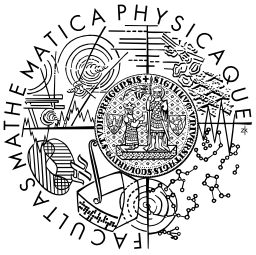


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

**BAKALÁŘSKÁ PRÁCE**

Michal Kolcun

# Validátor CSV súborov podľa odporúčaní CSV on the Web

Katedra softwarového inžinierstva

Vedoucí bakalářské práce: RNDr. Jakub Klímek, Ph.D.
Studijní program: Informatika
Studijní obor: Programovanie a vývoj software

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného 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 Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....  .....

Podpis autora

Chcel by som sa poďakovať vedúcemu bakalárskej práce RNDr. Jakubovi Klímkovi Ph.D. za odborné vedenie, cenné rady a množstvo investovaného času do vedenia bakalárskej práce. Vďaka patrí taktiež rodine, priateľke a priateľom, ktorí ma podporovali v mojom štúdiu.

Název práce: Validátor CSV súborov podľa odporúčaní CSV on the Web

Autor: Michal Kolcun

Katedra: Katedra softwarového inžinierstva

Vedoucí bakalářské práce: RNDr. Jakub Klímek, Ph.D., Katedra softwarového inžinierstva

Abstrakt: Cieľom práce bolo vytvorenie validátoru CSV súborov podľa odporúčaní CSV on the Web, ktoré bližšie špecifikujú dátový formát CSV. Takáto validácia je potrebná pre zvýšenie kvality dát na webe. Validátor poskytuje používateľom viacero prívetivých používateľských rozhraní v podobe CLI aplikácie, webovej služby a webovej aplikácie. Validačná funkcionalita je implementovaná v podobe knižnice v populárnom objektovo-orientovanom jazyku C#. V texte práce priblížime vývoj validátora a všetkých jeho používateľských rozhraní od samotných doporučení CSV on the Web, analýzy existujúcich riešení, návrhu architektúry, až po samotnú implementáciu, testovanie a evaluáciu výsledného softvérového diela.

Klíčová slova: CSV CSV on the web RDF JSON;JSON-LD Validácia dát Relačné dáta;Tabuľkové data Dátové modely Dátové schémy Dátové formáty Metadáta Slovník metadát C# Webová aplikácia Webová služba Command-line aplikácia Knižnica

Title: Validator of CSV files according to the CSV on the Web Working Group recommendations

Author: Michal Kolcun

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Klímek, Ph.D., Department of Software Engineering

Abstract: Abstract: The goal of this thesis was to create a validator of CSV files according to the CSV on the Web recommendations, which enhance CSV data format. This validation is needed to increase the quality of data on the web. Validator offers multiple user-friendly interfaces in the form of CLI app, web service and web app. The validator itself is a library implemented in the popular object-oriented programming language C#. In this thesis we will describe the development of the validator, starting with the recommendations and analysis of existing solutions, through the design of architecture and user interfaces, and finish with the description of implementation, testing, and evaluation of software solution.

Keywords: CSV CSV on the web RDF JSON JSON-LD Data validation Relational data Tabular data Data models;Data schemas Data formats Metadata Metadata vocabulary C# Web application Web service Command-line application Library

# Obsah

# Úvod

S tabuľkovými dátami sa stretneme v praxi mnohokrát, nakoľko ľuďom príde prirodzené štruktúrovať svoje myšlienky do tabuľkovej podoby. S príchodom tabuľkových sa dostavila taktiež potreba takéto dáta zapisovať. Z tohto dôvodu začalo vznikať množstvo dátových formátov, ktoré umožňujú zapisovať tabuľkové dáta do strojovo čitateľnej podoby.

Častým problémom takýchto vznikajúcich formátov však bola chýbajúca dôkladná a formálna špecifikácia pravidiel, ako vytvárať takéto súbory. S príchodom internetu sa tieto problémy stupňovali, nakoľko autori si konkrétne pravidlá zvykli domýšľať, a preto vznikalo množstvo podobných, no ne-validných dokumentov v rozličných formátoch, ktoré sa bežne šírili po internete.

CSV formát sa stal najpopulárnejším dátovým formátom pre tabuľkové dáta. Pre tento formát bol dokonca dvakrát vydaný informačný dokument v podobe RFC*(Request For Comments)*, konkrétne RFC4180[1] a RFC7111[2].

Tieto štandardizačné dokumenty nám definujú všetky formálne aspekty validného CSV dokumentu, ktoré by mali producenti dodržať. Problém však je, že takýto formát nám neumožní bližšie špecifikovať podrobnosti o tabuľke, akými sú napríklad: koľko má tabuľka stĺpcov, akého dátového typu sú hodnoty v jednotlivých stĺpcoch, či vytvárať skupiny spolu-súvisiacich tabuliek, s ktorými sa môžeme stretnúť napríklad v databázovom svete. Z tohto dôvodu nie sme schopní jednoducho validovať takéto súbory, čo vedie k častým chybám ako sú napríklad zlé dátové typy v stĺpci, chýbajúce hodnoty v záznamoch a podobne.

Tieto dôvody boli podnetom pre vznik doporučení CSV on the Web skupinou W3C Working Group v roku 2016. Definujú mimo iné aj vlastný tabuľkový model, ktorý slúži ako abstrakcia nad konkrétnymi dátovými formátmi a taktiež metadátový slovník, ktorý poskytuje spôsob pre anotáciu CSV súborov.

Takýto metadátový slovník nám umožní vytvárať metadátové súbory k CSV súborom, ktoré dodajú všetky vyššie spomenuté, potrebné a chýbajúce informácie k validácii, transformácii či prezentácii tabuľkových dát. Vydaná bola taktiež sada referenčných integračných testov, ktorá slúži k vyhodnoteniu funkčnosti jednotlivých procesorov takýchto súborov.

Dôležité je však, aby si producenti CSV súborov podľa doporučení CSV on the Web mohli jednoducho a pohodlne overiť, či nimi vyprodukované tabuľkové a metadátové súbory podliehajú štandardom.

V tejto práci sa preto zameriame na vytvorenie validátora CSV súborov, ktorý používateľom poskytne jednoduché rozhranie pre validáciu v podobe knižnice, CLI aplikácie, webovej služby a webovej aplikácie, čo umožní produkciu kvalitnejších CSV súborov.

V úvode práce sa čitateľ oboznámi s najčastejšími dátovými formátmi pre zápis tabuľkových dát, bude mu vysvetlený obsah doporučení CSV on the Web v podobe tabuľkového modelu a metadátového slovníku, čo mu dodá potrebné teoretické znalosti k pochopeniu práce.

Neskôr si priblížime existujúce implementácie validátorov, ktoré sú však málo zdokumentované, nedostatočne udržiavané, používateľsky neprívetivé, či dokonca, nefunkčné vzhľadom k referenčným testom.

Priblížime si taktiež na návrh architektúry jednotlivých častí validátora, návrh

používateľského rozhrania pre webovú aplikáciu, webovú službu či CLI aplikáciu.

Neskôr sa pozrieme na konkrétne implementačné detaily, ktoré vyplynuli zo samotného vývoja aplikácie. Následne sa používateľ môže oboznámiť s dokumentáciou softvérového diela v podobe používateľskej, vývojárskej a administratívnej dokumentácie, ktoré boli pre väčšiu použiteľnosť bakalárskej práce zverejnené pomocou technológie Github Pages v anglickom jazyku.

Spoločne si overíme funkčnosť validátora za pomoci integračných a unit testov a otestujeme výkon knižnice, aby sme používateľom priblížili, čo môžu od validátora po výkonnostnej stránke očakávať.

Na záver si validátor vyhodnotíme a porovnáme s referenčnými riešeniami vzhľadom k implementácii referenčných integračných testov, ale takisto aj z hľadiska výkonu validácie na tabuľkových súboroch z praxe.

# 1. Tabuľkové dáta

V nasledujúcej kapitole definujeme tabuľkové dáta 1.1.1. Pozrieme sa taktiež na problémy spojené s tabuľkovými dátami v praxi a taktiež predstavíme najčastejšie a najbežnejšie dátové formáty pre prácu s tabuľkovými dátami.

## 1.1 Úvod

V tejto sekcii si definujeme pojem tabuľkové dáta 1.1.1 a bližšie priblížime problémy z praxe, s ktorými sa môžeme typicky stretnúť pri práci s nimi.

**Definícia 1.1.1. Tabuľkové dáta** sú dáta, ktoré sú štruktúrované do riadkov, kde každý riadok obsahuje informácie ohľadom istej veci. Takáto **vec** je rovnaká pre každý riadok tabuľkových dát. Každý riadok obsahuje rovnaký počet buniek (niektoré môžu byť prázdne), ktoré poskytujú hodnoty vlastností danej veci popísanej daným riadkom. V tabuľkových dátach hodnoty buniek v jednom stĺpci poskytujú hodnoty práve jednej **vlastnosti**. Napríklad 3. stĺpec môže vyjadrovať vlastnosť vek.

### 1.1.1 Tabuľkové dáta v praxi

S tabuľkovými dátami 1.1.1 sa v praxi stretol určite každý z nás.

Tabuľkové dáta 1.1.1 sa častokrát prenášajú na webe v textovom formáte nazývanom CSV. Tento formát je však nedostatočne špecifikovaný, a preto sa môžeme stretnúť s viacerými jeho variantami.

Niektorí používajú pojem CSV na označenie ľubovolného textového formátu so separátormi. Iní sa držia prísnejšie definovaného významu pomocou RFC4180 1.3.

V neskoršej kapitole 2.1 si definujeme tabuľkový dátový model, ktorý slúži ako abstrakcia nad rôznymi syntaxami a dátovými formátmi používanými v praxi pri výmene tabuľkových dát. Tento model poskytuje anotácie resp. metadáta o množinách jednotlivých tabuliek, riadkov, stĺpcov či buniek. Tieto anotácie sú väčšinou poskytnuté za pomoci dodatočných súborov nazývaných metadátový súbor (metadata file), v sekcii 2.4 si ukážeme, ako takýto metadátový súbor lokalizovať. Kapitola 3 popisuje, čo môže daný metadátový súbor obsahovať.

#### Výhody tabuľkových dát

Aj napriek mnohým štandardizačným nedostatkom ponúkajú tabuľkové dáta množstvo výhod:

- štruktúrovanosť - tabuľkové dáta sú organizované do jasne stanovenej štruktúry, čo nám umožňuje ľahšie čítanie a pochopenie zložitých informácii ľudmi. Vysoká štruktúrovanosť taktiež zvyšuje strojovú čitateľnosť takýchto formátov.

- kompaktnosť - tabuľkové formáty ako je CSV využívajú len minimum nadbytočných informácii pre uloženie dát, akými sú napríklad čiarky v CSV

súbore. Na druhú stranu formáty, ako je napríklad XML, obsahujú množstvo pomocných značiek, ktoré zvyšujú veľkosť takýchto súborov.

- ľudská čitateľnosť - formáty ako CSV sú navrhnuté pre spracovávanie ako strojmi, tak aj ľuďmi, čo robí takéto formáty jednoduchými.
- efektívne dotazovanie - tabuľkové dátové formáty umožňujú rýchle dotazovanie nad jednotlivými záznamami pomocou jazykov akými sú napríklad SQL.
- interoperabilita - množstvo systémov už podporuje otvorené tabuľkové formáty, čo uľahčuje zdieľanie dát a spoluprácu takýchto systémov.

### Produkcia tabuľkových dát

Tabuľkové dáta môžu byť produkované viacerými spôsobmi:

- manuálnou produkciou používateľmi - tabuľkové dáta sú vytvárané manuálne používateľmi za pomoci aplikácii na ich vytváranie, akými sú napríklad *Microsoft Excel* či *Google Docs*.
- automatickou kolekciou - dáta môžu byť automaticky zbierané zo senzorov, napríklad teplota vzduchu v danom meste, a ukladané do tabuľkovej podoby, ktorá je pre takéto prípady použitia jednoduchá a prirodzená.
- dáta môžu byť extrahované už z existujúcich dát, akými sú napríklad webové stránky a následne uložené do tabuľkovej podoby. Podobnú funkcionalitu poskytujú napríklad web scrapery.
- generovaním - pri simulácii, modelovaní a iných výpočtových metódach môžu byť tabuľkové dáta generované na reprezentáciu hypotetických scenárov či predikcii.
- množstvo relačných databáz umožňuje exportovať ich obsah práve v podobe tabuľkových dát, nakoľko je to najprirodzenejší spôsob reprezentácie relačného dátového modelu.

### Spracovávanie tabuľkových dáta

Tabuľkové dáta môžu byť spracovávané viacerými spôsobmi. Môžeme si ich napríklad nahrať do aplikácii pre tvorbu, úpravu a prácu s tabuľkovými dátami, akými sú napríklad *Google Docs* či *Microsoft Excel*.

Tabuľkové dáta môžu byť spracovávané transformátormi, ktoré umožňujú transformovať tabuľkové súbory do iných formátov akými sú napríklad RDF či XML.

Spracovávať tabuľkové dáta musia aj validátory, ktoré pomáhajú zvyšovať kvalitu produkovaných tabuľkových dát.

Aplikácie pre prezentáciu dát, napríklad analytických, nám umožňujú pracovať s tabuľkovými dátami a vizualizovať ich pomocou tabuliek či grafov.

Tabuľkové dáta sú vhodnými taktiež pre importovanie dát do nie len relačných databázových systémov a taktiež na ich exportovanie, vďaka ich kompaktnosti a štruktúrovanosti.

## 1.2 TSV

V tejto sekcii si priblížime menej populárny, no stále využívaný dátový formát zvaný TSV a spôsoby jeho využitia.

**Definícia 1.2.1. Tab-separated values (TSV)** je jednoduchý textový formát pre uloženie tabuľkových dát. Jednotlivé záznamy sú oddelené znakmi nových riadkov a jednotlivé vlastnosti v rámci záznamu sú oddelené tabom. Jediným štandardom pre TSV je špecifikácia media typu vydaná autoritou IANA[3].

### 1.2.1 Použitie

TSV[3] je široko podporovaný formát, používaný na jednoduchý prenos tabuľkových dát medzi dvomi systémami podporujúcimi tento formát. Môžeme ho použiť napríklad na prenos dát z databázy.

### 1.2.2 Escapovanie

TSV dosahuje svojej jednoduchosti aj vďaka zakázaniu tab charakterov v rámci jednotlivých vlastností. Kvôli čomu je potrebné dané charaktery vyskytujúce sa vo vlastnostiach escapovať. Následujúca tabuľka 1.1 popisuje escape sekvencie a ich význam.

| Escape sekvencia | Význam |
|---|---|
| `\n` | line feed |
| `\t` | tab |
| `\r` | carriage return |
| `\\` | backslash |

Tabuľka 1.1: Escape sekvencie a ich význam

### 1.2.3 Konce riadkov

Jediná špecifikácia[3] hovorí, že záznamy sú oddelené pomocou EOL(End Of Line), konkrétne charaktery však nie sú špecifikované, preto sa v praxi stretneme aj s oddeľovaním za pomoci LF(line feed) typickým pre Unixový svet, či za pomoci kombinácie CR(carriage return) + LF typickým pre svet Windows.

## 1.3 RFC4180

Prvý pokus o štandardizáciu a aj najrozšírenejšiu variantu CSV súborov v súčastnosti prinieslo RFC 4180[1]. V tejto sekcii si priblížime vlastnosti definované týmto dokumentom pre CSV súbory.

RFC4180[1] definuje, aké vlastnosti by mal dodržať validný CSV súbor. Definuje následujúce pravidlá:

1. Každý záznam v tabuľkových dátach by sa mal nachádzať na osobitnom riadku. Jednotlivé riadky by mali byť oddelené pomocou dvojice znakov CR+LF

2. Posledný riadok môže, no nemusí obsahovať dvojicu znakov CR+LF.

3. Validný CSV súbor môže obsahovať na prvom riadku hlavičku, ktorá obsahuje rovnaký počet stĺpcov ako všetky zvyšné záznamy v tabuľke.

4. Hlavička a každý záznam tabuľky môže obsahovať viacero vlastností, ktoré sú oddelené pomocou čiarky. Každý riadok musí obsahovať rovnaký počet vlastností.

5. Každá vlastnosť môže byť uzavretá v dvojitých úvodzovkách("), ktoré sa používajú ako escape znak. Ak vlastnosť nie je uzavretá v dvojitých úvodzovkách, samotný znak dvojitej úvodzovky sa nemusí v položke objaviť.

6. Vlastnosti, obsahujúce znaky CR,LF, dvojité úvodzovky a čiarky musia byť uzavreté v dvojitých úvodzovkách.

7. Ak sa použijú dvojité úvodzovky na uzavretie vlastnosti, potom dvojitá úvodzovka použitá vnútri vlastnosti sa musí zdvojiť.

Definuje taktiež základné kódovanie ako US-ASCII, no neskoršie RFC7111 [2] ho predefinuje na UTF-8.

## 1.4 RFC7111

RFC7111 slúži ako update pre RFC4180[1] a rozširuje ho. V tejto sekcii si priblížime zmeny, ktoré nastali s vydaním RFC7111, ktoré sa stalo najmodernejším štandardom pre CSV súbory.

### 1.4.1 Zmeny voči RFC4180

**RFC7111** prináša 2 zmeny:

- Zavádza interpretáciu používania URI fragmentu, ktorý je definovaný v RFC3986[4].

- Taktiež mení defaultné kódovanie pre CSV súbory z US-ASCII na **UTF-8**.

### 1.4.2 Fragment identifier

Podobne ako pri RFC5147[5], ktoré definuje fragment identifikátory pre textové súbory, odkazovanie sa na špecifické časti dokumentu môže byť užitočné aj pre CSV súbory, pretože to umožňuje používateľom a aplikáciam vytvárať viac špecifické referencie a odkazy. Používateľ sa môže odkázať na konkrétny bod záujmu vrámci zdroja a nie výlučne na celý zdroj.

Fragment identifikátory sú užitočné len vtedy, ak ich implementuje klient, keďže sú interpretované výlučne klientom.

Napríklad identifikátor `#row` nám umožní špecifikovať riadky CSV súboru, na ktoré sa chceme odkázať:

```
http://example.com/data.csv#row=4
```

## 1.5 Best-practice CSV podľa odporúčaní CSV on the Web

Neexistuje žiadny štandard pre CSV (okrem RFC7111[2]), a preto sa autori tabuľkového modelu 2.1 rozhodli vydať doporučenie. Definujú metódu pre zapisovanie tabuľkových dát podliehajúcich tabuľkovému dátovému modelu 2.1 v podobe CSV. Doporučujú autorom dodržať následujúce obmedzenia, keďže implementácie by mali takto formátované CSV súbory správne a konzistentne spracovávať:

- **Content Type** - správne nastavený content-type pre CSV súbor by mal byť `text/csv`. Napríklad, keď je CSV posielané za pomoci protokolu `HTTP`, v odpovedi by mala byť zahrnutá hlavička `Content-Type` so správne nastavenou hodnotou na `text/csv`.

- **Kódovanie** - CSV súbory by mali byť zakódované pomocou kódovania `UTF-8` a mali by podliehať `Unicode Normal Form C` definovanej v `UAX15`[6]. Ak je použité iné kódovanie, malo by byť špecifikované za pomoci `charset` parametra v rámci `Content-Type` hlavičky.

- **Konce riadkov** - konce riadkov, oddeľujúce jednotlivé záznamy v rámci tabuľky, by mali byť v CSV súbore buď dvojica `CR` a `LF` (`U+000D U+000A`) alebo `LF` (`U+000A`). Podpora pre konce riadkov v podobe `LF` bola pridaná pre platformy iné ako `Windows`, a tým sa toto doporučenie odlišuje od RFC7111[2].

- **Riadky** - každý riadok CSV súboru musí obsahovať rovnaký počet čiarkou-oddelených vlastností. Vlastnosti obsahujúce čiarky, konce riadkov či úvodzovky by mali byť escapované tak, že celú vlastnosť ohraničíme úvodzovkami. Ak chceme použiť úvodzovky vrámci escapovanej vlastnosti, zdvojíme ich.

- **Hlavičky** - prvý riadok každého CSV súboru by mal obsahovať čiarkou-oddelené názvy stĺpcov. Takýto riadok sa nazýva **hlavičkový riadok** *(header line)*, ktorý poskytne názvy jednotlivých stĺpcov. Ak `CSV` súbor napriek tomu hlavičku neobsahuje, je potrebné to indikovať za pomoci hlavičky `Content-type` a parametru `header=absent`.

  Napríklad: `Content-Type: text/csv;header=absent`

# 2. Model pre tabuľkové dáta a metadáta

V nasledujúcej kapitole si priblížime tabuľkový dátový model[7], ktorý je súčasťou špecifikácie CSV on the Web[8]. Definujeme si základné pojmy vyskytujúce sa v špecifikácii, a taktiež si ukážeme niektoré základné mechanizmy pre vytváranie a spracovávanie takéhoto modelu.

## 2.1 Tabuľkový dátový model

**Anotovaný tabuľkový dátový model** *(annotated tabular data model)* je model pre tabuľky, ktoré sú anotované metadátami.

**Anotácie** *(annotations)* poskytujú informácie o bunkách, riadkoch, stĺpcoch, tabuľkách a skupinách tabuliek s ktorými sú asociované. Hodnoty týchto anotácií môžu byť listy, štruktúrované objekty alebo atomické hodnoty.

**Základné anotácie** *(core annotations)* ovplyvňujú práce procesorov, môžu byť však prítomné aj ďalšie, doplňujúce anotácie. Pod pojmom **procesor** označujeme program spracovávajúci tabuľkové dáta, a teda týmto pojmom označujeme aj samotný validátor.

Anotácie môžu byť zapísané priamo v metadátovom súbore 3.1, vnorené v tabuľkovom súbore alebo vytvorené v procese generovania anotovanej tabuľky.

Stringové hodnoty v rámci tabuľkového dátového modelu **musia** obsahovať iba *Unicode* charaktery.

V nasledujúcich pod-sekciách si popíšeme jednotlivé konštrukty používané v tabuľkovom modely.

### 2.1.1 Skupina tabuliek

**Skupina tabuliek** *(Table group)* popisuje množinu anotovaných tabuliek 2.1.2 a s nimi zviazané anotácie. Medzi **základné anotácie** skupiny tabuliek patria:

- **id** - identifikátor skupiny tabuliek.
- **notes** - poznámky k danej skupine tabuliek.
- **tables** - zoznam tabuliek 2.1.2 patriacich do skupiny tabuliek. Skupina tabuliek **musí** obsahovať aspoň 1 tabuľku 2.1.2.

Skupiny tabuliek môžu obsahovať ľubovoľné množstvo dodatočných anotácií. Patria tu napríklad titulné názvy, popisy skupiny tabuliek, zdroj tabuľkových dát či odkazy na iné skupiny tabuliek s podobným obsahom.

### 2.1.2 Tabuľky

**Anotovaná tabuľka** *(annotated table)* je tabuľka anotovaná dodatočnými metadátami. Medzi **základné anotácie** tabuliek patria:

- **columns** - zoznam stĺpcov 2.1.2 tabuľky. Tabuľka **musí** obsahovať aspoň 1 stĺpec 2.1.2 a poradie stĺpcov 2.1.2 je signifikantné a musí byť zachované.
- **table direction** - smer, v ktorom sa majú zobrazovať stĺpce 2.1.2 danej tabuľky.
- **foreign keys** - zoznam cudzích kľúčov tabuľky.
- **id** - identifikátor tabuľky.
- **notes** - poznámky k danej tabuľke.
- **rows** - zoznam riadkov 2.1.4 danej tabuľky. Tabuľka **musí** obsahovať aspoň jeden riadok 2.1.4. Poradie riadkov 2.1.4 je signifikantné a musí byť zachované.
- **schema** - *URL* odkazujúce sa na schému 3.4.4, ktorá opisuje danú tabuľku.
- **suppress output** - vyjadruje, či by mal byť výsledok vygenerovaný danou tabuľkou potlačený na výstupe.
- **transformations** - zoznam špecifikácií popisujúcich transformáciu danej tabuľky do iných dátových formátov ako napríklad JSON či RDF.

Tabuľky môžu obsahovať ľubovoľné množstvo dodatočných anotácií. Patria tu napríklad titulné názvy, popisy tabuliek, zdroj tabuľkových dát, či odkazy na iné tabuľky s podobným obsahom.

### 2.1.3 Stĺpce

**Stĺpec** *(column)* reprezentuje vertikálne usporiadanie buniek 2.1.5 v tabuľke 2.1.2 . Medzi **základné anotácie** stĺpcov patria:

- **datatype** - očakávaný dátový typ 2.1.5 pre bunky v danom stĺpci.
- **default** - defaultná hodnota pre bunky 2.1.5, ktoré sú prázdne.
- **lang** - kód očakávaného jazyka pre bunky 2.1.5v danom stĺpci. Kód je vyjadrený vo formáte BCP47 [9]
- **name** - názov daného stĺpca.
- **null** - string (*reťazec*), či stringy. Ak sa stringová hodnota bunky 2.1.5 v danom stĺpci zhoduje s jednou z týchto hodnôt, je vyhodnotená ako `null` .
- **number** - poradie stĺpca v rámci tabuľky 2.1.2. Číslovanie začína od 1.
- **ordered** - vyjadruje, či má byť poradie hodnôt v rámci jednej bunky 2.1.5 zachované.
- **required** - boolovská hodnota vyjadrujúca možnosť výskytu prázdnych buniek 2.1.5 v danom stĺpci.

