Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Porovnání nástrojů SoapUI a Postman pro testování rozhraní REST API

DIPLOMOVÁ PRÁCE

Studijní program: Aplikovaná informatika

Studijní obor: Informační systémy a technologie

Autor: Bc. Vojtěch Votlučka

Vedoucí diplomové práce: doc. Ing. Alena Buchalcevová, Ph.D.

Praha, Duben 2019

Prohlášení

Prohlašuji, že jsem diplomovou práci Porovnání nástrojů SoapUI a Postman pro testování rozhraní REST API vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne 22. dubna 2019 .....

Vojtěch Votlučka

Poděkování

Chtěl bych poděkovat doc. Ing. Aleně Buchalcevové, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnovala. Mé poděkování patří též mým rodičům, celé rodině i blízkým přátelům za pomoc a podporu během studia.

Abstrakt

Tato diplomová práce se zabývá porovnáním nástrojů SoapUI a Postman pro testování rozhraní REST API. Práce se nejdříve zaobírá testováním REST API obecně. Dále práce představuje vybrané nástroje pro testování, konkrétně tedy SoapUI a Postman. Práce analyzuje možnosti použití nástrojů pro testování REST API. K porovnání nástrojů se zde stanovují kritéria, na jejichž základě poté samotné porovnání probíhá. Díky užití těchto stanovených kritérií se v práci vybírá vhodný nástroj pro názornou ukázku užití při testování reálné aplikace. Dokumentace této ukázky tvoří taktéž pomocnou příručku pro další užití k testování jiných aplikací.

K dosažení cílů práce se užívají metody popisu a explanace. Tyto metody umožňují jak uvedení do tématu testování REST API, tak i samotné představení zvolených nástrojů pro testování REST API. Jednou z hlavních užitých metod je multikriteriální analýza, která slouží k hodnocení zvolených nástrojů, tj. SoapUI a Postman. Dalšími užitými metodami jsou komparace a následná syntéza zjištěných poznatků, což slouží především v praktické části k demonstraci vhodného nástroje pro účely testování REST API.

Výsledkem práce je především porovnání dvou vybraných nástrojů pro testování rozhraní REST API, jmenovitě nástrojů SoapUI a Postman. Dílčími výsledky práce je tedy představení nástrojů SoapUI a Postman, analýza možností užití nástrojů pro testování REST API, a následná definice kritérií k porovnání, která jsou užita pro volbu vhodného nástroje pro praktické užití vybraného nástroje pro testování na reálné aplikaci. Toto aktuální představení oblasti testování REST API a daných nástrojů může taktéž sloužit jako návod k volbě vhodného nástroje pro testování REST API.

Klíčová slova

Postman, REST API, SoapUI, testování softwaru.

Abstract

This thesis deals with the comparison of SoapUI and Postman tools for testing the REST API. The thesis first deals with testing of REST API in general. Furthermore, the work presents selected tools for testing, namely SoapUI and Postman. The thesis analyses possibilities of using tools for the purpose of testing REST API. To compare the tools, the criteria are set in the thesis. Based on the criteria, the comparison itself takes place. Thanks to the use of these criteria, a suitable tool for illustrative use in testing a real application is selected. The documentation for this demo is also an auxiliary guide for further use in testing of other applications.

The methods of description and explanation are used to achieve the goals of the thesis. These methods allow both the introduction of the REST API testing topic and the introduction of the selected REST API testing tools. One of the main methods used is

multicriterial analysis, which is used to evaluate selected tools, that is SoapUI and Postman. Other methods used are comparison and subsequent synthesis of findings, which are used mainly in the practical part to demonstrate a suitable tool for the purposes of testing the REST API.

The result of this thesis is mainly a comparison of two selected tools for testing the REST API, namely SoapUI and Postman. Partial results of the work are the introduction of SoapUI and Postman products, analysis of the use of tools for testing REST API, and subsequent definition of criteria for comparison, which are then used to select a suitable tool for demo. In the demo, the selected tool for testing is used on a real application. This up-to-date presentation of the REST API testing area and tools can also serve as a guide for choosing the right testing tool.

Keywords

Postman, REST API, SoapUI, software testing.

Obsah

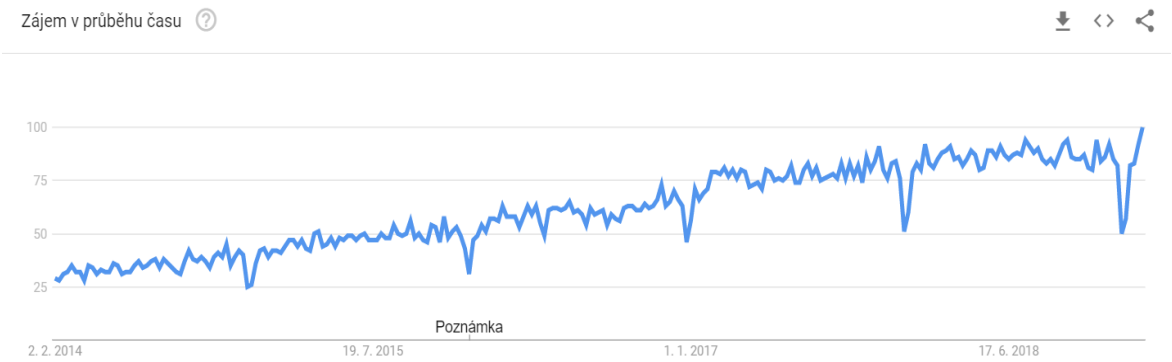
Úvod 8
Cíle práce 9
Cílová skupina 11
Použité metody 11
Výstupy a přínosy práce 12
Struktura práce 12
1 Rešerše zdrojů 13
1.1 Elektronické informační zdroje VŠE 13
1.2 Theses.cz 15
1.3 Google Scholar 17
1.4 Závěr rešerše 19
2 Testování REST API 20
2.1 Představení testování softwaru 20
2.2 Představení REST API 21
2.3 Možnosti testování REST API 26
2.4 Nástroje pro testování REST API 29
3 Kritéria pro porovnání nástrojů 31
3.1 Pomocné zdroje pro definici kritérií 31
3.2 Definice kritérií 32
3.3 Stanovení vah ke kritériím 37
3.4 Hodnocení kritérií 38
3.5 Tabulka pro aplikaci kritérií 39
4 Nástroj SoapUI 40
4.1 Základní představení SoapUI 40
4.2 Produkty SoapUI a jejich rozšíření 40
4.3 Možnosti nástroje SoapUI 44
4.4 Testování REST API pomocí SoapUI 45
5 Nástroj Postman 51
5.1 Základní představení Postman 51
5.2 Produkty Postman a jejich rozšíření 52
5.3 Možnosti nástroje Postman 53
5.4 Testování REST API pomocí Postman 55

6 Porovnání nástrojů SoapUI a Postman 58
6.1 Hodnocení nástroje SoapUI 58
6.2 Hodnocení nástroje Postman 62
6.3 Porovnání nástrojů 66
6.4 Výsledek porovnání nástrojů 68
7 Užití vybraného nástroje pro testování konkrétní aplikace 69
7.1 Výběr vhodné aplikace pro testování REST API 69
7.2 Manuální testování pomocí vybraného nástroje 71
7.3 Automatizace testování pomocí vybraného nástroje 74
7.4 Zhodnocení vybraného nástroje při testování 77
Závěr 78
Použitá literatura 80
Seznam obrázků 84
Přílohy I
Příloha A: Stavové kódy HTTP I
Příloha B: Ukázkový dokument WADL V
Příloha C: Ukázkový dokument WSDL VII

Úvod

V dnešní době není pravděpodobně potřeba zmiňovat, že moderní technologie se rozvíjí nezadržitelným tempem. Veškeré publikované poznatky mohou být tedy obzvláště v tomto oboru zanedlouho neaktuálními či dokonce irelevantními. Nejinak je tomu i v oblasti testování, které je dnes již nedílnou součástí při vývoji softwaru či práci s daty.

Testování jako nedílná součást vývoje softwaru, se samozřejmě týká i oblasti práce s daty. I při práci s daty se uživatelé mohou setkat s veřejnými rozhraními neboli API, tj. rozhraními pro programování aplikací. Toto rozhraní je potřeba definovat, pokud ho ovšem nedefinuje REST, což je architektura navržená pro distribuované prostředí. Tato architektura rozhraní, která je blíže definována v kapitole 2, nabírá v dnešní době na popularitě, jak je vidět z Obr. 1, který zachycuje trend za posledních 5 let (dle dat od společnosti Google).



Obr. 1 - Trend termínu REST API za 5 let [1]

Oblast REST API, která je primárně zaměřena na data, pracuje s informacemi ze serveru pomocí HTTP volání [2]. Testování této oblasti je důležité, jelikož kvalita a korektnost dat může být pro firmy velmi důležitá.

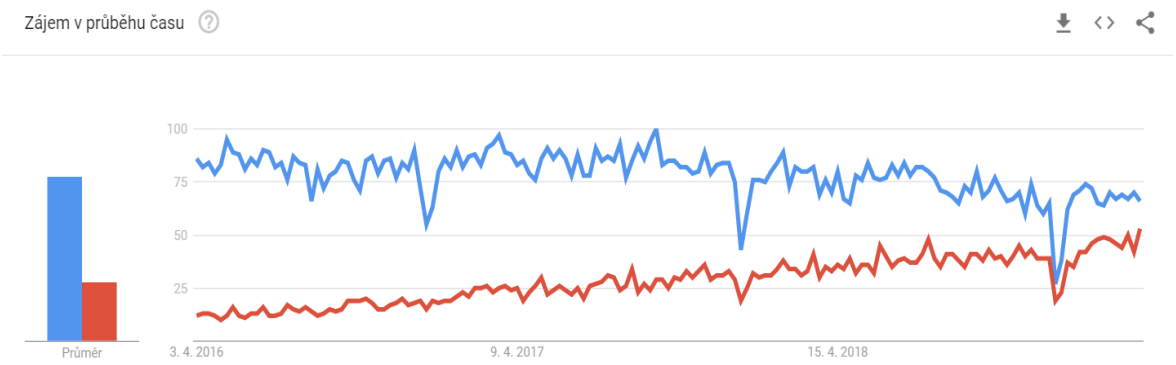
Vzhledem k popularitě REST API je i nástrojů pro testování tohoto rozhraní velká řada. Nástroje pro testování REST API jsou k dispozici jak bezplatné, tak placené. Taktéž jsou některé nástroje vhodné pouze pro manuální testování, zatímco jiné umožňují i automatizaci testování. Více k těmto nástrojům je uvedeno v kapitole 2.

Pro účely této práce byly vybrány 2 známé nástroje v tomto odvětví, které jsou v základní verzi oba bezplatné a umožňují oba typy testování, tj. jak manuální, tak automatizované. Jedná se o nástroje SoapUI a Postman. Ačkoliv se jedná o populární nástroje v oblasti testování rozhraní REST API, aktuální srovnání mezi nimi chybí, a to je tedy primárním cílem této práce.

Nástroj SoapUI je nejvíce používaným testovacím nástrojem pro REST API na trhu [3]. Jeho podrobné představení je v kapitole 4. Tento nástroj je na trhu již od roku 2006, kdy vstoupil na trh jako první bezplatný nástroj pro testování API. V dnešní době ho užívají miliony uživatelů po celém světě [3].

Nástroj Postman je znatelně mladším nástrojem pro testování REST API, který vyšel poprvé na trh koncem roku 2012. V dnešní době ovšem již nabízí větší nabídku produktů, nejen bezplatných, ale i zpoplatněných, které poskytují pokročilejší funkcionality. Popularita tohoto nástroje, který má přes 5 milionů uživatelů po celém světě, je ovšem taktéž veliká [4]. Více k tomuto nástroji je zmíněno v kapitole 5.

Popularita SoapUI, jakožto nejpoužívanějšího nástroje pro testování rozhraní REST API v posledních 2 letech spíše stagnovala, zatímco Postman, jakožto nástroj o 6 let mladší, na popularitě prozatím nabírá a stále více se přibližuje zmíněnému leaderu trhu. To je nejlépe patrné z dat společnosti Google za poslední 3 roky, která jsou zachycena zde na Obr. 2.



Obr. 2 - Trend nástrojů SoapUI a Postman za 3 roky [5]

Právě srovnání těchto populárních testovacích nástrojů REST API je primárním cílem této práce a je zachyceno v kapitole 6. Toto srovnání, a kritéria pro ono srovnání stanovená v kapitole 3, jsou dále užita k volbě vhodného nástroje a zachycení testování na reálné aplikaci v kapitole 7.

Cíle práce

Hlavním cílem diplomové práce je porovnání nástrojů SoapUI a Postman pro testování rozhraní REST API. Tento cíl je naplněn celou touto prací, především poté kapitolou 6, která využívá poznatků z předchozích kapitol práce.

Hlavní cíl je možné rozložit do následujících dílčích cílů:

DC 1 – Představení nástrojů SoapUI a Postman

Prvním dílčím cílem je představení těchto dvou nástrojů. Ačkoliv tento cíl již byl částečně naplněn i jinými pracemi, jeho aktualizace je zcela jistě něčím, co může být mnohým nápomocné. Konkrétně u nástroje Postman, který je znatelně novější a funkcionality přibývají i v poslední době, je představení jeho možností něco, co prozatím chybí. V neposlední řadě je toto aktuální představení i předpokladem pro naplnění primárního cíle.

DC 2 – Analýza možností použití nástrojů pro testování REST API

Druhým dílčím cílem je analýza možných užití obou nástrojů pro testování REST API. Součástí analýzy budou jak možnosti manuálního testování, tak testování automatizovaného. Užití nástrojů pro manuální testování je mezi uživateli nástrojů poměrně známé, ovšem možnosti automatizace jsou často opomíjené a na první pohled ukryté. Obzvláště v případě nástroje Postman se jedná často o něco, o čem uživatel bez studování dokumentace nemusí ani vědět.

DC 3 – Definice kritérií pro porovnání nástrojů

Třetím dílčím cílem je definice kritérií pro porovnání nástrojů. K porovnání nástrojů je potřeba zvolit vhodná kritéria specifická pro oblast REST API a dané nástroje. K jejich definici je potřeba využít pomocných zdrojů, na jejichž základě mohou být poté stanovena konkrétní kritéria a způsob rozdělení vah i jejich hodnocení. Mimo pomocné zdroje bude k definici kritérií nápomocna i praxe v oboru autora práce.

DC 4 – Užití definovaných kritérií k volbě vhodného nástroje

Pro ověření stanovené metodiky k porovnání nástrojů pro testování REST API a získaných poznatků v této práci, je stanoven čtvrtý dílčí cíl. Ke splnění tohoto cíle je potřeba aplikovat definovanou teorii a kritéria na výběr vhodného nástroje pro testování REST API. Užitím metodiky by mělo dojít k identifikaci vhodného testovacího nástroje pro manuální i automatizované testování rozhraní REST API.

DC 5 – Užití vybraného nástroje pro testování konkrétní aplikace

Posledním dílčím cílem je samotné testování vybraného produktu, a to pomocí testovacího nástroje, který vzejde z naplnění čtvrtého dílčího cíle. K dosažení tohoto cíle bude provedena základní ukázka manuálního a automatizovaného testování vybraného produktu, na čemž budou demonstrovány především silné stránky vybraného nástroje. Nedílnou součástí je dokumentace postupu testování a zhodnocení testovacího nástroje.

Cílová skupina

Cílovou skupinou této práce jsou všichni lidé se zájmem o moderní technologie. Práce je ovšem tematicky bližší lidem, kteří se pohybují v oblasti testování. Základní pojetí o testování je předpokladem k porozumění celé práci, ačkoliv některé pojmy budou vysvětleny přímo v práci. Zájemci o testování REST API jsou další skupinou, pro kterou by práce mohla být přínosem. Tématika REST API i jeho testování je ovšem v práci přiblížena, takže znalosti v této oblasti nejsou nutností pro porozumění práci. Pro ty, kterým je i tato oblast známá, může práce poskytnout informace o aktuálních možnostech, popř. základ pro podrobnější zpracování tématu či tématu podobného.

Použité metody

K dosažení cílů této práce je použito více metod, které jsou rozděleny na základě dílčích cílů, ke kterým jsou v této práci použity.

U prvního dílčího cíle se jedná především o metody popisu a explanace, které jsou užity pro představení zvolených testovacích nástrojů.

Stejné metody, tj. metody popisu a explanace, jsou použity i při dosažení druhého dílčího cíle, tzn. při analýze možností použití těchto nástrojů pro testování REST API. Tyto metody jsou rozšířeny ještě o metodu analýzy těchto možností, a následnou syntézu poznatků, která slouží nejen k naplnění tohoto cíle, ale i pro splnění čtvrtého dílčího cíle.

K naplnění třetího dílčího cíle slouží především metoda multikriteriální analýzy, díky které jsou definována kritéria pro porovnání těchto nástrojů. Pomocnou metodou je zde i syntéza analyzovaných zdrojů sloužící k definici konkrétních kritérií.

Ke splnění čtvrtého dílčího cíle jsou primárními metodami komparace a následná syntéza zjištěných poznatků, a to při užití kritérií pro selekci vhodného nástroje pro testování REST API.

Poslední, tj. pátý, dílčí cíl je naplněn pomocí metody demonstrace, kdy je zvolený nástroj pro testování REST API, který je vybrán na základě definovaných kritérií dříve v práci, aplikován při reálném užití při testování aplikace.

Výstupy a přínosy práce

Výstupy práce jsou analýza jednotlivých nástrojů pro testování REST API, definice kritérií pro porovnání nástrojů, porovnání těchto nástrojů, prezentace jejich výhod a nevýhod, a představení možností užití nástrojů pro manuální i automatizované testování REST API.

Zmíněná analýza nástrojů a kritéria definovaná touto prací mohou sloužit jako návod pro volbu vhodného nástroje a demo v praktické části poté jako základní příručka pro další využití k manuálnímu i automatizovanému testování jiných aplikací.

Přínosem práce je také aktuální představení oblastí, které testování REST API zahrnuje, a poskytnutí obecného postupu pro testování REST API. Celá práce jako celek může být podpůrným dokumentem pro podrobnější zpracování problematiky či jejích dílčích částí.

Struktura práce

Práce je strukturována do číslovaných kapitol a podkapitol, a mimo standardní kapitoly, které jsou součástí akademických prací, se jedná o 6 hlavních kapitol.

V kapitole 2, Testování REST API, je představeno REST API včetně možností testování této oblasti a přehledu nástrojů pro testování.

V kapitole 3, Kritéria pro porovnání nástrojů, jsou definované zdroje i samotná kritéria k porovnání nástrojů a metodika jejich hodnocení.

V kapitole 4, Nástroj SoapUI, je představen nástroj SoapUI, jednotlivé produkty, jeho možnosti, a testování REST API pomocí tohoto nástroje.

V kapitole 5, Nástroj Postman, je představen nástroj Postman, kde jsou analogicky informace o jednotlivých produktech, možnostech, a testování REST API pomocí daného nástroje.

Kapitola šestá, Porovnání nástrojů SoapUI a Postman, poté hodnotí oba zmíněné nástroje, navzájem je srovnává za užití metodiky definované ve třetí kapitole, a prezentuje výsledky porovnání těchto nástrojů.

Kapitola sedmá, Užití vybraného nástroje pro testování konkrétní aplikace, je kapitolou, která vychází z naplnění předchozí kapitoly a zahrnuje ukázku testování konkrétní aplikace z praxe včetně dokumentace testování a zhodnocení daného testovacího nástroje.

1 Rešerše zdrojů

Pro rešerši zdrojů jsem na základě zvolených klíčových slov vyhledával odborné zdroje zabývající se podobnou tématikou. Klíčová slova, která nebyla stejná v českém a anglickém jazyce, byla vyhledávána v obou jazycích.

Pro rešerši byla zvolena následující klíčová slova:

- REST API
- REST API testování / REST API testing
- SoapUI testing / SoapUI testování
- Postman API testing / Postman API testování
- Comparison of SoapUI and Postman / Srovnání SoapUI a Postman
- Comparison of testing tools / Srovnání testovacích nástrojů

Odborné zdroje, které byly k vyhledávání použity jsou tyto:

- Google Scholar – volně přístupný webový vyhledávač, který byl spuštěn roku 2004 společností Google. Indexuje text i metadata odborné literatury napříč obory i formáty [6].
- Theses.cz – národní registr závěrečných prací, který je využívaný vysokými školami v ČR i zahraničí. Mimo jiné slouží i jako systém k odhalování plagiátů [7].
- Elektronické informační zdroje VŠE – vyhledávač poskytující přehled elektronických informačních zdrojů dostupných na VŠE

1.1 Elektronické informační zdroje VŠE

V elektronických informačních zdrojích VŠE bylo dosaženo těchto výsledků dle jednotlivých klíčových slov.

REST API

Pro tento termín bylo nalezeno velké množství výsledků, přes 70 tisíc, ovšem relevantní k tématu práce zde nebyly výsledky žádné. I v případě omezení výsledků na závěrečné

práce, kdy bylo výsledků kolem 5 set, výsledky nekorespondovaly s tématem této práce ani okrajově.

REST API testing

Anglický termín REST API testing zobrazil ve výsledcích 17 tisíc položek, ovšem ačkoliv některé pojednávaly o SoapUI, jednalo se buď o nepatrnou zmínku či příliš specifické zaměření, např. bezpečnostní testování. Světlou výjimkou byl článek Tobiase Fertiga, a to *Model-driven Testing of RESTful APIs* [8]. Tento článek, který byl publikován jako součást záznamu z konference roku 2015, zmiňuje různé aspekty automatizovaného testování REST API. Převážná část práce se ovšem zaměřuje na tvorbu SW generátoru pro testovací případy.

REST API testování

Česká verze termínu již ovšem žádné ani okrajově relevantní zdroje ve výsledcích nezobrazila.

SoapUI testing

Anglický výraz SoapUI testing získal kolem 200 výsledků. Mnoho článků bylo opět příliš specificky zaměřených, aby souvisely s tématem této práce, ovšem publikace *Web services testing with soapUI: build high quality service-oriented solutions by learning easy and efficient web services testing with this practical, hands-on guide* [9] od Charithy Kankanamge obsahuje informace, které ač z roku 2012 mají podobnou tématiku a zabývají se nástrojem SoapUI, který je nedílnou součástí této práce. Dále se ve výsledcích nachází i studie *Web Service Testing Tools: A Comparative Study* [10]. Tato studie z roku 2013 nabízí srovnání více nástrojů, ovšem zaměřuje se na výkonnostní testování, takže souvisí s prací spíše okrajově.

SoapUI testování

Česká verze pojmu zobrazila pouze jeden výsledek, a tím je publikace od Charithy Kankanamge, která byla získána již při vyhledávání pomocí anglického synonyma.

Postman API testing

Tento výraz zobrazil kolem 70 výsledků, převážně se jednalo o články, které byly irelevantní k testování i tématu této práce. Jedinou výjimkou byl článek *API Evangelist: Postman Collections Will Take Your API Productivity To The Next Level* [11]. Tento krátký článek zmiňuje nástroj Postman, ovšem velmi povrchně, a pro účely této práce zcela nedostatečně.

Postman API testování

Český ekvivalent výrazu již nepřinesl žádný relevantní výsledek.

Comparison of SoapUI and Postman

V případě anglického výrazu byl jedinou zmínkou ve výsledcích článek, který reportoval novinky na trhu v roce 2015, kde se nevyskytovaly žádné užitečné informace.

Srovnání SoapUI a Postman

V případě českého termínu srovnávajícího tyto dva nástroje také nebylo dosaženo žádných výsledků.

Comparison of testing tools

Anglická verze tohoto termínu přinesla o poznání více výsledků, konkrétně kolem 600 tisíc. Jednalo se ovšem o irelevantní výsledky spojené pouze z jednotlivými slovy výrazu.

Srovnání testovacích nástrojů

Tento český termín získal pouze jeden a k tomu irelevantní výsledek.

1.2 Theses.cz

Ve vyhledávání portálu Theses.cz byly níže uvedené výsledky, rozdělené dle užitých klíčových slov.

REST API

Pojem REST API získal 800 výsledků, ovšem převážně zabývajícím se vývojem a návrhem REST architektury, tj. něčím pro účely této práce irelevantním.

REST API testing

Taktéž anglický verze pojmu REST API testing získala kolem 800 výsledků, ovšem tématika článků, potažmo jejich relevantnost odpovídala předchozímu termínu.

REST API testování

Česká verze termínu zobrazila ve vyhledávání 700 výsledků, ovšem jednalo se o výsledky irelevantní, či příliš úzce zaměřené, jakož například zátěžové testování REST API.

SoapUI testing

Anglický termín SoapUI testing získal 40 výsledků, kde se jako částečně relevantní k tématu této práce jeví především diplomová práce *Testování webových služeb nástrojem SoapUI* [12] od Petra Sobotky z roku 2015, která vytváří metodiku pro testování pomocí tohoto nástroje. Další prací je ověření této metodiky v diplomové práci *Ověření metodiky Testování webových služeb nástrojem SoapUI* [13] Radky Jirmusové z roku 2016. Poslední prací, která souvisí do jisté míry s touto prací je diplomová práce *Systémové integrační testování s použitím SoapUI* [14] od Jakuba Petržilky z roku 2015.

SoapUI testování

Česká verze SoapUI testování získala podobný počet výsledků i stejné práce jako anglický ekvivalent. Zbytek prací není k tématu práce nikterak relevantní.

Postman API testing

Tento výraz zobrazil 20 výsledků, z nichž žádný nebyl spojen s tímto tématem.

Postman API testování

Český ekvivalent dosáhl identický výsledků, tedy žádný relevantní zdroj nebyl nalezen.

Comparison of SoapUI and Postman

Tento anglický termín nepřinesl žádné relevantní výsledky.

Srovnání SoapUI a Postman

Český ekvivalent výrazu přinesl pouze jeden, a to irelevantní výsledek nezabývající se tématem.

Comparison of testing tools

Anglická verze termínu nezobrazila žádné relevantní výsledky.

Srovnání testovacích nástrojů

Česká verze termínu přinesla kolem 1000 výsledků, kde jako relevantní k tématu mohou býti jmenovány především tři bakalářské práce. Jedná se o práci Michaela Žabky *Analýza software testing nástrojů* [15]. Tato práce se zaobírá srovnáním testovacích nástrojů, a především pro stanovení kritérií a vah může být vhodným pomocným zdrojem této práce. Nejinak je tomu v případě práce Dominika Bláhy *Srovnání Open Source nástrojů pro správu testů využitelných při vývoji softwaru jednotlivci* [16]. I tato práce může sloužit jako pomocný zdroj při vypracování této práce, konkrétně stanovení kritérií a vah. Poslední částečně relevantní prací je *Srovnání nástrojů pro správu defektů* [17] od Martina Kulíčka. Tato práce řeší také srovnání, ale díky specializaci na nástroje pro správu defektů je relevantnost nižší než u předchozích dvou prací.

1.3 Google Scholar

Třetím vyhledávačem užitým k nalezení zdrojů je portál Google Scholar. Výsledky dle klíčových slov jsou opět níže.

REST API

V případě výrazu REST API bylo nalezeno velké množství výsledků, konkrétně kolem 460 tisíc, ovšem převážně zaměřující se na návrh REST API či kompletně irelevantní. Pro účely této práce tedy žádné relevantní výsledky nalezeny nebyly.

REST API testing

Anglický výraz REST API testing vrátil přes 120 tisíc výsledků, ovšem tematicky opět buď kompletně irelevantní či zaměřující se na návrh REST API.

REST API testování

Česká verze termínu na tom byla o trochu lépe. V 1500 výsledků byly i práce, které byly částečně relevantní pro tématiku této práce. Jedná se především o práci *Testování REST API* [18] z roku 2012, kde se v teoretické části zmiňuje REST API a testování, než se přejde k návrhu a implementaci vlastní aplikace. Dále ve výsledcích byl článek o tvorbě vlastní aplikace pro testování REST API, který se nazývá *Rozšířené testování REST API* [19]. Tento článek z roku 2014 ovšem již souvisí s tématem jen velmi okrajově.

SoapUI testing

Anglický výraz SoapUI testing zobrazil kolem 1200 výsledků, jedná se o publikaci [9] od Kankanamge, která se vyskytovala již v elektronických informačních zdrojích VŠE. Dále se zde objevuje také článek *Web service testing tool soapui and its analysis* [20], kde se vyskytují údaje k SoapUI, ovšem je již z roku 2010, tj. nemůže být považován za aktuální. Dalším článkem, který se vyskytuje ve výsledcích je *Web services testing challenges and approaches* [21]. Tento článek pojednává o přístupu k testování webových služeb a zmiňuje některé faktory testování i nástroj SoapUI.

