactu. Tyto funkce umožňují React komponentám využívat stavy bez nutnosti definice tříd. Celkově nabízejí mnohem elegantnější správu, organizaci a efektivitu kódu. Všechny názvy hook funkcí musí začínat řetězcem **use**.

useGoogleFeedErrors
: Funkce, která používá `useMutation` hook definovaný v kapitole 4.7.3. Slouží k provolání serverového endpointu[1] určeného ke kontrole vstupního souboru. Pokud soubor neobsahoval žádné kritické chyby, přesměruje uživatele na stránku generování výsledných xml feedů. V opačném případě se uživateli ukáže stránka chyb google feedu, kde budou vyobrazeny všechny nevalidní hodnoty a jejich lokace ve vstupním souboru.

useGenerateFeed
: Jakožto předchozí hook i `useGenerateFeed` funkce využívá `useMutation`. Významem `useGenerateFeed` je provolání serverového endpointu, jehož posláním je vytvoření výsledných xml feedů.

5.1.4 Vzhled aplikace

V této části bude popsán výsledný vzhled aplikace a stavy, do kterých se uživatel může při práci s aplikací dostat. Veškerý vzhled a funkcionalita uživatelského rozhraní je docílena kombinací všech elementů, které byly popsány v předchozích kapitolách.

Všechny stránky aplikace obsahují hlavičku, patičku a tlačítko pro výběr vstupního souboru. Po nahrání souboru mohou nastat tři stavy. Jestliže došlo k chybě na straně serveru, nebo vstupní soubor neodpovídá specifikaci google merchant xml feedu, zobrazí se uživateli chybová stránka. Pokud je soubor v pořádku, dojde k jeho validaci dle specifikace a vyhodnotí se kritické a nekritické chyby. V případě nálezu kritických chyb nebude možné další zpracování souboru a bude zobrazena stránka chyby google feedu. V opačné situaci se uživateli ukáže stránka generování.

5.1.4.1 Hlavní stránka

Hlavní stránka slouží pouze pro výběr vstupního souboru, pro který by chtěl uživatel výsledné feedy vytvořit.

5.1.4.2 Stránka chyby google feedu

Po přesměrování na tuto stránku se uživateli ukáže tabulka, jež bude zobrazovat všechny kritické chyby, které byly při validaci vstupního souboru nalezeny. Každý

[1] Funkce dostupné skrze API.

řádek tabulky bude představovat jednu kritickou chybu. Obsahovat bude následující položky:

- chybová hláška,
- cesta k špatně zadané hodnotě,
- kód chyby,
- nevalidní hodnota.

Tabulka je dynamická a uživatel si bude moct vybrat kolik řádků najednou bude tabulka ukazovat a na další řádky bude mít možnost se proklikat pomocí jednoduché navigace na přepínání stránek tabulky. Tuto tabuku zobrazuje obrázek 5.1.

Google feed errors

Code	Message	Property path	Invalid value
8e179f1b-97aa-4560-a02f-2a8b42e49df7	Avaibility must be one of "in_stock", "out_of_stock", "preorder", "backorder"!	[0].availability	in k
8e179f1b-97aa-4560-a02f-2a8b42e49df7	Avaibility must be one of "in_stock", "out_of_stock", "preorder", "backorder"!	[1].availability	in stock
8e179f1b-97aa-4560-a02f-2a8b42e49df7	Avaibility must be one of "in_stock", "out_of_stock", "preorder", "backorder"!	[2].availability	in stock
8e179f1b-97aa-4560-a02f-2a8b42e49df7	Avaibility must be one of "in_stock", "out_of_stock", "preorder", "backorder"!	[3].availability	in stock
8e179f1b-97aa-4560-a02f-2a8b42e49df7	Avaibility must be one of "in_stock", "out_of_stock", "preorder", "backorder"!	[4].availability	in stock

Rows per page: 5 1–5 of 18

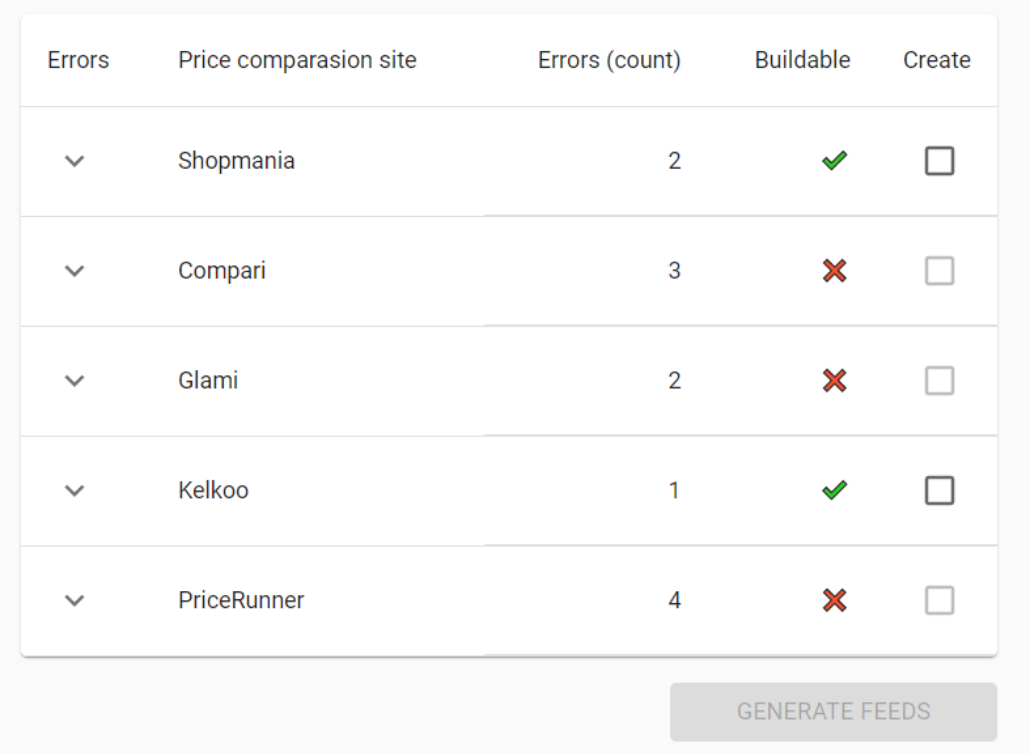
Obrázek 5.1: Tabulka chyb google feedu

5.1.4.3 Stránka generování

Přesměrováním na stránku generování dojde k zobrazení tabulky, jejíž řádky budou reprezentovat evropské cenové srovnávače. Každý srovnávač bude obsahovat tyto údaje:

- název srovnávače,
- součet kritických a nekritických chyb,
- informaci zda je pro daný srovnávač možné výsledný xml feed vytvořit,
- zaškrtávací pole, kterým aplikaci řekneme, že pro tento srovnávač chceme feed vygenerovat,
- rozbalovací menu.