- **separator** - stringová hodnota, na základe ktorej získame viaceré hodnoty jednej bunky 2.1.5. Hodnoty získame rozdelením stringovej hodnoty bunky 2.1.5 na základe separátoru.
- **source number** - absolútna pozícia stĺpca v súbore, v ktorom sa daný stĺpec nachádza. Číslovanie začína od 1.
- **suppress output** - vyjadruje, či by mal byť výsledok vygenerovaný daným stĺpcom potlačený na výstupe.
- **table** - odkaz na tabuľku 2.1.2, v ktorej sa daný stĺpec nachádza.
- **text direction** - smer textu buniek 2.1.5 v danom stĺpci.
- **titles** - množina ľudsky-čitateľných titulných názvov pre daný stĺpec. Každý z názvov môže mať s ním asociovaný jazykový kód [9].
- **virtual** - boolovská hodnota vyjadrujúca, či je daný stĺpec virtuálny. **Virtuálny stĺpec** sa používa k rozšíreniu dátového zdroja, pre pokročilejšie transformácie do iných formátov, napríklad do formátu `RDF`.

Stĺpce môžu obsahovať navyše doplňujúce informácie, ako napríklad popis daného stĺpca.

### 2.1.4 Riadky

**Riadok** *(row)* reprezentuje horizontálne usporiadanie buniek 2.1.5 vrámci tabuľky 2.1.2. Medzi **základné anotácie** riadkov patria:

- **cells** - zoznam buniek 2.1.5 v danom riadku. Riadok **musí** obsahovať jednu bunku 2.1.5 z každého stĺpca 2.1.2 tabuľky. Poradie buniek 2.1.5 vrámci riadku je signifikantné a musí byť zachované.
- **number** - pozícia riadka vrámci danej tabuľky 2.1.2. Číslovanie začína od 1.
- **primary key** - zoznam buniek 2.1.5, ktorý dokopy vytvára unikátny identifikátor pre daný riadok.
- **titles** - množina ľudsky-čitateľných titulných názvov pre daný riadok. Každý z názvov môže mať priradený jazykový kód BCP47[9].
- **referenced rows** - zoznam párov (cudzí kľuč, riadok), kde riadok sa nachádza v tej istej skupine tabuliek 2.1.1.
- **source number** - absolútna pozícia riadka v súbore, v ktorom sa daný riadok nachádza.
- **table** - odkaz na tabuľku 2.1.2, v ktorej sa daný riadok nachádza.

Riadky môžu obsahovať ľubovoľné množstvo dodatočných anotácií. Patrí tu napríklad istota informácie obsiahnutej v danom riadku či informácia o zdroji daného riadku.

### 2.1.5 Bunky

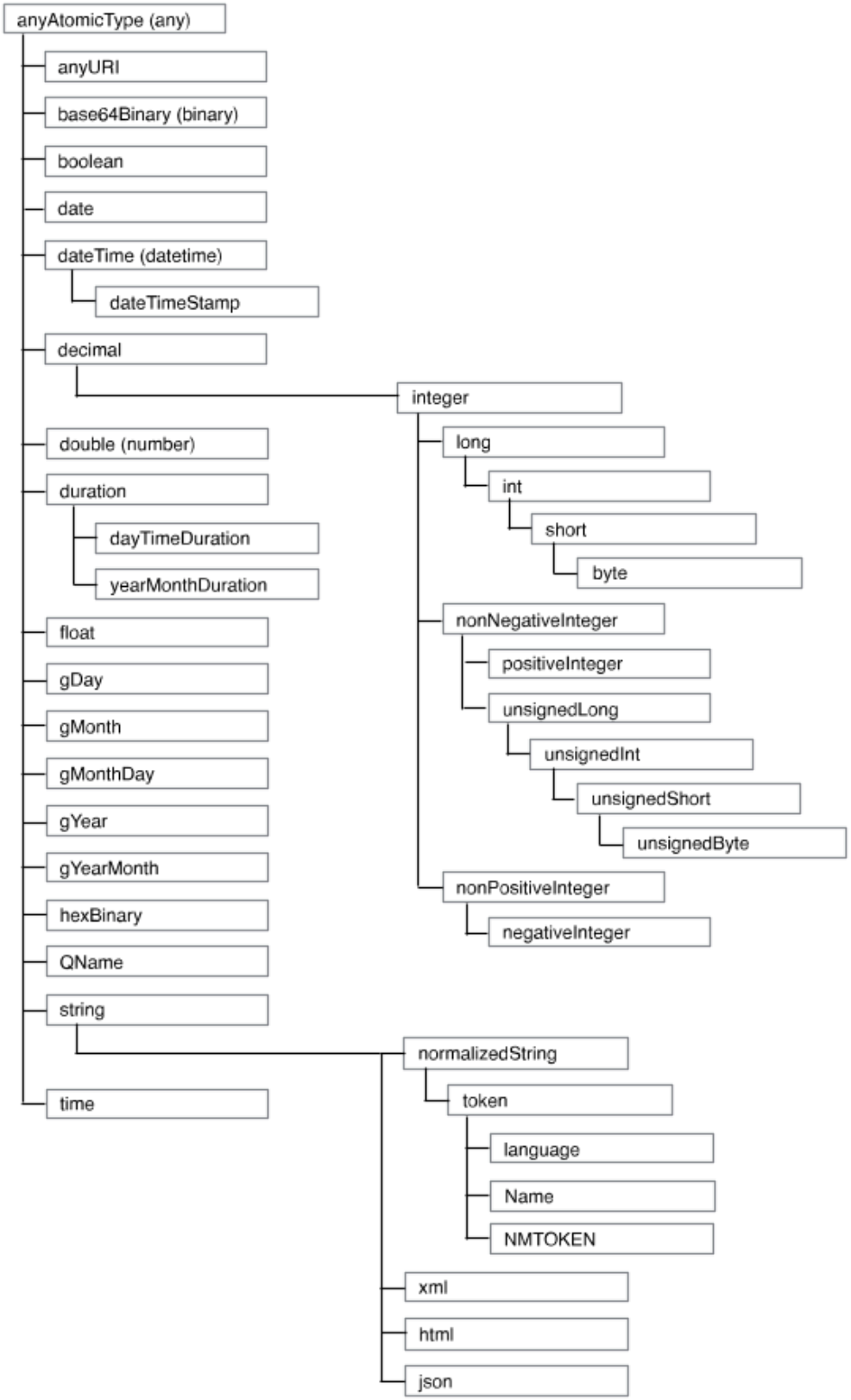
**bunka** *(cell)* reprezentuje prienik riadka 2.1.4 so stĺpcom 2.1.2 v rámci tabuľky 2.1.2. Medzi **základné anotácie** buniek patria:

- **column** - odkaz na stĺpec 2.1.2, v ktorom sa daná bunka nachádza.
- **errors** - množina validačných chýb *(errors)* vzniknutých pri parsovaní hodnoty bunky.
- **ordered** - vyjadruje, či hodnoty v danej bunke sú zoradené v prípade, že daná bunka obsahuje list hodnôt.
- **row** - odkaz na riadok 2.1.4, v ktorom sa daná bunka nachádza.
- **string value** - nespracovaná textová hodnota bunky priamo v `CSV` súbore.
- **table** - odkaz na tabuľku 2.1.2, v ktorej sa daná bunka nachádza.
- **text direction** - smer textu v danej bunke.
- **value** - sémantická hodnota bunky. Táto hodnota môže byť **list hodnôt**, z ktorých každá môže mať priradený **dátový typ** 2.1.5 iný ako string, prípadne jazykový kód BCP47 [9].

## 2.2 Dátové typy

Stĺpce 2.1.2 a bunky 2.1.5 môžu byť anotované dátovým typom *(datatype)*. Tento Dátový typ indikuje typ hodnoty získanej parsovaním stringovej hodnoty bunky.

Dátové typy sú založené na podmnožine dátových typov definovaných v *XML schéme* [10] a sú znázornené na obrázku Figure 2.1.




Obr. 2.1: Zabudované dátové typy, založené na *XML schéme* [10].


Medzi **základné anotácie** dátových typov patria:

- **id** - absolútne *URL* identifikujúce dátový typ.

- **base** - absolútne *URL* identifikujúce dátový typ od ktorého je daný dátový typ odvodený.
- **format** - string či objekt popisujúci formát hodnoty bunky 2.1.5 daného dátového typu.
- **length** - číslo určujúce presnú dĺžku hodnoty bunky 2.1.5.
- **minimum length** - číslo určujúce minimálnu dĺžku hodnoty bunky 2.1.5 daného dátového typu.
- **maximum length** - číslo určujúce maximálnu dĺžku hodnoty bunky 2.1.5 daného dátového typu.
- **minimum** - číslo vyjadrujúce maximálnu (inkluzívnu) hodnotu bunky 2.1.5 daného dátového typu.
- **maximum** - číslo určujúce maximálnu (inkluzívnu) hodnotu bunky 2.1.5 daného dátového typu.
- **minimum exclusive** - číslo vyjadrujúce minimálnu exkluzívnu hodnotu bunky 2.1.5 daného dátového typu.
- **maximum exclusive** - číslo vyjadrujúce maximálnu exkluzívnu hodnotu bunky 2.1.5 daného dátového typu.

## 2.3 Spracovanie tabuliek

V nasledujúcej sekcii si priblížime postup, akým procesory spracovávajú tabuľkové dáta. Vo väčšine prípadov začne validátor spracovávanie práve z metadátového súboru 3.1. V takomto prípade pristupujeme k metadátovému súboru, ako k prepisujúcemu *(overriding)* 1 a nepokračujeme v ďalšej lokalizácií metadát. Tento fakt nám môže pomôcť ušetriť nadbytočné požiadavky na lokalizáciu metadát 2.4, a tým urýchliť proces spracovávania tabuľkových dát.

Existujú však aj prípady, kedy validátor začne spracovávanie z tabuľkového súboru. V takomto prípade musí prebehnúť lokalizácia metadát 2.4. Následne bude tabuľkový súbor spracovaný rovnako, ako v prípade, keď začíname z práve lokalizovaného metadátového súboru.

Nasleduje proces kontroly kompatibility 3.4.3 lokalizovaných metadát s tabuľkovým súborom, normalizácie URL 2.3.3 adries, parsovanie hodnôt buniek 2.1.5 a s ním spojená validácia hodnôt buniek 2.1.5.

### 2.3.1 Vytvorenie anotovanej tabuľky

Po lokalizácii metadát 2.4, je metadátový súbor normalizovaný 3.5.

Ak začíname metadátovým súborom, tento krok zahŕňa normalizáciu daného metadátového súboru 3.5 a overenie kompatibility 2.3.2 vnorených metadát pre každý tabuľkový súbor, na ktorý sa odkazuje daný metadátový súbor.

Ak začíname z tabuľkového súboru, tento krok zahŕňa lokalizáciu metadát 2.4 a jeho následnú normalizáciu.

Ak proces začína **tabuľkovým súborom** validátor:

1. Získa daný tabuľkový súbor
2. Získa prvý metadátový súbor 2.4 .
3. Pokračujeme rovnako, ako v prípade, kedy začíname metadátovým súborom.

Ak proces začína **metadátovým súborom** validátor:

1. Získa daný metadátový súbor *(UM)*.
2. Normalizuje daný metadátový súbor na základe postupu zo sekcie 3.5 .
3. Pre každú tabuľku 2.1.2 z *UM*:
   (a) Získa deskriptor dialektu *(DD)* 3.4.9, buď priamo z metadátového súboru *UM*, prípadne na základe hlavičiek `HTTPS`, ktoré boli nájdené pri získavaní vzdialeného (*remote*) tabuľkového súboru.
   (b) Naparsuje daný tabuľkový súbor na základe *DD* a vytvorí základný tabuľkový model *(T)* a vyextrahuje vnorené metadáta *(EM)* z hlavičky.
   (c) Overí, že *TM* sú kompatibilné s na základe postupu z pod-sekcie 3.4.3.
   (d) Použije metadata *TM* na pridanie anotácií do tabuľkového modelu *T*.

### 2.3.2 Kompatibilita metadát

Pri spracovaní tabuľkového súboru za pomoci lokalizovaného metadátového súboru 2.4, validátor musí overiť, že tabuľkový a metadátový súbor sú **kompatibilné**. Typicky vy-extrahujeme vnorené metadáta z hlavičky tabuľkového súboru a následne overíme ich kompatibilitu s metadátovým súborom na základe tabuľkovej kompatibility 3.4.3.

### 2.3.3 Normalizácia URL

Lokalizácia metadát a kontrola kompatibility zahŕňa porovnávanie URL adries. Pri porovnávaní URL adries musí validátor použiť *Syntax-based normalizáciu*[4].

Validátor musí použiť *Schema-based Normalizáciu* pre protokoly `HTTP(80)` a `HTTPS(443)`.

### 2.3.4 Parsovanie buniek

Na rozdiel od iných dátových modelov, sú tabuľkové dáta určené pre jednoduché prezeranie ľuďmi. Preto je bežné, že dáta sú reprezentované ľudsky čitateľným spôsobom. Vlastnosti ako *default, lang, null, required* či *separátor* poskytujú informácie potrebné pre parsovanie z ich stringovej do sémantickej podoby.

Na základe týchto informácii validátor kontroluje, či sú dáta v tabuľke správne uložené.

Napríklad, ak máme stĺpec, ktorý má definovaný dátový typ v metadátovom súbore následovne:

`"datatype": "integer"`

pre bunku so stringovou hodnotou `"99"` bude jej sémantická hodnota `99`. Avšak, ak bunka obsahuje hodnotu, ktorá nie je validné číslo (*integer*), validátor vygeneruje validačnú chybu a takáto bunka nebude mať priradenú sémantickú hodnotu.

## 2.4 Lokalizácia metadát

Tabuľkové dáta môžu mať so sebou asociovaných množstvo anotácií. Aplikácia musí byť schopná nájsť súbory obsahujúce tieto anotácie na základe algoritmu, ktorý definuje poradie postupov, akými sme schopní metadata lokalizovať. Toto poradie je následovné:

1. **Prepisujúce Metadata** *(Overriding Metadata)* poskytnuté užívateľom validátora. V našom prípade validátor umožňuje používateľovi nahrať tento súbor, prípadne poskytnúť URL alebo cestu kde sa tento súbor nachádza, v závislosti na type aplikácie.

2. **Link HTTP hlavička** Ak užívateľ neposkytol validátoru potrebné metadáta 1. spôsobom, a zároveň validátor získal tabuľkové dáta pomocou `HTTP` protokolu, musí získať pripojené metadáta z **Link** hlavičky. Hlavička musí byť typu:

   - `rel="describedby"` a
   - `type=äpplication/csvm+json", type=äpplication/ld+json"` alebo `type=äpplication/json"`

   Tento súbor sa považuje za príslušný metadátový súbor len v prípade, že popisuje získaný súbor s tabuľkovými dátami. V prípade viacerých validných metadátových súborov získaných pomocou **Link** hlavičiek sa použije posledný lokalizovaný.

3. **Defaultné lokácie a Site-wide konfigurácia.** Ak sa nám nepodarilo získať metadátový súbor spôsobmi č. 1. a 2., validátor sa musí pokúsiť získať tento súbor z takzvanej well-known URI adresy: `/.well-known/csvm`, ktorá je definovaná v RFC5785[11]. Takto získaný súbor musí obsahovať na každom riadku *URI* šablónu, ktorá je definovaná v RFC6570[12]. Počínajúc prvým riadkom musí validátor:

   (a) Rozšíriť *URI* šablónu, kde za premennú url dosadí url súboru s tabuľkovými dátami, pre ktoré sa snažíme metadátový súbor lokalizovať.

   (b) Rezolvovať takto získanú URL adresu vzhľadom k URL adrese požadovaného súboru s tabuľkovými dátami.

   (c) Pokúsiť sa získať metadátový súbor zo získanej URL adresy.

   (d) Ak sa nenašiel žiadny metadátový súbor na danej URL adrese, alebo daný metadátový súbor sa neodkazuje na súbor s tabuľkovými dátami, musia byť tieto 4 kroky vykonané aj na ďalších riadkoch.

Ak sa na well-known adrese nenachádza relevantný súbor s URI šablónami, musí validátor postupovať, ako by mu bol poskytnutý súbor s nasledujúcim obsahom:

```
{+url}-metadata.json
csv-metadata.json
```

4. **Vnorené metadáta.** Viaceré syntaxe pre tabuľkové dáta umožňujú zápis metadát priamo do súboru s tabuľkovými dátami. Definícia takejto syntaxe by mala obsahovať popis, ako na-mapovať takúto syntax na anotovaný tabuľkový model. Parsovanie CSV súboru na základe základného CSV dialektu dokáže vy-extrahovať názvy stĺpcov z prvého riadku tabuľkových dát. Akékoľvek ďalšie metadáta sa už z daného CSV súboru získať nedajú.

   Pri použití neštandardného CSV dialektu prichádzame o možnosť extrahovania akýchkoľvek metadát zo samotného CSV súboru.

   Takýmto spôsobom vieme zistiť počet a názvy stĺpcov danej tabuľky. Napríklad z tabuľky:

   ```
   GID,On Street,Species,Trim Cycle,Inventory Date
   1,ADDISON AV,Celtis australis,Large Tree Routine Prune,10/18/2010
   2,EMERSON ST,Liquidambar styraciflua,Large Tree Routine Prune,6/2/2010
   ```

   vieme vyextrahovať následujúce informácie:

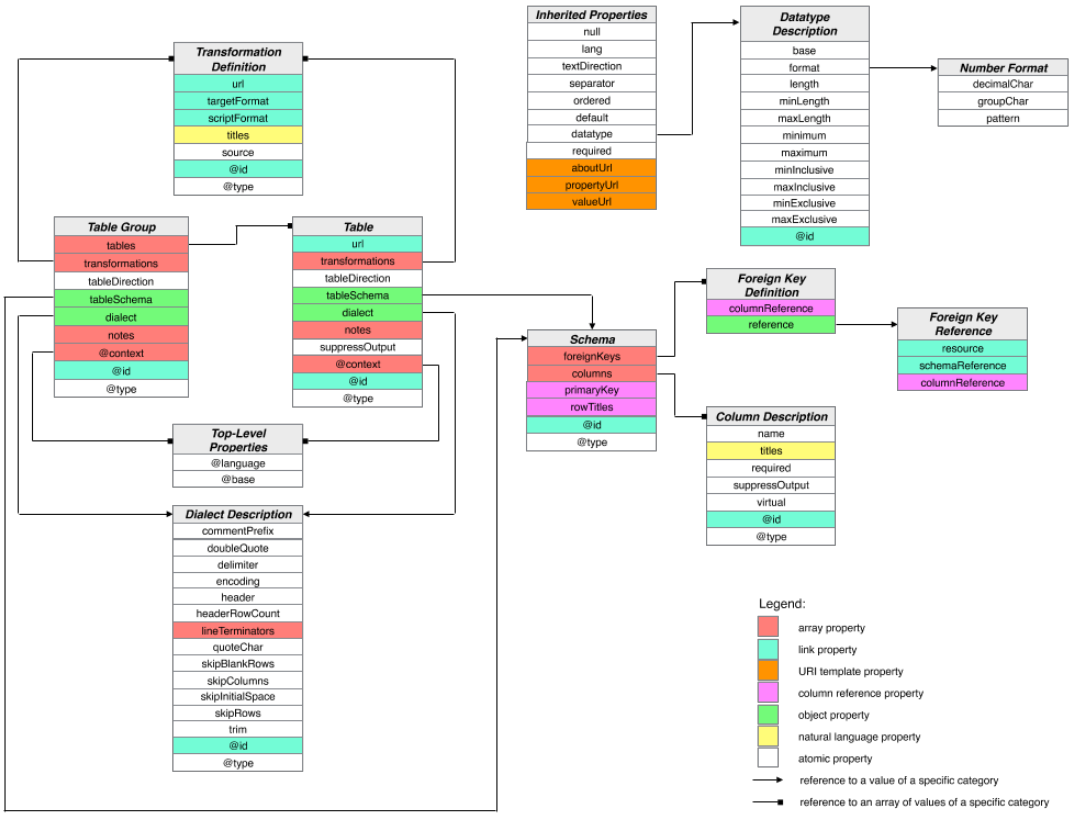
   - Tabuľka obsahuje 5 stĺpcov.
   - Názvy stĺpcov sú postupne: *GID, On Street, Species, Trimg Cycle, Inventory Date*

# 3. Slovník metadát

V tejto kapitole si uvedieme a priblížime jednotlivé konštrukty definované v metadátovom slovníku [13], ktoré sa môžu vyskytovať v súbore obsahujúcom metadáta. Neskôr si priblížime normalizáciu takýchto konštruktov.

## 3.1 Metadátový formát

Metadátový dokument je JSON dokument, ktorý má na najvyššej úrovni objekt. Tento objekt popisuje buď skupinu tabuliek 2.1.1 alebo tabuľku samotnú 2.1.2. Tento metadátový dokument môže obsahovať iné referencované alebo vnorené popisné objekty pre tabuľky a stĺpce. Takéto objekty nazývame popisujúce objekty, prípadne deskriptory.




Obr. 3.1: Obrázok ukazujúci vlastnosti rôznych metadátových deskriptorov. Copyright © 2023 W3C®[13]

## 3.2 Deskriptor

**Deskriptor** (*Descriptor/description object*) môže obsahovať viacero vlastností. Patria medzi ne:

- vlastnosti, ktoré slúžia na identifikáciu tabuľky či stĺpca, ktorého anotujú. Zhodujú sa so základnými anotáciami daných objektov v tabuľkovom dátovom modely 2.1.

- vlastnosti, ktoré sa používajú na vytváranie anotácií skupiny tabuliek 3.4.2, tabuliek 3.4.3, či stĺpcov 3.4.6, ktoré anotujú. Napríklad vlastnosť *name* na tabuľke 3.4.3.
- dedičné vlastnosti. Napríklad vlastnosť, ktorá je špecifikovaná na skupine tabuliek 3.4.2 môže byť zdedená každou tabuľkou 3.4.3 v danej skupine tabuliek 3.4.2.

## 3.3 Syntax vlastností

Deskriptor 3.2 môže obsahovať viacero typov vlastností *(properties)*:

- **Array vlastnosti** *(Array Properties)* 3.3.1
- **Link vlastnosti** *(Link Properties)* 3.3.2
- **URI šablónové vlastnosti** *(URI Template Properties)* 3.3.3
- **Stĺpec referencujúce vlastnosti** *(Column reference Properties)* 3.3.4
- **Objektové vlastnosti** *(Object Properties)* 3.3.5
- **Vlastnosti prirodzeného jazyka** *(Natural Language Properties)* 3.3.5
- **Atomické vlastnosti** *(Atomic Properties)* 3.3.6

V tejto sekcii si zadefinujeme každý typ vlastnosti a uvedieme si pár príkladov pre lepšie pochopenie.

### 3.3.1 Array vlastnosti

**Array vlastnosti** *(Array properties)* držia pole jedného alebo viacerých objektov, zvyčajne popisujúcich objektov - deskriptorov 3.2.

Napríklad: vlastnosť *tables* je array vlastnosť. Deskriptor skupiny tabuliek môže obsahovať:

```
"tables": [{
    "url": "https://example.org/countries.csv",
    "tableSchema": "https://example.org/countries.json"
}, {
    "url": "https://example.org/country_slice.csv",
    "tableSchema": "https://example.org/country_slice.json"
}]
```

v tomto prípade obsahuje *tables* vlastnosť pole dvoch tabuľkových deskriptorov 3.4.3.

### 3.3.2 Link vlastnosti

**Link vlastnosti** *(Link properties)* obsahujú referenciu na iný zdroj za pomoci `URL` adresy. Ich hodnota je **string**, ktorá sa rezolvuje (*resolve*) vzhľadom k `base URL`.

Napríklad: *url* vlastnosť patrí medzi Link vlastnosti. Deskriptor tabuliek môže obsahovať:

```
"url": "example-2014-01-03.csv"
```

v tomto prípade má *url* vlastnosť jednu hodnotu a tou je odkaz na `example-2014-01-03.csv` , ktoré rezolvujeme vzhľadom k *base URL* 3.4.1 metadátového dokumentu, v ktorom sa tento odkaz nachádza.

### 3.3.3 URI šablónové vlastnosti

**URI šablónové vlastnosti** *(URI template properties)* môžu obsahovať *URI šablónu* [12], ktorá sa použije k vygenerovaniu *URL* adresy. Tieto šablóny sú expandované v kontexte každého riadka kombináciou šablóny a premenných s hodnotami, tento proces je definovaný v RFC6570[12].

Napríklad *aboutUrl* vlastnosť obsahuje URI šablónu, ktorá slúži pre generovanie identifikátoru jednotlivých riadkov:

```
"aboutUrl": "http://example.org/example.csv#row.{_row}"
```

Následné *aboutUrl* anotácie, ktoré sú generované by vyzerali následovne:

```
http://example.org/example.csv#row.1
http://example.org/example.csv#row.2
...
```

### 3.3.4 Stĺpec referencujúce vlastnosti

**Stĺpec referencujúce vlastnosti** *(Column reference properties)* obsahujú jednu alebo viac referencií na iné stĺpcové deskriptory 3.4.6. Referencovaný stĺpcový deskriptor musí obsahovať vlastnosť *name*. Stĺpec referencujúce vlastnosti referencujú stĺpcové deskriptory 3.4.6 za pomoci hodnôt typu:

- **string** - ktorý sa musí zhodovať s vlastnosťou *name* na referencovanom stĺpcovom deskriptore.
- **pole** - pole prvkov vyššie uvedených.