SoapUI testování

Česká verze výrazu již přináší pouze 60 výsledků a relevantní žádný. Převážně zdroje informují jen o testování obecně a obsahují krátkou zmínku o SoapUI.

Postman API testing

Tento výraz dosáhl 1000 výsledků, ovšem téměř bez zmínky o nástroji Postman. Většina zdrojů zmiňovala testování obecně či byla kompletně irelevantní.

Postman API testování

Česká verze klíčového výrazu získala kolem 30 výsledků, ovšem zdroje obsahovali maximálně zmínku o nástroji Postman, tj. pro účely této práce irelevantní.

Comparison of SoapUI and Postman

Tento výraz získal 18 výsledků, kdy jeden z nich byl okrajově relevantní. Jedná se o kapitolu z knihy, konkrétně *Testing REST APIs* [22]. Tato kapitola hovoří o nástrojích k testování API, ovšem pouze povrchově.

Srovnání SoapUI a Postman

Česká verze výrazu získala 18 výsledků, ovšem ani jeden relevantní.

Comparison of testing tools

Tento výraz získal miliony výsledků, ovšem pravděpodobně díky jednotlivým částem výrazu, jelikož ani nejrelevantnější z výsledků nesouvisely s touto tématikou.

Srovnání testovacích nástrojů

Český výraz získal kolem 13 tisíc výsledků, ovšem žádný, který by byl relevantní k této práci.

1.4 Závěr rešerše

Ačkoliv některé výsledky rešerše s tématem práce souvisejí, přímo tématem této práce se žádný z nalezených zdrojů dosud nezabýval. Příbuzná témata, která řešila srovnávání testovacích nástrojů mohou sloužit jako pomocné zdroje při stanovení kritérií a vah kritérií v této práci. Jmenovitě z rešerše se jedná o práci Dominika Bláhy [16] či Michaela Žabky [15].

Z hlediska představení jednotlivých nástrojů může být poté pomocným zdrojem publikace od autorky Kankanamge [9], která se zabývá SoapUI. Pro účely představení nástroje Postman je díky nedostatku zpracovaných prací primárním zdrojem samotná dokumentace nástroje.

2 Testování REST API

Pro představení oblasti testování REST API, je zapotřebí si nejdříve představit samotnou oblast testování softwaru. S narůstající popularitou této oblasti se již tento pojem v oblasti IT stává známějším, takže toto téma první podkapitoly poskytne pouze stručný náhled do tématu.

Po úvodu do tématu testování, je již možné se zaměřit na představení oblasti rozhraní REST API, jež má svá specifika a pro další práci s ním je zapotřebí ji definovat. Toto představení je nedílnou součástí práce, které poskytuje základ k podrobnějšímu rozebírání problematiky.

Uvedení do problematiky testování i oblasti rozhraní REST API, je základem další podkapitoly, která se již věnuje možnostem testování právě tohoto rozhraní. Stejně jako u jiných témat souvisejících s oblastí testování, i zde se jedná o informace týkající se jak manuálního, tak i automatizovaného testování tohoto rozhraní.

Pro testování REST API samozřejmě existuje celá řada nástrojů a jejich základní rozdělení a příklady firem i nástrojů jako takových, jsou demonstrovány v poslední podkapitole. Pro zvolené nástroje, které práce porovnává, je samozřejmě k dispozici více informací v následujících kapitolách práce, tj. ve 4. a 5. kapitole.

2.1 Představení testování softwaru

Testování softwaru je dnes již nedílnou součástí vývoje softwaru a pojmem, který již vstoupil do povědomí mnoha lidí. O testování softwaru, jeho definici i rozdělení existuje velká řada publikací. Tato podkapitola tedy jen základně zavede do problematiky testování a toho, co to obnáší.

Definicí testování softwaru je velká řada, jedním z příkladů je například definice z publikace *Guide to the Software Engineering Body of Knowledge* od Alaina Abrana a Jamese W. Moora. Ta praví, že testování je aktivita prováděná za účelem ohodnocení kvality produktu, sloužící k jeho zlepšení pomocí identifikace defektů a problémů. Testování softwaru se skládá z dynamického ověření chování programu na omezené sadě vhodně zvolených testovacích případů, a to z obvykle nekonečné domény exekucí oproti očekávanému chování. [23]

Kategorií testů je velké množství, jejichž výčet a definice by vydaly na samostatnou práci. Mezi základní kategorie patří funkční testy, testy použitelnosti, testy spolehlivosti, výkonnostní testy, testy podpory či bezpečnostní testy. I tyto kategorie mohou být dále děleny do více podkategorií, které ovšem nejsou předmětem této práce. Pro účely této

práce je důležité ještě rozdělení na testy automatické a manuální. Toto rozdělení, jak již názvy napovídají, rozlišuje testy dle toho, zda jsou prováděny člověkem či programem. Pro opakované testy, stejně jako například testy zátěžové, je vhodnější využít automatizace, zatímco manuální testování je doménou testování, které je prováděno sporadicky či vyžaduje lidské posouzení. Faktorem je samozřejmě i finanční náročnost, popřípadě úspora, kterou s sebou daný způsob testování přináší. [24]

2.2 Představení REST API

Pro představení REST API je zapotřebí si nejdříve definovat architekturu REST[1]. Pro uvedení do tohoto tématu je pravděpodobně nejvhodnějším zdrojem disertační práce *Architectural Styles and the Design of Network-based Software Architectures* [2] z roku 2000, jejímž autorem je Roy Fielding, spoluautor protokolu HTTP[2].

REST je styl architektury pro distribuované systémy hypermédií. Jedná se o hybridní styl derivovaný z několika stylů síťové architektury, který je kombinován s dodatečnými omezeními, které definují jednotné rozhraní konektoru. [2]

Architektura REST má 6 základních principů [2], kterými se řídí. Konkrétně se jedná o tyto principy:

1. **Klient-Server**
   Prvním prvkem tohoto hybridního stylu je styl architektury klient-server. Oddělením uživatelského rozhraní od datového úložiště se zlepšuje přenositelnost uživatelského rozhraní napříč platformami a zlepšuje se škálovatelnost pomocí zjednodušení serverových komponent. Pravděpodobně nejpodstatnější z hlediska webu je ovšem to, že separace umožňuje komponentům nezávislý vývoj, čímž plní požadavky pro několik organizačních domén.
2. **Bezstavovost**
   Dalším principem, který slouží pro interakci mezi klientem a serverem, je to, že komunikace musí být bez stavu, tj. každý požadavek z klienta na server musí obsahovat veškeré nezbytné informace k jeho porozumění, a nemůže využívat žádného uloženého obsahu na serveru. Stav relace je tedy ukládán na klientu. Toto omezení zlepšuje vlastnosti viditelnosti, spolehlivosti a škálovatelnosti. Viditelnost je zlepšená díky tomu, že monitorovací systém nemusí řešit více nežli jednotlivý požadavek, aby chápal celý jeho kontext. Spolehlivost je vylepšena tím, že zjednodušuje úlohu obnovy z částečných selhání. Škálovatelnost je vylepšena tím, že server nemusí ukládat stav požadavků a může tedy rychle uvolňovat prostředky,

[1] REST – Representational State Transfer, tj. reprezentativní přenos stavu
[2] HTTP – Hypertext Transfer Protocol, tj. protokol pro přenos hypertextu

a nadále zjednodušovat implementaci, díky tomu, že nemusí nijak řídit prostředky mezi požadavky. Na druhou stranu nevýhodou tohoto omezení je, že se může snížit výkon sítě, pokud jsou posílána opakující se data v sérii požadavků, která nemohou být sdílena serverem. Taktéž díky tomu server nemá kontrolu nad konzistentním chování aplikace, jelikož je závislý na správné implementaci sémantiky napříč několika klientskými verzemi.

3. **Mezipaměť**

   Pro vylepšení efektivity sítě je jedním z omezení i omezení mezipaměti. Tato omezení vyžadují, aby data z odpovědi na požadavek byla označena implicitně či explicitně, dle toho, zda se mohou ukládat do mezipaměti či nikoliv. Pokud je odpověď označena tak, že se tam data ukládat mohou, klientská mezipaměť má možnost uložit data z odpovědi na později pro použití u ekvivalentních požadavků. Výhodou je zde to, že tato omezení mohou částečně či kompletně eliminovat některé interakce, zlepšit efektivitu, škálovatelnost, a výkonnost zmenšením průměrné latence série interakcí. Nevýhodou je ovšem to, že mezipaměť může snížit spolehlivost v případě, že se uložená data výrazně liší oproti datům, která by byla jinak získána ze serveru při přímém zaslání požadavku.

4. **Jednotné rozhraní**

   Hlavní vlastností, která odlišuje styl architektury REST od ostatních síťových stylů, je důraz na jednotné rozhraní mezi komponentami. Pomocí užití principu obecnosti ze softwarového inženýrství na rozhraní komponent, je celá architektura systému zjednodušena a viditelnost interakcí je vylepšena. Implementace jsou odděleny od služeb, které poskytují, což umožňuje nezávislý vývoj. Nevýhodou zde je to, že jednotné rozhraní zhoršuje efektivitu, jelikož informace je přenášena ve standardizované formě namísto formy specifické pro dané potřeby aplikace. Rozhraní REST je navržené tak, aby bylo efektivní pro velkoplošný datový přenos hypermédií, optimalizované pro běžné případy webu, ale to vede k tomu, že rozhraní není optimální pro ostatní formy interakcí architektury. Pro dosažení jednotného rozhraní je zapotřebí několik omezení architektury, které řídí chování komponent. REST je definován 4 omezeními rozhraní, tj. identifikací zdrojů, manipulací se zdroji pomocí reprezentace, samo-popisnými zprávami, a hypermédii, jakožto zdrojem aplikačního stavu.

5. **Vrstvený systém**

   Vrstvený systém umožňuje, aby architektura mohla být tvořena hierarchickými vrstvami pomocí omezení chování komponent, jako je například to, že žádná z komponent nevidí za vrstvu, s kterou interaguje. Pomocí omezení znalosti systému na jednu vrstvu, je architektura vázána na komplexitu systému a

umožňuje nezávislost komponent. Vrstvy mohou být použity pro zapouzdření starých služeb a ochranu nových služeb před starými klienty, což zjednodušuje komponenty díky přesunu málo používané funkcionality na sdíleného prostředníka. Prostředníci mohou být použiti pro zlepšení škálovatelnosti systému pomocí umožnění rozdělení zátěže služeb na více sítí a procesorů. Základním nedostatkem vrstvených systémů je to, že zhoršují latenci při zpracování dat, potažmo tedy výkon. Pro síťový systém, který podporuje omezení mezipaměti, může toto být kompenzováno benefity sdílené mezipaměti u prostředníků. Díky užití mezipaměti na hranici organizační domény se může značně zlepšit výkon. Tyto vrstvy umožňují bezpečnostní omezení při pohybu dat přes hranice domény, což je požadováno firewally.

6. **Kód na vyžádání**

   Poslední z omezení pro REST je kód na vyžádání. REST umožňuje rozšíření klientské funkcionality stažením a exekucí kódu ve formě appletů či skriptů. Toto zjednodušuje klienty díky redukování množství potřebných vlastností, které musí být naimplementovány. Umožnění stažení vlastností po nasazení zlepšuje rozšiřitelnost systému. Na druhou stranu to také zhoršuje viditelnost, a jedná se tedy o volitelné omezení architektury REST.

Druhou částí termínu REST API, je API[3]. Jedná se o rozhraní pro programování aplikací. Zjednodušeně se jedná o prostředníka, který umožňuje komunikaci mezi dvěma aplikacemi. Zároveň se jedná o vrstvu bezpečnosti, kdy druhá aplikace nedostává přístup ke všemu z aplikace, ale pouze k věcem přístupným skrze toto rozhraní. V případě webových API, jichž se REST API týká, se jedná o rozhraní, které spojuje webové aplikace přes komunikaci po internetu. Existuje více než 13 tisíc veřejných webových API, které mohou být použity pro různé účely. Mimo tato veřejná rozhraní existují i soukromá webová rozhraní, kterých jsou stovky tisíc, a které nejsou dostupné pro veřejné užití, a slouží například k rozšíření schopností a služeb společnostem. [25]

Zdroje jsou primární cestou, kterou klient interaguje s API. Zdroje v REST reprezentují buď typy objektů či primární brány k oblastem aplikace. Díky oddělené architektuře je možné použít více akcí na stejném zdroji. Zdroje jsou zpravidla pro jednoduchost pojmenované podstatnými jmény, například *user*. Až pomocí metod jsou pak specifikované činnosti, které jsou na zdroji prováděné. Každý zdroj by měl být

[3] API – Application Programming Interface, tj. rozhraní pro programování aplikací

identifikován pomocí URI[4]. Tento jednotný identifikátor zdroje je nejdůležitější a nejjednodušší prvek webové architektury. [9]

Pro přístup ke zdrojům používá REST API vlastní jazyk, který se skládá ze 4 základních příkazů. Jedná se o příkazy DELETE, GET, POST a PUT, což jsou metody protokolu HTTP. Pro přístup pomocí těchto příkazů je dále nutné znát URI neboli identifikátor zdroje. Odpovědí na tyto příkazy jsou poté v případě úspěchu HTTP kódy začínající 2, jako například 200, a v případě neúspěchu začínající 4, jako například 400. Konkrétní koncová čísla odpovědí poté charakterizují další detaily, jakožto důvod chyby či typ odpovědi. [2]

GET je základní metodou pro přístup k datům a pro její zavolání stačí znát pouze dané URI. Touto metodou tedy získáme data daného zdroje. Příkladem může být „*HTTP GET http://www.google.com/users/123*“. Odpovědí na tento příkaz by poté byl HTTP kód odpovědi s případným tělem zprávy obsahujícím obsah ve formě XML nebo JSON. Metoda GET pouze přistupuje ke zdrojovým datům, ovšem nikterak je nemění. Jako taková patří do skupiny bezpečných metod. Taktéž příkaz patří do metod idempotentních, tj. více identických žádostí musí vrátit pokaždé stejný výsledek, dokud jiné API (např. příkaz POST či PUT) nezmění stav zdroje na serveru. [26]

Pokud URI žádosti odkazuje na proces produkující data, tato produkovaná data budou vrácena jako entita odpovědi, a nikoliv zdrojový text procesu, pokud tedy tento text není výstupem procesu. Pro tento příkaz, ve chvíli, kdy je zdroj nalezen, musí server vrátit HTTP odpověď 200 (OK), společně s tělem odpovědi, kde je zpravidla obsah ve formátu XML či JSON. Pokud zdroj nalezen není, server vrátí HTTP odpověď 404 (NOT FOUND). Stejně tak v případě špatného formování žádosti vrátí server HTTP odpověď 400 (BAD REQUEST). [26]

Dalším možným příkazem je POST, který slouží k tvorbě dat. Při odeslání příkazu není známo přesné URI, jelikož daný zdroj bude tímto příkazem teprve vytvořen a v případě úspěšného volání bude nové URI vráceno v odpovědi. Příkaz se tedy zasílá na společný identifikátor. Příkladem zde může být „*HTTP POST http://www.google.com/users*“. [2]

V ideálním případě, kdy byl zdroj příkazem vytvořen, HTTP odpověď by měla mít kód 201 (CREATED) a obsahovat entitu popisující stav žádosti s odkazem na nový zdroj a hlavičkou lokace. Pokud akce nevytvoří zdroj, který by mohl být identifikovaný URI, může být odpovědí pouze 200 (OK) či 204 (NO CONTENT). Odpovědi na tuto metodu se nemohou ukládat do mezipaměti, pokud odpověď neobsahuje vhodné hlavičky Cache-

[4] URI - Uniform Resource Identifier, tj. jednotný identifikátor zdroje

Control a Expires. POST není metoda bezpečná ani idempotentní, tj. dva identické příkazy POST vytvoří dva různé zdroje se stejnými informacemi (kromě ID zdroje). [2]

Třetím možným příkazem je DELETE, který slouží ke smazání dat. Stejně jako příkaz GET i zde je příkaz volán s konkrétním URI, které jím bude smazáno. Příkladem budiž „*HTTP DELETE http://www.google.com/users/123*“. Při opakovaném volání bude vrácena negativní odpověď, jelikož zdroj již nebude existovat. [9]

Úspěšná odpověď na příkaz DELETE by měla být HTTP 200 (OK), pokud odpověď obsahuje entitu popisující stav, HTTP 202 (ACCEPTED), pokud akce byla zařazena do fronty, či HTTP 204 (NO CONTENT), pokud akce byla vykonána, ale odpověď neobsahuje entitu. DELETE operace jsou idempotentní. Pokud se zdroj smaže, je odstraněn z kolekce zdrojů. Opakované volání příkazu DELETE na daný zdroj výsledek nezmění, ovšem vrátí HTTP 404 (NOT FOUND), jelikož zdroj již byl smazán. [9]

Posledním ze základních příkazů je PUT, který slouží k aktualizaci zdroje. Na rozdíl od příkazu POST se provádí ovšem nad konkrétním zdrojem. V případě, že zdroj, na který odkazuje neexistuje, bude nově vytvořen. Příkladem příkazu je „*HTTP PUT http://www.google.com/users/123*“. Pokud je tímto příkazem zdroj nově vytvořen, musí dorazit HTTP odpověď 201 (CREATED), pokud je existující zdroj změněn, poté je očekáván HTTP kód 200 (OK) nebo 204 (NO CONTENT), které označují úspěšné splnění příkazu. [27]

V REST API jsou ještě důležitější než představené příkazy především stavové kódy HTTP, tj. odpovědi na tyto příkazy. Ty poskytují klientovi požadovaná data a informace o vykonání příkazů. Právě jejich otestování a ověření správných odpovědí je poté důležité k zaručení kvality REST API.

HTTP definuje 40 standardních stavových kódů, které poskytují výsledky klientova příkazu. Tyto kódy jsou rozděleny do 5 kategorií [2].

Tab. 1 - kategorie stavových kódů HTTP (data z práce Fieldinga [2], vlastní zpracování)

KATEGORIE	POPIS
1xx: Informační	Komunikuje informace na úrovni protokolového přenosu.
2xx: Úspěch	Indikuje úspěšné přijetí klientova příkazu.
3xx: Přesměrování	Indikuje, že klient musí provést další akci k úspěšnému splnění příkazu.
4xx: Chyby klienta	Tato kategorie poukazuje na chyby na straně klienta.
5xx: Chyba serveru	Tato kategorie zmiňuje chyby serveru.

Z těchto výše zmíněných kategorií je pouze vzorek konkrétních kódů, které se vztahují k problematice REST API, a tyto jsou vysvětleny níže v Příloha A: Stavové kódy HTTP.

Pro popis webových služeb REST slouží WADL[5], což je XML popis webových služeb založených na HTTP, který je čitelný počítačem [9]. Tento model zachycuje zdroje, které poskytuje služba i jejich vztah. Pomocí užití tohoto popisu lze zjednodušit znuvupoužitelnost webových služeb založených na architektuře REST [9]. Ukázkový WADL společnosti Amazon, který zahrnuje metody a parametry užívané při testování REST API je k dispozici v přílohách, konkrétně to je Příloha B: Ukázkový dokument WADL. Z tohoto dokumentu lze automaticky vygenerovat kód, kterým se dá služba REST zavolat.

Použit může být i WSDL[6], což je taktéž XML popis webových služeb ovšem uzpůsoben především na webové služby SOAP[7] [28]. Vzhledem k tomu, že není uzpůsoben pro účely REST může být použit sice jako pomocný dokument, ale pro účely REST je mnohem vhodnější WADL, což je jeho ekvivalent právě pro služby REST. Pro srovnání dokumentů je ukázkový WSDL k dispozici v přílohách, a je to konkrétně Příloha C: Ukázkový dokument WSDL. Z tohoto dokumentu lze vygenerovat kód pro zavolání služby SOAP.

2.3 Možnosti testování REST API

Pro testování REST API je možné rozdělit běžné přístupy na 4 kategorie, a to jednotkové testování, funkční testování webových služeb, integrační testování webových služeb a výkonnostní (či zátěžové) testování. Jednotlivé kategorie jsou rozebrány níže [9]:

- Jednotkové testování REST API funguje stejně jako jednotkové testování klasických počítačových programů. Každá operace musí být zvláště testována, aby se ověřilo, že splňuje požadavky. Tyto testy jsou zpravidla psané pomocí rozhraní jednotkových testů pro daný programovací jazyk. Zodpovědnou osobou za tyto testy je autor webové služby, který by měl pokrýt logiku operací webových služeb dostatečným počtem jednotkových testů.
- Druhou kategorií je funkční testování webových služeb. Toto testování nastává poté, co je webová služba již nasazena. Cílem funkčního testování je zajištění toho, že očekávaná byznysová funkcionalita je webovou službou opravdu poskytována. Pro funkční testování existuje více přístupů:

[5] WADL – Web Application Description Language, tj. jazyk pro popis webové aplikace
[6] WSDL – Web Services Description Language, tj. jazyk pro popis webových služeb
[7] SOAP – Simple Object Access Protocol, tj. protokol pro přístup k jednoduchým objektům

    - Testování pomocí nástrojů je jedním z přístupů funkčního testování REST API. Hlavním cílem je zde podpořit automatické generování a podávání požadavků na webovou službu. Vzhledem k tomu, že rozhraní webové služby je XML dokument čitelný počítačem, není jednoduché si dokument přečíst a tvořit testy manuálně. Z tohoto důvodu mohou nástroje být použity k odkazování na dokument, z kterého jsou korespondující požadavky generovány automaticky a mohou být testerem posílány bez dalších modifikací.
    - Použití klientského API, které poskytuje kontejner služby je další z možností funkčního testování REST API. V tomto případě není vyžadován žádný dodatečný nástroj pro testování REST API.
- Třetí kategorií je integrační testování webových služeb. Toto testování je za účelem ověření správné funkčnosti s dalšími komponentami. Může se jednat například o komunikaci s komponentou brokera, prostředníka, který transformuje případně špatně formulovaný požadavek na správný formát požadovaný službou.
- Čtvrtou kategorií je zátěžové testování, které ověřuje pomocí testovacích nástrojů, že mimo správné funkčnosti zvládá služba i větší zátěž a ověřuje výkon při různých podmínkách.

Ve srovnání s tradičními testovacími přístupy s sebou testování webových služeb nese i jistá specifika, kterým se musí testování uzpůsobit.

Tato specifika jsou především užití externích webových služeb, implikace spojené s užitím komplexních standardů a protokolů, a neexistující uživatelské rozhraní webových služeb. Tato specifika jsou představena níže [9]:

- Užití externích webových služeb je vzhledem ke škálovatelnosti a rozšiřitelnosti systému možné, a některé webové služby tedy mohou být vyvinuty a hostovány třetí stranou. Tyto služby mohou být dynamicky objevené a použité na základě byznysových požadavků. Ačkoliv to akceleruje dodání řešení, testování takového systému se stává komplexnějším, jelikož kvalita i dostupnost služeb třetích stran je mimo kontrolu testera.
- Dalším ze specifik je to, že webové služby mohou používat různé specifikace. Tester pro jejich správné otestování potřebuje znalost těchto standardů a konceptů, aby mohl službu efektivně otestovat. Díky tomu může v případě testování webových služeb testerovi déle trvat, než může s testováním začít. Webové služby také mohou být k dispozici přes několik přenosových protokolů. Testování tedy nemusí být limitováno pouze na otestování skrze HTTP, ale například i přes JMS[8] či VFS[9], což by vyžadovalo změny při nastavení testování, stejně jako jinou sadu testů.
- Pravděpodobně nejvíce očividným specifikem je rozdíl oproti tradičnímu testování webových aplikací, kde mohou být testovací scénáře identifikovány jednoduše pomocí zkoumání grafického uživatelského rozhraní komponent. Operace webových služeb jsou ovšem dostupné okolnímu světu pomocí kontraktů ve formě

[8] JMS – Java Message Service, tj. služba pro posílání zpráv Javou
[9] VFS – Virtual File System, tj. systém virtuálních souborů

čitelné počítačem (výše zmíněnými dokumenty WADL či WSDL). Pro derivaci testovacích scénářů musí tedy testeři použít těchto dokumentů, což je složitější ve srovnání se zkoumáním uživatelského rozhraní.

Testování REST API se liší od testování grafického uživatelského rozhraní také tím, že se zaměřuje na vrstvu byznysové logiky softwarové architektury. Nezaměřuje se tedy na prezentační vrstvu, tj. vzhled aplikace. Webové aplikace jsou založeny na modelu architektury o 3 vrstvách. Prezentační vrstva se stará o uživatelské rozhraní, logický vrstva (neboli byznysová) řeší právě API, zatímco datová vrstva je doménou databází. V ideálním případě by vrstvy neměly vědět nic o platformě, technologii, ani vzájemné struktuře. Vrstva byznysové logiky je komplexnější než zbylé vrstvy, a testování na této vrstvě se nazývá API testování. [29]

Namísto standardních vstupů a výstupů se v případě REST API posílají API volání, jejichž výstupem je odpověď systému. Jak již bylo zmíněno výše, k tomu vedou 2 cesty, buď se k tomuto účelu dá použít specializovaný nástroj pro testování REST API či se dají volání posílat pomocí vlastního kódu. Pro nastavení testovacího prostředí je zapotřebí, aby byly nakonfigurovány databáze a server dle aplikačních požadavků, jehož funkčnost se ověří pomocí API volání. Možnosti výstupu API jsou buď data různých formátů, stav pomocí stavového kódu, či zavolání jiné API funkce. [30]

Stejně jako v případě testování obecně, i v případě testování REST API, jsou zde možnosti manuálního i automatizovaného testování. V případě manuálního testování je možné poslat API volání přímo v kódu či za pomoci vhodného testovacího nástroje. K tomu je zapotřebí znát URI, tj. identifikátor zdroje, zvolit metodu REST API pro volání, zadat hlavičku specifikující očekávaný typ odpovědi, popřípadě specifikovat tělo zprávy, pokud to metoda vyžaduje. Poté může již tester zaslat API volání a získat výstup odpovědi. Ty mohou být ověřeny vůči očekávanému výstupu neboli asercemi. Veškeré tyto kroky lze automatizovat a provádět tak opakovaně či zátěžově, stejně jako automatizace může i generovat všechny požadavky automaticky a najednou z dokumentu WADL.

V případě testování REST API je u některých testovacích nástrojů možné využít i podpory mockování, které testování zjednodušuje. Mockování znamená, že namísto skutečné softwarové komponenty (API), je vytvořena a používána její náhrada, tzv. mock. Ta se chová jako originální API, ale postrádá některé z funkčních i nefunkčních charakteristik originální komponenty. Pro testování je možnost mockování užitečná, jelikož některé komponenty nemusí být dostupné kvůli bezpečnosti, výkonu, udržitelnosti či nejsou ještě vyvinuty. Základní mock může v těchto případech pomoci rozjet testování, jelikož věci jako základní testovací skripty či plánování exekucí zvládne i jednoduchý mock. Mock je samozřejmě limitován tím, že není plnou simulací korespondující komponenty, takže zpravidla neumožní veškeré testování, ale k odstartování testování zcela dostačuje. Vhod přichází mockování i při práci s externími komponentami, kdy mock není ovlivněn případnými nekonzistencemi v externí komponentě. Mocky mají rychlejší čas odezvy, šetří náklady oproti použití externích API a umožňují simulovat rozličné časy odezev. Mockování API je tedy koncept, který má řadu výhod při testování i vývoji. Umožňuje otestovat neočekávané chování a závislosti na externích komponentách. Umožňuje

izolovaný vývoj a brzký start na tvorbě funkčního testování, i předtím, než jsou skutečné komponenty dostupné. [31]

2.4 Nástroje pro testování REST API

Pro testování REST API existují specializované nástroje. Ty umožňují jejich manuální testování, některé z nich i testování automatizované, ovšem možnosti v této oblasti se napříč nástroji značně liší. Jelikož testovacích nástrojů existuje celá řada, a to ať již bezplatných či placených, nelze se věnovat všem z nich. Jak již z úvodu vyplývá, nástroj SoapUI je leaderem na trhu, nicméně nástroj Postman již popularitou moc nezaostává, jak je patrné z *Obr. 2 - Trend nástrojů SoapUI a Postman za 3 roky* v úvodu práce. S rozvojem samotné oblasti REST API se pojí i vývoj velkého množství nových testovacích nástrojů. Pro představu, o jak velké množství nástrojů k testování REST API se jedná, zde je pouze výčet těch nejpopulárnějších z nich [32]:

- Soap UI
  - stránky výrobce: https://www.soapui.org/
  - nástroj pro testování REST API v bezplatné i placené verzi
- Postman
  - stránky výrobce: https://www.getpostman.com/
  - nástroj pro testování REST API v bezplatné i placené verzi
- Katalon Studio
  - stránky výrobce: https://www.katalon.com
  - bezplatný nástroj pro testování REST API
- Assertible
  - stránky výrobce: https://assertible.com/
  - nástroj pro testování REST API v bezplatné i placené verzi
- Apigee
  - stránky výrobce: https://apigee.com/api-management/
  - placený nástroj pro testování REST API
- Apache JMeter
  - stránky výrobce: https://jmeter.apache.org/
  - bezplatný nástroj pro testování REST API
- Rest-Assured
  - stránky výrobce: http://rest-assured.io/
  - bezplatný nástroj pro testování REST API
- Tricentis Tosca
  - stránky výrobce: https://www.tricentis.com/

    - placený nástroj pro testování REST API
- Karate DSL
    - stránky výrobce: https://github.com/intuit/karate
    - bezplatný nástroj pro testování REST API
- LoadFocus
    - stránky výrobce: https://loadfocus.com/
    - placený nástroj pro testování REST API

Nástroje se stejně jako i u testování jiných oblastí liší v mnoha aspektech. Hlavní z těchto aspektů jsou zmíněny při definici kritérií pro porovnání dvou z těchto nástrojů, tj. SoapUI a Postman. Rozdíly jsou kupříkladu v grafickém rozhraní aplikací, možnostech automatizace, parametrizace či podpoře mockování. Tyto a více kritérií jsou představeny v následující kapitole, která se zabývá kritérii, které se dají aplikovat jak na zvolené nástroje, tak i na nástroje pro testování REST API obecně.

