Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Grafický konfigurátor importu dat

Bakalářská práce

Autor: Jakub Urbanec
Studijní obor: Aplikovaná informatika

Jilemnice Srpen 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

……

V Jilemnici dne 16.8.2021 Jakub Urbanec

Poděkování:

Děkuji vedoucí bakalářské práce Mgr. Daniele Ponce Ph.D. za metodické vedení práce, odborné připomínky a cenné rady ke zpracování této bakalářské práce.

Anotace

Bakalářská práce si dává za cíl vytvoření grafického konfigurátoru dat pro běžného uživatele. Tento cíl podkládá aplikace bakalářské práce, ve které je možné data transformovat do jiné podoby. Aplikace bakalářské práce bude využívána především v systémech internetových obchodů, kde je nezbytně nutné dodržovaní ustálených forem dat, která jsou posílána do těchto systémů. Výsledkem této práce je aplikace, která transformuje data do ustálené podoby, jež je vymezena v bakalářské práci. Poté budou moci systémy internetových obchodů tato data využít ve svůj prospěch.

Annotation

Title: Graphical data import configurator

The main purpose of this Bachelor Thesis is to create a graphic data configurator for regular user. This purpose is supported by the Bachelor Thesis application itself in which it is possible to transfer the form of data. The Bachelor Thesis application will be used mainly in e-shop systems where it is absolutely necessary to keep the stable required form of the data which are being sent to said systems. The result of this thesis is an application which transforms data to stable required form defined in the Bachelor Thesis. These data can be then used to e-shop systems' advantage.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Bakalářská práce se zabývá problematikou zpracování dat internetových obchodech. Tyto obchody využívají často xml, json nebo csv souborů od dodavatelů, ve kterých se nacházejí všechny informace o produktech, které dodavatel dodává. Z tohoto vyplývá, že většina výrobců nemá ustálenou strukturu dat a tím pádem je těžší dané soubory zpracovávat. Zaměstnanci nebo vlastníci internetových obchodů mají poté problém data zapracovat nebo i vůbec zpracovat a musí data dále upravovat, aby výsledná podoba odpovídala řešení jejich systému pro internetový obchod. Tento problém je dále rozšířen v další kapitole bakalářské práce, která pojednává o e-shopu.

2 E-shop

Aplikace bakalářské práce bude vytvářena hlavně pro potřeby e-shopů. Není žádným problémem, že je na světě spousta různých typů e-shopů, od prodejců zvířat až po prodej zbraní. Aplikace k bakalářské práci bude usnadňovat zpracování dat kterémukoliv e-shopu, který bude mít možnost zpracovávat xml.

K čemu je e-shopům xml soubor? Takový soubor s daty, pokud je ucelený a obsahuje všechny potřebné elementy, které umí daný e-shop používat umožňuje pomoci v hlídání skladovosti, přidávání nových produktů, hlídání cen produktů, změn názvů produktů a dalších, ale vše se točí kolem produktů.

Co je to ale e-shop? Dle definic může být e-shop definován jako: „Pod pojmem elektronický obchod se obecně rozumí podnikání prostřednictvím elektronických prostředků. To zahrnuje nejen obchodování se zbožím (hmotným i nehmotným) a službami, ale i všechny související kroky od reklamy přes uzavření smlouvy a její plnění, a to včetně poprodejní podpory a služeb.“ (1) Nebo dle jiné definice může být též definován: „E-shop, neboli internetový obchod, je webová aplikace sloužící primárně k prodeji zboží. V nejjednodušší podobě jde o katalog produktů tříděný do kategorií a doplněný nákupním procesem.“ (2) Obě tyto definice vyjadřují pravý aspekt slova e-shop, pouze každá jiným způsobem. E-shop je internetový obchod, na kterém můžete prodávat cokoliv od zvířat po zbraně, pokud k tomu máte patřičná oprávnění. Ale jinak má e-shop podobná pravidla jako normální kamenný obchod, pouze tedy e-shop nemusí mít kamennou prodejnu, ale může mít jen adresu, na které je vedený, to znamená, že pro případné reklamace můžete na zadanou adresu poslat zboží.

V této době jsou e-shopy hojně používané z důvodů epidemie onemocněním COVID-19. Ale takový e-shop musí být připraven na řadu úskalí, která přináší. E-shopy jsou hodně napadány hackerskými útoky, často nezvládají nápor lidí, nebo nemají dobře zabezpečená data uživatelů. Také nedodržují v některých případech obchodní zákoník a podobně. Největším problémem e-shopů je jejich jednoduché vytvoření, z toho důvodu dnes existuje hodně podvodných e-shopů, které vás podvedou. Proti tomu se snaží bojovat velké e-shopy a další internetové stránky formou hodnocení

e-shopů a podobně. Mezi největší takovéto stránky patří ve světe google.com a v česku heureka.cz.

Mezi největší e-shopy na světě patří amazon.com a ebay.com. V České republice se mezi největší e-shopy řadí alza.cz, mall.cz a datart.cz. Největší světové e-shopy prodávají veškerý sortiment, oproti tomu v české republice jsou nejvíce oblíbenými e-shopy prodávající elektroniku. Toto má též za následek epidemie COVID-19.

Dále je problémem u e-shopů, že jsou velmi rozšířené, to znamená, že pokud něco chcete začít prodávat, již je velká pravděpodobnost, že to prodává někdo jiný a ve větším počtu nebo i levněji, z tohoto důvodu existují i srovnávače cen a podobně. Ale pokud jste vlastníkem e-shopu, tak nechcete produkty prodávat pod cenou nebo bez výdělku. V tomto směru pomáhá následující kapitola, o podobných řešeních, kde tyto aplikace vám pomohou se zpracování produktů, nebo i v případě Napojse o hlídání cen.

3 Podobná řešení

Mezi podobné aplikace, které budou mít podobné funkce jako aplikace bakalářské práce je aplikace s názvem Napojse. Tato aplikace je využívána spolu se systémem pro internetový obchod s názvem shoptet. Dalšími podobnými aplikacemi mohou být webové stránky, zabývající se přímo transformací dat do jiných typů dat, například transformace typu csv na typ xml.

3.1 Napojse

„Nastavíte si, na co si vzpomenete. Jednoduše a intuitivně.“ (3) Tato aplikace nabízí možnost od importu dodavatelských dat až po vybrání produktů, které chce uživatel importovat. Aplikace bakalářské práce se zabývá pouze první částí, neboli částí s importováním dat od dodavatele. Aplikace Napojse zahrnuje pouze dodavatele, kteří si zaplatí za uvedení jejich produktů do seznamu podporovaných dodavatelů přímo v této aplikaci. Poté si klient z internetového obchodu vybere dodavatele, které chce importovat do svého e-shopu a které produkty chce v tomto importu zahrnout. Vše je zpoplatněno a není automaticky vytvořeno, vždy musí programátor na pozadí nastavit vše potřebné a vytvořit potřebné transformace. Další podobnou aplikací je xemel.

3.2 Xemel

„Jednoduchý editor pro rychlou úpravu XML feedů“ (4) Jak je napsáno v definici, xemel je webová aplikace, která slouží k úpravě XML feedů bez potřeby znalostí programovacích jazyků. Tato aplikace má jednoduché rozhraní, ve kterém vypisuje elementy z XML feedu a poté tyto elementy zobrazí barevně zvýrazněné v seznamu. Do tohoto seznamu dále umožnuje přidávat další elementy a pravidla, která v daných elementech budou provádět další logiku, například spojení jména s kódem produktu. Dále nabízí zobrazení průběžných výsledků xml feedu. Též podporuje možnost úpravy více xml feedu najednou. Ale stejně jako webová aplikace napojse je xemel placený.

Oproti tomu část webových konvertorů je bez potřeby zaplacení částky a vše si vytváří uživatel sám. Z tohoto důvodu je zde možnost použití webových převaděčů.

3.3 Webové konvertory

Webové převaděče jsou oblíbené a existuje jich mnoho kdekoliv na internetu, dokážou transformovat převážně jakákoliv data do jiných dat. Stačí zadání dat do příslušného převaděče a poté během několika sekund se vytvoří přetransformovaná data v příslušeném typu, například v typu xml z typu csv. Velkým problémem těchto konvertorů je fakt, že nejsou nijak konfigurovatelné.

V této podkapitole a též v předchozí podkapitole o Napojse zazněly pojmy xml a csv. Tyto typy značkovacích jazyků využívají tyto webové stránky ke zpracování a zápisu dat. Též e-shopy využívají tyto značkovací jazyky. Z tohoto důvodu se v další kapitole podíváme na značkovací jazyky.

4 Značkovací jazyky pro potřeby e-shopů

E-shopy a ostatní aplikace využívají často značkovací jazyky k ukládání a posílání dat do jiných aplikaci, nebo též k přijímají tato data z jiných aplikací a zpracovávají je. V této kapitole budou popsány tři nejrozšířenější značkovací jazyky, prvním z nich je xml.

4.1 XML

Extensible Markup Language, česky rozšiřitelný značkovací jazyk je obecný značkovací jazyk, který je podmnožinou staršího značkovacího jazyku SGLM. XML je použito pro serializaci a zpracování dat pro uživatelskou nebo strojovou čitelnost.
Cílem značkovacího jazyku XML je jednoduchost, obecnost a použitelnost na internetu. Z těchto důvodu je v dnešní době hojně používaný a existuje spoustu programu na jeho zpracování pomocí XSL, XQuery, XProc a podobně.

4.1.1 Základní struktura XML

Značkovací jazyk XML se skládá z atributů a elementů, kde atributy poskytují doplňující informace k elementům.

```
<product>
   <ID>1234</ID>
   <title>Pánské boty</title>
   <code>PB1234</code>
   <description>Modré pánské boty velikosti 45.</description>
   <parameters>
      <parameter velikost="45"/>
      <parameter barva="modrá"/>
   </parameters>
</product>
```

Ukázka kódu 1 Jazyk XML; Zdroj: autor;

V ukázce kódu 1 můžeme vidět rodičovský element product, který obsahuje data od ID po parameters. Každý element obsahuje data, v případě elementu ID obsahuje informaci o ID 1234. Dále v ukázce vidíme použití pole, kde element parameters se bere v ukázce jako pole všech parametrů. V elementech parameter můžeme vidět použití atributů velikost a barva, které obsahují informaci o velikosti a barvě.

Po prvním značkovacím jazykem, následuje druhý značkovací jazyk, který je též velmi rozšířený a jednoduchý pro čtení, jde o formát csv.

4.2 CSV

Comma-separated values, česky hodnoty oddělené čárkami je jednoduchý souborový formát pro výměnu dat. Soubor ve formátu CSV je složen ze řádků, kde každá položka je oddělena znakem čárka, hodnoty položek mohou být odděleny uvozovkami. V některých typech CSV se používají i jiné znaky pro oddělení položek, jako jsou středník a tabulátor, v případě tabulátoru se poté nenazývá již soubor CSV ale TSV (Tab Separated values).