Napríklad vlastnosť *primaryKey* patrí medzi stĺpec referencujúce vlastnosti. Môže obsahovať jednoduchú referenciu ako:

```
"tableSchema": {
  "columns": [{
    "name": "GID"
  }, ... ],
  "primaryKey": "GID"
}
```

### 3.3.5 Objektové vlastnosti

**Objektové vlastnosti** *(Object properties)* obsahujú buď objekt alebo referenciu na objekt za pomoci *URL* adresy. Ich hodnotami môžu byť:

- **string** - rezolvovaný ako *URL* voči *base URL* 3.4.1.
- **objekt** - interpretovaný ako štruktúrovaný objekt.

Medzi objektové vlastnosti patrí napríklad *dialect*. Môže byť vyjadrený ako *URL*, čo nám indikuje často používaný dialekt:

```
"dialect": "http://example.org/tab-separated-values"
```

alebo ako štruktúrovaný objekt:

```
"dialect": {
  "delimiter": "\t",
  "encoding": "utf-8"
}
```

### 3.3.6 Vlastnosti prirodzeného jazyka

**Vlastnosti prirodzeného jazyka** *(Natural language properties)* obsahujú stringy v prirodzenom jazyku. Ich hodnotami môže byť:

- **string** - interpretovaný ako text v prirodzenom defaultnom 3.4.1 jazyku.
- **array** - interpretované ako alternatívne texty v defaultnom 3.4.1 jazyku.
- **objekty** - ktorých vlastnosti musia byť jazykové kódy BCP47[9] a ich hodnotami sú buď stringy alebo polia poskytujúce texty v danom prirodzenom jazyku.

Napríklad vlastnosť *titles* na stĺpcovom deskriptore 3.4.6 patrí medzi vlastnosti prirodzeného jazyka.
Môže byť vyjadrená ako string v defaultnom jazyku 3.4.1:

```
"titles": "Project title"
```

Pole stringov v defaultnom jazyku 3.4.1:

```
"titles": [
  "Project title",
  "Project"
]
```

či objekt s konkrétnymi jazykovými kódmi podľa BCP47[9]:

```
"titles": {
  "en": "Project title",
  "fr": "Titre du projet"
}
```

### 3.3.7 Atomické vlastnosti

**Atomické vlastnosti** *(Atomic properties)* obsahujú **atomické hodnoty**. Ich hodnotami môžu byť:

- **čísla** - interpretované ako integery či doubly.
- **boolovské hodnoty** - interpretované ako boolovské hodnoty *(true/false)*.
- **stringy** - interpretované na základe danej vlastnosti.
- **objekty** - definované na základe danej vlastnosti.
- **polia** - polia čísel, boolovských hodnôt, stringov či objektov.

## 3.4 Konkrétne deskriptory

V tejto sekcii si priblížime konkrétne deskriptory 3.2, definované metadátovým slovníkom[13], ktoré sa môžu vyskytovať v metadátovom dokumente.

### 3.4.1 Top-level vlastnosti

**Top-level objekt** *(Top-level object)* metadátového dokumentu alebo objektu referencovaného za pomoci objektovej vlastnosti 3.3.5 musí obsahovať *@context* vlastnosť. Táto vlastnosť patrí medzi Array vlastnosti 3.3.1 a musí obsahovať jednu z následujúcich hodnôt:

- stringovú hodnotu `http://www.w3.org/ns/csvw` .
- pole obsahujúce v presnom poradí string a objekt, kde string je rovný `http://www.w3.org/ns/csvw`. Objekt reprezentuje lokálnu definíciu kontextu a obsahuje jednu či obe z následujúcich vlastnosti:
  - **@base** - atomická vlastnosť, ktorá obsahuje URL voči ktorému sa rezolvujú všetky URL v rámci metadátového dokumentu.
  - **@language** - atomická vlastnosť, ktorá definuje defaultný jazyk pre hodnoty prirodzeného jazyka 3.3.5 v rámci metadátového dokumentu.

### 3.4.2 Deskriptor Skupiny tabuliek

**Deskriptor skupiny tabuliek** *(Table group descriptor)* je JSON objekt, ktorý popisuje skupinu tabuliek 2.1.1.

**Povinné vlastnosti**

- **tables** - array vlastnosť 3.3.1, ktorá pozostáva z deskriptorov tabuliek 3.4.3.

**Nepovinné vlastnosti**

- **dialect** - objektová vlastnosť 3.3.5, ktorá poskytuje práve 1 deskriptor dialektu 3.4.9.
- **notes**- array vlastnosť 3.3.1, ktorá poskytuje arrray objektov reprezentujúcich akékoľvek anotácie na skupine tabuliek.

- **tableSchema** - objektová vlastnosť 3.3.5, ktorá poskytuje deskriptor schémy 3.4.4, ktorá sa použije ako defaultná pre všetky tabuľky 3.4.3 v skupine tabuliek 3.4.2.
- **transformations** - array vlastnosť 3.3.1, ktorá poskytuje definície transformácii, ktorá poskytuje mechanizmy na transformáciu tabuľkových dát do iných formátov.
- mnohé ďalšie.

### 3.4.3 Deskriptor tabuľky

**Deskriptor tabuľky** *(Table descriptor)* je JSON objekt, ktorý opisuje tabuľku vnútri CSV súboru.

**Povinné vlastnosti**

- **url** - link vlastnosť 3.3.2, ktorá udáva práve jedno `URL` CSV súboru, kde je tabuľka uložená, relatívne vzhľadom k lokácií metadátového dokumentu - *baseUrl* 3.4.1.

**Nepovinné vlastnosti**

- **dialect** - definované rovnako ako pre deskriptor skupiny tabuliek 3.4.2.
- **transformations** - definované rovnako ako pre deskriptor skupiny tabuliek 3.4.2.
- **tableSchema** - objektová vlastnosť 3.3.5, ktorá poskytuje deskriptor schémy 3.4.4 pre danú tabuľku.
- A ďalšie...

#### Kompatibilita tabuliek

Dva deskriptory tabuliek 3.4.3 sú kompatibilné, ak majú rovnaké normalizované vlastnosti *url* a majú kompatibilné schémy 3.4.4 .

### 3.4.4 Deskriptor schémy

**Schéma** je definícia tabuľkového formátu, ktorý môže byť spoločný pre viaceré tabuľky 3.4.3.

**Deskriptor schémy** *(Schema descriptor)* je JSON objekt, ktorý kóduje informácie o schéme, popisujúcej štruktúru tabuľky.

Všetky vlastnosti deskriptora schémy sú **nepovinné**:

- **columns** - array vlastnosť 3.3.1 obsahujúca deskriptory stĺpcov 3.4.6 danej tabuľky.
- **foreingKeys** - array vlastnosť 3.3.1 obashujúca deskriptory cudzích kľúčov 3.4.5 danej tabuľky.
- **primaryKey** - stĺpec referencujúca vlastnosť 3.3.4, ktorá obsahuje referenciu na deskriptor stĺpca 3.4.6 alebo pole takýchto referencií.
- a ďalšie...

**Kompatibilita schém**

Dve schémy sú **kompatibilné**, ak majú rovnaký počet nevirtuálnych deskriptorov stĺpcov 3.4.6, a nevirtuálne deskriptory stĺpcov 3.4.6 na rovnakom indexe sú kompatibilné. Dva deskriptory stĺpcov 3.4.6 sú kompatibilné, ak:

- jeden z deskriptorov neobsahuje ani vlastnosť *name* ani *titles* vlastnosť.
- existuje case-sensitive zhoda medzi vlastnosťami *name* jednotlivých stĺpcov.
- existuje neprázdny prienik medzi hodnotami *titles*, zhodujúce sa hodnoty musia mať zhodujúce sa jazykové kódy BCP47[9].

### 3.4.5 Deskriptor cudzieho kľúča

**Deskriptor cudzieho kľúča** *Foreign key descriptor* je JSON objekt, ktorý musí obsahovať iba následujúce vlastnosti:

- *columnReference* - stĺpec referencujúca vlastnosť 3.3.4 obsahujúca referenciu či zoznam referencií stĺpcových deskriptorov 3.4.6. Tieto deskriptory tvoria **referencujúce stĺpce**.
- *reference* - objektová vlastnosť 3.3.5 ktorá identifikuje **referencovanú tabuľku** 3.4.3 a množinu **referencovaných stĺpcov** 3.4.6.

### 3.4.6 Dekskriptor stĺpcov

**Deskriptor stĺpca** *(Column descriptor)* je JSON objekt popisujúci jeden stĺpec v rámci tabuľky. Všetky vlastnosti stĺpcového deskriptora sú **nepovinné**:

- *name* - atomická vlastnosť 3.3.6, ktorá dáva stĺpcu konkrétny názov.
- *titles* - vlastnosť prirodzeného jazyka 3.3.5, ktorá určuje alternatívne názvy pre daný stĺpec.
- *virtual* - boolovská atomická vlastnosť 3.3.6 indikujúca, či sa jedná o virtuálny stĺpec, neprítomný v skutočnom tabuľkovom súbore.
- a ďalšie ...

### 3.4.7 Dedičné vlastnosti

Stĺpce 2.1.2 či bunky 2.1.5 môžu obsahovať anotácie založené na vlastnostiach deskriptorov skupiny tabuliek 3.4.2, tabuliek 3.4.3, schém 3.4.4 či stĺpcov 3.4.6. Tieto vlastnosti sa nazývajú: **dedičné vlastnosti***(inherited properties)*. Týchto vlastnosti je vo všeobecnosti veľa, a preto si spomenieme najdôležitejšie z nich:

- *datatype* - atomická vlastnosť 3.3.6 obsahujúca buď string, ktorý tvorí hlavný dátový typ 2.1.5 bunky alebo deskriptor dátového typu.
- *default* - atomická vlastnosť 3.3.6obsahujúca string, ktorý sa používa pre vytvorenie defaultnej hodnoty bunky.
- *separator* - atomická vlastnosť 3.3.6 obsahujúca string, ktorý sa použije na oddelenie hodnôt vrámci jednej bunky.

### 3.4.8 Spoločné vlastnosti

Deskriptory skupín tabuliek 3.4.2, tabuliek 3.4.3, schém 3.4.4 či stĺpcov 3.4.6 môžu obsahovať ľubovolné množstvo **spoločných vlastností** *(common properties)*, ktorých mená sú buď absolútne *URL* alebo prefixované mená.

Napríklad deskriptor tabuľky 3.4.3 môže obsahovať `dc:description`, `dcat:keyword` či `schema:copyrightHolder` vlastnosti, poskytujúce popis, kľúčové slová či *copyright držiteľa*, tak ako je to definované v daných slovníkoch.

### 3.4.9 Deskriptor dialektu

Pred príchodom CSV[1] sa na internete pohybovalo množstvo súborov obsahujúcich tabuľkové dáta. Medzi najčastejšie patrili DSV(delimiter separated values), ktoré nie sú definované v žiadnom RFC, a teda nie sú strojovo čitateľné.

Ďalším príkladom môže byť TSV(Tab-Separated Values)[14]. Prípadne, kým bolo vydané RFC4180[1] v roku 2005, nebola dobre definovaná štruktúra samotného CSV, a preto mnohé subjekty používali rôzne varianty.

Aby W3C umožnilo pracovať aj s takýmito súbormi, umožňuje definovať popisujúci objekt dialektu. Ten umožňuje definovať špeciálny formát CSV súboru, ktorý chceme spracovať. Umožňuje definovať: separátor hodnôt v rámci jedného riadku, prefix riadku s komentárom, kódovanie, prítomnosť hlavičky, znaky konca riadku a iné.

Podrobná dokumentácia je dostupná v kapitole 5.9 v metadátovom slovníku [13].

## 3.5 Normalizácia deskriptorov

Deskriptor 3.2 je **normalizovaný** následovne:

1. Ak daná vlastnosť patrí medzi spoločné vlastnosti 3.4.8, normalizujeme ju do objektu s vlastnosťami *@value* prípadne *@language*, *@id* alebo *@type*.

2. Ak daná vlastnosť patrí medzi Array vlastnosti 3.3.1, normalizujeme každý element poľa na základe tohto algoritmu.

3. Ak daná vlastnosť patrí medzi link vlastnosti 3.3.2, vytvoríme z jej hodnoty absolútnu `URL` adresu za pomoci *baseUrl* 3.4.1 a takúto `URL` adresu normalizujeme postupom popísaným v pod-sekcii 2.3.3.

4. Ak daná vlastnosť patrí medzi objektové vlastnosti 3.3.5 a má hodnotu stringu, takýto string referencuje JSON dokument obsahujúci jeden samostatný objekt. Tento objekt získame a normalizujeme každú jeho vlastnosť rekurzívne na základe tohto algoritmu. Stringovú hodnotu vlastnosti nahradíme takýmto výsledným objektom.

5. Ak daná vlastnosť patrí medzi objektové vlastnosti 3.3.5 a má objektovú hodnotu, normalizujeme rekurzívne každú vlastnosť daného objektu na základe tohto algoritmu.

6. Ak daná vlastnosť patrí medzi vlastnosti prirodzeného jazyka 3.3.5 a jej hodnota nie je objekt, hodnotu takejto vlastnosti nahradíme objektom, ktorého vlastnosti sú jednotlivé jazykové kódy a ich hodnotami sú polia.

7. Ak daná vlastnosť patrí medzi atomické vlastnosti 3.3.6 a jej hodnota môže byť string alebo objekt, normalizujeme takúto vlastnosť špecificky na základe konkrétnej vlastnosti.

# 4. Existujúce riešenia

V tejto kapitole sa pozrieme na už existujúce implementácie validátorov CSV súborov podľa odporúčaní CSVW on the web[8]. Nahliadneme na nimi poskytované funkcionality a porovnáme si ich na základe nasledujúcich vlastností:

1. jednoduchá inštalácia na operačnom systéme Windows 11 pri iných rozhraniach, ako webová aplikácia.
2. jednoduchá inštalácia na operačnom systéme Linux-Fedora pri iných rozhraniach, ako webová aplikácia.
3. podpora schémy CSV on the Web[8].
4. detailná používateľská dokumentácia.
5. podpora pre tabuľkové súbory s kódovaním UTF-8 s BOM s priloženou schémou CSV on the Web.
6. používateľské rozhranie v podobe webovej aplikácie.
7. používateľské rozhranie v podobe CLI aplikácie.
8. podpora relatívnych ciest pri práci s lokálnymi súbormi.
9. počet implementovaných referenčných integračných testov[15].

## 4.1 CSV Lint

**CSVLint**[16] je ruby gem knižnica poskytujúca validáciu CSV súborov voči viacerým schematickým štandardom: *JSON Table Schema* a **CSV on the Web**.

### 4.1.1 Webová aplikácia

Webová aplikácia je dostupná na adrese `https://csvlint.io/`. Obsahuje príjemné používateľské rozhranie a podporuje validáciu CSV súborov vzhľadom k RFC4180[1]. Pri pokuse o validáciu csv súboru s priloženou CSV on the Web[8] schémou, validátor nefunguje a server vracia chybu s HTTP kódom 500. Naopak aplikácia umožňuje nahrať schému vo formáte *JSON Table schema*, avšak na tento fakt používateľ nie je na validačnej stránke upozornený, čo znižuje používateľskú prívetivosť. Validácia samotných CSV súborom vzhľadom k RFC4180[1] je plne podporovaná.

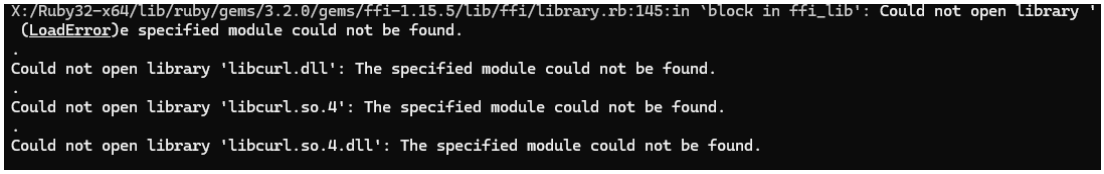
#### CLI aplikácia

### 4.1.2 CLI aplikácia

Inštalácia samotnej knižnice prebieha za pomoci Package manažéra *RubyGems*.

### Windows

Postupovali sme podľa pokynov dostupných v GitHub repozitári, no aplikáciu sa nám nepodarilo spojazdniť. V prom rade, je potrebné mať na operačnom systéme Windows 11 nainštalovaný program `Curl`, ktorý nie je jeho súčasťou. Po jeho do-inštalovaní pri snahe o validáciu jednoduchého CSV súboru dostávame následovnú chybu:




Obr. 4.1: Chyba pri používaní CSVLint na systéme Windows

Knižnica zjavne nevie nájsť nainštalovaný curl program na našom sytéme. Pravdepodobne systém nebol určený pre použitie na operačnom systéme Windows. Návod na odstránenie tejto chyby sme v dokumentácii knižnice nenašli.

Všimli sme si však, že v zdrojových súboroch existuje súbor `docker_notes_-for_windows.txt`. Na tento súbor však neexistuje žiaden odkaz z hlavnej užívateľskej dokumentácie a aj napriek postupovaniu podľa daných inštrukcii, ktoré sú pre menej skúsených docker používateľov náročné nasledovať, sa nám nepodarilo konkrétnu validáciu spustiť. Od programu sme dostali následovnú chybovú hlášku:

```
/usr/bin/env: ‘ruby\r’: No such file or directory
```

### Linux

Na operačnom systéme Linux-Fedora sme po na-inštalovaní aplikácie za pomoci príkazu:

```
gem install csvlint
```

boli schopní jednoducho validovať CSV súbory pomocou príkazu:

```
csvlint myfile.csv
```

príklad výsledku validácie na jednom z referenčných integračných testov[15]:

```
csvlint \
--schema=https://w3c.github.io/csvw/tests/test035/csv-metadata.json
.....
https://w3c.github.io/csvw/tests/test035/gov.uk/data/professions.csv
is VALID
1. check_options
...
https://w3c.github.io/csvw/tests/test035/gov.uk/data/organizations.csv
is VALID
...
https://w3c.github.io/csvw/tests/test035/senior-roles.csv
is VALID
1. inconsistent_values. Column: 5
...
```

```
https://w3c.github.io/csvw/tests/test035/junior-roles.csv
is INVALID
1. unmatched_foreign_key_reference. Row: 2,3. [""]
2. unmatched_foreign_key_reference. Row: 2,5. [""]
```

Na platforme Linux-Fedora bolo používanie bezproblémové a vedeli sme jednoducho validovať tabuľky s metadátami, ale aj bez nich. Za príjemné považujem zoskupenie chýb a varovaní podľa tabuľky.

### 4.1.3 Zhrnutie

Inštalácia CLI aplikácie od CSVLint[16] pre použitie na operačnom systéme Windows bola komplikovaná a nepodarilo sa nám ju dokončiť, naopak pre operačný systém Linux-Fedora bolo jej použitie jednoduché.

CSVLint poskytuje dve používateľské rozhrania v podobe CLI aplikácie a webovej aplikácie. Webová aplikácia avšak nepodporuje CSV on the Web[8] schéma.

Validátor poskytuje príjemnú a detailnú používateľskú dokumentáciu.

CSVLint[16] nepodporuje prácu s tabuľkovými súbormi kódované pomocou UTF-8 s BOM s priloženou CSV on the Web[8] schémou. Pri pokuse o takúto validáciu dostávame chybu indikujúcu nekompatibilitu vnorených metadát podobnú:

```
ockovani-spotreba.csv is INVALID
1. invalid_header. Row: 1,1. id
```

Problémom zrejme je, že byty reprezentujúce BOM sú priradené k obsahu prvej bunky tabuľky, ktorá reprezentuje názov prvého stĺpca. V dôsledku čoho názov prvého stĺpca tabuľky nie je kompatibilný s názvom stĺpca v metadátovom súbore.

CSVLint poskytuje dve používateľské rozhrania v podobe CLI aplikácie a webovej aplikácie. Webová aplikácia nepodporuje CSV on the Web[8] schéma.

CLI aplikácia podporuje prácu s relatívnymi cestami pri lokálnych súboroch.

CSVLint implementuje všetkých 281 referenčných integračných testov[15].

Zhrnutie jednotlivých výsledkov môžeme vidieť v tabuľke 4.1.

| **Funkcionalita** | **CSVLint** |
|---|---|
| Inštalácia Windows 11 | NIE |
| Inštalácia Linux-Fedora | ÁNO |
| Podpora schémy CSVW CLI) | ÁNO |
| Podpora schémy CSVW(Web. app) | NIE |
| Detailná užívateľská dokumentácia | ÁNO |
| Podpora UTF-8 s BOM | NIE |
| Webová aplikácia | ÁNO |
| CLI aplikácia | ÁNO |
| Relatívne cesty | ÁNO |
| 100% integračných testov | ÁNO |

Tabuľka 4.1: Zhrnutie funkcionalít poskytovaných CSVLint

## 4.2 pycsvw

**pycsvw** *(Python implementation of the W3C CSV on the Web specification)*[17] je implementácia CSV validátora na základe doporučení CSV on the Web[8].

### 4.2.1 CLI rozhranie

Prvá vec, ktorú sme postrehli po zobrazení domovskej stránky GitHub[17] repozitára projektu je chýbajúca dokumentácia, či už používateľská alebo administrátorská. Problematické je teda už samotné spustenie aplikácie.

Samotná aplikácia je knižnicou napísanou v programovacom jazyku Python.

Objavili sme súbor `setup.py`, a preto sme sa pokúsili nainštalovať potrebné závislosti pomocou príkazu:

```
pip install -e .
```

Vďaka čomu sme boli schopní na-inštalovať balíčky: `language_tags rdflib` a `uritemplate`. Následne sme sa pokúsili spustiť program za pomoci:

```
python pycsvw
```

Avšak stretli sme sa s následovným chybovým hlásením:

```
python.exe: can't find '__main__' module in 'C:\\pycsvw'
```

To nám indikovalo problém s pomenovaním súboru `main`. Z tohto dôvodu sme ho premenovali na súbor `__main__` a pokúsili sa o opätovné spustenie pomocou príkazu:

```
python pycsvw
```

avšak opäť sme boli konfrontovaní následovným chybovým hlásením:

```
  Traceback (most recent call last):
File "<frozen runpy>", line 198, in _run_module_as_main
File "<frozen runpy>", line 88, in _run_code
File "C:\csvw-parser\pycsvw\__main__.py", line 1, in <module>
  from StringIO import StringIO
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  ModuleNotFoundError: No module named 'StringIO'
```

Tento problém je pravdepodobne spôsobený verziou Python3, kde `StringIO` je súčasťou modulu `io` , nie `StringIO`. Požadovaná verzia Pythonu však nie je nikde špecifikovaná, a problémy, ktoré musí používateľ vyriešiť pred samotným spustením sú obrovské.

### 4.2.2 Zhrnutie

Knižnica spĺňa len zlomok referenčných integračných testov[15] (158/281) a náročná inštalácia spojená s nulovou dokumentáciou robia takúto aplikáciu pre bežného používateľa takmer nepoužiteľnou.

Detailné výsledky môžeme vidieť v tabuľke 4.2.

| Funkcionalita | pycsvw |
|---|---|
| Inštalácia Windows 11 | NIE |
| Inštalácia Linux-Fedora | NIE |
| Podpora schémy CSVW | - |
| Detailná užívateľská dokumentácia | - |
| Podpora UTF-8 s BOM | - |
| Webová aplikácia | NIE |
| CLI aplikácia | NIE |
| Relatívne cesty | - |
| 100% integračných testov | NIE |

Tabuľka 4.2: Zhrnutie funkcionalít poskytovaných pycsvw

## 4.3 RDF::Tabular

**RDF::Tabular**[18] je knižnica implementovaná pomocou programovacieho jazyka `Ruby`. Plne podporuje špecifikáciu CSV on the Web[8] vzhľadom k referenčným integračným testom[15]. Jej autor, **Gregg Kellogg**, patrí medzi autorov samotnej špecifikácie, a preto patrí medzi jednu z najviac funkčných a stále udržiavaných implementácii.

Inštalácia je jednoduchá, stačí mať na cieľovom operačnom zariadení, či už s operačným systémom Windows 11 alebo Linux-Fedora, nainštalovaný programovací jazyk Ruby a jeho package manažér `RubyGem`. Knižnicu nainštalujeme pomocou príkazu:

```
gem install rdf-tabular
```

Následne vieme validáciu spustiť pomocou príkazu:

```
rdf validate --input-format tabular IRI
```

Vyskúšali sme funkčnosť na jednom z integračných testov:

```
rdf validate \
--input-format tabular https://w3c.github.io/csvw/tests/test080-metadata.json
```

Výsledok tohto testu bol správny a dostali sme chybu:

```
rdf validate --input-format tabular
https://w3c.github.io/csvw/tests/test080-metadata.json
ERROR   Column has invalid property '@id': "_:foo", must not start with '_:'
```

Výsledok však obsahoval aj množstvo nesúvisiacich chybových hlásení, ktoré môžu používateľa zahltiť.

Pre prehľadnosť by bolo lepšie takéto errory nezobrazovať, a zobrazovať ich len pri použití prepínača napríklad: `-verbose`.

### 4.3.1 Zhrnutie

Inštalácia za pomoci package managera *Ruby Gem* bola bezproblémová pre oba operačné systémy Windows 11 a Linux-Fedora, avšak nepríjemné považujem za prvotnú potrebu inštalácie programovacieho jazyka Ruby.

RDF::Tabular[18] plne podporuje schému CSV on the Web[8].