3 Kritéria pro porovnání nástrojů

Tato kapitola naplňuje třetí dílčí cíl, tj. definuje kritéria pro porovnání nástrojů pro testování REST API. Zatímco, pro účely této práce budou definovaná kritéria použita k porovnání dvou nástrojů, a to SoapUI a Postman, mimo rámec této práce, mohou být kritéria dále užita i ke srovnání nástrojů jiných.

K porovnání nástrojů je potřeba zvolit vhodná kritéria, která jsou do jisté míry specifická, a to jak pro oblast REST API, tak i pro dané nástroje. K definici těchto kritérií jsou použity pomocné zdroje, které jsou blíže definovány v první podkapitole. V další podkapitole jsou již definována konkrétní kritéria pro tuto práci, a v následujících podkapitolách poté stanoveny váhy jednotlivých kritérií a jejich hodnocení.

Tyto dva nástroje nebyly prozatím žádnými odbornými zdroji porovnány, takže neexistují zdroje, které by poskytly stoprocentně správná kritéria pro tento účel. Existuje ovšem řada pomocných zdrojů, které mohou ke stanovení kritérií pro účely této práce pomoci. Tyto zdroje společně se znalostmi těchto nástrojů a studiem technické dokumentace obou nástrojů [33; 34] pomohou stanovit vhodná kritéria pro porovnání nástrojů SoapUI a Postman.

3.1 Pomocné zdroje pro definici kritérií

Jako pomocné zdroje mohou sloužit informace z bakalářských prací Dominika Bláhy [16], Michaela Žabky [15] či Michala Kváčka [35]. Ani v jednom z případů se ovšem nejedná o porovnání nástrojů pro testování REST API, takže kritéria mohou sloužit jen jako částečná inspirace a podklad k analýze pro tuto práci.

V případě práce Dominika Bláhy, kde srovnává nástroje pro správu testů, se kritéria poměrně rozcházejí, nicméně některá jsou aplikovatelná i v případě této práce. Aplikovatelná zde jsou například kritéria, jako je podpora integrace, reporting, podpora importu/exportu souborů, uživatelské rozhraní, podpora či dokumentace.

V případě práce Michaela Žabky, kde se zabývá analýzou software testing nástrojů, se již dá najít více kritérií aplikovatelných v této práci. Jedná se především o automatizované testování, reporting, integraci, customizaci, dokumentaci, instalaci, cenu a podporu.

Poslední ze zmíněných je Michal Kváček, který se zabývá službou na testování REST API, kde především kritéria týkající se funkcionality mohou být aplikovatelná i v této práci. Některá kritéria se shodují i s předchozími pracemi, ale dají se zde nalézt i kritéria nová,

která jsou specifická pro tuto oblast. Jedná se především o mockování serverů, definice vlastních hlaviček, uživatelské role či notifikace o výsledcích testů.

Mimo tyto bakalářské práce se ovšem vhodná kritéria a jejich kategorizace dají dohledat i na webových serverech, především na těch, zabývajících se srovnáváním nástrojů, jmenovitě například na serverech ToolsQA [36], Guru99 [37] či Toptal [38]. Vzhledem k tomu, že dané weby řeší jmenovitě i tyto nástroje, či nástroje příbuzné, dají se použít jako vhodný filtr kritérií z předchozích prací i k definice kritérií nových.

3.2 Definice kritérií

Pro samotnou definici kritérií, která jsou vybrána na základě analýzy pomocných zdrojů a znalosti testovacích nástrojů a jejich specifik, bylo zvoleno pro účely této práce rozdělení kritérií do třech základních kategorií. V daných kategoriích jsou poté kritéria vyjmenována společně se základním vysvětlením, na co se kritérium soustředí. Splnění kritéria je poté hodnoceno nejen absolutně, ale i ve srovnání právě s druhým testovacím nástrojem.

Základní kategorie obsahující kritéria pro porovnání jsou kategorie *Funkcionalita*, kategorie *Uživatelské prostředí*, a kategorie *Dostupnost*. U daných kritérií je uvedena i stupnice hodnocení naplnění kritérií, která je blíže přiblížena v následujících dvou podkapitolách.

První kategorií je Funkcionalita, která se zabývá funkční stránkou testovacích nástrojů. Do kritérií nebyly zařazeny některé základní funkčnosti, jelikož je splňují všechny nástroje pro testování REST API a nikterak by tedy nepřispěly k porovnání nástrojů. Jedná se především o základní možnosti nástroje, jako je posílání volání API, podpora HTTP metod, definice zdrojů či tvorba testovacích projektů.

V kategorii Funkcionalita (KF) jsou definována tato kritéria:

- KF1 – Možnosti automatizace testování
    - Toto kritérium řeší možnosti automatizace testování, především tedy, zda nástroj vyžaduje externí aplikace pro automatizaci či umožňuje běh z příkazové řádky.
    - Stupnice hodnocení:
        - 0 – neumožňuje automatizaci
        - 1 – umožňuje automatizaci pouze v rámci nástroje
        - 3 – umožňuje automatizaci mimo nástroj pomocí externích aplikací
        - 4 – umožňuje automatizaci mimo nástroj bez dodatečných aplikací
        - 5 – má více možností automatizace včetně běhu z příkazové řádky

- KF2 – Možnost integrace s externími nástroji
    - Kritérium řeší především integraci s nástroji pro průběžnou integraci, ale může zahrnovat i případné další nástroje, s kterými lze nástroj pro testování REST API integrovat, jako jsou nástroje pro týmovou spolupráci či správa projektů.
    - Stupnice hodnocení:
        - 0 – neumožňuje integraci s jinými nástroji
        - 1 – umožňuje integraci s nástroji, ale ne pro průběžnou integraci
        - 3 – umožňuje integraci s nástroji pro průběžnou integraci
        - 5 – umožňuje integraci s více nástroji
- KF3 – Možnost definice vlastních parametrů u požadavků
    - Zde je důležitá především možnost definice vlastních hlaviček, ovšem i dalších parametrů, které mohou usnadnit testování
    - Stupnice hodnocení:
        - 0 – nemá možnost definice vlastních parametrů
        - 3 – má možnost definice vlastních hlaviček
        - 5 – má možnost definice i více parametrů
- KF4 – Možnost mockování serverů
    - V tomto případě se hodnotí možnosti testovacích nástrojů mockovat servery, tedy testovat bez užití již existující infrastruktury
    - Stupnice hodnocení:
        - 0 – nemá možnost mockovat servery
        - 5 – má možnost mockovat servery
- KF5 – Možnosti reportingu a notifikací
    - Zde se řeší možnosti reportování, a to jak z hlediska typu, podrobnosti, vzhledu, tak i přizpůsobení reportů či notifikací o nových výsledcích
    - Stupnice hodnocení:
        - 0 – neexistuje možnost reportů
        - 2 – existují základní možnosti reportů v programu
        - 3 – má možnost exportování reportů v různých formátech (bez customizace)
        - 4 – má možnost exportování reportů v různých formátech (s customizací)
        - 5 – má pokročilé možnosti reportování i notifikací
- KF6 – Možnost paralelizace testů
    - Toto kritérium je poměrně jednoznačné a zabývá se možností jet více testů paralelně, což samozřejmě ovlivní výsledný testovací čas

    - Stupnice hodnocení:
        - 0 – neexistuje možnost paralelizace testů
        - 5 – existuje možnost paralelizace testů
- KF7 – Možnosti týmové spolupráce a definice rolí
    - V kritériu se řeší možnosti týmové spolupráce daných nástrojů a definice různých uživatelských rolí
    - Stupnice hodnocení:
        - 0 – neumožňuje možnosti sdílení práce v týmu
        - 3 – umožňuje možnosti sdílet práci v týmu v omezeném rozsahu
        - 4 – umožňuje definici uživatelských rolí
        - 5 – umožňuje sdílet práci v týmu bez omezení
- KF8 – Možnosti rozšíření funkcí nástroje
    - Toto kritérium se zabývá možností rozšíření nástroje o další funkcionality, součástí hodnocení je i případná cenová politika daných rozšíření
    - Stupnice hodnocení:
        - 0 – nemá možnosti rozšíření
        - 1 – má možnosti rozšíření pomocí předplatného
        - 2 – má základní jednorázově placené možnosti rozšíření
        - 3 – má pokročilé jednorázově placené možnosti rozšíření
        - 4 – má základní bezplatné možnosti rozšíření
        - 5 – má pokročilé bezplatné možnosti rozšíření
- KF9 – Podpora jiných typů testování (mimo funkční testy)
    - Zde se hodnotí možnosti nástrojů vykonávat i jiné typy testů, než jsou funkční, tj. například testy zátěžové či bezpečnostní
    - Stupnice hodnocení:
        - 0 – umožňuje pouze funkční testy
        - 3 – umožňuje i jiný typ testů
        - 5 – umožňuje více typů jiných testů
- KF10 – Kompatibilita se soubory z jiných aplikací (import/export)
    - Poslední kritérium z kategorie funkcionality se zabývá podporou importování a exportování souborů z jiných nástrojů, tj. potažmo i jednoduchostí přejít k nástroji od konkurence
    - Stupnice hodnocení:
        - 0 – nemá možnost importu ani exportu
        - 1 – má možnost exportu
        - 4 – má možnost exportu i importu
        - 5 – podporuje více externích aplikací

Druhou kategorií je Uživatelské prostředí, které se zabývá především nároky na uživatele a možnostmi si program přizpůsobit. Zde je již množství kritérií menší i vzhledem k menší váze kategorie v porovnání s první zmíněnou.

V kategorii Uživatelské prostředí (KU) jsou definována tato kritéria:

- KU1 – Nutnost znalosti programování pro užití nástroje k testování
    - Kritérium řeší nutnost uživatele/testera znát programovací jazyk pro případnou automatizaci, případně do jaké míry je toto nezbytné
    - Stupnice hodnocení:
        - 0 – neumožňuje automatizaci bez pokročilé znalosti programování
        - 2 – umožňuje automatizaci se základními znalostmi programování
        - 3 – umožňuje základní automatizaci bez znalosti programování
        - 5 – umožňuje pokročilou automatizaci bez znalosti programování
- KU2 – Jednoduchost a možnosti instalace nástroje
    - Toto kritérium se zabývá možnostmi instalace nástroje, její jednoduchostí a srozumitelností
    - Stupnice hodnocení:
        - 0 – instalace nenabízí možnosti specifikace místa ani typu instalace
        - 1 – instalace nabízí možnosti volby místa i typu instalace
        - 3 – instalace je jednoduchá, ovšem s výhradami
        - 5 – instalace je jednoduchá i srozumitelná bez výhrad
- KU3 – Vzhled uživatelského rozhraní
    - Zde se klade důraz na přehlednost uživatelského prostředí a design. Ačkoliv design je kritérium poměrně subjektivní, bude také odůvodněno
    - Stupnice hodnocení:
        - 0 – chaotický design aplikace
        - 1 – pomalé přechody v aplikaci
        - 3 – strukturovaný design aplikace
        - 5 – moderní design aplikace
- KU4 – Možnosti nastavení nástroje
    - Kritérium řeší možnosti a jednoduchost případných změn nastavení testovacího nástroje
    - Stupnice hodnocení:
        - 0 – program neumožňuje upravit nastavení
        - 1 – program umožňuje jen základní změny nastavení ovšem složitě
        - 3 – program umožňuje jen základní změny nastavení jednoduše
        - 5 – program umožňuje změny nastavení bez omezení

- KU5 – Možnosti přizpůsobení vzhledu nástroje
    - Důraz je zde především na možnosti přizpůsobení, tj. customizaci daného nástroje pro potřeby konkrétních uživatelů
    - Stupnice hodnocení:
        - 0 – program nemá možnosti přizpůsobení
        - 3 – program má jen základní možnosti přizpůsobení
        - 5 – program má pokročilé možnosti přizpůsobení

Třetí a poslední kategorií je Dostupnost, která se zabývá licenční politikou či jednoduchosti řešení problémů, a to jak svépomocí, tak s podporou.

V kategorii Dostupnost (KD) jsou definována tato kritéria:

- KD1 – Typ licence
    - Kritérium řeší cenu daného testovacího nástroje i typ licence, tj. možnosti úpravy daného nástroje
    - Stupnice hodnocení:
        - 0 – program nemá bezplatnou licenci
        - 3 – program je bezplatný ovšem nikoliv open-source
        - 5 – program je open-source
- KD2 – Jazyk / lokalizace
    - Zde se řeší podpora jazyků testovacích nástrojů a případná oficiální i neoficiální lokalizace do češtiny
    - Stupnice hodnocení:
        - 0 – program nemá žádnou lokalizaci
        - 3 – program má neoficiální lokalizaci
        - 5 – program má oficiální lokalizaci
- KD3 – Nápověda / dokumentace
    - V tomto kritériu se řeší existence a podrobnost i přehlednost nápovědy testovacích nástrojů a jejich dokumentace
    - Stupnice hodnocení:
        - 0 – neexistuje žádná nápověda
        - 1 – existuje základní nápověda
        - 2 – existuje rozsáhlá nápověda
        - 3 – existuje základní dokumentace
        - 5 – existuje pokročilá dokumentace
- KD4 – Podpora
    - Kritérium se zabývá možností podpory nástroje, a to například podporou různých komunikačních prostředků

    - Stupnice hodnocení:
        - 0 – neexistuje možnost bezplatné podpory
        - 3 – existuje kanál bezplatné podpory
        - 5 – existuje více kanálů bezplatné podpory
- KD5 – Uživatelská komunita
    - Poslední kritérium kategorie Dostupnost se zabývá velikostí uživatelské komunity a možnostmi s komunitou řešit problémy s nástroji
    - Stupnice hodnocení:
        - 0 – neexistuje uživatelská komunita
        - 2 – existuje uživatelská komunita bez oficiálního fóra
        - 3 – existuje velká uživatelská komunita bez oficiálního fóra
        - 4 – existuje uživatelská komunita s oficiálním fórem
        - 5 – existuje velká uživatelská komunita s oficiálním fórem

3.3 Stanovení vah ke kritériím

Ke stanovení vah kritérií je použita metoda pořadí. Jedná se o jednoduchou metodu, ve které se stanoví pořadí kritérií dle preferencí. Metoda spočívá v tom, že jednotlivé varianty se hodnotí čísly 1, 2,..., n, kde n symbolizuje počet variant. Váha kritérií se počítá dle vzorce $v_j = \frac{b_j}{\sum_{j=1}^{n} b_j}$. Váha kritéria je tedy vypočtena tak, že v případě kritéria na prvním pořadí se celkový počet kritérií v čitateli vydělí součtem bodů všech kritérií ve jmenovateli. Sestup kritéria v pořadí o 1 znamená i snížení čitatele o 1, což vede ke snížení jeho váhy. Jmenovatel se nemění, jelikož součet všech bodů u kritérií je neměnný. [39]

V případě kategorie *Funkcionalita*, kde je 10 kritérií, tedy budou váhy 10/55 až 1/55. V případě kategorií *Uživatelské prostředí* a *Dostupnost*, kde je kritérií 5, tedy budou váhy 5/15 až 1/15.

Výsledkem porovnání bude poté i srovnání kritérií na úrovni kategorií, kde to tedy obdobně budou váhy 3/6 až 1/6.

Pořadí, dle kterého jsou váhy rozděleny je patrné již z výčtu, který byl seřazen právě dle důležitosti pro potřeby testování REST API. Seřazeny jsou dle pořadí jak kritéria, tak dané kategorie.

Funkcionalita je na seznamu nejvýše, jelikož funkcionalita programu je nejdůležitějším aspektem, který rozhoduje o volbě vhodného testovacího nástroje. V rámci kategorie jsou poté jednotlivé funkce na základě důležitosti pro testování REST API.

Uživatelské prostředí je další v pořadí, následované kategorií Dostupnost. Důvod pro zvolení tohoto pořadí zbývajících kategorií je to, že věci jako nutnost znalosti programování pro užití nástroje k testování či možnosti nastavení jsou důležitější než kupříkladu podpora či nápověda. V případě volby placených nástrojů by kritérium Typ licence mohlo s pořadím zahýbat, ovšem od počátku se pracuje s bezplatnými verzemi obou nástrojů, takže typ licence je řešen spíše ve vztahu k případné úpravě kódu programu.

3.4 Hodnocení kritérií

Pro hodnocení zvolených kritérií byl jako inspirace použit systém hodnocení z odborné publikace z roku 2018, a to konkrétně z knihy Zlepšování procesů při budování informačních systémů [40] od docentky Aleny Buchalcevové. Pro účely hodnocení přišla vhod především kapitola Systém hodnocení a výběru metodik METES [40 stránky 137-151].

Kritéria jsou hodnocena stupnicí 0-5, kdy jednotlivé hodnoty závisí na stupni splnění daného kritéria, což je níže podrobněji rozepsáno. V případě méně stupňů naplnění daného kritéria nemusí být celá stupnice vyčerpána, tj. například v případě kritéria s možností pouze ano či ne, jsou použité pouze hodnoty 0 a 5. Stupnice je definována v předchozí podkapitole vždy u jednotlivých kritérií.

V rámci každé z kategorií budou kritéria nejdříve hodnocena zvlášť. Po vyhodnocení těchto kritérií bude hodnocena celá kategorie celkově a poté budou nástroje porovnány také pomocí souhrnného hodnocení za dané kategorie. Dané hodnoty budou vždy násobeny vahami, které byly stanoveny v předchozí podkapitole, a to konkrétně pomocí metody pořadí.

K přehlednosti hodnocení budou kritéria vyhodnocena v tabulce, která je vytvořena v další podkapitole, a bude obsahovat hodnoty a výpočty včetně souhrnného hodnocení obou nástrojů.

3.5 Tabulka pro aplikaci kritérií

Tab. 2 - vzor pro hodnocení kritérií (vlastní tvorba)

Tabulka hodnocení				
Kategorie	Kritérium	Hodnocení	Váha	Celkem
Funkcionalita	KF1			
	KF2			
	KF3			
	KF4			
	KF5			
	KF6			
	KF7			
	KF8			
	KF9			
	KF10			
			Celkem:	
Uživatelské prostředí	KU1			
	KU2			
	KU3			
	KU4			
	KU5			
			Celkem:	
Dostupnost	KD1			
	KD2			
	KD3			
	KD4			
	KD5			
			Celkem:	
Celkové hodnocení:				

4 Nástroj SoapUI

V této kapitole je představen první z testovacích nástrojů ke srovnání, a to konkrétně SoapUI, včetně souvisejících produktů, možností tohoto nástroje a jeho využití pro účely testování REST API.

4.1 Základní představení SoapUI

SoapUI je leaderem na trhu v rámci open source testovacích nástrojů pro REST API. Díky tomu, že se jedná o open source program, je tedy volně šířen a jeho funkce mohou být upraveny tak, aby vyhovovaly požadavků každého uživatele. Může být použit pro funkční i nefunkční testování (např. zátěžové či bezpečnostní testy). Ačkoliv není limitován použitím pro webové služby, testování webových služeb je pro něj stěžejní. Při testování webových služeb dokáže zastávat roli klienta i služby. Uživatel může rychle vytvořit funkční i nefunkční testy při použití jednoho prostředí.

Prvně byl program SoapUI na trh uveden v říjnu 2005 ve verzi 1.0 a vznikl díky tomu, že Ole Lensmer při práci na architektuře založené na služby potřeboval nástroj pro testování k podpoře agilního vývoje [9]. Ve svém volném čase tedy vyvinul SoapUI, který se postupem času proměnil v open source projekt a komunita se začala rychle rozrůstat. Tvůrce programu, Ole Lensmer nejdříve projekt řídil skrze společnost Eviware, dokud v červenci roku 2011 nedošlo k její akvizici společností SmartBear Software, do které tedy nyní SoapUI patří [9]. Od té doby se počet verzí, i s nimi spojených nových funkcí a vylepšení, vyšplhal v době napsání k verzi 5.5.0.

Program může být použit k testování REST API i SOAP webových služeb. Testování tímto nástrojem je jednoduché, a i pro tvorbu zátěžových testů lze použít již existující funkční testy, kde pomocí několika kliknutí uživatel zvládne vytvořit na jejich základě testy zátěžové. SoapUI běží na většině operačních systémů, mimo Windows podporuje i Mac či různé distribuce Linuxu.

4.2 Produkty SoapUI a jejich rozšíření

SoapUI jakožto nejrozšířenější nástroj na trhu k testování REST API, nabízí velkou řadu funkcionalit, a ačkoliv začínal jako bezplatný open-source nástroj, v dnešní době již nabízí i placené funkcionality nabízející pokročilé možnosti, a to ve své verzi SoapUI Pro. Placená verze nástroje dokáže znatelně ušetřit čas při tvorbě a údržbě testů, stejně jako odemknout nové funkce oproti své bezplatné variantě. Cena placené verze nástroje je 595€ ročně [41].

Pro srovnání těchto variant SoapUI a výčet hlavních funkcionalit obou z nabízených možností, může posloužit srovnávací tabulka viz Tab. 3 níže.

Tab. 3 - rozdíly mezi verzemi SoapUI (data z webu SoapUI [42; 43], vlastní zpracování)

Funkce	SOAP UI (Open Source)	SOAP UI PRO
Podporované technologie		
SOAP/WSDL	✓	✓
REST	✓	✓
JMS	✓	✓
AMF	✓	✓
JDBC	✓	✓
HTTP	✓	✓
Automatizace		
Funkcionální testy	✓	✓
Zátěžové testy	✓	✓
Mockování služeb	✓	✓
Vygenerování kódu	✓	✓
Příkazová řádka	✓	✓
Maven	✓	✓
Nástroje pro průběžnou integraci	✓	✓
Obecné funkce		
Samostatná aplikace	✓	✓
Vzory kódu Groovy	✗	✓
Podpora více prostředí	✗	✓
Sdílení licencí	✗	✓
Funkce funkcionálního testování		
Pokrytí WSDL	✗	✓
Pokrytí Žádost/Odpověď	✗	✓
Aserce zpráv	✓	✓
Refaktorování testů	✗	✓
Spouštění více testů	✓	✓

Testy řízené daty	✗	✓
Skriptovací knihovny	✗	✓
Jednotkové reportování	✗	✓
Manuální testovací kroky	✓	✓
Funkcionality bezpečnostního testování		
Skenování hranic	✓	✓
Nevalidní typy	✓	✓
SQL injekce	✓	✓
XPath injekce	✓	✓
XML bomba	✓	✓
Skenování Fuzzing	✓	✓
Skriptování napříč sítěmi (XSS)	✓	✓
Nastavitelné skenování	✓	✓
Reportování	✗	✓
Funkcionality zátěžového testování		
Rychlé zátěžové testy z funkcionálních	✓	✓
Konfigurovatelné zátěžové strategie	✓	✓
Aserce zátěžových testů	✓	✓
Statistiky v reálném čase	✓	✓
Monitorování výkonu	✓	✓
Exportování statistik	✓	✓
Sestavení pomocí skriptů Groovy	✓	✓
Integrace LoadUI	✓	✓
Reportování	✗	✓
Reporty		
JUnit reporty	✓	✓
Export reportů	✗	✓
Pokrytí WSDL	✗	✓
Pokrytí testovací sady	✗	✓

Pokrytí testovacího případu	✗	✓
Pokrytí asercí	✗	✓
Pokrytí nahrání zpráv	✗	✓
Podpora		
Podpora e-mailem	✗	✓
Podpora telefonem	✗	✓

Z tabulky Tab. 3 je patrné, že ačkoliv existují rozdíly mezi oběma produkty, velké množství užitečných funkcí je dostupné i v bezplatné verzi.

SoapUI PRO ovšem narozdíl od bezplatné varianty umožňuje testování řízené daty. Lze tedy pracovat s externími zdroji dat, jako jsou textové soubory, XML, Groovy, Excel, soubory, či databáze. Díky tomu je možné škálovat testy množstvím vstupů skrze zmíněné zdroje.

Další prémiovou funkcí dostupnou pouze v placené verzi je pokrytí testy. SoapUI PRO nabízí testerům možnost získat report se statistikou, které funkcionality jsou dobře otestovány, ale i které oblasti již tolik nejsou. Reporty umožňují i konkrétně určit, co nebylo otestováno a které aserce v testech chybí.

Placená verze SoapUI umožňuje navíc i debuggování testů. Uživatel tedy může spustit test k bodu přerušení a zobrazit současné hodnoty SoapUI atributů. Rozhraní zjednodušuje jak test flow, proměnné, vlastnosti, žádosti, kontext, tak i mnoho dalších věcí, které usnadňují tvorbu testů a jejich zlepšení.

Podpora více prostředí je další z rozdílů oproti bezplatné variantě. Díky té lze pracovat s více prostředími, jako jsou QA, vývojové či produkční. V bezplatné variantě je potřeba vždy specifikovat dané koncové body pro exekuci v různých prostředích. SoapUI PRO na druhou stranu umožňuje jednoduše přepínat mezi prostředími.

Reporting je další z oblastí, kde je více rozdílů mezi oběma produkty. Oba produkty nabízejí reporty, ovšem bezplatná verze nabízí pouze základní reporty v aplikaci bez možnosti exportu. SoapUI PRO nabízí možnost customizovat detailní reporty na úrovni projektu, testovací sady, testovacích případů i load testů. Taktéž tyto reporty dokáže exportovat do formátů jako jsou PDF, HTML, Word či Excel.

V kategorii bezpečnostního testování mají obě varianty možnosti testování na zranitelnosti jako jsou XML bomby, SQL injekce, fuzzing a skriptování napříč sítěmi (XSS). Pouze

SoapUI PRO ovšem umožňuje testy zranitelnosti pomocí generátoru bezpečnostních testů na kliknutí myši.

Pro netechnického uživatele může produkt SoapUI PRO nabídnout také více možností při tvorbě SQL dotazů. Ačkoliv i bezplatná varianta tvorbu dotazů zvládne, prémiová varianta nabízí i jednoduché grafické rozhraní právě pro tento účel. Tato vlastnost pomáhá zrychlit i testování řízené daty.

Jak již u placených verzí obecně bývá, velké rozdíly jsou v oblasti podpory uživatelů. SoapUI v open-source variantě nabízí pouze online fórum, zatímco placená verze má možností o dost více. Jedná se o nonstop emailovou podporu, podporu po telefonu či bezplatné tréninky. [43]

Tvorba testovacích scénářů v SoapUI PRO je také jednodušší a nabízí více možností. Mimo klasické zadání koncového bodu, lze také importovat definice API (jakožto OAS 3.0), WSDL, kolekce, či nahrávat provoz na API. [43]

Aserce testů a jejich tvorba nabízí v obou variantách základní možnosti, jako jsou „obsahuje“, „neobsahuje“ či „rovná se“. Aserce jsou jednou z klíčových funkcí nástroje, díky níž nástroj validuje odpovědi při exekuci testovacích kroků na základě srovnání s částí zprávy či zprávou celou. Může se jednat např. o konkrétní HTTP kód. V placené verzi ovšem SoapUI umožňuje i pokročilé tvorby asercí na základě SLA, shodě s JSON schématem, swaggerem a dalšími.