„Data uložená v takto formátovaném souboru se velmi jednoduše a nenáročně dají přenášet i bez dalšího složitého software mezi různými systémy. Stejný účel dnes plní i modernější, univerzálnější, ale pro přípravu již složitější formát XML.“ (5)

Němcová	Božena	Babička, Divá Bára
Čapek	Karel	Válka s mloky, RUR

Tabulka 1; Ukázka dat Zdroj: autor;

Data z tabulky 1 by poté vypadala zapsána ve formátu CSV takto:

```
Němcová, Božena, „Babička, Divá Bára“
Čapek, Karel, „Válka s mloky, RUR“
```

Ukázka kódu 2 Formát CSV; Zdroj: autor;

Posledním značkovacím jazykem, který patří dle autora mezi nejrozšířenější formát dat na internetu je json.

4.3 JSON

JavaScript Object Notation, česky JavaScriptový objektový zápis je způsob zápisu dat, který je určen pro přenos dat, která mohou být organizovaná v polích nebo agregovaná v objektech. Vstupem jsou libovolná data, výstupem je vždy řetězec.

Mezi výhody patří jeho obecnost, neboli může sloužit pro přenos dat jakémukoliv programovacímu nebo skriptovacímu jazyku. Oproti tomu jeho nedostatkem je nemožnost definovat znakovou sadu a optimalizace pro přenos binárních dat.

```
{"Příjmení":"Němcová","Jméno":"Božena","Díla":"Babička, Divá Bára"}
{"Příjmení":"Čapek","Jméno":"Karel","Díla":"Válka s mloky, RUR"}
```

Ukázka kódu 3 Formát JSON; Zdroj: autor;

Značkovací jazyk a formáty uvedené výše jsou zpracovány často webovými aplikacemi. Tyto aplikace jsou často jednoduché a jejich řešení není tolik komplexní. Pro vytváření webových stránek se používá velmi mnoho programovacích jazyků, ale nejrozšířenější programovací jazyky budou rozebírány v další kapitole.

5 Webové jazyky pro vytváření stránek

Jazyků pro vytvoření webových stránek je velké množství, ale v této kapitole se autor zaměří na pár programovacích jazyků, které jsou dle něho nejvíce rozšířené. Programovací jazyk, který zde nebude zahrnut, ačkoliv je velmi populární v tomto směru je php, jelikož autor je toho názoru, že tento programovací jazyk se spíše používá pro backend webových stránek a pro aplikace, než jen pro jednoduchou webovou stránku. Oproti tomu prvním programovacím jazykem, který se používá k vytváření webových stránek je html.

5.1 HTML

HTML neboli „Hypertext Markup Language“, v češtině „hypertextový značkovací jazyk“, je používán pro tvorbu obsahu webových stránek, které propojuje hypertextovými odkazy. Tento obsah mohou tvořit tabulky, texty, seznamy, obrázky, videa a další prvky.

„Každá informace má přesně definovaný význam, který vyjadřuje konkrétní prvek (element). Prvek definuje počáteční a koncová značka (tag). Počáteční značka může obsahovat atributy, které jsou tvořený příslušným názvem a hodnotou.“ (6) Základní šablonu HTML tvoří hlavička (head) a tělo (body). Hlavička HTML šablony může obsahovat titulek (title), nastavení formátu textu (meta charset) a při použití programovacího jazyku pro stylování (např. CSS) může poté obsahovat buď odkaz (link) na soubor CSS nebo můžou být styly zadáné přímo v hlavičce pomocí elementu „<style>“. V těle HTML šablony se nachází obsah webové stránky a často i různé skripty (script).

```
<!doctype html>
<html lang="cs">
   <head>
      <meta charset="utf-8">
      <title></title>
      <link rel="stylesheet" href="styly/style.css">
    </head>
    <body>
      <p id=“text1“> Lorem ipsum dolorem </p>
      <script src="js/script.js"></script>
   </body>
</html>
```

Ukázka kódu 4 Jazyk HTML; Zdroj: autor;

V ukázce kódu 4 je vytvořeno v hlavičce propojení mezi webovou stránkou a stylovacím souborem „style.css“. Dále je zde nastaven jazyk stránky na českou (lang) a typ textu na utf-8 (meta charset). V těle HTML šablony je vytvořen element p, obsahující text a tomuto elementu je přidán atribut id s názvem „text1“. Poté je webová stránka propojena v těle s JavaScriptovým souborem „script.js“. V této ukázce byl již okrajově použit další programovací jazyk s názvem JavaScript.

5.2 JavaScript

JavaScript je multiplatformním, objektově orientovaným a událostmi řízeným skriptovacím a programovacím jazykem, jenž se využívá v internetových stránkách. Skripty jsou zpracovány na straně klienta. Jeho syntaxe je podobná programovacím jazykům C, C++ a Java.

JavaScript se může zapisovat buď přímo do HTML kódu do elementu „<script>“ nebo propojit s JavaScriptovým souborem pomocí atributu „src“ v elementu „<script>“.

Ačkoli je JavaScript podobný programovacímu jazyku PHP, který je též využíván při tvorbě webových stránek, tak jejich hlavním rozdílem je jejich náročnost a rychlost. Důvodem je fungování Javascriptu na straně klienta, kde poté není server zatěžován, na rozdíl od programovacího jazyku PHP.

Javascript obsahuje spoustu frameworku, jako například React, Vue nebo Angular. Postupně se JavaScript přesunul i díky frameworkům na mobilní zařízení, kde ho využívají aplikace hlavně na systémech Android a iOS.

Kromě výhody v oblasti frameworků a rychlosti JavaScriptu je další výhodou integrace s programovacími jazyky HTML a CSS, které pomohou k ještě rychlejšímu načtení stránky. Jelikož je kód vykonáván na straně klienta, tak je zde další výhodou změna obsahu bez potřeby fyzického znovunačtení stránky.

```
<!DOCTYPE html>
<html>
   <body>
      <p id="text1"></p>

       <script>
 document.getElementById("text1").innerHTML = „Hello World!“;
       </script>

   </body>
</html>
```

Ukázka kódu 5 Jazyk JavaScript, HTML; Zdroj: autor;

Ukázka kódu 5 zobrazí v elementu p text „Hello World!“. Text je zobrazen díky atributu id „text1“, do kterého je poté nahrán text za pomoci metody „document.getElementById()“.

Webové stránky by ale vypadaly špatně bez dalšího programovacího jazyku, který se nazývá CSS a je možnost jej též použít v JavaScriptu, kde můžeme JavaScriptem nastavit nějakému elementu určitou vlastnost, například elementu p nastavit barvu pozadí na zelenou. Více již v následující kapitole o programovacím jazyku css.

5.3 CSS

CSS neboli „Cascading Style Sheets“, v češtině „kaskádové styly“, je metoda pro úpravu grafického zobrazení webových stránek. Tato metoda vznikla z důvodu možnosti oddělit obsahovou část dokumentu od vzhledové části.

Definice kaskádových stylů je tvořena z pravidel, tato pravidla obsahují selektory a bloky deklarací. V každém bloku deklarací je obsažena deklarace, která obsahuje informaci, jak se má daný element upravit.

```
body p {
    background-color: black;
    color: red;
}
```

Ukázka kódu 6 Jazyk CSS; Zdroj: autor;

V ukázce kódu 6 si upravíme element p, který se nachází v prvku body. Element p bude mít nastavenou červenou barvu textu a pozadí tohoto elementu bude nastaveno na černé.

Důležité při využívání souboru CSS je jeho deklarace v hlavičce stránky, pro kterou chceme daný CSS styl využít. Pro propojení souboru CSS se stránkou slouží element „<link>“. Druhou možností je přímé zapsaní CSS stylu do hlavičky.

```
<!doctype html>
<html lang="cs">
   <head>
      <meta charset="utf-8">
      <title></title>
      <style>
         #text1 {
            background-color: black;
            color: red;

         }
      </style>
    </head>
    <body>
      <p id=“text1“> Lorem ipsum dolorem </p>
   </body>
</html>
```

Ukázka kódu 7 Jazyk CSS, HTML; Zdroj: autor;

V ukázce kódu 7 je zadán styl přímo v hlavičce a využívá definice atributu id elementu p. Tedy v tomto případě kód změní u elementu p s id „text1“ barvu pozadí na černou a barvu písma na červenou. Dalším programovacím jazykem je jazyk xls, který není určený přímo na vytváření webových stránek, ale pomáhá těmto stránkám v oblastech získávání a zpracování dat.

5.4 XLS

„eXtensible Stylesheet Language (XSL) pracuje na zcela jiném principu než kaskádové styly. XSL je jazyk, kterým vytváříme pravidla pro transformaci jedné třídy XML dokumentů na jiný XML dokument. Výsledný dokument, který vznikne aplikováním transformačních pravidel, je použit pro zobrazení.“ (7)

Programovací jazyk xls je tedy výhodný, pro použití převodu značkovacích jazyků nebo formátů dat na značkovací jazyk xml. Programovací jazyk xls má podobnou strukturu jako značkovací jazyk xml. Tento programovací jazyk se často propojuje

s jazykem html, kde na straně serveru zpracovává data a umožňuje zobrazení xml souborů na všech typech prohlížečů.

5.4.1 Příklad xsl transformace

V příkladu bude ukázána xsl transformace textu značkovacího jazyka xml, který je vypsán níže a poté zpracovány hodnoty v elementech a vypsány v rámci výsledku.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<Article>
 <Title>My Article</Title>
 <Authors>
  <Author>Mr. Foo</Author>
  <Author>Mr. Bar</Author>
 </Authors>
 <Body>This is my article text.</Body>
</Article>
```

Ukázka kódu 8 Jazyk XML; Zdroj: (8);

Dále následuje xsl transformace:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:output method="text"/>

 <xsl:template match="/">
  Article - <xsl:value-of select="/Article/Title"/>
  Authors: <xsl:apply-templates select="/Article/Authors/Author"/>
 </xsl:template>

 <xsl:template match="Author">
  - <xsl:value-of select="." />
 </xsl:template>

</xsl:stylesheet>
```

Ukázka kódu 9 Jazyk XSL; Zdroj: (8);

Na ukázce kódu je vidět podobnost programovacího jazyka xsl se značkovacím jazykem xml. Transformace na ukázce kódu výše převádí všechny textové hodnoty obsažené v elementech do textové podoby, kterou dále vypisuje ve výsledku a přidává text Article na začátek výsledku a poté po jméně článku přidává daší text Authors, pod který jsou poté vypsáni autoři daného článku. Výsledkem transformace jsou tato data:

```
Article - My Article
Authors:
- Mr. Foo
- Mr. Bar
```

Ukázka kódu 10 Zdroj: (8);

Po vytyčení problému e-shopů s importem dat se zde nachází další kapitola, která již zahrnuje autorovo řešení tohoto problému. Tato kapitola o praktické části

obsahuje vše, co autor použije k pokusu o vyřešení problému, který dle jeho názoru je pro e-shopy docela zásadní.

6 Praktická část

Praktická část se zabývá celkovým vytvořením aplikace bakalářské práce, od jednotlivých požadavků až po implementaci aplikace. V každé praktické části se aplikace bakalářské práce, dále jen aplikace bude vyvíjet až do funkčního celku. Po vytvoření celé aplikace bude popsána a vyzkoušena základní funkcionalitu aplikace.

6.1 Požadavky na aplikaci

V této části budou rozepsány požadavky běžného uživatele a poté požadavky od vývojáře aplikace.

