Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Bakalářská práce

Použitelnost automatizovaných testů v rámci vývoje projektu

Adam Grunt

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Adam Grunt

Systémové inženýrství a informatika
Informatika

Název práce

Použitelnost automatizovaných testů v rámci vývoje projektu

Název anglicky

Usability of automated tests in software project development

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku testování softwaru v rámci životního cyklu projektu. Hlavním cílem je testování základních funkcionalit systému prostřednictvím manuálních a automatizovaných testů a jejich porovnání.

1) Seznámení se s problematikou testování.
2) Popsání životního cyklu vývoje systému v rámci projektového řízení.
3) Popsání specifik automatizovaných a manuálních testů.
4) Analýza vhodnosti použití automatizovaných testů pro určitý proces v rámci projektu.
5) Zhodnocení, doporučení a navržení dalšího postupu.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. V teoretické části jsou popsány specifika manuálních a automatizovaných testů. Získané vědomosti z teoretické části jsou využity v praktické části pro analýzu vhodnosti použití automatizovaných testů pro zvolený proces.

Doporučený rozsah práce

30-60 stránek

Klíčová slova

projekt, testování, front-end, automatizace, agilita

Doporučené zdroje informací

A., Buchalcevová. 2005. Metodiky vývoje a údžby informačních systémů. Praha : Grada, 2005. 80-247-1075-7

R., Paton. 2002. Testování softwaru. Praha : Computer Press, 2002. 80-7226-636-5

SVOZILOVÁ, A. *Projektový management : systémový přístup k řízení projektů.* Praha: Grada, 2016. ISBN 978-80-271-0075-0.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Petra Pavlíčková, Ph.D.

Garantující pracoviště

Katedra systémového inženýrství

Elektronicky schváleno dne 25. 1. 2020

doc. Ing. Tomáš Šubrt, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 13. 2. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 21. 02. 2020

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Použitelnost automatizovaných testů v rámci vývoje projektu" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 22. března 2020 _____

Poděkování

Rád bych touto cestou poděkoval paní Ing. Petře Pavlíčkové, Ph.D., z katedry systémového inženýrství ČZU v Praze, za vedení práce, poskytnuté rady a konzultace, dále bych rád poděkovat kolegům a kamarádům za poskytnutou podporu a cenné rady.

Také bych chtěl poděkovat především své rodině, za veškerou podporu při psaní bakalářské práce a během celé doby studia.

Použitelnost automatizovaných testů v rámci vývoje projektu

Abstrakt

Práce je zaměřena na problematiku testování softwaru v rámci životního cyklu projektu za využití manuálního a automatizovaného způsobu testování daného procesu. Konkrétně se věnuje zanalyzování vhodnosti využití automatizace oproti manuálnímu způsobu testování z hlediska časové náročnosti.

Teoretická část popisuje životní cyklus vývoje softwaru v rámci projektového řízení. Část práce pojednává o životním cyklu vývoje softwaru v rámci agilních metodik a charakterizuje její typy. Dále práce rozebírá problematiku životního cyklu testování. Práce se také detailněji věnuje testování softwaru, cílům testování a detailním popisům pro jednotlivé typy testů podle úrovně a účelu využití. V závěru teoretické části se nachází popis manuálního a automatizovaného testování.

Praktická část práce se zabývá analýzou vhodnosti využívání, manuálního a automatizovaného způsobu testování. Na základě předem získaných informací z teoretické části a dokumentace produktu, jsou vytvořeny testovací případy. Pro ně jsou pak vytvořeny konkrétní manuální a automatizované testy. Po vytvoření testů je provedena jejich exekuce. Z naměřených hodnot časové náročnosti tvorby a exekuce pro všechny manuální i automatizované testy je provedena analýza vhodnosti využití daného způsobu testování.

V závěru práce jsou výsledky, do jaké míry a za jakých podmínek je výhodné využít automatizované testování oproti manuálnímu zhodnoceny, a na jejich základě je vytvořeno doporučení.

Klíčová slova: projekt, testování, front-end, automatizace, agilita

Usability of automated tests in software project development

Abstract

The work is focused on issues of software testing within the project life cycle using manual and automatized method of testing the process. Specifically, it focuses on analysing the suitability of automatization versus manual testing in terms of time.

The theoretical part describes the life cycle of software development within the project management. Part of the work deals with software development lifecycle within agile methodologies and characteristics its types. Furthermore, the thesis analyses the life cycle of testing. The thesis also deals with software testing, testing goals and detailed descriptions for individual types of tests according to the level and purpose of use. At the end of the theoretical part there is a description of manual and automated testing.

The practical part of the thesis deals with the analysis of suitability of use, manual and automated way of testing. Test cases are created on the basis of previously obtained information from the theoretical part and product documentation. Specific manual and automated tests are then created for them. After the tests are created, they are executed. From the measured values of time-consuming creation and execution for all manual and automated tests is analysed the suitability of the use of the method of testing.

At the end of the work, the results, to what extent and under what conditions it is advantageous to use automated testing compared to manual, are evaluated and based on them is made a recommendation.

Keywords: project, testing, front-end, automation, agility

Obsah

1 Úvod **12**

2 Cíl práce a metodika **13**

2.1 Cíl práce 13

2.2 Metodika 13

3 Teoretická východiska **14**

3.1 Životní cyklus vývoje softwaru 14

3.1.1 Plánování a analýza 14

3.1.2 Návrh 15

3.1.3 Vývoj a implementace 15

3.1.4 Testování 15

3.1.5 Údržba 16

3.2 Modely životního cyklu vývoje softwaru 16

3.2.1 Vodopádový model 16

3.2.2 V-model 17

3.2.3 Spirálový model 17

3.3 Agilní metodika řízení projektu 18

3.3.1 SCRUM 18

3.3.2 DSDM 19

3.4 Testování softwaru 20

3.4.1 Cíle testování 20

3.5 Životní cyklus testování softwaru 20

3.5.1 Analýza požadavků 21

3.5.2 Plán testování 21

3.5.3 Příprava testovacích scénářů 21

3.5.4 Příprava testovacího prostředí 22

3.5.5 Exekuce testů 22

3.5.6 Uzavření testů, zhodnocení stavu 23

3.6 Typy testů 23

3.6.1 Front-end a Back-end testy 23

3.6.2 Testy podle úrovně rozlišení 24

3.6.3 Testy podle účelu testování 25

3.6.4 Manuální testování 27

3.6.5 Automatizované testování 28

4 Vlastní práce **29**

4.1 Popis testované webové aplikace 29

4.2 Základní popis využitého nástroje pro tvorbu a exekuci manuálních testů 29

4.3 Základní popis využitého nástroje pro tvorbu automatizovaných testů 30
4.4 Specifikace případů užití 30
4.4.1 Průchod částí aplikace – Design v projektu 31
4.4.2 Vytvoření nové Story 32
4.4.3 Upravení Story 34
4.4.4 Odstranění Story 35
4.5 Tvorba manuálních testů 37
4.5.1 Manuální test Průchod částí aplikace – Design v projektu 37
4.5.2 Manuální test Vytvoření nové Story 38
4.5.3 Manuální test Upravení Story 39
4.5.4 Manuální test Odstranění Story 40
4.6 Exekuce manuálních testů 41
4.6.1 Exekuce testu Průchod částí aplikace – Design v projektu 42
4.6.2 Exekuce testu Vytvoření nové Story 44
4.6.3 Exekuce testu Upravení Story 46
4.6.4 Exekuce testu Odstranění Story 48
4.7 Tvorba automatizovaných testů 50
4.7.1 Automatizovaný test Průchod částí aplikace – Design v projektu 52
4.7.2 Automatizovaný test Vytvoření nové Story 54
4.7.3 Automatizovaný test Upravení Story 56
4.7.4 Automatizovaný test Odstranění story 58
4.8 Exekuce automatizovaných testů 60
4.8.1 Exekuce automatizovaného testu Průchod částí aplikace – Design v projektu 60
4.8.2 Exekuce automatizovaného testu Vytvoření nové Story 61
4.8.3 Exekuce automatizovaného testu Upravení Story 62
4.8.4 Exekuce automatizovaného testu Odstranění Story 62

5 Výsledky a diskuse 64
5.1 Výsledky manuálních testů 64
5.1.1 Výsledky testu Průchod částí aplikace – Design v projektu 65
5.1.2 Výsledky testu Vytvoření nové Story 65
5.1.3 Výsledky testu Upravení Story 65
5.1.4 Výsledky testu Odstranění Story 65
5.2 Výsledky automatizovaných testů 66
5.2.1 Výsledky aut. testu Průchod částí aplikace – Design v projektu 66
5.2.2 Výsledky aut. testu Vytvoření nové Story 67
5.2.3 Výsledky aut. testu Upravení Story 67
5.2.4 Výsledky aut. testu Odstranění Story 68
5.3 Vyhodnocení naměřených hodnot 68
5.4 Doporučení 69

5.5 Diskuse 70
5.5.1 Problém s celkovým E2E testem a vázaným ID v testované webové aplikaci 70
5.5.2 Nevýhody využití absolutní Xpath 72
5.5.3 Nevyužití žádné konkrétní sofistikované metody pro analýzu vhodnosti 72

6 Závěr 73

7 Seznam použitých zdrojů 74

8 Přílohy 76

Seznam obrázků

Obrázek 1: Epic v Designu 32
Obrázek 2: Formulář na vytvoření Story 34
Obrázek 3: Formulář na upravení Story 35
Obrázek 4: Odstranění Story 37
Obrázek 5: Manuální scénář Průchod částí aplikace v nástroji Juno (vlastní zpracování autora) 38
Obrázek 6: Manuální scénář Vytvoření nové Story v nástroji Juno (vlastní zpracování autora) 39
Obrázek 7: Manuální scénář Upravení Story v nástroji Juno (vlastní zpracování autora) 40
Obrázek 8: Manuální scénář Odstranění Story v nástroji Juno (vlastní zpracování autora) 41
Obrázek 9: Přihlášení do webové aplikace Juno (vlastní zpracování autora) 42
Obrázek 10: Oblast Story (vlastní zpracování autora) 43
Obrázek 11:Ukázka záznamu v oblasti History (vlastní zpracování autora) 43
Obrázek 12: Odhlášení ze systému Juno (vlastní zpracování autora) 43
Obrázek 13: Vyhodnocení test Průchod částí aplikace – Design v projektu (vlastní zpracování autora) 44
Obrázek 14: Formulář na vytvoření Story (vlastní zpracování autora) 45
Obrázek 15: Tabulka Storek – Vytvořená Story (vlastní zpracování autora) 45
Obrázek 16: Vyhodnocení testu Vytvoření nové Story (vlastní zpracování autora) 46
Obrázek 17: Formulář na upravení Story (vlastní zpracování autora) 47
Obrázek 18: Tabulka Storek – Upravená Story (vlastní zpracování autora) 47
Obrázek 19: Vyhodnocení testu Upravení Story (vlastní zpracování autora) 48
Obrázek 20: Vyskakovací okno na Odstranění Story (vlastní zpracování autora) 49
Obrázek 21: Tabulka Storek – Odstraněná Story (vlastní zpracování autora) 49
Obrázek 22: Načtení WebDriveru (vlastní zpracování autora) 51
Obrázek 23: Načtení webové aplikace (vlastní zpracování autora) 51
Obrázek 24: Vývojový diagram testu na průchod částí aplikace – Design v projektu (vlastní zpracování autora) 53
Obrázek 25: Vývojový diagram testu Vytvoření nové Story (vlastní zpracování autora) 55
Obrázek 26: Vývojový diagram testu Upravení Story (vlastní zpracování autora) 57
Obrázek 27: Vývojový diagram testu Odstranění Story (vlastní zpracování autora) 59
Obrázek 28: Vyhodnocení automatizovaného testu Průchod částí aplikace – Design v projektu (vlastní zpracování autora) 60
Obrázek 29: Vyhodnocení automatizovaného testu Vytvoření nové Story (vlastní zpracování autora) 61

Obrázek 30: Vyhodnocení automatizovaného testu Upravení Story (vlastní zpracování autora) 62
Obrázek 31: Vyhodnocení automatizovaného testu Odstranění Story (vlastní zpracování autora) 63

Seznam tabulek

Tabulka 1: Výsledná tabulka naměřených hodnot pro sadu manuálních testů (vlastní zpracování autora) 64
Tabulka 2: Naměřené hodnoty pro manuální test Průchod částí aplikace – Design v projektu (vlastní zpracování autora) 65
Tabulka 3: Naměřené hodnoty pro manuální test Vytvoření nové Story (vlastní zpracování autora) 65
Tabulka 4: Naměřené hodnoty pro manuální test Upravení Story (vlastní zpracování autora) 65
Tabulka 5: Naměřené hodnoty pro manuální test Odstranění Story (vlastní zpracování autora) 66
Tabulka 6: Výsledná tabulka naměřených hodnot pro sadu automatizovaných testů (vlastní zpracování autora) 66
Tabulka 7: Naměřené hodnoty pro aut. test Průchod částí aplikace – Design v projektu (vlastní zpracování autora) 67
Tabulka 8: Naměřené hodnoty pro aut. test Vytvoření nové Story (vlastní zpracování autora) 67
Tabulka 9: Naměřené hodnoty pro aut. test Upravení Story (vlastní zpracování autora) 68
Tabulka 10: Naměřené hodnoty pro aut. test Odstranění Story (vlastní zpracování autora) 68
Tabulka 11: Výsledná tabulka pro porovnání manuálních a automatizovaných testů (vlastní zpracování autora) 69

Seznam použitých zkratek

E2E = end to end (od začátku do konce)

Aut. = automatizovaný

SDLC = Software development life cycle (Životní cyklus vývoje softwaru)

DSDM = Dynamic Systems Development Method (Metoda vývoje dynamického systému)

STLC = Software Testing Life Cycle (Životní cyklus testování softwaru)

GUI = Graphic user interface (Grafické uživatelské rozhraní)

IT = informační technologie

1 Úvod

Životní cyklus vývoje a všechny jeho fáze jsou nedílnou součástí celého IT světa, ať už se jedná o vývoj, testování či ostatní fáze. Avšak jedna činnost nemůže fungovat bez činnosti druhé. V dnešní době si již snad ani nelze představit, nevyužívat různé typy softwaru. Využívají se napříč všemi odvětvími po celém světě, ať už se jedná o dopravní podniky, zdravotnictví, bankovnictví, účetnictví. V podstatě téměř v každém odvětví se již nachází software, který ulehčuje a pomáhá lidem s prací.

Jak již bylo řečeno, jedna činnost nemůže fungovat bez činnosti druhé – to znamená, že sice bez návrhu bychom neměli co vytvořit, bez vývoje by sice bylo co vytvořit, ale nikdy by se tak nestalo. V případě neexistence testování by sice byl vytvořen software, ale nikdy by neodpovídal zadání, byl by velmi poruchový a často nefunkční. Z tohoto důvodu je testování velmi důležitou činností, která zaručuje, že software funguje přesně tak, jak byl vymyšlen a je od něj očekáváno.