Podporu pro průběžnou integraci nabízejí obě varianty, a lze tedy i v základní verzi přistupovat ke spouštění z příkazové řádky či integraci s Maven. Placená verze poté již pouze rozšiřuje možnosti integrace pomocí prémiových pluginů. [43]

4.3 Možnosti nástroje SoapUI

SoapUI je nástroj pro testování, kterým se dají testovat nejen REST API, ale i SOAP webové služby. Nabízí možnosti funkcionálního testování, výkonnostního testování, regresního testování a mnoho dalších, z nichž některé byly rozebrány detailněji již v předchozí podkapitole, jelikož jsou závislé na variantě produktu.

Podpora protokolů a technologií je u nástroje obsáhlá a SoapUI se integruje s řadou populárních nástrojů pro testování. Dokáže se integrovat například s nástrojem pro řízení projektu Apache Maven, nástroji pro průběžnou integraci jako je HUDSON, Subversion či GIT či nástroji pro jednotkové testování jako je JUnit [44].

Nástroj SoapUI pro tvorbu požadavků a posílání API volání umožňuje definici parametrů, a to nejen vlastní hlavičky, ale i dalších typů parametrů jako jsou QUERY, TEMPLATE,

MATRIX či PLAIN. Tyto požadavky dokáže samozřejmě ukládat do projektů a následně i exportovat pro znovupoužití v jiných projektech. Mimo export vlastních projektů zvládá SoapUI i importování kolekcí z konkurenčního nástroje Postman, který je taktéž předmětem této práce.

U funkčních testů podporuje program funkci přetažení parametrů pro tvorbu testovacích sad a kroků a urychluje tak vývoj komplexních testovacích scénářů bez nutnosti psát další skripty. U vytvořeného projektu lze přidávat jednotlivé testovací sady, které obsahují testovací kroky pro danou službu. Projekt může být použit vícekrát a SoapUI má funkci klonování, která umožňuje duplikovat existující testovací sady a použít je v jiných projektech. Nástroj taktéž umožňuje debuggování testů pro sledování průběhu testů.

SoapUI umožňuje i zátěžové testy a je možné testy distribuovat napříč více agenty. Jednoduše se dá tedy simulovat vysoký objem dat a reálná zátěž. K tomuto má nástroj funkci zvanou LoadUI. Mimo to umožňuje nástroj v prémiové verzi i pokročilé reportování k zachycení výkonnostních parametrů a i end-to-end systémový monitoring. [9]

SoapUI umožňuje automatizaci jak pomocí uživatelského rozhraní, tak i pomocí Groovy. V nástroji pro to stačí přidat pouze testovací krok v dané sadě. Skript Groovy obsahuje vlastní knihovny a může také integrovat další knihovny založené na Javě. Znalost Javy je pro tyto účely užitečná, jelikož pomocí tohoto skriptování je možné sestavit komplexní scénáře. Samotná automatizace poté může být spuštěna jak přímo z nástroje, tak i příkazové řádky.

Mimo testování zvládne i simulovat webové služby, nahrávat testy a na jejich základě vytvářet REST specifikace (WADL). Hlavní z možností nástroje byly představeny v této podkapitole, a také v kapitole předchozí, jelikož rozdíly ve funkcích jsou i mezi bezplatnou a placenou verzí. Kompletní výčet poté může poskytnout dokumentace SoapUI [33].

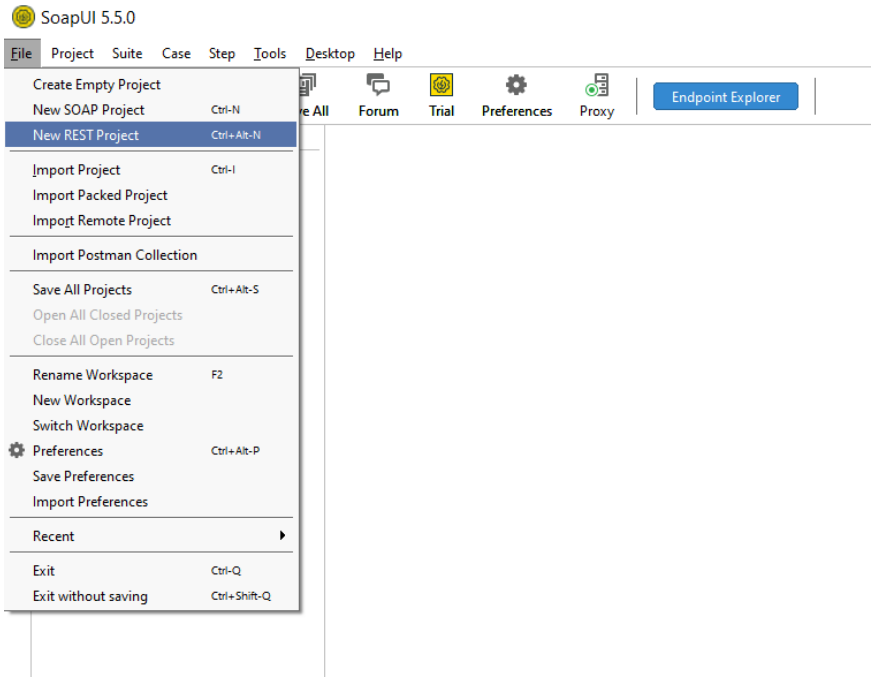
Pro účely této práce je nejpodstatnější možnost nástroje testovat REST API, proto se druhé velké kategorii testování, kterou tvoří SOAP webové služby, práce věnovat nebude.

4.4 Testování REST API pomocí SoapUI

Právě testování REST API specificky pomocí tohohle nástroje má za úkol přiblížit tato podkapitola. Definici REST API či testování obecně se věnují první kapitoly práce, ovšem zde je pouze základ pro testování pomocí tohoto nástroje, který umožňuje jak manuální, tak automatizované testování.

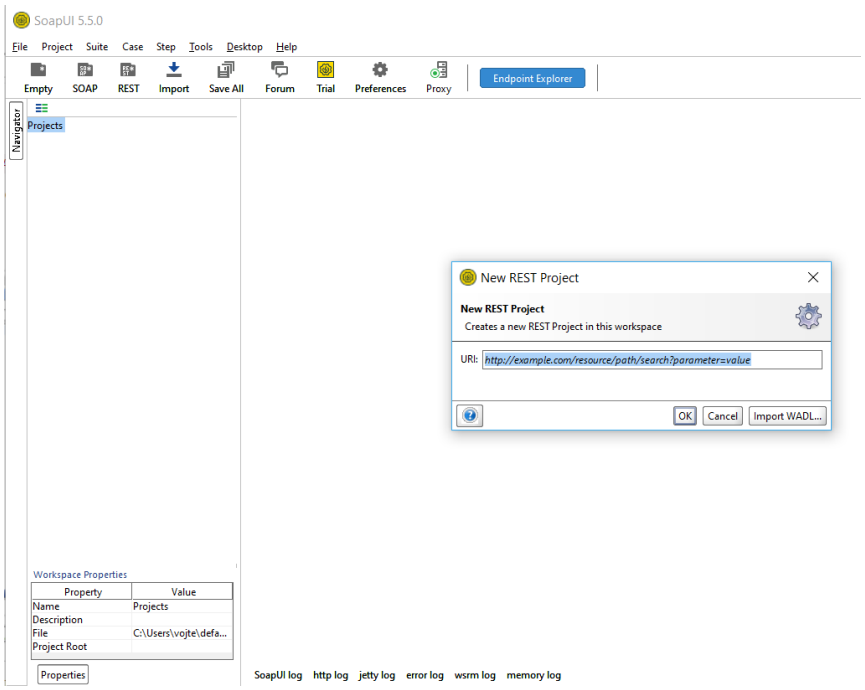
Po samotné instalaci nástroje se již může přistoupit k vytvoření REST projektu. Základem testováním v SoapUI jsou právě projekty. Poté, co se vytvoří projekt, se možnosti nástroje rozšiřují o funkční testy, zátěžové testy, mockování služeb, a další.

Pro začátek testování pomocí tohoto nástroje je tedy v programu vytvořen projekt, a to například skrze menu *File -> New REST Project*. Mimo tuto možnost lze projekt vytvořit i klávesovou zkratkou CTRL+ALT+N (ve Windows).



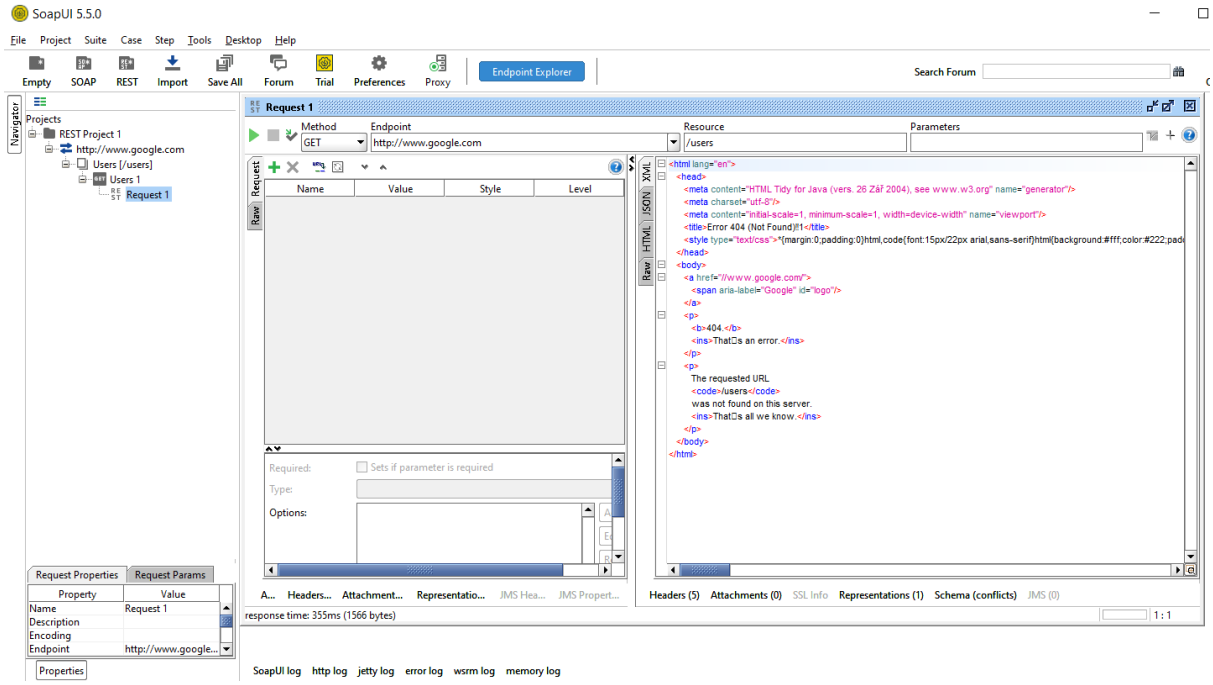
Obr. 3 – Vytvoření projektu v menu SoapUI (screenshot ze SoapUI v5.5.0)

Po zvolení možnosti vytvoření REST projektu se v programu otevře dialog vyžadující specifikace URI zdroje.



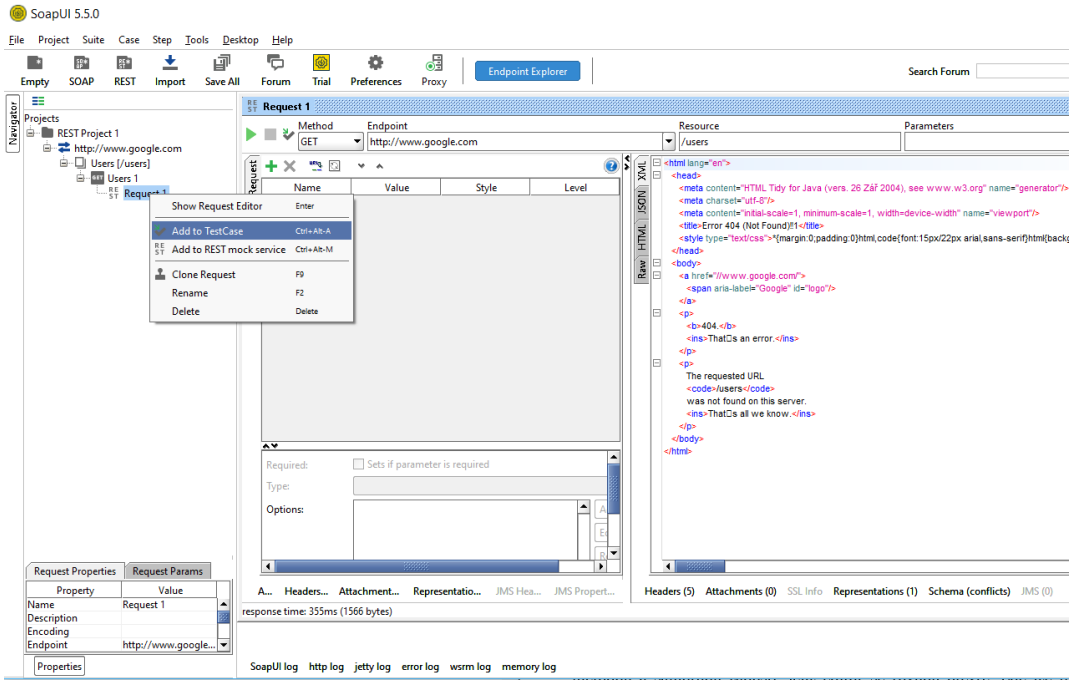
Obr. 4 - Zadání URI při založení projektu v SoapUI (screenshot ze SoapUI v5.5.0)

Po zadání adresy URI se již otevře okno prvního požadavku. Zde stačí kliknout na zelené tlačítko, které požadavek odešle. Poté se uživatelovi zobrazí odpověď API.



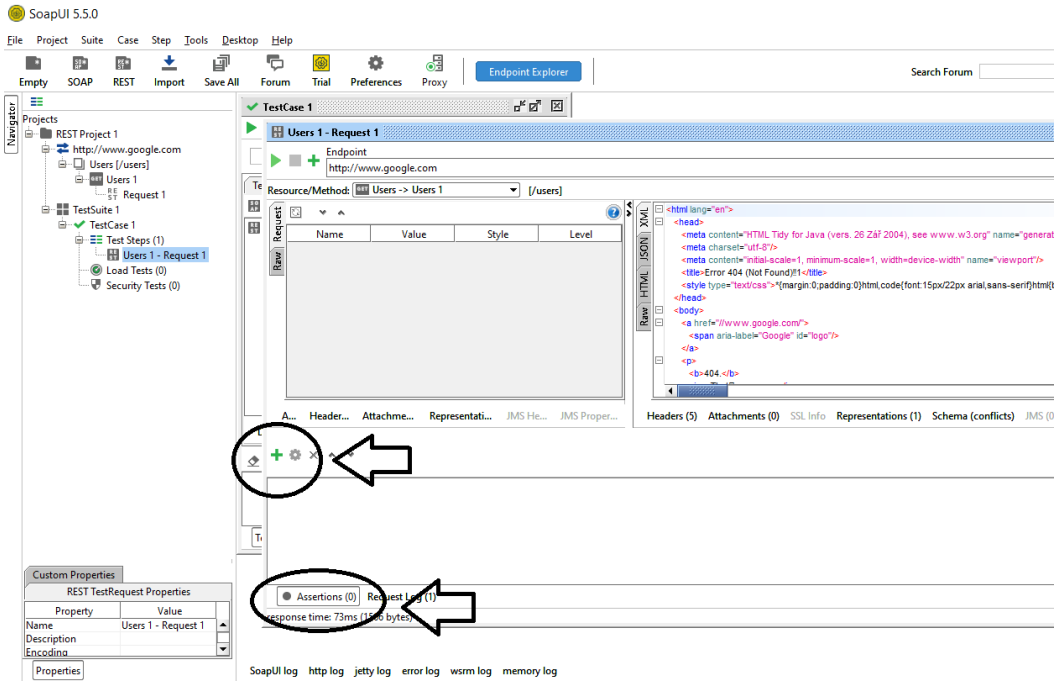
Obr. 5 – Požadavek v SoapUI a odpověď API (screenshot ze SoapUI v5.5.0)

Nyní když existuje projekt, ve kterém je specifikováno URI zdroje a požadavek, může uživatel přejít ke tvorbě prvního testovacího případu.



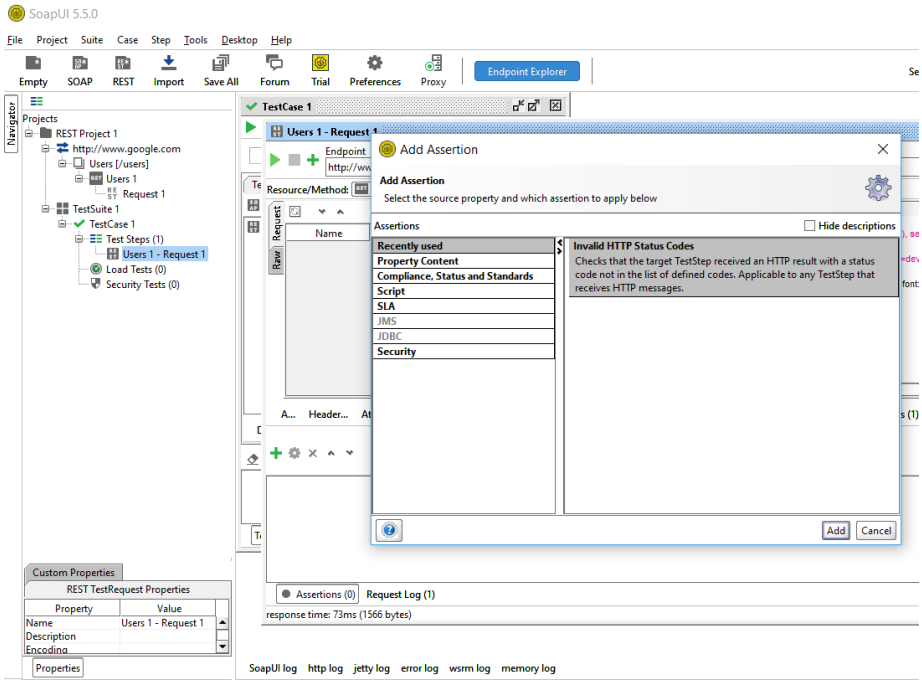
Obr. 6 - Tvorba testovacího případu v SoapUI (screenshot ze SoapUI v5.5.0)

Po kliknutí na přidání do testovacího případu v okamžiku, kdy uživatel nemá testovací sadu, vytvoří SoapUI i tu. Po vytvoření testovací sady a testovacího případu může již uživatel spustit testovací případ. Ten v tento okamžik ještě nic neověřuje a po prvním spuštění testu je tedy potřeba přidat aserci.



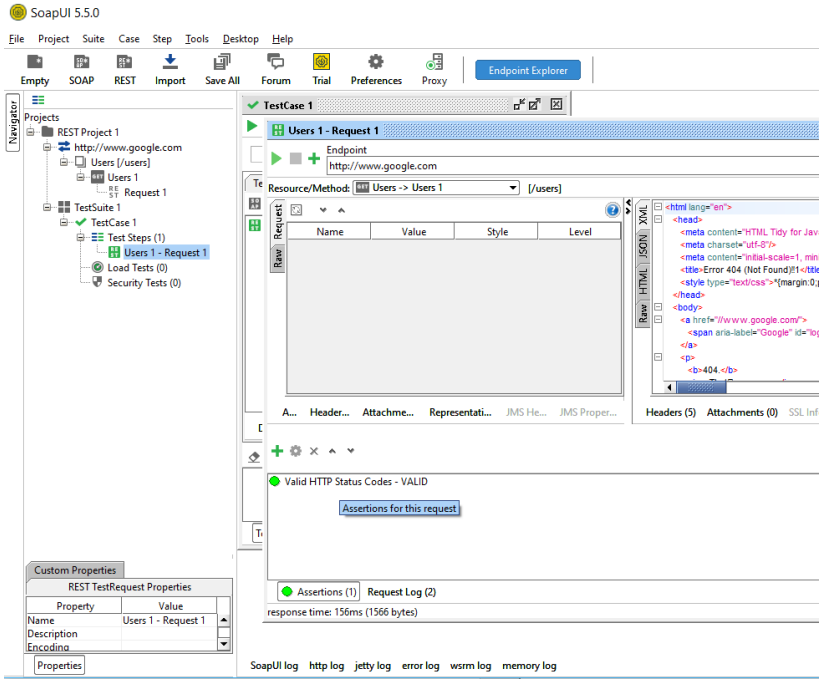
Obr. 7 - Přidání aserce do testu v SoapUI (screenshot ze SoapUI v5.5.0)

Po kliknutí na přidání aserce „+“ lze přes uživatelské rozhraní zvolit, co chce uživatel v testu ověřit. Lze například přidat ověření HTTP kódu, který API volání v odpovědi vrací.



Obr. 8 - Specifikace aserce v testu v SoapUI (screenshot ze SoapUI v5.5.0)

Po přidání aserce do testu pomocí *Add* se programu řekne například HTTP kód, který uživatel očekává a potvrdí *OK*. Po vytvoření aserce program ověří automaticky výsledek předchozího volání a zobrazí barvu na základě toho, zda test prošel či nikoliv.



Obr. 9 - Úspěšný test v SoapUI (screenshot ze SoapUI v5.5.0)

Pomocí těchto několika jednoduchých kroků tedy došlo v programu SoapUI k vytvoření projektu, specifikace zdroje, vytvoření požadavku, testovací sady a testovacího případu včetně jeho ověření. Během několika okamžiků tedy uživatel bez jakékoliv znalosti programování takto může otestovat REST API a vytvořený testovací případ nyní

opakovaně spouštět a rozšiřovat. Možností nabízí program samozřejmě více, jak je vysvětleno i v předchozích podkapitolách, které se tomuto programu věnují, ovšem toto byla pouze krátká demonstrace jednoduchosti použití programu pro testování REST API.

Poté již je možné po ukončení testu vytvořit report s výsledky a v případě prémiové verze i exportovat report do více formátů, kupříkladu do HTML, které se po vygenerování reportu automaticky otevře v prohlížeči.

5 Nástroj Postman

V této kapitole je představen druhý z testovacích nástrojů ke srovnání, a to konkrétně nástroj Postman, včetně souvisejících produktů, možností tohoto nástroje a jeho využití pro účely testování REST API.

5.1 Základní představení Postman

Postman je aktuálně jeden z nejpopulárnějších nástrojů pro testování API. Začal jako vedlejší projekt k zjednodušení API workflow v testování a vývoji, a to v roce 2012, kdy byl vytvořen programátorem Abhinavem Asthanou z Indie [45]. Původně se jednalo o aplikaci prohlížeče Chrome, která byla ovšem v roce 2016 zrušena a nahrazena nativní aplikací desktopovou aplikací [45]. Nástroj aktuálně používá přes 6 milionů uživatelů, 200 tisíc společností a 130 milionů API [4]. Postman je populární kvůli mnoha aspektům, z nichž několik základních je zmíněno níže.

Jeden z klíčových aspektů tohoto nástroje je jeho jednoduchost a uživatelská přívětivost. Uživatel si také může vytvořit účet, pomocí kterého může poté přistupovat k datům odkudkoliv, kde je Postman aplikace nainstalována.

Specifikem programu je taktéž užití kolekcí, které Postman tvoří pro svá API volání. Každá kolekce může tvořit podadresáře a vícero žádostí. To napomáhá především k organizaci testovacích sad. Postman taktéž umožňuje kolaboraci a kolekce i prostředí mohou být importovány či exportovány, což činí sdílení souborů velmi jednoduchým. Taktéž nástroj umožňuje vytvoření odkazu, pomocí kterého se můžou kolekce sdílet.

Nástroj taktéž umožňuje tvorbu více prostředí, a větší množství prostředí zamezuje nutnému opakování testů, jelikož stejná kolekce může být použita vícekrát. Zde přichází také do hry bohaté možnosti parametrizace, které Postman nabízí.

Při tvorbě testů umožňuje nástroj ověřování úspěšné HTTP odpovědi, což může být přidáno ke každému API volání k zajištění pokrytí testy. S tímto souvisí i možnosti automatizace, k čemuž slouží především nástroje Collection Runner či Newman, s kterými Postman pracuje, a testy také mohou běžet ve více iteracích, což šetří čas na opakující se testy.

Postman také umožňuje u vytvořených testů debugging, což je užitečné například k tomu, aby se zjistilo, jaká data byla obdržena v odpovědi. V neposlední řadě se nástroj může i integrovat s nástroji průběžné integrace, aby byly zachovány vývojové praktiky. [45]

5.2 Produkty Postman a jejich rozšíření

Stejně jako u nástroje SoapUI, i nástroj Postman je nabízen v bezplatné a placené verzi. Některé funkcionality jsou sdílené v obou verzích, ovšem některé pokročilejší jsou k dispozici pouze ve verzích placených, kde do rozdílů v licencích přichází i velikosti týmů.

Postman nabízí tři verze svého produktu, jedná se v prvé řadě o produkt Postman, který je bezplatný a je určen jedincům či malým týmům. Druhou verzí je Postman Pro, který stojí 8$/uživatel/měsíc a je určen týmům do 50 uživatelů. Tato verze již zahrnuje profesionální možnosti kolaborace a plný přístup k pokročilým funkcím programu. Třetí verzí produktu je Postman Enterprise, který stojí 18$/uživatel/měsíc a je určen týmům všech velikostí. V této verzi jsou i speciální funkce pouze pro Enterprise edici, zahrnující možnosti jednotného přihlášení (SSO), 10x vyšší limit pro pokročilé funkce a rozšířenou podporu. [46]

V tabulce Tab. 4 níže jsou zachyceny hlavní rozdíly mezi jednotlivými plány, které produkt nabízí. Výčet veškerých rozdílů je poté dohledatelný v dokumentaci Postman [34].

Tab. 4 - rozdíly mezi verzemi Postman (data z webu Postman [46], vlastní zpracování)

Funkce	**Postman**	**Postman Pro**	**Postman Enterprise**
Aplikace pro Mac, Windows, Linux	✓	✓	✓
Neomezené množství kolekcí, proměnných, prostředí a spuštění	✓	✓	✓
Neomezený prostor pro osobní a týmové práce	✓	✓	✓
Postman centrum nápovědy a podpora komunity	✓	✓	✓
Kolaborace (sdílené žádosti)	25	neomezené	neomezené
Dokumentace API (měsíční zobrazení dokumentů)	1000	100 000	1 milion
Mockování serveru (měsíční volání serveru)	1000	100 000	1 milion
Postman API (měsíční API volání)	1000	100 000	1 milion

API monitoring (měsíční volání)	1000	10 000	100 000
Možnost nakoupit extra volání na monitoring	✗	✓	✓
Emailová podpora, více časových pásem	✗	✓	✓
Jednotné přihlášení (SSO)	✗	✗	✓
Prémiová podpora a dedikované vedení účtů	✗	✗	✓
Dedikované IP adresy pro API monitoring	✗	✗	✓
Možnosti platby		Jen kreditní karta	Faktura nebo kreditní karta
Cena za uživatele na měsíc	0$	8$	18$
Upgrade monitoringu	Zahrnuto: 1000 volání měsíčně Upgrade na Pro nebo Enterprise pro více monitoringu	Zahrnuto: 10 000 volání měsíčně Předplatné: 20$ měsíčně za balíček 50 000 volání Jednotlivé dokupování: 0,75$ za 1000 volání	Zahrnuto: 100 000 volání měsíčně Předplatné: 20$ měsíčně za balíček 50 000 volání Jednotlivé dokupování: 0,75$ za 1000 volání

5.3 Možnosti nástroje Postman

Nástroj Postman nabízí široké možnosti využití a velkou řadu funkcionalit. Oproti jiným nástrojům se jednotlivé varianty produktu liší především počtem licencí a limitem měsíčního provozu, ovšem většina funkcí je dostupná i v bezplatné verzi. Z tohoto důvodu se mohou jednotlivé funkcionality zpravidla zmínit jako obecné bez rozdělení na jednotlivé produkty.