Dostupná používateľská dokumentácia nie je príliš detailná, čo môže mať za dôsledok zlé použitie aplikácie. Podstatný detail, ako prítomnosť prepínača `--validate` je spomenutá len v poznámke pri príkladoch použitia.

Aplikácia nepodporuje tabuľkové súbory kódované za pomoci UTF-8 s BOM s priloženou CSV on the Web schémou. Pri pokuse o validáciu takýchto súborov dostávame chybu podobnú:

```
Column 1 doesn't match on titles: "" vs " id"
(RDF::ReaderError)
```

ktorá nasvedčuje podobnému problému ako u CSVLint 4.1, že takéto BOM byty sú pridané k názvu prvého stĺpca tabuľky a tým pádom sa názov prvého stĺpca nezhoduje s názvom špecifikovaným v metadátovom súbore.

Neposkytuje žiadne pokročilejšie používateľské rozhranie v podobe webovej aplikácie.

Aplikácia nepodporuje prácu s relatívnymi cestami, či absolútnymi cestami a pre spustenie validácie sa musíme nachádzať priamo v priečinku s tabuľkovými a metadátovými súbormi. V opačnom prípade sa stretávame s chybou podobnou:

```
No such file or directory @ rb_sysopen -
/home/drexem/desktop/school/bp/performancetesting/ockovani-spotreba.csv
(RDF::ReaderError)
```

RDF::Tabular[18] spĺňa všetkých 281 referenčných integračných testov[15].

| **Funkcionalita** | **RDF::Tabular** |
|---|---|
| Inštalácia Windows 11 | ÁNO |
| Inštalácia Linux-Fedora | ÁNO |
| Podpora schémy CSVW | ÁNO |
| Detailná užívateľská dokumentácia | NIE |
| Podpora UTF-8 s BOM | NIE |
| Webová aplikácia | NIE |
| CLI aplikácia | ÁNO |
| Relatívne cesty | NIE |
| 100% integračných testov | ÁNO |

Tabuľka 4.3: Zhrnutie funkcionalít poskytovaných RDF::Tabular

## 4.4 csvw-validator

**csvw-validátor**[19] je implementácia doporučení CSV on the Web[8] v programovacom jazyku Java. Implementácia ponúka 3 používateľské rozhrania: *Webová aplikácia*, *Webová služba* a *command-line aplikácia*.

### 4.4.1 Webová aplikácia

Webová aplikácia nás privíta veľmi príjemným a intuitívnym používateľským rozhraním. Webová aplikácia je taktiež lokalizovaná do dvoch jazykov: Anglický a Český, čo považujeme za príjemnú skutočnosť.

Webová aplikácia nám však umožňuje validovať len Tabuľky2.1.2 a neumožňuje pracovať so Skupinami tabuliek 2.1.1 u lokálnych súboroch.

Taktiež webová aplikácia nám nevysvetľuje, ako sa mapuje prípad prepisujúcich *(overriding)* metadát, teda prípad, kedy používateľ poskytne tabuľkový súbor a zároveň metadátový súbor. V takomto prípade sa nemusí metadátový súbor odkazovať na tabuľkový a mapovanie deifnuje aplikácia. Po otestovaní tohto prípadu nám bolo poskytnuté varovanie, že validácia prebehla bez metadátového súboru, čo značí, že tento samotný prípad nie je implementovaný.

Aplikácia umožňuje zapnutie striktného módu, ktorý pridá prísnejšie validačné pravidlá.

### 4.4.2 CLI aplikácia

Aplikáciu sme sa pokúsili nainštalovať postupom, ktorý je uvedený v používateľskej dokumentácii. Začali sme krokom:

```
mvn clean install
```

avšak boli sme konfrontovaný s nekompatibilnou verziou Javy (21). V dokumentácii sa píše, že aplikácia bola testovaná na verzii Java 8 a novšie by mali byť podporované, no nie sú testované. Preto sme prešli na verziu 8. Opakovali sme pokus o inštaláciu:

```
mvn clean install
```

Opäť sme dostali chybovú hlášku týkajúcu sa použitia externých repozitárov v nástroji Maven.

Riešením bolo odstránenie riadkov:

```
<mirror>
  <id>maven-default-http-blocker</id>
  <mirrorOf>external:http:*</mirrorOf>
  <name>
  Pseudo repository to mirror external repositories initially using HTTP.
  </name>
  <url>http://0.0.0.0/</url>
</mirror>
```

z konfigurácie Mavenu. Tento postup však nebol nikde v dokumentácii spomenutý, čo spôsobilo množstvo problémom pri spustení. Avšak tieto problémy mohli prameniť v nedostatočných skúsenostiach s prácou v nástroji Maven. Následne sme boli schopní vyskúšať validátor za pomoci príkazu v tvare:

```
java -jar csvw-validator-cli-app-1.0.0-SNAPSHOT.jar [-f <FILE>] [-s <SCHEMA>]
[-o <OUTPUT>] [--strict] [-h] [--rdf] [--csv]
```

konkrétne:

```
java -jar validator.jar -f http://www.w3.org/2013/csvw/tests/test286.csv
```

a dostali sme výstup:

```
Tabular URL: http://www.w3.org/2013/csvw/tests/test286.csv
Metadata URL: http://www.w3.org/2013/csvw/tests/test286-metadata.json
Result: ERROR
Strict mode: true
Total errors: 1
Warning errors: 0
Error errors: 0
Fatal errors: 1
Processed tables: 0
Processed rows: 0
Processed columns: 0
Errors:
FATAL: Cant download valid metadata from url {0}.
```

čo značí, že nebolo možné stiahnuť metadátový súbor z danej adresy. Taktiež chybová hláška:

```
FATAL: Cant download valid metadata from url {0}.
```

s prítomnosťou reťazca:

```
{0}
```

svedčí o zlom formátovaní stringu. Použili sme príklad priamo z dokumentácie a súbory sa na daných adresách naozaj vyskytujú. Dôvod vzniku tejto chyby ostáva neznámy.

### 4.4.3 Zhrnutie

Inštalácia nástroja csvw-validator[19] je vďaka distribúcii pomocou nástroja Maven jednoduchá pre oba operačné systémy, s výnimkou konfigurácie samotného Mavenu.

csvw-validator podporuje schému CSV on the Web[8].

Aplikácia obsahuje detailnú používateľskú a dokonca aj administrátorskú a vývojársku dokumentáciu.

Aplikácia podporuje prácu s tabuľkovými súbormi kódovaných pomocou UTF-8 s BOM, no na prítomnosť BOM bytov neupozorňuje.

csvw-validátor poskytuje používateľské rozhranie v podobe webovej aplikácie, webovej služby aj CLI aplikácie.

Validátor nepodporuje prácu s relatívnymi adresami k lokálnym súborom a pri pokuse o takúto validáciu hlási chybu:

```
Invalid usage of program. Use -h or --help.
At least file or schema must be specified.
```

čo značí, že takéto cesty nedokáže rozpoznať.

Validátor však neimplementuje 100% integračných testov, čo znamená, že neposkytuje plnú funkcionalitu špecifikovanú v CSV on the Web[8]. Medzi základné nedostatky patrí chýbajúca podpora pre cudzie kľúče 3.4.5 a deskriptory dialektu 3.4.9.

| Funkcionalita | csvw-validator |
|---|---|
| Inštalácia Windows 11 | ÁNO |
| Inštalácia Linux-Fedora | ÁNO |
| Podpora schémy CSVW | ÁNO |
| Detailná užívateľská dokumentácia | ÁNO |
| Podpora UTF-8 s BOM | ÁNO |
| Webová aplikácia | ÁNO |
| CLI aplikácia | ÁNO |
| Relatívne cesty | NIE |
| 100% integračných testov | NIE |

Tabuľka 4.4: Zhrnutie funkcionalít poskytovaných csvw-validator

## 4.5 Zhrnutie

Spoločné zhrnutie všetkých existujúcich riešení môžeme vidieť v tabuľke 4.5. Žiadne z týchto riešení neposkytuje plnú testovanú funkcionalitu. Niektorým chýba pokročilejšie používateľské rozhranie, iným detailná dokumentácia alebo jednoduchá inštalácia pre oba operačné systémy Windows 11 a Fedora-Linux.

Implementácia pycsvw 4.2 je takmer nepoužiteľná, a preto sa v záverečnom porovnaní nevyskytuje.

Ideálne by nami implementovaný validátor mal poskytovať všetky spomínané funkcionality. V kapitole 10 si náš validátor vyhodnotíme a dokonca zistíme, že poskytuje aj značne väčšiu výkonnosť, než väčšina referenčných implementácii.

| Funkcionalita | CSVLint | RDF::Tabular | csvw-validator |
|---|---|---|---|
| Inštalácia Windows 11 | NIE | ÁNO | ÁNO |
| Inštalácia Linux-Fedora | ÁNO | ÁNO | ÁNO |
| Podpora schémy CSVW | ÁNO(CLI) | ÁNO | ÁNO |
| Detailná užívateľská dokumentácia | NIE | NIE | ÁNO |
| Podpora UTF-8 s BOM | ÁNO | NIE | ÁNO |
| Webová aplikácia | NIE | NIE | ÁNO |
| CLI aplikácia | ÁNO | ÁNO | ÁNO |
| Relatívne cesty | ÁNO | NIE | NIE |
| 100% integračných testov | ÁNO | ÁNO | NIE |

Tabuľka 4.5: Zhrnutie funkcionalít poskytovaných csvw-validator

# 5. Analýza Validátora

V nasledujúcej kapitole si priblížime jednotlivé používateľské role typických používateľov validátora. Detailne definujeme funkčné a nefunkčné požiadavky pre validátor, a taktiež sa pozrieme na jeho typické prípady použitia.

## 5.1 Užívateľské role

1. **Developer** - vytvárajúci aplikáciu, ktorá vyžaduje prácu s CSV súbormi. Aplikácia môže napríklad prijímať CSV súbory od používateľov, nad ktorými budú bežať výpočty a pred spustením takýchto výpočtov je potrebné overiť validitu takýchto súborov.

   Iný príklad môže byť výmena tabuľkových dát medzi aplikáciami tretích strán. Pri práci s cudzími CSV súbormi bude developer vyžadovať isté kritéria, ktoré musí daný CSV súbor spĺňať - RFC4810[1] alebo odporúčania CSV on the Web[8]. Pred ďalším spracovaním prijatého súboru využije developer validátor, ktorý skontroluje validitu CSV súboru či už voči RFC4180[1] alebo odporúčaniam CSV on the Web[8]. Podobne bude developer postupovať pri odosielaní CSV súboru vygenerovaného jeho aplikáciou - overí správnosť CSV súboru a prípadne priloženého metadátového súboru.

   Pri vytváraní webovej aplikácie by tak mohol napríklad učiniť pomocou webovej služby. Ak by potreboval naopak čo najvyššiu efektivitu, mohol by využiť služby CLI aplikácie, ktorá nepotrebuje k svojmu behu nadbytočnú komunikáciu po sieti, prípadne integrovať samotnú validačnú knižnicu.

2. **Študent vzdelávajúci sa v oblasti dátových formátov** - ak študent študuje dátový formát CSV, bude sa pokúšať vytvoriť najprv validný CSV súbor vzhľadom k RFC4810[1], neskôr vzhľadom k doporučeniam CSV on the Web[8].

   Takýto študent typicky využije služby Webovej aplikácie. Tá mu poskytne možnosť rýchleho overenia validity, bez potreby akejkoľvek dodatočnej inštalácie.

## 5.2 Požiadavky

Následujúce požiadavky vznikli na základe predstavy o typickom používateľovi a jeho potrebách, v spolupráci a odbornej konzultácií vedúceho bakalárskej práce. Taktiež bol braný ohľad na odporúčania CSV[8] on the Web a s nimi spojenými referenčnými integračnými testami[15].

### 5.2.1 Funkčné požiadavky

1. Aplikácia bude mať 3 používateľské rozhrania: Webová služba, Webová aplikácia a konzolová aplikácia. Dostupná bude taktiež knižnica s validačnou funkcionalitou.

2. Knižnica musí používateľovi umožniť pracovať s lokálnymi súbormi, ku ktorým užívateľ poskytne cestu.

3. Vzdialené súbory - Knižnica musí používateľovi umožniť prácu so vzdialenými *(remote)* súbormi, ku ktorým používateľ poskytne validnú IRI na základe RFC3987[20]

4. Práca s IRI - Knižnica musí byť schopná lokalizovať súbory s validnými IRI na základe RFC3987[20]

5. Knižnica musí byť schopná pracovať s dialektmi 3.4.9.

6. Knižnica musí byť schopná validovať CSV súbory podľa pravidiel definovaných v RFC4180[1] 1.3 a neskôr rozšírených o RFC7111[2] 1.4.

7. Knižnica musí byť schopná validovať CSV súbory podľa ich schémy na základe odporúčaní CSV on the Web[8].

8. Knižnica musí byť schopná lokalizovať metadata, podľa pravidiel popísaných v sekcii 2.4.

9. Knižnica musí byť schopná generovať výsledok validácie vo formáte CSV, spĺňajúcom štandardy RFC4180[1], popísaných v sekcii 1.3, a tento výsledok poskytnúť vhodným spôsobom používateľovi.

10. Knižnica musí byť schopná generovať výsledok validácie vo formáte RDF v serializácii Turtle, a tento výsledok poskytnúť vhodným spôsobom používateľovi.

11. Knižnica musí byť schopná overiť kompatibilitu metadát a CSV súboru na základe pravidiel definovaných v pod-sekcii 3.4.3.

12. Knižnica musí byť schopná vytvoriť warningy a errory v prípade porušenia validity CSV súboru, prípadne priloženého metadátového súboru, a vhodným spôsobom ich zobraziť užívateľovi.

13. Knižnica musí byť schopná parsovať metadátový dokument, ktorého formát je definovaný v sekcii 3.1.

14. Knižnica musí zvládnuť validovať skupiny tabuliek popísaných pomocou deskriptora skupiny tabuliek definovaného v pod-sekcii 3.4.2.

15. Konzolová aplikácia musí používateľovi umožniť pracovať s lokálnymi súbormi, ku ktorým používateľ poskytne cestu.

16. Webová a konzolová aplikácia bude poskytovať 2 jazykové varianty a to Slovenskú a Anglickú, s možnosťou jednoduchej rozšíriteľnosti na ďalšie jazykové varianty.

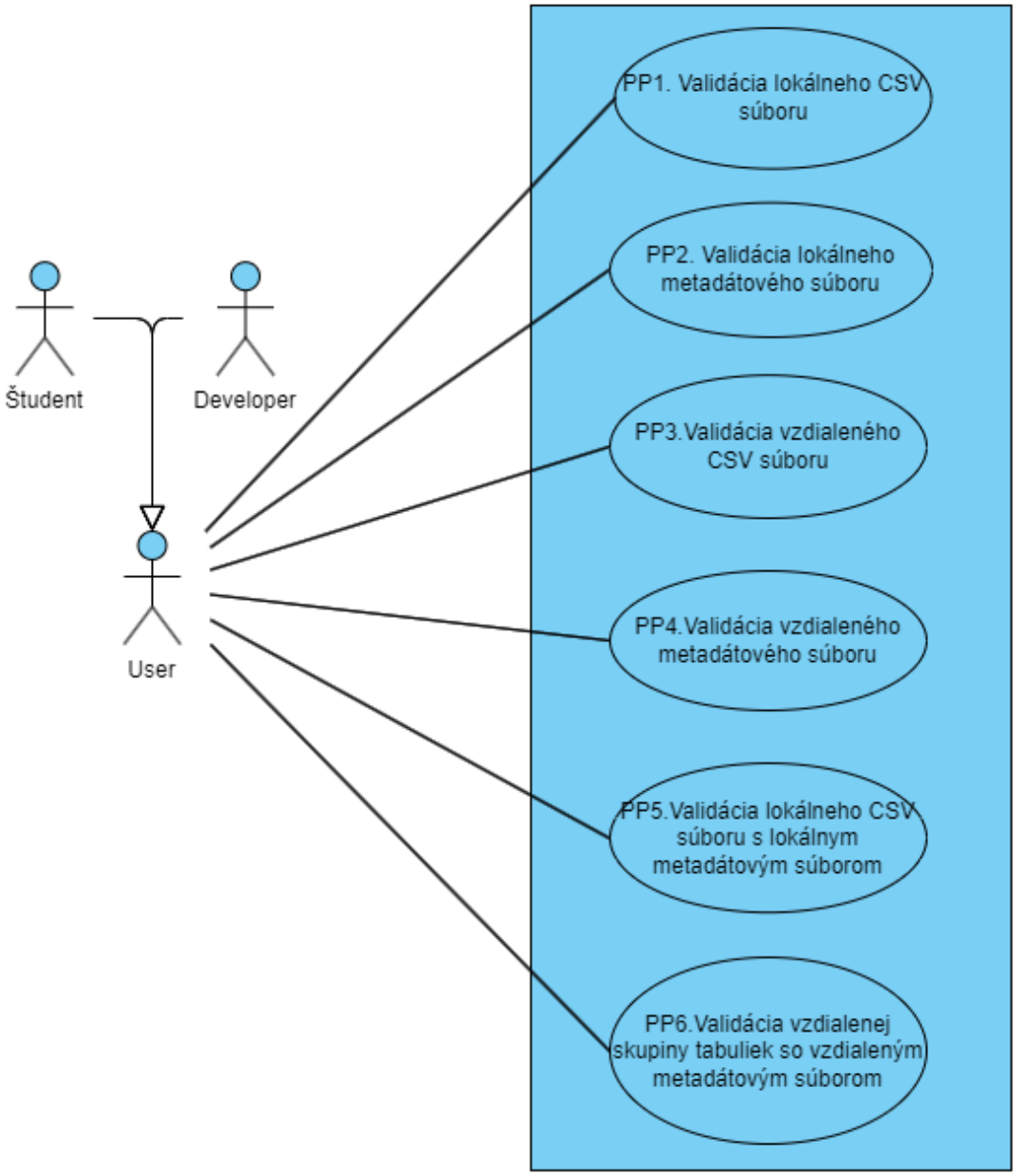
17. Webová aplikácia musí užívateľovi umožniť pracovať s lokálnymi súbormi, ktoré užívateľ nahrá na webový server za pomoci drag and drop poľa.

18. Webová aplikácia musí byť schopná poskytnúť výsledok validácie vo forme reportu na webovej stránke.

### 5.2.2 Nefunkčné požiadavky

1. Aplikácia bude napísaná v programovacom jazyku C#.
2. Architektúra aplikácie by mala umožňovať rozšíriteľnosť o ďalšie validačné pravidlá.
3. Aplikácia by mala byť multiplatformná, za pomoci Dockeru a prehľadnej administrátorskej dokumentácie.
4. Webová aplikácia by mala byť intuitívna a používateľsky prívetivá.
5. Aplikácia by mala poskytovať vysvetlenie varovaní a chýb, ktoré sa vyskytli pri validácií, najlepšie pomocou odkazu, kde sa používateľ môže dozvedieť viac podrobností.

## 5.3 Prípady použitia




Obr. 5.1: Prípady použitia

PP1. **Validácia lokálneho CSV súboru**

1. Používateľ spustí požadovanú variantu validátora.
2. V prípade webovej aplikácie používateľ nahrá lokálny súbor pomocou formulára. V prípade CLI aplikácie používateľ poskytne k takémuto súboru cestu a zvolí formát výsledku validácie pomocou prepínača.
3. Používateľ spustí validáciu.

4. Validátor sa pokúsi lokalizovať metadátový súbor 2.4. Ak je metadátový súbor lokalizovaný, postupujeme rovnako ako v prípade PP2.
5. V prípade webovej aplikácie si validátor stiahne lokálny CSV súbor používateľa na server.
6. Validátor validuje daný CSV súbor.
7. Validátor umožní vygenerovať výsledok validácie v požadovanom formáte. V prípade Webovej aplikácie pomocou tlačidiel. V prípade CLI aplikácie tak používateľ učinil v prvom kroku.
8. V prípade webovej aplikácie sa zobrazí výsledok validácie vo forme reportu priamo na webovej stránke. V prípade CLI aplikácie sa zobrazí výsledok za pomoci vhodného výpisu na príkazovom riadku.

PP2. **Validácia lokálneho metadátového súboru**

1. Používateľ spustí požadovanú variantu validátora.
2. V prípade webovej aplikácie používateľ nahrá lokálny metadátový súbor pomocou formulára. V prípade CLI aplikácie používateľ poskytne k takémuto súboru cestu a zvolí formát výsledku validácie pomocou prepínača.
3. Používateľ spustí validáciu.
4. V prípade webovej aplikácie si validátor stiahne lokálny metadátový súbor používateľa na server.
5. Validátor validuje daný metadátový súbor.
6. Validátor umožní vygenerovať výsledok validácie v požadovanom formáte. V prípade Webovej aplikácie pomocou tlačidiel. V prípade CLI aplikácie tak používateľ učinil v prvom kroku.
7. V prípade webovej aplikácie sa zobrazí výsledok validácie vo forme reportu priamo na webovej stránke. V prípade CLI aplikácie sa zobrazí výsledok za pomoci vhodného výpisu na príkazovom riadku.

PP3. **Validácia vzdialeného CSV súboru**

1. Používateľ spustí požadovanú variantu validátora.
2. V prípade webovej aplikácie používateľ zadá validné IRI CSV súboru pomocou textového poľa. V prípade CLI aplikácie používateľ poskytne k takémuto súboru IRI a zvolí formát výsledku validácie pomocou prepínača.
3. Používateľ spustí validáciu.
4. Validátor sa pokúsi lokalizovať metadátový súbor 2.4. Ak je metadátový súbor lokalizovaný, postupujeme rovnako ako v prípade PP2.
5. Obe varianty validátora stiahnu vzdialený CSV súbor.
6. Validátor validuje daný CSV súbor.
7. Validátor umožní vygenerovať výsledok validácie v požadovanom formáte. V prípade Webovej aplikácie pomocou tlačidiel. V prípade CLI aplikácie tak používateľ učinil v prvom kroku.

8. V prípade webovej aplikácie sa zobrazí výsledok validácie vo forme reportu priamo na webovej stránke. V prípade CLI aplikácie sa zobrazí výsledok za pomoci vhodného výpisu na príkazovom riadku.

PP4. **Validácia vzdialeného metadátového súboru**

1. Používateľ spustí požadovanú variantu validátora.
2. V prípade webovej aplikácie používateľ zadá validné IRI metadátového súboru pomocou textového poľa. V prípade CLI aplikácie používateľ zadá IRI metadátového súboru a zvolí formát výsledku validácie pomocou prepínača.
3. Používateľ spustí validáciu.
4. Obe varianty validátora stiahnu vzdialený metadátový súbor.
5. Validátor validuje daný metadátový súbor.
6. Validátor umožní vygenerovať výsledok validácie v požadovanom formáte. V prípade Webovej aplikácie pomocou tlačidiel. V prípade CLI aplikácie tak používateľ učinil v prvom kroku.
7. V prípade webovej aplikácie sa zobrazí výsledok validácie vo forme reportu priamo na webovej stránke. V prípade CLI aplikácie sa zobrazí výsledok za pomoci vhodného výpisu na príkazovom riadku.

PP5. **Validácia lokálneho CSV súboru s lokálnym metadátovým súborom**

1. Používateľ spustí požadovanú variantu validátora.
2. V prípade webovej aplikácie používateľ nahrá lokálny CSV súbor pomocou drag and drop poľa. V prípade CLI aplikácie používateľ poskytne k takémuto súboru cestu a zvolí formát výsledku validácie pomocou prepínača.
3. V prípade webovej aplikácie používateľ nahrá lokálny metadátový súbor pomocou drag and drop poľa. V prípade CLI aplikácie používateľ poskytne k takémuto IRI a zvolí formát výsledku validácie pomocou prepínača.
4. Požívateľ spustí validáciu.
5. V prípade webovej aplikácie si validátor stiahne lokálny CSV a metadátový súbor používateľa na server.-
6. Validátor validuje daný CSV súbor.
7. Validátor validuje daný metadátový súbor.
8. Validátor validuje daný CSV súbor vzhľadom k priloženému metadátovému formátu.
9. Validátor umožní vygenerovať výsledok validácie v požadovanom formáte. V prípade Webovej aplikácie pomocou tlačidiel. V prípade CLI aplikácie tak používateľ učinil v prvom kroku.
10. V prípade webovej aplikácie sa zobrazí výsledok validácie vo forme reportu priamo na webovej stránke. V prípade CLI aplikácie sa zobrazí výsledok za pomoci vhodného výpisu na príkazovom riadku.