S tím, jak si člověk snaží usnadnit veškerou práci, tak si snaží usnadnit i práci v testování, a to tím, že na místo manuálního testování se začíná hojně využívat testování automatizované, které je prováděno pomocí stroje.

Bakalářská práce je zaměřena právě na problematiku manuálního a automatizovaného testování a analýzy vhodnosti využití těchto testů. Obsahem teoretické části je popis životního cyklu vývoje, detailní popis některých využívaných modelů, rozebrané téma agilní metodiky a typů metodik, které se v agilitě nacházejí. Teoretická část je zaměřena na problematiku manuálního testování, automatizovaného testování a cíle testování. Je v ní rozebrán životní cyklus testování a jednotlivé typy testů podle úrovně a účelu využití.

Praktická část se zabývá testováním části aplikace pomocí manuálních a automatizovaných testů. Nachází se zde i popis testované aplikace a využitých nástrojů pro tvorbu a exekuci vytvořených testovacích scénářů. Jsou zde vytvořeny případy užití pro jednotlivé scénáře načerpané ze znalosti a dokumentace aplikace, dále se zde nachází vývojové diagramy určující průběh testů. Poté je zde vytvořena detailní tvorba a exekuce manuálních a automatizovaných testů, z těchto činností jsou naměřeny výsledné hodnoty pro jejich časovou náročnost.

V závěru jsou výsledky zhodnoceny, je provedena analýza a doporučení vhodnosti využívaní manuálního a automatizovaného testování.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je zaměřena na problematiku testování softwaru v rámci životního cyklu projektu. Hlavním cílem je testování základních funkcionalit systému prostřednictvím manuálních a automatizovaných testů a jejich porovnání.

1) Seznámení se s problematikou testování.
2) Popsání životního cyklu vývoje systému v rámci projektového řízení.
3) Popsání specifik automatizovaných a manuálních testů.
4) Analýza vhodnosti použití automatizovaných testů pro určitý proces v rámci projektu.
5) Zhodnocení, doporučení a navržení dalšího postupu.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. V teoretické části jsou popsány specifika manuálních a automatizovaných testů. Získané vědomosti z teoretické části jsou využity v praktické části pro analýzu vhodnosti použití automatizovaných testů pro zvolený proces.

3 Teoretická východiska

3.1 Životní cyklus vývoje softwaru

Životní cyklus vývoje softwaru neboli SDLC je systematický proces pro vytváření softwaru, který zajišťuje kvalitu, bezchybnost a korektnost vytvořeného softwaru. Cílem procesu SDLC je vytvořit vysoce kvalitní software, který splňuje očekávání zákazníků. Vývoj systému by měl být dokončen v předem definovaném termínu v rámci zvolených omezených nákladů. SDLC sestává z detailního plánu, který obsahuje informace, jak naplánovat, vytvořit a udržovat konkrétní software. SDLC se dělí na několik fází, každá fáze životního cyklu má své vlastní vstupy, proces a výstupy, které vstupují do další následující fáze. SDLC většinou obsahuje 5 hlavních fází po sobě jdoucích procesů: [1]

1. Plánování a analýza
2. Návrh
3. Vývoj a implementace
4. Testování
5. Údržba

3.1.1 Plánování a analýza

Během této fáze jsou během schůzek všechny potřebné informace získávány od zákazníka za účelem vývoje produktu podle jeho představ. Všechny nejasnosti musí být do detailu vyřešeny přesně v této fázi. Dále analytik a manažer často sjednává schůzky se zákazníkem za účelem získání informací. Zjišťuje, co zákazník chce vytvořit, kdo bude koncovým uživatelem a jaký je účel produktu. Před samotným vytvořením produktu je velmi důležité jeho základní porozumění a znalost produktu. [2]

Tým poté definuje nový software dle požadavků od zákazníka, určí potřebné náklady a zdroje. Také podrobně popíše všechny související rizika a poskytne plány na minimalizování těchto rizik. V této fázi je vytvořen dokument Specifikace softwarových požadavků. [3]

Po analýze vyplývající z fáze plánování budou k dispozici 3 možnosti: [4]

1. vytvořit nový systém
2. vylepšit stávající systém
3. ponechat stejný systém takový, jaký je

3.1.2 Návrh

Tato fáze SDLC se zabývá přeměnou softwarové specifikace na návrhový plán nazvaný „Návrhová specifikace“. Po jeho vytvoření obě zúčastněné strany tento plán prostudují a nabídnou vlastní zpětnou vazbu a návrhy na zlepšení nebo opravení nejasností vyplynulých z požadavků. [3]

Základní návrh může být vytvořen pomocí tužky a papíru, kdy vznikne návrh sloužící k základnímu náhledu, jak bude systém vypadat a fungovat. Po další zpětné vazbě od zákazníka bude již vytvořen podrobný a rozšířený návrh, který bude splňovat všechny technické a funkční požadavky. [4]

Pokud zde bude systém špatně navrhnut, tato chyba téměř jistě povede v lepším případě k překročení nákladů, v nejhorším případě pak k úplnému zhroucení projektu. [3]

3.1.3 Vývoj a implementace

Jakmile skončí fáze návrhu systému, přijde na řadu další fáze a tou je vývoj. V této fázi přijdou na řadu vývojáři, kteří začnou vyvíjet systém pomocí zvoleného programovacího jazyka. Ve fázi vývoje bývají úkoly děleny na jednotlivé menší části pro jejich přesnější a jednodušší zpracování. Tato fáze bývá nejdelší ze všech fází SDLC. Vývojář musí přesně dodržovat zadání od zákazníka. Vývojář postupuje a vytváří kód podle dokumentů vytvořených v předešlých fázích. Tato fáze je tím jednodušší, čím pečlivěji a detailněji byly provedeny předešlé fáze. [1]

3.1.4 Testování

Po dokončení fáze vývoje a úspěšné implementaci softwaru neboli jeho nasazení na testovací prostředí, následuje fáze testování veškerých funkčností vytvořeného systému. Testovací tým postupně začne testovat funkčnost celého systému, oproti zadání čili specifikacím od zákazníka. To se provádí z důvodu ověření, že celá aplikace funguje přesně podle požadavků od zákazníka.

V průběhu této fáze testovací tým nachází chyby – takzvané defekty, které zaeviduje a přiřadí vývojáři, který defekty opraví pomocí reportovacích nástrojů. Následně nasadí tuto opravu na testovací prostředí, změní její stav a odešle zpět testovacímu týmu k provedení nových testů. Testy následně prokážou, zda se chybu podařilo opravit. Tento proces se opakuje do chvíle, než software neprojevuje žádné chyby, je stabilní a obsahuje všechny zadané funkčnosti z požadavků. [1]

3.1.5 Údržba

Tato fáze obnáší pravidelnou údržbu a kontrolu systému, snaží se předcházet možným výpadkům systému. Může se jednat o výměnu starého hardwaru, monitorování výkonu systému, zlepšování systému pomocí drobných novějších aktualizací z důvodu kompatibility pro nové komponenty, aby se zajistila funkčnost a kompatibilita systému na novějších platformách, a aby dokázal odolat současným bezpečnostním hrozbám. [1]

3.2 Modely životního cyklu vývoje softwaru

Model životního cyklu softwaru je návrh a znázornění jak postupovat v životním cyklu vývoje softwaru. Modely SDLC mohou mít odlišný přístup, ale základní fáze a aktivita zůstávají podobné pro všechny modely. Nikde není známé ani ověřené, který z modelů je nejlepší, každý model má své výhody a nevýhody. V této kapitole budou rozebrány 3 známé modely: [6]

1. Vodopádový model
2. V-model, vychází z vodopádového
3. Spirálový model

3.2.1 Vodopádový model

Vodopádový model je nejstarší a výchozí model pro následující novodobější modely. Model signalizuje vodu tekoucí po vodopádu ve tvaru schodů, kdy voda teče postupně shora dolů po jednotlivých fázích SDLC, až k výslednému produktu. Vodopádový model je sekvenční, tzn. že každá další fáze může začít nejdříve po skončení té předešlé. Vývojový tým na konci každé fáze zkontroluje, zdali je možné začít s novou fází. V případě, že se produkt nenachází ve stavu, kdy je možné přejít k další fázi, zůstává ve stejné fázi, dokud není zcela dokončena. [7]

Tento model má výhodu ve své jednoduchosti, která je důležitá pro jeho pochopení a provedení, dále pro odhalení a odstranění chyby, které je možní zjistit již v počátečních fázích SDLC. Tím se předchází zbytečně vysokým nákladům v pozdější fázi. [6]

Nevýhodami tohoto modelu je časová náročnost z důvodu sekvenční posloupnosti v modelu. Z tohoto důvodu nelze model použít v krátkodobých projektech. Model není vhodné využít pro projekty s nejasným zadáním, u kterých probíhají změnové požadavky a úpravy zadání v průběhu užívání, jelikož není možné vytvořit krok zpět. [2]

3.2.2 V-model

V-model je znám také jako model verifikace a validace a jedná se o rozšíření Vodopádového modelu. Model funguje sekvenčně, což znamená, že každá fáze musí být dokončena před spuštěním následující fáze. Rozdílem v těchto modelech jsou výstupy v jednotlivých výstupech životního cyklu. Výstup z každé fáze životního cyklu prochází ověřováním a testováním. Díky tomu lze předejít chybám v rané fázi a nepřenášet je do dalších fází, čímž se snižují možné budoucí škody a náklady na opravu. Díky této funkčnosti paralelního programování a testování se V-model často využívá. V-model ve velké většině využívá tyto 4 typy testů pro dané fáze testování: [5], [2]

1. Akceptační
2. Systémové
3. Integrační testy
4. Unit testy (Jednotkové)

3.2.3 Spirálový model

Spirálový model si zakládá na iterativním přístupu, a především zavádí opakovanou analýzu všech rizik a testování při přechodu mezi fázemi. Spirálový model funguje na principu navazování nových částí na již vytvořenou a otestovanou část. Po dosažení hrubé specifikace požadavků se provádí první část vývoje, v pozdějších fázích se tato specifikace postupně upravuje a upřesňuje i komunikací se zákazníkem. [6]

Po dokončení každé fáze dochází k testování, vyhodnocení a předání dílčích výsledků jako vstupů do nové fáze. Produkt je tedy testován pravidelně, a to již od raných fází. Díky pravidelnosti a častému opakování testů je velmi vhodné použít na toto testování automatizované testy. Tyto testy se pouze budou udržovat a upravovat podle aktuálně vydané verze. Tím, že se aplikace testuje pravidelně a po částech, dochází k včasnému odhalení chyb. Spirálový model bývá charakteristický svými 4 kroky: [6]

1. Určení cíle, omezení a alternativ
2. Vyhodnocení alternativ a řešení rizik
3. Vývoj a verifikace další úrovně produktu
4. Plánování další fáze

Nevýhodou tohoto modelu jsou možné vysoké náklady, jelikož je zapotřebí velké množství iterací testů a časová náročnost. Spirálový model je vhodné používat pouze pro velké projekty. [2]

3.3 Agilní metodika řízení projektu

Za vznikem agilní metodiky stojí nedostatky tradičních přímých modelů projektového řízení jako jsou zpracování celého projektu v dlouhém cyklu vývoje projektu, časové zpoždění mezi jednotlivými fázemi SDLC a vysoká rizikovost pozdního odhalení chyb nebo změn. Agilita je postavena na týmové spolupráci, ochotě a otevřenosti změnám a velmi časté otevřené komunikaci.

Hlavní cíle agility lze shrnout v následujících bodech: [8]

1. Průběžná inovace, která předpokládá, že se výsledek projektu shoduje se zadáním zákazníka během vlivů tržního prostředí.
2. Průběžné přizpůsobování, projektu s cílem splnit i budoucí potřeby zákazníka.
3. Zrychlené zavedení produktu na trh, čímž se zlepší rychlost návratnosti investic.
4. Vysoká přizpůsobivost lidí a procesů, z důvodu lepší schopnosti reakce na změnu požadavků projektu.
5. Vyšší spolehlivost výsledků projektu, zlepšující ziskovost zadavatele projektu a jeho podnikatelský úspěch. [8]

Agilita se využívá na velkých složitých projektech komplexních systémů. Na začátku projektů není možné detailně definovat všechny požadavky, které se navíc v průběhu projektu častokrát mění a upravují. Požadavky se postupně během vývoje a jednotlivých iterací zdokonalují. Často se požadavky získávají z vytvořených prototypů a dále zpětnou vazbou se zákazníkem. [9]

Jedny z nejvyužívanějších metodik v rámci agility je SCRUM a DSDM.