Tuto tabulku je možné vidět na obrázku 5.2.

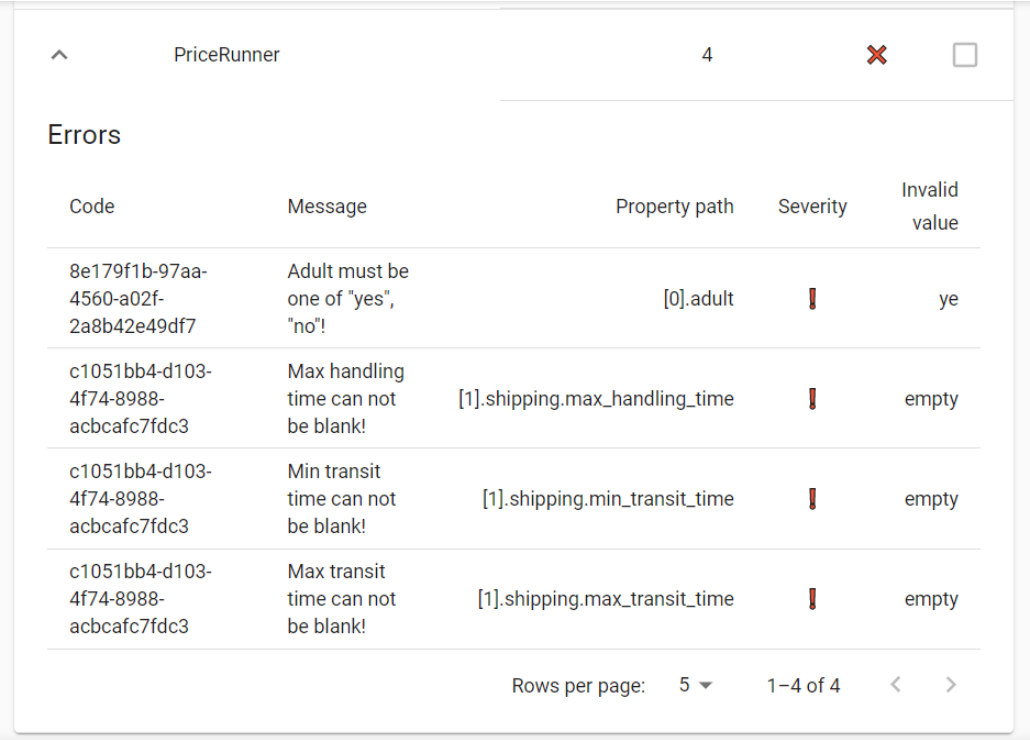


Obrázek 5.2: Tabulka všech srovnávačů

Po stisknutí tlačítka rozbalovacího menu dojde k zobrazení tabulky obdobné ze stránky chyby google feedu. Tato tabulka bude obsahovat všechny chyby, které vznikly při validaci daného srovnávače vůči zadanému vstupnímu souboru. Příkladem takové chyby může být atribut, který v google xml feedu není povinný a není v souboru vyplněn, ale srovnávač ho vyžaduje jako povinný atribut. Vzhled této tabulky lze vidět na obrázku 5.3.

Následně bude mít uživatel možnost si vybrat, pro které cenové srovnávače chce výsledné feedy vytvořit s předpokladem, že daný cenový srovnávač neobsahoval

kritické chyby. Po stisknutí tlačítka **generovat** se odešle požadavek na generování a uloží se do fronty požadavků. Jakmile bude mít server volné prostředky, požadavek zpracuje a výsledky následně pošle na elektronickou poštu jako klikatelné odkazy. Po kliknutí na odkaz se daný feed stáhne a internetovému obchodu stačí daný soubor nahrát na odpovídající stránku, a poté zde prodávat své zboží.



Obrázek 5.3: Detail chyb pro srovnávač PriceRunner

5.1.4.4 Chybová stránka

Jestliže došlo z nějakého důvodu k interní chybě serveru nebo uživatel zadal vstupní soubor ve špatném formátu nebo došlo k jakékoliv nestandardní situaci, zobrazí se uživateli chybová stránka. Ta je stejná jako hlavní stránka s rozdílem velkého nadpisu oznamujícího chybu.

5.2 Server

Server je implementován v programovacím jazyce PHP s pomocí moderního webového frameworku Symfony. Ten dodržuje architekturu **MVC**. Veškerá logika serveru se nachází ve složce `/src`. Jsou zde ve velké míře používány komponenty, jejichž význam byl vysvětlen v kapitole 4.3. Postupně zde budou vyobrazeny všechny důležité

třídy a funkce, které byly použity pro vytvoření výsledné aplikace. Mimo zmiňovaný adresář budou popsány konfigurační adresáře, adresáře obsahující šablony a adresáře, které souvisí s funkcionalitou serveru.

5.2.1 Formát vstupního souboru

Pro správnou funkcionalitu je nutné dodržet definovaný formát pro vstupní soubor.