V aplikaci Postman se dají pro každé volání psát a spouštět testy pouze pomocí jazyku JavaScript, a tento kód se ukládá do záložky s testy. Testy jsou spouštěny po obdržení odpovědi na volání API. Postman nabízí také možnosti automatizace, kdy je možné veškeré vytvořené testy a žádosti zahrnout do jedné automatizované testovací sekvence s nástrojem Postman Collection Runner. Mimo to je možné kolekce pomocí instalace dodatečného nástroje Newman, což je utilita příkazové řádky, integrovat i do existující procesu průběžné integrace, jako je Jenkins či TeamCity. Za pomoci nástroje Newman, který je vytvořen konkrétně pro účely spouštění kolekcí nástroje Postman, je možné provádět automatizované testy i mimo aplikaci. Nástroj je taktéž potřeba k tvorbě reportů mimo aplikaci. [34]

Mezi jednu z funkcí programu patří možnosti návrhu a mockování. Postman umožňuje vytvářet API specifikace přímo uvnitř svých kolekcí. Díky možnosti rozdělení vývoje pomocí mockovací služby nástroje umožňuje vývojářům front-endu i back-endu pracovat současně.

Další z hlavních funkcionalit jsou možnosti debuggování. Nástroj umožňuje poslání žádostí k verifikaci připravenosti API. Je možné zahrnout taktéž skripty, testy a části kódu před samotným posláním žádosti k maximalizaci užitečnosti debuggingu. Možnost prověřit odpovědi API je přímo v programu. Zároveň je možné využít proměnné i prostředí k uložení a znovupoužití na více místech. To vše je možné uložit do kolekce nástroje Postman.

Dokumentace a její možnosti jsou taktéž důležitou součástí programu. Umožňuje vytvářet detailní dokumentaci pro API, její nastavení na veřejnou či soukromou, a její zobrazení ve webovém prohlížeči. K rychlé orientaci vývojářů je možné dokumentaci i stáhnout přímo do jejich instance nástroje Postman. Dokumentaci je možné obohatit také zahrnutím testů, příkladů, popisů a části kódu, a to ve více jazycích. Zároveň je možné dokumentaci obohatit i o vlastní logo či různé barvy. [34]

Možnosti API monitorování jsou taktéž rozsáhlé a monitorování může být nastaveno na preferované kadenci. Testovat se dá na výkon i chování, a verifikovat, že API odpovídá a zároveň pracuje, jak má. Výsledky monitoringu jsou poté dostupné přímo v rámci přehledu v programu. Když je monitoring nastaven, API mohou být sledovány i z webové stránky programu Postman.

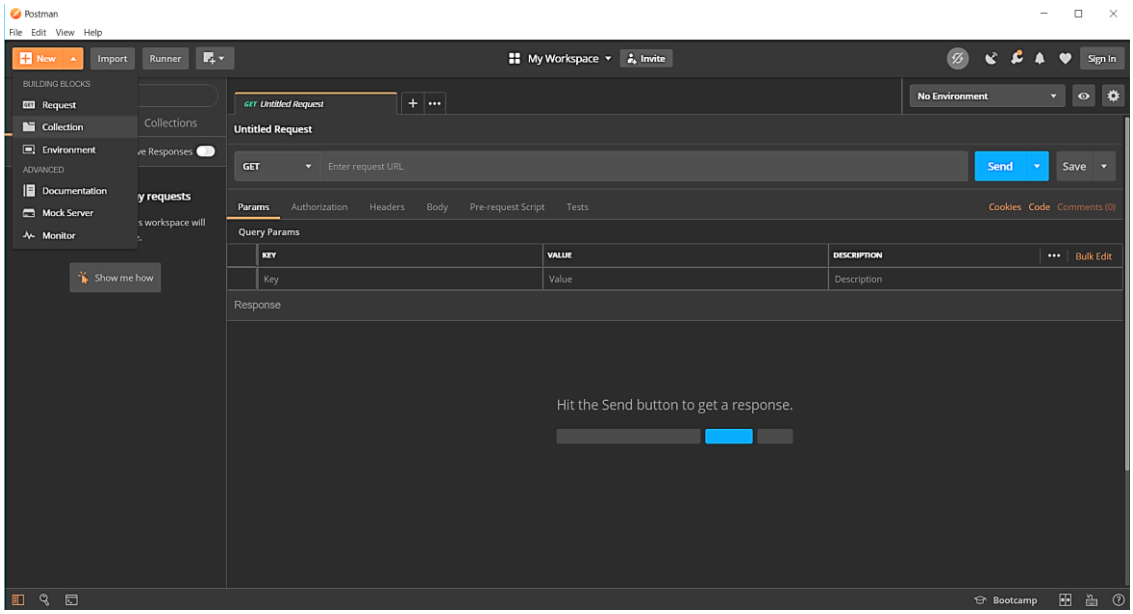
Možnosti publikace jsou také obsáhlé, a díky tomu je zjednodušený i nástup vývojářů, kteří mohou publikované API s kolekcemi Postman a dokumentací využít. Na stránkách či repositáři GitHub je možné zahrnout tlačítko „Spustit v Postman“, které umožní vývojáři jednoduše stáhnout API přímo do své Postman instance. API může být také zahrnuto v síti Postman API, což je kompletní list API, který je dostupný z webu i přímo z aplikace. [34]

5.4 Testování REST API pomocí Postman

Použití nástroje Postman pro testování REST API je velmi jednoduché a intuitivní. Postman poskytuje kolekci API volání, kterou lze následovat při testování API aplikací. V rozbalovacím tlačítku lze vybrat metodu volání API, nastavit autorizaci, hlavičku, tělo žádosti a další informace specifické pro API volání. Poté stačí již jen stisknout tlačítko pro odeslání žádosti.

Stejně jako u nástroje SoapUI, i v nástroji Postman existuje více způsobů, jak vytvořit kolekci, požadavky i testy. Jeden ze způsobů je v práci představen v této podkapitole.

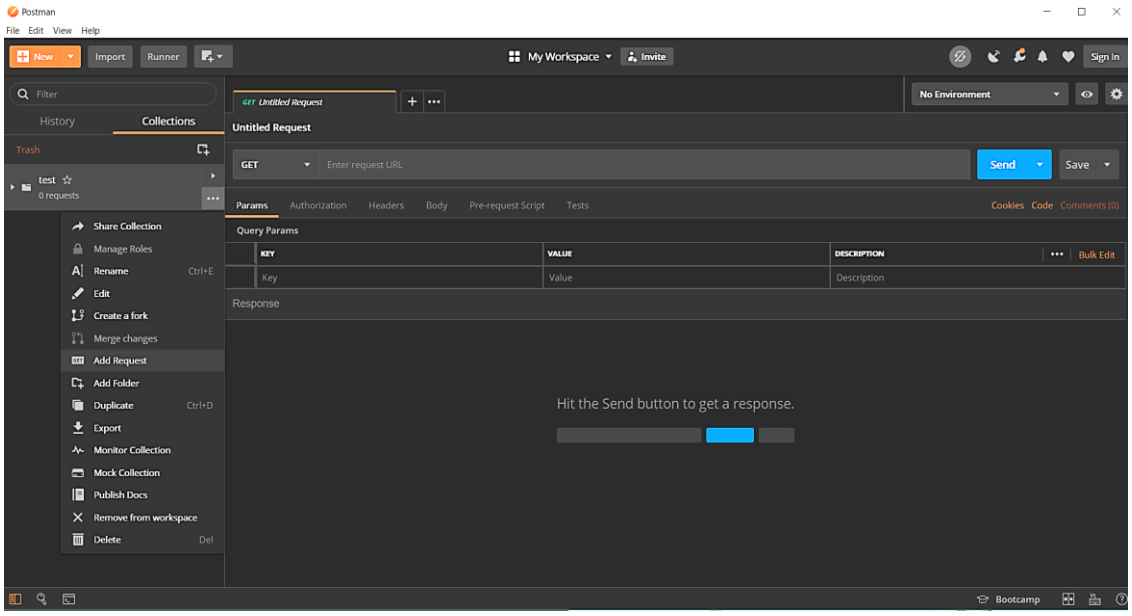
Pro přehledné vytvoření požadavku je v nástroji Postman nejdříve vytvořena kolekce, což je základní prvek, s kterým nástroj pracuje a v kterém uchovává související požadavky. K založení kolekce uživateli stačí kliknout na „+“ a zvolit *Collection*.



Obr. 10 - Tvorba kolekce v Postman (screenshot z Postman v7.0.9)

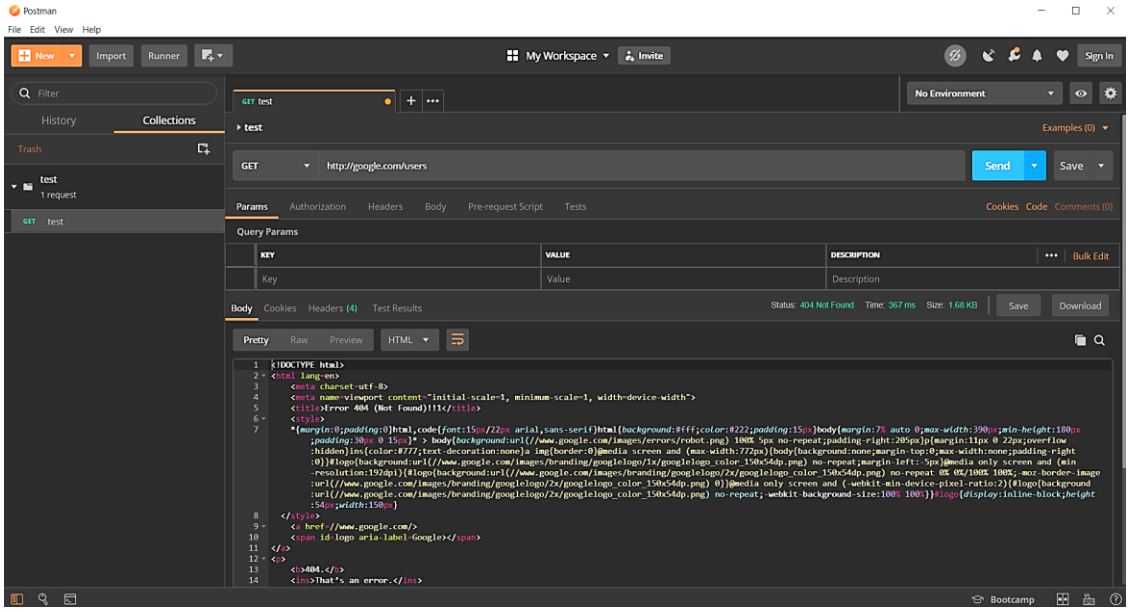
Poté se již uživateli otevře okno, kde může zvolit vhodné pojmenování kolekce. Potvrzením pomocí kliknutí na tlačítko *Create* se kolekce vytvoří.

V samotné kolekci je možné přidat první požadavek, který je pak možné otestovat. K přidání požadavku lze zvolit ikonu se třemi tečkami vedle názvu kolekce. Poté se uživateli zobrazí možnost *Add Request* k přidání nového požadavku do kolekce.



Obr. 11 - Přidání požadavku v Postman (screenshot z Postman v7.0.9)

Po zvolení vhodného pojmenování a případného popisu se prázdný požadavek uloží ke kolekci. K použití požadavku je nyní potřeba specifikovat URI, tj. adresu zdroje. Poté je již možné požadavek poslat pomocí tlačítka *Send*, tj. zaslat API volání na server. Po zaslání požadavku se již uživateli ukáže odpověď, kterou server vrací.

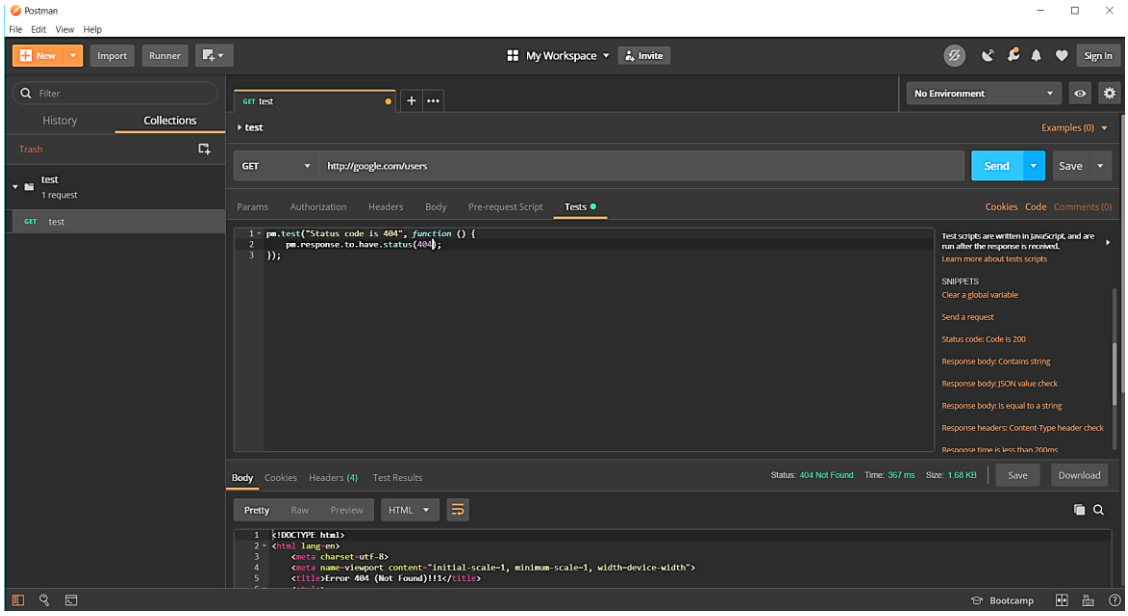


Obr. 12 - Odpověď na požadavek v Postman (screenshot z Postman v7.0.9)

V kolekci je tedy již vytvořen první požadavek, na který se vrátila i první odpověď. Prozatím ovšem chybí jakýkoliv test, který je tedy potřeba v nástroji vytvořit.

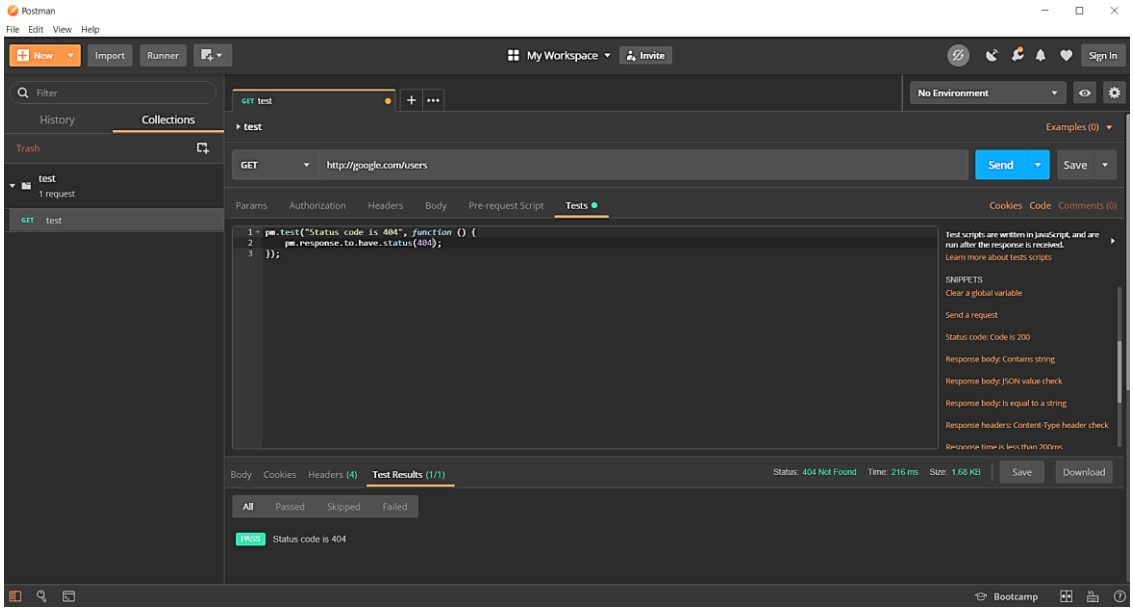
Pro vytvoření testu je potřeba přepnout se u požadavku do záložky *Tests*. Okno je v této záložce nejdříve prázdné. Nyní je možné test napsat, k čemuž program používá JavaScript. Bohužel v případě nástroje Postman neexistuje možnost testy specifikovat pouze pomocí uživatelského rozhraní. Vedle okna testů jsou ovšem *Snippets* neboli tzv. útržky kódu,

které může uživatel částečně použít pro napsání testů. I v případě potřeby úpravy těchto útržků kódu je ovšem nutné provést úpravy již v kódu JavaScriptu. V programu existuje například útržek pro ověření stavového kódu 200, ovšem pro ověření jiného kódu již uživatel musí upravovat v kódu, ačkoliv v tomto případě se nejedná o úpravu nikterak složitou (pro uživatele se základní znalostí kódu).



Obr. 13 - Tvorba testu v Postman (screenshot z Postman v7.0.9)

Posledním krokem k úspěšnému otestování prvního požadavku je již ověření stavu testu. Pod oknem s testem se po zaslání testu v záložce *Test Results* již zobrazuje výsledek testu.



Obr. 14 - Výsledek testu v Postman (screenshot z Postman v7.0.9)

Jak je tedy vidět, vytvořit první kolekci, požadavek i test není ani v programu Postman příliš náročné, ovšem uživatel již musí přijít do styku s kódem. Výsledek se zobrazí v programu, ovšem pro případné reporty je potřeba doinstalovat nástroj Newman.

6 Porovnání nástrojů SoapUI a Postman

V této kapitole je na základě stanovených kritérií provedeno hodnocení zvolených nástrojů, tj. aplikace SoapUI a aplikace Postman. V obou případech je zvolena bezplatná verze nástrojů. Pro vyhodnocení kritérií u daných aplikací se vychází z kapitol 4 a 5, věnovaným právě těmto produktům. Dále je užívána znalost obou testovacích nástrojů z praxe i oficiální dokumentace dostupná k těmto aplikacím [33; 34]. Vyhodnocení je popsáno jak v textu níže v podkapitolách, tak i dosazením do hodnotící tabulky Tab. 2, která byla vytvořena v kapitole 3.

6.1 Hodnocení nástroje SoapUI

Prvním nástrojem, který je hodnocen na základě kritérií stanovených v kapitole 3, je SoapUI. K samotnému posouzení kritérií, a tedy i následnému hodnocení slouží jak kapitola 4 se základními informacemi o aplikaci, tak i oficiální dokumentace k nástroji [33]. V neposlední řadě k posouzení kritérií pomáhá i znalost nástroje z praxe.

Nejdříve přichází na řadu hodnocení v kategorii *Funkcionalita*. Tato kategorie o 10 kritériích je stěžejní, a i z hlediska následného hodnocení má nejvyšší váhu.

- KF1 – První z kritérií v této kategorii jsou *Možnosti automatizace testování*. Zde SoapUI jako leader na trhu ukazuje bohaté možnosti automatizace, jelikož mimo možnosti automatizace pomocí TestRunneru přímo v aplikaci SoapUI, umožňuje pomocí skriptu TestRunneru, který je součástí instalace SoapUI i automatizaci bez běžící aplikace, a to dokonce i z příkazové řádky s možností využít bohaté možnosti parametrizace. V tomto kritériu tedy nástroj získal maximální hodnocení 5.
- KF2 – Dalším z funkčních kritérií je *Možnost integrace s externími nástroji*. Toto kritérium řeší možnosti integrace aplikace s aplikacemi jinými. Jak plyne i z kapitoly věnované SoapUI, v tomto ohledu má nástroj taktéž bohaté možnosti. Nejen, že se dokáže integrovat s nástroji průběžné integrace, ale i nástroji jako Apache Maven či JUnit. V neposlední řadě umožňuje i ostatním aplikacím integrovat třídy ze SoapUI, například za účelem sestavení požadavků či testovacích případů. Zde tedy nástroj taktéž splňuje kritérium s plným hodnocením, tj. 5.
- KF3 – Třetím kritériem je *Možnost definice vlastních parametrů u požadavků*. Toto kritérium si klade otázku, zda nástroj umožňuje definovat vlastní hlavičku a další vlastní parametry. Nástroj SoapUI umožňuje definici jak vlastní hlavičky, tak i parametrů jako je QUERY, TEMPLATE, MATRIX či PLAIN. Splňuje tedy kritérium na maximum bodů, tj. 5.
- KF4 – Dalším z kritérií je *Možnost mockování serverů*. Zde je otázka kritéria velmi jednoduchá, a to, zda tu možnost nástroj má či nikoliv. Nástroj SoapUI mockování služeb zvládá, takže kritérium splňuje a získává hodnocení 5.
- KF5 – Pátým kritériem jsou *Možnosti reportingu a notifikací*. Ačkoliv v tomto kritériu má SoapUI možnosti poměrně bohaté, jelikož je srovnávaná bezplatná

verze aplikace, tak ta má oproti své placené variantě pouze možnost si reporty prohlédnout v aplikaci, ovšem bez možnosti exportu. Vzhledem k této skutečnosti, že aplikace umožňuje pouze základní reporty přímo v programu, získává v tomto kritériu hodnocení 2.

- KF6 – Šestým kritériem je *Možnost paralelizace testů*. Zde je hodnocení pouze ve dvou hodnotách, které vypovídají o tom, zda nástroj umožňuje současné spouštění více testů či nikoliv. Nástroj SoapUI umožňuje testy spouštět jak sekvenčně, tak paralelně, tj. kritérium splňuje a hodnocení má 5.
- KF7 – Dalším z kritérií jsou *Možnosti týmové spolupráce a definice rolí*. Nástroj SoapUI má i v tomto kritériu bohaté možnosti, ovšem nikoliv u analyzované bezplatné varianty, ale pouze v placené verzi. Z hlediska bezplatné varianty produktu nástroj neumožňuje žádnou týmovou spolupráci ani definici rolí. Z tohoto kritéria získává tedy hodnocení 0.
- KF8 – Osmým kritériem v pořadí jsou *Možnosti rozšíření funkcí nástroje*. Nástroj SoapUI umožňuje rozšíření o velkou řadu funkcí, ovšem pouze přechodem na placenou variantu produktu. Za účelem tohoto rozšíření je tedy potřeba zaplatit, a to konkrétně formou předplatného. Jedná se tedy o pravidelné platby za rozšíření, díky čemuž v tomto kritériu program získává hodnocení 1.
- KF9 – Devátým kritériem je *Podpora jiných typů testování (mimo funkční testy)*. Nástroj SoapUI umožňuje testovat i zátěžovými testy, další typy testování jako je testování řízené daty či bezpečnostní testování jsou již výhradou prémiové verze. Vzhledem k možnosti jiného typu testu získává tedy nástroj v tomto kritériu hodnocení 3.
- KF10 – Posledním funkčním kritériem je *Kompatibilita se soubory z jiných aplikací (import/export)*. SoapUI v tomto ohledu umožňuje jak export vlastních souborů, tak import souborů od svého konkurenta Postman, dokazující mimo jiné blízkost daných nástrojů. Import z jiných externích aplikací SoapUI oficiálně nepodporuje. Díky těmto důvodům získává v tomto kritériu hodnocení 4.

Další z kategorií kritérií je *Uživatelské prostředí*. Tato v pořadí druhá kategorie obsahuje 5 kritérií, dle kterých je kategorie vyhodnocena.

- KU1 – Prvním kritériem kategorie je *Nutnost znalosti programování pro užití nástroje k testování*. Nástroj SoapUI ve své prémiové verzi obsahuje možnost automatizovat pouze pomocí několika kliknutí a bez jakékoliv znalosti programování. V bezplatné variantě je ovšem prostředí aplikace taktéž intuitivní, a i díky kvalitní dokumentaci k programu není potřeba k základní automatizaci znalost programování. Pro pokročilejší automatizaci je již ovšem příhodná znalost Groovy skriptování, díky čemuž v tomto kritériu dostává nástroj hodnocení 3.
- KU2 – Druhým kritériem je *Jednoduchost a možnosti instalace nástroje*. Z hlediska instalace nabízí nástroj možnosti volby destinace, instalovaných komponent, zástupců i následného spuštění programu. Instalátor je intuitivní a program je do pár minut nainstalován. Instalační balíček, u kterého to bylo testováno je verze 5.5.0 a má 133 MB. Vzhledem k tomu, že ke kritériu není žádných výhrad, nástroj zde získává hodnocení 5.

- KU3 – Třetím kritériem je *Vzhled uživatelského rozhraní*. Aplikace je přehledná, přechody v rámci aplikace nikterak nezadrhávají a obsah okna je logicky strukturovaný. Nicméně s přihlédnutím k samotnému designu je na první pohled patrné, že program z hlediska vzhledu neprošel dlouho aktualizací a užívá design, který již je řádku let zastaralý. Z tohoto důvodu získává nástroj v tomto kritériu hodnocení 3.
- KU4 – Dalším kritériem jsou *Možnosti nastavení nástroje*. Nástroj SoapUI má bohaté možnosti nastavení, které jsou přístupné mimo menu i přímo z hlavní obrazovky. Žádná omezení v úpravách nejsou. Toto kritérium je tedy hodnoceno 5.
- KU5 – Posledním kritériem této kategorie jsou *Možnosti přizpůsobení vzhledu nástroje*. Z hlediska přizpůsobení vzhledu nabízí nástroj SoapUI jen základní možnosti přizpůsobení, jako je chování programu po startu či řazení položek na obrazovce. Vzhled nelze nikterak zásadně customizovat. Z tohoto důvodu získal program v tomto kritériu hodnocení 3.

Poslední z kategorií pro hodnocení nástroje SoapUI je *Dostupnost*. Tato kategorie s nejnižší váhou obsahuje 5 kritérií, které vypovídají o dostupnosti aplikace.

- KD1 – Prvním kritériem kategorie *Dostupnost* je *Typ licence*. Nástroj SoapUI je v testované variantě nejen bezplatný, ale i Open Source, díky čemuž je volně šiřitelný i rozšiřitelný. Díky této flexibilitě má v tomto kritériu nástroj hodnocení 5.
- KD2 – Dalším z kritérií v kategorii je *Jazyk / lokalizace*. Nástroj SoapUI nenabízí žádné možnosti oficiální ani neoficiální lokalizace uživatelského rozhraní a veškerá interakce funguje v anglickém jazyce. Z tohoto důvodu v tomto kritériu dostává aplikace hodnocení 0.
- KD3 – Třetím kritériem této kategorie je *Nápověda / dokumentace*. SoapUI nabízí rozsáhlé možnosti nápovědy i dokumentace, včetně zdarma přístupných tutoriálů, které je možné nainstalovat při instalaci programu. Kritérium je hodnoceno v tomto případě 5 body.
- KD4 – Čtvrtým kritériem je *Podpora*. SoapUI pro svou Open Source verzi produktu nenabízí žádné možnosti podpory mimo komunitní fórum, které spadá do kritéria dalšího. Z tohoto důvodu je toto kritérium hodnoceno 0 body.
- KD5 – Posledním kritériem této kategorie je *Uživatelská komunita*. Ačkoliv bezplatná verze nástroje nezahrnuje podporu od společnosti SmartBear, která produkt vytvořila, uživatelská komunita i fórum k ní je veliké. Oficiální fórum má přes 80 tisíc členů [47]. Z tohoto důvodu je kritérium naplněno maximálně a hodnoceno 5 body.

Po zhodnocení daných kritérií je na řadě dosazení hodnot do tabulky a vypočítání celkového hodnocení na základě váhy jednotlivých kategorií i kritérií. Celkové hodnocení je poté použito při porovnání nástrojů SoapUI a Postman v dalších podkapitolách.

Tab. 5 - hodnocení kritérií pro SoapUI (vlastní tvorba)

Tabulka hodnocení				
Kategorie	Kritérium	Hodnocení	Váha	Celkem
Funkcionalita	KF1	5	10/55	0,91
	KF2	5	9/55	0,82
	KF3	5	8/55	0,73
	KF4	5	7/55	0,64
	KF5	2	6/55	0,22
	KF6	5	5/55	0,45
	KF7	0	4/55	0
	KF8	1	3/55	0,05
	KF9	3	2/55	0,11
	KF10	4	1/55	0,07
			Celkem:	4
Uživatelské prostředí	KU1	3	5/15	1
	KU2	5	4/15	1,33
	KU3	3	3/15	0,6
	KU4	5	2/15	0,67
	KU5	3	1/15	0,2
			Celkem:	3,8
Dostupnost	KD1	5	5/15	1,67
	KD2	0	4/15	0
	KD3	5	3/15	1
	KD4	0	2/15	0
	KD5	5	1/15	0,33
			Celkem:	3
Celkové hodnocení:				**3,77**

Z tabulky Tab. 5 tedy vyplývá, že nástroj získal dle hodnocení stanovenými kritérii a vahami celkové hodnocení 3,77 z 5.