3.3.1 SCRUM

Scrum jako jedna z nejrozšířenějších agilních metodik se zaměřuje primárně na organizaci vývojového týmu. Největší výhodou Scrumu je dostatečná předvídatelnost a odhadnutelnost náročnosti a množství práce. Scrum se především využívá u velkých stabilních firem. Struktura bývá dělena většinou na 4-10 lidí, kteří by měli být na stejném místě. [10]

Scrum používá tzv. Backlogy, což jsou v podstatě seznamy úkolů, ve kterých se postupně hromadění další a další úkoly, a postupně se z tohoto Backlogu úkoly zpracovávají. Backlog mívá 2 verze:

1. Produkční Backlog obsahuje seznam úkolů, které se budou provádět v budoucnu.

2. Sprint Backlog obsahuje úkoly, které se mají vykonávat v tomto časovém období.[10]

Ve Scrumu se často prostřednictvím denních konzultací zvanými „Stand-Up“ nebo pomocí třiceti denních iterací označovaných jako „sprint“. Sprint je časové období, ve kterém by měl být realizován Sprint Backlog. Na konci každého sprintu je dodána vybraná skupina užitných vlastností. [11]

Scrum má definované čtyři hlavní fáze životního cyklu:

1. Plánování, jedná se o fázi specifikací prvních požadavků, plánu dodávky a cílů.
2. Vynášení, fáze odkládání nefunkčních požadavků do Backlogu.
3. Vývoj, v této fázi se realizují úkoly s nejvyšší prioritou ze Sprint Backlogu, nejčastěji ve 30denní lhůtě(sprintu). Každý člen se věnuje nějakému dílčímu úkolu, probíhají zde denní schůzky na monitorování stavu. Na konci sprintu dochází k předvedení a zhodnocení výsledků.
4. Dodávka, v této fázi dochází k předávání produktu uživatelům. [11]

3.3.2 DSDM

Agilní metodika DSDM vychází z vodopádového řízení projektu ve spojení s agilitou. Jedná se o jednu z prvních agilních metodik. Tato metodika si zakládá na včasném dodání výsledků. Poté hledí na aspekty jako jsou dodržení rozpočtu a výhodnost z obchodního hlediska. [12]

Rozdíl v této metodice oproti ostatním z agilních metodik je v tom, že popisuje celý průběh životního cyklu softwaru. V této metodice jsou role v týmu rozděleny tak, aby každý byl zodpovědný za určitou část projektu. Nejedná se zde tedy o týmovou zodpovědnost jako ve Scrumu, ale o zodpovědnost jednotlivce, je zde velmi důležitá komunikace a vzájemný respekt. [12]

Metodika má definováno těchto hlavních 9 principů:

1. Aktivní kooperace s uživatelem
2. Rozhodovací tým
3. Časté dodávky produktů
4. Silná podpora podnikových cílů
5. Iterativní vývoj

6. Dynamika během vývoje
7. Hrubě stanovené požadavky
8. Testovaní během celého SDLC
9. Spolupráce mezi členy týmu [11]

3.4 Testování softwaru

Testování softwaru probíhá z důvodu, že nikdy není žádný software dokonalý, nikdy projekt nepostupuje přesně podle některého z modelů životního cyklu vývoje. Testováním se rozumí odhalování chyb a možných problémů v již vyvinutém softwaru, čímž se výrazně ovlivňuje kvalitu výsledného softwaru a dokáže zamezit možným mimořádným nákladům. [7]

3.4.1 Cíle testování

Cílem testování je objevovat chyby a možné problémy softwaru. Nejedná se o činnost, která má ukázat, že v softwaru chyby nejsou. Testováním se snažíme nalézt co největší množství chyb a minimalizovat tak jejich počet v softwaru a tím udělat software uživatelsky přístupný, téměř nikdy ovšem nelze docílit bezchybného stavu. Testování se tedy provádí za účelem objevení co největšího počtu existujících chyb. Zjednodušeně je možno uvést, že testování funguje jako porovnávání očekávaného stavu se stavem skutečným pro jednotlivé funkcionality softwaru. [13]

3.5 Životní cyklus testování softwaru

Životní cyklus testování softwaru, dále jen zkráceně STLC je posloupnost činností, které se provádějí v průběhu procesu testování. Testování softwaru není jen jednoduchá / izolovaná aktivita. Testování se skládá z řady navazujících činností prováděných metodicky za účelem zlepšení kvality testovaného softwaru. Součástí STLC jsou následující fáze: [14]

1. Analýza požadavků
2. Plán testování
3. Příprava testovacích scénářů
4. Příprava testovacího prostředí
5. Exekuce testů
6. Uzavření testů, zhodnocení stavu

3.5.1 Analýza požadavků

Během první fáze STLC testovací tým studuje požadavky z hlediska testování za účelem identifikace, co vše bude testováno a jak. Tým často komunikuje se zúčastněnými stranami jako klientem, obchodním analytikem, technickými vedoucími, systémovými architekty, aby do detailu a podrobně porozuměl všem požadavkům a kritériím. Požadavky se dělí na funkční (jak software funguje, co musí dělat) a nefunkční (zabezpečení a výkon systému). [14]

Aktivity spojené s touto fází:

1. Identifikace typu testů
2. Zanalyzování priorit a zaměření testů
3. Příprava matice požadavků
4. Identifikace testovací prostředí
5. Analýza použitelnosti automatizace

3.5.2 Plán testování

V této fázi vedoucí týmu určí pracnost a náklady a podle toho vytvoří testovací strategii. Vedoucí týmu vychází z požadavků z předešlé fáze, určí zde plánované testy, cíl testování, předpoklady. Jedná se o nejdůležitější fázi v STLC. Výstupem z této fáze je vytvořený testovací plán. [14]

Aktivity spojené s touto fází:

1. Příprava testovací strategie pro určité typy testů
2. Zvolení testovacího nástroje
3. Odhad pracnosti
4. Plánování zdrojů, určování rolí a odpovědností

3.5.3 Příprava testovacích scénářů

Tato fáze zahrnuje vytvoření a ověření testovacích případů, popřípadě testovacích scriptů, zdali je možnost využít automatizaci. Také jsou zde analyzována a připravena veškerá testovací data. Ve scénářích je detailně zahrnut postup, očekávaný výsledek, předpoklady a popis. [14]

Aktivity spojené s touto fází:

1. Vytvoření testovacích scénářů / aut. scriptů, pokud je to možné
2. Ověření testů a scriptů

3. Příprava testovacích dat (pokud je připravené prostředí)

3.5.4 Příprava testovacího prostředí

Výstupem z této fáze je připravené testovací prostředí pro spuštění exekuce testů. Testovací prostředí rozhoduje o softwarových a hardwarových vlastnostech a komponentách, na kterých poběží software, který je testován. Příprava testovacího prostředí je jedním z kritických momentů celého testovacího procesu, může být prováděno souběžně s fází přípravy testovacích scénářů, mimo přípravu testovacích dat.

O přípravu testovacího prostředí se z většiny případů stará vývojový tým. Testovací tým pouze po přípravě testovacího prostředí provede Smoke testy, pro otestování funkčnosti nového prostředí. [14]

Aktivity spojené s touto fází:

1. Příprava požadovaného softwaru a hardwaru
2. Nastavení požadované architektury
3. Nastavení prostředí a testovacích dat
4. Spuštění Smoke testů

3.5.5 Exekuce testů

Během této fáze dochází ke spouštění exekucí testů. O exekuci se starají testeři, kteří ji provádějí na základě testovacího plánu, testovacích scénářů a testovacích dat vytvořených v předešlých fázích. Objevené chyby jsou hlášeny a předávány zpět vývojovému týmu k opravě. Po opravě a nasazení opravy na prostředí bude provedeno opětovné testování tzv. retest. Pokud tester nenajde během exekuce testovacího scénáře chybu, označí test stavem „Passed“ a pokračuje dále v testování. [14]

Aktivity spojené s touto fází:

1. Exekuce testů stanovené testovacím plánem
2. Dokumentace výsledků (Report)
3. Evidence chyb do požadavků
4. Retestování chyb
5. Sledování chyb do jejich uzavření

3.5.6 Uzavření testů, zhodnocení stavu

V aplikaci se po provedení exekuce nacházejí chyby, které se postupně opravují vývojáři. Po ukončení fáze testování může docházet ke schůzkám mezi testovacím týmem a vývojáři, za účelem vyčištění veškerých nalezených chyb. Cílem této fáze je odstranit problémová místa procesu pro následující testovací cykly. [14]

Aktivity spojené s touto fází:

1. Zhodnocení výsledků na základě požadavků, času, náročnosti, nákladů
2. Příprava výsledného reportu
3. Příprava výsledků pro zákazníka
4. Zhodnocení a opravy nalezených chyb

3.6 Typy testů

Testování jako takové je velmi obecný pojem, je známo, že každý software je jiný a nelze testovat stejným způsobem. Z tohoto důvodu jsou testy děleny do několika skupin podle několika hledisek. Nejdůležitějšími hledisky jsou dle využívaných služeb a typu pohledu na testy. Dále se jedná o rozřazení podle úrovně testování a podle účelu testování. [15]

3.6.1 Front-end a Back-end testy

Tyto typy testů se liší místem a částí aplikace, kde jsou prováděny. „Front-end“ testy se provádí na uživatelském rozhrání. „Back-end“ testy se provádí na aplikačním rozhraní, provolávání služeb a databáze.[16]

Front-end

Front-end testování je typ testování, který kontroluje prezentační vrstvu architektury aplikace. V laickém rčení se jedná o činnost sloužící k otestování GUI, to, co je vidět na obrazovce. U webové aplikace by front-end testování zahrnovalo kontrolu funkčností a prvků jako jsou formuláře, grafy, nabídky, zprávy a jiné. Front-end testování je termín, který zahrnuje různé testovací strategie. K provedení tohoto typu testování tester potřebuje velmi odbornou znalost požadavků a dokumentace. [16]

Back-end

Back-end testování je typ testování, který kontroluje aplikační a databázovou vrstvu architektury aplikace. Back-end testování slouží ke kontrole služeb na straně serveru nebo databáze. Například data zadaná na front-endu aplikace budou zkontrolována back-end testy, zdali se uložila v databázi. [16]

Při back-end testování není nutné používat GUI, jelikož není v testech zahrnuto. Data mohou být přímo předaná pomocí zavolání služby obsahující zadané parametry, aby se odpověď tzv. „Response“ získala v přehledném a využitelném formátu. Příklady výsledku volání mohou být ve formátu XML nebo JSON. [16]

3.6.2 Testy podle úrovně rozlišení

Tento typ testů vychází z V-modelu z kapitoly 3.2.2, kde každé příslušné fázi odpovídá jeden typ testů. Tyto fáze jsou základem, na kterém jsou testy vytvořeny. Testy pod úrovně rozlišení vychází ze zaměření testů na konkrétní chyby. Typy testů podle úrovně rozlišení jsou: [15]

1. Unit (jednotkové)
2. Integrační
3. Systémové
4. Akceptační

Unit testy (jednotkové)

Unit testy přicházejí v první fázi testování, jedná se o testování samostatných částích aplikace. Ve většině případů se jedná o testy prováděné vývojáři z důvodu, že zde dochází k testování jednotlivých metod a tříd programu.

Unit testy se málokdy provádí na již zaběhlých projektech, jelikož by muselo dojít k „refactoringu“ kódu nebo jeho úpravám. Z toho důvodu se tyto testy provádí ve fázi návrhu aplikace. [17]

Integrační testy

Integrační testy již nejsou připravovány vývojovým týmem, ale v tomto typu testování přichází na řadu testovací tým. Integrační testy jsou nadstavbou pro testy jednotkové, integrační testy spočívají v testování komunikace mezi jednotlivými

komponentami v aplikaci. Nejde zde ovšem pouze o tuto komunikaci, ale i o integraci s operačním systémem, rozhraními a hardwarem.

Tento typ testů není nutné provádět vždy a na každém projektu, ale za podmínky, že následující typy testů budou provedeny s maximálním úsilím. Avšak čím dříme chybu objevíme, tím lépe, proto se testy zavádějí. [17]

Systémové testy

Systémové testy bývají označovány jako SIT. Jedná se spojení testů systémových a integračních. Tyto testy se již zaměřují na aplikaci jako celek. Přicházejí v pozdější fázi vývoje a ověřují funkčnost aplikace z pohledu zákazníka. Testování probíhá podle připravených testovacích scénářů, které obsahují možné scénáře, které mohou v aplikaci nastat. Nalezené chyby se zaznamenávají a předávají na vývojový tým k opravě. Probíhají zde funkční i nefunkční testy. Tento typ testů se provádí přímo před předáním aplikace zákazníkovi. Tento typ testů je komplexní a jeden z nejdůležitějších.

Akceptační testy

Akceptační neboli UAT testy bývají zpravidla prováděny na straně zákazníka. Po protestování a odladění aplikace předešlými typy testů dojde k předávce zákazníkovi. Ten buď sám nebo pomocí svého týmu testerů otestuje aplikaci. Testy odpovídají testovacím scénářům, které byly připraveny na schůzkách mezi zákazníkem a dodavatelem. Nalezené chybu vůči specifikaci jsou předávány na vývojový tým. Tyto opravy bývají prováděny v co nejkratší době. V této fázi testování se očekává pouze malé množství chyb, které bývají rychle opraveny.

3.6.3 Testy podle účelu testování

Testy v této kategorii bývají děleny podle účelu, na co konkrétně se test zaměřuje. Testy často bývají děleny na E2E testy, funkční a nefunkční testy a jiné. V této kapitole budou detailněji rozebrány následující typy: [15]

1. Regresní testy
2. Smoke testy
3. Funkční testy
4. Nefunkční testy
5. E2E testy

Regresní testy

Tento typ testů se zabývá problematikou zásahů do aplikace a kontrolou správné funkčnosti všech částí, i těch které neměly být zásahem ovlivněny. Nejčastějším zásahem bývá oprava chyby nebo přidání nové funkčnosti. [13]

Testy bývají velmi často automatizovány, a to z důvodu jejich častého opakování, test má vždy stejný průběh a výsledek. Tím pádem, pokud se projeví nějaká chyba na místě, kde by neměla automatizovaný test ji ihned odhalí. [18]

Smoke testy

„Smoke testy“ je typ testů, které jsou prováděny za účelem otestování stability systému, hlavních částí a jejich funkčnosti. Cílem není testování za účelem nalézat chyby jako u regresních testů, ale ověření připravenosti systému pro jiné typy testů. Jde o rychlé testy, obvykle obsahující jednoduchý průchod skrz aplikaci, které dokážou ověřit části aplikace. [5]

Testy bývají často automatizovány z důvodu častého opakování, jelikož se provádějí po nasazení nové verze aplikace, aby se zjistilo, zdali je prostředí stabilní. Většinou po nich přichází na řadu testy systémové. [5]

Tyto testy mívá na starost testovací tým. V některých případech a projektech jsou případy, že tyto testy provádí pracovník odpovědný za správu testovacích prostředí. [19]

Funkční testy

Z názvu funkční testy je již zřejmé, že tyto testy mají na výslednou bezporuchovost softwaru velký vliv. Proto je jim věnována velká pozornost během celého testovacího cyklu softwaru. Testy se využívají na jakkoliv velkém a složitém projektu. Do této kategorie se nejčastěji řadí integrační, systémové a akceptační testy. Tyto testy mají za úkol odhalovat velké množství chyb. Jejich cílem je tedy otestovat aktuální stav funkčnosti aplikace oproti funkčním požadavkům. [17]

Nefunkční testy

Nefunkční testy se oproti testům funkčním liší v jejich zaměření. Zatím, co funkční testy jsou zaměřeny na funkčnost aplikace, nefunkční testy spočívají v testování všech

vlastností aplikace, které nejsou jejími funkcemi, ale jsou velmi podstatné pro její správné fungování.

