Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Použití nástroje pro plánování a řízení testů v projektu TbUIS

Plzeň 2020 Tomáš Kotous

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš KOTOUS**
Osobní číslo:	**A17B0254P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Použití nástroje pro plánování a řízení testů v projektu TbUIS**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s obecnými požadavky na aplikace pro plánování a řízení testů. Dále se seznamte s projektem TbUIS.
2. Vyberte minimálně pět existujících aplikací pro plánování a řízení testů. Navrhněte multikriteriální hodnocení vybraných aplikací, přičemž klíčová kriteria budou vycházet z použitelnosti této aplikace v předmětu KIV/OKS. Na základě výsledků hodnocení vyberte jednu z aplikací, jejíž používání popište.
3. Prostudujte API vybrané aplikace. Navrhněte a implementujte způsob odesílání výsledků automatických testů přes API zvolené aplikace.
4. Ve zvolené aplikaci připravte sadu požadavků a na nich navazující sadu testovacích případů systému UIS z projektu TbUIS. Rozšiřte stávající sadu funkcionálních testů o testy, které chybějí pro 100% pokrytí sady požadavků.
5. Ověřte kvalitu vytvořených testů, navržené metody popište a vyhodnoťte.

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

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Pavel Herout, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **7. října 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

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

V Plzni dne 15. října 2019

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2020

Tomáš Kotous

Abstract

The aim of this bachelor's thesis was find the most suitable software from the category of test management systems, which can be used both in teaching and in research of new testing methods. The Squash TM tool was selected and at the same time a set of support programs was created that communicate with this tool via its API (Application programming interface). This set will significantly increase the usability of the Squash TM tool in teaching, where students can immediately start with pre-prepared Requirements (RQM) and Test Cases (TC). Part of the practical section was to verify the usability of such created applications.

Abstrakt

Cílem této bakalářské práce bylo nalezení nástroje z kategorie test management systémů, který je možné použít jak ve výuce tak i ve výzkumu nových testovacích metod. Byl vybrán nástroj Squash TM a zároveň byla vytvořena sada podpůrných programů, které s tímto nástrojem přes jeho API (Application programming interface) komunikují. Tato sada významně zvýší použitelnost nástroje Squash TM ve výuce, kdy mohou studenti ihned začít experimentovat s předpřipravenými Requirements (RQM) a Test Cases (TC). Součástí praktické části bylo ověření použitelnosti takto vytvořených aplikací.

Obsah

1 Úvod

Testování software, má-li být úspěšné a udržovatelné, se v současné době neobejde bez podpůrných softwarových nástrojů používaných pro organizaci testování. Těchto nástrojů je značné množství kategorií a velmi rozdílných kvalit a schopností.

Za naprostý základ, který by měly používat všechny softwarové projekty, lze považovat dvě kategorie nástrojů. Prvním z nich je nástroj pro plánování a řízení testů – test management systém. Tento nástroj umožní „udržet systém a pořádek“ v rozsáhlých sadách testů, přičemž termín „rozsáhlý“ představuje stovky až deseti tisíce testovacích případů. Nástroj je používaný víceméně pouze testery.

Druhým typem nástroje je software pro řízení chybových reportů, který má hned několik jiných používaných názvů, např. správa incidentů, bug report system, issue tracking systém atd. Umožňuje opět „udržet systém a pořádek“ v sadách defektů, které jsou nalezeny testováním a reportovány testery vývojářům. Tento nástroj je interface mezi testerem a vývojářem.

Ve výuce předmětu KIV/OKS se jako praktická ukázka nástroje z kategorie Test management až do akademického roku 2018/19 používal systém TestLink. Tento systém sice vyhovuje požadavkům předmětu, ale v porovnání s ostatními existujícími nástroji, např. Mantis Bug Tracker (z kategorie řízení chybových reportů), je zřejmé, že je TestLink již koncepčně zastaralý. Z tohoto důvodu je vhodné kvalifikovaně vyhledat jiný nástroj, který by mohl TestLink nahradit jak ve výuce, tak i ve výzkumu nových testovacích metod. Požadavky na nový nástroj jsou díky dlouholetým zkušenostem s klady a zápory TestLinku poměrně přesně známy.

Vyhledání vhodného nástroje představuje první část bakalářské práce. Ve druhé části práce bude nutné připravit předem neurčené množství podpůrných programů, jejichž počet a možnosti se budou odvíjet od možností API nástroje vybraného v první části práce.

Programy by s využitím API zvoleného nástroje měly umožnit jeho snadnou použitelnost jak ve výuce předmětu KIV/OKS, tak i ve zmíněném výzkumu. Úsilí bude zaměřeno zejména do oblastí importu a exportu sad požadavků a testovacích případů, importů výsledků proběhlých testů, zajištění kontrolovatelného pokrytí případů užití požadavky a potažmo i testovacími případy. Pozornost by měla být věnována i jejich zpětné trasovatelnosti.

Součástí práce bude i praktické ověření použitelnosti vybraného nástroje na konkrétní testované aplikaci.

2 Aplikace pro plánování a řízení testů

Abychom pochopili, proč existují aplikace pro plánování a řízení testů, musíme si nejdříve vysvětlit, proč je testování tolik důležité a jaký je jeho smysl. Testování můžeme chápat jako proces odhalování defektů před okamžikem, kdy mohou uškodit. Uvážíme-li, že je téměř nemožné vytvořit produkt bez defektů, může nám jejich odhalování poskytovat informaci o kvalitě testovaného softwaru. Chceme-li tedy vydat hodnotný softwarový produkt, je nezbytné jej nejdříve otestovat.

Aplikace, jež jakýmkoliv způsobem pomáhají tohoto cíle dosáhnout, jsou v dnešní době již nedílnou součástí testování. Jejich hlavním úkolem je zkvalitnění a zrychlení procesu testování. Právě mezi ně se řadí i aplikace pro plánování a řízení testů.

Tyto nástroje se především zaměřují na zaznamenávání testovacích dokumentů (mezi ně patří například testovací plán či testovací případ), monitorování průběhu testování a vymezení jakým způsobem se testování bude ubírat.

Testování softwaru není jednoznačně metodicky kodifikováno, existuje řada i různorodých praxí osvědčených postupů, a proto existují aplikace pro plánování a řízení testů s odlišnými vlastnostmi a nabízí testerům různorodá prostředí. Při výběru těchto aplikací je vhodné se zamyslet, jakým způsobem budeme testovat, aby se plně využil jejich potenciál.

2.1 Možnosti aplikací

V této sekci se nachází přehled nejčastějších funkcí, které aplikace poskytují. Ty nemusí obsahovat veškeré níže popsané vlastnosti, jejich kvantita totiž neimplikuje kvalitu softwaru jako celku. Existují nástroje, jež obsahují velké množství funkcionalit, ale stávají se nepřehlednými a v praktickém světě nevyužitelnými. Proto ve většině případů obsahují v praxi osvědčené nástroje různé kombinace těchto dovedností. Je vhodné věnovat čas již při vybírání a pečlivě zvážit, jaké aspekty by nám nástroj měl poskytovat, a tím předejít zbytečným migracím na jiný nástroj.

Mezi nejdůležitější se řadí tyto vlastnosti [6]:

- **Správa testovacích dokumentů** – vytváření, upravování a odstraňování testovacího plánu, testovacích případů, požadavků a ostatních důležitých dokumentů,
- **správa uživatelů** – přidávání a odebírání členů týmu,
- **uživatelské role** – uživatelům je umožněno přidělit role, které specifikují, co daná osoba může vykonávat (například Manažer – spravuje testovací dokumenty, Tester – spouští testy),
- **přiřazování testů** – uživatel, ideálně v definované roli, může přiřadit test jinému členu,
- **manuální spuštění testů** – aplikace podporuje ručně spustit jednotlivé testy a zaznamenat jejich výsledky,
- **reportování proběhlého testu** – zobrazení podstatných informací o proběhlém testu, například status, datum testování,
- **verzování TC (Test Case) a RQM (Requirement)** – testovací případy a požadavky se mohou vyskytovat v určitých verzích,
- **zobrazení proběhlých testů** – přehledné zobrazení záznamů již dokončených testů.

2.2 Očekávání

Podle [8] tester od těchto nástrojů požaduje především možnost propojení testovacích případů s požadavky, které pomáhá k pochopení spojitosti a tím k zajištění lepšího pokrytí testované aplikace testy. V ideálním případě je vždy jeden testovací případ navázán na jeden požadavek, ovšem v reálném světě je daleko častější případ více testovacích případů navázaných na stejný požadavek. V takovém případě je přehledné zobrazení propojení velkým pozitivem.

Druhým nejčastějším očekáváním je použitelnost aplikace. To, do jaké míry je aplikace schopná předvídat, co tester v danou chvíli potřebuje, je neocenitelná vlastnost pro testování. Jedná se například o možnost ihned propojit testovací případ při jeho vytvoření.

Další motivací pro používání těchto nástrojů je zrychlení testovacího procesu, to vede k očekávanému snížení celkových nákladů na testování. Požadovaného zrychlení lze dosáhnout vícero způsoby. Nejčastějším je automatizované testování. Při jeho použití odpadá nutnost testy manuálně spouštět a navíc nedochází k chybám, způsobené při opakovaném spuštění stejných testů, tzv. „human error“ (při opakování totožných úkonů, může čas od času člověk udělat chybu).

Dále je vhodné, aby nástroj poskytoval uspořádané zobrazení testovacích dokumentů. Tester potřebuje udržovat pořádek i ve stovkách či tisících TC a RQM a dobře se v nich orientovat.

Možnost vytvářet přehledné reporty z testů patří mezi další očekávání. Ty jsou důležitým aspektem pro demonstraci průběhu testování a podávání informací nadřízeným.

2.3 Spolupráce s ostatními softwary

Při vývoji kvalitního softwaru je vhodné předpovídat, jaké další tematicky obdobné softwary bude uživatel používat. Nativní podpora s takovým nástojem usnadní uživateli práci. Nejčastějším případem je propojení aplikace pro řízení testů s programem pro správu chyb, přičemž vznikne nová funkcionalita, například možnost reportování defektu při manuálním spuštění testu. V rámci testování se jedná zejména o tyto oblasti nástrojů [9]:

- **Test management tools** – nástroje pro plánování a řízení testů,
- **Bug tracking tools** – nástroje pro správu defektů,
- **Test automatization tools** – nástroje pro spravování automatizovaných testů (testy, které se dají opakovaně spouštět),
- **Supportive test tools** – ostatní nástroje, které testerovi jakýmkoliv způsobem pomáhají při testování, například databázové nástroje.

3 Projekt TbUIS

Tato kapitola pojednává o projektu TbUIS, který zde bude popsán ve stručnosti, vzhledem k tomu, že již byl součástí diplomové práce [12] a popsán v bakalářských pracích [13, 14] a diplomové práci [15]. Uvedené informace jsou čerpány ze stránek [11].

3.1 O projektu

Projekt vznikl jako reakce na absenci netriviální realistické aplikace, na které by bylo možné experimentovat s nově vyvíjenými testovacími metodami. Většina ověřování těchto metod byla aplikována na triviálních aplikacích, kde nedochází k jejich účinnému vyzkoušení, a to zejména z praktického hlediska. Motivací pro vytvoření tohoto projektu bylo poskytnutí testované aplikace (SUT – System Under Test) umožňující realistické vyhodnocení nových testovacích metod. Projekt je vnímán jako celek složený z několika podstatných částí. Jedná se o aplikace: UIS, ErrorSeeder a Defect-clones. Tyto části jsou níže popsány.

3.2 UIS

UIS (University Information System) je simulace reálné webové aplikace. Do ní je možné se přihlásit v roli studenta, či učitele. Rolím byly přiděleny různé smysluplné aktivity, které mohou uživatelé provozovat, za účelem nastínění reálného používání. Student si může zapisovat a odebírat předměty, popřípadě si zapisovat zkouškové termíny, zatímco učiteli je umožněno začít vyučovat, či naopak končit s výukou konkrétních předmětů a hodnotit studenty.

Důležitou vlastností této aplikace je předpokládaná bezchybnost. Ta byla dosažena extrémně rozsáhlým testováním. Zejména se jednalo o JUnit testy (jednotkové testy) a funkcionální testy (viz sekce 3.5). Díky absenci defektů slouží otestovaná aplikace jako vzor, ze kterého jsou dále tvořeny kopie, do kterých jsou uměle zanášeny defekty.

3.3 ErrorSeeder

ErrorSeeder (Správce poruchových entit) je nedílnou součástí TbUIS. Jedná se o desktopovou aplikaci, která je schopna do korektně fungujícího software zanést defekty a tím vytvářet z původní bezchybné aplikace tzv. „poruchové klony“.

Výsledným produktem po použití této aplikace je standardní formát webové aplikace, tj. soubor s příponou `.war`, jenž se může ihned nasadit na vlastní server. Uživateli je v aplikaci ErrorSeeder umožněno shlukovat logicky navazující verze do šablon, které může opětovně využívat pro export.

Hlavním účelem aplikace ErrorSeeder je tedy vytváření verzí / klonů aplikace UIS se zanesenými defekty, na kterých se potom mohou evaluovat nově vyvíjené testovací metody a postupy.

3.4 Defect-clones

Defect-clones (poruchové verze aplikace) jsou jednotlivé verze UIS, do kterých jsou zaneseny defekty pomocí aplikace ErrorSeeder.

Klony jsou vytvářeny pro experimenty nově vytvořených testovacích metod. Musí obsahovat jedinečný identifikátor, pro rozlišení od ostatních entit. Dále obsahují jméno, popis zanesené chyby a scénář, který definuje akci, při které defekt nastane.

3.5 Funkcionální testy

Pro představu, jaký mají funkcionální testy význam, je potřeba nejdříve vysvětlit pojem funkcionální testování. To ověřuje, do jaké míry je aplikace funkční. Pracuje na principu černé skříňky (tester nezná vnitřní detaily implementace, jsou mu k dispozici pouze vstupy a výstupy). Je nezbytné mít k dispozici případy užití, či jiné zdroje informací ohledně vstupů a výstupů. Poté je možné vytvářet jednotlivé funkcionální testy. Ty mohou testovat jednu aktivitu, celý scénář, očekávanou funkčnost aplikace nebo reakci na nesprávné používání aplikace (tato rozdělení jsou dále popsány níže). V případě absence již zmiňovaných informací se jedná o tzv. průzkumné testování (tester namátkou zkouší chování aplikace, a tím ověřuje její předpokládanou funckionalitu).

Právě tyto informace mají úzkou spojitost s nástroji pro plánování a řízení testů. Přehledné zaznamenávání požadavků a testovacích případů značně pomáhá při systematickém vytváření funkcionálních testů.