6.2 Hodnocení nástroje Postman

Druhým hodnoceným nástrojem je Postman. Stejně tak jako nástroj SoapUI, je v této kapitole posouzen dle stanovených kritérií a následně ohodnocen. K posouzení kritérií i v tomto případě slouží kapitola 5 se základními informacemi o tomto nástroji, ale i oficiální dokumentace aplikace [34]. Opět jsou i zde tyto zdroje doplněny znalostí daného nástroje z praxe.

Prvně je nástroj hodnocen na základě kritérií v kategorii *Funkcionalita*. V této kategorii je nástroj hodnocen na základě 10 kritérií.

- KF1 – Prvním kritériem na řadě jsou *Možnosti automatizace testování*. Nástroj Postman umožňuje automatizaci v rámci nástroje. Pro možnost spouštění testů i mimo aplikaci Postman je ovšem potřeba nástroje Newman, s kterým se musí Postman integrovat. Kvůli nutnosti instalace dodatečné aplikace pro automatizaci mimo aplikaci, je tedy kritérium hodnoceno 3 body.
- KF2 – Druhým funkčním kritériem je *Možnost integrace s externími nástroji*. Postman je možné integrovat s nástroji průběžné integrace. Integrace s jinými nástroji, kupříkladu s nástrojem pro týmovou spolupráci Slack či GitHub, již je ovšem specifikem placených verzí produktu. Z tohoto důvodu je tedy kritérium hodnoceno 3 body.
- KF3 – Dalším kritériem v kategorii je *Možnost definice vlastních parametrů u požadavků*. Postman umožňuje definici vlastních hlaviček. Ačkoliv ostatní parametry jako jsou QUERY či TEMPLATE se již definují složitěji než u jiných nástrojů, jako je SoapUI, i to je možné pomocí specifikace proměnných. Z tohoto důvodu je tedy i přes to naplněno maximální hodnocení kritéria, tj. 5.
- KF4 – Čtvrtým kritériem je *Možnost mockování serverů*. Kritérium řeší, zda je nástrojem tato možnost poskytována či nikoliv. Postman tuto možnost poskytuje, i když v bezplatné verzi je omezená na konkrétní počet volání. Kritérium tedy nástroj splňuje, i když v případě větších projektů by již bylo potřeba zaplatit za vyšší počet API volání. Toto kritérium je tedy hodnoceno 5 body.
- KF5 – Dalším kritériem v pořadí jsou *Možnosti reportingu a notifikací*. Postman sám o sobě možnosti reportingu mimo aplikaci nemá. Možnosti reportování poskytuje pouze s pomocí další nástroje Newman. Aplikace jako taková tedy umožňuje pouze reportování v programu a je hodnocena 2 body.
- KF6 – Šestým kritériem je *Možnost paralelizace testů*. Hodnocení kritéria je v tomto případě založeno na jediné otázce, a to, zda nástroj umožňuje spouštět více testů zároveň. Dříve tuto možnost nástroj nenabízel, ovšem v aktuální době je to již jedna z podporovaných funkcionalit. Kritérium tedy nástroj splňuje a je hodnoceno 5 body.
- KF7 – Sedmým kritériem této kategorie jsou *Možnosti týmové spolupráce a definice rolí*. Postman možnosti týmové spolupráce nabízí. Prostor pro spolupráci

v bezplatné verzi omezen není, ovšem jak je u nástroje zvykem, je omezen počet, a to konkrétně počet žádostí, které lze sdílet v rámci týmu. Počet žádostí v bezplatné verzi je poněkud limitních 25. Pro více možností včetně definice rolí by již bylo potřeba přejít na některou z placených variant nástroje. Nicméně alespoň základní možnost týmové spolupráce nástroj poskytuje a kritérium je tak hodnoceno 3 body.

- KF8 – Dalším kritériem jsou *Možnosti rozšíření funkcí nástroje*. Nástroj Postman má velkou řadu užitečných funkcí, o které se dá rozšířit. Ovšem politikou nástroje je za přidané funkce platit, a tak je to spojené pouze s nutností přejít na některou z placených verzí nástroje. V případě přechodu na placenou verzi se odemknou pokročilejší funkce nástroje, ovšem je potřeba za to pravidelně platit, jelikož licence se hradí formou předplatného. Díky tomuto důvodu tedy získává program v tomto kritériu hodnocení 1.
- KF9 – Devátým kritériem je *Podpora jiných typů testování (mimo funkční testy)*. Nástroj Postman v základu podporuje pouze funkční testování. Na trhu existují nástroje, které jsou schopné zužitkovat kolekce nástroje Postman pro zátěžové testování, ovšem není to součástí tohoto nástroje. Z tohoto důvodu získává nástroj v tomto kritériu hodnocení 0.
- KF10 – Posledním kritériem z kategorie funkcionalit je *Kompatibilita se soubory z jiných aplikací (import/export)*. Postman v tomto ohledu mírně zaostává, a i když nástroj SoapUI dokáže importovat kolekce z nástroje Postman, obráceně to nefunguje. Postman je tedy schopný exportovat vlastní kolekce ovšem nikoliv importovat soubory z jiných nástrojů. Existuje možnost poněkud krkolomného získání některých dat ze SoapUI pomocí exportu do swagger formátu, ovšem nejedná se tedy o podporu jiných nástrojů, ale pouze souborů swagger, které nástroje standardně podporují. Z tohoto důvodu u tohoto kritéria získává nástroj hodnocení 1.

Další kategorií kritérií je *Uživatelské prostředí*. Tato kategorie obsahuje 5 kritérií, která slouží k vyhodnocení prostředí aplikace.

- KU1 – Prvním kritériem kategorie je *Nutnost znalosti programování pro užití nástroje k testování*. Nástroj Postman umožňuje automatizaci pomocí specifikace testů v JavaScriptu. Ačkoliv nástroj poskytuje ukázky základních příkazů jako jsou aserce, případné úpravy probíhají zásahy do kódu JavaScriptu, nikoliv přes UI rozhraní. Z tohoto důvodu je možná automatizace ovšem alespoň se základními znalostmi programování. Pokročilé schopnosti díky případným vzorům nejsou pro většinu automatizace potřeba. Toto kritérium je tedy hodnoceno 2 body.
- KU2 – Druhým kritériem této kategorie je *Jednoduchost a možnosti instalace nástroje*. Možnosti instalace u nástroje Postman (alespoň na testovaném systému Windows) jsou takřka nulové. Po spuštění instalátoru se pouze zobrazí informace o tom, že se nástroj instaluje, bez jakékoliv možnosti volby umístění či instalovaných komponent. Instalace byla testována u verze 7.0.7 a balíček měl 72 MB. Ačkoliv instalace byla rychlá a proběhla bez problémů, nemožnost specifikace alespoň základních věcí při instalaci získaly aplikaci u tohoto kritéria hodnocení 0.
- KU3 – Dalším z kritérií je *Vzhled uživatelského rozhraní*. Aplikace je přívětivá a přehledná. Veškeré přechody v aplikaci fungují plynule a struktura okna je

intuitivní. Přidanou hodnotou je u této aplikace její moderní design, který působí dobře a aktuálně. Vzhledem k nulovým výhradám, co se do přívětivosti designu týče, získal Postman v tomto kritérium hodnocení 5.

- KU4 – Čtvrtým z kritérií jsou *Možnosti nastavení nástroje*. Postman má nastavení dostupné jak z menu, tak i z hlavní obrazovky. Nastavení je moderně a intuitivně organizované bez jakýchkoliv uživatelských omezení. Z tohoto důvodu je tedy toto kritérium hodnoceno 5 body.
- KU5 – Posledním kritériem kategorie jsou *Možnosti přizpůsobení vzhledu nástroje*. Z hlediska přizpůsobení vzhledu nabízí nástroj mimo změny základního chování některých částí okna i kompletní změnu vzhledu pomocí motivu. Z tohoto důvodu získal program tedy v tomto kritériu hodnocení 5.

Poslední z hodnocených kategorií nástroje Postman je *Dostupnost*. Kategorie zahrnuje 5 kritérií, které analyzují dostupnost aplikace.

- KD1 – Prvním hodnoceným kritériem této kategorie je *Typ licence*. Postman nabízí několik variant produktů. V testované verzi se jedná o nástroj bezplatný, ovšem ačkoliv je to freeware, nejedná se o nástroj Open Source, tj. možnosti rozšíření či úprav možností jsou limitovány. Díky této skutečnosti je nástroj hodnocen 3 body.
- KD2 – Druhým kritériem kategorie *Dostupnost* je *Jazyk / lokalizace*. Postman je kompletně v angličtině a nenabízí oficiální ani neoficiální cesty lokalizace nástroje do češtiny ani jiných jazyků. Proto je toto kritérium hodnoceno 0 body.
- KD3 – Dalším z kritérií je *Nápověda / dokumentace*. Nástroj Postman obsahuje obsáhlé možnosti nápovědy i dostupné dokumentace. Od instalace programu po možnosti případů užití, vše lze dohledat v dokumentaci nástroje. Proto je toto kritérium s hodnocením 5.
- KD4 – Čtvrtým kritériem je *Podpora*. Postman ve své bezplatné verzi nezahrnuje žádné možnosti podpory a spoléhat se uživatel může v případě problému pouze na komunitní fórum. Z tohoto důvodu je toto kritérium hodnoceno 0 body.
- KD5 – Posledním kritériem kategorie je *Uživatelská komunita*. Komunitní fórum v případě bezplatné verze nástroje Postman nahrazuje chybějící možnosti podpory a komunita fóra skýtá velké množství aktivních členů. Z tohoto důvodu je toto kritérium hodnoceno 5 body.

Výsledky zhodnocení kritérií jsou nyní dosazeny do tabulky, ve které je vypočítáno celkové hodnocení pomocí vah kategorií i kritérií. V další podkapitole je poté tato tabulka užita pro porovnání nástrojů SoapUI a Postman.

Tab. 6 - hodnocení kritérií pro Postman (vlastní tvorba)

Tabulka hodnocení				
Kategorie	Kritérium	Hodnocení	Váha	Celkem
Funkcionalita	KF1	3	10/55	0,55
	KF2	3	9/55	0,49
	KF3	5	8/55	0,73
	KF4	5	7/55	0,64
	KF5	2	6/55	0,22
	KF6	5	5/55	0,45
	KF7	3	4/55	0,22
	KF8	1	3/55	0,05
	KF9	0	2/55	0
	KF10	1	1/55	0,02
			Celkem:	3,36
Uživatelské prostředí	KU1	2	5/15	0,67
	KU2	0	4/15	0
	KU3	5	3/15	1
	KU4	5	2/15	0,67
	KU5	5	1/15	0,33
			Celkem:	2,67
Dostupnost	KD1	3	5/15	1
	KD2	0	4/15	0
	KD3	5	3/15	1
	KD4	0	2/15	0
	KD5	5	1/15	0,33
			Celkem:	2,33
Celkové hodnocení:				**2,96**

Z tabulky Tab. 6 vyplývá, že aplikace Postman získala dle vyhodnocení daných kritérií a vah celkové hodnocení 2,33 z 5.

6.3 Porovnání nástrojů

V předchozích dvou podkapitolách byla dříve stanovená kritéria použita na hodnocení jednotlivých nástrojů SoapUI a Postman. Tento základ nyní poslouží ke srovnání těchto dvou nástrojů mezi sebou. Pro základní orientaci v rozdílech porovnání je nejdříve sestavena tabulka kombinující hodnoty z jednotlivých nástrojů vedle sebe. Ta je dále vysvětlena a rozdíly mezi kritérii a následným hodnocením přiblíženy.

Tab. 7 – srovnání hodnocení nástrojů SoapUI a Postman (vlastní tvorba)

Tabulka hodnocení						
Kategorie	Kritérium	Hodnocení SoapUI	Hodnocení Postman	Váha	Celkem SoapUI	Celkem Postman
Funkcionalita	KF1	5	3	10/55	0,91	0,55
	KF2	5	3	9/55	0,82	0,49
	KF3	5	5	8/55	0,73	0,73
	KF4	5	5	7/55	0,64	0,64
	KF5	2	2	6/55	0,22	0,22
	KF6	5	5	5/55	0,45	0,45
	KF7	0	3	4/55	0	0,22
	KF8	1	1	3/55	0,05	0,05
	KF9	3	0	2/55	0,11	0
	KF10	4	1	1/55	0,07	0,02
				Celkem:	4	3,36
Uživatelské prostředí	KU1	3	2	5/15	1	0,67
	KU2	5	0	4/15	1,33	0
	KU3	3	5	3/15	0,6	1
	KU4	5	5	2/15	0,67	0,67
	KU5	3	5	1/15	0,2	0,33
				Celkem:	3,8	2,67

Dostupnost	KD1	5	3	5/15	1,67	1
	KD2	0	0	4/15	0	0
	KD3	5	5	3/15	1	1
	KD4	0	0	2/15	0	0
	KD5	5	5	1/15	0,33	0,33
				Celkem:	3	2,33
Celkové hodnocení SoapUI / Postman:					**3,77**	**2,96**

V tabulce Tab. 7 jsou zachyceny rozdíly mezi hodnocením jednotlivých nástrojů, řádky s odlišnými hodnotami jsou pro přehlednost zvýrazněny. Jednotlivé rozdíly jsou níže vysvětleny, a to podle kategorií kritérií.

V první kategorii, tj. kategorii *Funkcionalita,* je mezi nástroji rozdíl v 5 z 10 kritérií. První z kritérií, ve kterém se nástroje liší, je KF1. Nástroj Postman v něm ztrácí díky nemožnosti automatizace mimo nástroj bez pomoci dodatečných aplikací, což SoapUI zvládá. Druhým rozdílem mezi aplikacemi je kritérium KF2, kde Postman postrádá možnost integrace i s jinými nástroji, než jsou nástroje pro průběžnou integraci. Na druhé straně nástroj SoapUI toto umožňuje. Dalším rozdílem v této kategorii je kritérium KF7, kde pro změnu nástroj Postman získává více bodů díky alespoň základní možnosti sdílení práce v týmu, zatímco SoapUI nemá v této oblasti žádné možnosti. Rozdíl je také v kritériu KF9, kde SoapUI díky možnosti dalšího typu testování převažuje nad nástrojem Postman, který zvládá pouze funkční testy. Posledním rozdílem v této kategorii je kritérium KF10, kde Postman nabízí pouze možnost exportu souborů, zatímco SoapUI umožňuje i import právě z konkurenčního nástroje Postman. V této kategorii je tedy 5 rozdílných kritérií mezi nástroji, z čehož pouze 1 je ve prospěch nástroje Postman. Tomu odpovídá i hodnocení kategorie, které dopadlo lépe u nástroje SoapUI.

Druhá kategorie, tj. kategorie *Uživatelské prostředí*, obsahuje rozdíl ve 4 z 5 kritérií. Prvním rozdílem je kritérium KU1, kde nástroj SoapUI má lepší hodnocení díky možnosti alespoň základní automatizace bez znalosti programování. Postman zde předpokládá alespoň základy programování. Dalším rozdílem je kritérium KU2, které se zabývá instalací, kde nástroj Postman nenabízí uživateli žádné možnosti, zatímco nástroj SoapUI umožňuje specifikovat veškeré důležité parametry během instalace. Třetím rozdílem je v této kategorii kritérium KU3, které řeší design nástroje, kde pro změnu získává lepší hodnocení nástroj Postman, který má prostředí modernější oproti zastaralému vzhledu nástroje SoapUI. Posledním rozdílem v této kategorii je kritérium KU5, kde získává navrch taktéž nástroj Postman díky možnosti změny motivu celého okna, oproti pouze možnosti základních změn v nástroji SoapUI. V této kategorii, kde byl rozdíl ve 4 kritériích, byly 2 kritéria ve prospěch každého z nástrojů, ovšem v případě nástroje SoapUI to bylo ve 2

nejdůležitějších kritériích kategorie, čemuž odpovídá jeho vyšší stav v celkovém hodnocení kategorie.

Poslední kategorií je kategorie *Dostupnost*, kde se mezi nástroji nachází pouze jeden rozdíl, a to v kritériu zabývajícím se typem licence. Zde získává navrch nástroj SoapUI, který má díky Open Source licenci pro uživatele více možností, zatímco Postman nabízí taktéž bezplatný produkt, ovšem nikoliv Open Source. Díky tomuto rozdílu v kategorii vede nástroj SoapUI.

Vzhledem k tomu, že ve všech 3 kategoriích má navrch nástroj SoapUI, i celkové hodnocení vychází samozřejmě ve prospěch tohoto nástroje. Konkrétní hodnoty celkového hodnocení jsou dále rozebrány v následující podkapitole.

6.4 Výsledek porovnání nástrojů

Jak je patrné z rozebraných rozdílů mezi nástroji v předchozí podkapitole, nástroj SoapUI vyšel v porovnání lépe nežli nástroj Postman. Lépe vyšel ve všech 3 kategoriích porovnání, s různě velkými rozdíly mezi celkovým hodnocením kategorie.

V nejdůležitější z kategorií, tj. kategorii zabývající se funkcionalitou získal nástroj SoapUI hodnocení 4 z 5, zatímco konkurenční nástroj Postman získal pouze 3,36 z 5. Ačkoliv je tento rozdíl nejmenší z rozdílů mezi kategoriemi hodnocení, díky důležitosti této kategorie tvoří největší rozdíl v celkovém hodnocení.

V dalších kategoriích, tj. kategoriích zabývajících se uživatelským prostředím a dostupností, získal nástroj SoapUI hodnocení 3,8 a 3, zatímco nástroj Postman pouze 2,67 a 2,33. Ani tyto méně důležité kategorie tedy nástroji Postman výsledek nevylepšily.

Díky výsledku tohoto srovnání je, jako aplikace pro názorné použití pro testování REST API v poslední kapitole, vybrán nástroj SoapUI.

7 Užití vybraného nástroje pro testování konkrétní aplikace

V této kapitole je znázorněno užití testovacího nástroje, který vzešel z porovnání v předchozí kapitole. Jedná se tedy o nástroj SoapUI, který byl díky důležitým výhodám oproti svému konkurentovi vybrán. Nástroj SoapUI i jeho praktické užití již v některých pracích demonstrováno bylo, tato krátká ukázka tedy spíše demonstruje některé silné stránky, které se projevily při porovnání s nástrojem Postman. K demonstraci je použita reálná aplikace z praxe, jejíž výběr je tématem první podkapitoly.

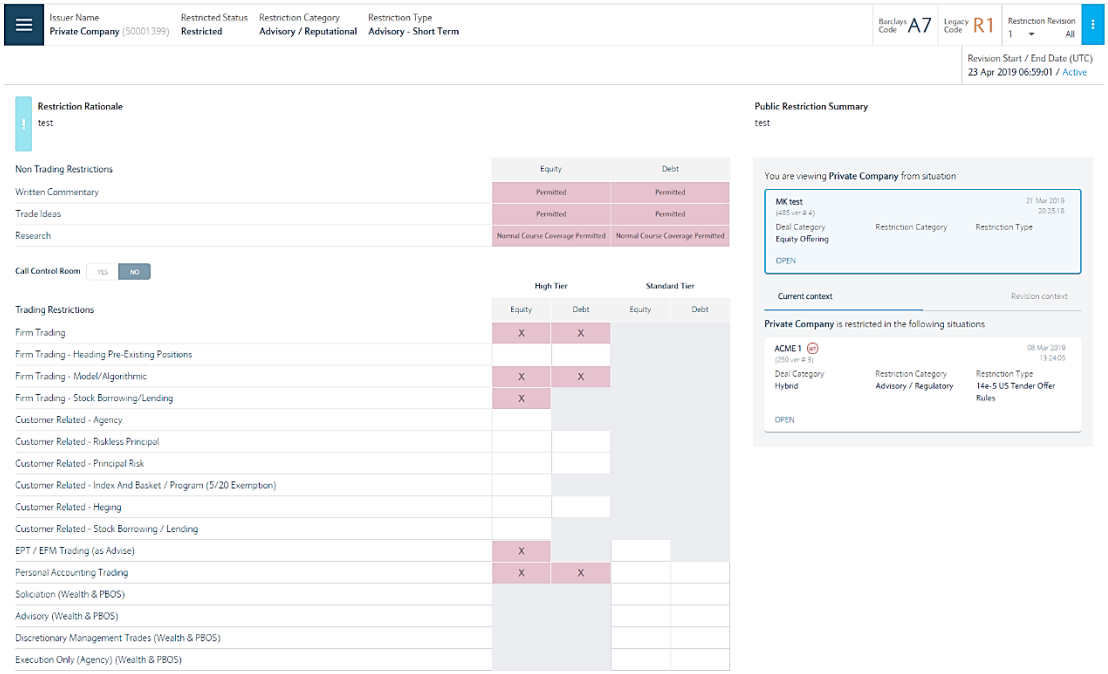
7.1 Výběr vhodné aplikace pro testování REST API

Pro demonstraci vybraného nástroje pro testování REST API, je nejdříve potřeba si zvolit vhodný reálný produkt z praxe. Ke zvolení vhodného produktu na tuto ukázku je důležitých několik faktorů. Především se jedná o faktor dostupnosti REST API, aby ze serveru chodily správné odpovědi, stejně jako to, aby samotný produkt potřebné rozhraní REST API podporoval, a tato demonstrace na něm mohla být provedena.

Vzhledem k aktuálnímu zaměstnání ve společnosti Barclays, tj. nadnárodní finanční skupině, a náplni práce související s produktem, který přes rozhraní REST API komunikuje, byl zvolen tento produkt pro účely této krátké ukázky. Jedná se o produkt Compliance, což je v této společnosti oblast dohlížející na interní dodržování předpisů, a je jí taktéž přezdíváno tzv. „vnitřní policie banky“. Tento konkrétní produkt se zabývá restrikcemi, které jsou uvaleny na obchodování a dohlížejí tedy jak na dodržování předpisů zaměstnanci, tak i na samotné informování zaměstnanců o těchto restrikcích a předejití možnosti jejich záměrnému či náhodnému porušení.

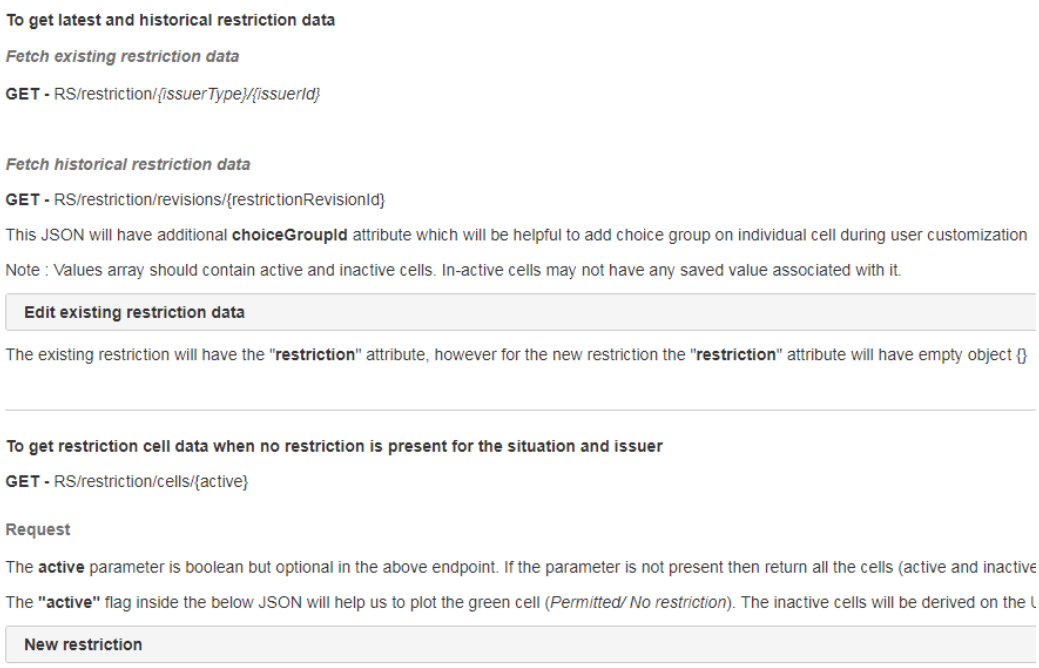
Vzhledem k bezpečnosti, vnitřním nařízením i stálému vývoji produktu, nemohou některé údaje být zmíněny a zároveň musí být senzitivní informace ve screenshotech začerněny, ovšem nejedná se o informace podstatné pro účely demonstrace použití tohoto nástroje pro testování REST API.

Pro ukázku je zde přiložen screenshot z webové aplikace, která získává přes REST API data, která jsou poté poskytována uživatelům.



Obr. 15 - Webová aplikace získávající data pomocí REST API (screenshot z interní aplikace Barclays)

Data, která aplikace skrze REST API získává, jsou získávána i upravována pomocí velké řady požadavků, které REST API umožňuje. Zde je pouze názorná ukázka, jak taková definice REST API metod a jejich popis může v praxi vypadat, a jak u daného produktu vypadá právě ve společnosti Barclays.



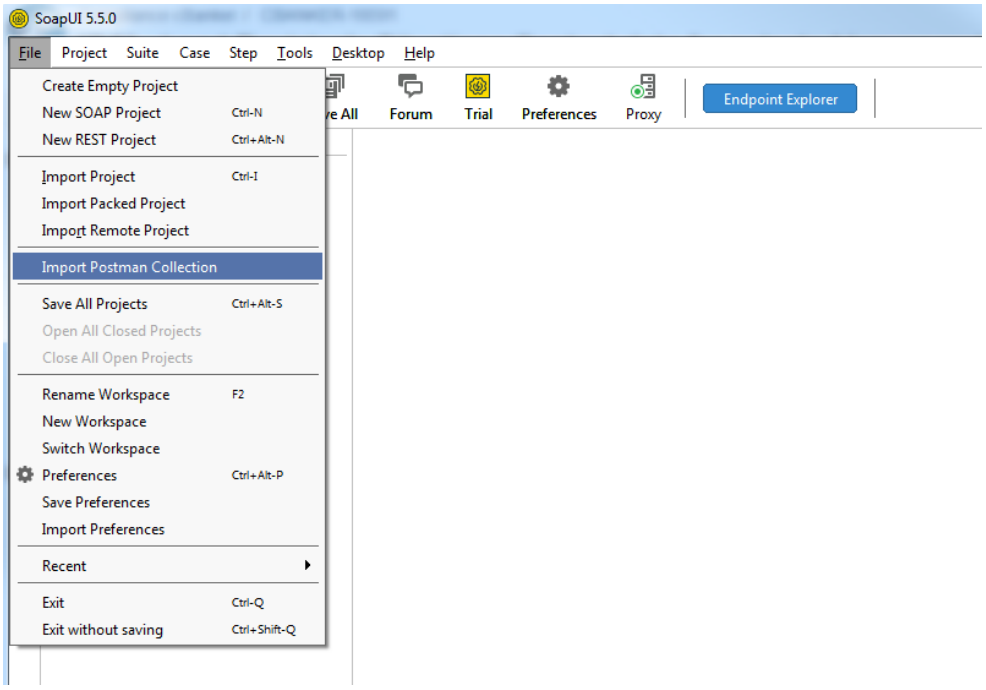
Obr. 16 - Ukázka definice REST API pro uživatele v Barclays (screenshot z interní aplikace)

7.2 Manuální testování pomocí vybraného nástroje

Nyní, když je zvolen nástroj pro testování REST API i aplikace, která bude testována, se již může přejít k samotné ukázce testování. Při této ukázce jsou demonstrovány silné stránky testovacího nástroje, který z porovnání vyplynul, tj. nástroje SoapUI.