Mezi nefunkční testy patří například výkonové nebo jinak řečeno zátěžové testy (Performance), které mají za úkol otestovat, že aplikace i pod zátěží, čímž je myšleno více aktivních uživatelů využívajících aplikaci současně bude pracovat s dostatečnou rychlostí odezvy a vydrží funkční, „nespadne“. Dále se testuje například jak aplikace bude snášet nárůst zátěže. V neposlední řadě se testuje aplikace vůči hardwaru, například zdali zbytečně nezatěžuje paměť a procesor. [17]

E2E testy

E2E neboli end-to-end testy se využívají k testování, zda „průchod“ aplikací funguje od začátku do konce přesně podle požadavků. Testy se provádějí za účelem identifikace závislostí systému a zajistit, aby mezi různými částmi systému a systémy okolními, které souvisejí s daným průběhem testu a jsou potřeba k dokončení procesu byly předávány správné informace. Testování E2E tedy zahrnuje otestovaní toho, aby všechny integrované komponenty aplikace fungovaly podle očekávání. Průchodem tohoto typu testu jsou tedy také funkce jako komunikace s databází, sítí, hardwarem a dalšími aplikacemi. [19]

3.6.4 Manuální testování

Manuální testování bývá nejčastější způsob využívaný pro testování softwaru, a to především z důvodu malých nákladů. Manuální testy jsou prováděny přímo člověkem – testerem. Provádějí se často, pokud je potřeba pro testování využít lidský faktor pro rozhodnutí. Testovány jsou příslušné softwary podle předem vytvořených testovacích scénářů, které jsou navrženy podle specifických požadavků a dokumentací daného softwaru. Aby bylo možné příslušné testy provádět musí být funkční prostředí testovací aplikace a testeři musí znát testovanou aplikaci nebo musí mít detailně vypracované testovací scénáře krok po kroku, kde každý krok obsahuje očekávaný výsledek.

Manuální testy se využívají především u aplikací, které procházejí vývojem nebo po nasazení nových verzí aplikace, změnových požadavků, přidání nových funkčností nebo opravě chyb. Pokud se jedná o složitější testy, časová náročnost se zvyšuje a pokud se testy častokrát opakují, začíná se využívat testů automatizovaných. [20]

3.6.5 Automatizované testování

Automatizované testy bývají využívány, pokud se testy manuální začínají stávat časově náročnými a často opakovanými. Automatizované testy provádí sám software za pomocí scriptů, který byly vytvořeny pomocí nějakého testovacího nástroje testerem. Tyto scripty mají za úkol otestovat funkčnost aplikace bez pomoci a zásahu člověka. Automatizované testy je vhodné využívat až po dokončení vývojového cyklu, a to z důvodu, že pokud by nebyl cyklus dokončen a aplikace se stále měnila, musely by se scripty pokaždé upravovat nebo vytvářet, což by mohlo být finančně i časově náročné.

Automatizované testování se začíná vyplácet až tehdy, jeli počet provádění testů velký a častý. Největší výhodou automatizovaného testování je rychlost provedení a možnost kdykoliv test spustit znovu kolikrát je potřeba. Ovšem jejich nevýhodou je časová náročnost na jejich tvorbu. [20]

4 Vlastní práce

Vlastní práce je zaměřena na vytvoření E2E manuálních a automatizovaných testů, které budou prováděny na webové aplikaci Juno. Manuální testy, budou vytvořeny a provedeny v produkční verzi aplikace Juno. Automatizované testy budou vytvořeny v aplikace Eclipse pomocí frameworku určeného pro automatizaci webových aplikací, a to Selenium Webdriver za využití programovacího jazyka Java. Závěrem vlastní práce bude zhodnocení využitelnosti automatizovaných testů.

4.1 Popis testované webové aplikace

Webová aplikace testovaná v rámci bakalářské práce se nazývá Juno. Jedná se o nový testovací nástroj, který je možno využívat jak pro tvorbu, tak pro exekuci jednotlivých testů. Uživatelé zde mohou plánovat celou testovací analýzu pro konkrétní projekty. Mohou si zde vytvářet projekty, které mohou sloužit pouze pro exekuci testů nebo také komplexní projekty, které obsahují celou test analýzu, od požadavků až po reportovaní chyb a následné opětovné spuštění testů. Jednotlivé testy mohou být spojeny pomocí test plánů, které následně mohou být spojeny pomocí kampaní, což je nadstavba nad projekty.

V aplikaci mají uživatelé různé možnosti v jejím ovládání, jsou limitovány systémovými a projektovými oprávněními.

4.2 Základní popis využitého nástroje pro tvorbu a exekuci manuálních testů

Pro vytvoření a následnou exekuci manuálních testů byl využit webový testovací nástroj Juno, dostupný na adrese https://juno.one/. Tento nástroj byl zvolen z dřívějších zkušeností autora s tímto testovacím nástrojem. Nástroj byl také zvolen z důvodu možné exekuce vytvořených testů právě v tomto nástroji.

Před samotným vytvořením testů je potřeba vytvořit projekt, ve kterém budou tyto testy uloženy. Po vytvoření projektu je možné již vytvořit samotné testy. Jelikož ale autor bude testy v tomto nástroji provádět, připraví k tomu nejprve potřebné oblasti. Prvními z nich je testovací plán a testovací sprint. Testovací plán a sprint slouží pro určení období a rozdělení testů, které mají být v tomto období provedeny. Po jejich vytvoření přichází na řadu tvoření testovacích scénářů. V oblasti testovací analýzy je možné vytvořit scénáře pomocí několika možností jako jsou nový scénář, klonování, import. Po vytvoření scénářů

nezbývá, než je přiřadit do konkrétního testovacího plánu a sprintu a přiřadit zodpovědného testera za jejich exekuci.

Tester provede exekuci testů přímo v daném projektu testy mohou skončit v několika stavech „Passed“, „Failed“, „Not Completed“, „No Run“. Pokud skončí v jiném stavu, než Passed je možné rovnou v tomto kroku vytvořit chybu a přiřadit ji na vývojáře. Po provedení testu je zobrazen celkový čas strávený exekucí testu.

4.3 Základní popis využitého nástroje pro tvorbu automatizovaných testů

K vytvoření a exekuci automatizovaných testů byl využit nástroj Eclipse IDE dostupný na adrese http://www.eclipse.org/ide. Autor zvolil tento nástroj z důvodu předešlých zkušeností při tvorbě automatizovaných testů a jednodušší demonstrace jejich tvorby.

Do vývojového prostředí Eclipse musejí být před samotným vytvářením automatizovaných testů naimportovány knihovny pro podporu Selenium WebDriver pro jazyk Java. Do projektu musí být dále naimportován Chrome Driver. Knihovny a Chrome driver jsou volně dostupné na adrese www.seleniumhq.org/download a jsou nezbytné pro tvorbu automatizovaného testu. V rámci knihoven je také naimportován rozšíření tzv. framework JUnit, který slouží pro psaní testů v Selenium. JUnit není nutnou podmínkou pro tvorbu automatizovaných testů, ale značně tvorbu usnadňuje.

V praxi by bylo vhodné provést testování na všech prohlížečích podporovaných testovanou aplikací.

4.4 Specifikace případů užití

V této části práce za budeme zabývat obecným popisem testovacích případů pro zvolenou testovanou oblast z důvodu dosavadního nepokrytí testy a pro demonstraci automatizovaných a manuálních testů a jejich použití. V následujících částech se již bude nacházet přímo detailní tvorba testovacích scénářů.

Vývoj manuálních a automatizovaných testů je závislý na detailním popisu testovacích případů, které obsahují konkrétní postup pro otestování funkčnosti aplikace.

Seznámením s celkovou funkčností aplikace bylo zjištěno, že aplikace je navržena tak, že práce v ní je rozdělena na mnoho rolí a další role mohou být libovolně tvořeny.

Z tohoto důvodu je zvolena pro testování pouze 1 role, a to hlavní role Super Admin. Super Admin obsahuje veškerá oprávnění v aplikaci a tím nemůže dojít k situaci, kdy by test nedoběhl kvůli nedostatečným oprávněním. Na základě těchto poznatků a v rámci rozsahu bakalářské práce jsou zvoleny následující 4 případy užití na otestování základních funkčností ve vybrané oblasti:

1. Průchod částí aplikace – Design v projektu
2. Vytvoření nové Story
3. Upravení Story
4. Odstranění Story

4.4.1 Průchod částí aplikace – Design v projektu

Popis:

První testovací případ se zabývá otestováním hlavních funkčností oblasti s názvem „Design v projektu“. Po přihlášení do webové aplikace se uživatel nachází na „Homepage“, uživatel přejde do záložky projekty, v té otevře projekt a v navigačním menu zvolí oblast „Design“. Zde nejprve otevře „Epic“, ve kterém se nachází nové funkčnosti k otestování. Zde uživatel postupně otestuje funkčnost „Story“ a jejího detailu, „Test Coverage“ a jeho detailu, „History“ a návrat na „Homepage“. Po úspěšném projití těchto částí následuje odhlášení z aplikace. Testovací scénář pro otestování těchto funkčností bude obsahovat následující kroky.

Základní kroky:

1. Otevření webové aplikace Juno
2. Přihlášení do aplikace
    - Email: agrunt@denevy.eu
    - Heslo: Heslo.123
3. Zavření uvítacího okna
4. Otevření projektu
5. Otevření Designu v projektu
6. Otevření Epicu v Designu
7. Otevření Story v Epicu
8. Otevření detailu Story
9. Vrácení zpět na Epic

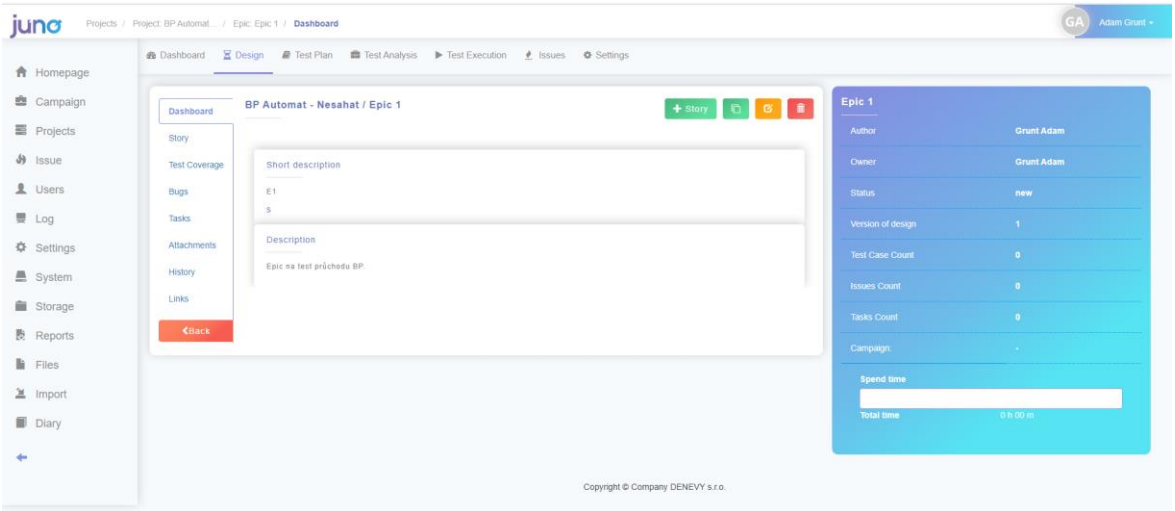
10. Otevření Test Coverage v Epicu
11. Otevření detailu Test Coverage
12. Vrácení zpět na Epic
13. Otevření History
14. Vrácení na Homepage
15. Odhlášení z aplikace

Aktéři:

- Uživatel
- Aplikace

Předpoklady:

- Uživatel je založen v aplikaci a má příslušnou roli Super Admin
- V aplikaci se nachází vytvořený Projekt
- Projekt obsahuje vytvořený Epic
- Epic obsahuje vytvořenou Story



Obrázek 1: Epic v Designu

4.4.2 Vytvoření nové Story

Popis:

Druhý testovací případ slouží k otestování funkčnosti vytvoření nové Story v Epicu. Uživatel se nejprve musí přihlásit do webové aplikace, nyní se nachází na Homepage. Uživatel přejde do záložky projekty, v té otevře projekt a v navigačním menu zvolí oblast

Design. Zde otevře Epic, ve kterém se nachází oblast Story, která bude v tomto testovacím scénáři testována, konkrétně vytvoření nové Story v Epicu. Po úspěšném vytvoření Story, se uživatel vrátí na Homepage a následuje odhlášení z aplikace. Testovací scénář pro otestování těchto funkčností bude obsahovat následující kroky.

Základní kroky:

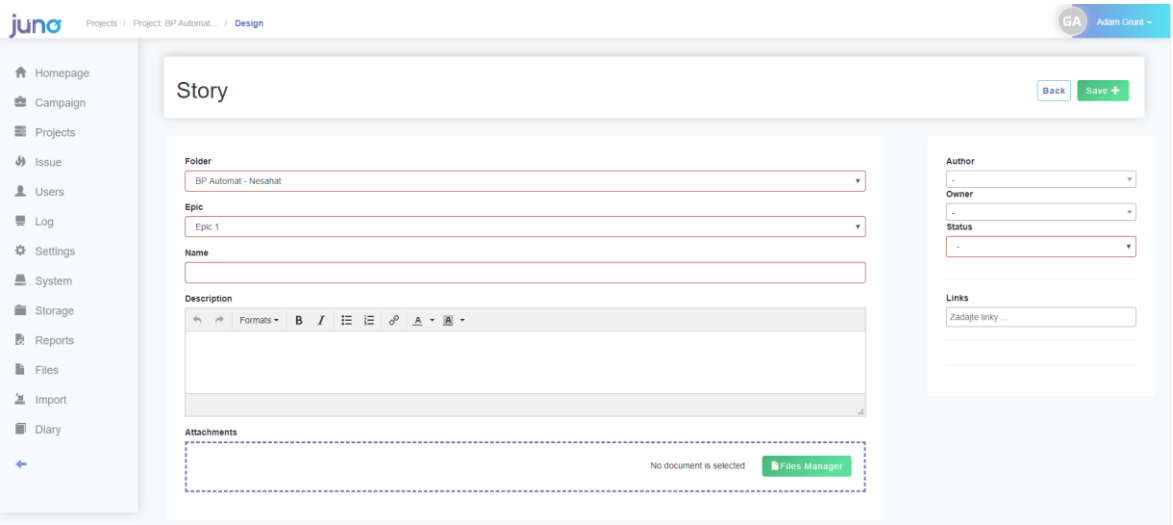
1. Otevření webové aplikace Juno
2. Přihlášení do aplikace
    - Email: agrunt@denevy.eu
    - Heslo: Heslo.123
3. Zavření uvítacího okna
4. Otevření projektu
5. Otevření Designu v projektu
6. Otevření Epicu v Designu
7. Otevření Story v Epicu
8. Vytvoření nové Story
    - Name: A + náhodně vygenerovaná čísla
    - Status: New
    - Owner: Adam Grunt
    - Author: Adam Grunt
9. Vrácení na Homepage
10. Odhlášení z aplikace

Aktéři:

- Uživatel
- Aplikace

Předpoklady:

- Uživatel je založen v aplikaci a má příslušnou roli Super Admin
- V aplikaci se nachází vytvořený Projekt
- Projekt obsahuje vytvořený Epic



Obrázek 2: Formulář na vytvoření Story

4.4.3 Upravení Story

Popis:

Třetí testovací případ je zaměřen na otestování funkčnosti upravení Story v Epicu. Uživatel se nejprve přihlásí do webové aplikace, zobrazí se Homepage. Uživatel přejde do záložky projekty, v té otevře projekt a v navigačním menu zvolí oblast Design. Zde otevře Epic, ve kterém se nachází oblast Story, která bude v tomto testovacím scénáři testována, konkrétně upravení Story v Epicu. Po úspěšném upravení neboli editaci Story se uživatel vrátí na Homepage a následuje odhlášení z aplikace. Testovací scénář pro otestování těchto funkčností bude obsahovat následující kroky.