6.1.1 Požadavky uživatelů

Uživatelé, kteří budou tuto aplikaci převážně používat jsou vlastníci nebo pracovníci v e-shopech. Z tohoto důvodu jsou zde některé mírně specifičtější požadavky než jiné obecné. Pro uživatele, kteří pracují v e-shopech, nebo je vlastní, je hlavním požadavkem možnost úpravy dat, neboli zvolení dat, která si chtějí importovat skrze aplikaci. Dalším požadavkem je jednoduchá ovladatelnost a možnost znovupoužitelnosti. Posledním požadavkem je možnost vkládání jak souborů, tak i obyčejného textu.

6.1.2 Požadavky vývojáře

Vývojář této aplikace má požadavky hlavně na vizuální stránku, neboli zobrazení výsledných a vkládaných dat. Dále též jako uživatelé internetových obchodů na možnost znovupoužitelnosti a jednoduchou ovladatelnost. Dalším požadavkem, který ale není nutný, je možnost importu jiných typů souborů nebo textu, které nejsou značkovacího jazyka XML.

6.1.3 Shrnutí požadavků

Z výše vypsaných požadavků lze usoudit, že aplikace by měla být uživatelský přívětivá, neboli jednoduše ovladatelná, dále též vizuální zobrazení dat, které by mělo též vypadat přívětivě. Mezi dalšími požadavky je znovupoužitelnost, možnost vybrání pouze určitých dat k importu a poté vkládání dat pomocí textu nebo nahráním souboru. Požadavkem, který není nutný, ale je takzvaný „nice to have“ je

možnost vkládání více různých typů dat než jen typ XML. Toto jsou celkové požadavky na celou aplikaci.

6.2 Analýza

V předchozí kapitole byly vytyčeny požadavky na aplikaci, které by poté měla tato aplikace alespoň okrajově splňovat. V této části se autor podívá blíže na požadavky, které byly vypsány výše.

6.2.1 Vybrání dat k importování

Prvním z uživatelských požadavků je vybrání dat k importování. V tomto případě se jedná převážně o část, ve které uživatel již nahrál daná data. Po nahrání by tedy měl mít uživatel možnost nějakým způsobem vybrat nebo přiřadit data, která si přeje zahrnout do výsledku celé aplikace. Ale v tomto směru se nabízí otázka, zdali chce uživatel vybrat pouze část dat, například ID produktu a cenu, nebo chce vybrat například jen produkty, které jsou nějakého typu? Pokud by měl uživatel na mysli oba tyto případy, poté by to znamenalo v pokročilejší možnosti ovládání, větší komplexnost ovladatelnosti a též v těžší porozumění aplikaci. To by mělo za následek nedodržení dalších podmínek, které uživatel nebo vývojář určil. S tímto požadavkem tedy poté souvisí další požadavek, který mluví o jednoduché ovladatelnosti.

6.2.2 Jednoduchá ovladatelnost

Tento požadavek byl též určen v uživatelské části. Jak již bylo psáno v předchozím podtitulu s názvem vybrání dat k importu, je zde problém, jak si představit jednoduchou ovladatelnost? Znamená tato podmínka jít krok po kroku v aplikaci, kdy se vždy ztratí předchozí úsek a v aplikaci je vždy vyobrazen jen krok, který se má zrovna provést? Nebo naopak mít vše na jedné stránce, ve které se můžete kdykoliv vrátit k předchozímu kroku a například upravit data, která byla dříve vložena? Dle názoru autora, je tento požadavek spíše specifikován jako možnost jednoduchosti ovládání v tom směru, kdy uživatel vidí, co se děje a ví, co by mělo následovat a co se jíž stalo, neboli že by ho měla aplikace celým procesem provádět, ale vždy mít možnost se vrátit k předchozímu bodu. V tomto případě by tedy dávala smysl další podmínka, ve které je zmíněno, aby aplikace umožňovala vkládání dat více způsoby.

6.2.3 Více způsobů vkládání dat

Další podmínkou, která je zmíněna již výše je více způsobů vkládání dat. V tomto směru je daná podmínka relevantní, jelikož pracovník v internetovém obchodě jistě nechce stále vytvářet nové soubory, tahat z nich jen části kódu a podobně, nebo naopak jen kopírovat části ze souboru a nemoci tam nahrát celý soubor. Tato funkce je již v dnešní době rozšířena a z tohoto důvodu by jistě neměla v aplikaci chybět a též pomůže podmínkám v předchozím bodu, neboli jednoduché ovladatelnosti a též i vybrání dat k importu. Tento požadavek poměrně dost souvisí s dalším požadavkem, kterým je vizualizace dat.

6.2.4 Vizualizace dat

Požadavek k vizualizaci dat je v dané aplikaci jistě přínosem, jelikož se i podle zobrazení dat odvíjí spokojenost s celkovým dojmem celé aplikace. Z tohoto důvodu by jistě měl být na toto kladen důraz. Dle názoru autora je tento požadavek určen hlavně z důvodu uživatelské přívětivosti a též souvislosti s předchozími podmínkami. Jelikož kdo by chtěl používat aplikaci, která nedokáže ani správně zobrazit data? Odpovědí na tuto otázku jistě několik, ale dle názoru autora by to byl nikdo. K této i předchozím podmínkám se též pojí následující podmínka znovupoužitelnost.

6.2.5 Znovupoužitelnost

V tomto požadavku se jistě nabízí otázka: „Kdo by chtěl vytvářet aplikaci, která se použije jen jednou?“. Dle názoru autora by měla být každá aplikace vytvářena tak, aby bylo možné ji používat vícekrát nebo ji i později rozšířit. V tomto případě již díky předchozím podmínkám má velkou šanci aplikace být znovupoužitelná nebo i později rozšířena o další funkcionalitu. K znovupoužitelnosti jistě vybízejí i předchozí podmínky typu vybrání dat k importování nebo další podmínky, která pojednává o importu různých typů dat.

6.2.6 Import různých typů dat

Tento požadavek je není povinný, ale dle autorova názoru by jistě aplikaci dodal na popularitě a oblíbenosti mezi uživateli. Největším problémem tohoto požadavku je

určení, které typy dat se mají vůbec importovat, poté též vytvoření pro ně transformace a plně je zaintegrovat do celé aplikace. Tato funkcionalita by jistě potěšila nejednoho zaměstnance v internetovém obchodu.

6.3 Návrh

Po vytyčení podmínek, které by měla aplikace splňovat a jejich analýze je zde další kapitola, která předchozí dvě kapitoly spojuje a snaží se najít optimální návrh pro splnění všech podmínek. V první část této kapitoly bude o návrhu zpracování podmínek a poté následně v druhé části kapitoly bude návrh celé aplikace, kde budou zahrnuty podmínky, vypsané a zpracované výše.

6.3.1 Návrh zpracování podmínek

V této podkapitole bude popsán návrh zpracování jednotlivých podmínek v rámci celé aplikace a podle názoru autora nejlepší možnost, jak splnění těchto podmínek dosáhnout

6.3.1.1 Vybrání dat k importu

Tato podmínka má více významů, jak již bylo řečeno v analýze této podmínky. Z toho důvodu je možné zpracovat následující podmínku do mnoha různých návrhů. Dle názoru autora je nejjednodušším způsobem, jak zpracovat tuto podmínku zahrnout jí do dvou bodů.

V prvním bodě nám již pomůže jiná podmínka, jelikož není jednodušší řešení než data, která bude chtít uživatel vybrat si již vykopíruje ze souboru. V tomto případě to znamená, že pokud by uživatel chtěl například pouze prvních 10 produktů z nabídky výrobce, tak si může tato data jednoduše vykopírovat do textového pole a poté je zpracovat.

Druhým bodem je vybrání dat, která bude chtít uživatel vypsat na konci aplikace. Dle autora je v tomto případě jedním z nejjednodušších řešení zpracovat data do seznamu, ve které budou uvedeny elementy, které obsahují data uživatele a poté je přiřazovat k datům, která jsou již určena v aplikaci. V tomto směru se jedná o dva seznamy, kdy jeden obsahuje základní data aplikace, které se nemění a druhý seznam obsahuje uživatelská data, která se vždy mění podle obsahu. Poté pokud si data přiřadí, tak se hodnota těchto dat ve výsledném poli zobrazí na místě, kam byla přiřazena. S tímto souvisí další podmínka, která pojímá o jednoduché ovladatelnosti.

6.3.1.2 Jednoduchá ovladatelnost

Tato podmínka souvisí obecně s celou aplikací a je na každém, jak k této podmínce přistoupí. Z hlediska návrhu je nejlepší možností uživateli vždy zobrazovat další část funkcionality, poté co dokončí předchozí kroky, ale zároveň nechat předchozí kroky viditelné, aby nemusel uživatel v případě zájmu o změnu dat restartovat celou aplikaci a přijít tak například o postup. Jednoduché ovladatelnosti též přispěje další podkapitola, která je o způsobech vkládání dat.

6.3.1.3 Více způsobů vkládání dat

Zde je nejlepším řešením si nejprve zjistit, která data vlastně uživatel chce vkládat a podle toho mu zobrazit danou věc. Není nejlepším řešením nechat zobrazené jak textové pole, do kterého může zadávat data a nad tím mít možnost nahrát soubor. Toto jistě souvisí s předchozí podmínkou, kde by poté až po výběru, jak chce uživatel data zadat, by mělo být zobrazena daná možnost. V tomto směru by se jistě vše vyřešilo v případě souboru, nahrání ho do inputu a poté zpracovat tento soubor pomocí jQuery a posílat ho dále do aplikace. To samé by se hodilo i k textovému poli, kde by poté též data mohla být posílána a zpracována v jQuery. Data vyobrazená v textovém poli by mohla být nějak formátována, v tomto případě o této problematice pojímá další podkapitola o vizualizaci dat.

6.3.1.4 Vizualizace dat

Tato podkapitola byla již zahrnuta v jiných částech návrhu aplikace. V tomto případě z podmínek uvedených výše bude nejlepší možností vypsat data v textových polích a pomocí seznamů.

Data vypsaná v textových polích by byla ještě zvýrazněna díky dalším nástrojům, které by pomohli vizuální stránce jak dat, tak i celé aplikace. Též pro data v seznamu bude využita komponenta, která umožní hezčí zobrazení dat pro uživatele a manipulaci s nimi. V těchto případech se použije jQuery v seznamech a Codemirror pro textové pole, kdy obě tyto komponenty pomohou dopomoci vizuálnímu vzhledu daných dat. Dále následuje podkapitola, bez které by celá aplikace byla k ničemu, neboli podkapitola o znovupoužitelnosti.

6.3.1.5 Znovupoužitelnost

Do této podkapitoly se též mohou zahrnout všechny předchozí a následující podkapitoly, jelikož toto je teoreticky cílem celého návrhu a výsledkem celé aplikace, aby byla znovupoužitelná. K této části se pojí návrh ovladatelnosti a výsledků, které aplikace vrátí, aby byla data správná a chtěná. K tomuto se pojí poslední kapitola této části, import různých typů dat.

6.3.1.6 Import různých typů dat

V této podkapitole je hlavním požadavkem vybrat typy, které budou zpracovány dále v aplikaci. Dle názoru autora se mezi nejpoužívanější typy v tomto směru řadí xml, csv a json. Značkovací jazyk xml, formáty csv a json tedy budou v aplikaci zahrnuty. Kdy si nejprve uživatel vybere, jaký značkovací jazyk nebo formát bude zadávat, pomocí výběru ze seznamu a poté po zadání uživatelských dat, se tato data v backendové části převedou na typ xml, který bude dále zpracován. Tato možnost je zde z důvodu jednodušší práci dále v aplikaci, aby nemuselo být mnoho dalších podobných metod pouze z důvodu jiného typu, proto je zde vybrán jediný výstup dat, a to ve značkovacím jazyku typu xml. Po tomto návrhu požadavku dále následuje návrh aplikace, do které budou tyto požadavky včetně tohoto zaintegrovány.