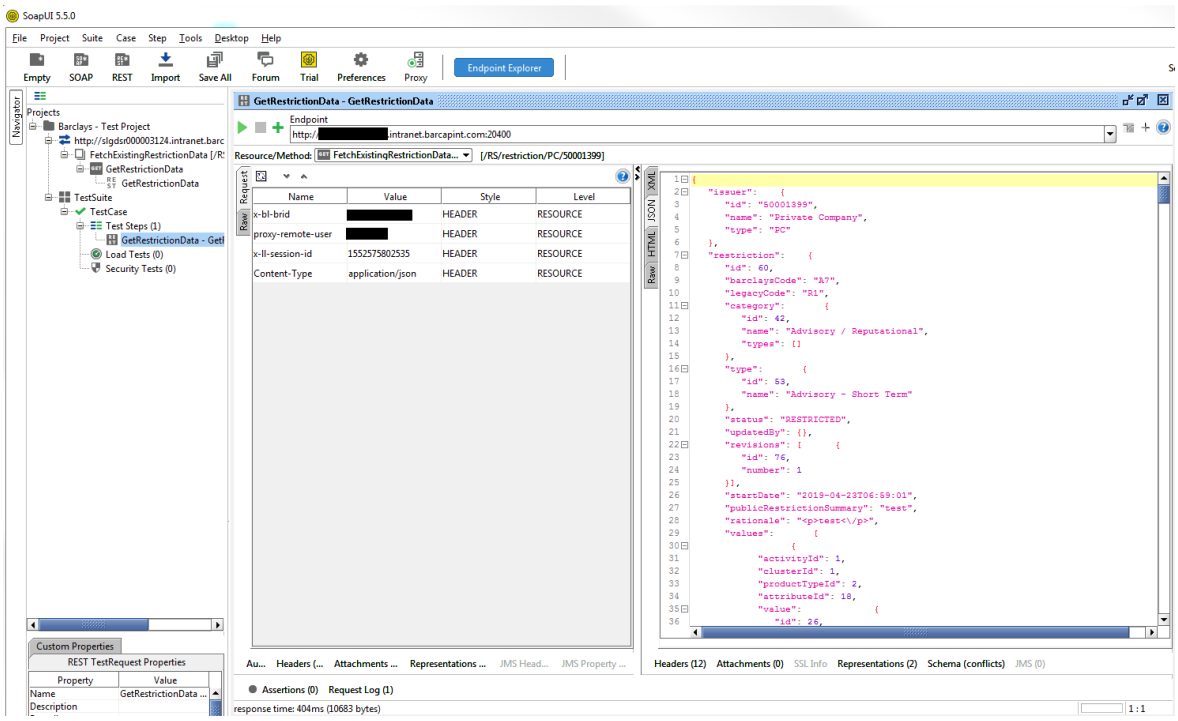
Před samotným testováním je potřeba nástroj nainstalovat, kde se projevuje první z výhod programu. Ve srovnání s konkurenčním nástrojem Postman totiž nástroj SoapUI uživateli umožňuje při instalaci zvolit umístění, komponenty i vytvoření zástupců.

Po instalaci a spuštění programu je možné si při použití nástroje demonstrovat další z výhod nástroje, což je možnost importovat mimo projekty SoapUI, které byly dříve aplikací vytvořeny, i externí soubory právě konkurenčního nástroje Postman, tj. v jeho případě tzv. kolekce.



Obr. 17 - Možnost importovat kolekce Postman v nástroji SoapUI (screenshot ze SoapUI v5.5.0)

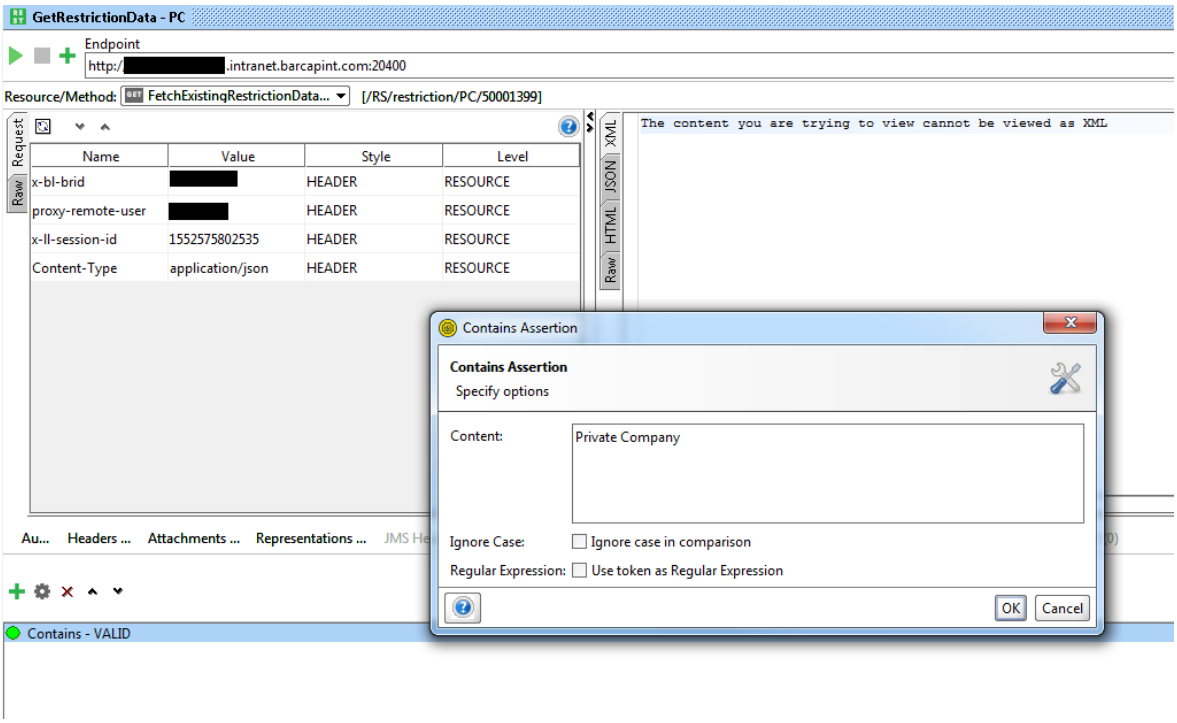
V případě ukázky testování je ovšem pro demonstraci silných stránek nástroje SoapUI zvolena tvorba projektu nového. Po vytvoření nového projektu a nového požadavku, jak je zachyceno i v kapitole 4 věnující se testování nástrojem SoapUI, lze již přejít k vytvoření prvního testu. Toho se docílí tím, že se u požadavku pravým kliknutím myši zvolí vytvoření nového testu.



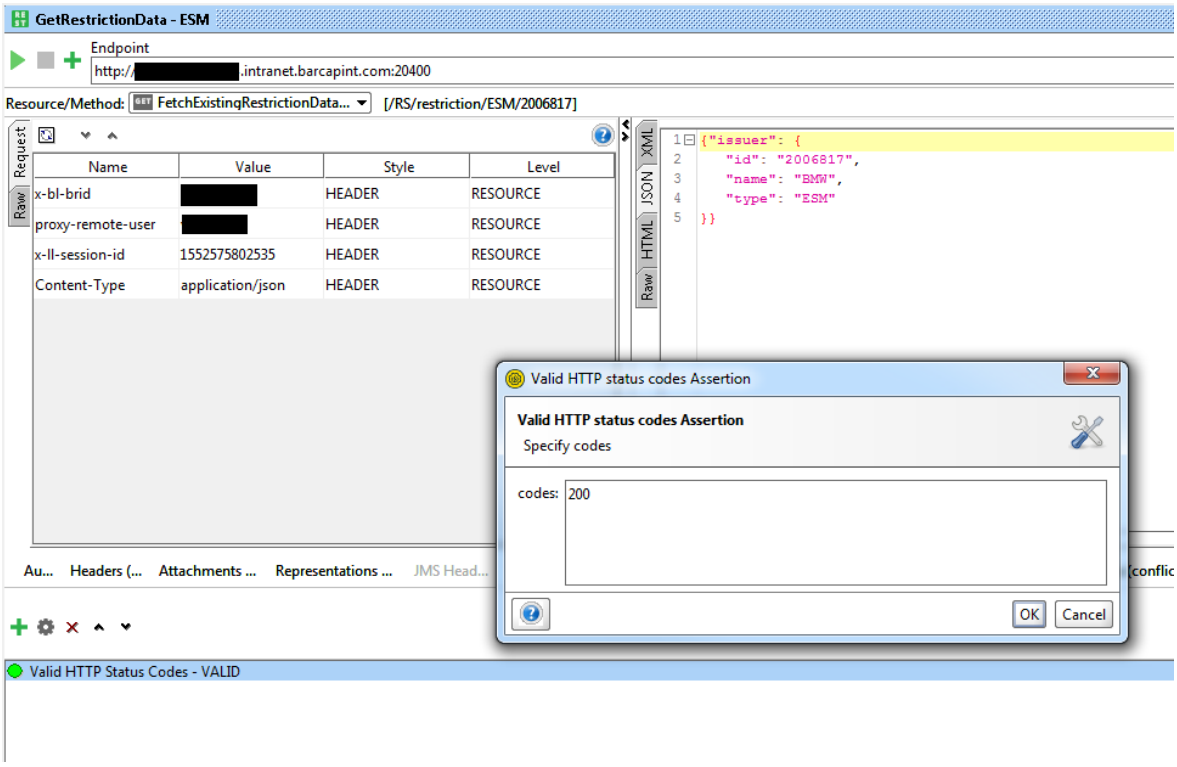
Obr. 18 - Ukázka testu v nástroji SoapUI (screenshot ze SoapUI v5.5.0)

Jak je patrné z Obr. 18 - Ukázka testu v nástroji SoapUI (screenshot ze SoapUI v5.5.0), u testu lze definovat mimo adresu URI zdroje i hodnoty hlaviček, které v tomto případě slouží k autentizaci a specifikaci očekávaného formátu dat. Po definici těchto hodnot lze již zvolit zelený trojúhelník, který spustí test.

Pro ověření správnosti odpovědi a toho, zda dorazila očekávaná data, je potřeba vytvořit aserce, které toto ověří. Pomocí zeleného „+“ je tedy aserce přidána. Zde je vidět další z výhod nástroje, a to, že lze aserci přidat i bez jakéhokoliv kontaktu s kódem a lze ji tedy jednoduše upravovat nezávisle na znalosti programování.



Obr. 19 - Ukázka aserce textu v SoapUI (screenshot ze SoapUI v5.5.0)



Obr. 20 - Ukázka aserce HTTP kódu v SoapUI (screenshot ze SoapUI v5.5.0)

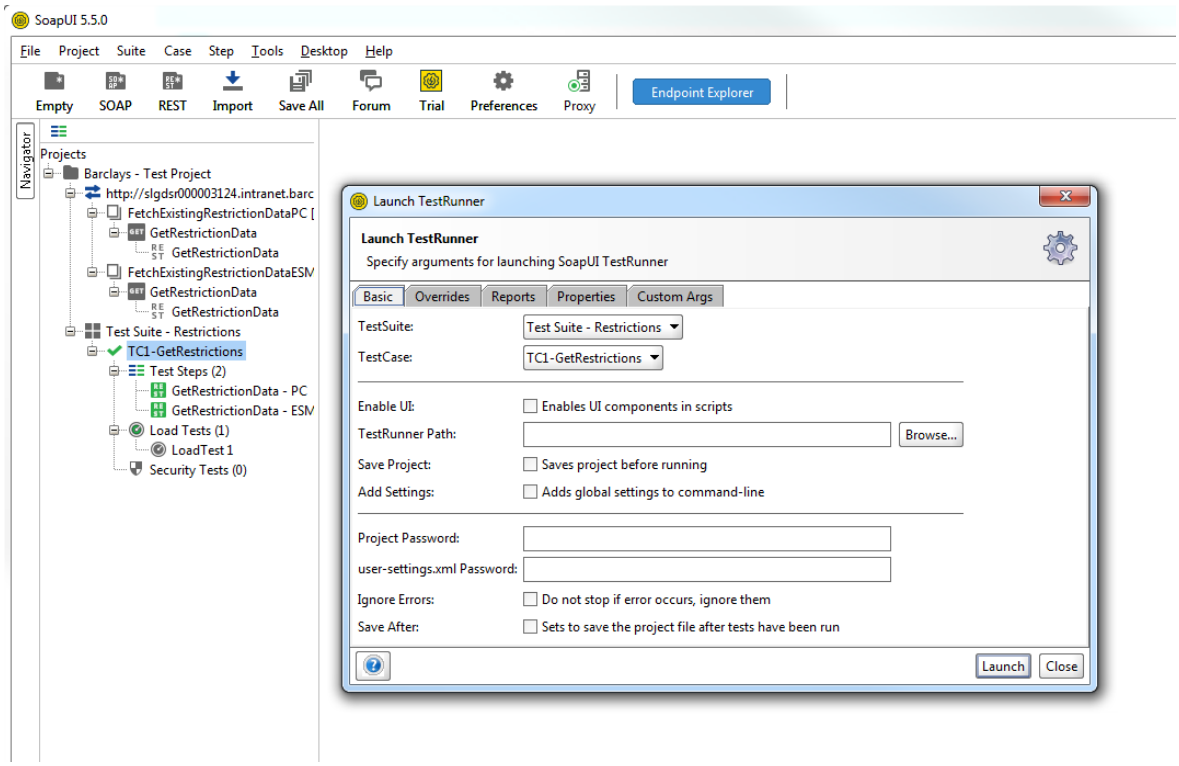
Jak lze vidět na Obr. 19 - Ukázka aserce textu v SoapUI (screenshot ze SoapUI v5.5.0), přidání aserce očekávané textové hodnoty v těle odpovědi na požadavek, je i bez jakékoliv znalosti programování jednoduché a jasné. Stejně tak to demonstruje i aserce úspěšného HTTP kódu 200, která je patrná z Obr. 20 - Ukázka aserce HTTP kódu v SoapUI (screenshot ze SoapUI v5.5.0).

Při tvorbě testovacího scénáře se podařilo tedy demonstrovat silné stránky nástroje a pro účely práce není potřeba demonstrovat podobný postup na zbytku testovacích scénářů. Nyní se v další podkapitole demonstrují výhody programu při automatizaci testování.

7.3 Automatizace testování pomocí vybraného nástroje

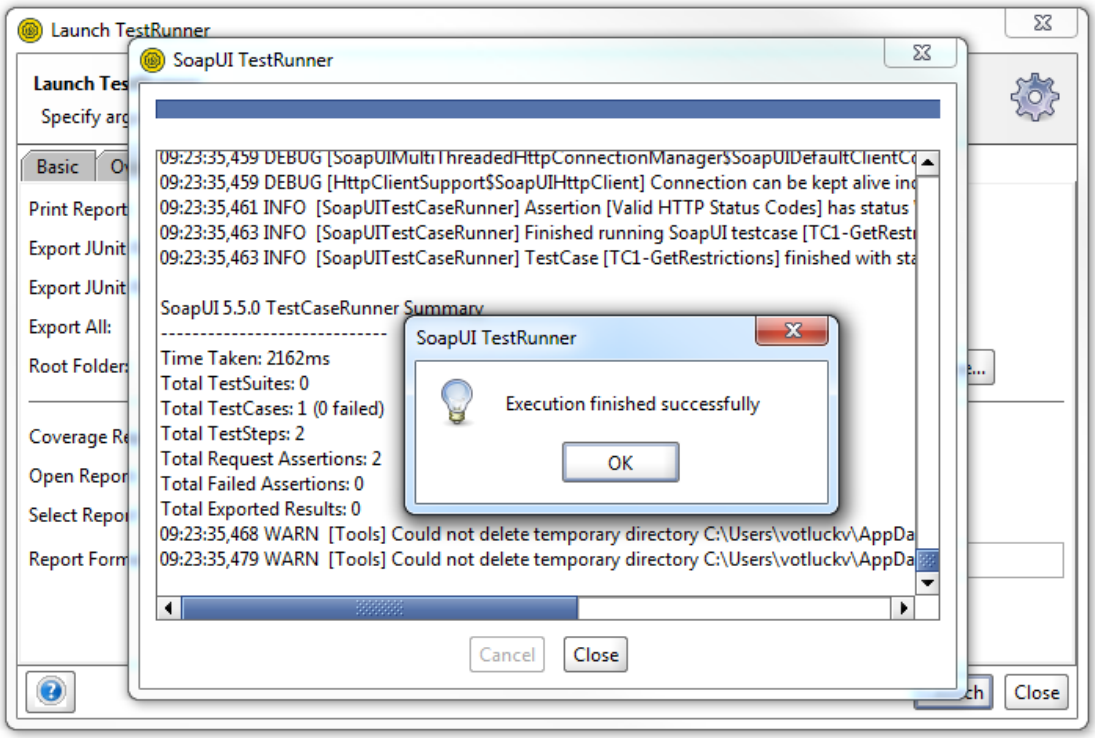
Z hlediska automatizace testování program SoapUI taktéž nezaostává, jak je patrné i z kapitoly 4, která se nástroji věnuje. V kapitole 6 se ukázalo, že ve srovnání s nástrojem Postman má výhody i v této oblasti, a právě ty se demonstrují nyní v této podkapitole.

První a zásadní výhodou programu je, že pro testování v SoapUI i automatizované testy není potřeba žádná znalost kódování a spuštění TestRunneru, který v nástroji SoapUI umožňuje automatizované testování, je velmi jednouché. TestRunner umožňuje specifikovat velkou řadu možností, ovšem základní pro uživatele je především možnost zvolit testovací sady, popř. jednotlivé testovací scénáře, které chce spustit. Nástroj tedy zvládá testy, které již byly manuálně provedeny v předchozí části ukázky, opakovaně spouštět bez jakýchkoliv zásahů.



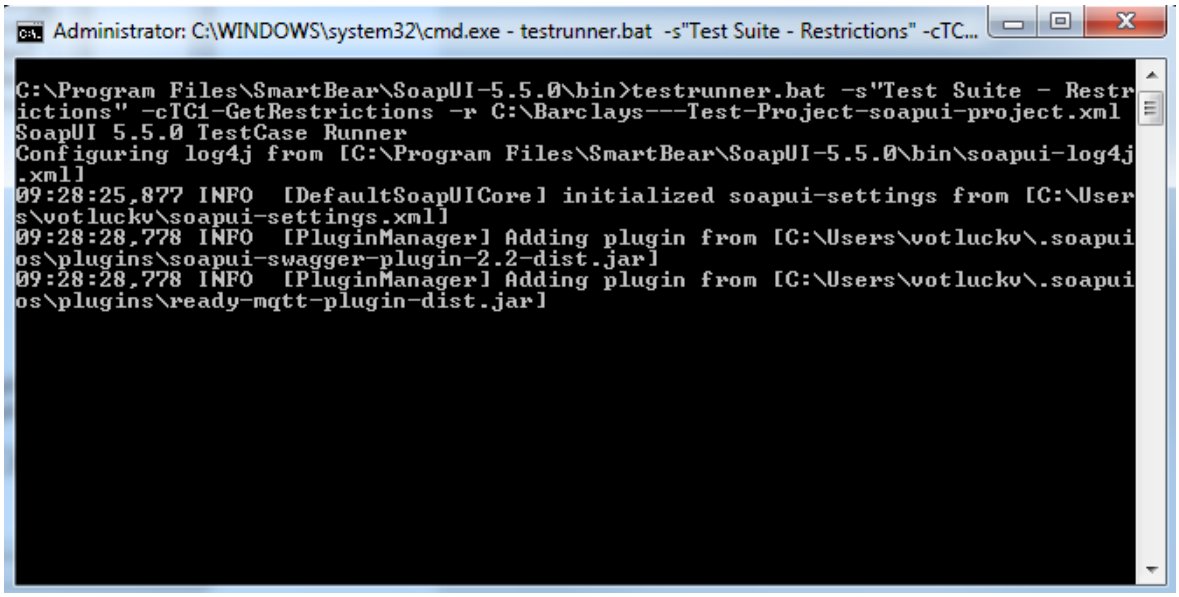
Obr. 21 - Možnosti TestRunneru v SoapUI (screenshot ze SoapUI v5.5.0)

Jak je patrné z Obr. 21 - Možnosti TestRunneru v SoapUI (screenshot ze SoapUI v5.5.0), automatizované testování je možné přímo z nástroje SoapUI a po provedení je možné vidět výsledky přímo v programu. To je vidět na Obr. 22 - Výsledky TestRunneru v SoapUI (screenshot ze SoapUI v5.5.0).



Obr. 22 - Výsledky TestRunneru v SoapUI (screenshot ze SoapUI v5.5.0)

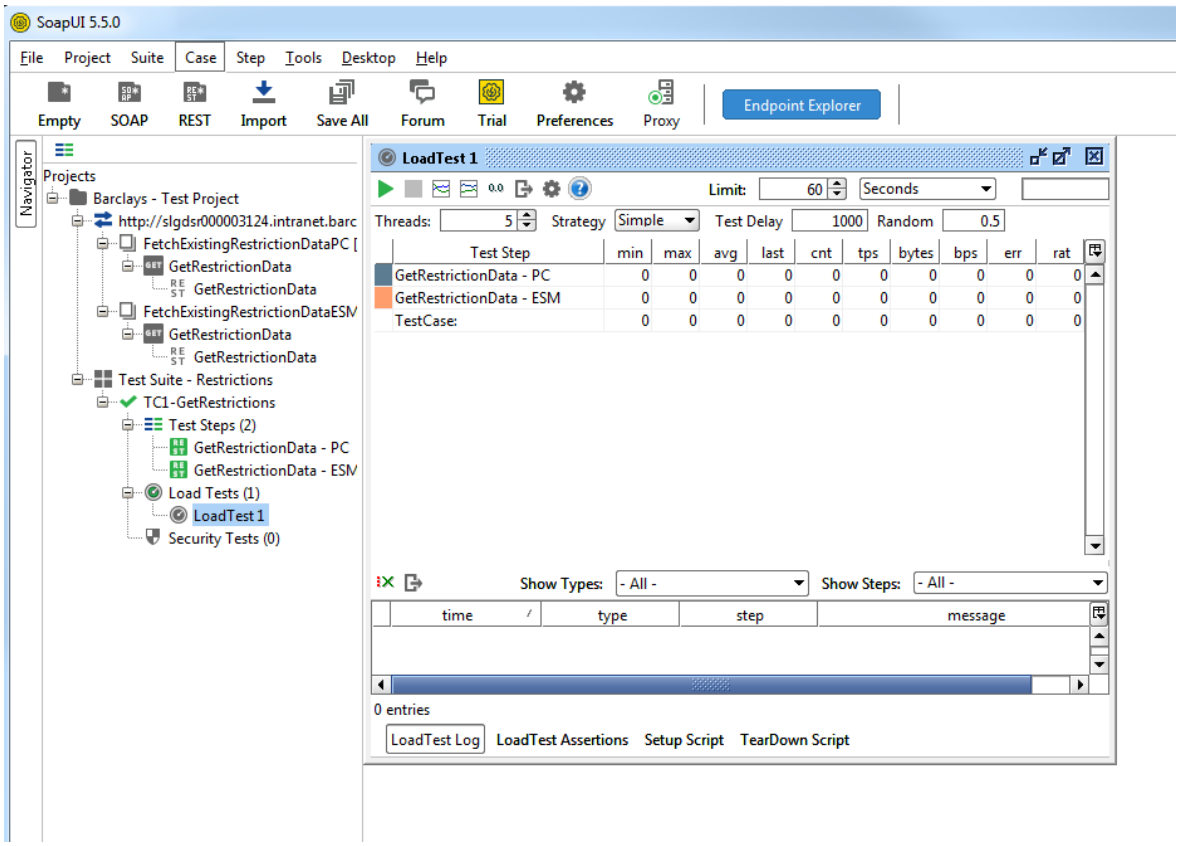
Mimo automatizace v programu je velká výhoda SoapUI i to, že je možné díky skriptům TestRunneru, které jsou součástí základní instalace programu, spustit automatizaci i z příkazové řádky včetně definice velkého množství parametrů. Výsledky jsou viditelné i v příkazové řádce ve stejné formě, jako v případě užití TestRunneru přímo v SoapUI.



Obr. 23 - TestRunner z příkazové řádky (screenshot ze SoapUI v5.5.0)

Poslední silnou stránku nástroje, kterou stojí za to vyzdvihnout, mimo spouštění automatizace z příkazové řádky viditelné na Obr. 23 - TestRunner z příkazové řádky (screenshot ze SoapUI v5.5.0), jsou jeho další možnosti testování.

SoapUI umožňuje z vytvořených funkčních testů i jednoduché vytvoření zátěžových testů, které mohou být taktéž spouštěny i z příkazové řádky. K vytvoření těchto testů stačí v testovací sadě projektu zvolit kategorii *Load Tests,* kde se při kliknutí pravým tlačítkem myši nabídne možnost *New LoadTest*. Při této volbě program automaticky vytvoří zátěžové testy z existujících funkčních testů. K tomu nabízí program velkou řadu parametrů, jako je možnost zvolení strategie, zpoždění testu, randomizace času mezi požadavky či počet instancí v kterých testy poběží. Samozřejmě lze některé testy z testovací sady vynechat a zatěžovat jen žádoucí koncové body serveru.



Obr. 24 - Zátěžové testy v SoapUI (screenshot ze SoapUI v5.5.0)

Jak je z Obr. 24 - Zátěžové testy v SoapUI (screenshot ze SoapUI v5.5.0) patrné, lze samozřejmě nastavit i kroky, které budou provedeny před či po spuštění zátěžových testů, stejně jako dobu, po kterou testy poběží. V případě důležitých aplikací, což je i případ testované bankovní aplikace, je simulace zátěže vítanou funkcionalitou, a možnost provádět tyto testy pomocí stejného nástroje velké plus.

7.4 Zhodnocení vybraného nástroje při testování

Úkolem ukázky zvoleného nástroje pro testování REST API byla demonstrace možností nástroje SoapUI, konkrétně tedy především silných stránek testovacího nástroje, které vyplynuly ze srovnání nástroje SoapUI vůči nástroji Postman.

Pro testování byla zvolena aplikace z bankovní sféry, která testování REST API vyžaduje a je byla pro účely této práce dostupná. Při testování nebyly zjištěny žádné důvody, které by využití nástroje SoapUI bránily, ani slabé stránky, které by nebyly při představení nástroje či následném porovnání nástrojů odhaleny.

Vzhledem k tomu, že v případě bankovní aplikace je potřeba využít i zátěžových testů, to, že tuto funkcionalitu program také nabízí, může společnosti ušetřit dodatečné náklady. Šetří to totiž nejen čas na zaškolování uživatelů na další aplikace, ale i prostředky, které by musely být vynaloženy na údržbu dodatečných aplikací či úhradu licencí v případě placených aplikací.

Při užití nástroje se tedy podařilo ověřit silné stránky programu, které ze srovnání vyplynuly a zároveň proces zdokumentovat, což umožní využít části ukázky i jako základní návod při tvorbě testů v nejnovější verzi programu SoapUI. Jako zevrubné příručky pro testování pomocí nástroje SoapUI ovšem slouží především jiné práce, jelikož to není cílem této práce.

Závěr

Tato diplomová práce měla za cíl porovnat dva rozšířené nástroje pro testování rozhraní REST API, jmenovitě se jedná o nástroje SoapUI a Postman. Tento hlavní cíl práce byl rozdělen na dílčí kritéria, která vedla k naplnění tohoto primárního cíle.

K naplnění hlavního cíle i cílů dílčích, práce nejdříve představila oblast REST API, a to konkrétně v 2. kapitole práce, což poskytlo čtenáři náhled do problematiky před podrobnějším řešením konkrétních specifik této oblasti. Mimo analýzu trhu a vysvětlení základních pojmů, poskytla tato kapitola také základ pro naplnění cílů v následujících kapitolách.

V kapitole 3. byly použité pomocné zdroje, které se zabývají problematikou srovnávání či REST API. Analýza těchto pomocných zdrojů společně se znalostí problematiky z praxe, pomohly naplnit 3. dílčí cíl, tj. definovat kritéria pro porovnání nástrojů. V této kapitole tedy vznikla kritéria, která byla v práci dále aplikována a užita pro naplnění jejich dalších dílčích cílů. Mimo definici kritérií byly za pomoci odborných zdrojů stanoveny i váhy a hodnocení aplikované na tato kritéria.

V kapitolách 4 a 5 byly představeny nástroje SoapUI a Postman, tedy oba ze srovnávaných nástrojů pro testování REST API touto prací. Jednalo se o představení aktuálních funkcí a možností nástrojů. Mimo představení nástrojů byly v této kapitole zmíněny i výhody a nevýhody nástrojů, které byly poté při samotném porovnání demonstrovány. Tyto dvě kapitoly tedy vedly k naplnění 1. dílčího cíle, tj. představení obou nástrojů SoapUI a Postman.

Kapitoly 4 a 5 zároveň analyzovaly možnosti nástrojů SoapUI i Postman v oblasti testování oblasti REST API. Analyzovány byly možnosti jak manuálního, tak i automatizovaného testování. Tato analýza sloužila i jako základ pro samotné porovnání nástrojů, které ve svých kritériích srovnávalo i tyto možnosti. Tímto práce naplnila 2. dílčí cíl, tj. analýzu možností použití nástrojů pro testování REST API.