Základní kroky:

1. Otevření webové aplikace Juno
2. Přihlášení do aplikace
    - Email: agrunt@denevy.eu
    - Heslo: Heslo.123
3. Zavření uvítacího okna
4. Otevření projektu
5. Otevření Designu v projektu
6. Otevření Epicu v Designu
7. Otevření Story v Epicu
8. Upravení již vytvořené Story
    - Změna Name: Edit + náhodně vygenerovaná čísla

- Změna Type: intermediate
- Změna Statusu: for approval
- Změna Ownera: Kolarik Lubomir
- Změna Authora: Vesela Katerina

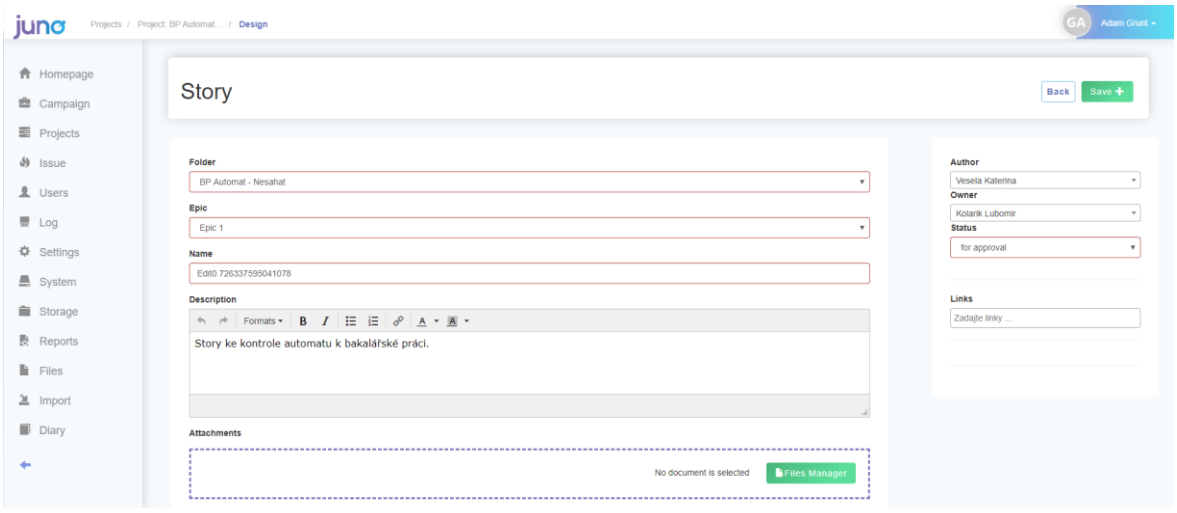
9. Vrácení na Homepage
10. Odhlášení z aplikace

Aktéři:

- Uživatel
- Aplikace

Předpoklady:

- Uživatel je založen v aplikaci a má příslušnou roli Super Admin
- V aplikaci se nachází vytvořený Projekt
- Projekt obsahuje vytvořený Epic
- Epic obsahuje vytvořenou Story



Obrázek 3: Formulář na upravení Story

4.4.4 Odstranění Story

Popis:

Čtvrtý testovací scénář je zaměřen na otestování funkčnosti odstranění Story z Epicu. Uživatel se nejprve přihlásí do webové aplikace, zobrazí se Homepage. Uživatel přejde do záložky projekty, v té otevře projekt a v navigačním menu zvolí oblast Design.

Zde otevře Epic, ve kterém se nachází oblast Story, která bude v tomto testovacím scénáři testována, konkrétně smazání Story v Epicu. Po úspěšném odstranění Story se uživatel vrátí na Homepage a následuje odhlášení z aplikace. Testovací scénář pro otestování těchto funkčností bude obsahovat následující kroky.

Základní kroky:

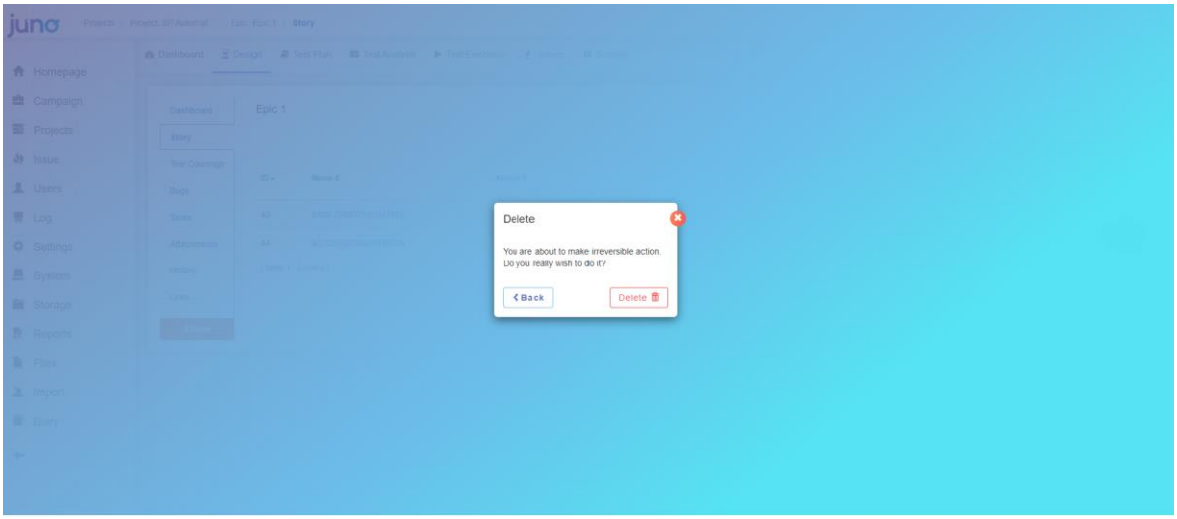
1. Otevření webové aplikace Juno
2. Přihlášení do aplikace
    - Email: agrunt@denevy.eu
    - Heslo: Heslo.123
3. Zavření uvítacího okna
4. Otevření projektu
5. Otevření Designu v projektu
6. Otevření Epicu v Designu
7. Otevření Story v Epicu
8. Odstranění již vytvořené Story
9. Vrácení na Homepage
10. Odhlášení z aplikace

Aktéři:

- Uživatel
- Aplikace

Předpoklady:

- Uživatel je založen v aplikaci a má příslušnou roli Super Admin
- V aplikaci se nachází vytvořený Projekt
- Projekt obsahuje vytvořený Epic
- Epic obsahuje vytvořenou Story



Obrázek 4: Odstranění Story

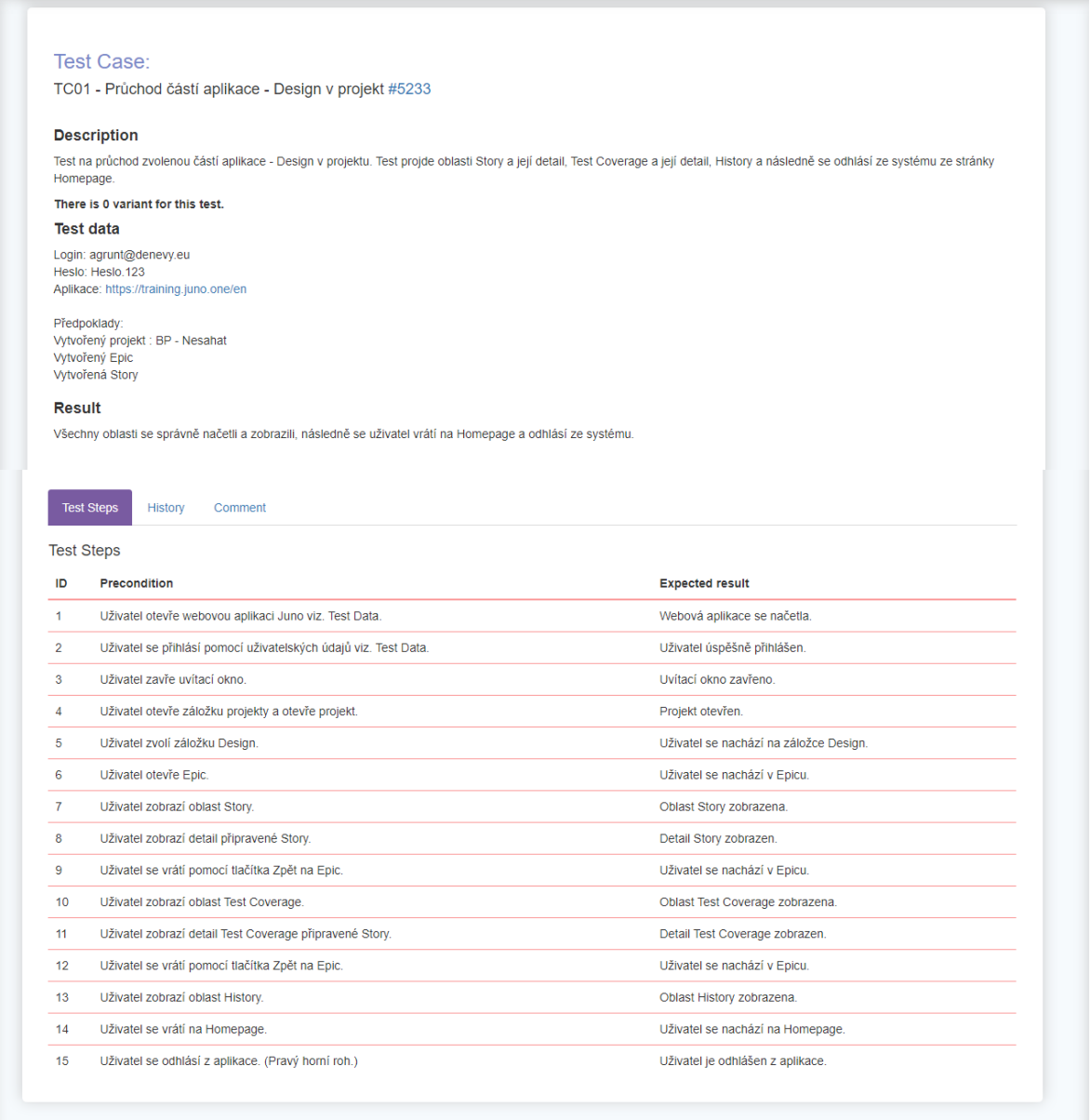
4.5 Tvorba manuálních testů

Z analýz a specifikace případů užití, je možné vytvořit konkrétní testovací scénáře pro manuální i automatické testování. Následující testy jsou zaměřeny pouze na vybranou část aplikace a zvolený proces vybraných funkcionalit.

Pro proces manuálního testování bude využit nástroj Juno, které obsahuje kompletní správu testů a jejich exekucí. Nástroj byl vybrán na základě autorovo předešlé zkušenosti a možnosti využití již obdržené licence z předchozího využívání nástroje. V rámci bakalářské práce nástroj poskytne veškeré potřebné funkčnosti pro vytvoření a provedení exekuce testů a jejich následné vyhodnocení jak pro samotný, tak hromadný běh.

4.5.1 Manuální test Průchod částí aplikace – Design v projektu

V nástroji Juno je možné vytvořit testovací scénář v projektu na záložce „Test Analysis“. Nejprve však musí být vytvořena složka a takzvaný „Test Set“, který bude obsahovat všechny testovací scénáře. Pro každý testovací scénář je nutné zadat jeho název, popis, testovací data, výsledek, prioritu a jednotlivé testovací kroky. Celkové zpracování testu trvalo 10 minut a 43 vteřin.



Test Case:

TC01 - Průchod částí aplikace - Design v projekt #5233

Description

Test na průchod zvolenou částí aplikace - Design v projektu. Test projde oblasti Story a její detail, Test Coverage a její detail, History a následně se odhlásí ze systému ze stránky Homepage.

There is 0 variant for this test.

Test data

Login: agrunt@denevy.eu
Heslo: Heslo.123
Aplikace: https://training.juno.one/en

Předpoklady:
Vytvořený projekt : BP - Nesahat
Vytvořený Epic
Vytvořená Story

Result

Všechny oblasti se správně načetli a zobrazili, následně se uživatel vrátí na Homepage a odhlásí ze systému.