V projektu TbUIS byly tyto testy v první verzi vytvořeny bez jasného počátečního konceptu, což způsobilo jejich průběžný vývoj a změny. Následné exekuce testů jsou logovány do souboru. Je zcela evidentní, že chybí software, který by pomohl zpřehlednit jednotlivé testy a korektně a přehledně agregoval jejich výsledky. Kdyby byl použit některý z dostupných nástrojů typu Test management, nemuselo by k podobnému nepříznivému vývoji testů vůbec dojít. Vyhledání a zakomponování takového nástroje do projektu TbUIS je předmětem této bakalářské práce.

V projektu TbUIS je celkem 983 funkcionálních testů. Ty však mohou obsahovat jeden, či více assert testů. Dohromady je tedy v projektu 5105 testů. Testy se rozdělují do tří hlavních kategorií: Passive, Negative a Active. Veškeré testy včetně rozdělení jsou zaznamenány v tabulce 3.1.

	počet funkcionálních testů	počet assert testů
Passive	890	2702
Negative	29	52
Active	64	2351
Celkem	983	5105

Tabulka 3.1: Tabulka s funkcionálními testy projektu TbUIS

- **Passive** – pozitivní testy, ověřují správné nastavení aplikace, tj. existenci všech očekávaných webových elementů a také spolupráci s SŘBD (Systém řízení báze dat).
- **Negative** – negativní testy, slouží ke zjištění, zdali aplikace umí reagovat na nestardadní chování uživatele. Například přihlášení na zkouškový termín nezapsaného předmětu.
- **Active** – aktivní testy, zajišťují otestování jednorázové aktivity uživatele. Například zrušení zkouškového termínu se musí projevit jak u učitele, tak i u studenta.

4 Vybrané testovací nástroje

V této kapitole budou podrobněji popsány nástroje, které byly vybrány rozsáhlou rešerší webu [1, 2, 3, 4]. Provedená rešerše poskytla seznam několika slibných kandidátů na hledaný test management systém z několika desítek existujících systémů.

Vzhledem k faktu, že otestování aplikace TbUIS má podobné požadavky jako předmět KIV/OKS, byly vybrány nástroje s obdobnými vlastnostmi aktuálně používaného softwaru TestLink. S tímto nástrojem bylo porovnáváno celkem 21 softwarů, získaných již zmiňovanou rešerší webu. Z této množiny bylo vybráno 5 nástrojů, které svými deklarovanými vlastnostmi nejlépe vyhovovaly požadavkům. Tyto nástroje byly dále poměrně detailně prozkoumány, tj. byly nainstalovány a jejich deklarované vlastnosti versus stanovené požadavky byly skutečně ověřovány. Výsledky byly zpracovány pomocí tzv. multikritériálního hodnocení, které je popsáno v kapitole 5.

V tabulce 4.1 jsou zaznamenány vybrané nástroje s vlastnostmi, které byly stanoveny po dohodě s vedoucím bakalářské práce. V následujícím textu budou tyto vlastnosti přiblíženy.

Nezbytnou podmínkou pro použití nástroje i ve výuce KIV/OKS je bezplatnost, při výběru byl tedy kladen důraz, aby byl program alespoň v základní funkčnosti k dispozici zdarma, nebo ve formě „free trial“ (na určitý časový úsek může zákazník produkt používat), a to alespoň na 14 dní. Další potřebná vlastnost je hierarchická struktura požadavků a testovacích případů, která zvyšuje přehlednost. Při rešerši bylo zjištěno, že tato vlastnost je v praktickém používání kupodivu dosud málo používána, bylo tudíž obtížné vyhledat aplikace s touto funkcionalitou.

Další důležité vlastnosti jsou: RQM provázané na TC, export a import testovacích dokumentů, správa testovacího plánu a manuální spouštění testů. Bez těchto aspektů by aplikaci nebylo možné používat v rámci předmětu KIV/OKS. Mezi další, již méně podstatné, se řadí možnost vytvářet buildy a milníky, lokální instalace, webová aplikace, snadnost použití, přístupné API. Jako nejméně podstatné vlastnosti patří jednoduchá instalace, provázání s programem pro správu defektů (ideálně Mantis BT), verzování RQM a TC, definování uživatelských rolí, integrovaná nápověda a ochota tvůrců komunikovat a řešit problémy.

Poznámka: Zkratka TM v tabulce znamená Test Management, nikoliv TradeMark.

Tabulka 4.1: Přehled vlastností vybraných nástrojů

Vlastnosti \ Nástroje	Test Collab	Squash TM	Klaros TM	Kiwi TCMS	Kualitee
Zdarma	Ano	Ano	Ano	Ano	Ano
Hierarchická struktura RQM/TC	Ne	Ano	Ne	Ne	Ne
RQM provázané na TC	Ano	Ano	Ano	Ne	Ano
Export/import RQM a TC	Ne	Ano	Ano	Ano	Ne
Export reportů	Ano	Ano	Ano	Ano	Ano
Testovací plán	Ano	Ano	Ano	Ano	Ano
Ruční provádění testů	Ano	Ano	Ano	Ano	Ano
Buildy a milníky	Ne	Ne	Ne	Ne	Ne
Lokální instalace	Ne	Ano	Ano	Ano	Ne
Webová aplikace	Ano	Ano	Ano	Ano	Ano
RQM a TC s možností příloh	Ano	Ano	Ano	Ano	Ano
Snadné vytváření kroků TC	Ano	Ano	Ano	Ano	Ano
Snadnost použití	Ano	Ano	Ano	Ano	Ne
API aplikace	Ano	Ano	Ano	Ano	Ano
Jednoduchá instalace	Ano	Ano	Ano	Ne	Ano
Provázání s Mantis BT	Ano	Ano	Ne	Ano	Ne
Verzování RQM a TC	Ne	Ne	Ano	Ano	Ano
Uživatelé mají role	Ano	Ano	Ano	Ano	Ano
Integrovaná nápověda	Ano	Ne	Ano	Ne	Ne
Ochota tvůrců komunikovat	Ano	Ne	Ne	Ne	Ano

Dále budou jednotlivé systémy podrobněji popsány.

4.1 Test Collab

Test Collab patří mezi moderní testovací nástroje. Oproti svým konkurentům je rychlejší a nabízí své služby na svých serverech, takže se uživatel nemusí starat o lokální instalaci, ušetří místo na disku a má zaručeno, že využívá vždy aktuální verzi nástroje. V případě, že uživatel potřebuje být nezávislý, je mu poskytnuta možnost spravovat si veškeré nabízené služby na vlastním serveru.

Mezi základní funkcionalitu, kterou nástroj nabízí, patří správa testovacích případů a požadavků, tvorba milníků, manuální spouštění testů, export reportů z proběhlých testů, přidávání uživatelů a přiřazování rolí.

Je zaměřený především pro zákazníky, kteří při testování upřednostňují zejména čas. Při vývoji byl kladen důraz na intuitivní ovládání, takže seznámení se s nástrojem nezabírá tolik času oproti ostatním nástrojům. Pro ty, kteří by měli i nadále problémy, je poskytnuta integrovaná nápověda. Dále nabízí pokročilé možnosti, jako například `Reusable steps` (tester nemusí opakovaně zadávat kroky u testovacích případů, které jsou společné), či `Time tracker` (přehledné zobrazení stráveného času při testování).

Další motivací pro používání tohoto softwaru je uživatelské rozhraní. To poskytuje přehledné zobrazení komponent a moderní design. Uživatel není přehlcen všemi možnostmi nástroje, jak to většinou bývá u konkurenčních softwarů. Základní funkcionalita je poskytnuta ihned po založení nového projektu a doplňující možnosti, které uživatelé tak často nevyužívají, jsou pro jednotlivé projekty zpřístupněny po aktivaci v nastavení.

Test Collab také podporuje do určité míry přizpůsobení. Například pro správu požadavků si uživatel může vybrat, zdali chce používat správce požadavků zabudovaného, nebo poskytnutého třetí stranou. Bohužel toto přizpůsobení je pravděpodobně také důvodem, proč práce s požadavky zaostává. Chybí zde hierarchické uspořádání jednotlivých požadavků a možnost jejich exportu.

Nespornou výhodou je propojení s nástroji pro sledování defektů. Pro úspěšné propojení je potřeba vybrat požadovaný nástroj z výběrového menu předpřipravených možností, URL (Uniform Resource Locator) adresa, na které nástroj běží a jméno uživatele, popřípadě API klíč.

4.2 Squash TM

Squash TM byl vytvořen francouzskou firmou Henix a je neustále vyvíjen, což má samozřejmě příznivý vliv na jeho kvalitu.

Nástroj nabízí poměrně profesionální možnosti, které mohou být dále rozšířeny pomocí pluginů. Jedná se například o plugin, umožňující výměnu JSON (JavaScript Object Notation) souborů s aplikací přes HTTP (Hypertext Transfer Protocol) metody. Díky tomu lze propojit aplikaci s automatizovanými testy a zaznamenávat jejich průběh. Toto propojení bude dále součástí bakalářské práce. Další pluginy umožňují propojení se známými bugtrackery, nebo spojení s verzovacím systémem Git.

Výhodou oproti ostatním testovacím softwarům je hierarchické zobrazení testovacích případů a požadavků s funkcí `Drag and drop`, která umožňuje elegantní manipulaci s dokumenty napříč strukturou.

Nástroj je vhodný i pro začátečníky, takže vyhovuje nárokům předmětu KIV/OKS. Instalace je velmi komfortní, stačí stáhnout archiv obsahující nástroj, spustit jeden skriptovací soubor a program si zbytek obstará sám. Navíc se defaultně používá databáze H2, která slouží pro seznámení se s aplikací, což v rámci předmětu KIV/OKS je zcela dostačující. Odpadá povinnost instalace další databáze a nedochází ke kolizím s ostatními databázemi. Pro reálné využívání je doporučováno změnit databázi. Pro jednoduší přechod Squash nabízí Docker kontejnery pro databáze MariaDB a PostgreSQL.

Ani zde nechybí propojení s ostatními testovacími nástroji. Squash nativně podporuje bugtracker MantisBT. Pro propojení je potřeba jen zadat URL adresu, na které MantisBT běží. Po prvním pokusu využití bugtrackeru je uživatel dotázán na přihlašující údaje k účtu na server MantisBT. Poté již uživatel může nahlašovat defekty, které se uloží do stejnojmenného projektu vytvořeného v bugtrackeru.

4.3 Klaros testmanagement

Klaros testmanagement je produktem firmy **verit Informationssysteme GmbH**. Software Klaros vydala ve dvou verzí: `Community edition` a `Enterprise edition`. První zmiňovaná nabízí omezené možnosti placené verze `Enterprise edition`, ke které lze ovšem získat zkušební verzi na dobu 30 dní. Obě verze se musí lokálně nainstalovat a jejich spuštění na serveru zajišťuje software Apache Tomcat. Zásadní odlišností edicí je absence požadavků, testovacího plánu a iterací testů.

Systém nabízí možnost přidávat členy pro jednotlivé projekty, kterým lze určit uživatelské role. K dispozici jsou `Administrátor`, `Manažer`, `Tester` a `Host`. Každá role má přesně vymezeno, jaké akce může provádět. Mezi základní operace patří vytvoření projektu, správa testovacích případů a požadavků, manuální spuštění testů a zobrazení reportů z již proběhlých testů.

Oproti ostatním testovacím nástrojům Klaros podporuje více stylů softwarových vývojů. Konkrétněji umožňuje používání agilního testování (například Scrum), V-model, nebo Waterfall model. To se však na druhou stranu negativně projevuje na intuitivním ovládáním aplikace.

Od svých konkurentů zaostává designem a orientací v aplikaci. Některé uživatele může také odradit nepřehledné zobrazení testovacích dokumentů.

Mezi další funkcionalitu patří verzování testovacích případů a požadavků, import/export testovacích dokumentů do/z XML (Extensible Markup Language) či Excelovských souborů a možnost přidání dodatečných informací testovacím případům (například snímky z obrazovky).

4.4 Kiwi TCMS

Kiwi TCMS je open source testovací nástroj, který byl založen firmou Red Hat, Inc. Pro lokální instalaci je nutné mít nainstalovaný software Docker. Příkazem `docker pull kiwitcms/kiwi` se stáhnou dva kontejnery pro spuštění nástroje. Jeden slouží pro spuštění serveru a druhý obstarává uložení dat do databáze. Kiwi TCMS používá databázi MariaDB.

Nástroj disponuje minimalistickým designem, jenž obsahuje veškerou funkcionalitu pohromadě. Uživatel se nemusí „proklikat“ k požadované funkcionalitě a ihned ví, co může od aplikace čekat. Nevýhodou je však menší nabídka funkcionalit. Chybí zde například možnost spravovat požadavky a propojovat je s testovacími případy.

Na druhou stranu obstojně konkuruje placeným testovacím softwarům. Umožňuje vytvářet testovací plán, spravovat testovací případy, přidávat uživatele, manuálně spouštět testy, reportovat nalezené defekty, verzovat testovací dokumenty, vytvářet buildy a exportovat reporty z proběhlých testů.

Kromě těchto základních funkcionalit nabízí ještě možnost vytvoření produktu, jenž může být propojen se všemi testovacími dokumenty, které nástroj poskytuje.

Jednou z nevýhod tohoto nástroje je složité exportování a importování testovacích plánů a případů. Nástroj je podporuje jen vytvořením vlastního skriptu, který je schopen přes API aplikace tyto akce provést. Dalším nedostatkem je absence propojení s dalšími testovacími nástroji. Nástroj sice umožňuje nativně spravovat defekty, ale tato funkcionalita je s porovnáním s moderními bugtrackery značně zaostalá.

4.5 Kualitee

Kualitee je poměrně komplexní testovací nástroj. Poskytuje řadu možností pro testování a počítá se, že běžný tester všechny nevyužije. Řadí se mezi cloud-based (nástroj není potřeba lokálně instalovat) testovací nástroje. Byl vytvořen společností Kualitatem Inc. Ten jej distribuuje ve čtyřech variantách. První z nich je `Community`, která je jako jediná zdarma. To se projevuje na značném omezení funkcionalit. Jedná se zejména o snížení počtu uživatelů, projektů, testovacích případů a defektů. Další varianty jsou `Silver` a `Gold`. Ty jsou již placené a jsou zaměřené především pro menší firmy, neboť počet uživatelů je stále omezen. Na druhou stranu ostatní funkcionality jsou bez omezení a přibyla možnost zákaznické podpory. Poslední verzí je `Enterprise`, která je po větší firmy, neboť poskytuje neomezený počet uživatelů. Její cena je stanovena až po domluvě.

Nástroj vyniká především poskytováním mobilní aplikace pro flexibilní používání nástroje. To je ovšem dostupné pouze v placených verzí.

Další výhodou je příjemné uživatelské rozhraní, možnost posílání notifikací na emailovou adresu, manuální i automatické spouštění testů, správa testovacích dokumentů, export reportů z proběhlých testů a správa uživatelů.