Zdrojový kód 5.1: Ukázka formátu vstupního souboru

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:g="http://base.google.com/ns/1.0">
  <channel>
    <item>
      <g:id>id1</g:id>
      ...
    </item>
    ...
    <item>
      <g:id>id2</g:id>
      ...
    </item>
  </channel>
</rss>
```

Jak můžeme vidět z blokového kódu 5.1, je nutné definovat následující elementy.

- deklarující element xml, jenž definuje verzi xml,
- rss, který reprezentuje kořenový prvek a je obecně používán pro syndikaci obsahu[2],
- channel element, ve kterém už budou vnořeny jednotlivé **item** elementy, jež představují samotné produkty,

5.2.2 Constants

Pro každý cenový srovnávač zboží je zde definována vlastní třída. Tyto třídy slouží pouze pro uchování důležitých dat. Těmi mohou být například jméno srovnávače, jméno výstupního souboru nebo odkaz na funkci, jenž bude v aplikaci použita pro přemapování dat z vstupního google xml feedu do specifického srovnávače a mnoho dalších. Další velmi důležitou položkou je pole, ve kterém jsou specifikovány všechny validační omezení, jež budou použity při kontrole dat ze vstupního feedu. Příkladem těchto omezení by mohla být délka popisu či délka kategorie.

[2] Opětovné předávání aktuálních informací, které obsahuje internet.

Určitá data ze všech těchto tříd jsou přidána do jednoho společného pole. Tímto způsobem je možné jednoduše iterovat přes všechny cenové srovnávače a extrahovat potřebné informace. Toto řešení razantně sníží délku a zvýší čistotu výsledného kódu. Požadovaná data jsou následující.

NAME
: Jméno srovnávače.

GROUP_NAME
: Jméno, které bude použito k anotaci atributů google feedu. Tímto jménem budou označeny takové atributy google feedu, které nejsou povinné nebo je jejich nevalidní hodnota možná programově upravit. Například oříznout příliš dlouhý popis produktu.

GROUP_NAME_REQ
: Obdobné jako u **GROUP_NAME**. Rozdíl je, že tyto chyby jsou již kritické. Z toho vyplývá, že je není možné nijak upravit a pro srovnávač, jejž obsahuje takovou chybu nebude možné výsledný feed vygenerovat.

XML_NAME
: Jméno výstupního souboru daného srovnávače.

CREATE_ITEM_FUN
: Mapovací funkce, která přemapuje atributy vstupního google feedu do atributů unikátních pro cílový srovnávač.

TO_UPPER_CASE
: Tato informace sdělí programu, že feed obsahující tuto informaci bude mít veškeré elementy velkými písmeny.

TO_SNAKE_CASE
: Stejné jako u **TO_UPPER_CASE**. Odlišností je, že všechny elementy budou ve formátu **snake_case**. Tedy jednotlivá slova budou odděleny znakem '_'.

Poslední zde definovanou třídou je třída `Regex`. V té se vyskytují všechny možné regexy[3], které jsou použity pro komplexnější validace atributů. Například kontrola, že zboží má validní váhu a jednotku.

[3]Textový řetězec, jenž představuje vzor pro vyhledávání v textu.

5.2.3 Entity

Zde je definována nejdůležitější třída celé aplikace, jejíž jméno je `Item`. Tato třída reprezentuje google xml feed. Obsahuje velké množství atributů. Všechny tyto atributy jsou definovány v oficiální dokumentaci Googlu[4] k datu 10.2.2023. V této třídě dochází k validaci jednotlivých atributů a k mapování do ostatních cenových srovnávačů.

Validace je docíleno využitím komponenty `Validator`. Pomocí této komponenty je s využitím anotace každému atributu google feedu nastaveno omezení, které musí být pro srovnávače i samotný feed splněno. Problémem je skutečnost, že každý srovnávač má svá vlastní omezení a povinné elementy. Rozlišení je docíleno pomocí parametru **GROUPS**, který definuje pole srovnávačů, které dané omezení musí dodržet.

Příkladem může být například atribut `product_type` v kódovém bloku 5.2.

Zdrojový kód 5.2: Ukázka formátu vstupního souboru

```
#[Assert\NotBlank(
        message: 'Product type can not be blank!',
        groups: [
            KelkooConstants::GROUP_NAME_REQ,
            GlamiConstants::GROUP_NAME_REQ,
            CompariConstants::GROUP_NAME_REQ,
        ]
    )]
    #[Assert\Length(
        max: self::GLAMI_CONSTS[CATEGORY_LENGTH],
        maxMessage: 'Product type can not be longer than {{ limit }} characters!',
        groups: [
            GlamiConstants::GROUP_NAME,
        ],
    )]
    private ?string $product_type = null;
```

Zde lze vidět, že tento atribut musí být povinně neprázdný pro srovnávače Kelkoo, Glami a Compari. Fakt, že se jedná o povinný atribut lze vypozorovat z použitého klíče **GROUP_NAME_REQ**, který byl definován v kapitole 5.2.2.

Dále si můžeme všimnout, že je zde vyobrazeno omezení, které říká, že délka tohoto atributu u srovnávače Glami nesmí přesáhnout délku, která je definována v poli omezení, které bylo také popsáno v kapitole 5.2.2. V tomto případě se jedná pouze o nekritickou chybu, jelikož byl použit klíč **GROUP_NAME** a nikoliv **GROUP_NAME_REQ**. To že je chyba nekritická znamená, že i když k ní dojde, feed bude stále považován za validní a bude jej možné vygenerovat.

[4] https://support.google.com/merchants/answer/7052112?hl=en

Tímto způsobem jsou oanotovány skoro všechny atributy třídy `Item` a pro validaci následně stačí zavolat funkci `validate()`, která jako parametry přijme data, která chceme validovat a jméno skupiny, pro kterou validaci chceme uskutečnit. Pro příklad pokud chceme zjistit všechny kritické chyby pro srovnávač Glami, zadáme funkci parametr **GlamiConstants::GROUP_NAME_REQ**.

Pro každý cenový srovnávač je zde vytvořená třída. Ta také, jako v případě google feedu, obsahuje atributy, které odpovídají atributům definovaným ve specifikaci daného srovnávače. Do instancí těchto tříd budou přemapována data google feedu. Díky této operaci dojde k transformaci dat do podoby, která již odpovídá formátu daného srovnávače. Tyto instance budou dále použity pro vytvoření výsledného xml feedu pro daný srovnávač.

Mimo třídy srovnávačů a google feedu jsou zde definovány tři poslední třídy.

GoogleProductCategory

Tato třída reprezentuje produktovou kategorii google feedu. Aplikace obsahuje databázi se všemi existujícími kategoriemi google merchant feedu. Její hlavní využití je pro kontrolu, zda uživatelem zadaná kategorie skutečně existuje. Hodnoty do databáze byly nahrány z oficiálního datového souboru[5] poskytnutého Googlem merchant centrem. Ukázku z něj můžeme vidět v bloku 5.3. Jedná se vždy o unikátní identifikátor a k němu příslušný textový popis reprezentující skutečnou kategorii.

Zdrojový kód 5.3: Ukázka souboru s kategoriemi