6.3.2 Návrh celé aplikace

V této části budou navrhnuty postupy, jak budou zapracovány všechny podmínky a též vzhled a funkčnost celé aplikace jak z hlediska frontendu, tak i z hlediska backendové části.

6.3.2.1 Návrh aplikace

Pro jednodušší ovládání aplikace bude dle návrhu aplikace rozdělena do šesti bodů, kde body jsou výběr typu dat, výběr uživatelských dat, zpracování těchto dat, selekce a přiřazování uživatelských dat k základním datům, transformace dat a výsledek celé aplikace.

Výběr typu dat bude tvořen výběrovým seznamem, kdy si uživatel navolí, který typ dat bude pro svou aplikaci využívat. Po tomto výběru se uživateli zobrazí druhý bod.

Druhý bod bude nejprve tvořen ze dvou radio butonů, které mu dají na výběr, zdali si chce uživatel data zadávat ze souborů, nebo pouze do textového pole. Po výběru z těchto dvou možností se mu daná možnost zobrazí, tedy buď input, ve kterém si vybere soubor, který chce nahrát, nebo textové pole, do jejž zadá hodnoty v textové podobě. Po zadání dat a kliknutí na buton níže pod textovým polem se provede bod třetí.

V bodu třetím na pozadí proběhne výběr elementů z uživatelských dat. Po výběru všech elementů se přesune aplikace k bodu čtvrtému.

Ve čtvrtém bodě se zobrazí dva seznamy, ve kterých budou zadány základní data aplikace v prvním seznamu a v druhém budou elementy, které našla aplikace v uživatelských datech. Z druhého seznamu bude moci uživatel přetahovat a přiřazovat data k prvnímu seznamu, kde poté uvidí tato data přiřazena, pod již vytvořenými daty. Po přiřazení dat a kliknutí na buton, aplikace přejde k bodu pátému.

V pátém bodě se začnou data zpracovávat na pozadí, kde se nejprve v aplikaci vytáhnou přiřazená data k základním datům, poté se k těmto datům přiřadí jejich textové hodnoty a na závěr se vytvoří nová proměnná, ve které bude uložena hodnota zpracovaných uživatelských dat, která byla přiřazena k jejich novým elementům. Tato proměnná bude poslána na frontend a aplikace se přesune k poslednímu bodu.

Šestý bod, kde aplikace vypíše data z pátého bodu v textovém poli.

Po šesti bodech, kterými je tvořena celá aplikace, zazněl ještě pojem o základních datech aplikace. Základní elementy aplikace, ke kterým bude uživatel přiřazovat elementy z uživatelských dat, jsou tato:

Products – Základní element, do kterého se budou umisťovat všechny produkty, bude to jako pole produktů

Product – Element, do kterého se budou zařazovat všechny informace o produktu

ID – Jednoznačná a unikátní definice produktu

Title – Název produktu

EAN – Čárový kód produktu

Code – Kód produktu

Price – Cena produktu

DPH – DPH produktu
Description – Popis produktu
Weight – Váha produktu
Producer - Výrobce
Images - Pole obrázků
Image – Jméno obrázku
URL – Odkaz na obrázek
Parameters Pole parametrů
Parameter – Jméno parametru
Value – Hodnota parametru
Variations – Pole variant
Variation – Základní element varianty
ID – Unikátní a jednoznačné označení varianty
Title – Název varianty
EAN – Čárový kód varianty
Code – Kód varianty
Price – Cena varianty
DPH – DPH varianty
Weight – Váha varianty

Po vytyčení základních elementů, ke kterým bude uživatel přiřazovat vlastní data je zde další podkapitola, ve které je řešeno, jak zapracovat podmínky do návrhu aplikace.

6.3.2.2 Zapracování vytyčených podmínek

V podkapitole výše byl vypsán návrh celé aplikace, který byl rozdělen do šesti bodů, ve kterých bude navrhnuto zapracování podmínek do výše uvedených bodů.

První podmínkou je vybrání dat k importu, kdy je tato podmínka zapracována již v druhém bodu, ve kterém si uživatel vybírá, která data bude chtít importovat, tedy pokud se jedná pouze o počet dat a ne obsah. Pokud se jedná o obsah, kdy chce uživatel vybrat jen část jednoho produktu, který bude chtít ve výsledku aplikace vypsat, tak k tomuto slouží bod čtyři, ve kterém si daná data může přiřadit k příslušným základním elementům a zbytek, který nepřiřadí, tak ve výsledku

nebude zahrnut. Dalším podmínkou, která již trochu souvisí s touto je jednoduchá ovladatelnost.

Jednoduchá ovladatelnost je zapracována ve všech bodech aplikace, kde každý další bod následuje až poté, co se provede předchozí bod a z toho důvodu bude vždy uživatel vědět, v jaké části je. Dále je aplikace ovládaná vždy butony, nebo různými dalšími komponentami, jako jsou například radio butony. Tyto butony nám predikují další podmínku o více způsobech zadávání dat.

V tomto případě je podmínka o možnostech výběru, jak zadávat data zapracována v bodě dvě do radio butonu, kdy si uživatel vybere, který typ nahrání dat chtít použit, zdali pomocí souboru nebo zapsání dat do textového pole. S textovým polem souvisí další podmínka o vizualizaci dat.

Vizualizace dat je zapracována v bodech dva, čtyři a šest. Ve druhém bodě bude použit nástroj Codemirror, aby data, která budou zadávaná uživatel, pokud si uživatel vybere zadání do textového pole, byla naformátovaná a jejich elementy byly viditelně odděleny, jako v různých editorech těchto typů značkovacích jazyků a formátů. To samé jako v bodu dvě bude zahrnuté pro textové pole v bodu šest. Oproti tomu v bodu čtyři bude seznam vytvořen a spravován komponentou jQuery, která umožňuje jednoduchou implementaci a ovladatelnost seznamu. Komponenty jQuery a Codemirror jsou součástí další podmínky o znovupoužitelnosti.

Podmínka o znovupoužitelnosti bude zahrnuta ve všech bodech. Aplikace bude psána tak, aby ji bylo možné rozšířit a dále využívá komponenty, které znovupoužitelnosti aplikace dále pomohou, jako například komponenty jQuery a Codemirror. Tyto komponenty dle autorova názoru jistě též pomohou k znovupoužitelnosti a vracení se uživatelů k této aplikaci. Nejen tyto aplikace, ale i další podmínka o importu různých typů dat jistě aplikaci ve znovupoužitelnosti pomůže.

Poslední podmínkou, která není povinou je import různých typů dat, kdy tato část bude zahrnuta v prvním bodu aplikace, ve které si uživatel vybere, jakým typem značkovacího jazyka nebo formátem budou zadávaná data.

6.3.2.3 Návrh převádění uživatelských dat

Převádění uživatelských dat na výsledná data může být provedeno více způsoby, ale zde jsou vypsány dva způsoby, které jsou dle uživatele nejvíce efektivní.

Prvním způsobem je převádění uživatelských dat přímo v aplikaci a použitím metod, které by byly pro tento účel vytvořeny. Výhodou tohoto řešení je jeho možnost rozšiřitelnosti, kdy nové metody mohou být snadno přidávány a nemusí vůbec zasahovat do chodu aplikace. Oproti tomu hlavní nevýhodou tohoto řešení je optimalizace funkcí, z důvodu úsporného a rychlého chodu aplikace.

Druhým způsobem je využití xsl transformací. Kde by nejdříve musela být vytvořena základní xsl transformace, která by musel být optimalizována na jakýkoliv počet dat. Dále by tato xsl transformace musela být připravena na rozšiřování a další komunikaci s funkcemi. Výhodou xsl transformací je rychlost a optimalizovanost. Oproti tomu nevýhodou je složitější debuggování xsl transformací a dále složitost programování těchto transformací.

Dle autorova názoru jsou obě tyto možnosti dle návrhu dobrým řešením, ale v každé z nich se nacházejí nevýhody, se kterými je nutno počítat a vyřešit je. Podle ostatních částí návrhu bude tedy zvolen způsob první, který data zpracovává pomocí metod, jelikož je jednoduší a lepší k pozdější případné rozšiřitelnosti aplikace.

Po dokončení návrhu aplikace zbývá již jen jedna kapitola před ukázkou výsledné aplikace a tato kapitola se nazývá implementace aplikace.

6.4 Implementace aplikace

V této kapitole bakalářské práce je popsáno vytvoření aplikace, dále nástroje, které byly pro aplikaci použité a po implementaci aplikace budou popsány některé otázky, které při vývoji aplikace byly řešeny. První části implementace je o výběru správných nástrojů pro vývoj aplikace.

6.4.1 Výběr nástrojů

Nástroje, které budou vypsány níže, jsou vybrány podle autorova názoru a též se nejlépe hodí dle stejného názoru pro implementaci aplikace.
Na začátku této podkapitoly se nabízí otázka, jaký programovací jazyk bude využit pro vývoj celé aplikace? Dle autora je tímto programovacím jazykem PHP, jelikož s ním má autor již mnoho zkušeností a jelikož se bude jednat o webovou aplikaci, tak je PHP jasnou volbou. Tedy vystává otázka: „Co je PHP?“

6.4.1.1 PHP

PHP je zkráceninou pro dřívější spojení „Personal Home Page“, ale nyní je PHP spojováno s rekurzivní zkratkou „PHP: Hypertext Preprocessor“, který je přeložen jako „PHP: Hypertextový preprocesor“. PHP je skriptovací programovací jazyk, který je převážně používán jako programovací jazyk pro dynamické internetové stránky nebo pro webové aplikace. Dále může být programovací jazyk PHP použit k vytvoření desktopových a konzolových aplikací. Syntaxe tohoto jazyku je podobná jako programovací jazyky C, Java nebo Pascal. Tento programovací jazyk pracuje na straně serveru.
Mezi výhody programovacího jazyku PHP patří podpora mnoha různých knihoven, například databázových systémů MySQL, Oracle, PostgreSQL a dalších. Další výhodou je specializování programovacího jazyka pro vytváření webových aplikací, rozsáhlí soubor funkcí a jejich dokumentace a multiplatformnost.
Oproti tomu nevýhody jsou nekonzistentní pojmenování funkcí, například strpos() oproti str_replace(). Dále neudržování kontextu aplikace, který poté vytváří znovu, což má za následek oslabení výkonu.
Jednoduchým příkladem pro ukázku programovacího jazyka PHP je vypsaní textu „Hello World!“

```
<?php echo „Hello World!“; ?>
```

Ukázka kódu 11 Jazyk PHP; Zdroj: autor;

Po výběru programovacího jazyka vyvstává další otázka: „Který program se nejlépe hodí pro vývoj celé aplikace?“ Dle autora je nejlepší takovou aplikací PhpStorm, z důvodů jeho jednoduché rozšiřitelnosti. Dalším důvodem je zkušenosti, které autor s tímto programem již má, ale první otázkou, která bude dále zodpovězena je „Co je to PhpStorm?“

6.4.1.2 PhpStorm

PhpStorm je multi-platformní vývojové prostředí, které je zaměřeno na programovací jazyk PHP. Tento program vyvíjí česká společnost JetBrains a je celý naprogramován v programovacím jazyku Java. PhpStorm má funkcionalitu programu WebStorm, ale navíc k WebStormu přidává ještě podporu pro PHP a databázové jazyky, čemuž řadí nejvyšší prioritu, neboli „PhpStorm = WebStorm + PHP + DB/SQL“ (9)

V PhpStromu se dá kromě programovacího jazyku PHP též programovat programovacími jazyky Javascript, CSS, HTML, SQL a další. K všem podporovaným programovacím jazykům nabízí funkcionality typu našeptávání, opravy chyb, automatické formátování a generování kódů.