Nástroj je určen spíše pro pokročilé testery. Je zde příliš velká nabídka možností, která může začínajícího testera zmást.

5 Multikriteriální hodnocení

Při snaze vybrat nejvhodnější nástroj je nutné použít komplexní analýzu, která poskytuje informaci o kvalitě produktu z různých hledisek. Toho lze docílit multikriteriální analýzou (nebo též multikriteriální hodnocení). Podle [7] se rozděluje do čtyř na sebe navazujících částí. V následujících sekcích bude popsáno její aplikování při návrhu multikriteriálního hodnocení vybraných aplikací.

5.1 Definování hodnotících kritérií

Prvním a zároveň nejdůležitějším krokem je zaznamenání veškerých aspektů (tzn. všeho co nás zajímá o pozorované entitě). Tomu bychom měli věnovat nejvíce času, neboť to, jaká kritéria začleníme do hodnocení, zásadně ovlivní jeho výsledek. Dobrým zvykem je zanášet tyto údaje do tabulky pro zlepšení přehlednosti. Pro testovací nástroj byly po dohodě s vedoucím bakalářské práce vybrána kritéria, jež jsou zaznamenána v tabulce 4.1.

5.2 Kvantifikace kritérií

Po sepsání jednotlivých kritérií můžeme začít přiřazovat jejich číselné hodnoty v námi určeném rozsahu. Tyto hodnoty reprezentují, do jaké míry jsme spokojeni s daným kritériem (např. z rozsahu 1 – 10 zvolíme 10 pro kritérium, které zcela naplnilo naše očekávání). V našem případě byl zvolen rozsah 1 až 3, kde 1 reprezentuje nesplněné očekávání, 2 představuje kritérium, které vyhovuje částečně a 3 zastupuje zcela splněné očekávání. Z navržené tabulky multikriteriálního hodnocení na obr. 5.1 lze vidět, že dva nástroje obdržely stejný počet bodů a poskytují tudíž podobnou funkcionalitu. Neznamená to však, že se pro výuku v předmětu KIV/OKS hodí oba nástroje, rozhodujícím faktorem je až následné přidělení vah.

Poznámka: Znamená to, že kvantifikace kritérií je v podstatě velmi objektivní. Pokud by ji provedl jiný hodnotitel, dostal by pravděpodobně velmi podobné (možná i stejné) výsledky.

	Test Collab	Squash TM	Klaros TM	Kiwi TCMS	Kualitee
Zdarma	3	3	3	3	3
Hierarchická struktura RQM/TC	2	3	1	1	1
RQM provázané na TC	3	3	2	1	2
Exp/imp RQM a TC	2	2	3	2	2
Export reportů	3	3	3	2	3
Testovací plán	3	3	3	3	3
Ruční provádění testů	3	3	3	3	3
Buildy a milníky	2	2	1	2	2
Lokální instalace	1	3	3	3	1
Webová aplikace	2	3	2	3	3
RQM a TC s obrázky	3	3	3	1	3
Snadné vytváření kroků TC	3	3	3	3	3
Snadnost použití	3	3	3	3	1
API aplikace	3	3	3	3	3
Jednoduchá instalace	3	3	3	1	3
Provázání s Mantis BT	3	3	3	1	1
Verzování RQM a TC	2	2	3	3	1
Uživatelé mají role	3	3	3	3	3
Integrovaná nápověda	3	1	3	1	1
Ochota tvůrců komunikovat	3	1	1	1	3
Součet	**53**	**53**	**52**	**43**	**45**

Obrázek 5.1: Tabulka s ohodnocenými atributy nástrojů

5.3 Přidělení vah

Třetí částí je již zmíněné přidělení vah. Kritérium, které je pro nás nejdůležitější, dostane největší váhu, zatímco kritérium, které je pro nás postradatelné, dostane váhu nejmenší. Mezi klíčová kritéria byla vybrána hierarchická struktura RQM/TC, bezplatnost a existence testovacího plánu, a to zejména pro použitelnost nástroje v předmětu KIV/OKS. Tato kritéria obdržela váhu 10. Na druhou stranu, méně podstatné vlastnosti, jako například ochota tvůrců komunikovat, získaly nejmenší váhu 1. Přiřazování vah je čistě subjektivní. Pokud by tedy někdo potřeboval vybrat nástroj pro praktické používání, nemusí již provádět důkladnou rešerši a pokusně si instalovat zmiňované nástroje, ale stačí zaměnit váhy dle svých osobních preferencí. Tabulka s atributy a jejich přidělenými váhami je znázorněna na obr. 5.2

	Přidělená váha	Celková váha	Váha v %
Zdarma	10	0,08	8
Hierarchická struktura RQM/TC	10	0,08	8
RQM provázané na TC	9	0,08	8
Exp/imp RQM a TC	8	0,07	7
Export reportů	8	0,07	7
Testovací plán	10	0,08	8
Ruční provádění testů	9	0,08	8
Buildy a milníky	7	0,06	6
Lokální instalace	3	0,03	3
Webová aplikace	7	0,06	6
RQM a TC s obrázky	6	0,05	5
Snadné vytváření kroků TC	5	0,04	4
Snadnost použití	5	0,04	4
API aplikace	5	0,04	4
Jednoduchá instalace	2	0,02	2
Provázání s Mantis BT	5	0,04	4
Verzování RQM a TC	4	0,03	3
Uživatelé mají role	3	0,03	3
Integrovaná nápověda	1	0,01	1
Ochota tvůrců komunikovat	1	0,01	1
Součet	**118**	**1**	**100**

Obrázek 5.2: Tabulka s atributy a jejich váhami

5.4 Evaluace výsledků

Závěrečnou činností je vyhodnocení výsledků. Pro každou entitu vypočteme její výsledné skóre jako součet veškerých ohodnocených kritérií, vynásobených přidělenou váhou. Výsledek tohoto úsilí reprezentuje nejvhodnější položku. V našem případě vyšel testovací nástroj Squash TM viz obr. 5.3, který je popsán v kapitole 6.

	Test Collab	Squash TM	Klaros TM	Kiwi TCMS	Kualitee
Zdarma	0,24	0,24	0,24	0,24	0,24
Hierarchická struktura RQM/TC	0,16	0,24	0,08	0,08	0,08
RQM provázané na TC	0,24	0,24	0,16	0,08	0,16
Exp/imp RQM a TC	0,14	0,14	0,21	0,14	0,14
Export reportů	0,21	0,21	0,21	0,14	0,21
Testovací plán	0,24	0,24	0,24	0,24	0,24
Ruční provádění testů	0,24	0,24	0,24	0,24	0,24
Buildy a milníky	0,12	0,12	0,06	0,12	0,12
Lokální instalace	0,03	0,09	0,09	0,09	0,03
Webová aplikace	0,12	0,18	0,12	0,18	0,18
RQM a TC s obrázky	0,15	0,15	0,15	0,05	0,15
Snadné vytváření kroků TC	0,12	0,12	0,12	0,12	0,12
Snadnost použití	0,12	0,12	0,12	0,12	0,04
API aplikace	0,12	0,12	0,12	0,12	0,12
Jednoduchá instalace	0,06	0,06	0,06	0,02	0,06
Provázání s Mantis BT	0,12	0,12	0,12	0,04	0,04
Verzování RQM a TC	0,06	0,06	0,09	0,09	0,03
Uživatelé mají role	0,09	0,09	0,09	0,09	0,09
Integrovaná nápověda	0,03	0,01	0,03	0,01	0,01
Ochota tvůrců komunikovat	0,03	0,01	0,01	0,01	0,03
Součet	**2,64**	**2,8**	**2,56**	**2,22**	**2,33**

Obrázek 5.3: Tabulka s výsledky multikriteriálního hodnocení

6 Squash TM

Nástroj je dostupný z oficiálních stránek `https://www.squashtest.com/community-news`, a to ve verzi 1.21.0, která byla vydána v prosinci 2019. Od této verze lze nástroj spouštět pod Java SE Development Kit 11, což zásadně usnadnilo jeho používání, protože předchozí verze 1.20.x vyžadovala Javu 8. Nástroj vytvořila a rozvíjí francouzská firma, což má vliv i v některých aspektech jeho dokumentace. Stránky bohužel nepůsobí příliš dobrým dojmem, nejsou optimalizované a, ačkoliv tuto možnost nabízí, většina textu není přeložena do anglického jazyka, narozdíl od podrobné dokumentace, která je kompletně v angličtině. Po stažení archivu s nástrojem můžeme ihned nástroj používat, což je také jeden z důvodů, proč je tento nástroj vhodný pro výuku v předmětu KIV/OKS. Následné informace jsou převzaty z [10] a modifikovány podle vlastních zkušeností s používáním.

Program se ovládá pomocí vertikálního a horizontálního menu. První zmiňované obsahuje prostředí, která jsou typická pro tento nástroj. Ty oddělují logicky rozdílnou funkcionalitu. Jejich účel je přehledné zobrazení jedné funkcionality, a tím zároveň snižují riziko, že by se uživatel v projektu ztratil. Je jich celkem 6, případně 7, má-li uživatel projekt propojený s nástrojem pro správu defektů. Jedná se o tyto prostředí:

- **Requirement Workspace** – Prostředí nabízí správu požadavků, tj. vytvoření RQS (Requirement Suite) / RQM, následně jejich editace, popřípadě smazání. Dále ještě umožňuje jejich import a export.
- **Test Case Workspace** – Správa testovacích případů, poskytuje stejné funkcionality jako předchozí prostředí pro TS (Test Suite) / TC.
- **Campaign Workspace** – Zajišťuje správu kampaní, které slouží pro shlukování jednotlivých iterací testů, ve kterých si lze prohlédnou výsledky dílčích testů.
- **Home Workspace** – Slouží pro zobrazení úvodní stránky. Tu si uživatel může přizpůsobit, například zobrazit nejrůznější přehledové, vysoce agregované grafy o stavu testování.
- **Management Workspace** – Prostředí pro vytváření reportů, grafů a tabulek z testovacích dokumentů. Umožňuje vytvořit graf výsledků proběhlých testů v závislosti na čase a následně jej exportovat v `PNG`

formátu. Důležitou možností je, že takto vytvořené grafy lze zobrazovat v dashboardech ostatních prostředí. Typicky je to právě v Home Workspace.

- **Automation Workspace** – Poskytuje správu automatických testů, ty ale musí být pro daný projekt aktivovány.
- **Bugtracker Workspace** – Viditelné pouze v případě, že je projekt propojen s nástrojem pro správu defektů. Zajišťuje rychlý přesun mezi nástroji přesměrováním na URL adresu, zadanou při propojení. Momentálně podporuje nástroje Mantis BT a Jira.

Horizontální menu nabízí čtyři položky:

- **Global filter** – Globální filtr umožňuje zobrazení, respektive schování testovacích dokumentů zvoleného projektu.
- **Administration** – Poskytuje veškeré nastavení, které vyžaduje administrátorská práva. Jedná se například o správu uživatelů a vytváření projektů.
- **My account** – Zajišťuje úpravu přihlášeného uživatele, například možnost změny hesla svého účtu.
- **Logout** – Pomocí této položky je umožněno uživateli odhlásit se z aplikace.

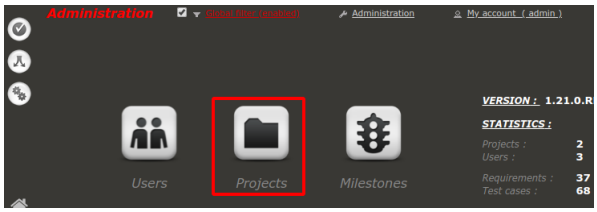
6.1 Vytvoření projektu

Nejdůležitějším krokem při testování je vytvoření testovacího projektu. Ten reprezentuje nejvyšší jednotku v hierarchii.

Pro vytvoření projektu v nástroji Squash TM je potřeba se nacházet v sekci pro administraci, do které se lze dostat jedině v případě, že má uživatel roli administrátora. Do této sekce se dostaneme po zvolení položky „Administration“ z horizontálního menu viz 6.1.

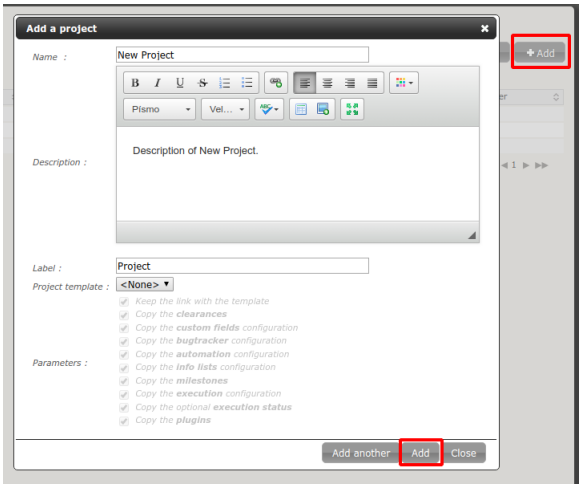
Obrázek 6.1: Položka administrace horizontálního menu

V ní poté zvolíme položku „Projects“ 6.2, která nám zobrazí všechny dosud vytvořené projekty.



Obrázek 6.2: Zobrazení správy projektů

Následně klikneme na tlačítko „Add“ s ikonou plus (obr. 6.3). Zobrazí se formulář, ve kterém vyplníme jméno projektu a případně i jeho popis a label a potvrdíme tlačítkem „Add“ bez ikony. Tímto je projekt vytvořen, stačí jen z menu pomocí položky „Global filter“ zaškrtnout nově vytvořený projekt pro jeho zviditelnění.

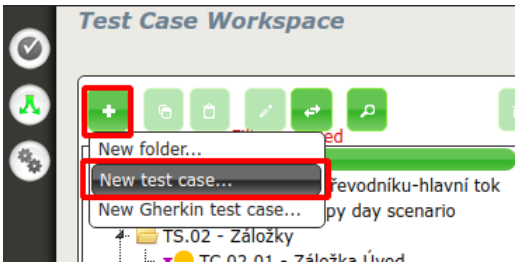


Obrázek 6.3: Vytvoření projektu

6.2 Vytvoření testovací případu

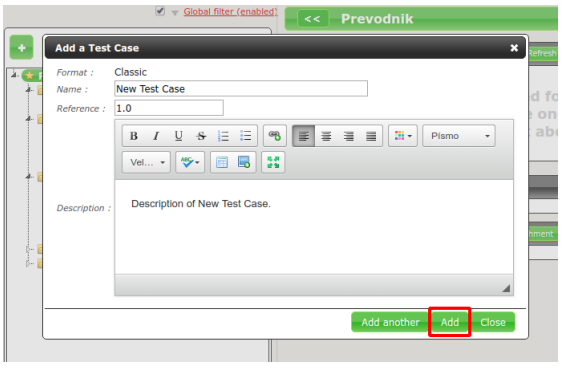
Testovací případ v nástroji Squash reprezentuje scénář testu a tvoří základní jednotku pro testování. Tyto jednotky se mohou dále shlukovat do složek tzv. „Test Folder“, které představují Test Suite a umožnují hierarchické členění.