PP6. **Validácia vzdialenej skupiny tabuliek so vzdialeným metadátovým súborom**

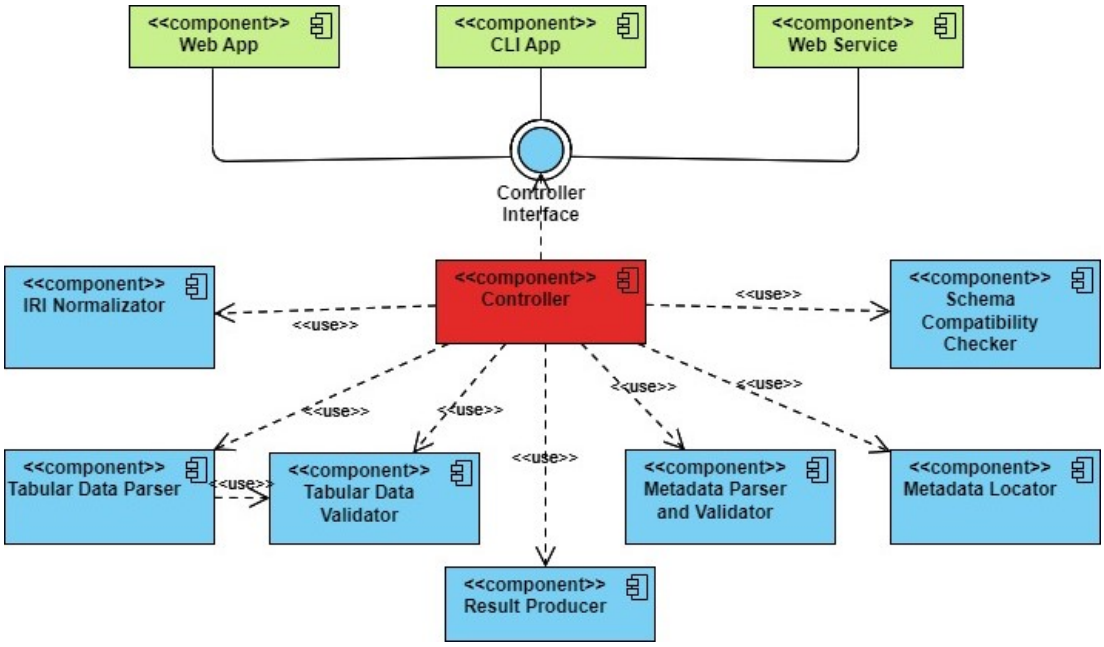
1. Používateľ spustí požadovanú variantu validátora.
2. V prípade webovej aplikácie používateľ zadá IRI deskriptora skupiny tabuliek 3.4.2. V prípade CLI aplikácie používateľ poskytne k takémuto deskriptoru IRI. Taktiež pomocou prepínača zvolí formát výsledku. Tento deskriptor sa už odkazuje na vzdialené CSV súbory.
3. Požívateľ spustí validáciu.
4. Validátor stiahne postupne všetky CSV súbory.
5. Validátor stiahne metadátový súbor.
6. Validátor validuje dané CSV súbory.
7. Validátor validuje daný metadátový súbor.
8. Validátor validuje postupne dané CSV súbory vzhľadom k metadátovému dokumentu.
9. Validátor umožní vygenerovať výsledok validácie v požadovanom formáte. V prípade Webovej aplikácie pomocou tlačidiel. V prípade CLI aplikácie tak používateľ učinil v prvom kroku.
10. V prípade webovej aplikácie sa zobrazí výsledok validácie vo forme reportu priamo na webovej stránke. V prípade CLI aplikácie sa zobrazí výsledok za pomoci vhodného výpisu na príkazovom riadku.

# 6. Design aplikácie

V následujúcej kapitole sa pozrieme na dizajn validátora a jeho jednotlivých častí - knižnica, webová aplikácia, webová služba, CLI aplikácia. Pozrieme sa taktiež aj na návrh jednotlivých používateľských rozhraní.

## 6.1 Architektúra

Knižnica bude pozostávať z viacerých komponentov, ktoré je možné vidieť na obrázku 6.1. Každá komponenta má na starosti 1 kompaktnú úlohu, ktorá je potrebná k zabezpečeniu potrebnej funkcionality, čo značí snahu o dodržanie Single Responsibility Principle (SRP), čo pomáha prehľadnosti kódu a uľahčuje jeho modifikáciu a údržbu.




Obr. 6.1: Komponenty aplikácie

Najdôležitejším komponentom je *Controller*, ktorý riadi chod celej knižnice a navonok poskytuje jednoduché rozhranie, ktoré využívajú komponenty poskytujúce používateľské rozhranie. Využíva návrhový vzor fasáda *(facade)*, ktorá obaľuje zložitú funkcionalitu poskytovanú knižnicou a navonok poskytuje jednoduché rozhranie pre používateľov knižnice. Takisto *Controller* implementuje návrhový vzor mediátor *(mediator)*, kedy riadi ostatné komponenty knižnice a tým znižuje ich coupling.

Komunikácia medzi ostatnými komponentmi je vylúčená, jedinou výnimkou sú komponenty *Tabular Data Parser* a *Tabular Data Validator* z dôvodu zníženia pamäťovej náročnosti a urýchlenia komunikácie, nakoľko parsovanie a validácia tabuľkových dát je hlavným bottle-neckom každej validácie.

Komponent *Tabular Data Parser* slúži na parsovanie tabuľkových dát a validáciu CSV súborov vzhľadom k RFC4180[1] a danému dialektu 3.4.9.

Komponent *Tabular Data Validator* validuje konkrétne hodnoty buniek, vzhľadom k danému metadátovému modelu. Validuje napríklad primárne kľúče, dátové hodnoty buniek a iné.

Dialekt 3.4.9 a metadátový model im poskytne komponent *Controller*, ktorý ich získa od komponentu *Metadata Parser and Validator*.

Komponenty *Tabular Data Validator* a *Tabular Data Parser* spolu musia komunikovať, keďže CSV súbor môže byť veľký a preto anotovaný tabuľkový dátový model2.1 sa nemusí zmestiť do pamäti. Z tohto dôvodu jeho validácia prebieha po jednotlivých záznamoch, ktoré sú predávané obom komponentom.

Validácia tabuľkového súboru na základe anotovaného modelu prebieha za pomoci rozhrania *ITValidationRule*, ktoré predstavuje validačné pravidlá vyplývajúce z metadátového modelu a špecifikácie samotnej. Ak by sme v budúcnosti chceli do-definovať vlastné pravidlá, môžeme jednoducho implementovať takéto rozhranie a predať objekty implementujúce toto rozhranie *Tabular Data Parser-u*, ktorý zavolá potrebné metódy na patričných miestach. To nám umožní naplniť požiadavok na rozšíriteľnosť o ďalšie validačné pravidlá, ktorý vyplýva z analýzy validátora 5.

Komponent *Metadata Parser and Validator* parsuje a validuje samotný metadátový dokument a vytvára metadátový model. Kontroluje správnosť samotného dokumentu - používanie validného JSONu podľa RFC7159[21], používanie správneho dialektu - JSON-LD, kontroluje prítomnosť všetkých povinných vlastností deskriptoru 3.2 a iné.

Validácia metadátového modelu na základe špecifikácií CSV on the Web prebieha za pomoci rozhrania *IMValidationRule*, ktoré predstavuje validačné pravidlá vyplývajúce zo špecifikácie. Ak by sme chceli v budúcnosti pridať ďalšie pravidlá, môžeme jednoducho implementovať takéto rozhranie a predať objekty implementujúce toto rozhranie *Metadata Parser-u*, ktorý zavolá potrebné metódy na patričných miestach. To nám pomôže naplniť požiadavok o rozšírenie validátora o dodatočné validačné pravidlá vyplývajúci z analýzy validátora 5.

Komponent *Metadata Locator* slúži k lokalizácií metadátového súboru v prípade, keď validátor začína len s tabuľkovým súborom. Využíva algoritmus definovaný v sekcii 2.4.

Komponent *IRI Normalizátor* slúži k syntaktickej a schematickej normalizácii `URL` adries podľa RFC3968[22].

Komponent *Schema Compatibility Checker* slúži na kontrolu kompatibility metadátového súboru 3.4.3 a vnorených metadát vy-extrahovaných z tabuľkových dát.

Komponent *Result Producer* vytvára výsledky validácie v podobe, ktorú si zvolil používateľ, a teda buď vo formáte CSV alebo RDF na základe výsledkov jednotlivých parserov a validátorov.

Komponenty *Web App* 6.1.1, *CLI App* a *Web Service* poskytujú používateľovi konkrétne používateľské rozhranie - grafické v prípade Webovej aplikácie, príkazy v prípade CLI aplikácie a API v prípade webovej služby. Komponenty komunikujú s rozhraním *Controller Interface*, ktoré im umožňuje validovať tabuľkové dáta a skupiny tabuľkových dát a tým poskytnúť používateľovi požadovanú funkcionalitu.

Takto navrhnuté komponenty nám umožňujú plne splniť všetky funkčné aj nefunkčné požiadavky definované v analýze validátora 5.

**Architektonický štýl**

Aplikácia momentálne nepodlieha žiadnemu známemu architektonickému štýlu. Máme komponent *Controller*, ktorý slúži ako fasáda k validačným funkcionalitám a následne 3 komponenty *Web App*, *CLI App* a *Web Service*, ktoré s týmto komponentom komunikujú za pomoci rozhrania *Controller Interface* a poskytujú používateľom jednoduché a intuitívne používateľské rozhranie. Jednotlivé komponenty vrámci knižnice boli navrhnuté s cieľom zabezpečiť tzv. Single Responsibility Principle, kedy sa snažíme o to, aby komponenta robila jednu, presne definovanú vec a robila ju správne. Implementácia návrhového vzoru mediátor komponentom *Controller* umožňuje dostatočnú izoláciu ostatných komponentov knižnice a zníženie komunikácie a závislosti medzi nimi.

Zvažované boli taktiež architektonické štýly:

- **Vrstevnatá architektúra** *(Layered architecture)* - kde by sme aplikáciu rozdelili na tri vrstvy: Prezentačnú vrstvu *(Presentation layer)*, Biznis vrstvu *(Business layer)* a Perzistenčnú vrstvu *(Persistance layer)*.

  Takýto štýl čiastočne aplikácia implementuje, nakoľko komponenty *Web App*, *Web Service* a *CLI App* môžeme považovať za vrstvu prezentačnú, komponenty tvoriace knižnicu za biznis vrstvu. Perzistenčná vrstva v aplikácii ostáva v pozadí, nakoľko ukladanie dát je potrebné len pri komponentoch *Web App* a *Web Service*, kde sa jedná len o jednoduché, dočasné uloženie výsledku validácie.

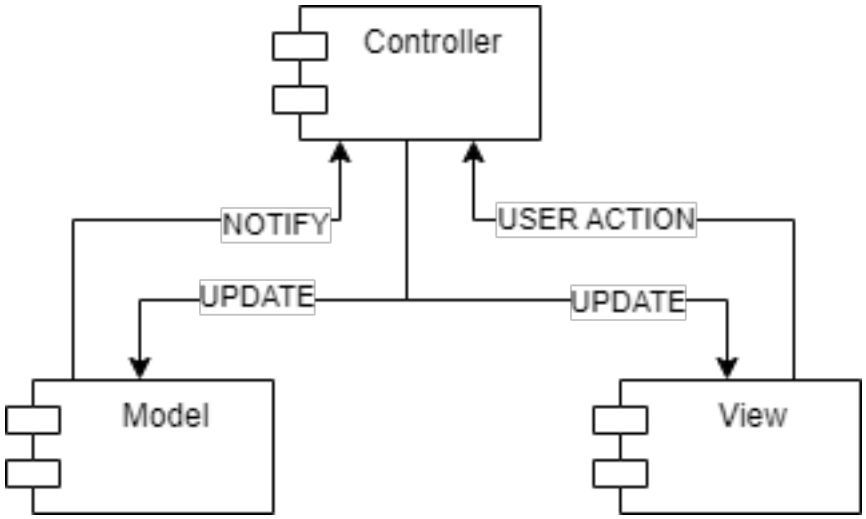
- **Distribuované architektonické štýly** - prinášajú so sebou značnú zložitosť na implementáciu. Taktiež doména problému nie je prirodzená pre členenie na tzv. služby, keďže hlavným požiadavkom na systém je validovanie tabuľkových súborov, prípadne s priloženým metadátovým súborom. S takýmito službami sa môžeme stretnúť u populárnych architektonických štýloch *Microservices Architecture* prípadne *Service Oriented Architecture*.

  Ak by sme potrebovali v budúcnosti zvýšiť napríklad výkon validátora, bolo by tak možné urobiť zavedením

### 6.1.1 Komponent Web App

Webová aplikácia je navrhnutá na dobre známom a jednoduchom návrhovom vzore pre webové aplikácie s názvom **Model-View-Controller(MVC)**. Rozhodli sme sa použiť takýto monolitický návrhový vzor pre jeho jednoduchosť a známosť, čo ho robí rýchlym na implementáciu a taktiež jednoduchým pre porozumenie.

**Model-View-Controller (MVC)**




Obr. 6.2: Komponenty návrhového vzoru Model-View-Controller (MVC

Tento návrhový vzor je založený na 3 hlavných komponentoch:

- **Controller** - komunikuje s modelmi a views. Je zodpovedný za obslúženie požiadavkov používateľa prichádzajúcich z views. Napríklad, ak príde požiadavka na zobrazenie výsledku validácie, obslúži ho controller s názvom `ResultController`. Ten nájde v databáze požadovaný výsledok - Model a poskytne ho správnemu View, ktoré ho zobrazí samotnému používateľovi.

  To, akú požiadavku má Controller obslúžiť sa definuje vo fáze smerovania, kedy `Router` na základe cesty požiadavku rozhodne a zavolá patričný `Controller`.

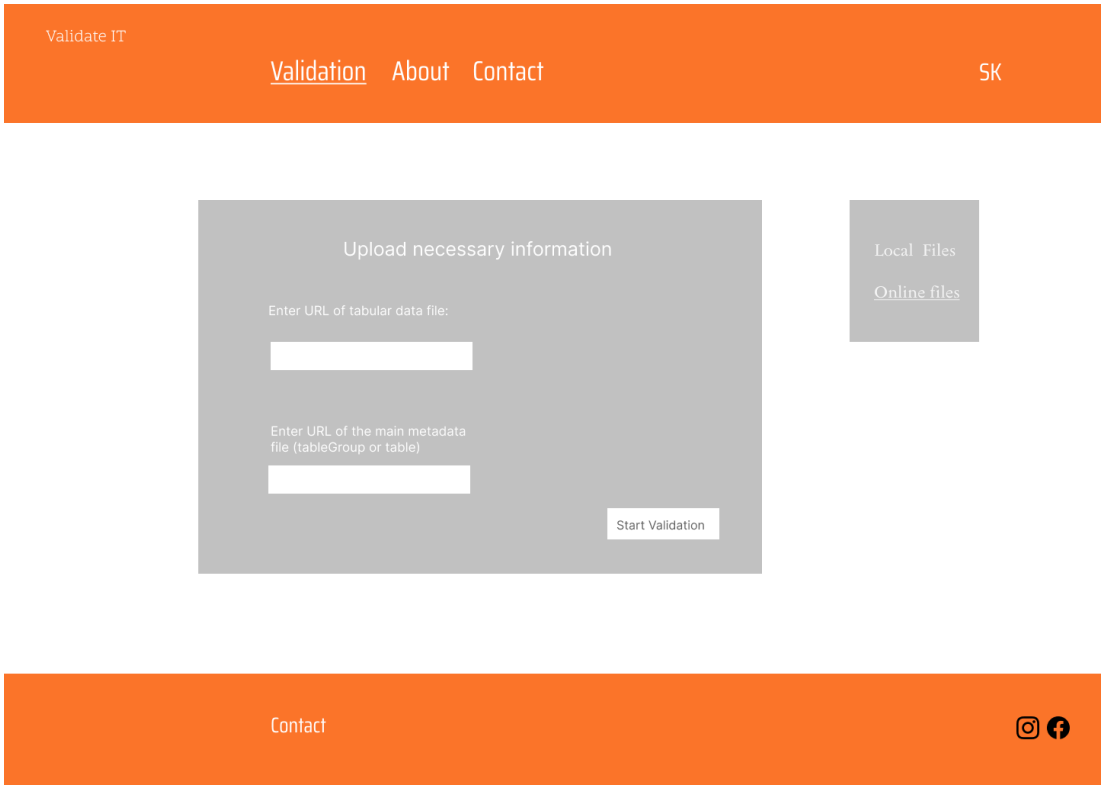
- **Model** - sprostredkováva komunikáciu s databázou a poskytuje vhodné rozhranie pre konkrétny `Controller`. Vhodne poskytuje dáta potrebné na obslúženie konkrétnych požiadavkov používateľa. Patria tu hlavne triedy spojené so zobrazovaním výsledkov ako napríklad `ResultModel`.

- **Views** - predstavuje najjednoduchší komponent. Poskytuje grafické rozhranie používateľovi a zobrazuje vhodným spôsobom `Model`, ktorý bol tomuto `View` predaný `Controllerom`.

Tento návrhový vzor nás najviac približuje k architektonickému štýlu: vrstevnatá architektúra *(layered architecture)*, kde si môžeme predstaviť mapovanie Controllerov do txv. *Biznis vrstvy*, Modelov do tzv. *Dátovej vrstvy* a Views do tzv. *Prezentačnej vrstvy*.

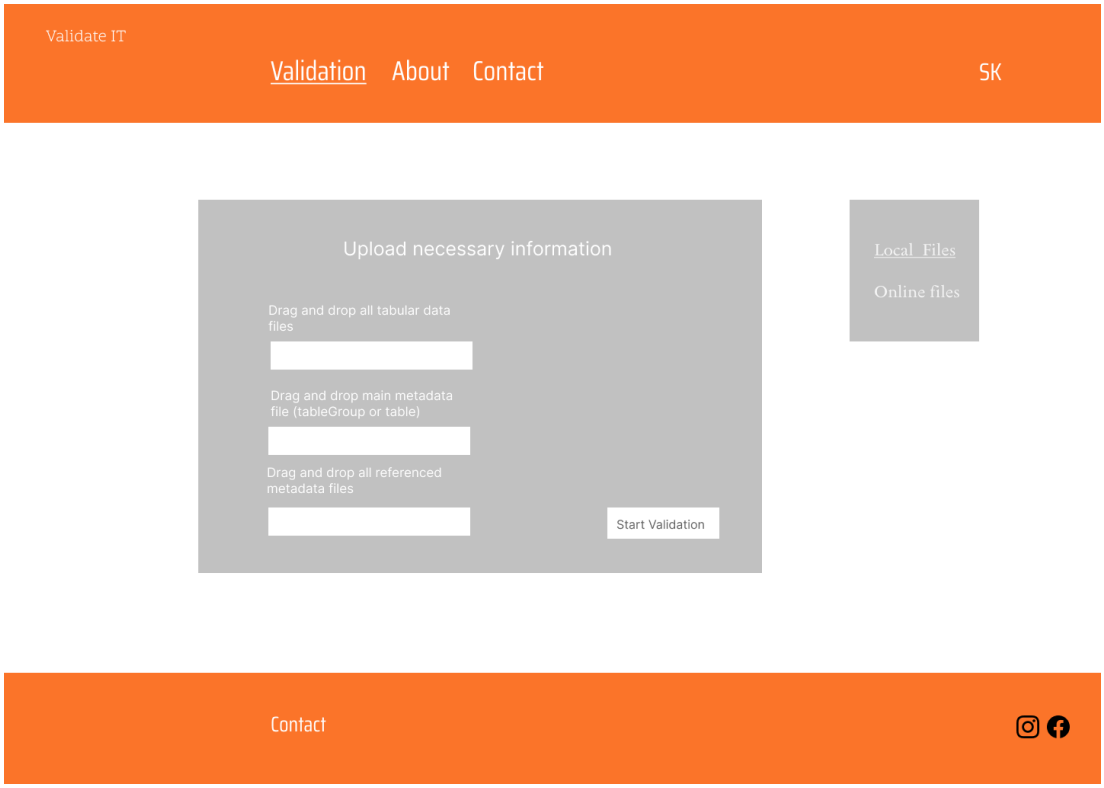
Zvažovali sme aj nad použitím zložitejších architektonických štýlov, akými sú napríklad **Microservices**, no zložitosť takýchto architektúr by príliš spomalila jeho vývoj, nakoľko projekt bol vyvíjaný v jednočlennom tíme a benefity v podobe vyššej škálovateľnosti a výkonu by boli takmer nevyužiteľné, nakoľko samotná webová aplikácia slúži primárne na poskytnutie jednoduchého používateľského rozhrania a samotná biznis logika je spracovaná v knižnici.

## 6.2 Návrh UI pre webovú aplikáciu



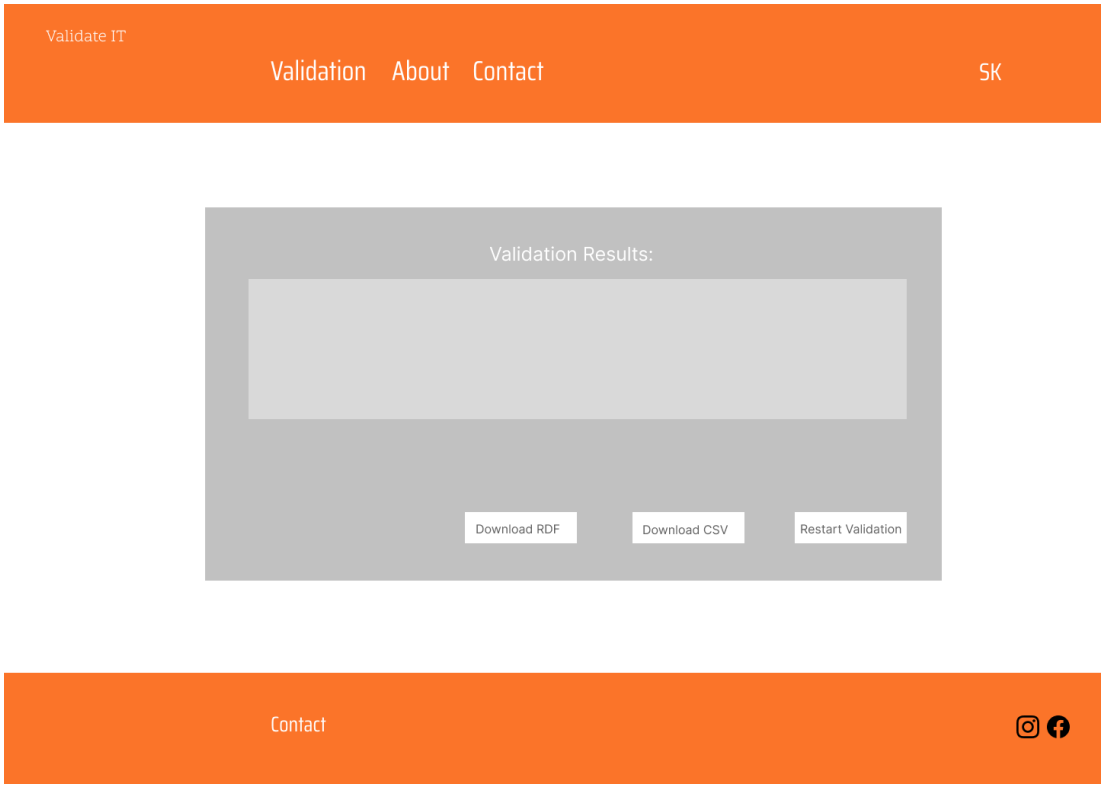

Obr. 6.3: Návrh grafického prostredia pre webovú aplikáciu: validácia online súborov

Na obrázku 6.3 je zobrazený návrh grafického rozhrania pre validáciu online súborov webovou aplikáciou. Používateľ môže zadať URL online súboru, na základe ktorého validátor lokalizuje metadátový súbor a prípadne aj ďalšie referencované metadátové a tabuľkové súbory. Používateľ môže taktiež zadať iba `URL` adresu vzdialeného metadátového súboru, na základe ktorého validátor lokalizuje potrebné tabuľkové súbory a referencované metadátové súbory.




Obr. 6.4: Návrh grafického prostredia pre webovú aplikáciu: validácia lokálnych súborov

Na obrázku 6.4 je zobrazený návrh grafického rozhrania pre validáciu lokálnych súborov webovou aplikáciou. Používateľ môže pomocou drag and drop poľa zvoliť, ktoré lokálne súbory chce validovať a taktiež nahrať hlavný metadátový súbor, ktorým je buď súbor popisujúci skupinu tabuliek alebo tabuľku samotnú. Používateľ musí v prípade referencovaných metadátových súborov v hlavnom metadátovom súbore nahrať takéto súbory pomocou 3. drag and drop poľa.




Obr. 6.5: Návrh grafického prostredia pre webovú aplikáciu: zobrazenie výsledkov validácie

Na obrázku 6.5 je zobrazený návrh grafického rozhrania pre zobrazenie výsledkov validácie v podobe reportu na webovej stránke. Používateľ je schopný spustiť celý validačný proces odznova pomocou tlačidla. Webová aplikácia umožňuje používateľovi taktiež stiahnuť výsledky validácie vo formáte CSV alebo RDF za pomoci tlačidiel.

## 6.3 Návrh CLI rozhrania

V tejto sekcii sa pozrieme na návrh rozhrania pre CLI aplikáciu a rozličné argumenty, ktoré pri jej spustení môže používateľ zvoliť.

### 6.3.1 Argumenty CLI aplikácie

CLI aplikácia bude prijímať následujúce argumenty:

- **--tabularIRI** - viacnásobný argument, pomocou ktorého používateľ predá IRI tabuľkového súboru resp. súborov, ktoré má validátor validovať. Napríklad:

```
--tabularIRI /opt/data/my.csv
```

- **--metadataIRI** - argument, pomocou ktorého používateľ predá IRI metadátového súboru. Napríklad:

```
--metadataIRI /opt/data/my_schema.json
```

- **--language** - argument, pomocou ktorého používateľ zvolí jazyk chybových hlášok vo výsledku validácie. Podporované budú dva jazyky: slovenský(sk-SK) a anglický(en-GB) Napríklad:

  ```
  --language sk-SK
  ```

- **--csvPath** - argument, pomocou ktorého používateľ predá cestu, kde má byť vygenerovaný výsledok validácie vo formáte CSV. Napríklad:

  ```
  --csvPath /opt/data/result/result.csv
  ```

- **--rdfPath** - argument, pomocou ktorého používateľ predá cestu, kde má byť vygenerovaný výsledok validácie vo formáte RDF. Napríklad:

  ```
  --csvPath /opt/data/result/result.ttl
  ```

- **--verbose** - argument, pomocou ktorého používateľ zapne detailnejšie chybové hlásenia vo výpise výsledku validácie na príkazovom riadku.