```
1 - Animals & Pet Supplies
3237 - Animals & Pet Supplies > Live Animals
2 - Animals & Pet Supplies > Pet Supplies
3 - Animals & Pet Supplies > Pet Supplies > Bird Supplies
7385 - Animals & Pet Supplies > Pet Supplies > Bird Supplies > Bird Cage Accessories
```

MessagesToProcess

Jelikož každý pracovník pracuje asynchronně, nemá informaci o aktuálně zpracovávaném feedu. Konkrétně jestli se jedná o poslední uživatelův feed. Je tedy nutné zajistit, aby pracovník věděl, že generuje poslední feed, a že má následně odeslat všechny vygenerované výsledky uživateli na e-mailovou adresu.

K tomuto účelu je vytvořena tabulka v databázi, kde se pod uživatelův specifický vygenerovaný hash uloží počet feedů, které čekají na zpracování. Každý pracovník po zpracování od tohoto čísla odečte jedničku. Jakmile pracovník

[5] https://www.google.com/basepages/producttype/taxonomy-with-ids.en-US.txt

zjistí, že zbývá pouze 1 feed na zpracování, zpracuje ho, odešle výsledky a daný záznam z tabulky smaže.

FeedGenerator

Instance této třídy představuje objekt, který se bude předávat do fronty zpráv pro zpracování. Jedná se tedy o **zprávu**, která byla popsána v kapitole 4.3.1. Tento objekt bude obsahovat informaci o uživatelově unikátním hashi a odkaz na cache paměť ,ve které jsou uloženy jeho zpracovaná vstupní data, která se využíjí k přemapování do cílových srovnávačů.

5.2.4 Services

Service srovnávačů

Každý srovnávač má definovanou svoji vlastní service třídu s jedinou metodou. Tato metoda slouží k přemapování atributů z instance google feedu do instance daného srovnávače. Jedná se tedy o mapovací funkci, na kterou se odkazovalo v kapitole 5.2.2. Funguje na velmi jednoduchém principu. V prvním kroku se vytvoří nová instance cílového srovnávače. Následně se volají postupně všechny `setter` metody daného srovnávače s parametry získaných z google feed instance. Každý `setter` srovnávače už v sobě má podmínky, které danou hodnotu zkontrolují a případně upraví do správného formátu. Například zkrátí délku textu pokud je moc dlouhý.

FormattingService

Service je určena k naformátování chyb, které vznikly validací jednotlivých atributů feedu. Validace byla popsána v kapitole 5.2.3. Po validaci jsou navráceny všechny vzniklé chyby. Ty jsou pomocí této service metody zformátovaný do podoby, se kterou se na straně klienta jednoduše pracuje. Z každé vygenerované chyby jsou během zpracování extrahovány následující data.

- **message** - Jedná se o chybovou zprávu. Tuto zprávu lze nadefinovat použitím klíče `message` v anotaci atributu.
- **code** - Kód chyby.
- **propertypath** - Celá cesta ke špatně zadané hodnotě.
- **invalidValue** - Špatně zadaná hodnota.

K těmto datům se přidá informace o počtu závažných chyb a celý objekt se pošle do uživatelského rozhraní pro zobrazení. Úplný formát je popsán json schematem[6] v kódovém bloku 5.4.

[6] Deklarativní jazyk umožňující anotovat a validovat JSON dokumenty.

Zdrojový kód 5.4: Ukázka formátu zpracovaných chyb