Pro jejich správu slouží již zmiňované prostředí „Test Case Workspace“, ve kterém je připraveno tlačítko s ikonou viz obr. 6.4, které se aktivuje po zvolení jedné ze složek, nebo jednoho z projektů, jejichž přítomnost může být ovlivněna přes globální filtr.



Obrázek 6.4: Zobrazení formuláře pro vytvoření testovacího případu

Po zvolení možnosti „New test case...“ je zobrazen formulář ve stylu pop-up okna. Ten obsahuje povinnou položku „Name“ pro jméno testovacího případu a dvě nepovinné položky „Reference“ a „Description“. Po vyplnění se uloží tlačítkem „Add“ viz obr. 6.5. V případě, že uživatel má v úmyslu přidávat více testovacích případů, je mu umožněno jej uložit přes tlačítko „Add another“, které navíc nezavře pop-up okno a resetuje veškeré položky formuláře, což ušetří uživateli čas.



Obrázek 6.5: Uložení testovacího případu

6.3 Vytvoření požadavku

Požadavek se nachází ve stejné úrovni jako testovací případ a je mu také umožněno seskupovat se do složek, v tomto případě tzv. „Requirement Folder“. Jeho vytvoření je analogické k vytvoření testovacího případu, což je pro uživatele velice intuitivní. Rozdíl je ten, že se uživatel musí nacházet v prostředí „Requirement Workspace“ a ve formuláři se vyskytují navíc dvě položky pro „Criticality“ (popisuje důležitost požadavku) a „Category“ (rozděluje požadavek do skupin, kde každá skupina má svoji ikonu).

6.4 Vytvoření kampaně

Kampaň představuje jednotku pro plánování testů. Umožňuje vytvářet jednotlivé iterace, které zavádí rozměr času do testování. Pro vytvoření kampaně je připraveno prostředí „Campaign Workspace“. Činnost je stejná jako v předchozích případech a formulář je totožný s formulářem pro vytvoření testovacího případu.

7 Squash TM API

7.1 API REST plugin

Jedna z předností tohoto nástroje je možnost doplnění funkcionality přes připravené pluginy. Mezi tyto pluginy patří „API REST plugin“, který umožňuje komunikovat s nástrojem bez nutnosti použití webového rozhraní. Toho je docíleno vzájemnou výměnou JSON souborů zasílaných pomocí HTTP požadavků.

Součástí pluginu je kvalitní a podrobně anglicky popsaná dokumentace, která je i volně k dispozici na webové stránce https://demo.squashtest-.org/squash/api/rest/latest/docs/api-documentation.html. Pro přístup je nezbytné zadat uživatelské jméno „guest_tpl“ a heslo „password“.

Plugin je stále vyvíjen, aktuálně se nachází ve verzi 1.1.5. Pro jeho kvalitu a časté používání se jej autoři rozhodli integrovat přímo do aplikace Squash TM, a to konkrétněji od její verze 1.18.0.

API podporuje tyto HTTP metody – `GET`, `POST`, `PATCH`, `DELETE` a tím umožňuje posílat, vytvářet, mazat a upravovat testovací dokumenty. Požadavky se posílají na tzv. Endpointy. Ty specifikují, na jakém místě se požadovaná data nacházejí. Ve Squash API se skládají ze základní adresy (Base URL) a jednotlivých identifikátorů. Base URL ve Squash vypadá následovně – `{squash root url}/api/rest/latest/`, přičemž `latest` zde představuje nejnovější verzi API. To může být zavádějící, neboť aktuálně autoři neplánují verzování API. Identifikátory na druhou stranu upřesňují o jakou entitu se jedná, například pro projekt – `/projects`. Pokud tedy máme v úmyslu vytvořit projekt ve Squash TM, který běží na defaultní adrese `http://localhost:8080/squash`, pošleme JSON soubor, který obsahuje informace o projektu v předepsaném formátu, pomocí HTTP metody `POST` na endpoint `http://localhost:8080/squash/api/rest/latest/projects`. Veškeré předepsané formáty JSON souborů jsou popsány v již zmiňované dokumentaci.

V průběhu zpracování této bakalářské práce bylo zjištěno, jak je API tohoto nástroje kvalitní a nabízí pokročilé možnosti. Pokud budou realizovány, pak výrazně pozitivně ovlivní jak snadnost použití Squash TM v předmětu KIV/OKS, tak i pro zamýšlené použití pro projekt TbUIS. To vedlo k posunutí důrazu této bakalářské práce na experimentování s tímto pluginem. Následné vyhodnocování výsledků práce bude nikoli na projektu TbUIS, ale na jednodušším cvičném projektu z KIV/OKS. Tento projekt (Převodník)

se podařilo zcela pokrýt požadavky a testovacími případy, a tak demonstrovat funkcionalitu vytvořených programů. Výsledkem těchto experimentů jsou 3 aplikace, které budou v této práci popsány.

7.2 Squash Test Result Writer

7.2.1 Analýza

Jedním z cílů této bakalářské práce je vytvoření aplikace, která umožní odesílání výsledků automatických testů přes API nástroje Squash TM. Tento požadavek vznikl na základě možnosti přehledného zobrazení výsledků testů, které Squash TM poskytuje. Díky této aplikaci bude nyní možné prohlédnout si jednotlivé výsledky automatických testů v nástroji Squash TM, a tím dostávat průběžné informace o progresu testování v závislosti na čase a využít značných možností vytváření sofistikovaných grafů v Squash TM.

Požadavky

Pro zajištění korektního chování aplikace bylo stanoveno několik požadavků. Níže je uveden výčet těchto požadavků.

- Implementace v programovacím jazyce Java, pro jednoduché propojení s frameworkem JUnit 5.
- Zvolení projektu, ve kterém se výsledky uloží.
- Vytvoření nové iterace po každém spuštění testů.
- Možnost zapsání 4 typů výsledku: `Passed`, `Failed`, `Blocked` a `Skipped`.

Návrh

Nejprve je nutné zajistit komunikaci se serverem Squash TM. To je možné zrealizovat pomocí modulu `java.net.http`. Ten poskytuje abstraktní třídu `HttpClient`, ze které lze vytvořit instanci s definovanou verzí HTTP, URL, portem a autentizací pomocí uživatelského jména a hesla. Poté lze využít abstraktní třídy `HttpRequest`, která zajišťuje odeslání HTTP metody a `HttpResponse`, která spravuje odpověď serveru. Její výhodou je, že není potřeba přidávat závislost další knihovny, neboť je začleněna do Javy, přesněji od verze `Java SE Development Kit 11`. Alternativou by byly moduly

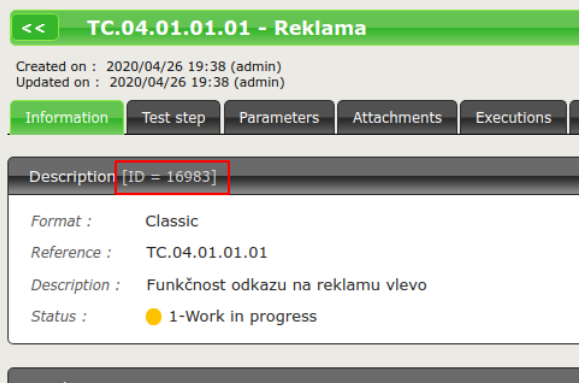
`org.apache.httpcomponents` a `com.squareup.okhttp`. Ty poskytují stejnou funkcionalitu, jako `java.net.http`.

Dále je nezbytné pracovat s JSON objekty. Java takovou funkcionalitu nativně neposkytuje, je tedy nevyhnutelné vybrat knihovnu umožňující tuto činnost. Jedním z kandidátů je `com.google.code.gson`. Ta umožňuje vytváření a čtení JSON objektů a podporuje generické typy, což se projevuje na její robustnosti. Další možností je `org.json.simple`, ta umožňuje také správu JSON objektů, ale narozdíl od předchozí knihovny funkcionalitu navíc neposkytuje a díky tomu nezabírá velké množství místa.

Zápis výsledků přes JUnit 5 framework lze uskutečnit dvěma způsoby. Jedním z nich je vytvoření třídy `SquashTestEvaluation`, jež bude implementovat rozhraní `TestWatcher`, které je poskytováno modulem `org.junit-jupiter.api.extension`. V této třídě se překryjí metody, které se volají na základě výsledku testu a přes parametr poskytují třídu `ExtensionContext`, jež obsahuje metodu `getDisplayName()`, která vrací identifikátor proběhlého testovacího případu. Jedná se o metody `testFailed(ExtensionContext ec, Throwable throwable)`, `testSuccessful(ExtensionContext ec)`, `testAborted(ExtensionContext ec, Throwable throwable)` a `testDisabled(ExtensionContext ec, Optional<String> optional)`. Důležité je přidat ke každé třídě obsahující testovací případy anotaci `@ExtendWith(SquashTestEvaluation)`.

Druhá možnost je vytvoření třídy, která bude implementovat rozhraní `TestExecutionListener` z balíku `org.junit.platform.launcher`. V této třídě bude překryta metoda `executionFinished`, která poskytuje identifikátor proběhlého testu a jeho výsledek.

Pro zápis výsledku testovacího případu je však nutné znát jeho identifilátor (`ID`), které mu bylo přiděleno při jeho vytvoření ve Squash TM (obr. 7.1).



Obrázek 7.1: ID testovacího případu ve Squash TM

API neposkytuje endpoint pro získání jeho `ID` podle jeho `reference` (identifikátor z JUnit testu, viz obr. 7.2).

```
@Test
@Tag(Tags.PASIVNI)
@Tag(Tags.SMOKE)
@DisplayName("TC_04.01_01.01: Titulek stránky")
void TC_04_01_01() {
  String titulek = Drivers.getDriver().getTitle();
  Check.checkText(titulek, Txt.TITULEK_APLIKACE, Id.PREV_NADPIS_FORMULARE_TN);
}
```

Obrázek 7.2: Identifikátor JUnit testu

Využije se tedy endpoint `{Base URL}/projects/{id}/test-cases-library/content` pro získání informací o veškerých testovacích případech projektu, ze kterých se uloží `ID` a `reference` do vhodné datové struktury, například `HashMap`, kde klíč bude `ID` a hodnota `reference`. Poté již bude možné získat požadované `ID` testovacího případu, a to v konstantním čase [5].

Poznámka: Při vytváření testovacích případů v nástroji Squash TM je nutné, aby se shodoval `Reference` s jeho identifikátorem, jinak nebude možné zapsat výsledek testu.

7.2.2 Implementace

Pro komunikaci mezi aplikací Squash TM byla použita zmiňovaná knihovna, a to z toho důvodu, že pro tento účel jsou zcela postačující a není nutné přidávat externí knihovnu. Dále pro práci s JSON objekty byla vybrána knihovna `org.json.simple`, kvůli její jednoduchosti.

Aplikace se skládá ze čtyř balíků – `apisquash`, `connection`, `evaluation` a `helpers`.

connection

Zajišťuje připojení k serveru, na kterém běží nástroj Squash TM a posílání HTTP metod. Skládá se ze dvou tříd. `ConnectionManager` využívá abstraktní třídu z knihovny `java.net.http` pro vytvoření HTTP klienta. Dále obsahuje metody `sendPostRequest(String body, String address)`, `sendPatchRequest(String body, String address)` a `sendGetRequest(String address)`. Ty využívají třídy `TestManager` a `ConnectToSquash` pro odesílání HTTP požadavků. `ConnectToSquash` obstarává inicializaci aplikace pomocí metody `init(String hostName, int portNumber, String userName, String userPassword, String projectName)`, která je dvakrát přetížená. To umožňuje inicializovat aplikaci s defaultními hodnotami, které jsou uloženy v pomocné třídě `Constants`.

apisquash

Balík obsahuje třídy `TestManager` a `TestType`. `TestManager` umožňuje vytvoření nezbytných testovacích dokumentů, například iterace a kampaně. Dále ještě poskytuje metodu `writeTestResult(String testName, TestType type)`, jež se stará o samotné zapsání výsledků testů do nástroje Squash TM. `TestType` je návrhový vzor `Výčtový typ` a reprezentuje možné výsledky testů. Jedná se o `SUCCESS`, `FAILURE`, `UNTESTABLE` a `BLOCKED`.

evaluation

Pro provázání aplikace s JUnit 5 frameworkem byl vytvořen balík `evaluation`. Ten obsahuje dvě třídy `SquashTestEvaluation` a `SquashTestExecutionListener`. Pro zápis výsledku testování obě třídy využívají metodu `writeTestResult(String testName, TestType type)` ze třídy `TestManager`.

helpers

Sdružuje pomocné třídy projektu. Jedná se o třídy `Constants`, `GeneralHelper` a `JsonHelper`. Třída `Constants` poskytuje defaultní nastavení pro připojení k nástroji Squash TM a používané endpointy. `JsonHelper` pracuje s knihovnou `org.json.simple` a poskytuje metody, které vrací JSON objekty pro jednotlivé HTTP požadavky. Třída `GeneralHelper` poskytuje ostatní pomocné metody. Jedná se například o vytvoření řetězce obsahující aktuální datum, používané pro defaultní pojmenování kampaně, ve formátu „yyyy-MM-dd“.

7.2.3 Použití

Pro použití aplikace je nutné mít připravené JUnit testy. Poté do metody, ze které jsou testy spouštěny, se vytvoří instance třídy `ConnectToSquash` a přes metodu `init(String hostName, int portNumber, String userName, String userPassword, String projectName)`, popřípadě přes některou z jejich přetížených verzí, se připojí k nástroji Squash TM. V případě, že jsou testy spouštěny přes třídu `LauncherDiscoveryRequest` z knihovny `org.junit.platform.launcher`, je možné zapisovat výsledky přes tzv. „Execution listener“ (Listener). Ten se ke třídě `LauncherDiscoveryRequest` registruje tímto způsobem: `launcher.registerTestExecutionListeners(newSquashTestExecutionListener());`

V druhém případě se ke každé třídě obsahující jednotlivé testy přidá anotace `@ExtendWith(SquashTestEvaluation)` (Watcher).

8 Provázání případů užití na požadavky

Pro vizualizaci provázání případů užití na požadavky byl vedoucím práce vytvořen program generující HTML stránku. Ta obsahuje tabulku (obr. 8.1), ve které jsou znázorněny jednotlivé požadavky a případy užití (UC).