V 6. kapitole byla definovaná kritéria použita pro porovnání obou nástrojů, a na základě informací z této práce i několikaleté praxe v oboru testování REST API byly nástroje porovnány. Díky definovaným kritériím se podařilo zvolit vhodný nástroj k testování REST API, který v práci poté slouží k naplnění dalšího dílčího cíle. Tato kapitola naplnila 4. dílčí cíl, tj. užití definovaných kritérií k volbě vhodného nástroje.

Poslední kapitolou práce, tj. 7., je užití vybraného nástroje pro testování konkrétní aplikace. V této kapitole byl použit vybraný nástroj, který vzešel ze srovnání v 6. kapitole, a to na základě stanovených kritérií v kapitole 3. Tato krátká ukázka nástroje v praxi demonstrovala některé ze silných stránek testovacího nástroje, tj. kritéria na základě kterých byl nástroj vybrán. Tato ukázka byla provedena na aplikaci z reálného světa, konkrétně na testování REST API jednoho z produktů nadnárodní finanční instituce

Barclays. Tímto byl tedy naplnění 5. dílčí cíl, který měl za cíl právě užití vybraného nástroje v praxi, čímž demonstroval výhody nástroje oproti konkurenci.

Díky naplnění veškerých dílčích cílů této práce se povedlo splnit i hlavní cíl práce, jímž bylo porovnat dva rozšířené nástroje pro testování rozhraní REST API. Nástroje SoapUI a Postman byly představeny a jejich možnosti analyzovány. Na základě stanovených kritérií byly poté nástroje porovnány a použití vítězného nástroje v praxi bylo poté demonstrováno na ukázce v poslední kapitole práce.

Práce mimo naplnění tohoto cíle poskytuje i prostor pro případné hlubší zpracování dalších témat. Jako základ může sloužit pro podrobnější rozebrání tématu testování REST API i jednotlivých z nástrojů SoapUI a Postman. Přínosem mimo představení těchto témat a aktuální analýzy problematiky je i to, že práce a definovaná kritéria k porovnání mohou sloužit i ke srovnání jiných nástrojů pro testování REST API.

Použitá literatura

1. **Google.** Google Trends. [Online] 2019. [Citace: 1. Únor 2019.] https://trends.google.com/trends/explore?date=2014-02-01%202019-02-01&q=REST%20API.

2. **Fielding, Roy Thomas.** Architectural Styles and the Design of Network-based Software Architectures. Irvinge : Disertační práce, University of California, 2000.

3. **SmartBear Software.** Open Source. *SoapUI*. [Online] 2019. [Citace: 15. Únor 2019.] https://www.soapui.org/open-source.html.

4. **Postman.** Company. *Postman*. [Online] 2019. [Citace: 15. Únor 2019.] https://www.getpostman.com/company.

5. **Google.** Google Trends. [Online] 2019. [Citace: 1. Duben 2019.] https://trends.google.com/trends/explore?date=2016-04-01%202019-04-01&q=SoapUI,Postman%20API.

6. **Google**. Google Scholar. [Online] 2019. [Citace: 20. Únor 2019.] https://scholar.google.cz/intl/cs/scholar/about.html.

7. **Masarykova univerzita.** Theses.cz. [Online] 2019. [Citace: 20. Únor 2019.] https://theses.cz/.

8. **Fertig, Tobias and Braun, Peter.** *Model-driven Testing of RESTful APIs*. New York : ACM, 2015. WWW '15 Companion Proceedings of the 24th International Conference on World Wide Web. 1497-1502. 978-1-4503-3473-0.

9. **Kankanamge, Charitha.** *Web services testing with soapUI build high quality service-oriented solutions by learning easy and efficient web services testing with this practical, hands-on guide*. New Edition. Birmingham : Packt publ, 2012. 9781849515665.

10. *Web Service Testing Tools: A Comparative Study*. **Hussain, Shariq, a další.** 1, Mahebourg : autor neznámý, 2013, IJCSI (International Journal of Computer Science Issues, Sv. 10, stránky 641-647. 1694-0814.

11. **Newstex Trade & Industry Blogs.** API Evangelist: Postman Collections Will Take Your API Productivity To The Next Level. [Online] 9. Březen 2015. [Citace: 1. Březen 2019.] https://search-proquest-com.zdroje.vse.cz/docview/1661577071?accountid=17203.

12. **Sobotka, Petr.** Testování webových služeb nástrojem SoapUI. Praha : Diplomová práce, Vysoká škola ekonomická v Praze, 2015. Vedoucí práce Alena Buchalcevová.

13. **Jirmusová, Radka.** Ověření metodiky Testování webových služeb nástrojem SoapUI. Praha : Diplomová práce, Vysoká škola ekonomická v Praze, 2016. Vedoucí práce Alena Buchalcevová.

14. **Petržilka, Jakub.** Systémové integrační testování s použitím SoapUI. Praha : Diplomová práce, Vysoká škola ekonomická v Praze, 2015. Vedoucí práce Alena Buchalcevová.

15. **Žabka, Michael.** Analýza software testing nástrojů. Praha : Bakalářská práce, Vysoká škola ekonomická v Praze, 2015. Vedoucí práce Miloš Maryška.

16. **Bláha, Dominik.** Srovnání Open Source nástrojů pro správu testů využitelných při vývoji softwaru jednotlivci. Praha : Bakalářská práce, Vysoká škola ekonomická v Praze, 2016. Vedoucí práce Jan Kučera.

17. **Kulíček, Martin.** Srovnání nástrojů pro správu defektů. Praha : Bakalářská práce, Vysoká škola ekonomická v Praze, 2016. Vedoucí práce Jan Hyblbauer.

18. **Haltuf, Ludvík.** Testování REST API. Praha : Diplomová práce, České vysoké učení technické v Praze, 2012. Vedoucí práce Jan Šedivý.

19. *Rozšířené testování REST API.* **Strnádek, Jan a Pejřimovský, David.** Plzeň : Západočeská univerzita v Plzni, 2014. Studentská vědecká konference.

20. *Web service testing tool soapui and its analysis.* **Ming, Luo Zuomin Zhu Yan Cheng.** 47, Shaanxi : Computer Applications and Software, 2010, Sv. 5.

21. *Web services testing challenges and approaches.* **Azzam, Sana, Al-Kabi, Naji a Alsmadi, Izzat.** 2012. Proceedings of the 1st Taibah University International Conference on Computing and Information Technology. stránky 291-296.

22. **Varga, Ervin.** Testing REST APIs. *Creating maintainable apis: a practical, case-study approach.* New York : Springer Science Business Media, 2016, stránky 159-169.

23. **Abran, Alain and Moore, James W.** *Guide to the Software Engineering Body of Knowledge.* Los Alamitos : IEEE Computer Society, 2004. 0-7695-2330-7.

24. **Borovcová, Anna.** *Testování webových aplikací.* Praha : Diplomová práce, Univerzita Karlova v Praze, 2008. Vedoucí práce Vladan Majerech.

25. **Stowe, Michael.** *Undisturbed REST.* San Francisco : MuleSoft, 2015. 978-1-329-11656-6.

26. **Fielding, et al.** Hypertext Transfer Protocol -- HTTP/1.1. *Internet Engineering Task Force.* [Online] 1999. [Citace: 10. Březen 2019.] https://tools.ietf.org/html/rfc2616.

27. **Gourley, David, a další.** *HTTP: The Definitive Guide.* Sebastopol : O'Reilly Media, 2002. 9781565925090.

28. **Richta, Karel.** Standardy WSDL a UDDI. *Matematicko-fyzikální fakulta Univerzita Karlova.* [Online] 2003. [Citace: 11. Březen 2019.] https://www.ksi.mff.cuni.cz/~richta/publications/Richta-MD-2003.pdf.

29. **Baskirt, Onur.** *SW Test Academy.* [Online] 2018. [Citace: 21. Březen 2019.] https://www.swtestacademy.com/rest-assured-tutorial-api-testing/.

30. **Guru99.** API Testing. *Guru99.* [Online] 2019. [Citace: 21. Březen 2019.] https://www.guru99.com/api-testing.html.

31. **SmartBear Sofware.** API Mocking. *SoapUI.* [Online] 2019. [Citace: 29. Březen 2019.] https://www.soapui.org/learn/mocking/what-is-api-mocking.html.

32. **Medium.com.** Top 10 API Testing Tools. *Medium.com.* [Online] 2019. [Citace: 19. Leden 2019.] https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9.

33. **SmartBear Software.** Documentation. *SoapUI.* [Online] 2019. [Citace: 23. Březen 2019.] https://www.soapui.org/soapui-projects/soapui-projects.html.

34. **Postman.** Postman Learning Center. [Online] 2019. [Citace: 22. Březen 2019.] https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/.

35. **Kváček, Michal.** Webová služba na testování REST API. Praha : Bakalářská práce, České vysoké učení technické v Praze, 2017. Vedoucí práce Jiří Hunka.

36. **ToolsQA.** Top 5 API Testing Tools. *ToolsQA.* [Online] 2019. [Citace: 13. Únor 2019.] http://toolsqa.com/blogs/top-api-testing-tools-review/.

37. **Guru99.** Introduction to SoapUI. *Guru99.* [Online] 2019. [Citace: 13. Únor 2019.] https://www.guru99.com/introduction-to-soapui.html.

38. **Toptal.** REST Assured vs. JMeter: A Comparison of REST Test Tools. *Toptal.* [Online] 2019. [Citace: 2. Březen 2019.] https://www.toptal.com/java/rest-assured-jmeter.

39. **Doubravová, Hana.** Vícekriteriální analýza variant a její aplikace v praxi. České Budějovice : Diplomová práce, Jihočeská univerzita v Českých Budějovicích, 2009. Vedoucí práce Jana Friebelová.

40. **Buchalcevová, Alena.** *Zlepšování procesů při budování informačních systémů.* Praha : Vysoká škola ekonomická v Praze, Nakladatelství Oeconomica, 2018. 978-80-245-2235-7.

41. **SmartBear Software.** Pricing. *SoapUI.* [Online] 2019. [Citace: 1. Duben 2019.] https://www.soapui.org/professional/soapui-pro/pricing.html.

42. **SmartBear Software**. Downloads. *SoapUI.* [Online] 2019. [Citace: 3. Březen 2019.] https://www.soapui.org/downloads/soapui.html.

43. **SmartBear Software**. Compare SoapUI OS vs SoapUI Pro. *SoapUI.* [Online] 2019. [Citace: 19. Březen 2019.] https://www.soapui.org/professional/soapui-pro/compare-os-to-pro.html.

44. **SmartBear Software**. API Testing. *SoapUI*. [Online] 2019. [Citace: 30. Březen 2019.] https://smartbear.com/solutions/api-testing/.

45. **Richardson, Alan.** *Automating and Testing a REST API: A Case Study in API testing using: Java, REST Assured, Postman, Tracks, cURL and HTTP Proxies*. 1. Watford : Compendium Developments, 2017. stránky 50-57. 9780956733290.

46. **Postman.** Plans & Pricing. [Online] 2019. [Citace: 30. Březen 2019.] https://www.getpostman.com/pricing.

47. **SmartBear Sofware.** SmartBear Community. *SoapUI*. [Online] 2019. [Citace: 1. Duben 2019.] https://community.smartbear.com/.

48. **Fielding, R. a Reschke, J.** RFC 7231. [Online] 2014. [Citace: 12. Leden 2019.] https://tools.ietf.org/html/rfc7231. 2070-1721.

49. **W3C.** Web Application Description Language. [Online] 2009. [Citace: 2. Duben 2019.] https://www.w3.org/Submission/wadl/.

50. **Microsoft.** CalculatorWsdl. *Microsoft*. [Online] 2018. [Citace: 1. Únor 2019.] https://docs.microsoft.com/en-us/windows/desktop/wsw/calculatorwsdl.

Seznam obrázků

Obr. 1 - Trend termínu REST API za 5 let [1] 8
Obr. 2 - Trend nástrojů SoapUI a Postman za 3 roky [5] 9
Obr. 3 – Vytvoření projektu v menu SoapUI (screenshot ze SoapUI v5.5.0) 46
Obr. 4 - Zadání URI při založení projektu v SoapUI (screenshot ze SoapUI v5.5.0) 47
Obr. 5 – Požadavek v SoapUI a odpověď API (screenshot ze SoapUI v5.5.0) 47
Obr. 6 - Tvorba testovacího případu v SoapUI (screenshot ze SoapUI v5.5.0) 48
Obr. 7 - Přidání aserce do testu v SoapUI (screenshot ze SoapUI v5.5.0) 48
Obr. 8 - Specifikace aserce v testu v SoapUI (screenshot ze SoapUI v5.5.0) 49
Obr. 9 - Úspěšný test v SoapUI (screenshot ze SoapUI v5.5.0) 49
Obr. 10 - Tvorba kolekce v Postman (screenshot z Postman v7.0.9) 55
Obr. 11 - Přidání požadavku v Postman (screenshot z Postman v7.0.9) 56
Obr. 12 - Odpověď na požadavek v Postman (screenshot z Postman v7.0.9) 56
Obr. 13 - Tvorba testu v Postman (screenshot z Postman v7.0.9) 57
Obr. 14 - Výsledek testu v Postman (screenshot z Postman v7.0.9) 57
Obr. 15 - Webová aplikace získávající data pomocí REST API (screenshot z interní aplikace Barclays) 70
Obr. 16 - Ukázka definice REST API pro uživatele v Barclays (screenshot z interní aplikace) 70
Obr. 17 - Možnost importovat kolekce Postman v nástroji SoapUI (screenshot ze SoapUI v5.5.0) 71
Obr. 18 - Ukázka testu v nástroji SoapUI (screenshot ze SoapUI v5.5.0) 72
Obr. 19 - Ukázka aserce textu v SoapUI (screenshot ze SoapUI v5.5.0) 73
Obr. 20 - Ukázka aserce HTTP kódu v SoapUI (screenshot ze SoapUI v5.5.0) 73
Obr. 21 - Možnosti TestRunneru v SoapUI (screenshot ze SoapUI v5.5.0) 74
Obr. 22 - Výsledky TestRunneru v SoapUI (screenshot ze SoapUI v5.5.0) 75
Obr. 23 - TestRunner z příkazové řádky (screenshot ze SoapUI v5.5.0) 75
Obr. 24 - Zátěžové testy v SoapUI (screenshot ze SoapUI v5.5.0) 76

Přílohy

V přílohách práce jsou podpůrné zdroje k pochopení tématu, na které je v práci odkazováno. V příloze A jsou k dispozici stavové kódy HTTP. V příloze B je ukázkový dokument WADL. V příloze C je ukázkový dokument WSDL.

Příloha A: Stavové kódy HTTP

V této příloze je seznam stavových kódu HTTP užívaných při práci s REST API. [48]

- **HTTP kód 200 (OK)**
  Tento stav indikuje, že REST API úspěšně vykonalo akci požadovanou příkazem klienta, a že není žádný vhodnější specifický kód v této kategorii kódů.
  Oproti kódu 204 musí kód 200 obsahovat tělo odpovědi. Informace v odpovědi se liší na základě konkrétní metody, která byla v příkazu použita.
- **HTTP kód 201 (CREATED)**
  Tento kód je v REST API používán vždy, když je vytvořen nový zdroj v kolekci. Stejně tak může být zdroj vytvořen i jako důsledek kontrolní akce.
  Nově vytvořený zdroj může být odkazovaný pomocí URI vráceného v entitě odpovědi, jehož specifické URI je v hlavičce v poli Lokace.
  Zdroj musí být na serveru vytvořen před odesláním tohoto stavového kódu. Pokud server tuto akci nemůže vykonat okamžitě, měl by namísto toho vrátit stavový kód 202 (ACCEPTED).
- **HTTP kód 202 (ACCEPTED)**
  Stavový kód 202 je typicky užívaný pro akce, jejichž zpracování trvá delší dobu. Indikuje, že příkaz byl přijat ke zpracování, ale že zpracování nebylo zatím ukončeno. Příkaz se může poté úspěšně dokončit, či během zpracování naopak nevykonat.
  Účel tohoto kódu je pro umožnění serveru akceptovat příkaz pro jiný proces bez nutnosti udržovat připojení agenta k serveru, než bude proces dokončen.
  Entita odeslána společně s touto odpovědí by měla zahrnovat aktuální stav příkazu a odkaz na monitor stavu či odhad, kdy by měl být příkaz vykonán.
- **HTTP kód 204 (NO CONTENT)**
  Kód 204 je obvykle zaslán jako odpověď na příkazy PUT, POST, či DELETE, když REST API neposílá zpět stavovou zprávu ani obsah v těle zprávy. V případě příkazu GET může být kód také zaslán a indikovat, že zdroj existuje, ale nemá žádný

stavový obsah v těle. Odpověď 204 nesmí zahrnovat zprávu v těle a je vždy terminována prvním prázdným řádkem po hlavičce.

- **301 (MOVED PERMANENTLY)**
  Stavový kód 301 indikuje, že zdrojový model REST API byl značně pozměněn a nová permanentní URI byla přiřazena ke zdroji, který klient požadoval. REST API by v tomto případě mělo specifikovat nové URI v hlavičce odpovědi v poli Lokace, a budoucí příkazy by měly být směrovány na dané URI.
  Tento kód je zřídka užíván v API, jelikož díky verzování API je možné ponechat jak nové, tak staré API.
- **302 (FOUND)**
  HTTP kód 302 je běžnou cestu k přesměrování URL. Odpověď s tímto kódem poskytne také URL v hlavičce v poli Lokace. Uživatelský agent je touto odpovědí vybízen k zaslání druhého, jinak identického, příkazu na nově specifikované URL v hlavičce.
- **303 (SEE OTHER)**
  Kód 303 indikuje, že zdroj ukončil zpracován, ale namísto zaslání potenciálně nechtěné odpovědi, zasílá klientovi URI zdroje s odpovědí. To může být jak dočasná stavová zpráva, tak jiný existující, více permanentní, zdroj.
  303 umožňuje REST API zaslat odkaz na zdroj, aniž by nutil klienta stahovat jeho stav. Namísto toho může klient zaslat příkaz GET na hodnotu z hlavičky Lokace.
- **304 (NOT MODIFIED)**
  Tento kód je podobný kódu 204 v tom, že tělo zprávy musí být prázdné. Rozdílem je ovšem, že u kódu 204 není zasláno z důvodu toho, že není nic k odeslání, ale u kódu 304 to je z důvodu toho, že se nic nezměnilo od verze, která je specifikována u příkazu v hlavičce If-Modified-Since či If-None-Match.
  V tomto případě není potřeba přeposílat zdroj, jelikož klient má předešlou verzi. Toto šetří kapacitu sítě a zpracování na straně serveru i klienta, jelikož se odesílají pouze data v hlavičce oproti celkové stránce.
- **307 (TEMPORARY REDIRECT)**
  Odpověď 307 indikuje, že REST API nezpracuji klientův příkaz. Namísto toho by klient měl příkaz zaslat znovu na URI specifikované v odpovědi v hlavičce Lokace. Ovšem budoucí příkazy by měly nadále používat originální URI.
  REST API může tento kód používat pro přidání dočasného URI na zdroj, který žádá klient. Například to může být při poslání příkazu na jiného hostitele.
  Dočasné URI by mělo být specifikováno v poli Lokace v odpovědi. Odpověď by měla obsahovat krátkou hypertextovou poznámku na nové URI.
- **400 (BAD REQUEST)**

400 je obecný chybový stav na straně klienta, když není žádný vhodnější z dané kategorie. Chyby mohou být ve špatné syntaxi, nevalidních parametrech příkazu atd. Klient by neměl zasílat příkaz znova bez modifikací.

- **401 (UNAUTHORIZED)**
  Chybová odpověď 404 indikuje, že se klient snaží komunikovat se zabezpečeným zdrojem bez správné autorizace. Klient mohl poskytnout buď špatné údaje či žádné. Odpověď musí obsahovat WWW-Authenticate hlavičku, která obsahuje požadavek zdroje.
  Klient může příkaz zaslat znovu s vhodnou autorizační hlavičkou. Pokud příkaz již zahrnoval autorizační hlavičku, tak odpověď oznamuje, že autorizace pro tyto údaje byla zamítnuta.
- **403 (FORBIDDEN)**
  Chyba 403 značí, že klientův příkaz je správný, ale REST API ho zamítlo, například z důvodu nedostatečných práv uživatele na zdroj. Autentizace v tomto případě nepomůže a příkaz by neměl být opakován.
- **404 (NOT FOUND)**
  Tento stavový kód indikuje, že REST API nemůže klientovo URI namapovat ke zdroji. Z odpovědi nelze poznat, zda je tento stav dočasný či permanentní. Pro informaci o permanentně nedostupném zdroji slouží kód 410 (GONE). Tento stavový kód je zpravidla užíván, když server nechce odhalit důvod, proč byl příkaz odmítnut.
- **405 (METHOD NOT ALLOWED)**
  Odpověď 405 indikuje chybu, kdy klient užívá HTTP metodu, která není zdrojem podporována. Například pokud je zdroj pouze ke čtení. Odpověď musí také obsahovat hlavičku Allow, která specifikuje HTTP metody, které zdroj podporuje.
- **406 (NOT ACCEPTABLE)**
  Chyba 406 indikuje, že API nemůže generovat žádné z požadovaných typů souborů, které klient požaduje v hlavičce příkazu Accept. Například, že klient požaduje XML soubor, ale API formátuje data pouze jako JSON.
- **412 (PRECONDITION FAILED)**
  Chybový stav 412 indikuje, že klient specifikoval prerekvizity v hlavičce příkazu, které se serveru nepodařilo splnit a namísto vykonání příkazu tedy zasílá tento stavový kód.
- **415 (UNSUPPORTED MEDIA TYPE)**
  415 je chybový stav, který oznamuje, že API nemůže zpracovat typ dat, který klient zaslal a indikoval v hlavičce příkazu Content-Type.
- **500 (INTERNAL SERVER ERROR)**

500 je generická chybová hláška. Většina webových aplikačních rámců zasílá tuto chybu, když zpracovává příkaz, který způsobí výjimku. Chyba 500 není nikdy na straně klienta, a proto se příkaz může vykonávat opakovaně. Tato chybová hláška generická zpráva, kdy nastala nečekaná chyba a není žádná vhodnější hláška v kategorii chyb.

- **501 (NOT IMPLEMENTED)**
  Server nepoznává metodu příkazu, či nemá možnost příkaz splnit. Zpravidla je to chybová hláška pro budoucí funkcionality.

Příloha B: Ukázkový dokument WADL

V této příloze je ukázkový WADL, a to konkrétně WADL společnosti Amazon. Tento dokument demonstruje některé z metod a parametrů, které jsou potřeba pro testování REST API. [49]

```
1 <application xmlns="http://wadl.dev.java.net/2009/02"
 2   xmlns:app="http://www.w3.org/2007/app"
 3   xmlns:atom="http://www.w3.org/2005/Atom">
 4
 5   <grammars>
 6     <include href="http://www.w3.org/2007/app/app.xsd"/>
 7   </grammars>
 8
 9   <resource_type id="entry_feed">
10     <method href="#getFeed"/>
11     <method href="#addEntryCollectionMember"/>
12   </resource>
13
14   <resource_type id="media_feed">
15     <method href="#getFeed"/>
16     <method href="#addImageCollectionMember"/>
17   </resource>
18
19   <resource_type id="entry">
20     ...
21   </resource_type>
22
23   <representation id="entry" mediaType="application/atom+xml"
24     element="atom:entry"/>
25
26   <representation id="feed" mediaType="application/atom+xml"
27     element="atom:feed">
28     <param name="first_link" style="plain"
29       path="/atom:feed/atom:link[@rel='first']">
30       <link resource_type="#entry_feed" rel="first"/>
31     </representation>
32     <param name="next_link" style="plain"
33       path="/atom:feed/atom:link[@rel='next']">
34       <link resource_type="#entry_feed" rel="next" rev="previous"/>
35     </param>
36     <param name="prev_link" style="plain"
37       path="/atom:feed/atom:link[@rel='previous']">
38       <link resource_type="#entry_feed" rel="previous" rev="next"/>
39     </param>
```

```
40      <param name="last_link" style="plain"
41        path="/atom:feed/atom:link[@rel='last']">
42        <link resource_type="#entry_feed" rel="last"/>
43      </param>
44    </representation>
45
46    <method name="GET" id="getFeed">
47      <response>
48        <representation href="#feed"/>
49      </response>
50    </method>
51
52    <method name="POST" id="addEntryCollectionMember">
53      <request>
54        <representation href="#entry"/>
55      </request>
56      <response status="201">
57        <param name="location" style="header" type="xsd:anyURI"
58          required="true">
59          <link resource_type="#entry" rel="self"/>
60        </param>
61        <representation href="#entry"/>
62      </response>
63    </method>
64
65    <method name="POST" id="addImageCollectionMember">
66      <request>
67        <representation mediaType="image/*"/>
68      </request>
69      <response status="201">
70        <param name="location" style="header" type="xsd:anyURI"
71          required="true">
72          <link resource_type="#entry" rel="self"/>
73        </param>
74        <representation href="#entry"/>
75      </response>
76    </method>
77
78  </application>
```

Příloha C: Ukázkový dokument WSDL

V této příloze je ukázkový WSDL. Jedná se o ukázkový soubor kalkulátoru z webových stránek Microsoftu [50]. Tento dokument demonstruje, jak vypadá verze dokumentu pro služby SOAP.

```
<wsdl:definitions
  xmlns:soap="https://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsu="https://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
  xmlns:soapenc="https://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tns="https://Example.org"
  xmlns:wsa="https://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsp="https://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsap="https://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
  xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  xmlns:msc="https://schemas.microsoft.com/ws/2005/12/wsdl/contract"
  xmlns:wsaw="https://www.w3.org/2006/05/addressing/wsdl"
  xmlns:soap12="https://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:wsa10="https://www.w3.org/2005/08/addressing"
  xmlns:wsx="https://schemas.xmlsoap.org/ws/2004/09/mex"
targetNamespace="https://Example.org"
  xmlns:wsdl="https://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xsd:schema targetNamespace="https://Example.org" elementFormDefault="qualified" >
  <xsd:element name="Add">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element minOccurs="0" name="a" type="xsd:int" />
        <xsd:element minOccurs="0" name="b" type="xsd:int" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="AddResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element minOccurs="0" name="result" type="xsd:int" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Subtract">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element minOccurs="0" name="a" type="xsd:int" />
        <xsd:element minOccurs="0" name="b" type="xsd:int" />
```

```
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="SubtractResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element minOccurs="0" name="result" type="xsd:int" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="ICalculator_Add_InputMessage">
    <wsdl:part name="parameters" element="tns:Add" />
  </wsdl:message>
  <wsdl:message name="ICalculator_Add_OutputMessage">
    <wsdl:part name="parameters" element="tns:AddResponse" />
  </wsdl:message>
  <wsdl:message name="ICalculator_Subtract_InputMessage">
    <wsdl:part name="parameters" element="tns:Subtract" />
  </wsdl:message>
  <wsdl:message name="ICalculator_Subtract_OutputMessage">
    <wsdl:part name="parameters" element="tns:SubtractResponse" />
  </wsdl:message>
  <wsdl:portType name="ICalculator">
    <wsdl:operation name="Add">
      <wsdl:input wsaw:Action="https://Example.org/ICalculator/Add"
message="tns:ICalculator_Add_InputMessage" />
      <wsdl:output wsaw:Action="https://Example.org/ICalculator/AddResponse"
message="tns:ICalculator_Add_OutputMessage" />
    </wsdl:operation>
    <wsdl:operation name="Subtract">
      <wsdl:input wsaw:Action="https://Example.org/ICalculator/Subtract"
message="tns:ICalculator_Subtract_InputMessage" />
      <wsdl:output wsaw:Action="https://Example.org/ICalculator/SubtractResponse"
message="tns:ICalculator_Subtract_OutputMessage" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="DefaultBinding_ICalculator" type="tns:ICalculator">
    <soap:binding transport="https://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="Add">
      <soap:operation soapAction="https://Example.org/ICalculator/Add" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
```

```
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Subtract">
      <soap:operation soapAction="https://Example.org/ICalculator/Subtract"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CalculatorService">
        <wsdl:port name="ICalculator" binding="tns:DefaultBinding_ICalculator">
            <soap:address location="https://Example.org/ICalculator" />
        </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```