PhpStorm je uzpůsoben pro podporu dnes nejvíce používaných frameworku, jako jsou Symfony, Laravel, Drupal, WordPress a další. Dále podporuje debuggování, refaktorování a unit testing kódů. Poté též podporuje Live Edit, neboli změna kódu je okamžitě provedena v prohlížeči. V neposlední řadě nabízí v novějších verzích podporu Code With Me, kde umožňuje kódování na stejném projektu ve stejný čas.

Další výhodou PhpStormu je jeho cena, jelikož pro studenty je zdarma. Z tohoto důvodů a důvodů jako je debuggování, refaktorování, podpora dalších programovacích jazyků, jednoduché přidávání dalších frameworků a podobně je dle autora PhpStorm pro tuto aplikaci jasnou volbou.

Tedy již je vybrán nástroj a programovací jazyk, ve kterém bude aplikace naprogramována, ale dále není zřejmé, jaké bude mít aplikace rozložení nebo jak budou data vypsána. Předchozí větu se dá shrnout za pomoci dvou frameworků s názvy Symfony a Docker. Tyto frameworky nám pomohou se základní šablonou

aplikace, s vytvořením serveru, na kterém poběží naše celá aplikace a s komunikací mezí aplikací a webem. Ale stále není jasné, co jsou tyto dva frameworky zač.

6.4.1.3 Symfony

Symfony je webový aplikační framework, který je používaný pro vývoj webových aplikací v programovacím jazyku PHP. Tento framework vychází z návrhového vzoru MVC, neboli architektura, která rozděluje datový model aplikace, uživatelskou část a řídící logiku.

Symfony má výhodu v tom, že dokáže vytvořit základní kostru jakékoliv aplikace. Tato kostra bude využita i v tomto případě pro aplikaci bakalářské práce. Základní kostra vypadá takto:

- App – Konfigurace, šablony a překlady aplikace
  - Cache – Zkompilované šablony
    - Dev – Vývojové prostředí
    - Prod – Produkční prostředí
  - Config – Nastavení aplikace, které je většinou uloženo v souboru typu YAML
  - Logs – Logy aplikace
  - Resources – Zdroje
    - Views – Šablony
- Src – Většina kódu, která podporuje architekturu MVC
  - Bundles – Samostatné části webu, které zajišťují funkci celé aplikace, v jejich podsložkách se mohou objevovat testy, kontroléry a podobně.
- Vendor – Závislosti od dodavatelů třetích stran
- Web – Adresář, který je přístupný z internetu
  - Bundles – Implementované bundles z Src, s již vlastními zdroji CSS, obrázky a další.

K Symfony se pojí ještě další nástroj s názvem Twig, který je v Symfony využíván.

6.4.1.4 Twig

Twig je šablonovací systém pro programovací jazyk PHP. Jeho syntaxe je odvozena od systémů Jinja2 a Django. Twig je open-source systém, který je spojen s frameworkem Symfony a Drupal.

Twig má výhody v rychlosti, neboli Twig optimalizuje kód do prostého PHP kódu, bezpečnosti, kde Twig má režim izolovaného prostoru, kde vyhodnocuje šablony,

zdali nejsou škodlivé. Další výhodou je flexibilita, kdy využívá lexer a analyzátor, které umožňují přidávat vlastní značky a filtry.

```
{% for user in users %}
  * {{ user.name }}
{% else %}
  No users have been found.
{% endfor %}
```

Ukázka kódu 12 Jazyk PHP; Zdroj: (10);

Na ukázce kódu 12 je vidět rozdíl mezi šablonovacím jazykem Twig a jinými, kdy odděluje PHP funkce pomocí složených závorek a procent, poté pomocí dvou spojených složených závorek označuje proměnné k vypsání. Po Twigu, který je součástí frameworku Symfony je zde další framework Docker.

6.4.1.5 Docker

„Docker je Open source projekt pro automatizaci nasazení aplikací jako přenosných a vlastních kontejnerů, které mohou běžet v cloudu nebo místně.“ (11)

Hlavními výhodami aplikace Docker jsou menší velikost, větší flexibilita a menší náročnost na spotřebu výkonu počítače oproti jiným virtuálním nástrojům. Oproti tomu je největší nevýhodou je svázanost s operačním systémem, který je využíván pro běh aplikací v kontejnerech.

Docker je složen ze tří základních součastí: Objektů, démona a registrů.

Docker objekty jsou různé entity, které jsou používány k sestavení aplikace, kde mezi základní entity se řadí kontejner, který je tvořen zapouzdřeným prostředím, jež spouští aplikace a je řízen pomocí Docker API nebo CLI. Další entitou je Image, která je tvořena šablonou pro vytváření kontejnerů a je využívána k odesílání a ukládání aplikací. Poslední entitou je služba, která umožňuje škálování kontejnerů na více démonů, jejíž výsledkem je sada spolupracujících démonů, které komunikují přes rozhraní Docker API, neboli jinak nazýváno jako roj.

Démon docker je jinak nazývaný dockerd, který je perzistentní neboli stálí proces, který spravuje a zpracovává objekty do kontejnerů. Démon čeká na požadavky, které jsou odeslány pomocí rozhraní Docker Engine API.

Docker registr je uložiště pro všechny image Docker, které poté mohou stahovat klienti, kteří se k nim připojí pomoci pull požadavky nebo pro nahrávání image Docker, který pomocí push požadavku nahrají. Tyto registry mohou být veřejné nebo soukromé. Mezi největší registry se řadí Docker Hub, který je považován za výchozí registr a Docker Cloud, který hledá image.

Docker nám v tomto případě zastane roli serveru, jelikož dle autorova názoru je Docker v tomto případě nejlepším řešením pro tuto roly a oproti tomu Symfony je nejlepší řešení v rámci struktury a funkčnosti aplikace.

Dále je již vybrán nástroj a programovací jazyk, ve kterém bude aplikace naprogramována a frameworky, které pomohou s jeho řízením a provozem. Ale ještě je zde problém, kde budou data zálohována, pokud by se cokoliv v rámci aplikace pokazilo? Odpovědí je nástroj GitHub.

6.4.1.6 GitHub

GitHub je webová služba, která podporuje vývoj softwarů za pomocí verzovacího nástroje git. GitHub umožňuje ukládání a zálohování open-source projektů, které mohou být buď veřejné nebo soukromé. Byl v roce 2018 odkoupen společností Microsoft a nyní je vyvíjen touto společností.

GitHub má mnoho vlastností, od komentářů k pull-requestům až po vlastní wiki. Mezi nejvíce používané vlastnosti GitHubu patří systém pro sledování problémů, wiki, historie verzování, hledání rozdílů, dokumentace včetně automaticky poskytovaných šablon a další.

Po vybrání všech těchto nástrojů jsou zde ještě dva nástroje, které již zazněly v kapitole o návrhu aplikace. Jde o nástroje CodeMirror a jQuery. Prvním z těchto nástrojů je CodeMirror.

6.4.1.7 CodeMirror

CodeMirror je JavaScriptová knihovna využívaná k zobrazení a zvýraznění kódu v obsahu webu. Má velmi bohaté API rozhraní a zaměřuje se na rozšiřitelnost.

Mezi výhody patří barevné zobrazení textu, kontrola syntaxe vybraného programovacího nebo značkovacího jazyka nebo formátu, automatické doplňování a vyhledávání v textu.

```
<!-- Create a simple CodeMirror instance -->
<link rel="stylesheet" href="lib/codemirror.css">
<script src="lib/codemirror.js"></script>
<script>
 var editor = CodeMirror.fromTextArea(myTextarea, {
  lineNumbers: true
 });
</script>
```

Ukázka kódu 13 Jazyk JavaScript, HTML; Zdroj: (12);

Ukázka kódu 13 ukazuje inicializaci CodeMirror knihovny a poté přídání funkcí CodeMirror elementu „textarea“ s atributem id „myTextarea“.

Po nástroji CodeMirror následuje jQuery, neboli jQuery UI a jeho další části Sortable a Nested Sortable.

6.4.1.8 jQuery UI

Sortable a Nested Sortable patří pod javascriptový framework jQuery UI, který je zaměřen na uživatelské rozhraní, jež má za cíl ulehčit vývojářům implementaci pokročilých efektů a vylepšit funkcionality HTML prvků.

jQuery UI je složeno z jádra (Core), poté z interakcí (Interactions), widgetů (Widgets), efektů (Effects) a nástroje (Utilities).

Jádro knihovny jQuery UI obsahuje moduly pro samotné jádro (Core), poté widgety (Widget), kde umožňuje sestavení widgetu s běžným API, dále modul myši (Mouse), jež obsahuje interakce s myší a jako poslední modul pozice (position), který je požit pro pozicování prvků relativně vůči sobě.

V interakcích jsou obsaženy metody pro pokročilou implementaci interakcí mezi uživatelem a rozhraním webové aplikace. Obsahuje metody Draggable, pro přesouvání určitého prvku, Droppable, pro implementaci chování drag and drop. Dále Resizable pro změnu velikost, Selectable pro vybrání více prvků najednou a Sortable pro třídění a zaření elementů pomocí myši.

Widgety obsahují pokročilejší UI elementy a metody. Obsahuje Accordion, neboli snadno rozklikávací vkládatelné položky, dále Autocomlepte, pro automatické

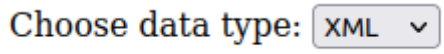
doplňování, Buton neboli tlačítko, které má různé styly a funkcionalitu. Dalším elementem je Tooltip neboli nápověda a další.
Efekty obsahuje grafické efekty. Obsahuje metody Show a Hide pro zobrazení a skrytí prvků. Dále Effect neboli efekty, kde je možnost vybrání různých grafických efektů.
Nástroje obsahují pomůcky pro usnadnění práce s jQuery UI. Mezi metody patří Position, který slouží k pozicování widgetů a Widget Factory, který umožňuje tvorbu vlastních widgetů.

6.4.2 Vývoj aplikace

Aplikace byla vytvářena dle bodů, které byly popsány v návrhové části. Po instalaci a přípravě všech nástrojů byla vytvořena základní struktura dle popisu v Symfony.