▼ Criticality Sum ↪				Use cases		
				01	02	03
7	11	17	←35 35→ ↓35	5	14/13	3
RQS.01			1		1	
RQS.02			3	1	1	1
RQS.03			4	4		
RQS.04			25		12/13	
RQS.05			2			2

Obrázek 8.1: Tabulka zobrazující provázání UC na RQM

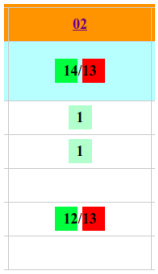
Na řádcích jsou umístěny skupiny požadavků a jednotlivé požadavky, zatímco sloupce představují případy užití. Požadavky jsou pro větší přehlednost schovány atributem `class=".collapsed"`, kterému je přes CSS (Cascading Style Sheets) selektor přidán styl `visibility: collapse`. Pro jejich zobrazení je nutné tuto třídu ručně smazat. Je tedy zřejmé, že tabulce chyběla logika, která by tuto funkcionalitu obstarávala automaticky. Na obrázku. 8.2 lze vidět rozbalenou skupinu požadavků.

RQS.05	2			2
RQM.05.01				1
RQM.05.02				1

Obrázek 8.2: Rozbalená skupina požadavků

V první buňce řádku tabulky se nachází prefix RQS, či RQM. Druhá buňka znázorňuje v případě RQS počet obsažených RQM, na druhou stranu pro RQM se místo celkového počtu (který by byl vždy roven jedné) zobrazuje ikona, reprezentující jeho naléhavost. Ikony představují ⏫ kritickou, 🔺 majoritní a 🔻 minoritní naléhavost.

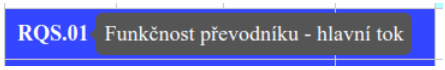
Sloupce jsou od požadavků barevně odlišeny. Příklad sloupce je zobrazen na obrázku 8.3.



Obrázek 8.3: Sloupec s UC

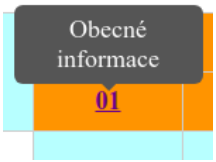
V první buňce sloupce se nachází prefix případu užití, který slouží i jako odkaz na daný UC. Druhá buňka poskytuje celkový počet prošlých / neprošlých výsledů testů, vztahujících se k požadavkům daného UC. Společné buňky pak již obsahují počet prošlých / neprošlých výsledů testů, vztahujících se k danému RQS, či RQM na řádce a daného UC ve sloupci.

V rámci bakalářské práce byla interaktivnost této tabulky významně vylepšena přidáním skriptu v programovacím jazyce Javascript a také stylováním pomocí CSS. Ke všem prefixům byl přidán tooltip, zobrazující jméno položky. Tento tooltip je možné přidat k jakémukoliv elementu. K požadovanému elementu stačí přidat atribut `class='tooltip'` a vnořit do něj element s atributem `class='tooltipText right'` (tooltip je zobrazen vpravo od elementu), popř. `class='tooltipText top'` (tooltip je zobrazen nahoře od elementu), který bude obsahovat text tooltipu. Přidáním těchto tříd je poté pomocí CSS selektorů vytvořen tooltip pro daný element. Například pro kód `<span class='tooltip'>RQS.01<span class='tooltipText right'>Funkčnost převodníku - hlavní tok</span></span>` se tooltip zobrazí stejně jako na obr. 8.4.



Obrázek 8.4: Prefix a jméno RQS

Pro přehledné zobrazení tooltipu ve sloupcích je možné tooltip zobrazit nohoře od elementu. Například `<span class='tooltip'>01<span class='tooltipText top'>Obecné informace</span></span>` zobrazí tooltip jako na obr. 8.5.



Obrázek 8.5: Prefix a jméno UC

Dále byla přes Javascript vytvořena funkce `init()`, jenž se nachází uvnitř funkce `$(document).ready()`. Ta je poskytnuta knihovnou jQuery a zajišťuje spuštění kódu až poté, co je stránka načtena. `init()` registruje požadovaným elementům posluchače. Skupinám požadavků byl přidán posluchač na kliknutí myši a vytvořena funkce, která po tomto kliknutí rozbalí svůj obsah (v případě, že obsahuje další skupinu požadavků, je tato skupina nerozbalena). Ikonám pod celkovým počtem požadavků dané naléhavosti byl přidán opět posluchač na kliknutí myši, který spustí funkci pro zobrazení veškerých požadavků s danou naléhavostí. Poslední posluchač byl přidán celkovému počtu požadavků daného UC. Po následném kliknutí jsou tyto požadavky zobrazeny.

Dodaná interaktivita tedy prakticky umožňuje nezávisle sbalovat či rozbalovat:

- Hierarchii RQS/RQM,
- RQM dle naléhavosti,
- RQM dle příslušnosti UC,
- RQM dle výsledku testu (passed/failed).

Interaktivita vypadá sice jednoduše, nicméně pro její implementaci bylo napsáno přes 100 řádek kódu v Javascriptu.

Jelikož se jedná o jednoúčelové zobrazení výsledků, které bude v TbUIS přikládáno jako report ke každému poruchovému klonu (jedna HTML stránka), je výhodné, aby byl Javascript připojen přímo do HTML stránky přes element `<script>`. Zcela funkční report je tak tvořen pouze jedním souborem, který je díky tomu snadno přemístitelný.

9 Sada požadavků a jejich testovacích případů

Jak již bylo zmíněno dříve, byla po dohodě s vedoucím práce část aktivit bakalářské práce nasměrována na vytváření programů pro komunikaci se systémem Squash TM s využitím jeho API. Dále bylo v předchozí kapitole popsáno, jakým způsobem lze sekundárně využít požadavky zapsané do Squash TM a získané z něj pomocí exportů. Všechny programy a softwarové části vytvořené v rámci této práce bylo ale nutné ověřit na nějakém produktu. Opět po dohodě s vedoucím práce byla vybrána cvičná školní aplikace *Převodník*. Ta má tu výhodu, že je již dlouhou dobu stabilní a má takový rozsah, který umožní bezproblémové vyzkoušení vytvořených programů.

9.1 Převodník

Převodník vznikl v rámci předmětu KIV/OKS. Slouží jako pomocná aplikace pro jednu ze samostatných prací. Představuje webovou aplikaci pro reciproční převod imperiálních a metrických délkových jednotek. Obsahuje záměrně zanesené chyby, které mají studenti odhalit při testování. Stránka je volně dostupná na adrese http://oks.kiv.zcu.cz/Prevodnik.

Skládá se ze tří stránek `Úvod`, `Převodník` a `Nápověda`. `Úvod` stručně popisuje projekt a obsahuje tři odkazy – na web KIV, web ZČU a webovou stránku s informacemi o předmětu KIV/OKS. Na stránce `Převodník` se nachází dvě textová pole, jedno pro zadání hodnot jednotek pro převod a druhé pro zobrazení výsledku převodu, dva výběrové seznamy pro zvolení jednotek a dvě tlačítka pro zahájení převodu a pro smazání obsahu textových polí. Na poslední stránce je poskytnut návod na použití webové aplikace.

9.2 Sada požadavků

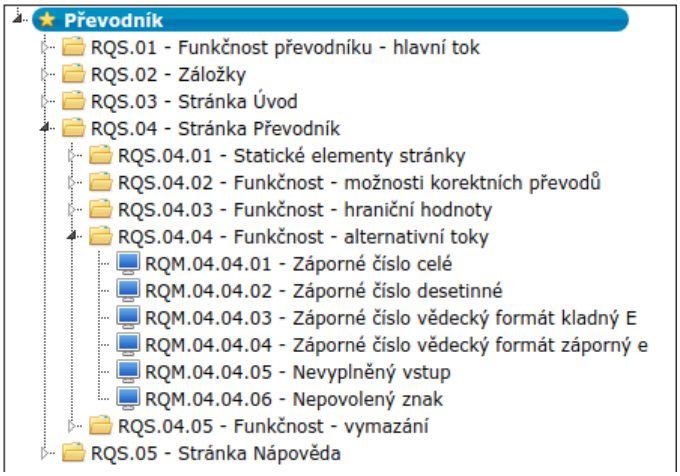
Na základě specifikace aplikace *Převodník* byla vytvořena sada požadavků. V souladu s teorií z přednášek KIV/OKS byly požadavky hierarchicky začleněny do skupin – tzv. RQS (Requirement Suite) a bylo pečlivě navrženo pojmenování. Skupiny požadavků obsahují prefix *RQS*, zatímco jednotlivé požadavky mají prefix *RQM*. Součástí prefixu je i číslování, které představuje úroveň vnoření, jedná se o dvoumístné číslo včetně nevýznamových nul.

Jednotlivé úrovně se oddělují tečkou. Například první požadavek v první skupině požadavků bude mít prefix `RQM.01.01`.

Návrh struktury sady požadavků je uveden v následující stromové struktuře.

```
Převodník
├── RQS.01: Funkčnost převodníku - hlavní tok
│   └── RQM.01.01: Happy day scenario
├── RQS.02: Záložky
│   ├── RQM.02.01: Záložka Úvod
│   ├── RQM.02.02: Záložka Převodník
│   └── RQM.02.03: Záložka Nápověda
├── RQS.03: Stránka Úvod
│   ├── RQM.03.01: Reklama
│   ├── RQM.03.02: Odkaz na KIV/OKS
│   ├── RQM.03.03: Odkaz na KIV
│   └── RQM.03.04: Odkaz na ZČU
├── RQS.04: Stránka Převodník
│   ├── RQS.04.01: Statické elementy stránky
│   │   ├── RQM.04.01.01: Titulek stránky
│   │   ├── RQM.04.01.02: Nadpis formuláře
│   │   ├── RQM.04.01.03: Label vstupního pole
│   │   ├── RQM.04.01.04: Textové pole Vstup
│   │   ├── RQM.04.01.05: Výběrový seznam pro vstupní hodnotu
│   │   ├── RQM.04.01.06: Label výstupního pole
│   │   ├── RQM.04.01.07: Textové pole Výstup
│   │   ├── RQM.04.01.08: Výběrový seznam pro výstupní hodnotu
│   │   ├── RQM.04.01.09: Tlačítko Převeď
│   │   └── RQM.04.01.10: Tlačítko Vymaž
│   ├── RQS.04.02: Funkčnost - možnosti korektních převodů
│   │   ├── RQM.04.02.01: Převody stejných jednotek
│   │   ├── RQM.04.02.02: Všechny varianty korektně zadaného
│   │   │   vstupu
│   │   ├── RQM.04.02.03: Převody všech jednotek na metry
│   │   ├── RQM.04.02.04: Převody metrů na všechny jednotky
│   │   └── RQM.04.02.05: Převody všech kombinací všech jednotek
│   ├── RQS.04.03: Funkčnost - hraniční hodnoty
│   │   ├── RQM.04.03.01: Převod nuly
│   │   └── RQM.04.03.02: Převod Double.MAX_VALUE
│   ├── RQS.04.04: Funkčnost - alternativní toky
│   │   ├── RQM.04.04.01: Záporné číslo celé
│   │   ├── RQM.04.04.02: Záporné číslo desetinné
│   │   ├── RQM.04.04.03: Záporné číslo vědecký formát kladný E
│   │   ├── RQM.04.04.04: Záporné číslo vědecký formát záporný e
│   │   ├── RQM.04.04.05: Nevyplněný vstup
│   │   └── RQM.04.04.06: Nepovolený znak
│   └── RQS.04.05: Funkčnost - vymazání
│       ├── RQM.04.05.01: Vymazání po úspěšném převodu
│       └── RQM.04.05.02: Vymazání po neúspěšném převodu
└── RQS.05: Stránka Nápověda
    ├── RQM.05.01: Reklama
    └── RQM.05.02: Odkaz na stránku Převodník
```

Celkem bylo tedy ve Squash TM podle tohoto návrhu vytvořeno 35 požadavků začleněných do 10 skupin požadavků. Pokud porovnáme zobrazení sady ve stromové struktuře s obrázkem 9.1 je zde viditelný rozdíl v přehlednosti. Pro udržení přehlednosti ve větších projektech, který obsahuje stovky až tisíce požadavků, je zcela zřejmé, že použití nástroje typu „Test management tool“ je nezbytné.



Obrázek 9.1: Zobrazení sady požadavků v nástroji Squash TM

Zapsané požadavky byly ze Squash TM exportovány do JSON souboru pomocí aplikace `Squash Manager` (viz kapitola 10) a následně pak byly zpět do Squash TM přes tuto aplikaci kontrolně importovány a výsledek správného importu byl ve Squash TM vizuálně ověřen.

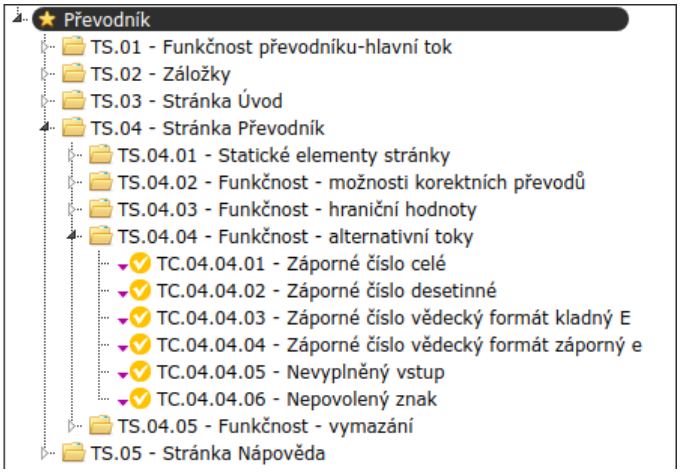
9.3 Sada testovacích případů

Na základě těchto požadavků je možné vytvořit jednotlivé testovací případy. Ty se také shlukují do testovacích sad – zde TS (Test Suite) a je dodržována obdobná strategie pojmenování. Pro testovací sadu je použit prefix `TS`, zatímco samostatné testovací případy obsahují prefix `TC`. Nejprve se z veškerých RQS vytvoří adekvátní TS a poté se z jednotlivých RQM vytvoří TC. V některých případech je ale nezbytné tento požadavek pokrýt nikoli pouze jedním testovacím případem, ale celou testovací sadou. Následkem je nárůst jedné úrovně v hierarchii.