Test Steps | History | Comment

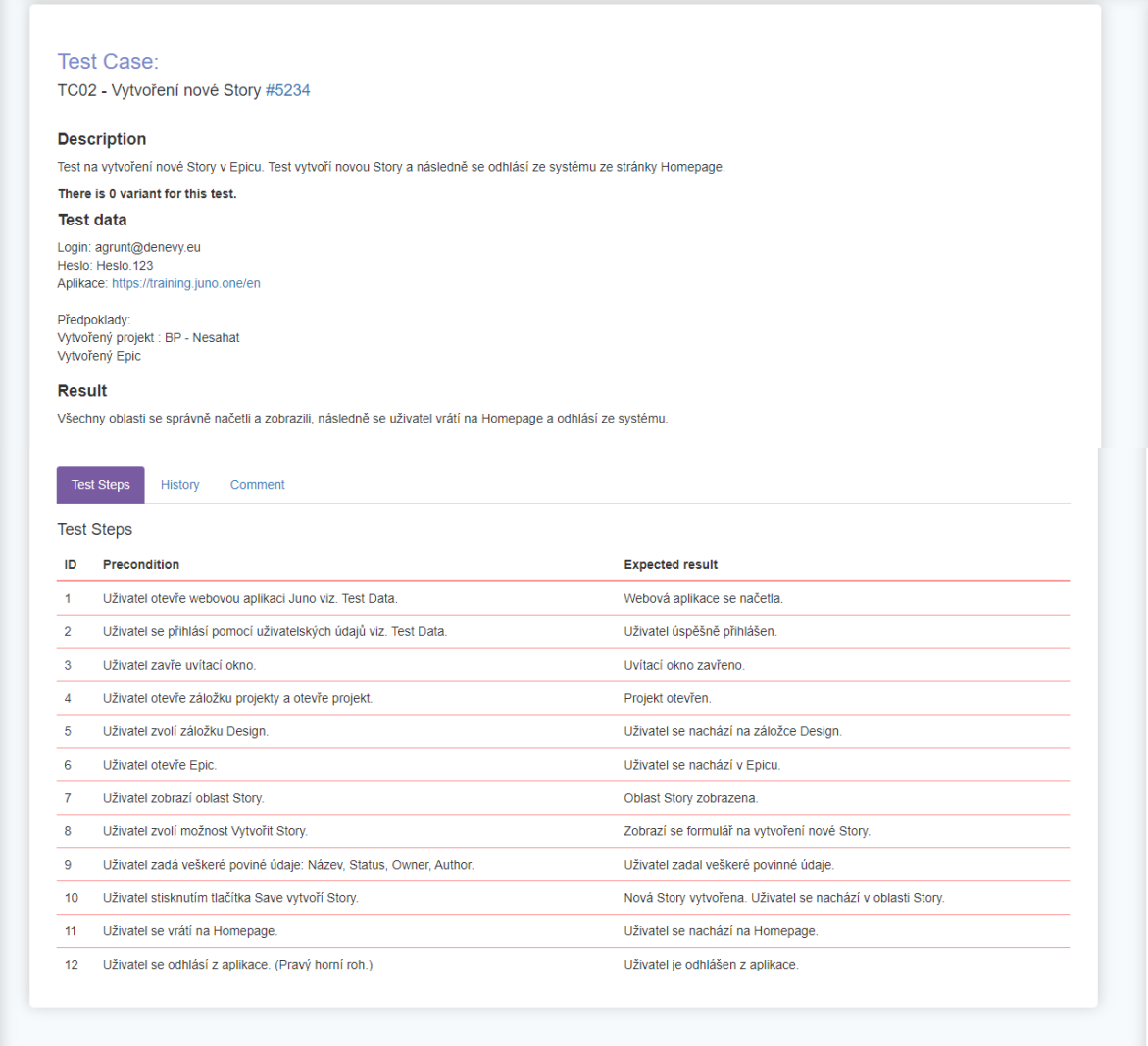
Test Steps

ID	Precondition	Expected result
1	Uživatel otevře webovou aplikaci Juno viz. Test Data.	Webová aplikace se načetla.
2	Uživatel se přihlásí pomocí uživatelských údajů viz. Test Data.	Uživatel úspěšně přihlášen.
3	Uživatel zavře uvítací okno.	Uvítací okno zavřeno.
4	Uživatel otevře záložku projekty a otevře projekt.	Projekt otevřen.
5	Uživatel zvolí záložku Design.	Uživatel se nachází na záložce Design.
6	Uživatel otevře Epic.	Uživatel se nachází v Epicu.
7	Uživatel zobrazí oblast Story.	Oblast Story zobrazena.
8	Uživatel zobrazí detail připravené Story.	Detail Story zobrazen.
9	Uživatel se vrátí pomocí tlačítka Zpět na Epic.	Uživatel se nachází v Epicu.
10	Uživatel zobrazí oblast Test Coverage.	Oblast Test Coverage zobrazena.
11	Uživatel zobrazí detail Test Coverage připravené Story.	Detail Test Coverage zobrazen.
12	Uživatel se vrátí pomocí tlačítka Zpět na Epic.	Uživatel se nachází v Epicu.
13	Uživatel zobrazí oblast History.	Oblast History zobrazena.
14	Uživatel se vrátí na Homepage.	Uživatel se nachází na Homepage.
15	Uživatel se odhlásí z aplikace. (Pravý horní roh.)	Uživatel je odhlášen z aplikace.

Obrázek 5: Manuální scénář Průchod částí aplikace v nástroji Juno (vlastní zpracování autora)

4.5.2 Manuální test Vytvoření nové Story

Obdobným způsobem byl vytvořen testovací případ pro vytvoření nové „Story“. Doba vytvoření testu byla 8 minut a 21 vteřin. Test je znázorněn na následujícím obrázku níže.



Test Case:

TC02 - Vytvoření nové Story #5234

Description

Test na vytvoření nové Story v Epicu. Test vytvoří novou Story a následně se odhlásí ze systému ze stránky Homepage.

There is 0 variant for this test.

Test data

Login: agrunt@denevy.eu
Heslo: Heslo.123
Aplikace: https://training.juno.one/en

Předpoklady:
Vytvořený projekt : BP - Nesahat
Vytvořený Epic

Result

Všechny oblasti se správně načetli a zobrazili, následně se uživatel vrátí na Homepage a odhlásí ze systému.

Test Steps | History | Comment

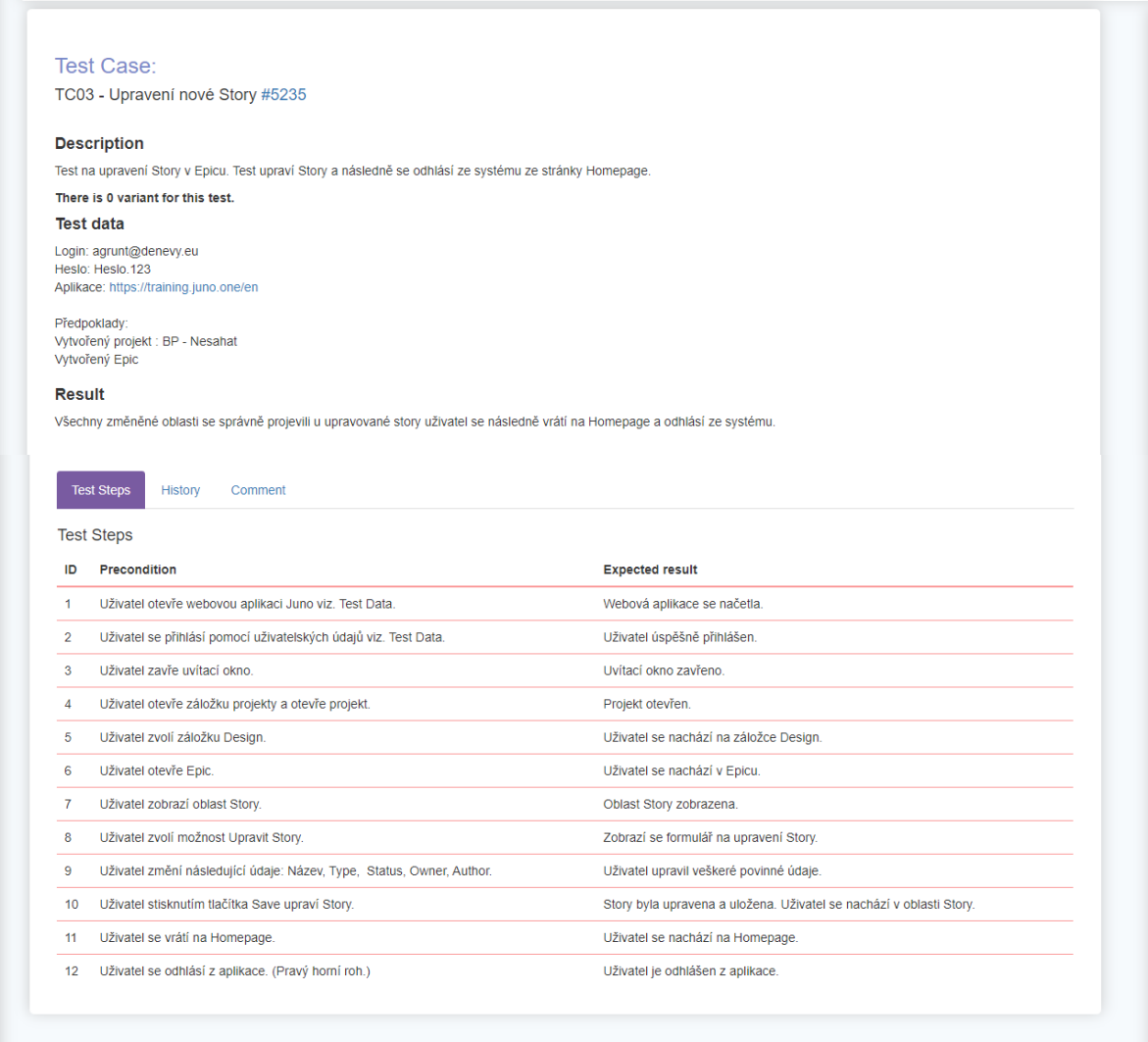
Test Steps

ID	Precondition	Expected result
1	Uživatel otevře webovou aplikaci Juno viz. Test Data.	Webová aplikace se načetla.
2	Uživatel se přihlásí pomocí uživatelských údajů viz. Test Data.	Uživatel úspěšně přihlášen.
3	Uživatel zavře uvítací okno.	Uvítací okno zavřeno.
4	Uživatel otevře záložku projekty a otevře projekt.	Projekt otevřen.
5	Uživatel zvolí záložku Design.	Uživatel se nachází na záložce Design.
6	Uživatel otevře Epic.	Uživatel se nachází v Epicu.
7	Uživatel zobrazí oblast Story.	Oblast Story zobrazena.
8	Uživatel zvolí možnost Vytvořit Story.	Zobrazí se formulář na vytvoření nové Story.
9	Uživatel zadá veškeré povinné údaje: Název, Status, Owner, Author.	Uživatel zadal veškeré povinné údaje.
10	Uživatel stisknutím tlačítka Save vytvoří Story.	Nová Story vytvořena. Uživatel se nachází v oblasti Story.
11	Uživatel se vrátí na Homepage.	Uživatel se nachází na Homepage.
12	Uživatel se odhlásí z aplikace. (Pravý horní roh.)	Uživatel je odhlášen z aplikace.

Obrázek 6: Manuální scénář Vytvoření nové Story v nástroji Juno (vlastní zpracování autora)

4.5.3 Manuální test Upravení Story

Následující scénář byl vytvořen jiným způsobem, a to pomocí klonování, což systém Juno nabízí a autor ho považuje za velký přínos. Tento a předešlý scénář mají velmi podobnou strukturu, tím pádem zde byla využita již zmíněná možnost klonování, která se kladně podepsala na rychlosti vytvoření scénáře na upravení „Story“. Autor tedy klonoval předešlý scénář a upravil pouze oblasti, ve kterých jsou scénáře odlišné. Doba vytvoření testu byla 4 minuty a 41 vteřin. Test je znázorněn na následujícím obrázku níže.



Test Case:

TC03 - Upravení nové Story #5235

Description

Test na upravení Story v Epicu. Test upraví Story a následně se odhlásí ze systému ze stránky Homepage.

There is 0 variant for this test.

Test data

Login: agrunt@denevy.eu
Heslo: Heslo.123
Aplikace: https://training.juno.one/en

Předpoklady:
Vytvořený projekt : BP - Nesahat
Vytvořený Epic

Result

Všechny změněné oblasti se správně projevili u upravované story uživatel se následně vrátí na Homepage a odhlásí ze systému.

Test Steps | History | Comment

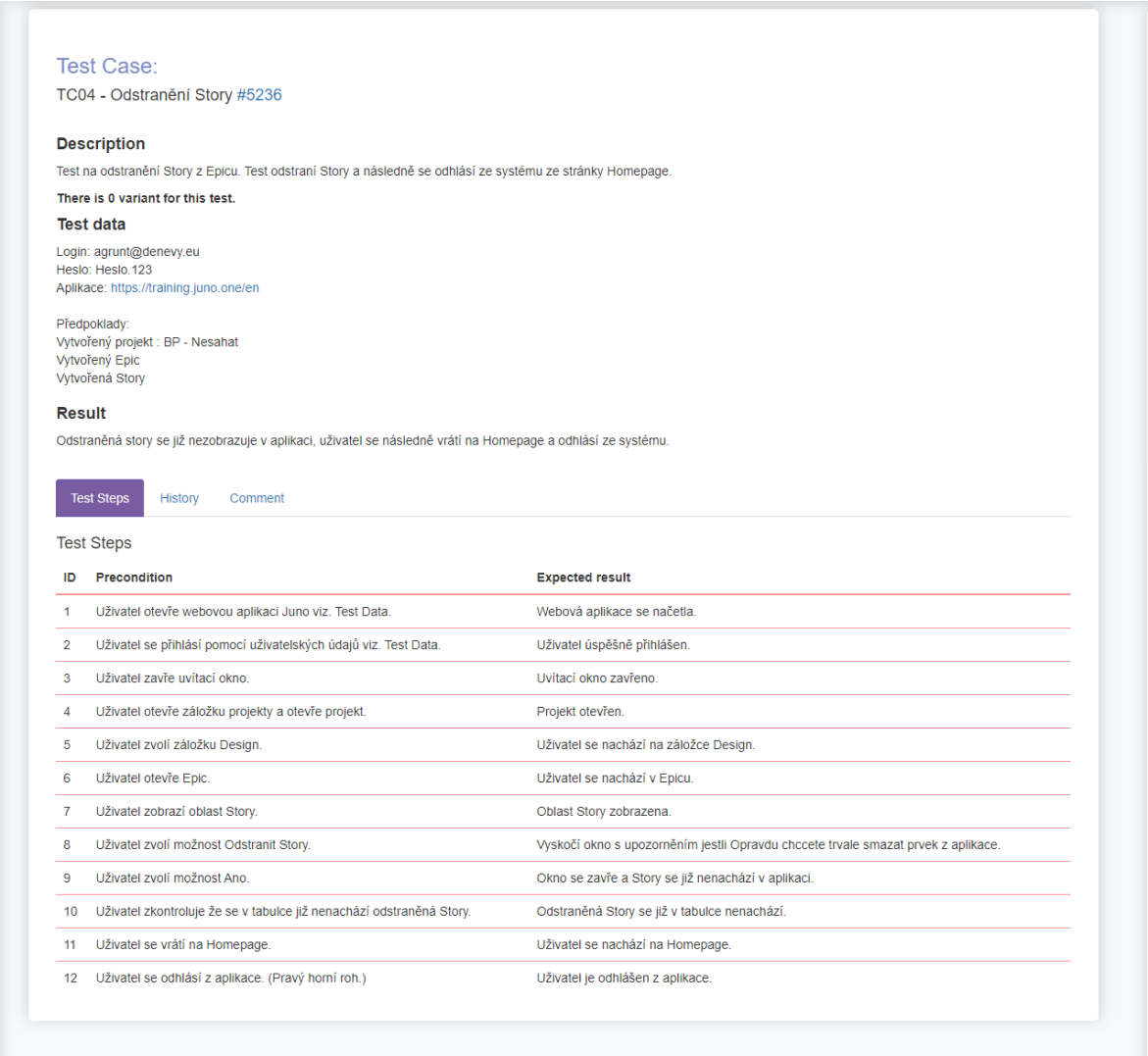
Test Steps

ID	Precondition	Expected result
1	Uživatel otevře webovou aplikaci Juno viz. Test Data.	Webová aplikace se načetla.
2	Uživatel se přihlásí pomocí uživatelských údajů viz. Test Data.	Uživatel úspěšně přihlášen.
3	Uživatel zavře uvítací okno.	Uvítací okno zavřeno.
4	Uživatel otevře záložku projekty a otevře projekt.	Projekt otevřen.
5	Uživatel zvolí záložku Design.	Uživatel se nachází na záložce Design.
6	Uživatel otevře Epic.	Uživatel se nachází v Epicu.
7	Uživatel zobrazí oblast Story.	Oblast Story zobrazena.
8	Uživatel zvolí možnost Upravit Story.	Zobrazí se formulář na upravení Story.
9	Uživatel změní následující údaje: Název, Type, Status, Owner, Author.	Uživatel upravil veškeré povinné údaje.
10	Uživatel stisknutím tlačítka Save upraví Story.	Story byla upravena a uložena. Uživatel se nachází v oblasti Story.
11	Uživatel se vrátí na Homepage.	Uživatel se nachází na Homepage.
12	Uživatel se odhlásí z aplikace. (Pravý horní roh.)	Uživatel je odhlášen z aplikace.