```
data: arrayOf(
    shape({
        name: string.isRequired,
        data: shape({
            errors: arrayOf(
                shape({
                    message: string.isRequired,
                    code: string.isRequired,
                    propertyPath: string.isRequired,
                    severity: string.isRequired,
                    invalidValue: string,
                })
            ),
            required: bool.isRequired,
        }),
    })
),

```

Zde lze tedy vidět, že se jedná o objekt **data**, jehož hodnotou je pole objektů. Každý tento objekt má tyto hodnoty:

- **name** - Jedná se o jméno cílového srovnávače.
- **data** - Objekt který obsahuje informace o chybách.
- **required** - Booleanovská[7] hodnota označující, zda daný srovnávač obsahuje kritické chyby či nikoliv.

Výše zmíněná hodnota **data** se zkládá z objektu, jenž obsahuje klíč **errors**, pod kterým je uloženo pole objektů reprezentující chyby. Formát těchto objektů byl definován výše.

FeedService

Třída určená k práci s google feedem. Nabízí dvě metody.

- **processGoogleFeed** - Po přečtení uživatelem nahraného souboru je potřeba všechny data jednotlivých položek zboží zpracovat a naplnit jimi instance třídy `Item`. Ta jak již bylo řečeno v kapitole 5.2.3 reprezentuje samotný google feed. Tato operace se provádí v této metodě za použití symfony komponenty `Serializer`.
- **generateCompXmlFeeds** - Poté co uživatel odeslal požadavek pro generování výsledných xml feedů dojde ihned k několika operacím. Nejprve je nutné přidat do databázové tabulky **MessagesToProcess** nový

[7] Hodnota, který nabývá pouze dvou hodnot. Pravda nebo Nepravda.

záznam. Tato informace je důležitá pro asynchronně pracující worky, aby věděli jestli se jedná o poslední uživatelův feed a případně mu odeslali e-mail s výsledky. Dále dojde k serializaci dat podstatných pro generování. Tedy přesněji uživatelův hash a data specifická pro daný srovnávač. Tyto zprávy jsou přidány do fronty zpráv, kde čekají než budou zpracovány pracovníkem s volnými zdroji.

Pomocné service

Tyto services jsou určeny pro úpravy výsledného xml feedu. Pro příklad service **UpperCaseNameConverterService** převede všechny jména elementů do velkých písmen.

5.2.5 Controller

Jedná se o metody, které přijímají HTTP požadavky a navracejí HTTP odpovědi. Jsou namapovány na určitou URL adresu. Tyto adresy lze jednoduše provolávat z uživatelského rozhraní. V aplikaci je nadefinováno celkově pět endpointů.

index

Vrací základní šablonu `index.html.twig`. Tato šablona bude popsána v kapitole 5.2.8. URL adresou je `/`.

analyseFeed

Tato routa je využita k analýze vstupního souboru. V prvním kroku dojde k přečtení soubor a jeho zpracování.. Poté bude zvalidován. Jestliže jsou nalezeny kritické chyby, endpoint navrátí všechny chyby. Pokud byl počet kritických chyb nulový, zvaliduje se feed vůči všem ostatním srovnávačům a výsledky se odešlou zpět. Současně s touto operací dojde k vytvoření unikátního uživatelova hash klíče. Pod ten se v cache paměti uloží uživatelův zpracovaný soubor. URL adresou je `/analyse/google-feed`.

generatePriceComp

Na straně klienta bude uživateli umožněno vybrat si cílové srovnávače, pro které bude chtít výsledné feedy vytvořit. Tyto srovnávače budou zpracovány a následně s použitím **FeedService** odeslány do fronty pro budoucí generování. URL adresou je `/generate/price-comparators`.

downloadFeed

Jakmile jsou feedy vytvořeny, jsou uloženy ve složce, jejímž jménem je uživatelův unikátní hash. V e-mailové schránce následně uživatel nalezne e-mail s klikatelnými odkazy, přes které tuto routu provolá a výsledky si stáhne. URL adresou je `/download/{folderName}/{fileName}`, kde **folderName** je již zmíněný hash a **fileName** je jméno souboru, který chce stáhnout.

nonExistentRoute
: Po zadání neexistující URL, přesměruje uživatele zpět na hlavní stránku.

5.2.6 Handlers

Obsahují jedinou třídu **FeedGeneratorHandler**. Jedná se o třídu obsahující funkcionalitu, kterou bude vykonávat každý pracovník. V tomto případě se jedná tedy o zpracování zprávy, které bylo definováno v kapitole 4.3.1. Každý pracovník si v prvním kroku načte z cache paměti zpracovaná uživatelovo data. Tyto data přemapuje do cílového srovnávače a pomocí komponenty **Serializer** vygeneruje výsledný feed. Ten poté uloží do složky označené hashem uživatele. Každý pracovník si z fronty požadavků načítá jeden srovnávač. Jakmile pracovník zpracovává poslední uživatelův feed, po jeho dokončení odešle uživateli zpět e-mail s odkazem na tyto nově vygenerované soubory. Následně data z cache paměti odstraní.

5.2.7 DataFixtures

Pro validaci atributu **GoogleProductCategory** je nutné znát všechny možné produktové kategorie. Ty se dají stáhnou zdarma z internetu. Pomocí třídy **GoogleProductCategoryFixtures** je soubor obsahující všechny kategorie přečten a jednotlivé položky se uloží do databázové tabulky **GoogleProductCategory**.

5.2.8 Templates

Aplikace obsahuje pouze jedinou šablonu jejíž název je `index.html.twig`. Tato šablona obsahuje pouze základní html, importy CSS a JS a jediný `div` s identifikátorem `root`. Tento div bude sloužit jako rodičovský prvek, který bude nadále zpracováván pomocí `React DOM`.

5.3 Ostatní soubory

V této kapitole budou popsány pouze některé konfigurační soubory a soubory, které jsou důležité pro správnou funkcionalitu aplikace.

docker-compose.yaml
: Pomocí docker-compose je možné definovat kontejnery a specifikovat jejich vzájemnou interakci. V této aplikaci byl využit pro nastartování redis serveru a mailhogu[8].

webpack.config.js
: Konfigurační soubor pro webpack.

[8] E-mailový testovací nástroj.

.nvrmc
: Soubor používaný nástrojem pro správu verzí node. Slouží k jednoduchému přepnutí verze node v aktuálním projektu.

package.json
: Obsahuje metadata týkající se projektu. Tedy v případě této bakalářské práce, uživatelského rozhraní. Dále obsahuje skripty a všechny závislosti aplikace.

composer.json
: Funkčností prakticky stejný jako package.json. Rozdílem je, že je použitý pouze pro PHP a nikoliv Javascript.

.env
: Konfigurační soubor určený pro uložení citlivých a speciálních dat. Těmi mohou být například hesla nebo klíče do databáze.

5.4 Pomocné nástroje

Jedná se o nástroje, které slouží k formátování kódu a k jeho celkové udržitelnosti a čitelnosti.

.prettier
: Nástroj automaticky formátující kód do podoby, definované v konfiguračním souboru. V aplikaci použit pro práci s Javascriptem. V kódovém bloku 5.5 je možné vidět kofiguraci, jenž byla použitu pro tvorbu této aplikace.

Zdrojový kód 5.5: Ukázka konfiguračního souboru pro prettier