Vzhledem k počtu testovacích případů jsou v následující stromové struktuře zobrazeny jen testovací sady.

```
Převodník
├── TS.01: Funkčnost převodníku - hlavní tok
├── TS.02: Záložky
├── TS.03: Stránka Úvod
├── TS.04: Stránka Převodník
│   ├── TS.04.01: Statické elementy stránky
│   │   ├── TS.04.01.01: Reklama
│   │   ├── TS.04.01.02: Nadpis formuláře
│   │   ├── TS.04.01.03: Label vstupního pole
│   │   ├── TS.04.01.04: Textové pole vstup
│   │   ├── TS.04.01.05: Výběrový seznam pro vstupní hodnotu
│   │   ├── TS.04.01.06: Label výstupního pole
│   │   ├── TS.04.01.07: Textové pole Výstup
│   │   ├── TS.04.01.08: Výběrový seznam pro výstupní hodnotu
│   │   ├── TS.04.01.09: Tlačítko převeď
│   │   └── TS.04.01.10: Tlačítko vymaž
│   ├── TS.04.02: Funkčnost - možnosti korektních převodů
│   │   ├── TS.04.02.01: Převody stejných jednotek
│   │   ├── TS.04.02.02: Všechny varianty korektně zadaného
│   │   │   vstupu
│   │   ├── TS.04.02.03: Převody všech jednotek na metry
│   │   ├── TS.04.02.04: Převody metrů na všechny jednotky
│   │   └── TS.04.02.05: Převody všech kombinací všech jednotek
│   ├── TS.04.03: Funkčnost - hraniční hodnoty
│   ├── TS.04.04: Funkčnost - alternativní toky
│   └── TS.04.05: Funkčnost - vymazání
└── TS.05: Stránka Nápověda
```

Celkově bylo vytvořeno 64 testovacích případů vnořených do 25 testovacích sad. Dále bylo postupováno zcela stejně, jako v případě požadavků, tj. ve Squash TM zapsané TS a TC byly exportovány do JSON souboru a následně zpět importovány do Squash TM. Výsledek importu do nástroje Squash TM je možné vidět na obr. 9.2



Obrázek 9.2: Zobrazení sady testovacích případů v nástroji Squash TM

10 Squash Manager

Programy pro export a import do Squash TM mezitím souběžně využíval vedoucí práce pro práci na probíhajícím projektu UIS. Na základě jeho požadavku byla přidána možnost off-line importů výsledků testů do Squash TM.

10.1 Analýza

V rámci experimentování s API bylo zjištěno, že lze vytvořit poměrně sofistikovanou aplikaci pro import a export testovacích dokumentů přes již zmiňované API. Squash TM tuto funkcionalitu sice přímo poskytuje, nicméně ta obsahuje významné nedostatky. Například nemožnost importovat a exportovat složky, což má za následek narušení hierarchického uspořádání. Vzhledem ke kvalitě poskytovaného API je tedy možné naprogramovat aplikaci pro export a import testovacích dokumentů, která nebude obsahovat tyto nedostatky.

10.1.1 Požadavky

Aplikace je cílena především pro studenty předmětu KIV/OKS, proto je tedy kladeno na aplikaci pár specifických požadavků:

- grafické uživatelské rozhraní pro jednoduché ovládání,
- rozhraní příkazového řádku pro případ vyskytnutí problému s GUI verzí,
- možnost připojení s defaultními hodnotami,
- důsledné detailní informování o průběhu akce, které je důležité i pro případnou kontrolu.

10.1.2 Návrh

Z požadavků je jasné, že bude nutné vytvořit dvě aplikace. Jedna bude poskytovat grafické rozhraní, zatímco druhá bude v CLI (Command Line Interface) verzi. Obě aplikace sdílejí stejnou logiku, bude tedy vhodné je umístit do jednoho projektu. Jediný rozdíl v logice bude výpis důležitých informací. Ty budou pro GUI (Graphical User Interface) verzi zobrazeny

přes komponentu pro zobrazení textu, zatímco v CLI budou vypisovány do konzole.

Připojení k nástroji Squash TM je i v tomto případě stejné jako v předchozí aplikaci. Stejně tak i práce s JSON soubory, vše je již popsáno v podsekci 7.2.1.

Pro zachování konzistence bude i zde použit programovací jazyk Java. Pro vytvoření GUI aplikace v tomto jazyce je nutné použít framework. Mezi nejvhodnější kandidáty se řadí – `AWT`, `Swing` a `JavaFX`. `Swing` a `AWT` jsou nativní frameworky, ale na druhou stranu neposkytují pokročilé možnosti jako `JavaFX`.

10.2 Formáty souborů

Pro import testovacích dokumentů je nutné mít JSON soubor v přesně definovaném formátu. Celkem byly navrženy 3 formáty, které jsou dále upraveny a přes API uloženy do Squash TM. Vzor JSON souboru pro import RQS/RQM je následující:

```
[
  {
    "rqs": {
      "name": "Funkčnost převodníku - hlavní tok",
      "prefix": "RQS.01",
      "description" : "<p>Očekávaná funkčnost převodníku -
         hlavní tok (Happy day scenario)</p>",
      "rqmList": [
        {
          "rqm": {
            "name": "Happy day scenario",
            "prefix": "RQM.01.01",
            "description": "<p>Funkčnost převodu z~palců (in)
                na milimetry (mm)</p>",
            "criticality": "major",
            "category": "functional"
          }
        }
      ]
    }
  }
]
```

Pro import TS/TS slouží následující formát:

```
[
  {
    "ts": {
      "name": "Nadpis formuláře",
      "prefix": "TS.04.01.02",
      "description" : "<p>Existence a pojmenování nadpisu </p
         >",
      "tcList": [
        {
          "tc": {
            "name": "Nadpis formuláře",
            "prefix": "TC.04.01.02.01",
            "prerequisite": "<p>Je zvoleno URL <a href='http
               ://oks.kiv.zcu.cz/Prevodnik'>http://oks.kiv.
               zcu.cz/Prevodnik</a></p>",
            "description": "<p>Testuje existenci a pojmenován
               í nadpisu <strong>Převodník jednotek délky</
               strong></p>",
            "importance": "medium",
            "nature": "functional",
            "type": "compliance",
            "steps": [
              {
                "action": "<p>Ověření existence a pojmenování
                     nadpisu <strong>Převodník jednotek délky
                    </strong></p>",
                "expected_result": "<p>Nadpis <strong>Převodn
                   ík jednotek délky</strong> existuje</p>"
              }
            ],
            "verified_requirements": [
              {
                "id": "04.01.02"
              }
            ]
          }
        }
      ]
    }
  }
]
```

Poslední vzor JSON souboru je pro offline import výsledků testů:

```
{
  "campaign": "Převodník",
  "iteration": "1. iterace",
  "results": [
    {
      "id": "TC.04.01.02.01",
      "result": "succes"
    }
  ]
}
```

10.3 Implementace

Byl vytvořen projekt obsahující dvě aplikace – `Squash Manager GUI` a `Squash Manager CLI`. Výpisy důležitých informací pro GUI verzi jsou realizovány přes komponentu `TextFlow`, kde pro znázornění chybové hlášky je změněna barva textu na červenou. Na druhou stranu pro CLI jsou tyto informace vypisovány pomocí metody pro zápis na standardní výstup `System.out.println()`, popřípadě `System.err.println()` pro chybové hlášky.

Pro vytvoření GUI aplikace byl zvolen framework `JavaFX`. Pro jeho výběr, mimo pokročilých možností, bylo rozhodnuto i pro možnost použití tzv. „WYSISYG“ (What You See Is What You Get) editoru. Jedná se o nástroj `SceneBuilder` a v něm byly vytvořeny veškeré scény pro aplikaci Squash Manager GUI. Framework je do aplikace zakomponován přes nástroj `Apache Maven`. Ten spravuje závislosti projektu a zajišťuje jejich správné stažení, a tím je možné jednoduše vytvořit výslednou `.jar` aplikaci na různých platformách.

V průběhu implementace bylo zjištěno, že vytvořený JSON soubor knihovnou `org.json.simple` v rámci exportu testovacích dokumentů je pro člověka nepřehledný (vše je v jedné řádce). Byla tedy přidána ještě jedna knihovna pro práci s JSON soubory – `com.google.code.gson`, která umožňuje vytvářet JSON soubory v „human-readable“ formátu.

Projekt obsahuje dvě složky – `java` a `resource`. `java` obsahuje veškeré zdrojové soubory, zatímco složka `resource` obsahuje ikonu, vytvořené scény a `.css` soubory se styly. Stromová struktura projektu vypadá následovně:

```
main
├── java
│   ├── cli
│   ├── gui
│   │   ├── helpers
│   │   └── scenes
│   └── logic
│       ├── api
│       └── helpers
└── resource
    ├── images
    ├── styles
    └── views
```

10.3.1 cli

Obsahuje třídu `SquashManager`, která se stará o spuštění CLI aplikace. Nejprve validuje předané parametry a pokud nejsou parametry v předepsaném formátu, je do konzole vypsáno použití aplikace. V případě platných parametrů se provede připojení k nástroji Squash TM. Uživatel je dotázán, zdali chce použít defaultní nastavení pro připojení a jestliže odmítne, je mu umožněno zadat jednotlivé parametry pro připojení. Tato možnost se využije v případě, že uživatel se potřebuje přihlásit pod jiným účtem, nebo nástroj Squash TM běží na jiné adrese než defaultní.

Příkazem `squash-plugin [-p project-name] [-r rqm-json-file] [-t tc-json-file]` je umožněn import TC a RQM. Pro jejich export je připraven příkaz `squash-plugin [-e project-name]`. Poslední příkaz `squash-plugin [-p project-name] [-c result-json-file]` zajišťuje import výsledků z proběhlých testů.

10.3.2 gui

Zahrnuje zdrojové soubory pro spuštění aplikace Squash Manager GUI. Nachází se v ní balíky `helpers` a `scenes` a třídy `SceneManager` a `SquashManager`. Balík `helpers` poskytuje rozměry jednotlivých oken aplikace a jejich názvy a texty použité v aplikaci. Balík `scenes` poskytuje třídy pro správu scén. Pro každou scénu byla vytvořena jedna třída, která se stará o její správné zobrazení a na základě zásahu uživatele ji upravuje. Třída `SceneManager` zobrazuje scény a spravuje jejich přechod. `SquashManager` se stará

o spuštění aplikace.

logic

Poskytuje balíky `helpers` a `scenes`, které zajišťují správné provedení akcí CLI a GUI aplikace. Balík `helpers` obsahuje třídy `Constants`, `GeneralHelper` a `JsonHelper`, které byly převzaty z aplikace `Squash Test Result Writer`. Balík `api` komunikuje s API nástroje Squash TM. Třída `ExportManager` poskytuje metodu `exportJson(String fileName, String projectId, TestDocumentType type)`, která exportuje testovací dokument do JSON souboru. Pro import JSON souborů slouží metoda `importJson(String fileName, String projectId, TestDocumentType type)` ze třídy `ImportManager`. Pro rozlišení, jaký testovací dokument se importuje, či exportuje byla vytvořena třída `TestDocumentType`, která slouží jako `Enum` pro tyto dokumenty.

10.3.3 images

Obsahuje ikonu pro aplikaci Squash Manager GUI.

10.3.4 styles

Sdružuje `.css` soubory, které upravují vzhled vytvořených scén.

10.3.5 views

Jednotlivé scény aplikace Squash Manager GUI. Každá scéna reprezentuje jedno okno aplikace.

10.4 Vytvoření spustitelné aplikace

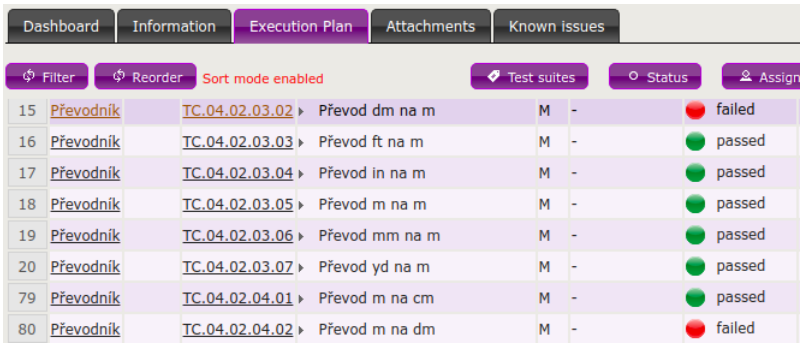
Pro používání aplikací `Squash Manager GUI` a `Squash Manager CLI` je nutné vytvořit spustitelný `.jar` soubor. K tomu je nutné mít k dispozici soubor `pom.xml` a složku `src`, která obsahuje složky `java` a `resources`, vnořené ve složce `main`. Dále je nutné mít nainstalovaný nástroj `Apache Maven`, který přes příkaz `mvn package` vytvoření spustitelné `.jar` soubory jak pro CLI, tak i pro GUI verzi. Tyto soubory jsou závislé na operačním systému, ve kterém byly vytvořeny. To znamená, že nelze vytvořit verzi GUI pro Linux zadáním příkazu `mvn package` v operačním systému Windows. Následné používání aplikací je popsáno v uživatelské příručce viz A.2.

11 Ověření a zobrazení výsledků

V dalším kroku byla ověřována funkčnost zápisu výsledků testů do nástroje Squash TM.

Pro toto ověření byl vedoucím práce poskytnut testovací program z dřívějška využívající Selenium WebDriver a technologii PageObject. Tento program bylo nutné přizpůsobit výše uvedené struktuře TS a TC.

Dále do něj byla integrována aplikace Squash Test Result Writer podle 7.2.3. Následně byly provedeny nezávislé on-line importy výsledků testů do Squash TM, a to oběma způsoby – Watcher i Listener. Z obrázku 11.1 lze vidět, že se v aplikaci Převodník vyskytují chyby, které tyto testy odhalily.

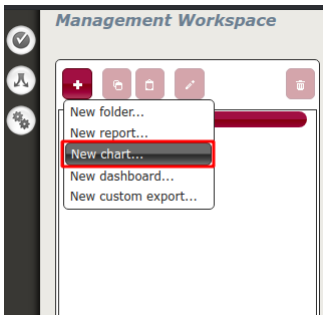


Obrázek 11.1: Výsledek automatických testů z aplikace Převodník

Díky automatickému zápisu výsledků testů bylo možné snadno provést několik kampaní s různými počty testů. Na výsledcích těchto importovaných kampaní bylo možné vyzkoušet tvorby grafů.

11.1 Tvorba grafu v průběhu testování ve Squash TM

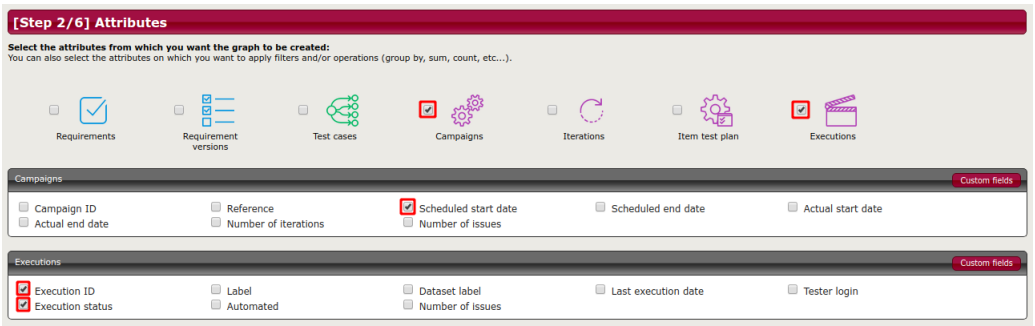
Vytvoření grafu je možné z prostředí `Management Workspace`, které poskytuje výběrové menu. Z výběru se zvolí položka „New chart“ viz obr. 11.2 a je doprovázeno průvodcem. Ten tvorbu rozděluje do šesti přehledných částí, které na sebe navazují.