Obrázek 7: Manuální scénář Upravení Story v nástroji Juno (vlastní zpracování autora)

4.5.4 Manuální test Odstranění Story

Poslední zvolený scénář byl znovu vytvořen pomocí klonování předešlého scénáře. Autor tedy opět klonoval předešlý scénář a upravil pouze oblasti, ve kterých jsou scénáře odlišné. Doba vytvoření testu byla 4 minuty a 20 vteřin. Test je znázorněn na následujícím obrázku níže.



Test Case:

TC04 - Odstranění Story #5236

Description

Test na odstranění Story z Epicu. Test odstraní Story a následně se odhlásí ze systému ze stránky Homepage.

There is 0 variant for this test.

Test data

Login: agrunt@denevy.eu
Heslo: Heslo.123
Aplikace: https://training.juno.one/en

Předpoklady:
Vytvořený projekt : BP - Nesahat
Vytvořený Epic
Vytvořená Story

Result

Odstraněná story se již nezobrazuje v aplikaci, uživatel se následně vrátí na Homepage a odhlásí ze systému.

Test Steps | History | Comment

Test Steps

ID	Precondition	Expected result
1	Uživatel otevře webovou aplikaci Juno viz. Test Data.	Webová aplikace se načetla.
2	Uživatel se přihlásí pomocí uživatelských údajů viz. Test Data.	Uživatel úspěšně přihlášen.
3	Uživatel zavře uvítací okno.	Uvítací okno zavřeno.
4	Uživatel otevře záložku projekty a otevře projekt.	Projekt otevřen.
5	Uživatel zvolí záložku Design.	Uživatel se nachází na záložce Design.
6	Uživatel otevře Epic.	Uživatel se nachází v Epicu.
7	Uživatel zobrazí oblast Story.	Oblast Story zobrazena.
8	Uživatel zvolí možnost Odstranit Story.	Vyskočí okno s upozorněním jestli Opravdu chccete trvale smazat prvek z aplikace.
9	Uživatel zvolí možnost Ano.	Okno se zavře a Story se již nenachází v aplikaci.
10	Uživatel zkontroluje že se v tabulce již nenachází odstraněná Story.	Odstraněná Story se již v tabulce nenachází.
11	Uživatel se vrátí na Homepage.	Uživatel se nachází na Homepage.
12	Uživatel se odhlásí z aplikace. (Pravý horní roh.)	Uživatel je odhlášen z aplikace.

Obrázek 8: Manuální scénář Odstranění Story v nástroji Juno (vlastní zpracování autora)

4.6 Exekuce manuálních testů

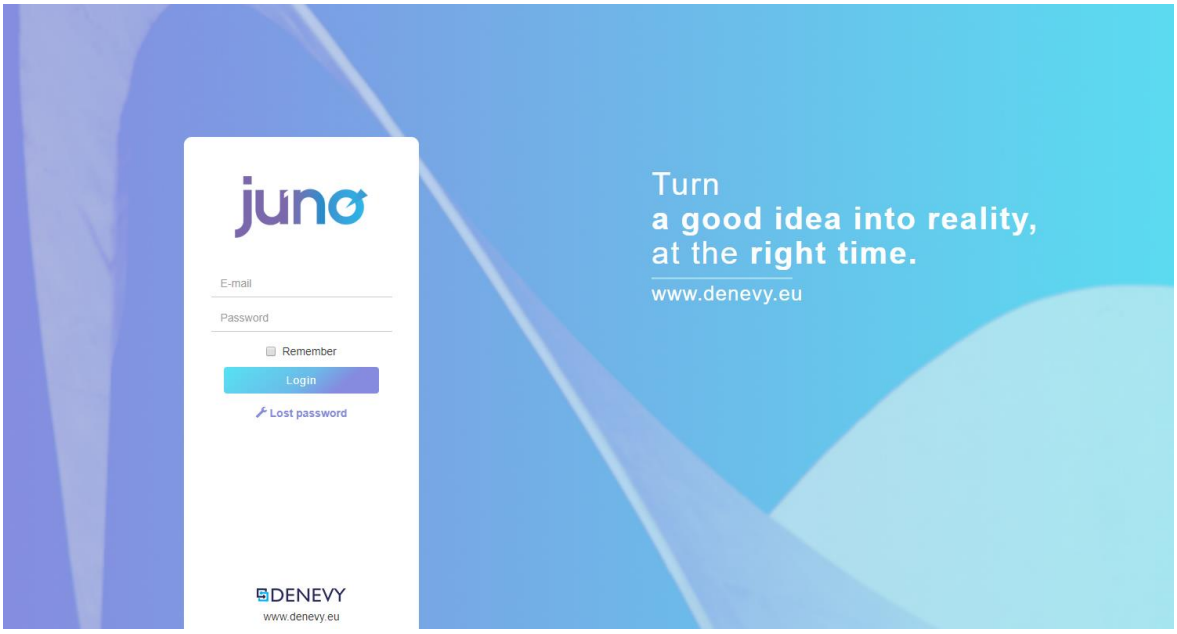
Exekucí testů nebo také spuštění či provedení testů krok po kroku je myšlena činnost pro dosažení stanoveného cíle. Po vytvoření manuálních testů přišlo jejich samotné vykonání.

Jelikož Juno je komplexní nástroj obsahující možnost provádět manuálních testy a uchovávat výsledné a průběžné stavy pro jednotlivé testy, autor tedy na základě vlastní zkušenosti a urychlení procesu vykonání testů skrze znalost systému tuto možnost využil.

4.6.1 Exekuce testu Průchod částí aplikace – Design v projektu

Před samotným otestováním procesu, musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné vykonat test. Tento scénář se zabývá jednoduchým průchodem částí aplikace.

Prvním krokem testu je přihlášení do webové aplikace Juno. Na následujícím obrázku je zobrazena přihlašovací stránka.



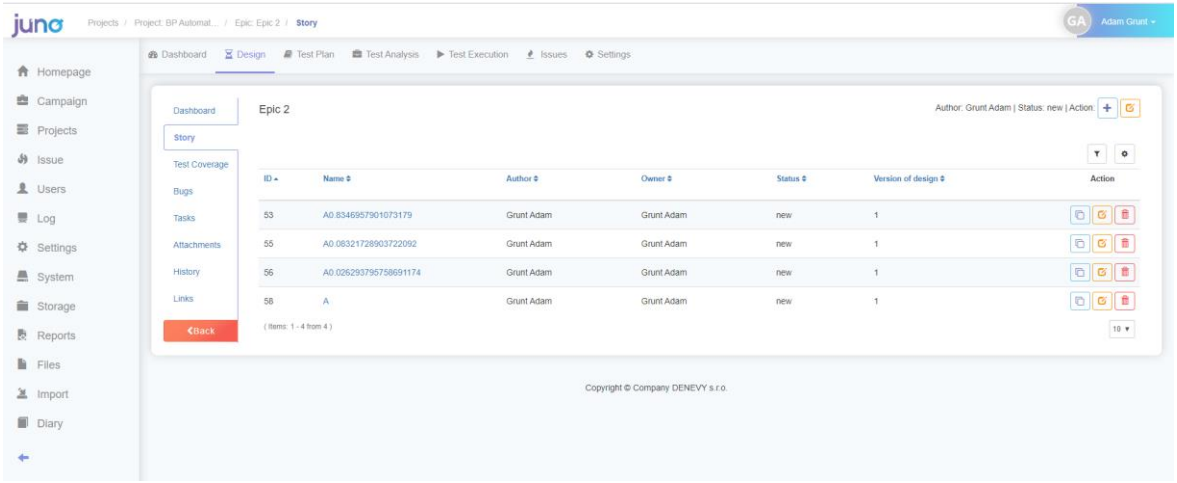
Obrázek 9: Přihlášení do webové aplikace Juno (vlastní zpracování autora)

Po úspěšném přihlášení je dalším krokem uzavření uvítacího okna pro nové uživatele. Po jeho uzavření se uživatel nachází na Homepage. Z této oblasti přejde do oblasti Projekt.

Nyní se uživatel nachází na záložce obsahující projekty, po splnění předpokladu je zde již vytvořený projekt a uživatel ho otevře. Po jeho otevření se nachází na Dashboardu projektu.

Uživatel otevře záložku Design, ve které se nacházejí objekty s názvem Epic. Otevře Epic, po jeho otevření se nachází na jeho Dashboardu.

Epic se dělí na několik testovaných částí, uživatel přejde do první z nich a to Story. Uživatel se nachází na záložce Story, zvolí a otevře Detail některé Story. Na následujícím obrázku je zobrazena oblast Story.



Obrázek 10: Oblast Story (vlastní zpracování autora)

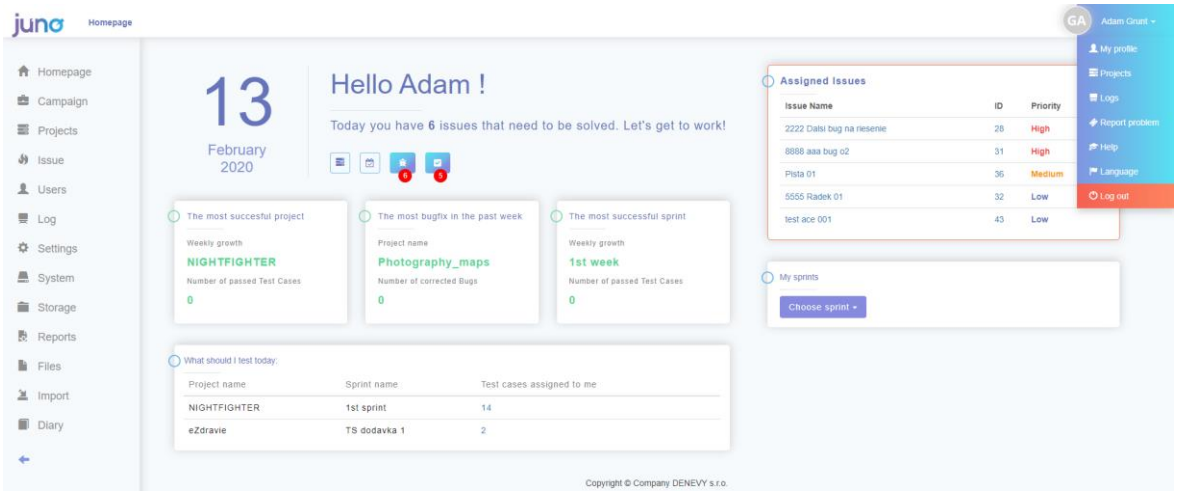
Uživatel v další fázi přejde na oblast Test Coverage, kde se nacházejí jednotlivé pokrytí Story testy, poté zobrazí její detail.

V dalším kroku uživatel přejde do oblasti History, kde jsou zaznamenány veškeré práce, vytvoření, úpravy nebo odstranění v některé z oblastí v Epicu. Na následujícím obrázku je ukázka záznamu v History, jedná se o vytvoření nové Story, záznam obsahuje autora, datum vytvoření, typ změny a status.

Name	Author of change	Type of change	Change description	Date of change	Status
A0.8346957901073179	Grunt Adam	new		26.11.2019	new

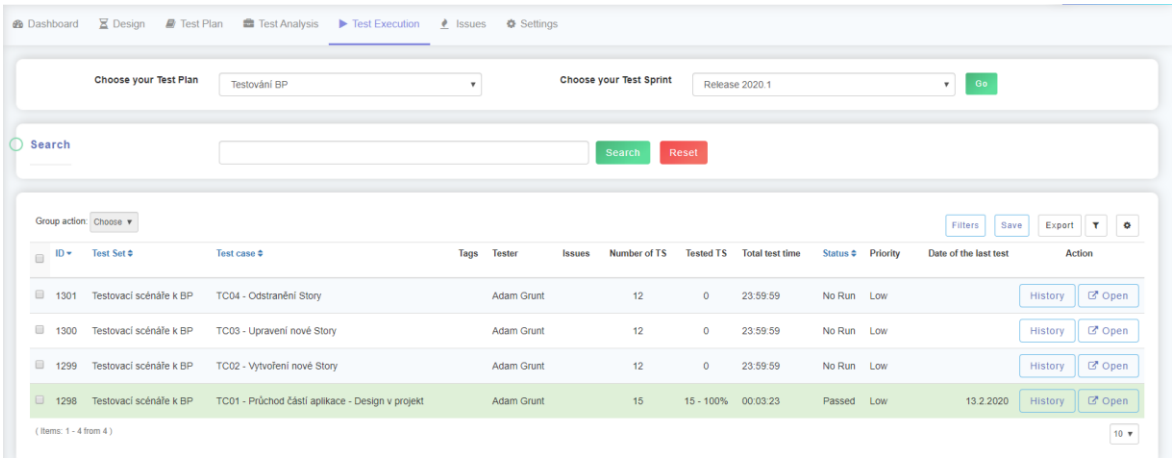
Obrázek 11:Ukázka záznamu v oblasti History (vlastní zpracování autora)

V poslední fázi uživatel přejde na Homepage a v pravém horním rohu se odhlásí z aplikace. Na následujícím obrázku je zobrazeno menu na Homepage sloužící pro odhlášení z aplikace.



Obrázek 12: Odhlášení ze systému Juno (vlastní zpracování autora)

Celý proces proběhl v pořádku a nikde se nezobrazila žádná chybová hláška, všechny protestované oblasti se načetly v pořádku a uživatel se odhlásil korektně ze systému. Test tedy bude v testovacím nástroji Juno vyhodnocen jako „Passed“.



Obrázek 13: Vyhodnocení test Průchod částí aplikace – Design v projektu (vlastní zpracování autora)

Na obrázku číslo 13. je vidět pod názvem „Total test time“ (v překladu do češtiny Celková doba testování) celkový čas strávený testováním. Test průchodu částí aplikace trval po dobu 3 minuty 23 vteřin.

Doba exekuce se odvíjí od zkušeností uživatele s aplikací, rychlostí internetu, opoždění odezvy a jiné technologické problémy.