```
1   module.exports = {
2     trailingComma: 'es5',
3     useTabs: true,
4     tabWidth: 4,
5     semi: false,
6     singleQuote: true,
7     jsxSingleQuote: true,
8     insert_final_newline: true,
9     bracketSpacing: true,
10    bracketSameLine: true,
11    arrowParens: 'always',
12    printWidth: 100,
13  }
14
```

ecs
: Nástroj nutící v PHP dodržet určitý standard psaní kódu. Jako další funkcionalitu nabízí detekování a opravu špatného kódu.

.editorconfig

Pomáhá s udržením konzistentního kódu v celém projektu. Svá pravidla nastavuje ještě před zapsáním libovolného kódu. Ukázkovým příkladem může být nastavení tabulátoru. V konfiguračním souboru se nastaví odsazení tabulátoru na dvě mezery. Následně kdykoliv bude v projektu použit tabulátor dojde k odsazení přesně dvě mezery.

Testování 6

Po vytvoření výsledné aplikace ji bylo nutné adekvátně otestovat. Primárně zvalidovat vygenerované xml feedy. Dále také zkontrolovat, zda se program chová v každém případě tak, jak je očekáváno. V této sekci bakalářské práce bude popsán postup ověření kvality software. Aplikace byla testovaná v prohlížečích Google Chrome a Microsoft Edge.

Asi nejdůležitější částí testů byla kontrola výsledných feedů. Zde se vyskytuje problém. Tím je, že stránky, pro které se výsledné feedy generují, se mění a přizpůsobují moderním trendům. Pro příklad před pár lety bylo možné si při výběru pohlaví u produktu vybrat pouze ze dvou možností. Muž nebo žena. To už v roce 2023 neplatí a výběrových možností je mnohem více a firmy se tomu snaží přizpůsobit. Z tohoto důsledku je jasné že nelze vymyslet testy, které by stroprocentně validovali feedy, protože může dojít každým dnem k jejich zmeně. Na tuto změnu by měl samozřejmě programátor reagovat a testy či aplikaci aktualizovat a opravit, aby byla stále relevantní a poskytovala validní data. Feedy byly testovány dvěma způsoby k únoru roku 2023.

Testování podle dokumentace

V této části byly výsledky generování porovnány s oficiální dokumentací daného srovnávače. Testovány byly jak názvy elementů, tak jejich hodnoty.

Testování pomocí aplikace Mergado

Aplikace Mergado nabízí zdarma audit feedu. Jediným problémem je, že se bohužel soubory, které chceme zvalidovat, nedají nahrát klasickým způsobem z lokálního souborového systému uživatele. Je tedy nutné je nahrát na vzdálený server a poskytnout aplikaci jejich URL adresu. K tomuto úkonu bylo firmou Retailys poskytnuto vzdálené úložiště na cloudově založeném úložišti nazývaném Microsoft azure storage.

Po validaci na Mergadu byly objeveny některé chyby a vzápětí byly opraveny. Jak ale bylo již zmíněno dříve, stránky srovnávačů svá specifika často mění a samotné Mergado občas obsahovalo zastaralé informace. Tohoto poznatku

bylo docíleno po porovnání navrácené chyby s aktuální oficiální dokumentací daného srovnávače.

Dále bylo potřebné staticky analyzovat kód uživatelského rozhraní a serveru. K tomuto účelu se využili již existující nástroje, které naskenují veškerý zdrojový kód a naleznou potencionální chyby a neošetřené větvě, kterých si programátor nemusel všimnout.

Server
Pro tento úkol byl použit nástroj PHPStan. Pomocí něj byla zkontrolována veškerá syntaxe, potencionální buggy, neshoda datových typů a mnoho dalšího.

Uživatelské rozhraní
Stejně jako u serveru, bylo nutné obdobným způsobem zkontrolovat i kód uživatelského rozhraní, jenž byl implementován v Javascriptu a nebylo tedy možné použít phpstan. K tomuto účelu byl využit Eslint.

6.1 Testování zátěže

Aplikace má smysl pouze v tom případě, kdy dokáže poskytovat data v omezeném a rozumném čase. Proto je nutné zkontrolovat, jak se aplikace chová pokud má pracovat s větším souborem. Jak již bylo řečeno dříve, aplikace k rovnoměrnému rozložení zátěže využívá asynchronně pracující **worky**. V této části budou otestovány různé scénáře s různým počtem pracovníků. K tomuto úkolu byl uměle vytvořen vstupní feed s 10000 produkty.

Pokus	Čas (sekundy)
1	10.6401
2	10.7037
3	10.1256
4	10.7237
5	10.4688

Tabulka 6.1: Tabulka zobrazující časy generování s použitím 1 pracovníka

V tabulce 6.1 můžeme vidět změřené časy generování výsledných XML feedů za využití jednoho pracovníka. V tomto případě byl průměrný čas **10.532** sekund. V tabulce 6.2 je vidět stejný test s použitím pěti pracovníků. Průměrný čas nyní činil **3.843** sekund. Můžeme tedy vidět, že došlo až k několikanásobnému navýšení rychlosti.

Pokus	Čas (sekundy)
1	3.3503
2	4.4549
3	3.8320
4	3.8053
5	3.7723

Tabulka 6.2: Tabulka zobrazující časy generování s použitím 5 pracovníků

6.2 Testovací scénáře

Jedná se o klíčovou část vývojového cyklu pomocí, které lze ověřit kvalitu výsledného produktu. Testovácí scénáře jsou definovány seznamem kroků, které by měl uživatel v aplikaci provést, aby se ověřila její funkčnost. Jsou psané tak, aby je pochopil i technicky nezdatný jedinec. Dají se dělit podle zaměření do několika kategorií. Nejznámější jsou.

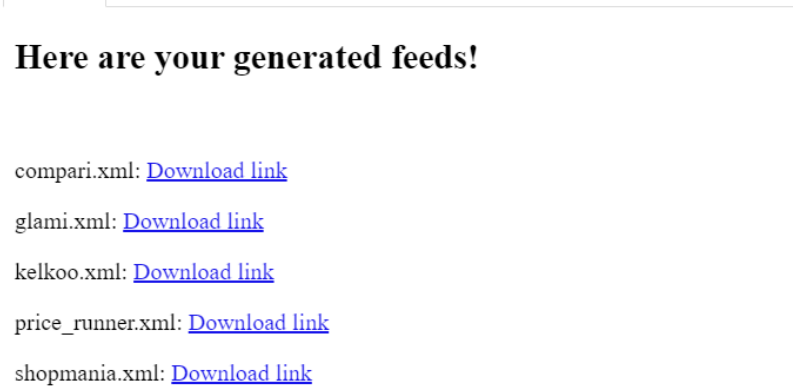
- **Pozitivní testy** - Ověřují očekávané chování při validním používání aplikace.
- **Negativní testy** - Kontrolují, zda se aplikace chová správně i při chybném používání aplikace. Tyto scénáře mají za úkol najít potenciální chyby.
- **Regresní testy** - Kontrolují, zda nově přidané změny nezpůsobily chyby v již ověřených funkcích.

V této kapitole otestujeme vytvořený nástroj pro generování feedů za použití pozitivních a negativních testovacích scénářů.