Obrázek 11.2: Vytvoření grafu

První část slouží pro zvolení, z jaké kategorie potřebujeme data do grafu promítnout. Je zde na výběr jeden z existujících projektů, určité množiny požadavků, testovacích případů či kampaní.

Po vybrání kategorie se přes tlačítko „Next“ dostaneme do druhé části. V ní najdeme tlačítka pro výběr atributů, které se budou podílet na tvorbě grafu. To znamená, že zde vybereme nejen atributy, které chceme zobrazit, ale i ty, které nám je mohou filtrovat. Např. pokud chceme zobrazit výsledky testů z proběhlé iterace, zaškrtneme z nabídky položku „Iterations“ a z ní vybereme atribut „Iteration ID“, podle kterého filtrujeme veškeré proběhlé testy podle námi zadaného ID iterace. V našem případě budeme chtít zobrazit počet proběhlých testů v daných dnech a jejich výsledek. Vybereme položky „Campaigns“ a „Executions“, následně zvolíme atribut „Scheduled start date“, „Execution ID“ a „Execution status“ jako v obr. 11.3.

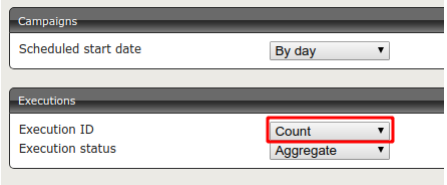


Obrázek 11.3: Výběr atributů

Třetí okno se zabývá nastavením filtrů pro vybrané atributy. Pokud bychom chtěli zobrazit pouze prošlé testy, vybrali bychom atribut „Execution status“, z výběrového menu pro modifikaci nastavili položku „Equals“ a nakonec zvolili z výběrového menu pro atribut hodnotu „3 - passed“. Tento

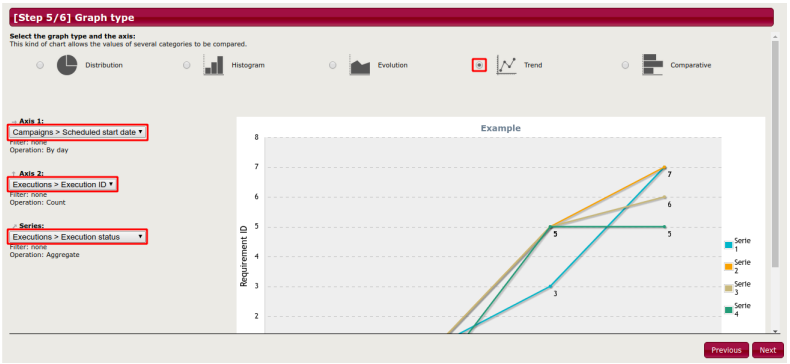
scénář vytvoří graf se všemi výsledky testů, takže v této sekci nic nebudeme nastavovat.

V následujícím části se upřesňuje interpretace atributů. Je zde na výběr „Sum“, „Count“ a „Aggregate“. „Sum“ s hodnotou atributu pracuje jako s číslem, „Count“ počítá množství položek pro daný atribut a „Aggregate“ vezme hodnotu atributu. Zde je nutné nastavit u atributu „Execution ID“ interpretaci na „Count“, viz obr. 11.4.



Obrázek 11.4: Nastavení atributů

Na předposledním okně jsou k dispozici typy grafů a výběr atributů pro zobrazení v grafu. V našem případě zvolíme typ grafu „Trend“ a popis os necháme jako na obr. 11.5.

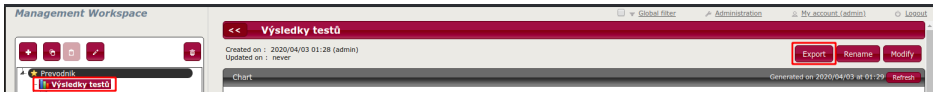


Obrázek 11.5: Výběr typu grafu

V posledním okně průvodce vidíme, jak bude náš graf vypadat po vytvoření, které se realizuje tlačítkem „Save“. K dispozici je zde pole pro nastavení jména grafu.

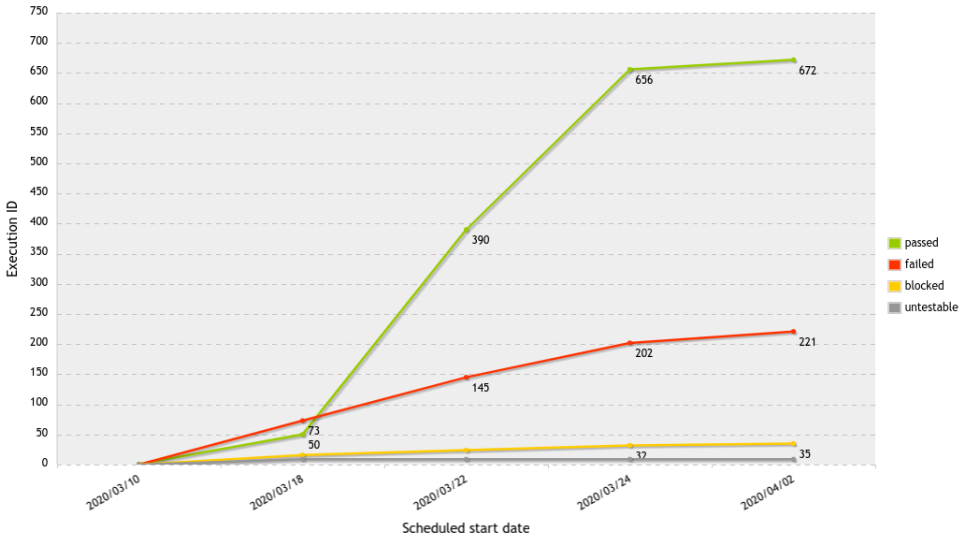
11.2 Export grafu

Po úspěšném vytvoření grafu lze tento graf exportovat tlačítkem „Export“ do formátu PNG viz 11.6.



Obrázek 11.6: Export grafu

Výsledný exportovaný graf lze vidět na obrázku 11.7.



Obrázek 11.7: Exportovaný graf

11.3 Pokrytí požadavků

Protože jsou k dispozici i exportované soubory RQM a TC ze Squash TM a též soubor výsledků testů (získaný logováním testů) je možné vytvořit HTML tabulku (viz kap. 8) zobrazující pokrytí UC požadavky včetně výsledků testů. V ní je možné procházet jednotlivé RQS a RQM a pozorovat jejich návaznost na UC, viz obr. 11.8.

▼ Criticality Sum ↪				Use cases		
				01	02	03
7	11	17	←35 35→ ↓35	5	14/13	3
RQS.01			1		1	
RQS.02			3	1	1	1
RQS.03			4	4		
RQS.04			25		12/13	
RQS.05			2			2

Obrázek 11.8: Tabulka s pokrytím aplikace Převodník

Funkčnost celého konceptu byla průběžně ověřována vedoucím práce na aplikace UIS. Na následujícím obrázku je uvedena ukázka stejného typu HTML souboru, ale pro aplikaci TbUIS.

▼ Criticality Sum ↪		Use cases																					
		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
41 38 180	←259 260→ ↓259	12	2	45	17	3	12	12	17	9	12	8	16	8	22	20	15	12	7	2	5/2	1	1
RQS.A	185	9	1	27	12	2	8	9	11	7	9	7	11	7	15	16	9	8	6	2	7	1	1
RQS.B	29	1	1	2	2	1	1	1	3	2	1	1	3	1	2	2	3	1	1				
RQS.C	31			10	3		2	2	2		1		2		2	2	3	2					
RQS.D	15	2		6			1		1		1				3			1					

Obrázek 11.9: Tabulka pokrytí požadavků aplikace UIS

12 Závěr

V bakalářské práci se podařilo pomocí multikriteriálního hodnocení nalézt vhodný nástroj kategorie plánování a řízení testů, který ve výuce KIV/OKS nahradil dříve používaný systém TestLink. Výběr nového nástroje Squash TM byl poměrně náročný, protože i přes velmi přesně stanovenou sadu požadavků na tento nástroj (dodaných vedoucím práce) je v této kategorii nástrojů stále značná konkurence. V nejužším výběru bylo podle 20 kriterií srovnáno celkem pět kandidátů. Jejich hodnocení byla v podstatě vyrovnaná (až na jednu výjimku), takže vítězný kandidát byl určen až v dalším kole, kdy ve spolupráci s vedoucím práce byla stanovena váhová kritéria pro jednotlivé požadavky.

Výběr nástroje byl však zdařilý, takže vedoucí práce jej zařadil (místo původního TestLinku) do výuky předmětu KIV/OKS již v akademickém roce 2019/20. Dosud nejsou známy žádné problémy, které by studenti s nástrojem měli, což nepřímo potvrzuje vhodnost výběru.

Existence API byla při výběru nástroje jedním z hlavních kriterií. Ovšem při samotném výběru nebylo možné u všech kandidátů prověřit skutečný rozsah API a jeho kvalitu. Ale i v tomto případě se nástroj Squash TM ukázal jako dobrá volba. Jeho API je kvalitní, udržované a velmi dobře dokumentované. Tato skutečnost, zjištěná až v průběhu řešení bakalářské práce, motivovala vedoucího práce k částečné změně zaměření druhé části práce – ověření nástroje Squash TM na projektu TbUIS. Těžiště této části práce bylo v souladu se zadáním práce zaměřeno na přípravu sady programů využívajících API nástroje, ovšem jejich ověřování proběhlo na školní testovací aplikaci Převodník. Výsledky jsou uvedeny v textu práce a v přílohách.

Aplikace UIS ale nebyla opomenuta, tyto nástroje na ní testoval paralelně vedoucí práce. Na základě jeho požadavků pak byla nad rámec zadání vytvořena podpora pro vizualizaci provázání případů užití na požadavky.

První sada vytvořených SW nástrojů pro import a export dat byla nasazena do použití ve výuce předmětu KIV/OKS souběžně s nástrojem Squash TM, tj. v letním semestru 2019/20. Použití desítkami studentů dostatečně prověřilo jejich kvality a robustnost.

Zadání práce bylo splněno v celém rozsahu.

Přehled zkratek

API	Application Programming Interface
RQM	Requirement
TC	Test Case
TbUIS	Testbed University Information System
SUT	System Under Test
UIS	University Information System
SŘBD	Systém Řízení Báze Dat
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
RQS	Requirement Suite
TS	Test Suite
UC	Use Case
CSS	Cascading Style Sheets
CLI	Command Line Interface
GUI	Graphical User Interface
JAR	Java Archive

Literatura

[1] *Top 20 Best Test Management Tools* [online]. [cit. 2019/11/11]. Dostupné z: https://www.softwaretestinghelp.com/15-best-test-management-tools-for-software-testers.

[2] *Top 15 Best Test Management Tools - Software Testing Class* [online]. [cit. 2019/11/11]. Dostupné z: https://www.softwaretestingclass.com/top-15-best-test-management-tools.

[3] *Best 25 Test Management Tools in 2019* [online]. [cit. 2019/11/11]. Dostupné z: https://www.guru99.com/top-20-test-management-tools.html.

[4] *TestLink Alternatives and Similar Software - AlternativeTo.net* [online]. [cit. 2019/11/11]. Dostupné z: https://alternativeto.net/software/testlink.

[5] *HashMap (Java Platform SE 8)* [online]. [cit. 2020/04/15]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html.

[6] *INSIGHTS: Why do we need a test management tool?- Inspired Testing* [online]. [cit. 2019/11/11]. Dostupné z: https://www.inspiredtesting.com/news-insights/insights/item/358-why-do-we-need-a-test-management-tool.

[7] *Multikriteriální analýza* [online]. [cit. 2019/11/15]. Dostupné z: http://spravnym.smerem.cz/Tema/Multikriteri%C3%A1ln%C3%AD%20anal%C3%BDza.

[8] *Expectations from a test management solution Part I- QASymphony* [online]. [cit. 2019/11/11]. Dostupné z: https://www.qasymphony.com/blog/what-testers-look-for-and-expect-from-a-test-management-solution.

[9] *Použití nástrojů v testování* [online]. [cit. 2019/11/12]. Dostupné z: http://test.swtestovani.cz/index.php?option=com_content&view=article&id=35.

[10] *Wiki Squash TM* [online]. [cit. 2020/04/15]. Dostupné z: https://sites.google.com/a/henix.fr/wiki-squash-tm.

[11] *TbUIS* [online]. [cit. 2019/11/13]. Dostupné z: `https://projects.kiv.zcu.cz/tbuis/web`.

[12] Matyáš, J. Aplikace s možností injekce chyb pro ověřování kvality testu. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2018.

[13] Poubová, J. Generování automatizovaných funkcionálních testů. Bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2019.

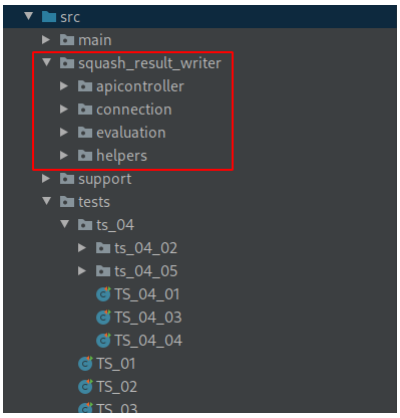
[14] Čarnogurský, J. Mikroframework na bázi komponent Symfony pro publikování vědeckých projektů. Bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2019.

[15] Šmaus, J. Rozhraní pro administraci aplikace s možností injekce chyb. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2019.

A Uživatelská příručka

A.1 Squash Test Result Writer

Pro online uložení výsledků spuštěných testů je nutné do aplikace, ve které jsou tyto testy spouštěny, přidat balík `squash_result_writer` do složky se zdrojovými soubory viz obr. A.1, nebo použít `.jar` a ten libovolně připojit.