## 6.4 Návrh rozhrania Webovej služby

Webová služba bude poskytovať používateľovi 2 endpointy:

1. **api/Validate** - na tomto endpointe očakáva služba JSON objekt poslaný za pomoci metódy *POST*. Objekt bude obsahovať následujúce vlastnosti:
   (a) *metadataUrl* - textová vlastnosť, ktorá obsahuje `URL` metadátového dokumentu.
   (b) *tabularUrl* - textová resp. array vlastnosť, ktorá obsahuje `URL` adresy tabuľkových súborov.

   ```
   {
   "metadataURL": "https://w3c.github.io/csvw/tests/test077-metadata.json",
   "tabularURL": "https://w3c.github.io/csvw/tests/tree-ops.csv"
   }
   ```

   Po úspešnom ukončení validácie príde používateľovi odpoveď obsahujúca v tele `resultID` validácie výsledku.

2. **api/GetResult/{resultID}** - na tomto endpointe očakáva služba *GET* požiadavku so správnym *resultID*. V odpovedi na túto požiadavku dostane používateľ JSON objekt popisujúci výsledok validácie, ktorý je potrebný pre vytvorenie reportu na webovej stránke. Takáto odpoveď bude predovšetkým obsahovať výsledok validácie a prípadne všetky validačné chyby a varovania.

Špecifikáciu `openApi` vo formáte yaml môžeme nájsť v priloženom GitLabovom repozitári v priečinku `Docs/WebService`. Dostupná je aj na adrese: `https://drexem.github.io/validateIT/swagger/index.html`.

# 7. Implementácia

Kapitola priblíži konkrétne implementačné detaily knižnice, CLI aplikácie, webovej služby a webovej aplikácie.

V úvode sú uvedené technológie, použité k vytvoreniu validátora a všetkých jeho častí.

Následne je priblížené členenie kódu do jednotlivých menných priestorov *(namespaces)* či najdôležitejšie triedy a rozhrania validátora.

## 7.1 Použité technológie

V tejto sekcii si popíšeme jednotlivé technológie a knižnice, ktoré boli použité k vytvoreniu validátora, či už samotnej knižnice, alebo konkrétnych aplikácii.

**Definícia 7.1.1. NuGet balíček**[23] *(NuGet package)* je ZIP súbor s koncovkou `.nupkg`, ktorý obsahuje skompilované súbory so strojovým kódom (DLL), súbory súvisiace s týmito súbormi, a taktiež popisujúci manifest, ktorý zahŕňa informácie o samotnom balíčku napr. číslo verzie balíčku.

Pre verzovanie aplikácie sme taktiež využili GitLab repozitár dostupný na adrese: `https://gitlab.mff.cuni.cz/kolcunm/csv-validator`. Obsah tohto repozitára môžeme nájsť taktiež v prílohe.

Pre vytvorenie dokumentácie za pomoci GitHub Pages sme použili GitHub repozitár dostupný na adrese: `https://github.com/drexem/validateIT`. V prílohe môžeme nájsť najdôležitejšie dokumentačné `markdown(.md)` súbory.

### 7.1.1 Knižnica

Knižnica využíva následujúce NuGet balíčky 7.1.1:

- **CsvHelper**[24] - knižnica umožňujúca prácu s CSV súbormi. Využívame ju na vytváranie výsledkov vo formáte CSV podliehajúcom RFC7111[2]. Knižnica poskytuje jednoduché rozhranie k zápisu do súborov CSV. Podotkneme však, že knižnica nepodporuje rozličné dialekty 3.4.9, a preto ju nebolo možné použiť pre parsovanie konkrétnych tabuľkových súborov.

- **dotNetRdf**[25] - knižnica umožňujúca prácu s RDF súbormi. Využívame ju k vytváraniu výsledkov validácie vo formáte RDF v serializácii Turtle. Knižnica poskytuje jednoduché rozhranie pre vytváranie a zapisovanie do RDF súborov.

- **ExtendedNumerics.BigDecimal**[26] - knižnica umožňujúca prácu s veľkými desatinnými číslami. Využívame ju k parsovaniu rozličných formátov čísel, ktoré sa môžu vyskytnúť v tabuľkovom súbore. Takéto čísla rozličných numerických dátových typov naparsujeme do spomínanej triedy a tým zjednotíme ďalšiu prácu s nimi.

- **Newtonsoft.Json**[27] - knižnica umožňujúca prácu s JSON súbormi. Využívame ju na parsovanie a prácu s metadátovým súborom vo formáte

JSON. Knižnica umožňuje jednoduchý prístup k JSON objektom, ich vlastnostiam, dátovým typom a mnoho iného. Patrí taktiež medzi najpoužívanejšie knižnice vrámci celého dotnetu s viac ako 1 miliardou stiahnutí.

- **Tavis.UriTemplates**[28] - knižnica umožňujúca prácu s URI šablónami. Využívame ju k expanzii URI šablón, na ktoré môžeme naraziť pri lokalizácii metadát 2.4 alebo aj v samotnom metadátovom súbore v podobe URI šablónových vlastnosti 3.3.3.
- **UTF.Unknown**[29] - knižnica umožňujúca detekciu kódovania textového súboru. Využívame ju k detekcii kódovania ako metadátových, tak aj tabuľkových súborov.

### 7.1.2 CLI Aplikácia

CLI aplikácia využíva následujúce NuGet balíčky 7.1.1:

- **CommandLineParser**[30] - knižnica umožňujúca jednoduché spracovanie argumentov z príkazovej riadky. Využívame ju k jednoduchšiemu vyhodnoteniu argumentov CLI aplikácie, na základe ktorých je spustená validácia.

### 7.1.3 Webová služba a Webová aplikácia

Webová služba a Webová aplikácia obe využívajú následujúce NuGet balíčky 7.1.1:

- **Microsoft.EntityFramework balíčky** - balíčky poskytujúce objektovorelačné mapovanie medzi C# objektami a databázou. Využívame ju prevažne na dočasné ukladanie výsledkov validácie do databázy a pre jednoduchšiu prácu so samotnou databázou, vďaka čomu nemusíme nikdy písať žiadne SQL dotazy.
- **Microsoft.NET.Build.Containers**[31] - balíček, ktorý nám umožňuje kontajnerizovaný vývoj aplikácie a uľahčuje deployment za pomoci nástroja Docker.

Webová služba využíva navyše NuGet balíčky 7.1.1:

- **Microsoft.AspNetCore.Cors**[32] - balíček umožňujúci komunikáciu webovej aplikácie s webovou službou. Umožňuje definovať, z akých adries server dovolí pristupovať k webovej službe.

Webová aplikácia využíva navyše NuGet balíčky 7.1.1:

- **Microsoft.AspNetCore.Mvc.Localization**[33] - balíček umožňujúci lokalizáciu webovej aplikácie. Umožňuje nám jednoduchú lokalizáciu používateľského rozhrania webovej aplikácie do anglického a slovenského jazyka.

### 7.1.4 Obecné

Validátor využíva následujúce technológie:

- **xunit**[34] - NuGet balíček 7.1.1, ktorý poskytuje rozhranie pre vytváranie unit a integračných testov. Použitý bol pre vytvorenie všetkých unit testov a taktiež pre vytvorenie integračných testov[15].
- **.NET 7.0** - behové prostredie pre .NET aplikácie. Validátor bol vyvíjaný v programovacom jazyku C#, ktorého behové prostredie je práve .NET. Verzia 7.0 je súčasným štandardom pre vývoj .NET aplikácii.
- **Docker** - umožňuje vytvárať a manažovať virtualizované aplikačné kontajnery na ľubovoľnom operačnom systéme. Využívame ho pre jednoduché nasadenie CLI aplikácie, webovej služby a webovej aplikácie.
- **coverlet.collector**[35] - NuGet balíček 7.1.1 umožňujúci analyzovať pokrytie kódu unit resp. integračnými testami. Využili sme ho k analýze pokrytia unit a integračných testov validátora.

## 7.2 Štruktúra projektu

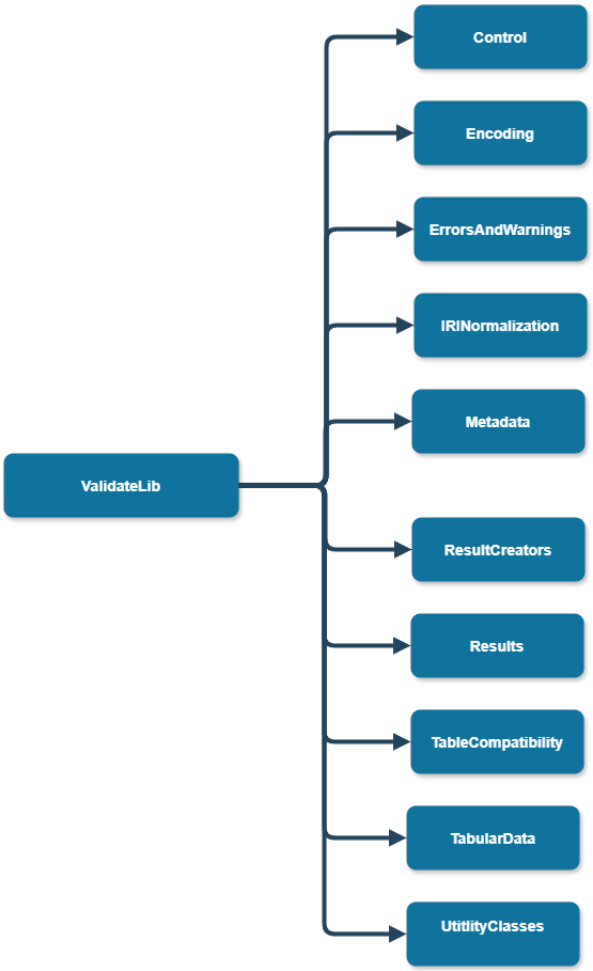
V tejto sekcii si popíšeme jednotlivé menné priestory validátora a jeho častí - knižnica, CLI aplikácia, webová aplikácia a webová služba. Pozrieme sa na zaujímavé detaily z implementácie a taktiež na prípadné odchýlenia od návrhu validátora v kapitole 6.

**Definícia 7.2.1. Menný priestor** *(namespace)* v C# spája triedy, štruktúry, rozhrania ... do logických skupín. Pomáha vyhýbať sa menným konfliktom a umožňuje organizovať a kategorizovať spolu súvisiaci kód.

**Definícia 7.2.2. C# projekt** je štruktúrovaná kolekcia súborov a zdrojov, ktoré sú spolu organizované pre vývoj aplikácie, knižnice alebo iného spustiteľného artefaktu za pomoci programovacieho jazyka C#.

### 7.2.1 Knižnica

Knižnica sa nachádza v projekte 7.2.2 **ValidateLib.csproj** a jej hlavným menným priestorom 7.2.1 je **ValidateLib**.




Obr. 7.1: Menný priestor ValidateLib

Menný priestor **ValidateLib** obsahuje následujúce vnorené menné priestory:

- **ValidateLib.Control** - hlavným bodom knižnice je jej vstupný bod a rozhranie, s ktorým môže koncový používateľ komunikovať. Rozhranie sa nazýva **IController** a nachádza sa práve vo vnorenom mennom priestore Control. Jedinou implementáciou tohto rozhrania je trieda **Controller**, ktorá riadi chod celého validátora a komunikáciu ostatných komponentov.
- **ValidateLib.Encoding** - nachádzajú sa tu triedy, ktoré poskytujú funkcionalitu spojenú s rozličnými kódovaniami súborov, napríklad detegujú samotné kódovanie či prítomnosť BOM bytov.
- **ValidateLib.ErrorsAndWarnings** - obsahuje triedy, ktoré predstavujú chyby a varovania, vznikajúce v procese validovania metadátových či tabuľkových súborov. Ak chceme vytvoriť nové varovanie resp. chybu, zavoláme patričnú metódu na triede **WarningFactory** resp. **ErrorFactory** a tá nám takýto objekt vytvorí. Triedy obsahujú detailný popis varovania resp. chyby a odkazy na stránky s užitočnými informáciami.
- **IRINormalizaton** - obsahuje triedu, ktorá sa stará o normalizovanie IRI 2.3.3 pre účely validácie.

- **ValidateLib.Metadata** - obsahuje triedy, ktoré poskytujú funkcionalitu pri spracovávaní metadátového dokumentu. Obsahuje napríklad jednotlivé deskriptory zo slovníka metadát 3, ich vlastnosti, triedu zodpovednú za lokalizáciu metadát 2.4 či extrakciu vnorených metadát 4. Obsahuje taktiež validátory a parsery jednotlivých deskriptorov 3.2 a hlavnú triedu **MetadataParserValidator**, ktorá riadi chod parsovania a validácie metadátového dokumentu a práve s touto triedou komunikuje trieda `Controller`.

- **ValidateLib.ResultCreators** - obsahuje rozhrania a triedy, ktoré poskytujú možnosť vytvárať súbory s výsledkami validácie v rozličných formátoch. Aktuálne podporované formáty sú CSV a RDF.

- **ValidateLib.Results** - obsahuje triedy, rozhrania, enumerácie, ktoré poskytujú informácie o výsledkoch validácie. Poskytujú informácie o type výsledku (validný/chybný/varovanie) a štatistikách o procese validovania akými sú napríklad počet spracovaných riadkov, stĺpcov a buniek.

- **ValidateLib.TableCompatibility** - obsahuje triedy a rozhrania, ktoré kontrolujú kompatibilitu 3.4.3 tabuľkových deskriptorov 3.4.3 a vnorených metadát 4 referencovaných tabuľkových súborov.

- **ValidateLib.TabularData** - menný priestor, ktorý spolu s **ValidateLib.Metadata** tvorí najdôležitejší menný priestor. Obsahuje triedy, rozhrania, ..., ktoré riadia parsovanie a validovanie jednotlivých tabuľkových súborov.

  Medzi najdôležitejšie triedy patrí `TabularDataTableGroupValidator`, ktorá riadi validáciu a parsovanie samotných tabuľkových súborov. Táto trieda využíva triedu `TabularDataTableValidator` na validáciu a parsovanie jednotlivých tabuliek.

  Trieda `TabularDataTableValidator` komunikuje s triedami z menného priestoru `ValidateLib.TabularData.Parsing`, ktoré majú na starosti parsovanie CSV súboru, a tým pádom medzi týmito dvomi komponentami je silný coupling. Tento coupling vznikol z dôvodu úspory pamäte, keďže sa snažíme držať v pamäti vždy práve jeden riadok práve spracovávaného tabuľkového súboru. Viac o problematike validácie tabuľkových súborov v pod-sekcii 7.2.1. Výnimkou sú prípady, kedy tabuľka má definované primárne, príp. cudzie kľúče, no tejto problematike sa venujeme neskôr v podsekcii 7.2.1.

- **ValidateLib.UtilityClasses** - obsahuje utility triedy pre rozličné potreby ako sú napríklad podpora pre rozličné typy vlastností deskriptorov a operácii s nimi spojených, pomocné triedy pre prácu so súbormi, IRI, dátovými typmi a podobne.

### Príspevok do špecifikácie

Pri vytváraní samotnej knižnice sa vyskytovali situácie, kedy samotná špecifikácia tabuľkového modelu[7] alebo metadátového slovníka[13] neboli dostatočne presné a obsiahle pre vyjasnenie zložitejších funkcionalít validátora.

Preto sme sa obrátili so žiadosťami o upresnenie na samotného autora Gregga Kelloga za a vytvorili sme issues na Githube samotnej špecifikácie.

Celkovo sme vytvorili 2 issues:

1. Prvá issue bola vytvorená kvôli nejasnostiam ohľadom povolených formátov pre numerické dátové typy. S pánom Greggom Kellogom sme riešili sémantické detaily jednotlivých znakov v zápise numerického formátu.

2. Druhá issue sa týkala nejasností integračných testov. Diskutovali sme o presnom dôvode, prečo by dané testy mali dosahovať požadované výsledky a taktiež, či sa nejedná a spor v súlade so špecifikáciou.

### Validácia tabuliek

Validátor musí validovať tabuľky na základe anotovaného modelu 2.1 tabuliek, ktorý vytvoril na základe metadátového súboru prípadne extrakciou vnorených metadát 4.

`ValidateLib.TabularData.Validation` zabezpečuje vyššie spomínanú funkcionalitu, ktorý definuje triedu `TabularDataTableValidator`. Táto trieda využíva menný priestor `ValidateLib.TabularData.Parsing`, ktorý obsahuje triedy parsujúce tabuľkový súbor.

Pre validáciu využíva trieda `TabularDataTableValidator` validačné pravidlá menného priestoru `ValidateLib.TabularData.Validation.ValidationRules`. Delíme ich na dva základné typy:

1. **ICellValidationRule** - tu patria validačné pravidlá, kedy k samotnému validovaniu nám stačí držať anotovanú bunku 2.1.5. Medzi takéto pravidlá patrí napríklad pravidlo `CellDatatypeValidationRule`, ktoré kontroluje, či hodnota bunky podlieha dátovému typu 2.1.5 stĺpca 2.1.2, v ktorom sa bunka 2.1.5 nachádza.

2. **IRowValidationRule** - tu patria validačné pravidlá, ktoré k samotnému validovaniu potrebujú držať anotovaný riadok 2.1.4. Medzi takéto pravidlá patrí napríklad pravidlo `NumberOfColumnsRowValidationRule`, ktoré kontroluje, či počet stĺpcov 2.1.2 v riadku sa zhoduje s počtom stĺpcov 2.1.2 anotovanej tabuľky 2.1.2.

Jedinou výnimkou je pravidlo `FKValidationRulesd` 7.2.1, ktoré kontroluje pravidlá týkajúce sa cudzích kľúčov.

### Primárne kľúče

Validátor musí vedieť validovať aj tabuľky, ktoré obsahujú primárne kľúče. Ak obsahuje tabuľkový deskriptor 3.4.3 deskriptor stĺpca 3.4.6, prípadne ich kombináciu, na ktoré sa odkazuje vlastnosť *primaryKey* deskriptora schémy 3.4.4 danej tabuľky, musia byť všetky hodnoty daného stĺpca, prípadne kombinácie daných stĺpcov, unikátne v rámci celej tabuľky 3.4.3. Takáto vlastnosť sa však nedá kontrolovať za prítomnosti jedného riadku v pamäti. Preto sme vytvorili validačné pravidlo `PrimaryKeyRowValidationRule`, ktoré si pamätá primárne kľúče už spracovaných riadkov a kontroluje, či primárny kľúč aktuálne spracovávaného riadka sa už nevyskytol v tabuľke. V prípade, že výskyt je opakovaný, vytvorí

toto validačné pravidlo chybu. Vďaka tomu si validátor však ukladá do pamäte potenciálne celý stĺpec tabuľky, a preto jmusíme pracovať s predpokladom, že sa takýto stĺpec vojde do pamäte. Pre väčšinu typických prípadov nie je použitie takéhoto obmedzenia problémom.

Bolo uvažované aj nad možnosťou postavenia indexu nad takýmito stĺpcami v sekundárnej pamäti, prípadne stromového indexu, avšak realizácia by bola časovo náročná, a tak sme sa rozhodli zamerať na dôležitejšie aspekty validátora. Zároveň s danou architektúrou je jednoduché vymeniť takéto validačné pravidlo za iné, ktoré bude pamäťovo efektívnejšie.

#### Cudzie kľúče

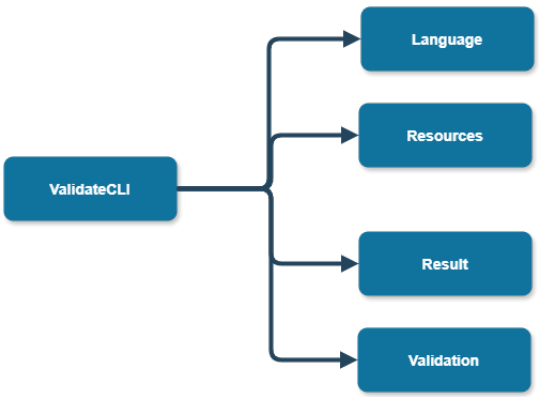
Validátor musí byť schopný validovať aj skupiny tabuliek 3.4.2, obsahujúce cudzie kľúče 3.4.5. Ak tabuľka 2.1.2 obsahuje cudzí kľúč 3.4.5, musíme skontrolovať pre všetky záznamy tabuľky, že takýto cudzí kľúč existuje v referencovanej tabuľke 2.1.2, a zároveň, že je unikátny vrámci danej tabuľky 2.1.2.

Avšak validovanie takejto podmienky nie je možné za obmedzenia, že v pamäti máme uložený len 1 riadok práve spracovávanej tabuľky. Validátor si preto zapamätá stĺpce, ktoré sú súčasťou cudzieho kľúča, prípadne sú ním referencované a následne validuje postupne všetky cudzie kľúče. Validátor postupne prechádza hodnoty cudzieho kľúča a kontroluje, či sa vyskytujú v referencovanej tabuľke, a zároveň, či sú v takejto tabuľke unikátne.

Opäť bolo zvážené aj postavenie indexov v sekundárnej pamäti nad takýmito stĺpcami. Takáto optimalizácia je možná, no časovo náročná. Vzhľadom k faktu, že by optimalizácia mala zanedbateľný vplyv na drvivú väčšinu prípadov, pre ktoré bol validátor navrhnutý - CSV súbory na webe, rozhodli sme sa ju neimplementovať.

### 7.2.2 CLI aplikácia

CLI aplikácia sa nachádza v projekte 7.2.2 **ValidateCLI.csproj** a jej hlavným menným priestorom 7.2.1 je **ValidateCLI**.



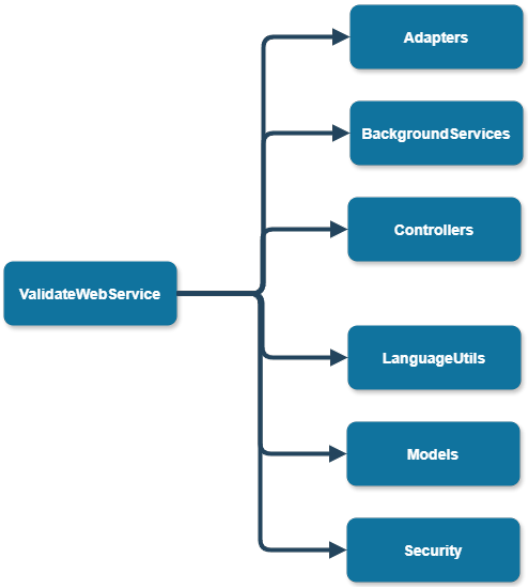

Obr. 7.2: Menný priestor ValidateCLI

Menný priestor **ValidateCLI** obsahuje následujúce vnorené menné priestory:

- **ValidateCLI.Language** - obsahuje triedy umožňujúce lokalizáciu CLI aplikácie do viacerých jazykov. Momentálne sú podporované dva jazykové varianty - slovenský a anglický.
- **ValidateCLI.Resources** - obsahuje súbory s lokalizovaným textom a taktiež triedu `LocalizationManager`, ktorá umožňuje prístup k takýmto textom.
- **ValidateCLI.Result** - obsahuje triedy poskytujúce funkcionality vytvárania výsledkov validácie ako sú napríklad zápis výsledkov do súborov v rozličných formátoch, či generovanie samotného textu výsledkov.
- **ValidateCLI.Validation** - obsahuje triedy zabezpečujúce spustenie validácie tabuľkových a metadátových súborov ale aj argumentov, ktoré predá na príkazovom riadku používateľ.

### 7.2.3 Webová služba

Webová služba sa nachádza v projekte 7.2.2 **WebService.csproj** a jej hlavným menným priestorom 7.2.1 je **ValidateWebService**.




Obr. 7.3: Menný priestor ValidateWebService

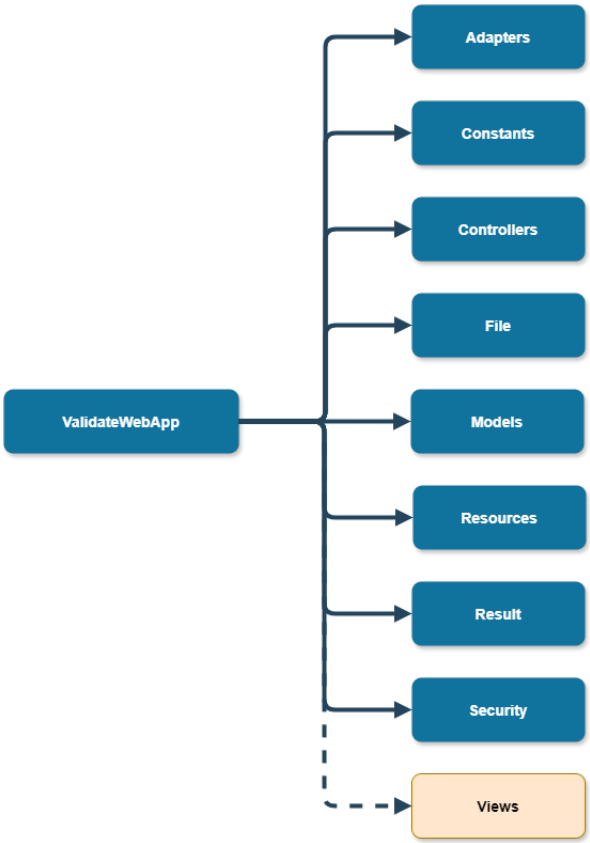
Menný priestor **ValidateWebService** obsahuje následujúce vnorené menné priestory:

- **ValidateWebService.Adapters** - obsahuje triedy, ktoré sa chovajú ako adaptéry. Tieto triedy adaptujú rozhrania iných modulov a tried tak, aby boli jednoduchšie použiteľné pre potreby webovej služby. Patrí tu napríklad trieda `ValitationResultAdapter`, ktorá transformuje validačný výsledok generovaný knižnicou do podoby, v ktorej sa jednoduchšie serializuje pre uloženie do databázy.