Validní soubor bez chyb

1. Klikněte na tlačítko **UPLOAD**.
2. Vyberte ze souborového systému validní soubor.
3. Vyberte srovnávače, pro které chcete vygenerovat výsledné xml feedy.
4. Klikněte na tlačítko **GENERATE**.
5. Zkontrolujte zda mailová adresa obsahuje nový e-mail s výsledky. Ten by měl obsahovat pouze ty výsledky, které jste označili pro generování. Na obrázku 6.1 můžeme vidět obsah mailu po vygenerování feedů pro všechny implementované srovnávače.

6. V mailu klikněte na odkazy a zkontrolujte, že jste byli přesměrováni a stáhl se vám daný feed. Každý odkaz je určený ke stáhnutí určitého feedu.
7. Otevřete stáhnutý soubor a zkontrolujte, že obsahuje data.



Obrázek 6.1: Ukázkový email

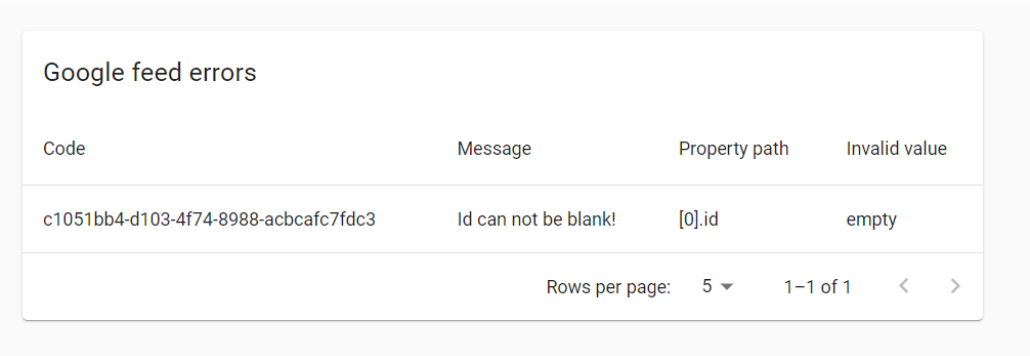
Validní soubor s chybami v datech

1. Klikněte na tlačítko **UPLOAD**.
2. Vyberte ze souborového systému soubor s kritickými chybami v datech.
3. Vyberte srovnávače, pro které chcete vygenerovat výsledné xml feedy.
4. Zkontrolujte, zda se zobrazuje tabulka s očekávanými google feed chybami.

Příklad: pokud nevyplním u prvního produktu povinnou položku **id**. Měla by se zobrazit tabulka, která mi tuto skutečnost sdělí. Ukázka na obrázku 6.2.

Nevalidní soubor

1. Klikněte na tlačítko **UPLOAD**.
2. Nahrajte nevalidní soubor.
3. Zkontroluje, zda se zobrazuje chybová stránka.
4. Zkontrolujte, zda aplikace nezamrzla a je stále funkční. (Tedy dá se vybrat nový soubor a je možné pokračovat v používání aplikace.)



Obrázek 6.2: Ukázka chyby v id produktu

Závěr 7

V rámci této bakalářské práce byla provedena analýza evropských cenových srovnávačů zboží. Ta byla popsána v kapitole 2. Dále byly prozkoumány technologie a postupy jakými vygenerovat na jeden průchod data pro co největší množství cílových srovnávačů.

Následně byla vytvořena kompletní webová aplikace, pomocí které je možné velmi jednoduše převádět vstupní soubory do formátů implementovaných evropských cenových srovnávačů zboží. Její implementace byla stručně popsána v kapitole 5.

Bylo nutné se seznámit s různými srovnávači a důkladně si nastudovat jejich unikátní specifikace. Dále bylo zapotřebí vymyslet vhodný způsob jakým data zpracovat a transformovat a následně zpátky předat uživateli. Neméně důležité bylo vybrání správné technologie pro vytvoření uživatelského rozhraní a rozhodnutí jakým způsobem a jakými prostředky uživateli veškerá data, například chyby feedu, zobrazit.

Serverová část byla implementována v jazyce php verze 8.0.10 s použitím velmi robustního a propracovaného frameworku Symfony. Uživatelské rozhraní se vytvořilo pomocí moderní a populární knihovny React a grafické knihovny material UI. V aplikaci se ve velké míře používají novodobé přístupy a technologie a je kladen důraz na efektivitu při zatížení serveru. Aplikace byla náležitě otestována a vygenerované výsledky zkontrolovány několika možnými způsoby.

Výsledkem je funkční aplikace s moderně vypadajícím uživatelským rozhraním, která umožňuje zákazníkům jednoduše transformovat data, díky čemuž můžou své produkty inzerovat na různých srovnávačích, a tím si zvýšit šanci prodeje. Zadání bylo splněno, ale jednoznačně se nabízí neomezené množství možností, jakými by se aplikace dala vylepšit. Velmi zajímavých vylepšením by mohlo být přidání editoru, pomocí kterého by měl uživatel možnost vygenerované chyby jednoduše opravit.

Uživatelská dokumentace

Tato sekce popisuje postup jakým aplikaci lokálně spustit. Před samotným spuštěním je nutné se ujistit že jsou stáhnuty a nainstalovany všechny prerekvizity. Těmi jsou:

- node verze 16.13.0. Pokud máte nainstalován **node version manager** stačí použít příkaz `nvm use`.
- php 8.0.10
- docker
- mysql server
- composer 2.1.9
- symfony 5.4.13

A.1 Spuštení

Nejdříve je nutné stáhnout celý projekt a uložit ho do libovolného adresáře. Dalším krokem je instalace závislostí. Přejdeme do adresáře `mvbk` a napíšeme tyto příkazy.

- **npm install** - Javascriptové závislosti.
- **composer install** - PHP závislosti.

Jakmile jsou nainstalovány všechny závislosti, je zapotřebí spustit docker. Toho se docílí příkazem `docker-compose up`. Poté je potřeba naplnit MYSQL databázi daty. K tomu se využije příkaz `php bin/console doctrine:fixtures:load`. Přístupové údaje k databázi jsou vidět v souboru `.env`.

V tomto okamžiku je již aplikace připravená a zbývají 2 poslední příkazy. Ke spuštění samotné aplikace stačí zadat `symfony server:start -d`. Nyní máme již spuštěnou aplikaci, která je přístupná z webového prohlížeče na adrese **http://localhost:8000/**. V posledním kroku již stačí jen zapnout pracovníky, kteří

budou požadavky na generování zpracovávat. Toho můžeme dosáhnout dvěma způsoby.

Konfigurace procesového správce
: Pro správu více pracovníků najednou je možné si nakonfigurovat procesové správce, kterým je například **supervisor** nebo **systemd**. Ukázku konfigurace **supervisora** můžeme najít zde: *https://symfony.com/doc/current/messenger.html#supervisor-configuration*.

Ruční zapnutí
: Otevřeme nový terminál a zadáme následující příkaz. php `bin/console messenger:consume async`. Tímto spustíte jednoho pracovníka. Proces můžeme opakovat podle toho, kolik pracovníků chcete mít aktivních.

Struktura přiloženého souboru

B

- `A20B0268P_prilohy.zip` ..... Soubor zip
- `Text_prace`
  - `img` ..... Obrázky bakalářské práce
  - `bp.tex` ..... LaTeX zdrojový kód
  - `bp.pdf` ..... Výsledný text bakalářské práce