6.4.2.1 Výběr typů dat

V první části vývoje aplikace byla vytvořena základní twig šablona, která obsahuje v části hlavičky odkazy na css soubory, poté v části skriptové odkazy na nástroje CodeMirror a jQuery a též obsahuje bloky, které jsou v částech těla šablony a skriptu šablony.
Po vytvoření základní šablony byla vytvořena druhá twig šablona, která rozšiřuje základní twig šablonu. V této nové twig šabloně je vytvořen seznam všech podporovaných značkovacích jazyků a formátů, ze kterých si může uživatel vybrat ten, který chce dále zpracovat. Hodnota tohoto seznamu byla pomocí jQuery uložena do proměnné pro budoucí použití.
Vytvořením tohoto seznamu se dále přesouváme do další části, kde si uživatel vybírá jak a která data bude dále nahrávat do aplikace.

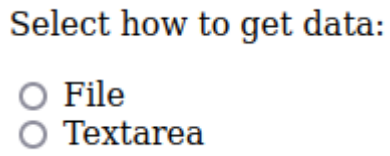


Obr. 1; Ukázka výběru typu dat; Zdroj: autor;

6.4.2.2 Výběr uživatelských dat

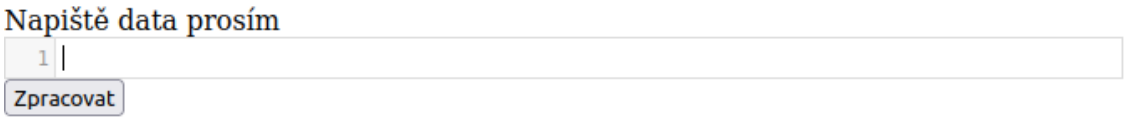
V této části byla rozšířena druhá twig šablona. Do této šablony je přidán seznam z radio butonů, který obsahuje možnosti typu: „Nahrání dat ze souboru“ nebo „Zadání dat do textového pole“. Tomuto seznamu byla přidána ve skriptové části

funkce, která pomocí jQuery sleduje, kdy je seznam vybrán a dle vybrané položky zobrazí buď textové pole nebo možnost nahrání souboru. Textovému poli byly dále přidána funkcionality z nástroje CodeMirror, který zvýrazňuje syntaxi nastaveného programovacího, značkovacího jazyka, nebo formátu který je nastavován pomocí proměnné, která byla vytvořena v přechozí části. Dále byl přidán buton, který nese jméno „Zpracovat“ a přesouvá vývoj aplikace k dalšímu bodu, kde se data dále zpracovávají.



Obr. 2; Ukázka výběru nahraní dat; Zdroj: autor

Obr. 3; Ukázka zadání dat pomocí souboru; Zdroj: autor



Obr. 4; Ukázka zadání dat pomocí textového pole; Zdroj: autor

Obr. 5; Ukázka zadání dat do textového pole a zvýraznění dat; Zdroj: autor;

6.4.2.3 Zpracování dat

Po vytvoření butonu s názvem „Zpracovat“ v předchozí části byla tomuto butonu přiřazena v jQuery funkce, která po kliknutí na tlačítko začne uživatelská data zpracovávat. V první řade byla vytvořena proměnná, do které se nahrává typ značkovacího jazyka, nebo formátu dat, poté do další proměnné se nahrávají data buď z textového pole, nebo data ze souboru. Pokud jde o data ze souboru, je zde přidána další funkce, která nahrává data ze souboru a po nahrání všech dat to proměnné volá též funkci, jako textové pole. V této funkci jsou nahraná data a typ značkovacího jazyka nebo formátu dat poslán pomocí metody post do backendové části.

V backendové části je vytvořena třída, která všechny tyto požadavky přijímá. V této třídě byla vytvořena funkce, která příjme požadavek ze zadané cesty a poté poslaná data načte do lokálních proměnných. Tyto proměnné poté dále podle typu buď hnedka zpracovává, pokud jde o typ xml, nebo je posílá dále do třídy, která data transformuje na značkovací jazyk xml. Třídy pro zpracování a transformování značkovacích jazyků a formátů dat jsou třídy pomocné.

Transformování dat na značkovací jazyk xml probíhá v pomocné třídě, která obsahuje metody pro zpracování formátů json a csv. Tyto metody poté vrací textové hodnoty v značkovacím jazyce xml. Poté dále probíhá zpracování značkovacího jazyka xml v další pomocné třídě. Kde po zpracování tohoto značkovacího jazyka jsou z něj pomocí metod této třídy vytaženy elementy, které se vracejí zpět na frontend.

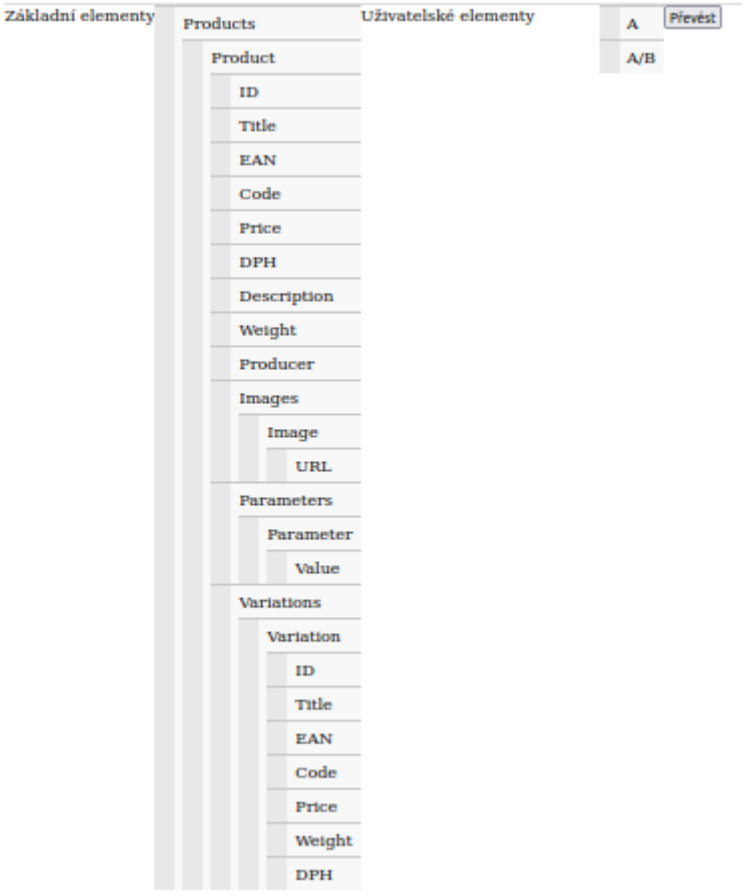
Po vrácení zpracovaných uživatelských dat na frontend proběhne v jQuery funkce, která tyto elementy vypíše do jednoho seznamu a základní elementy aplikace do druhého seznamu. Poté přichází další část implementace, kde uživatel selektuje a přiřazuje uživatelská dat k základním datům.

6.4.2.4 Selekce a přiřazování dat k základním datům

V přechozí části byly vytvořeny dva seznamy, které obsahují základní a uživatelská data. Oběma těmto seznamům je v jQuery přiřazena funkčnost z rozšíření nestedSortable. Díky nestedSortable je jednoduší implementace pohybu ze seznamu uživatelských dat do druhého seznamu. Seznam se základními daty má nastavený

zákaz pohybu jeho elementů, obsažených v něm. Seznam, který obsahuje uživatelská data má povolený pohyb elementů a vždy, po přetažení elementů do druhého seznamu je tento element duplikován a duplikovaný element se vytvoří v základním seznamu na místě, kam byl přesunut daný element. Tento element má pro lepší vizuální vlastnost na konci jména přidán utf-8 charakter „_“, pro lepší definici nových elementů.

Dále byl přidán buton s názvem „Převést“, který po přiřazení všech dat, jenž chce uživatel transformovat, posílá data přiřazenou jQuery funkcí metodou post do backendové části, kde dále probíhá transformace dat.



Obr. 6; Ukázka selekce a přiřazování dat; Zdroj: autor;

6.4.2.5 Transformace dat

Po přijmutí požadavku o zpracování dat, metoda, která tento požadavek zpracovává vytváří proměnou, do které ukládá data, která přišla z frontendu. Data dále posílá do pomocné třídy, kde v rámci metod proběhne selekce dat, která mají být zahrnuta ve výsledku aplikace, dále nové elementy, ke kterým uživatel přiřadil své elementy

a poté přiřazení těchto uživatelských dat k novým elementům. Poté transformace těchto nových uživatelských dat do xml proměnné, která je poté posílána zpět na frontend.

6.4.2.6 Výsledná data z aplikace

Výsledná data, která přišla jako odpověď z backendové funkce jsou dále vypisovány do vytvořeného textového pole, kterému je přiřazena ve skriptové části funkcionalita nástroje CodeMirror na zvýrazňování syntaxe xml dat.
Při implementaci vyvstala řada otázek, které budou řešeny níže v další podkapitole.

Převedená data

```
1 <?xml version="1.0"?>
2 <Products>
3   <Product>
4     <ID>Zkoužka</ID>
5   </Product>
6 </Products>
7
```

Obr. 7; Ukázka výsledných dat aplikace; Zdroj: autor;

6.4.3 Řešené otázky v implementaci aplikace

V této části bakalářské práce bude řešeno pár otázek, které byly řešeny v rámci implementace aplikace. Otázky jsou rozděleny do podkapitol, kde vždy otázka je názvem dané podkapitoly.

6.4.3.1 Jak specifikovat konec produktu v produktech?

Daná otázka vznikla v části, ve které aplikace zpracovává uživatelská data a z těchto dat poté vytahuje jejich elementy. Jelikož když jsou poté elementy přiřazeny, tak je složité najít základní element, který rozděloval tyto produkty na jednotkový produkt. V tomto směru autor zastává názor, že řešením je přiřazení těchto dvou uživatelských elementů k základním elementům s názvy products a product. Toto řešení je nejednoduší možností. Tato problematika je spojena i s další otázkou.

6.4.3.2 Jak vyřešit uživatelské elementy se stejnými jmény v jiných částech?

Příkladem této otázky je ID produktu a ID varianty, ve kterých uživatel po zpracování těchto elementů neví, které ID patří kam. V rámci aplikace byl tento

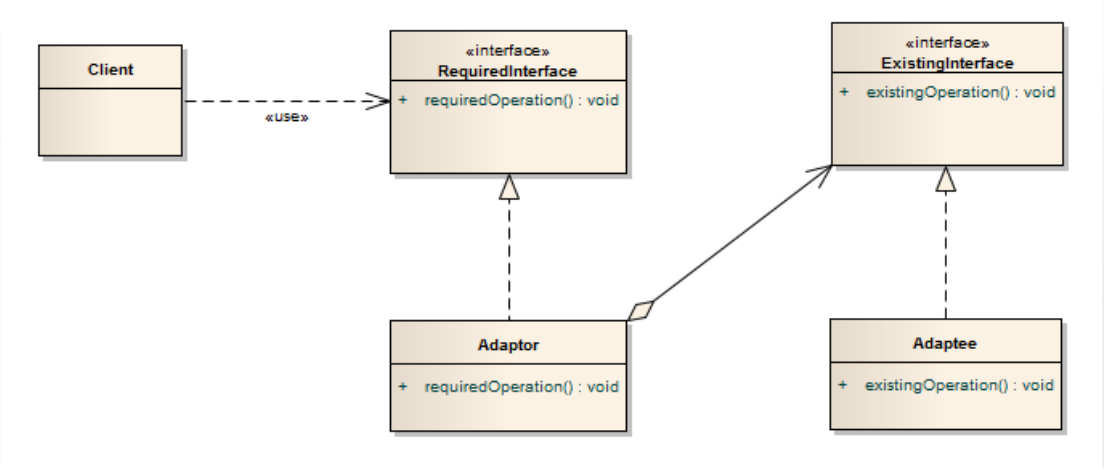
problém vyřešen vypisováním místo názvu elementu jeho cestu, aby byla menší pravděpodobnost záměny dat.