Obrázek A.1: Vložení balíku mezi zdrojové soubory

Poté lze využít jednu variant, pro zápis těchto testů. První varianta je použití posluchače (Listener). Pro tuto variantu je nezbytné přidat do metody, ve které jsou testy spouštěny, vytvoření instance třídy `ConnectToSquash` a volání metody `init(String hostName, int portNumber, String userName, String userPassword, String projectName)`. To zajistí připojení k nástroji Squash TM. Následně se posluchač zaregistruje do třídy `Launcher` z modulu `org.junit.platform.launcher`. Příklad připojení posluchače lze vidět na obrázku A.2.

```
*/
public class Main {
    public static void main(String[] args) {
        ConnectToSquash connectToSquash = new ConnectToSquash();
        connectToSquash.init( projectName: "Prevodnik");

        LauncherDiscoveryRequest request = LauncherDiscoveryRequestBuilder.request()
                .selectors(
                        selectPackage( packageName: "tests.ts_04.ts_04_02")
                )
                .build();
        Launcher launcher = LauncherFactory.create();
        launcher.registerTestExecutionListeners(new SquashTestExecutionListener());
        launcher.execute(request);
    }
}
```

Obrázek A.2: Připojení posluchače do projektu

V případě použití dozorce (Watcher) se každé třídě obsahující jednotlivé testy přidá anotace `@ExtendWith(SquashTestEvaluation.class)` viz obr. A.3.

```
@TestMethodOrder(MethodOrderer.Alphanumeric.class)

@ExtendWith(TestBase.class)
@ExtendWith(LogAndScreenshotTestWatcher.class)
@ExtendWith(SquashTestEvaluation.class)
public class TS_04_01 {

  static PrevodnikPage prevodnikPage;

  @BeforeAll
  public static void setUpBeforaAll() {
    Click.openUrlAndWait(Url.URL_PREVODNIK);
    prevodnikPage = new PrevodnikPage();
  }

  @Test
  @Tag(Tags.PASIVNI)
  @Tag(Tags.SMOKE)
  @DisplayName("TC.04.01.01.01: Titulek stránky")
  void TC_04_01_01() {
    String titulek = Drivers.getDriver().getTitle();
    Check.checkText(titulek, Txt.TITULEK_APLIKACE, Id.PREV_NADPIS_FORMULARE_TN);
  }
```

Obrázek A.3: Přidání anotace pro zapojení dozorce do projektu

Samozřejmě i zde se musí přidat připojení k nástroji Squash TM. Například před spuštěním všech testů viz obr. A.4.

```
/**
 * One-time initialization of web driver
 * Possibly several others initializations, eg. Logger etc.
 *
 * @param context parameter which Selenium supplies
 */
@Override
public void beforeAll(ExtensionContext context) {
  if (started == false) {
    started = true;
    context.getRoot().getStore(GLOBAL).put( o: "ahoj",  o1: this);
    driver = Configurations.getInstance().setConfiguration();
    ProjectLogger.logStartInfo();
    ConnectToSquash connectToSquash = new ConnectToSquash();
    connectToSquash.init( projectName: "Převodník");
  }
}
```

Obrázek A.4: Připojení k nástroji Squash TM v rámci použití dozorce

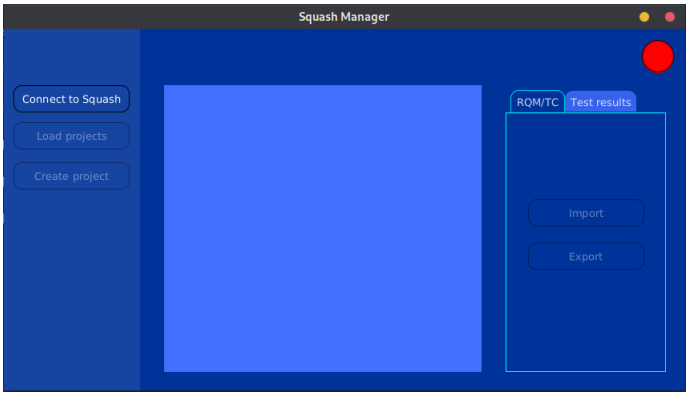
A.2 Squash Manager

Vytvoření GUI a CLI verze této aplikace vyžaduje nástroj Apache Maven (testováno na verzi 3.6.3) a Java SE 11. Příkazem `mvn package` (je nutné nacházet se ve složce se zdrojovými soubory a souborem `pom.xml` a mít správně nastaven Maven v prostředí systému) dojde k vygenerování složky `target`, která bude obsahovat spustitelný JAR soubor GUI verze (`squash-manager-gui.jar`) a CLI verze (`squash-manager-cli.jar`).

Poznámka: Pro následující podsekce se předpokládá, že je správně nastavena Java v prostředí systému a příkazy jsou spouštěny ze složky obsahující cílové soubory.

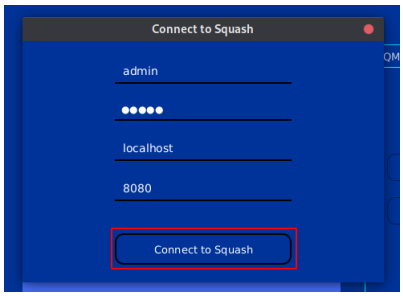
A.2.1 Squash Manager GUI

Spuštění aplikace Squash Manager GUI se provede následujícím příkazem – `java -jar squash-manager-gui.jar`. Zobrazí se hlavní okno aplikace viz obr. A.5. V levé části se nachází tlačítka pro připojení k nástroji Squash TM, načtení veškerých projektů a vytvoření projektu. Poslední dvě zmiňovaná tlačítka jsou zaktivována až po úspěšném připojení. Uprostřed aplikace se nachází seznam, který po načtení projektů bude obsahovat existující projekty. V pravé části jsou dostupná tlačítka pro import a export RQM/TC, popř. off-line import výsledků testů. Ty jsou zaktivovány po vybrání jednoho z projektů.

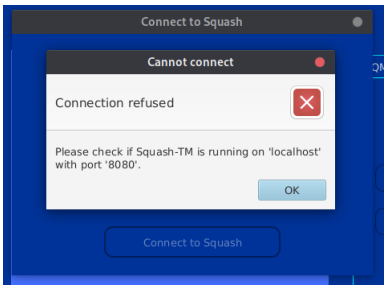


Obrázek A.5: Hlavní okno aplikace Squash Manager GUI

Nejprve je nutné připojit se k nástroji Squash TM. K tomu je připraveno tlačítko „Connect to Squash“ z levého menu. Po stisknutí je zobrazen formulář s vyplněnými defaultními hodnotami, které lze samozřejmě v případě potřeby změnit. Pro připojení slouží opět tlačítko „Connect to Squash“ viz obr. A.6. V případě, že Squash TM neběží na zadané adrese, je zobrazen alert informující o této skutečnosti (obr. A.7).

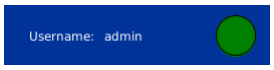


Obrázek A.6: Připojení k nástroji Squash TM

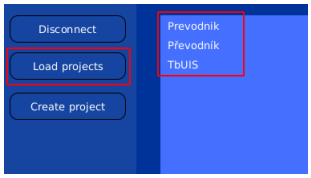


Obrázek A.7: Alert informující o selhání připojení

Pokud připojení proběhlo v pořádku (zobrazí se uživatelské jméno a zelená ikona viz obr. A.8) je možné načíst projekty, popřípadě jeden vytvořit. Načtení projektů obstarává tlačítko „Load projects“ a po úspěšném načtení jsou projekty zobrazeny v seznamu viz obr. A.9.



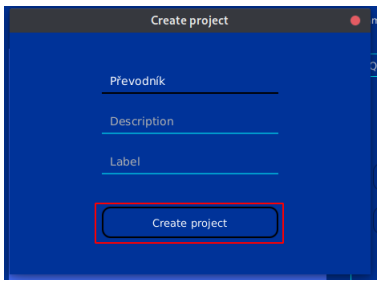
Obrázek A.8: Uživatelské jméno a ikona po úspěšném připojení



Obrázek A.9: Načtení projektů

Vytvoření projektu je umožněno přes tlačítko „Create project“ z levého menu, které zobrazí formulář pro vytvoření projektu. Ten obsahuje jednu povinnou položku pro jméno projektu a dvě nepovinné položky pro popis

a label projektu. Po vyplnění položek se projekt uloží tlačítkem „Create project“ viz obr. A.10.



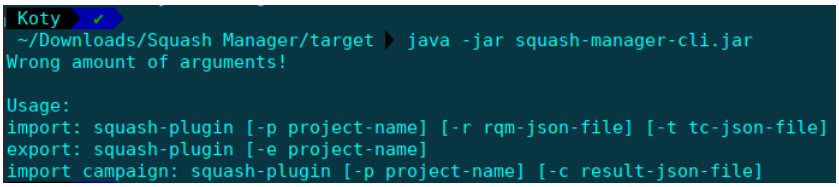
Obrázek A.10: Vytvoření projektu

Po zvolení jednoho z projektů je možné importovat, či exportovat testovací dokumenty. Pro import TC/RQM je nutné mít zvolenou záložku „RQM/TC“ z pravé části a stisknout tlačítko „Import“. Zobrazí se formulář ve kterém je nutné vybrat JSON soubory pro import RQM a TC. Ty musí být ve stanoveném formátu. Tlačítkem „Import“ se zahájí importování a pod tlačítkem se začnou vypisovat důležité informace. Export probíhá stejným způsobem přes tlačítko „Export“. Pro importování výsledků z testů je nezbytné vybrat záložku „Test results“ z pravé části. Poté je možné zobrazit formulář tlačítkem „Import“. V něm se vybere JSON soubor, ve kterém je mimo jiné zadáno jméno kampaně a iterace, do které se výsledky testů zapíší. V případě, že je jméno kampaně, popř. iterace prázdný řetězec, je vygenerováno ve formátu – `Campaign_<today>`, popř. `Iteration_<today>`, kde `<today>` je dnešní datum. V případě, že kampaň, popř. iterace pro dnešní datum již existuje, jsou výsledky zapsány do ní. Tlačítkem „Import“ se zahájí importovní. Důležité informace jsou opět vypisovány pod tlačítkem.

Pro odpojení z nástroje Squash je k dispozici tlačítko „Disconnect“, které po úspěšném připojení nahradilo tlačítko „Connect to Squash“, z levého menu.

A.2.2 Squash Manager CLI

Pro spuštění aplikace Squash Manager CLI je nutné zadat následující příkaz – `java -jar squash-manager-cli.jar` s parametry, nezbytnými pro provedení požadované činnosti. Ty jsou celkem tři – import RQM/TC, export RQM/TC a import výsledků z testů. Pokud nejsou předány parametry, nejsou ve správném pořadí, nebo nejsou korektní, je do konzole vypsána tato skutečnost a zároveň i použití aplikace viz obr. A.11.



Obrázek A.11: Ukázka spuštění aplikace s nevyhovujícími parametry

Při zadání korektních parametrů je uživatel dotázán, zdali chce použít defaultní hodnoty pro připojení a pokud uživatel odpoví negativně, je na tyto hodnoty dotázán. Pro import RQM/TC do existujícího projektu `Převodník` napíšeme příkaz:

```
java -jar squash-manager-cli.jar -p Převodník -r rqm.json -t
tc.json.
```

Pro jejich následný export zadáme příkaz:

```
java -jar squash-manager-cli.jar -e Převodník.
```

Ve složce, ve které jsme příkaz zadali, jsou vytvořeny dva soubory, jejichž jméno se skládá ze jména projektu, dnešního data a přípony RQM, popř. TC.

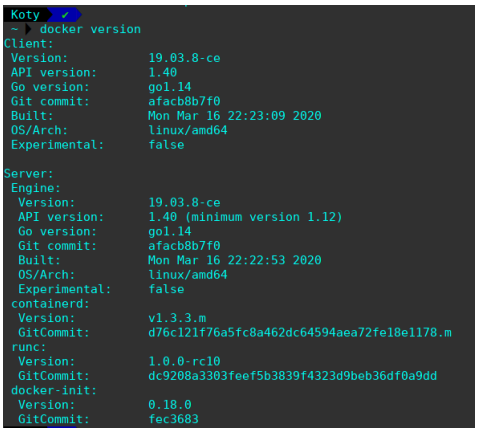
Import výsledků testů do projektu `Převodník` spustíme příkazem:

```
java -jar squash-manager-cli.jar -p Převodník -c results.json.
```

A.3 Spuštění Squash TM s databází MariaDB

A.3.1 Prerekvizity

Je nutné mít nainstalován software Docker (testováno s verzí 19.03.8), dostupný z https://www.docker.com/get-started. K ověření, zdali je Docker správně naistalován a v jaké verzi, slouží příkaz „`docker version`“, viz obr. A.12.



Obrázek A.12: Příkaz pro zjištění verze Dockeru

A.3.2 Stažení docker image

Nejprve je nutné stáhnout squasht-tm docker image. Příkaz „`docker pull squashtest/squash-tm`“ tento image stáhne do defaultního adresáře `/var/-lib/docker/` (Linux), popř. `C:\ProgramData\DockerDesktop` (Windows). Celkem se stáhnou 4 image. Jeden pro nástroj Squash TM a tři pro databáze – H2, PostgreSQL a MariaDB.

A.3.3 Spuštění MariaDB databáze

Pro spuštění databáze slouží příkaz:

```
docker run -it -d -name='squash-tm-mdb' -e MYSQL_ROOT_PASSWORD
=<heslo> -e MYSQL_USER=squashtm -e MYSQL_PASSWORD=<heslo> -
e MYSQL_DATABASE=squashtm -v squash-tm-db-mdb:/var/lib/mysql
mariadb:10.2.22-bionic -character-set-server=utf8mb4 -collati
on-server=utf8mb4_unicode_ci
```

a pro `<heslo>` se zadá heslo pro MySQL databázi a `squashtm` jméno pro MySQL databázi. Příkazy je možné zkopírovat z těchto stránek https://hub-.docker.com/r/squashtest/squash-tm odstavec „Deployment using MariaDB“.

Po stáhnutí nezbytných závislostí se nastartuje image `squash-tm-mdb`. V této chvíli je nutné počkat na inicializaci MariaDB databáze, což by mělo trvat zhruba 10 minut. Pro kontrolu, jak dlouho již image běží, lze zadat příkaz `docker ps -a` a ze sloupce „STATUS“ je možné vidět tento čas, viz obr. A.13.

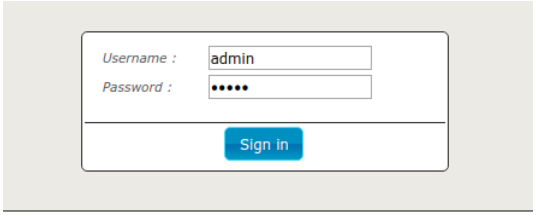
Obrázek A.13: Počet uběhlých minut od spuštění databáze

A.3.4 Spuštění Squash TM

Po uplynutí asi 10 minut od nastartování databáze můžeme spustit Squash TM příkazem:

```
docker run -it -d -name=squash-tm -link squash-tm-mdb:mysql -v
squash-tm-logs:/opt/squash-tm/logs -v squash-tm-plugins:/opt
squash-tm/plugins -p 8090:8080 squashtest/squash-tm
```

Nastartování aplikace trvá zhruba jednu minutu, poté zadáním adresy `localhost:8090/squash` dostaneme přihlašovací obrazovku. Zde vyplníme přihlašovací jméno „`admin`“ a heslo „`admin`“ viz A.14 a nyní můžeme Squash TM používat.



Obrázek A.14: Přihlášení do Squash TM