- `Aplikace_a_knihovny`
  - `generated_doc` ..... Soubory vygenerované dokumentace
  - `src` ..... Zdrojové kódy
  - `lib` ..... Knihovny třetích stran
  - `build.md` ..... Návod na lokální spuštění aplikace
- `Vstupni_data`
  - `data` ..... Adresář obsahující vstupní testovací soubory
- `Vysledky`
  - `generated_feeds` ..... Příklady vygenerovaných feedů
  - `time_test.xlsx` ..... Data testu běhu aplikace
- `Readme.txt` ..... Struktura přiloženého souboru

Bibliografie

[ČSO21] ČSOB. *22. díl: Proč jsou srovnávače zboží pro e-shopy důležitým marketingovým nástrojem?* 2021. Dostupné také z: https://www.pruvodcepodnikanim.cz/clanek/srovnavace-zbozi-pro-e-shopy/. [Online; Accessed 2 April. 2023].

[Doc23a] DOCS, MDN Web. *What is JavaScript?* 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Online; Accessed 17 April. 2023].

[Doc23b] DOCS, MDN Web. *XML introduction*. 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction. [Online; Accessed 17 April. 2023].

[Dun11] DUNLU PENG† Lidong CAO, Wenjie XU. *Using JSON for Data Exchanging in Web Service Applications*. 2011. Dostupné také z: https://www.researchgate.net/profile/Dunlu-Peng/publication/265874991_Using_JSON_for_Data_Exchanging_in_Web_Service_Applications/links/5523cd1f0cf2c815e07325ea/Using-JSON-for-Data-Exchanging-in-Web-Service-Applications.pdf. [Online; Accessed 1 May. 2023].

[Gac15] GACKENHEIMER, Cory. *Introduction to React*. 2015. Dostupné také z: https://books.google.cz/books?hl=cs&lr=&id=NZCKCgAAQBAJ&oi=fnd&pg=PR6&dq=react&ots=KBzsQsGz-f&sig=Qb-afV3UoMwfb1jM57Relzrjk9Q&redir_esc=y#v=onepage&q=react&f=false. [Online; Accessed 1 May. 2023].

[Hom21] HOMOCIANU, Daniel. *Front-End Frameworks for Development of Spa and Mpa Web Applications*. 2021. Dostupné také z: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3987838. [Online; Accessed 20 April. 2023].

[Chr20] CHRISTOPH, Arthur. *Vue, Angular, React Comparison Series: computed property*. 2020. Dostupné také z: https://dev.to/achristoph/vue-angular-react-comparison-series-computed-property-5c8p. [Online; Accessed 1 May. 2023].

[Kai14] KAI LEI Yining Ma, Zhi Tan. *Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js*. 2014. Dostupné také z: https://ieeexplore.ieee.org/abstract/document/7023652. [Online; Accessed 1 May. 2023].

[Kel22] KELKOOGROUP. *Our locations - Kelkoo Group*. 2022. Dostupné také z: https://www.kelkoogroup.com/about-us/locations/. [Online; Accessed 25 April. 2023].

[kex15] KEXUGIT. *ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET*. 2015. Dostupné také z: https://learn.microsoft.com/en-us/archive/msdn-magazine/2013/november/asp-net-single-page-applications-build-modern-responsive-web-apps-with-asp-net. [Online; Accessed 20 April. 2023].

[Koď20] KOĎOUSKOVÁ, Barbora. *Vývoj webových aplikací: single-page vs. multi-page aplikace*. 2020. Dostupné také z: https://www.rascasone.com/cs/blog/jednostrankove-vicestrankove-web-aplikace. [Online; Accessed 20 April. 2023].

[Laf08] LAFFEY, Des. *Value Configurations in E-Commerce: Evidence from Comparison Websites*. 2008. Dostupné také z: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1265&context=ecis2008. [Online; Accessed 20 April. 2023].

[Log23] LOGOLOOK. *Amazon Logo*. 2023. Dostupné také z: https://logolook.net/amazon-logo/. [Online; Accessed 19 April. 2023].

[Mer23] MERGADO. *Audit feedu zdarma*. 2023. Dostupné také z: https://audit.mergado.com/?_gl=1*1ebvgbr*_ga*MTc2MjY2ODI1OC4xNjY0OTE1Njg3*_ga_5JED3N4VLV*MTY2NjU5OTEwNi4yLjAuMTY2NjU5OTEwNi4wLjAuMA.. [Online; Accessed 2 May. 2023].

[Sym23a] SYMFONY. *Projects using Symfony*. 2023. Dostupné také z: https://symfony.com/projects. [Online; Accessed 1 May. 2023].

[Sym23b] SYMFONY. *Symfony*. 2023. Dostupné také z: https://symfony.com/. [Online; Accessed 1 May. 2023].

[Sym23c] SYMFONY. *Symfony Components*. 2023. Dostupné také z: https://symfony.com/components. [Online; Accessed 1 May. 2023].

[Sym23d] SYMFONY. *The Cache Component*. 2023. Dostupné také z: https://symfony.com/doc/current/components/cache.html. [Online; Accessed 1 May. 2023].

[Sym23e] SYMFONY. *The Config Component*. 2023. Dostupné také z: https://symfony.com/doc/current/components/config.html. [Online; Accessed 1 May. 2023].

[Sym23f] SYMFONY. *The Filesystem Component*. 2023. Dostupné také z: https://symfony.com/doc/current/components/filesystem.html. [Online; Accessed 1 May. 2023].

[Sym23g] SYMFONY. *The Messenger Component*. 2023. Dostupné také z: https://symfony.com/doc/current/components/messenger.html. [Online; Accessed 1 May. 2023].

[Sym23h] SYMFONY. *The Serializer Component*. 2023. Dostupné také z: https://symfony.com/doc/current/components/serializer.html. [Online; Accessed 1 May. 2023].

[Sym23i] SYMFONY. *Validation*. 2023. Dostupné také z: https://symfony.com/doc/current/validation.html. [Online; Accessed 1 May. 2023].

[Štr10] ŠTRAUCH, Adam. *Redis: key-value databáze v paměti i na disku*. 2010. Dostupné také z: https://zdrojak.cz/clanky/redis-key-value-databaze-v-pameti-i-na-disku/. [Online; Accessed 3 May. 2023].

[Tho19] THOMAS, James. *What is Webpack?* 2019. Dostupné také z: https://levelup.gitconnected.com/what-is-webpack-4fdb624597ae. [Online; Accessed 3 May. 2023].

[Tut16] TUTORIALSPOINT. *PHP*. 2016. Dostupné také z: https://www.tutorialspoint.com/php/php_tutorial.pdf. [Online; Accessed 1 May. 2023].

Seznam obrázků

Seznam tabulek

Seznam výpisů