6.4.3.3 Jak si zaměstnanec internetového obchodu upraví data, která chce zpracovávat?

Tato otázka je již zapracována do návrhu, ale dle autora se hodí i do této části z důvodu poukázání na závislosti uživateli. Jelikož pokud uživatel nevyplní některá data, nebo nepřiřadí žádný element k produktu a k poli všech produktů, tak aplikace nemůže fungovat. Z tohoto důvodu je zde velká závislost na uživateli dané aplikace. Dále následuje podkapitola, která pojednává o návrhovém vzoru, který byl zahrnut do implementace, aby byly splněny všechny podmínky kladené na aplikaci.

6.4.4 Návrhový vzor adaptér

V této podkapitole bude pospán návrhový vzor adaptér, který byl zahrnut do vývoje aplikace. Definicí adaptéru je: „Návrhový vzor adapter (Wrapper) slouží pro definici jiného (stabilního) rozhraní, než daná třída poskytuje.“ (13) Nebo následující definice: „přizpůsobení rozhraní třídy na rozhraní jiné třídy, spolupráce tříd s různým rozhraním.“ (14) A do třetice tato definice: „Upraví rozhraní na rozhraní očekávané klientem.“ (15)



Obr. 8; Návrhový vzor adaptér; Zdroj: (13)

Na obrázku vidíme použití návrhového vzoru adaptér. Adaptér je třída, která pomáhá rozšířit existující rozhraní pomocí dalších funkcí k tomu, aby dokázalo využívat další nově potřebné funkce, ale aby zároveň nebylo do základního rozhraní

zasahováno. V tomto případě tedy adaptér upraví nové operace do podoby, ve které je dokáže staré rozhraní zpracovat.
Příklad, který může charakterizovat tento návrhový vzor je takový, že pokud máme dva různé lidi, kteří mluví každý cizím jazykem, tak překladatel je v tomto směru pro ně adaptérem, jelikož překládá slova jednoho člověka do jazyka druhého. Podobným návrhovým vzorem, jako je adaptér, je návrhový vzor bridge.

6.4.4.1 Rozdíl návrhových vzorů adaptér a bridge

V první řade je určení, co vlastně bridge je, definicí bridge je: „Bridge je návrhový vzor pro strukturu objektů. Používá se, když chceme oddělit abstrakci od její implementace tak, aby se obě mohly měnit nezávisle. Klient posléze využije některou z implementací nepřímo prostřednictvím abstrakce.“ (16) Již v této definici je zřejmé rozdíl mezi těmito dvěma návrhovými vzory. „Návrhový vzor bridge je využíván při implementaci, oproti tomu, návrhový vzor adaptér je využíván až po implementaci. Z toho vyplývá, že návrhový vzor bridge je oproti návrhovému vzoru adaptér více náchylný k rozšířením.“ (14) Po vymezení těchto rozdílu se nabízí otázka, jak následující návrhová vzor je implementován v rámci aplikace?

6.4.4.2 Implementace návrhového vzoru adaptér

V aplikaci je tento vzor zahrnut v rámci možnosti zpracování nejen značkovacího jazyka xml, ale též i dalších značkovacích jazyků a formátů dat. Vzor byl aplikován na třídu, která transformuje další značkovací jazyky na značkovací jazyk xml. Tato třída obsahuje funkce, které jsou využity k přeformátování formátů json a csv do značkovacího jazyku xml. Výsledná data se vracejí zpět do základní třídy, kde jsou dále zpracovány jako značkovací jazyk xml.
V tomto případě byl tedy vzor aplikován v úpravách uživatelských dat, aby rozhraní nebylo tolik složité a zároveň se již nemuselo více rozšiřovat. Tímto vzorem též byla splněna podmínka, kterou si kladl vývojář aplikace, aby bylo možné zpracovávat více značkovacích jazyků a formátů než jen xml. Dále ale již následuje kapitola, ve které autor ukáže používání celé aplikace.

6.5 Ukázka aplikace

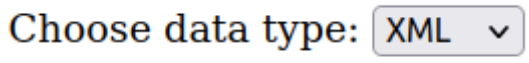
V této kapitole autor ukáže na přiložených obrázcích a poté popíše základní funkcionalitu a používání aplikace.

Nejdříve autor musel vytvořit ukázková data, která budou v rámci aplikace zpracována. Ukázková data jsou tato:

```
<Products>
  <Product>
    <Name>Kolo</Name>
    <ID>123</ID>
    <Cena>1000</Cena>
  </Product>
  <Product>
    <Name>Auto</Name>
    <ID>155</ID>
    <Cena>50000</Cena>
  </Product>
  <Product>
    <Name>Autobus</Name>
    <ID>235</ID>
    <Cena>200000</Cena>
    <Popis>Krásný hnědý autobus</Popis>
  </Product>
</Products>
```

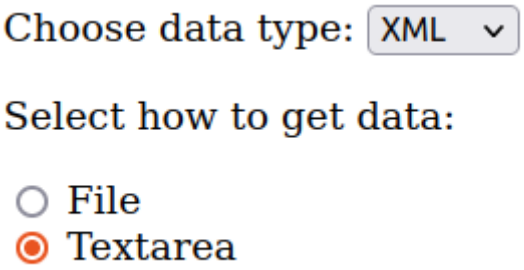
Ukázka kódu 14 Jazyk XML; Zdroj: autor;

Na začátku aplikace si uživatel musí zvolit, jaký typ značkovacího jazyka nebo formátu bude uživatel do aplikace posílat. V tomto případě byl vybrán značkovací jazyk xml.



Obr. 9; Výběr značkovacího jazyka XML; Zdroj: autor

Po vybrání značkovacího jazyka nebo formátu dat byla zobrazena další část, ve které si uživatel musí zvolit, jak bude data do aplikace posílat, jestli v rámci nahrání souboru nebo zadáním textových hodnot. V případě našeho příkladu byl vybrán způsob zadáním textových hodnot.



Obr. 10; Výběr textového pole XML; Zdroj: autor

Po vybrání možnosti zadání se dále zobrazilo textové pole, do kterého jsou zadávány v tomto případě textová data. Data byla zadána dle příkladu, který autor vytvořil výše. Na obrázku, který je umístěn pod tímto textem je též zobrazena syntaxe značkovacího jazyka xml díky nástroji CodeMirror. Po zadání dat a stisknutí tlačítka Zpracovat byla zobrazena další část aplikace.

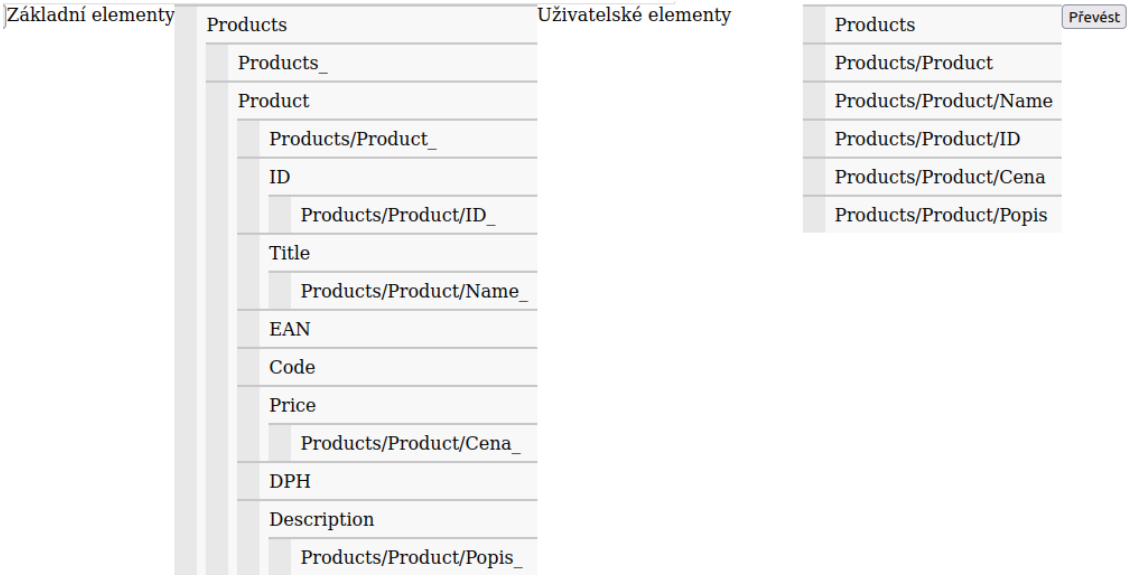
Obr. 11; Zadání hodnot do textového pole; Zdroj: autor

V této části aplikace jsou zobrazeny základní elementy a poté uživatelské elementy, které byly vybrány z textového pole. Uživatelské elementy jsou popsány cestou, jak

již byl psáno výše při implementaci. Přetahováním autor přiřadil vlastní elementy k základním elementům dle tabulky:

Základní elementy	Autorovi elementy
Products	Products
Product	Products/Product
ID	Products/Product/ID
Title	Products/Product/Name
Price	Products/Product/Cena
Description	Products/Product/Popis

Tabulka 2; Ukázka dat Zdroj: autor;



Obr. 12; Přiřazení uživatelských elementů k základním elementům; Zdroj: autor

Po přiřazení všech uživatelských elementů k základním elementům aplikace a kliknutí na tlačítko převést byl vypsán výsledek. Výsledek dle autorových dat vypadá následovně ve výsledku aplikace.