- **ValidateWebService.BackgroundServices** - patria tu triedy, ktoré reprezentujú služby bežiace v pozadí webovej služby. Jedinou aktuálne implementovanou background službou je `CleaningService`, ktorá periodicky maže staré výsledky validácie z databázy, vďaka čomu sa znižuje jej zaplnenie.
- **ValidateWebService.Controllers** - obsahuje triedy, tzv. Controllery, slúžiace na obsluhovanie požiadavkov zaslaných používateľmi webovej služby. Typický scénar je, že na webovú službu príde požiadavok *(request)*. Nasleduje fáza smerovania *(routingu)*, kedy sa zvolí podľa cesty requestu správny controller, ktorý obslúži daný request.
- **ValidateWebService.LanguageUtils** - obsahuje triedy, ktoré poskytujú pomocné funkcie k implementácii lokalizácie webovej služby.
- **ValidateWebService.Models** - obsahuje triedy, ktoré sa pomocou Entity Fameworku mapujú do databázy.
- **ValidateWebService.Security** - obsahuje triedy zabezpečujúce ochranu webovej služby pred nebezpečenstvom útoku. Momentálne sem zapadá napríklad trieda `Sanitizer`, ktorá sanitizuje vstup od získaný od používateľa webovej služby.

### 7.2.4 Webová aplikácia

Webová aplikáca sa nachádza v projekte 7.2.2 **WebApp.csproj** a jej hlavným menným priestorom 7.2.1 je **ValidateWebApp**.




Obr. 7.4: Menný priestor ValidateWebApp

Menný priestor **ValidateWebApp** obsahuje následovné vnorené menné priestory:

- **ValidateWebApp.Adapters** - obsahuje triedy slúžiace ako adaptéry. Tie slúžia ako obaly iných modulov, a poskytujú jednoduchšie rozhranie pre prácu s nimi v kontexte webovej aplikácie.

  Patrí tu napríklad trieda `DbAdapter`, ktorá obaľuje triedu `ServiceContext` slúžiacu na prístup k databáze, a poskytuje jednoduchšie rozhranie pre prácu s ňou, napríklad v Controlleroch.

- **ValidateWebApp.Constants** - menný priestor obsahujúci triedy s konštantnými symbolmi, ktoré využívajú komponenty webovej aplikácie.

  Patrí tu napríklad trieda `WebServiceConstants`, ktorá obsahuje adresu webovej služby, na ktorú smerujeme niektoré požiadavky *(requesty)* používateľov webovej aplikácie.

- **ValidateWebApp.Controllers** - obsahuje `Controllery` z populárneho návrhového vzoru MVC 6.1.1 definovanom v návrhu validátora. Takéto triedy sú zodpovedné za obsluhu jednotlivých požiadavkov prichádzajúcich na webovú aplikáciu.

- **ValidateWebApp.File** - obsahuje triedy poskytujúce potrebné funkcie pre spracovávanie lokálnych súborov nahratých požívateľom webovej aplikácie. Takéto súbory musíme uložiť dočasne na webový server, zapamätať si ich lokáciu a následne ich validovať.

- **ValidateWebApp.Models** - obsahuje modely definované v návrhovom vzore MVC 6.1.1. Takéto triedy poskytujú potrebné dáta k zobrazeniu pre `Views`.
- **ValidateWebApp.Resources** - obsahuje súbory s lokalizovaným textom a taktiež triedy poskytujúce prístup k nim.
- **ValidateWebApp.Result** - obsahuje prevažne adaptéry tried z knižnice, ktoré vytvárajú výsledky validácie v rozličných formátoch. Keďže webová aplikácia nevytvára takéto súbory hneď pri spracovaní požiadavku na validáciu, keďže väčšina používateľov takéto súbory nepotrebuje, serializujú sa objekty reprezentujúce chyby a varovania do databáze. Po istom čase však môže dôjsť k požiadavku na vytvorenie takéhoto súboru, no aplikácia má prístup len k serializovaným objektom. Preto je potrebné takéto serializované objekty najprv pred-spracovať, pred použitím tried `CsvResultCreator` či `RdfResultCreator` zabudovaných priamo v knižnici.
- **ValidateWebApp.Views** - obsahujú jednotlivé Views definované v návrhovom vzore MVC 6.1.1. Takéto `Views` poskytujú potrebné používateľské rozhranie pre používateľov a umožňujú vytvárať nové požiadavky na spracovanie pre webovú aplikáciu. V obrázku 7.4 sú znázornené inou farbou, keďže views sú implementované ako tzv. **Razor pages**, kde nedokážeme priamo využiť konštrukty menných priestorov.

# 8. Dokumentácia

V tejto kapitole nazrieme do dokumentácie validátora a jeho častí - knižnica, CLI aplikácia, webová aplikácia a webová služba. Správne softwarové dielo by malo obsahovať tri typy dokumentácie:

- **Používateľskú dokumentáciu** 8.1 - obsahuje pokyny, typické scenáre ako používateľ môže zaobchádzať s validátorom.
- **Vývojársku dokumentáciu** 8.2 - obsahuje dokumentáciu generovanú z kódu, ktorá je doplnená o dodatočné informácie potrebné pre vývojárov, ktorí by chceli na projekt nadviazať.
- **Administrátorská dokumentácia** 8.3 - obsahuje informácie potrebné pre spustenie knižnice, CLI aplikácie, webovej služby a webovej aplikácie.

Pre maximalizáciu koncových používateľov a uľahčenie ich práce sú takéto dokumentácie zverejnené za pomoci **GitHub Pages** v anglickom jazyku. Obsah repozitára `validateIT` je voľne dostupný na adrese `https://github.com/drexem/validateIT` a v prílohe práce môžeme nájsť dokumentačné `markdown(.md)` súbory v priečinku `github/docs`.

## 8.1 Používateľská dokumentácia

Používateľská dokumentáciu ukazuje príklady použitia, návody na použite a taktiež typické prípady použitia.

Naša používateľská dokumentácia pre:

- **Knižnicu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/lib/user/`
- **CLI aplikáciu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/cli/user/`
- **Webovú službu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/web_service/user/`
- **Webovú aplikáciu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/web_app/user/`

## 8.2 Vývojárska dokumentácia

Vývojárska dokumentácia popisuje všeobecnú architektúru projektu, zahŕňa generovanú dokumentáciu za pomoci doxygenu a taktiež popis najdôležitejších tried a rozhraní potrebných pre vývojára aplikácie.

Naša vývojárska dokumentácia pre:

- **Knižnicu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/lib/developer/`

- **CLI aplikáciu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/cli/developer/`
- **Webovú službu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/web_service/developer/`
- **Webovú aplikáciu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/web_app/developer/`

## 8.3 Administrátorská dokumentácia

Administrátorská dokumentácia obsahuje postupy a návody určené pre administrátora aplikácie. Typicky zahŕňa návody ako aplikáciu deploy-núť a prípadne konfigurovať.

Naša administrátorská dokumentácia pre:

- **Knižnicu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/lib/administration/`
- **CLI aplikáciu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/cli/administration/`
- **Webovú službu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/web_service/administration/`
- **Webovú aplikáciu** je dostupná na adrese:
  `https://drexem.github.io/validateIT/docs/web_app/administration/`

# 9. Testovanie

Testovanie tvorí neoddeliteľnú súčasť cyklu vývoja softvéru, zameranú na identifikovanie a opravu potenciálnych chýb a nezrovnalostí vrámci systému. Je sprevádzaná štrukturovaným prístupom k vyhodnocovaniu splnenia všetkých funkčných a kvalitatívnych požiadavkov, definovaných v špecifikácii, samotným systémom. Testy môžu byť vykonávané automaticky, kam napríklad patria **unit testy** a **integračné testy**.

Taktiež môžeme vykonať výkonnostné testy, kde analyzujeme výkon softvéru z hľadiska viacerých aspektov akými sú napríklad spotreba pamäte alebo dĺžky výpočtového času.

Software vyhodnotíme aj na základe spätnej väzby samotných koncových používateľov, ktorých sme identifikovali v sekcii 5.1 za pomoci štandardného nástroja *System Usability Scale*, čo predstavuje obecnú a široko používanú sadu otázok a metodiku k jej vyhodnoteniu k zmeraniu používateľskej prívetivosti systému.

**Definícia 9.0.1. Pokrytie kódu** *(Code coverage)* je metrika používaná vo vývoji softvéru, udávajúca množstvo vykonaného zdrojového kódu počas exekúcie testov. Často udávaná v podobe percenta porkytých riadkov alebo blokov zdrojového kódu.

V súvislosti s pokrytím zdrojového kódu sú často vyskytujúce sa pojmy:

- **Kolektor dát** *(Data collector)* - monitoruje spustenie a priebeh testov a získava informácie o jednotlivých behoch. O takýchto behoch vytvára správy *(reporty)* v rozličných formátoch, akými sú napríklad *JSON* či *XML*. Pre zber dát o testových behoch sme použili dátový kolektor zvaný **Coverlet**[36].
- **Generátor správ** *(Report generator)* - využíva dáta získané kolektormi dát, za pomoci ktorých generuje ľudsky čitateľné správy, zvyčajne v podobe pekne štylizovaných webových stránok. Pre účel generovania peknej ľudsky čitateľnej správy sme použili knižnicu **ReportGenerator**[37].

## 9.1 Unit testy

**Unit testy** patria k samotnému základu procesu vývoja softvéru. Takéto testy sú navrhnuté k overovaniu funkčnosti špecifických jednotiek kódu, než funkčnosti celého systému, typicky na úrovni metód či funkcii, zabezpečujúce funkčnosť jednotiek podľa požiadavkov zo špecifikácie. Primárnou funkciou je rýchle a včasné odhaľovanie chýb v softvére v samotnom počiatku vývojového procesu. Izolovaním jednotiek a ich testovaním v kontrolovanom prostredí môžu vývojári rýchlo identifikovať miesto vzniku chyby a reagovať patričným odstránením chyby. Ak jednotka musí komunikovať pri testovaní s inou jednotkou, použijeme takzvaný *Mock*, *Stub* prípadne *Fake*, ktoré predstavujú zjednodušenú implementáciu napodobujúcu funkčnosť pôvodnej jednotky.

### 9.1.1 Pokrytie unit testov

**Pokrytie zdrojového kódu unit testami**
V súčastnosti knižnica obsahuje **1539 parametrizovaných** unit testov.
Príklad parametrizovaného testu:

```
[Theory]
[InlineData("---02")]
[InlineData("---02Z")]
[InlineData("---02+00:00")]
public void ValidGDayWithoutPatternTest(string stringValue)
{
    DatatypeFactory.GetDatatype(stringValue, "gDay", null);
}
```

Pre každý väčší softvérový projekt je jedným z cieľov dostatočné pokrytie zdrojového kódu za pomoci automatizovaných unit testov. Zvyčajne sa takýto cieľ pohybuje v rozmedzí od 80% do 90% pokrytých riadkov, prípadne blokov zdrojového kódu.

Ako môžeme vidieť v tabuľke 9.1, unit testy pokrývajú zdrojový kód knižnice validátora vo výške 80.8%.

Zvyšných takmer 20% nepokrytého zdrojového kódu tvoria prevažne triedy reprezentujúce jednotlivé deskriptory, dátové typy, formáty dátových typov a veľké množstvo okrajových funkcionalít, ktoré validácia podľa špecifikácie **CSV on the Web ponúka**.

| Pokrytých riadkov | Nepokrytých riadkov | Percentuálne |
|---|---|---|
| 4682 | 1112 | 80.8% |

Tabuľka 9.1: Pokrytie zdrojového kódu - unit testy.

Samotný report je verejné dostupný všetkým používateľom validátoru na adrese:
`https://drexem.github.io/validateIT/docs/code_coverage/unit/html/index.html`.
Odkaz nájdeme aj v dokumentácii knižnice 8 a zdrojový kód nájdeme v priloženom GitHub repozitári.

## 9.2 Integračné testy

**Integračné testy** *(Integration tests)* predstavujú typ testovania softvérového systému, kde sú jednotlivé jednotky a komponenty systému testované ako skupina, aby sa zaistila ich správna prepojená funkčnosť. Narozdiel od unit testov, sa snažia kontrolovať interakcie medzi rozhraniami rozličných jednotiek a komponentov.

**W3C konzorcium** vydalo 281 integračných testov[15], ktoré by mali implementácie spĺňať, aby sa dali považovať za minimálne funkčné a bolo tak jednoduchšie porovnávať konkrétne implementácie. Knižnica úspešne implementuje všetkých vyššie spomínaných **281 integračných testov** [15].

Ukážka integračného testu:

```
[Fact]
public void ManifestValidationTest121()
{
```

```
    var controller = ControllerFactory.CreateController();
    var remoteTabularUrl =
        CSVW_TEST_URL_PREFIX + "test121.csv";
    var remoteMetadataUrl =
        CSVW_TEST_URL_PREFIX + "test121-user-metadata.json";

    var result =
    controller.ProcessOverriding(remoteMetadataUrl,remoteTabularUrl);
    Assert.Equal(
        ValidationResultType.VALID,
        result.GetValidationResultType
        );
}
```

### 9.2.1 Pokrytie zdrojového kódu integračnými testami + unit testami

Pri pohľade na tabuľku 9.2 môžeme vidieť, že unit testy spolu s integračnými testami pokrývajú dokonca až 82.6% zdrojového kódu, čo predstavuje viac než dostačujúce množstvo na to, aby sme mohli považovať validátor za funkčný.

Pre implementáciu testov:

- `https://w3c.github.io/csvw/tests/#manifest-validation#test014`
- `https://w3c.github.io/csvw/tests/#manifest-validation#test120`
- `https://w3c.github.io/csvw/tests/#manifest-validation#test122`

sme museli použiť testovacie rozhranie `ILinkLocationTest`, ktoré nám umožňuje umelo vložiť link HTTP hlavičky do odpovedí zo strany serveru, keďže server poskytujúci samotné testovacie súbory takéto hlavičky nevypĺňa.

| Pokrytých riadkov | Nepokrytých riadkov | Percentuálne(%) |
|---|---|---|
| 4790 | 1004 | 82.6 |

Tabuľka 9.2: Pokrytie zdrojového kódu - unit + integračné testy.

## 9.3 Výkonnostné testy

V tejto sekcii sa pokúsime vyhodnotiť validátor z hľadiska výkonu. Vytvoríme si benchmark, ktorý dokáže určiť približný očakávaný výkon validátora vzhľadom ku počtu stĺpcov 2.1.2, riadkov 2.1.4 a buniek 2.1.5 validovanej tabuľky 2.1.2.

### 9.3.1 Výkonnostný benchmark

**Definícia 9.3.1. Benchmark** je štandardizovaná metóda či množina kritérií používaná k vyhodnoteniu výkonu, efektívnosti či kvality systému, softvérovej aplikácie, hardvérového komponentu či procesu. V kontexte vývoja softvéru benchmark často zahrňuje meranie výpočtového času, spotreby pamäte alebo iných relevantných metrík časti kódu alebo celého systému za špecifických, presne definovaných podmienok.

Vytvorili sme benchmark 9.3.1, ktorý testuje priemerný výpočtový čas validátora pri validovaní tabuľkového súboru s popisujúcim metadátovým súborom. Benchmark bude vykonaný za pomoci knižnice BenchmarkDotNet[38], ktorá umožňuje takýto benchmark 9.3.1 vytvoriť jednoducho pomocou anotácií v zdrojovom kóde.

Benchmark sa bude vykonávať na súboroch z praxe, ktoré sú voľné dostupné na stránkach otvoreného dátového katalógu data.gov.cz. Testovať budeme najmä prípad, kedy máme metadátový súbor, ktorý popisuje tabuľkový súbor. Jedná sa o najčastejší prípad použitia validátora a patrí medzi najzákladnejšie dôvody vzniku samotného systému.

Konkrétne testovacie súbory sú zaznamenané v tabuľke 9.3, kde sú taktiež poskytnuté odkazy pre získanie daných súborov. Zároveň sú v tabuľke 9.3 zachytené aj veľkosti jednotlivých tabuľkových súborov, ktoré chceme validovať. Pri výbere sme preferovali súbory s počtom riadkov väčším ako 1000 pre získanie väčšej presnosti, keďže u menších súborov môže byť väčšina času strávená samotným sťahovaním vzdialeného metadátového súboru.

Projekt, ktorý slúžil pre benchmark je dostupný v priloženom GitLab repozitári v priečinku `Performance/PerformanceTests`.

| Zdroj | Schéma | Tabuľkový súbor | Počet riadkov | Počet stĺpcov |
|---|---|---|---|---|
| Test01 | odkaz | odkaz | 61361 | 12 |
| Test02 | odkaz | odkaz | 1338084 | 17 |
| Test03 | odkaz | odkaz | 53999 | 7 |
| Test04 | odkaz | odkaz | 980131 | 7 |
| Test05 | odkaz | odkaz | 19531 | 15 |
| Test06 | odkaz | odkaz | 1401 | 17 |
| Test07 | odkaz | odkaz | 9872 | 9 |
| Test08 | odkaz | odkaz | 164029 | 8 |
| Test09 | odkaz | odkaz | 174270 | 2 |
| Test10 | odkaz | odkaz | 34339 | 14 |

Tabuľka 9.3: Súbory použité pre benchmark

### Testovací stroj

Testovanie prebiehalo na stroji:

1. **Operačný systém** - Windows 11
2. **.NET SDK** - 8.0.203
3. **CPU** - AMD Ryzen 7 4800H, 1 CPU, 16 logických a 8 fyzických jadier
4. **GPU** - Nvidia GTX 1660 TI Max-Q
5. **veľkosť RAM** - 16GB

#### Benchmark s remote súbormi

Výsledky benchmarku 9.3.1 s remote súbormi môžeme vidieť v tabuľke 9.4. Každý test 01-10 bol spustený opakovane 16-krát a výpočtový čas trvania testu bol následne zpriemerovaný.

Testované súbory obsahujú spolu **2 837 017 riadkov**.

Testované súbory obsahujú spolu **33 269 815 buniek**.

Súčet priemerných časov jednotlivých testov činí **74.46s**.

Priemerne sa spracuje za sekundu **38 101** riadkov.

Priemerne sa za sekundu spracuje **465 572** buniek.

| Test | Priemerný čas | Stĺpcov | Riadkov | Buniek |
|---|---|---|---|---|
| Test01Remote | 3.13s | 12 | 61361 | 736 332 |
| Test02Remote | 25.25s | 17 | 1 338 084 | 22 747 428 |
| Test03Remote | 2.11s | 7 | 53 999 | 377 993 |
| Test04Remote | 32.03s | 7 | 980 131 | 6 860 917 |
| Test05Remote | 1.72s | 15 | 19 531 | 292 965 |
| Test06Remote | 0.46s | 17 | 1 401 | 23 817 |
| Test07Remote | 0.70s | 9 | 9 872 | 88 848 |
| Test08Remote | 4.12s | 8 | 164 029 | 1 312 232 |
| Test09Remote | 2.48s | 2 | 174 270 | 348 540 |
| Test10Remote | 2.46s | 14 | 34 339 | 480 746 |

Tabuľka 9.4: Výsledky benchmarku s remote súbormi

#### Benchmark s lokálnymi súbormi

Výsledky benchmarku 9.3.1 s lokálnymi súbormi môžeme vidieť v tabuľke 9.5. Každý test bol spustený opakovane 16-krát a výpočtový čas trvania testu bol následne zpriemerovaný.

Testované súbory obsahujú spolu **2 837 017 riadkov**.

Testované súbory obsahujú spolu **33 269 815 buniek**.

Súčet priemerných časov jednotlivých testov činí **57.15s**.

Priemerne sa za sekundu spracuje **49 641** riadkov.

Priemerne sa za sekundu spracuje **582 148** buniek.

Lokálne testy bežia očakávane rýchlejšie, než ich remote varianta, ktorá musí pred začatím validácie stiahnuť, niekedy aj niekoľko-megabajtové, súbory.

### 9.3.2 Identifikácia bottle-neckov

Pre identifikáciu bottle-neckov sme využili profilovacie nástroje aplikácie *Visual Studio 2022 Professional*, ktoré umožňujú detailne sledovať viaceré aspekty pri behu aplikácie, akými sú napríklad spotreba pamäte, využitie zdrojov CPU a iné.

My sme sa zamerali na identifikáciu bottle-neckov výkonu, keďže aplikácia je pamäťovo nenáročná (až na výnimky 7.2.1).

| Test | Priemerný čas | Stĺpcov | Riadkov | Buniek |
|---|---|---|---|---|
| Test01Local | 1.58s | 12 | 61361 | 736 332 |
| Test02Local | 14.73s | 17 | 1 338 084 | 22 747 428 |
| Test03Local | 2.23 | 7 | 53 999 | 377 993 |
| Test04Local | 32.01s | 7 | 980 131 | 6 860 917 |
| Test05Local | 1.13s | 15 | 19 531 | 292 965 |
| Test06Local | 0.11s | 17 | 1 401 | 23 817 |
| Test07Local | 0.16s | 9 | 9 872 | 88 848 |
| Test08Local | 2.28s | 8 | 164 029 | 1 312 232 |
| Test09Local | 1.23s | 2 | 174 270 | 348 540 |
| Test10Local | 1.69s | 14 | 34 339 | 480 746 |

Tabuľka 9.5: Výsledky benchmarku s lokálnymi súbormi

Z tejto štatistiky vyskytlo, že takmer 65% výpočtových zdrojov CPU sa aktuálne vynakladá na metódy spojené s parsovaním samotného tabuľkového súboru. Parsovanie tabuľkového súboru teda predstavuje hlavný bottle-neck aplikácie a preto sa tu vyskytuje obrovský priestor pre budúce vylepšenia.

## 9.4 Testovanie používateľmi

Pre overenie používateľskej prívetivosti webovej aplikácie sme sa rozhodli využiť *System Usability Scale*[39]. Jedná sa o štandardizovanú sadu 10 otázok a metodiku k ich vyhodnoteniu, ktorá nám určí skóre, ktoré slúži ako všeobecný indikátor používateľskej prívetivosti softvéru.

### 9.4.1 Dotazník

Dotazník obsahuje sadu 10 typických otázok/tvrdení v anglickom jazyku. Tester následne zvolí číslo od 1 do 5, vzhľadom na to, ako veľmi s daným tvrdením súhlasí. Ak používateľ označí 5 , znamená to, že silne súhlasí. Ak používateľ označí 1, znamená to, že silne nesúhlasí.

**Otázky/tvrdenia v dotazníku:**

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.

7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

### 9.4.2 Priebeh testovania

Používateľskú prívetivosť sme testovali na 3 ľuďoch, ktorí spĺňajú typické používateľské role definované v sekcii 5.1. Títo testeri zastávajú roľu študentov vzdelávajúcich sa v oblasti dátových formátov.

Pre testovanie webovej aplikácie sme využili príklady, ktoré sú dostupné v používateľskej dokumentácii 8.1, konkrétne na adrese `https://drexem.github.io/validateIT/docs/web_app/user/#typical-usage-scenarios` Jedná sa o typické prípady použitia, ktoré sme vyvodili počas analýzy validátora 5.

Po vyskúšaní si typických prípadov použitia validátora boli testeri požiadaní o vyplnenie vyššie spomínaného dotazníku.

### 9.4.3 Spôsob vyhodnotenia dotazníku

Z odpovedí každého testera vyrátame tzv. SUS skóre. To vypočítame následovne:

- Za každú nepárnu otázku pripočítame počet bodov z odpovede zmenšený o 1.
- Za každú párnu otázku pripočítame 5 bodov zmenšených o počet bodov z odpovede.
- Daný počet bodov vynásobíme číslom 2.5.

Následne takéto SUS skóre získané od 3 testerov spriemerujeme a získame priemerné SUS skóre nášho systému.,

Taktiež vieme systém na základe dosiahnutého skóre zaradiť do jednej z kategórii [40]:

- **A = skóre > 80.3** - používatelia takýchto systémov pravdepodobne doporučia jeho použitie aj svojim priateľom a známym.
- **B = skóre > 74** - používateľnosť takýchto systémov vnímaná používateľmi je vyššia, než 70% iných systémov.
- **C = skóre > 68**
- **D = skóre > 51**
- **F = skóre <= 51** - patrí tu 15% najhorších systémov z hľadiska dosiahnutého SUS skóre.