Data, která jsou zobrazena na posledním obrázku vypadají v textové podobě takto:

```
<?xml version="1.0"?>
<Products>
  <Product>
    <Title>Kolo</Title>
    <ID>123</ID>
    <Price>1000</Price>
  </Product>
  <Product>
    <Title>Auto</Title>
    <ID>155</ID>
    <Price>50000</Price>
  </Product>
  <Product>
    <Title>Autobus</Title>
    <ID>235</ID>
    <Price>200000</Price>
    <Description>Krásný hnědý autobus</Description>
  </Product>
</Products>
```

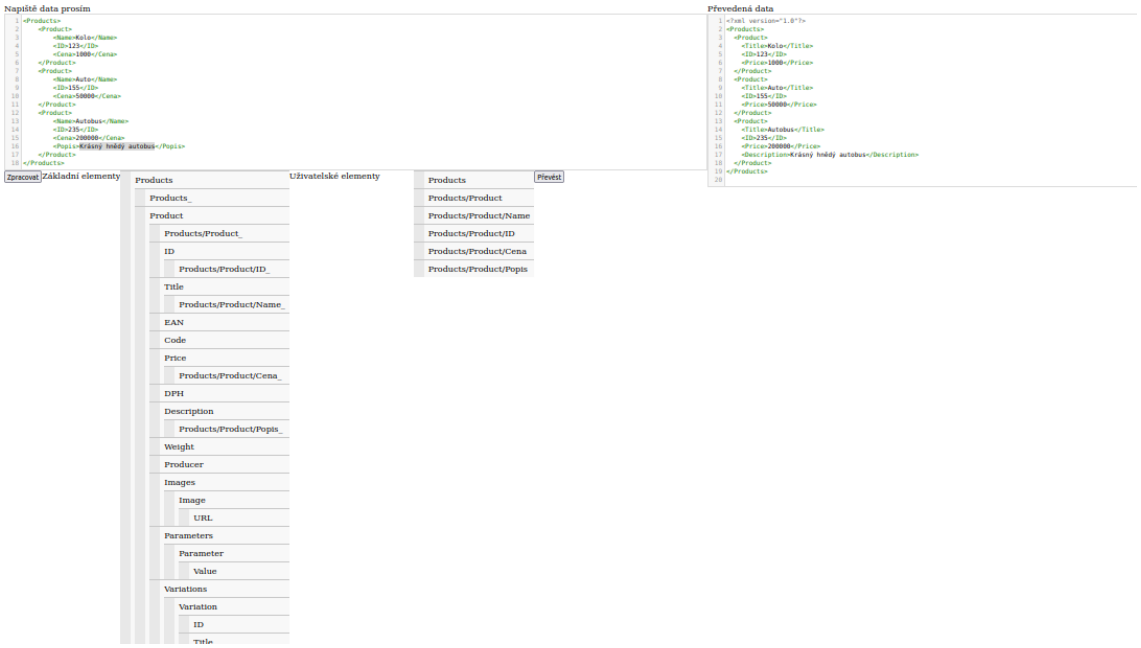
Ukázka kódu 15 Jazyk XSL; Zdroj: autor;

Převedená data

```
1  <?xml version="1.0"?>
2  <Products>
3    <Product>
4      <Title>Kolo</Title>
5      <ID>123</ID>
6      <Price>1000</Price>
7    </Product>
8    <Product>
9      <Title>Auto</Title>
10     <ID>155</ID>
11     <Price>50000</Price>
12   </Product>
13   <Product>
14     <Title>Autobus</Title>
15     <ID>235</ID>
16     <Price>200000</Price>
17     <Description>Krásný hnědý autobus</Description>
18   </Product>
19 </Products>
20
```

Obr. 13; Výsledek zpracovaných dat z aplikace; Zdroj: autor

Jak již bylo napsáno v návrhové části, postup aplikací je provázen vždy zobrazováním dalších části po dokončení předchozích částí, ale zároveň se staré části neskrývají z důvodu možnosti úpravy dat. Následující obrázek ukazuje celý průchod aplikací s daty, které byly vypsány výše. Po tomto obrázku dále následuje kapitola, která je shrnutím celé bakalářské práce.



Obr. 14; Vizuální obrázek celé aplikace; Zdroj: autor

7 Shrnutí práce

Bakalářská práce se věnuje problematice internetových obchodů v oblastech konverze a importu dat do sytému těchto internetových obchodů. Bakalářská práce obsahuje popis tohoto problému a dále možnosti, kterými by se dal problém vyřešit. Teoretická část obsahuje popis problému konverze dat v rámci internetových obchodů, dále obsahuje podobná řešení, ze kterých autor získával částečnou inspiraci. V podobných řešeních je vypsána webová aplikace napojse, dále xemel a poté webové konvertory, které dokážou konvertovat jakýkoliv značkovací jazyk nebo formát dat. Poté se dále teoretická část věnuje programovacím jazykům, kterými jsou psány převážně webové aplikace a v poslední kapitole teoretické části jsou popsány základní značkovací jazyk xml a formáty json a csv.

Po teoretické části následuje praktická část, ve které se autor snaží problém vyřešit pomocí aplikace, která převádí data do ustálené formy ve značkovacím jazyce xml. Dále aplikace plní podmínky, které byly určeny v požadavcích na aplikaci. Tyto podmínky jsou určeny zaměstnanci internetových obchodů a poté vývojářem aplikace. Mezi podmínky patří výběr dat k importování, jednoduchá ovladatelnost, znovupoužitelnost, možnost zadávání dat více způsoby, vizualizace dat a poslední nepovinou podmínkou je možnost převedení různých typů značkovacích jazyků a formátů dat. Po vytyčení podmínek následuje analýza daných podmínek a poté návrh celé aplikace. Po návrhu následuje část o vývoji aplikace.

Aplikace je vyvíjena ve vývojovém prostředí programu PhpStorm, pro vývoj aplikace je hlavně využit programovací jazyk php a dále z velké části programovací jazyky html a javascript s jeho rozšířením s názvem jQuery. Pro chod aplikace jsou dále využity frameworky Symfony a Docker.

Bakalářská práce obsahuje celý popis vývoje této aplikace v rámci praktické části a na konci praktické části obsahuje ukázky z již vytvořené aplikace. Základní charakteristikou této aplikace je možnost převodu značkovacího jazyku xml a dále návrh převodu formátů json a csv do základní určené šablony značkovacího typu xml. Dále následuje kapitola, ve které jsou vyvozeny závěry a další doporučení k rozšíření této aplikace.

8 Závěry a doporučení

Konverze dat v internetových obchodech je velmi široký a obecný pojem, proto existuje již mnoho řešení, jak je uvedeno v teoretické části o podobných řešeních. Aplikace bakalářské práce obsahuje jen řešení pro drobnou část celé této rozsáhle problematiky. Autor se v této aplikaci snaží zpracovávat základní značkovací jazyky a formáty dat, ale dle skutečných potřeb internetových obchodů jsou tyto značkovací jazyky a formáty dat (xml, csv a json) pouze částí všech daných značkovacích jazyků a formátů dat.

Aplikace by se jistě dala rozšířit i o typ excelovských dokumentů, ale implementace tohoto typu je už složitější a dle autorova názoru by se musela celá aplikace upravit. Pokud by programátor chtěl tedy tento typ zahrnout, byla by jednoduší možnost poté dělat převody dat přes xsl transformace, které se k tomuto typu dat více hodí.

Autor si též zvolil návrhový vzor adaptér, ale po pozdějším uvážení by bylo lepší zvolit návrhový vzor bridge, který umožňuje jednoduší rozšíření o další komponenty. Dále by se jistě dala aplikace rozšířit o databázovou stránku, kdy by se poté data ukládala do serverové databáze, což též podporuje framework Docker.

Autor by rád doporučil všem, kdo budou vytvářet podobné aplikace, aby bylo hlavním důvodem vývoje této aplikace nejen zpeněžení, ale též vývoj celé aplikace k usnadnění a šetření času zaměstnancům internetových obchodů.

Dle zkušeností ale autor není jediný, kdo tuto problematiku řeší a bude řešit, proto jistě tato bakalářská práce přinese jistou inspiraci budoucím programátorům, kteří se snaží problém s konverzí dat vyřešit.

9 Seznam použité literatury

[1] Elektronický obchod. *BusinessInfo.cz - Oficiální portál pro podnikání a export.* [Online] 1997. [Citace: 15. 08 2021.] https://www.businessinfo.cz/navody/elektronicky-obchod-ppbi/.

[2] Štráfelda, Jan. Co je e-shop. *Jan Štráfelda: průvodce online projektem.* [Online] [Citace: 15. 08 2021.] https://www.strafelda.cz/e-shop.

[3] Co to umí. *Napojse.cz. Napojení na dodavatele.* [Online] 2021. [Citace: 15. 08 2021.] https://www.napojse.cz/stranka/co-to-umi .

[4] Jednoduchý a rychlý editor XML feedů. *Xemel. Jednoduchý a rychlý editor XML feedů.* [Online] Xemel, 2017. [Citace: 15. 08 2021.] https://www.xemel.cz/.

[5] CSV formát. *Shoptet.* [Online] Shoptet, a.s. [Citace: 15. 08 2021.] https://www.shoptet.cz/slovnik-pojmu/csv/.

[6] Značkovací jazyk HTML. *Tvorba webu – HTML, CSS, JS.* [Online] Web. [Citace: 15. 08 2021.] https://web.vavyskov.cz/znackovaci-jazyk.html.

[7] XSL -- stylový jazyk pro náročné. *Domovská stránka Jirky Koska -- "VŠE O WWW".* [Online] 1999. [Citace: 15. 08 2021.] https://www.kosek.cz/clanky/xml/xml-xsl.html.

[8] XSLT Basic Example - Web APIs. *Developer mozilla.* [Online] MDN, 2005. [Citace: 15. 08 2021.] https://developer.mozilla.org/en-US/docs/Web/API/XSLTProcessor/Basic_Example.

[9] PhpStorm: The Lightning-Smart IDE for PHP Programming by JetBrains. . *etBrains: Essential tools for software developers and teams .* [Online] [Citace: 15. 08 2021.] https://www.jetbrains.com/phpstorm/ .

[10] Home - Twig - The flexible, fast, and secure PHP template engine. *Home - Twig - The flexible, fast, and secure PHP template engine.* [Online] 2010. [Citace: 15. 08 2021.]

[11] Co je Docker? . *Microsoft Docs.* [Online] Microsoft , 2021. [Citace: 15. 08 2021.] https://docs.microsoft.com/cs-cz/dotnet/architecture/containerized-lifecycle/what-is-docker.

[12] CodeMirror. *CodeMirror .* [Online] [Citace: 15. 08 2021.] https://codemirror.net/#.

[13] Adapter. *Algoritmus.* [Online] 2015. [Citace: 15. 08 2021.] https://www.algoritmy.net/article/1635/Adapter.

[14] 0-designpatterns.pdf. *Dokuwiki .* [Online] [Citace: 15. 08 2021.] https://cw.fel.cvut.cz/old/_media/courses/a7b36ass/0-designpatterns.pdf.

[15] BÖHMER, Marian. *Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj].* Brno : Brno: Computer Press, 2012. 978-80-251-3338-5.

[16] Vojtěch, Hordějčuk. GoF: Bridge (most) - Vojtěch Hordějčuk. *Vojta Hordějčuk aka voho - Software Engineer and Bedroom Music Producer.* [Online] 2008. [Citace: 15. 08 2021.] http://voho.eu/wiki/bridge/.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Jakub Urbanec**
Studium:	I1700154
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Grafický konfigurátor importu dat**
Název bakalářské práce AJ:	Graphical data import configurator

Cíl, metody, literatura, předpoklady:

Cíl práce: vytvořit grafický konfigurátor importu strukturovaných dat pro běžného uživatele

Osnova:

1. Úvod\par
2. Analýza typů dokumentů k importu\par
3. Podobné programy pro transformaci dat\par
4. Návrh uživatelského rozhraní z hlediska uživatelské přívětivosti\par
5. Popis implementace\par
6. Shrnutí výsledků \par
7. Závěry a doporučení \par
8. Seznam použité literatury\par
9. Zadání práce\par

SIKOS, Leslie. *Mastering Structured Data on the Semantic Web: From HTML5 Microdata to Linked Open Data*. New York: Apress, 2015. ISBN 978-1484210505.

DOMES, Martin. *SEO: jednoduše*. Brno: Computer Press, 2011. Naučte se za víkend (Computer Press). ISBN 978-80-251-3456-6.

E-trend: spolehlivé tipy pro váš e-shop. Modletice: Direct Parcel Distribution CZ, [2013?]-2017.

Garantující pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Mgr. Daniela Ponce, Ph.D.
Oponent:	Ing. Martina Husáková, Ph.D.
Datum zadání závěrečné práce:	21.10.2014