### 9.4.4 Vyhodnotenie dotazníku

V Table 9.6 môžeme vidieť odpovede testerov na jednotlivé otázky v dotazníku.

| Otázka | Tester 1 | Tester 2 | Tester 3 |
|---|---|---|---|
| Otázka č. 1 | 4 | 5 | 5 |
| Otázka č. 2 | 1 | 1 | 1 |
| Otázka č. 3 | 5 | 5 | 5 |
| Otázka č. 4 | 1 | 1 | 1 |
| Otázka č. 5 | 5 | 5 | 5 |
| Otázka č. 6 | 1 | 2 | 1 |
| Otázka č. 7 | 5 | 5 | 5 |
| Otázka č. 8 | 1 | 1 | 1 |
| Otázka č. 9 | 5 | 5 | 5 |
| Otázka č. 10 | 1 | 1 | 1 |

Tabuľka 9.6: Odpovede testerov v dotazníku

V Table 9.7 môžeme vidieť skóre validátora, vzhľadom k odpovediam poskytnutými jednotlivými testermi. Na základe týchto hodnôt vieme vyrátať priemerné skóre systému: $98.\overline{3}$.. Takéto skóre patrí do triedy **A** - najlepšej triedy systémov z hľadiska používateľnosti, čo naznačuje používateľskú prívetivosť systému.

| Tester | Skóre |
|---|---|
| Tester č. 1 | 97.5 |
| Tester č. 2 | 97.5 |
| Tester č. 3 | 100 |

Tabuľka 9.7: Usability skóre na základe odpovedí testerov

# 10. Vyhodnotenie validátora

V následujúcej kapitole sa pokúsime objektívne vyhodnotiť implementáciu validátora **validateIT**. Vyhodnotíme ho na základe porovnania s konkurenčnými implementáciami z hľadiska spĺňania integračných testov[15] ale aj z hľadiska výkonu. V poslednej časti si zhrnieme možné nedostatky validátora a možné vylepšenia do budúcna.

## 10.1 Vyhodnotenie voči integračným testom

V kapitole 4 sme ukázali existujúce implementácie validátorov podľa doporučení CSV on the Web. Teraz si zhrnieme, do akej miery sú jednotlivé riešenia efektívne z hľadiska spĺňania integračných testov [15]. Ako môžeme vidieť v tabuľke 10.1, len 2 zo 4 referenčných implementácii spĺňajú všetky integračné testy na 100%. Tabuľka zobrazuje mená jednotlivých implementácii a počet testov, ktoré úspešne implementujú. Celkový počet integračných testov[15] je 281.

| **CSV Lint** | **pycsvw** | **RDF::Tabular** | **csvw-validator** |
|---|---|---|---|
| 281/281 | 158/281 | 281/281 | 262/281 |
| 100% | 56.2% | 100% | 93.2% |

Tabuľka 10.1: Implementácia integračných testov - referenčné riešenia.

Implementácia **CSV Lint** síce spĺňa všetkých 281 integračných testov[15], avšak inštalácia pre operačný systém **Windows** je komplikovaná a chýbajúca administrátorská dokumentácia 8.3 situáciu komplikuje. Webová aplikácia neumožňuje pracovať so schémami CSV on the Web[13], čo obmedzuje jej použitie v praxi.

Implementácia **pycsvw** s nízkou úspešnosťou integračných testov[15] **56.2%**, chýbajúcou dokumentáciou, náročnou inštaláciou a nutnosťou použitia `Python2`, ktorý dnes už nie je štandardom, je pre spoľahlivú validáciu tabuľkových súborov podľa odporúčaní CSV on the Web[8] takmer nepoužiteľná.

Implementácia od jedného z autorov odporúčaní **Gregga Kellogga** implementuje všetkých 281 integračných testov[15]. Táto knižnica však má komplikovanú inštaláciu pre operačný systém Windows, a taktiež chýba pokročilejšie používateľské rozhranie v podobe webovej aplikácie, prípadne klasickej aplikácie, čo limituje vedomostne používateľov takejto aplikácie.

Implementácia **csvw-validator** je ako jediná napísaná v jednom populárnejších jazykov súčastnosti - jave, avšak táto implementácia spĺňa iba 262/281 integračných testov[15]. Implementácia nepodporuje dialekty 3.4.9 a taktiež ani cudzie kľúče subsection 3.4.5, čo dosť obmedzuje použitie implementácie, nakoľko existuje množstvo tabuliek, ktoré používajú cudzie kľúče, prípadne používajú iný separátor ako je čiarka. Implementácia však ponúka príjemné používateľské rozhranie v podobe webovej aplikácie.

Naša implementácia v podobe **validateIT** spĺňa všetkých 281 integračných testov [15], plne podporuje odporúčania CSV on the Web[8], ako môžeme vidieť v tabuľke 10.2.

Nedovolíme si však tvrdiť, že validátor neobsahuje žiadne chyby, keďže ako sme spomenuli v sekcii section 9.1 unit testy a integračné testy nepokrývajú 100% široké možnosti validácie. Zároveň, validátor bol testovaný len malou skupinou ľudí, čo zvyšuje potencionálne riziko výskytu neobjavených chýb.

| **CSV Lint** | **pycsvw** | **RDF::Tabular** | **csvw-validator** | **validateIT** |
|---|---|---|---|---|
| 281/281 | 158/281 | 281/281 | 262/281 | **281/281** |
| 100% | 56.2% | 100% | 93.2% | **100%** |

Tabuľka 10.2: Implementácia integračných testov - referenčné riešenia + validateIT.

## 10.2 Vyhodnotenie validátora vzhľadom k výkonu

Ako sme si ukázali už v sekcii 9.3, validátor dosahuje obstojné čísla vzhľadom na počet zvalidovaných riadkov, prípadne buniek, za sekundu. V tejto sekcii sa ale pokúsime objektívne porovnať výkon **validateIT** validátora vzhľadom k iným referenčným implementáciám chapter 4.

### 10.2.1 Vyhodnotenie výkonu v porovnaní s referenčnými implementáciami

Výkon validátoru si porovnáme s dvomi referenčnými validátormi - **CSV Lint** 4.1 a **RDF::Tabular** 4.3. Obidva referenčné validátory plne podporujú 100% všetkých integračných testov [15] a poskytujú rozumné rozhranie pre použitie na príkazovom riadku. Implementáciu **csvw-validator** nebudeme do testov zahŕňať, na koľko nepodporuje 100% integračných testov [15] a chýbajúca podpora pre dialekty 3.4.9 jej umožňuje rýchlejšie a menej obecné parsovanie tabuľkových súborov. Knižnica **pycsvw** section 4.2 je s nízkym počtom implementovaných integračných testov [15] (56.2%) irelevantná pre objektívne porovnávanie výkonu.

#### Testovací stroj

Testovanie prebiehalo na stroji:

1. **Operačný systém** - Linux fedora 6.7.5-100.fc38.x86_64
2. **.NET SDK** - 8.0.203
3. **CPU** - AMD Ryzen 7 4800H, 1 CPU, 16 logických a 8 fyzických jadier
4. **GPU** - Nvidia GTX 1660 TI Max-Q
5. **veľkosť RAM** - 16GB

### Testovacie súbory

Testovaných bolo 5 lokálnych tabuľkových súborov s popisujúcimi metadátovými súbormi a vždy začínalo z popisujúceho metadátového súboru. Testovacie súbory môžeme vidieť v tabuľke 10.3. Použili sme lokálnu validáciu pre obmedzenie skreslenia internetového pripojenia.

| Zdroj | Schéma | Tabuľkový súbor | Počet riadkov | Počet stĺpcov |
|---|---|---|---|---|
| Test01 | odkaz | odkaz | 61361 | 12 |
| Test02 | odkaz | odkaz | 53999 | 7 |
| Test03 | odkaz | odkaz | 164029 | 8 |
| Test04 | odkaz | odkaz | 174270 | 2 |
| Test05 | odkaz | odkaz | 34339 | 14 |

Tabuľka 10.3: Súbory použité pre porovnanie výkonu

V niektorých súboroch sme museli odobrať niektoré spoločné vlastnosti 3.4.8, nakoľko obsahovali odkazy obsahujúce znaky českej abecedy a implementácie **RDF::Tabular** a **CSV Lint** nepodporujú takéto znaky v URL a hlásia chybu.

Taktiež sme museli odstrániť Byte Order Mark (BOM) z tabuľkových súborov `Test01 Test03 Test05`, ktoré spôsobovali chybu v oboch referenčných validátoroch, nakoľko týchto pár bytov nedokázali rozpoznať a pridali ich k názvu prvého stĺpca tabuľky. Z toho následne vznikla chyba, kedy schéma 3.4.4 v metadátovom súbore nie je kompatibilná 3.4.4 s vnorenými metadátami item 4, nakoľko názvy stĺpcov sa odlišujú. Validátor **validateIT** dokáže rozpoznať prítomnosť BOM bytov, ohlási varovanie a následne úspešne zvaliduje požadované súbory.

Takto zmenené súbory sú dostupné na adrese a taktiež sú obsiahnuté v priloženom GitLab repozitári.

### Meranie výpočtového času

Meranie výpočtového času prebiehalo za pomoci jednoduchého bash scriptu, ktorý využíva linuxový nástroj time. Každý z testovaných súborov bol pustený takýmto skriptom opakovane 10-krát po sebe a výsledný čas sme spriemerovali. Bash script použitý pre meranie jedného testu:

```
#!/bin/bash
# Check if the command argument is provided
if [ $# -eq 0 ]; then
    echo "Usage: $0 <command>"
    exit 1
fi
TIMEFORMAT="%E";
COMMAND="$@";
NUM_RUNS=10

ELAPSED_TIME=‘time (
    for (( i=1; i<=$NUM_RUNS; i++ ))
    do
```

```
        $COMMAND > /dev/null
    done
) 2>&1‘;

echo "The command $COMMAND has been run $NUM_RUNS times";
echo "Elapsed time: $ELAPSED_TIME seconds";
exit 0;
```

Implementácia **RDF::Tabular** je natoľko pomalá, že sme museli limitovať výpočtový čas pre jednotlivé testovanie vo všetkých testoch. Limitovali sme ho na 100-násobok výpočtového času validátora **validateIT**. Takéto skoré ukončenie testu je v Table 10.4 vyznačené pomocou symbolu + pri výpočtovom čase **RDF::Tabular**.

Pri testovaní výkonu **RDF::Tabular** sme dokonca nemohli použiť relatívne, či absolútne cesty pre zadanie cesty k metadátovému súboru pre spustenie validácie. Validáciu sme museli spúšťať validáciu priamo v priečinku obsahujúcom dané testovacie súbory.

Príklad problematického spustenia **RDF::Tabular**:

```
rdf validate
--input-format tabular TestFiles/Test01/ockovani-spotreba.csv-metadata.json
```

Príklad fungujúceho spustenia **RDF::Tabular**:

```
rdf validate --input-format tabular ockovani-demografie.csv-metadata.json
```

### Výsledky merania

Výsledky jednotlivých testov môžeme vidieť v tabuľke 10.4.

| Zdroj | validateIT | RDF::Tabular | CSV Lint | Počet riadkov |
|---|---|---|---|---|
| Test01 | 1.89s | 190+s | 10.81s | 61361 |
| Test02 | 3.05s | 305+s | 10.37s | 53999 |
| Test03 | 2.46s | 246+s | 22.60s | 164029 |
| Test04 | 1.51s | 175+s | 17.46s | 174270 |
| Test05 | 1.77s | 177+s | 6.09s | 34339 |

Tabuľka 10.4: Výsledky merania pre porovnanie výkonu

**Test01** pre **RDF::Tabular** bol vypnutý po 190s:

```
timeout -v 190s
    rdf validate --input-format tabular ockovani-spotreba.csv-metadata.json
timeout: sending signal TERM to command ‘rdf’
```

**Test02** pre **RDF::Tabular** bol vypnutý po 305s:

```
timeout -v 305s
    rdf validate --input-format tabular 2007.json
timeout: sending signal TERM to command ‘rdf’
```

**Test03** pre **RDF::Tabular** bol vypnutý po 246s:

```
timeout -v 246s
    rdf validate --input-format tabular ockovani-demografie.csv-metadata.json
timeout: sending signal TERM to command ‘rdf’
```

**Test04** pre **RDF::Tabular** bol vypnutý po 175s:

```
timeout -v 175s rdf validate --input-format tabular
                  dopravni-urazy-diagnozy-t.csv-metadata.json
timeout: sending signal TERM to command ‘rdf’
```

**Test05** pre **RDF::Tabular** bol vypnutý po 177s:

```
timeout -v 177s
    rdf validate --input-format tabular mestske-casti.csv-metadata.json
timeout: sending signal TERM to command ‘rdf’
```

Pre overenie správneho spúšťania testov pre **RDF::Tabular** sme sa pokúsili zmenšiť testované súbory a v takýchto prípadoch program dobehol do požadovaného časového limitu. Problém je teda zakorenený v neefektívnosti implementácie.

### Zhrnutie výsledkov

Ako môžeme vidieť v tabuľke 10.5, validátor **validateIT** je minimálne 3.4-krát rýchlejší, než jeho konkurent **CSV Lint** a priemerne 6.65-krát rýchlejší. Rýchlejší je najmä u väčších súborov s vyšším počtom riadkov a buniek akými sú napríklad **Test03** či **Test04**.

Referenčná implementácia **RDF::Tabular** spotrebovala vždy minimálne 100-násobok času potrebného pre dokončenie testu validátorom **validateIT**, no napriek tomu nebola nikdy schopná dokončiť validáciu včas. Preto je implementácia **RDF::Tabular** minimálne 100-násobne pomalšia, než implementácia **validateIT**.

| Zdroj | validateIT | CSV Lint | $\frac{CSV\,Lint}{validateIT}$ | Počet riadkov |
|---|---|---|---|---|
| Test01 | 1.89s | 10.81s | 5.71 | 61361 |
| Test03 | 3.05s | 10.37s | 3.4 | 53999 |
| Test08 | 2.46s | 22.60s | 9.18 | 164029 |
| Test09 | 1.51s | 17.46s | 11.56 | 174270 |
| Test10 | 1.77s | 6.09s | 3.44 | 34339 |

Tabuľka 10.5: Porovnanie validateIT a CSV Lint

## 10.3 Budúci vývoj

Ako už bolo spomenuté v sekcii 9.3, hlavným bottle-neckom validátora z hľadiska výkonu tvorí parsovanie tabuľkových súborov. To by sa dalo v budúcnosti vylepšiť za pomoci viac-vláknového spracovania, prípadne vytvorením efektívnejšieho parsera, ktorý by bol navrhnutý špeciálne pre potreby parsovania CSV súborov. Aktuálny parser využíva totiž niektoré obecnejšie metódy na prácu s reťazcami definované v samotnom jazyku C#.

Potencionálnemu vylepšeniu by mohli podliehať aj validačné pravidlá pre primárne kľúče 7.2.1 a cudzie kľúče 7.2.1, ktoré momentálne obmedzujú maximálnu možnú veľkosť súboru, ktorý sme schopní validovať.

Refaktorizácia istých častí aplikácie by mohla prispieť ku krajšiemu a čitateľnejšiemu kódu. Rovnako navrhnutá architektúra 6.1 by sa dala vylepšiť pre prípadnú vyššiu výkonnosť a rozšíriteľnosť.

Veríme však, že s detailnou a rozsiahlou dokumentáciou 8, budúci vývojári validátora budú schopní vykonať vyššie spomínané úpravy.

Na záver by sme radi dodali, že náš NuGet balíček s knižnicou validátora je pomerne populárny a momentálne dosiahol už vyše **290 stiahnutí**. S prácou bola odovzdaná verzia NuGet balíčka 1.0.3. V GitLab a GitHub repozitároch je commit odovzdania označený tagom `bp-final`.

# Záver

Cieľom práce bolo vytvorenie funkčného validátoru CSV súborov podľa doporučení CSV on the Web vydaných skupinou W3C Working Group. Validátor mal byť schopný validovať CSV súbory s pripojeným metadátovým súborom obsahujúcim tabuľkovú schému. V prípade výskytu chýb mal rozumne oznámiť problémy používateľovi s prípadnými odkazmi, kde by sa používateľ mohol dozvedieť viac o danej problematike. Validátor mal taktiež poskytnúť 3 rozumné používateľské rozhrania v podobe CLI aplikácie, webovej služby a webovej aplikácie.

Existujúce riešenia validátorov momentálne buď postrádajú dokumentáciu, sú malo udržiavané, častokrát sú spojené s problematickým spustením pre operačný systém Windows, neimplementujú všetky referenčné integračné testy alebo neposkytujú prívetivé používateľské rozhranie. Existujú 2 riešenia s webovou aplikáciou, avšak jedno z nich nepodporuje schémy podľa doporučenia CSV on the Web a druhé z nich neimplementuje všetky integračné testy, a teda ho nemôžeme považovať za plne funkčné.

Validátor sme implementovali v podobe knižnice v jazyku C#, a zverejnili sme ju pre jednoduché použitie aj v podobe nugetového balíčku, ktorý už dosahuje vyše 290 stiahnutí používateľmi. Takúto knižnicu využívajú za pomoci rozumného rozhrania aplikácie poskytujúce používateľské rozhrania v podobe CLI aplikácie, webovej služby a webovej aplikácie.

Validátor plne implementuje všetkých 281 integračných testov CSV on the Web a taktiež obsahuje množstvo unit testov, čo do istej miery dokazuje jeho funkčnosť.

Pre maximalizáciu používateľskej prívetivosti sme vykonali prieskum System Usability Scale (SUS), čo nám umožnilo vyhodnotiť validátor po stránke používateľskej prívetivosti.

Taktiež sme vytvorili detailnú a verejnú dokumentáciu softvérového diela, ktorá je dostupná za pomoci technológie GitHub Pages pre všetkých potenciálnych používateľov validátora v anglickom jazyku. Tým sme maximalizovali dosah našej práce pre široké publikum.

Práca sa dá rozšíriť o implementáciu transformácie CSV súborov do formátov JSON či RDF, kde definícia takýchto transformácii je súčasťou doporučení CSV on the Web. Možnosť o vylepšenie si žiadajú aj časti zabezpečujúce validáciu cudzích a primárnych kľúčov, čo do istej miery obmedzuje maximálnu veľkosť tabuľkových súborov, ktoré sme schopní spracovávať. Takisto priestor na zlepšenie vieme nájsť pri parsovaní CSV súborov, čo tvorí najväčší bottleneck z hľadiska výkonu validácie.

# Zoznam použitej literatúry


[1] Network Working Group. Common format and mime type for comma-separated values (csv) files. https://www.rfc-editor.org/rfc/rfc4180, 2005. Prehliadané dňa: 2023-04-18.

[2] Open Data Institute. Uri fragment identifiers for the text/csv media type. https://www.rfc-editor.org/rfc/rfc7111, 2014. Prehliadané dňa: 2023-04-18.

[3] IANA. Iana - tab-separated values (tsv). https://www.iana.org/assignments/media-types/text/tab-separated-values. Prehliadané dňa: 2024-02-20.

[4] Network Working Group. Uniform resource identifier (uri): Generic syntax. https://www.rfc-editor.org/rfc/rfc3986, 2005. Prehliadané dňa: 2024-02-20.

[5] Network Working Group. Uri fragment identifiers for the text/plain media type. https://www.rfc-editor.org/rfc/rfc5147, 2008. Prehliadané dňa: 2024-02-20.

[6] Mark Davis; Ken Whistler. Unicode normalization forms. https://www.unicode.org/reports/tr15/, 2012. Prehliadané dňa: 2024-02-20.

[7] W3C. Model for tabular data and metadata on the web. https://www.w3.org/TR/2015/REC-tabular-data-model-20151217/, 2015. Prehliadané dňa: 2023-04-18.

[8] W3C. Csv on the web: A primer. https://www.w3.org/TR/tabular-data-primer/, 2016. Prehliadané dňa: 2024-03-11.

[9] Phillips;M. Davis. Tags for identifying languages. https://www.rfc-editor.org/info/bcp47, 2009. Prehliadané dňa: 2024-03-07.

[10] David Peterson; Sandy Gao; Ashok Malhotra; Michael Sperberg-McQueen; Henry Thompson; Paul V. Biron. W3c xml schema definition language (xsd). https://www.w3.org/TR/xmlschema11-2/, 2012. Prehliadané dňa: 2024-03-07.

[11] Internet Engineering Task Force (IETF). Defining well-known uniform resource identifiers (uris). https://datatracker.ietf.org/doc/html/rfc5785, 2010. Prehliadané dňa: 2023-04-23.

[12] Internet Engineering Task Force (IETF). Uri template. https://datatracker.ietf.org/doc/html/rfc6570, 2012. Prehliadané dňa: 2023-04-23.

[13] W3C. Metadata vocabulary for tabular data. https://www.w3.org/TR/2015/REC-tabular-metadata-20151217/, 2015. Prehliadané dňa: 2023-04-18.

[14] University of Minnesota. Definition of tab-separated-values (tsv). https://www.iana.org/assignments/media-types/text/tab-separated-values, Uknown. Prehliadané dňa: 2023-04-18.

[15] W3C. Csvw test cases. https://w3c.github.io/csvw/tests/#manifest-validation, 2015. Prehliadané dňa: 2024-03-12.

[16] Data Liberation Front. Csv lint. https://github.com/Data-Liberation-Front/csvlint.rb. Accessed: 2024-04-23.

[17] Sebastian Neumaier;Jürgen Umbrich. Python implementation of the w3c csv on the web specification. https://github.com/sebneu/csvw-parser. Prehliadané dňa: 2024-03-12.

[18] Gregg Kellogg. Tabular data rdf reader and json serializer. https://github.com/ruby-rdf/rdf-tabular. Accessed: 2024-04-23.

[19] Vojtech Malý. csvw-validator. https://github.com/malyvoj3/csvw-validator. Prehliadané dňa: 2024-03-12.

[20] Network Working Group. Internationalized resource identifiers (iris). https://www.rfc-editor.org/rfc/rfc3987, 2005. Prehliadané dňa: 2023-04-18.

[21] Internet Engineering Task Force (IETF). The javascript object notation (json) data interchange format. https://datatracker.ietf.org/doc/html/rfc8259, 2017. Prehliadané dňa: 2023-08-15.

[22] G. Camarillo. The internet assigned number authority (iana) header field parameter registry for the session initiation protocol (sip). https://datatracker.ietf.org/doc/html/rfc3968, 2004. Prehliadané dňa: 2023-07-13.

[23] Microsoft. An introduction to nuget. https://learn.microsoft.com/en-us/nuget/what-is-nuget, 2022. Prehliadané dňa: 2024-03-25.

[24] Josh Close. Csvhelper. https://joshclose.github.io/CsvHelper/, 2009. Prehliadané dňa: 2024-03-25.

[25] dotNetRDF Project. dotnetrdf. https://dotnetrdf.org/, 2007. Prehliadané dňa: 2024-03-25.

[26] Adam White. Extendednumerics.bigdecimal. https://www.nuget.org/packages/ExtendedNumerics.BigDecimal. Prehliadané dňa: 2024-03-25.

[27] James Newton-King. Json.net. https://www.newtonsoft.com/json. Prehliadané dňa: 2024-03-25.

[28] Tavis Rudd. Tavis uri templates. https://github.com/tavis-software/Tavis.UriTemplates. Prehliadané dňa: 2024-03-25.

[29] Julian Verdurmen. Utf.unknown. https://github.com/CharsetDetector/UTF-unknown?tab=readme-ov-file. Prehliadané dňa: 2024-03-25.

[30] Eric Newton. Commandlineparser. https://www.nuget.org/packages/CommandLineParser/. Prehliadané dňa: 2024-03-25.

[31] Microsoft. Microsoft.net.build.containers. https://www.nuget.org/packages/Microsoft.NET.Build.Containers. Prehliadané dňa: 2024-03-25.

[32] Microsoft. Microsoft.aspnetcore.cors. https://www.nuget.org/packages/Microsoft.AspNetCore.Cors. Prehliadané dňa: 2024-03-25.

[33] Microsoft. Microsoft.aspnetcore.mvc.localization. https://www.nuget.org/packages/Microsoft.AspNetCore.Mvc.Localization/. Prehliadané dňa: 2024-03-25.

[34] xunit; dotnetfoundation. xunit. https://www.nuget.org/packages/xunit. Prehliadané dňa: 2024-03-25.

[35] Toni Solarin-Sodara; dotnetfoundation. coverlet.collector. https://www.nuget.org/packages/coverlet.collector. Prehliadané dňa: 2024-03-25.

[36] Tom Englundd. Coverlet. https://github.com/coverlet-coverage/coverlet. Prehliadané dňa: 2024-04-01.

[37] Coverlet Team. Reportgenerator. https://github.com/danielpalme/ReportGenerator. Prehliadané dňa: 2024-04-01.

[38] Andrey Akinshin. Benchmarkdotnet. https://github.com/dotnet/BenchmarkDotNet. Prehliadané dňa: 2024-04-02.

[39] John Brooke. Sus: a quick and dirty usability scale. https://www.researchgate.net/publication/319394819_SUS_--_a_quick_and_dirty_usability_scale, 1996. Accessed: 2024-04-19.

[40] Jeff Sauro. Measuring usability with the system usability scale. https://measuringu.com/sus/. Prehliadané dňa: 2024-04-15.

# Zoznam obrázkov

# Zoznam tabuliek

# Zoznam použitých skratiek

- **DSV** - Delimiter-Separated Values
- **CSV** - Comma-Separated Values
- **TSV** - Tab-Separated Values
- **RDF** - Resource Definition Framework
- **CSVW** - CSV on the Web
- **BOM** - Byte Order Mark
- **CLI** - Command Line Interface
- **URL** - Uniform Resource Locator
- **IRI** - Internationalized Resource Identifier
- **URN** - Uniform Resource Name
- **URI** - Uniform Resource Identifier
- **IANA** - Internet Assigned Numbers Authority
- **RFC** - Request For Comments
- **CR** - Carriage Return
- **LF** - Line Feed
- **UTF** - Unicode Transformation Format
- **HTTP** - Hypertext Transfer Protocol
- **HTTPS** - Hypertext Transfer Protocol Secure
- **JSON** - Javascript Object Notation
- **EOL** - End Of Line