4.6.2 Exekuce testu Vytvoření nové Story

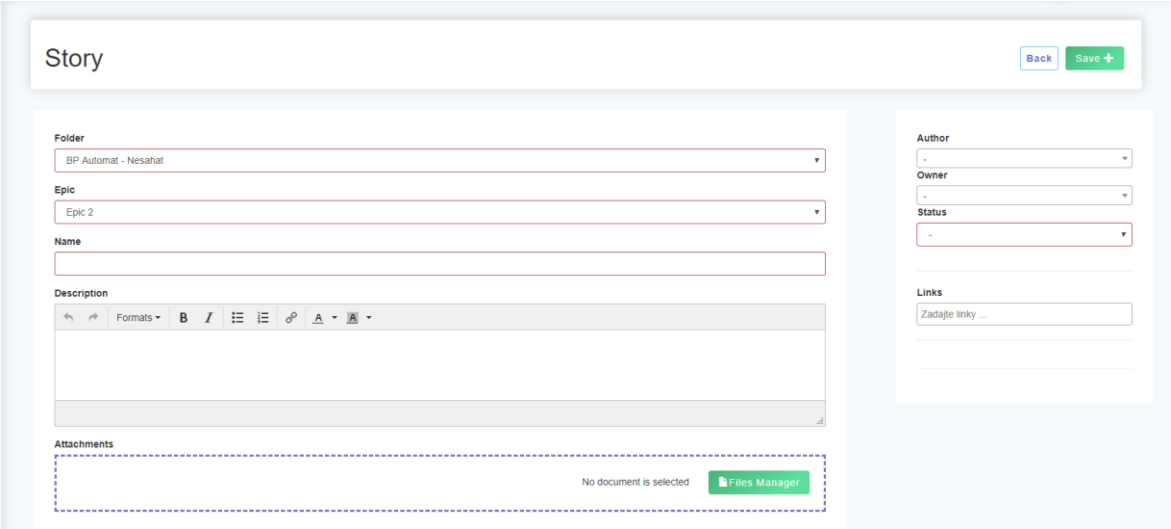
Před samotným otestováním procesu, musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné vykonat test. Tento scénář se zabývá vytvořením nové Story.

Prvním krokem testu je přihlášení do webové aplikace Juno. Po úspěšném přihlášení je dalším krokem uzavření uvítacího okna pro nové uživatele. Po jeho uzavření se uživatel nachází na Homepage.

Z této oblasti přejde do oblasti Projekt. Nyní se uživatel nachází na záložce obsahující projekty, po splnění předpokladu je zde již vytvořený projekt a uživatel ho otevře. Po jeho otevření se nachází na Dashboardu projektu.

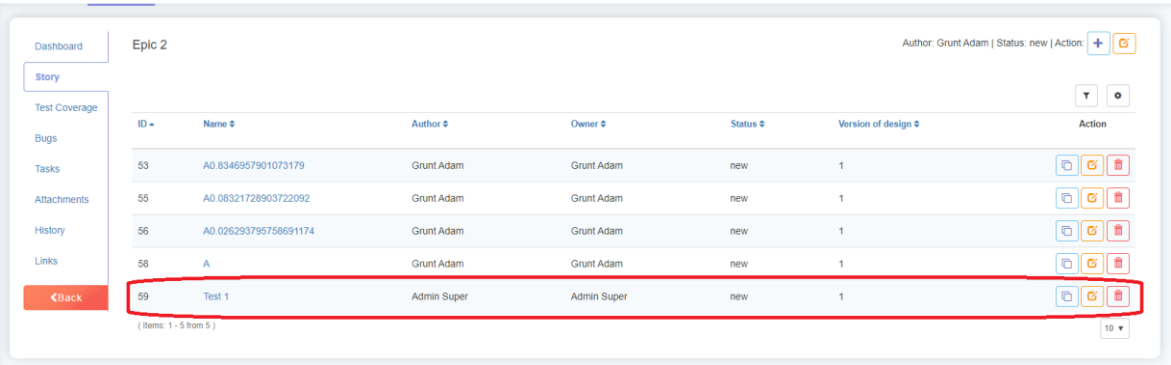
Uživatel otevře záložku Design, ve které se nacházejí objekty s názvem Epic. Otevře Epic, po jeho otevření se nachází na jeho Dashboardu.

Uživatel přejde do oblasti Story. Uživatel se nachází na záložce Story a zvolí možnost vytvořit novou Story. Na následujícím obrázku je zobrazen formulář pro vytvoření nové Story.



Obrázek 14: Formulář na vytvoření Story (vlastní zpracování autora)

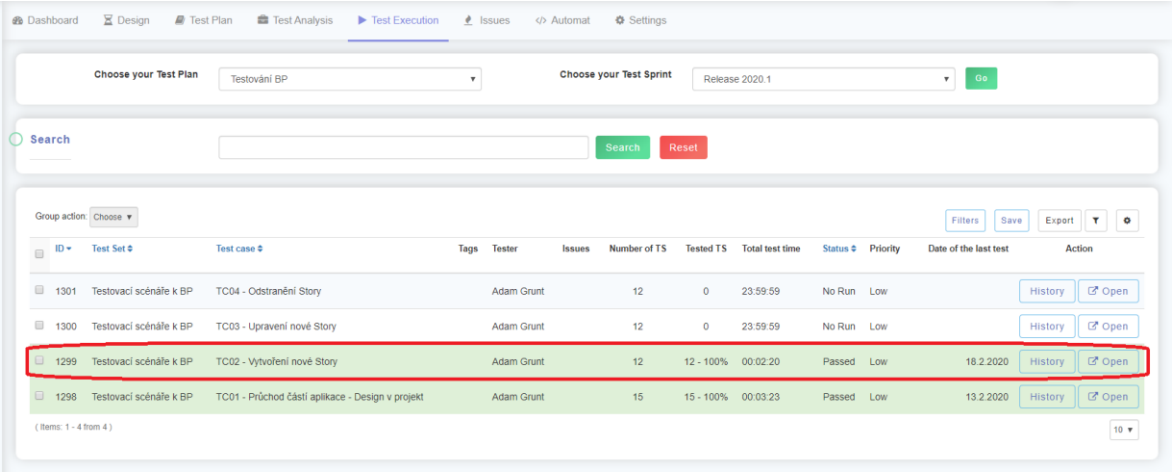
Uživatel zadá veškeré potřebné údaje, jako jsou Název, Status, Owner, Author a potvrdí vytvoření nové story tlačítkem Save. Na následujícím obrázku je zobrazena nově vytvořená Story v tabulce všech Storek.



Obrázek 15: Tabulka Storek – Vytvořená Story (vlastní zpracování autora)

V poslední fázi uživatel přejde na Homepage a v pravém horním rohu se odhlásí z aplikace. Na následujícím obrázku je zobrazeno menu na Homepage sloužící pro odhlášení z aplikace.

Celý proces proběhl v pořádku a nikde se nezobrazila žádná chybová hláška, nově vytvořená Story je uložena v aplikaci a uživatel se korektně odhlásil ze systému. Test tedy bude v testovacím nástroji Juno vyhodnocen jako „Passed“.



Obrázek 16: Vyhodnocení testu Vytvoření nové Story (vlastní zpracování autora)

Na obrázku číslo 16. je po provedení exekuce opět vidět pod názvem „Total test time“ (v překladu do češtiny Celková doba testování) celkový čas strávený testováním. Test Vytvoření nové Story trval po dobu 2 minuty 20 vteřin.

V porovnání s prvním testem je test proveden rychleji, a to z důvodu menší rozsáhlosti oblasti testování.

4.6.3 Exekuce testu Upravení Story

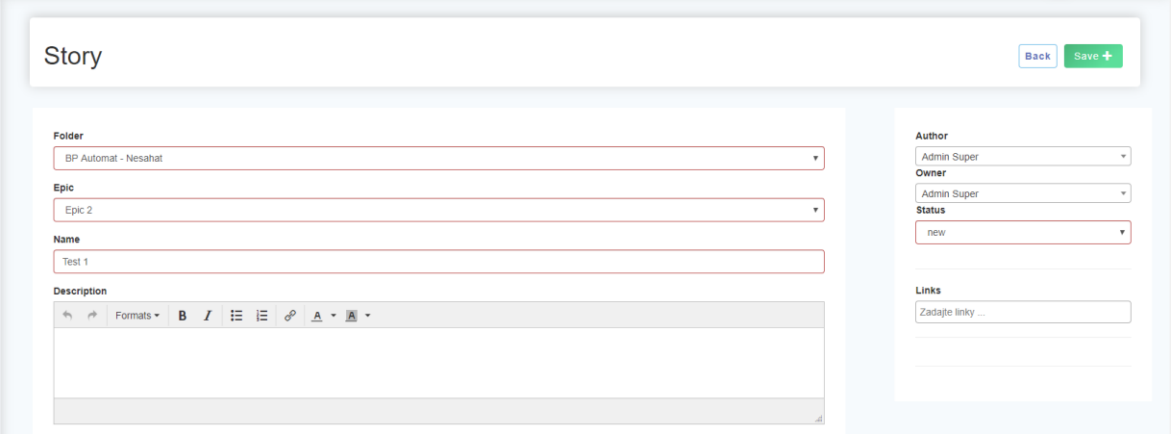
Před samotným otestováním procesu musí být splněny veškeré předpoklady a vytvořena testovací data. Po zajištění těchto podmínek je možné vykonat test. Tento scénář je zaměřen na upravení Story.

Prvním krokem testu je přihlášení do webové aplikace Juno. Po úspěšném přihlášení je dalším krokem uzavření uvítacího okna pro nové uživatele. Po jeho uzavření se uživatel nachází na Homepage.

Z této oblasti přejde do oblasti Projekt. Nyní se uživatel nachází na záložce obsahující projekty, po splnění předpokladu je zde již vytvořený projekt a uživatel ho otevře. Po jeho otevření se nachází na Dashboardu projektu.

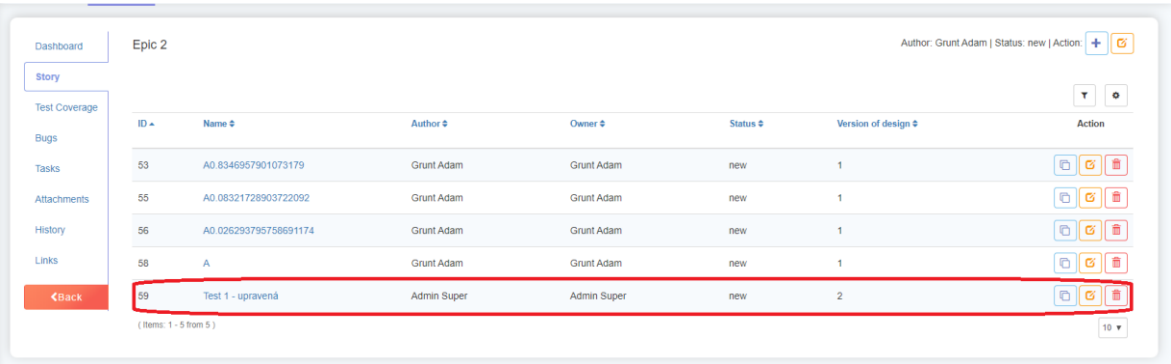
Uživatel otevře záložku Design, ve které se nacházejí objekty s názvem Epic. Otevře Epic, po jeho otevření se nachází na jeho Dashboardu.

Uživatel přejde do oblasti Story. Uživatel se nachází na záložce Story a zvolí možnost upravit Story. Na následujícím obrázku je zobrazen formulář na upravení Story.



Obrázek 17: Formulář na upravení Story (vlastní zpracování autora)

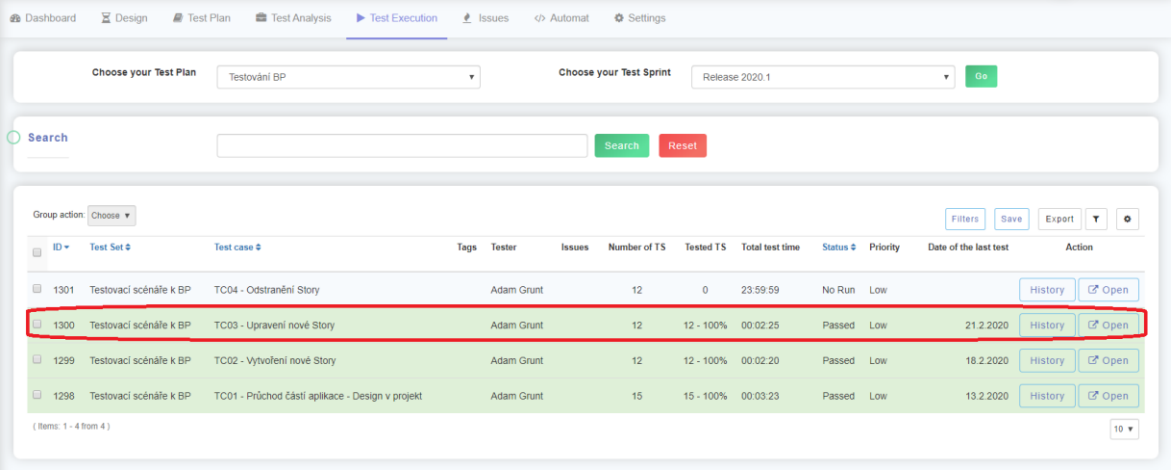
Uživatel upraví údaje v kolonkách Název, Type, Status, Owner, Author a potvrdí upravení Story pomocí tlačítka Save. Na následujícím obrázku je zobrazena upravená Story.



Obrázek 18: Tabulka Storek – Upravená Story (vlastní zpracování autora)

V poslední fázi uživatel přejde na Homepage a v pravém horním rohu se odhlásí z aplikace. Na následujícím obrázku je zobrazeno menu na Homepage sloužící pro odhlášení z aplikace.

Celý proces proběhl v pořádku a nikde se nezobrazila žádná chybová hláška, upravená Story je uložena v aplikaci s nově zadanými údaji a uživatel se korektně odhlásil ze systému. Test tedy bude v testovacím nástroji Juno vyhodnocen jako „Passed“.



Obrázek 19: Vyhodnocení testu Upravení Story (vlastní zpracování autora)

Na obrázku číslo 19. je po provedení exekuce opět vidět pod názvem „Total test time“ (v překladu do češtiny Celková doba testování) celkový čas strávený testováním. Test Upravení Story trval po dobu 2 minuty 25 vteřin.

V porovnání s prvním testem je test proveden rychleji, a to z důvodu menší rozsáhlosti oblasti testování.

V porovnání s testem druhým na Vytvoření nové Story se časy testování téměř shodují, jelikož test obsahuje stejně kroků a podobně náročný proces.

4.6.4 Exekuce testu Odstranění Story

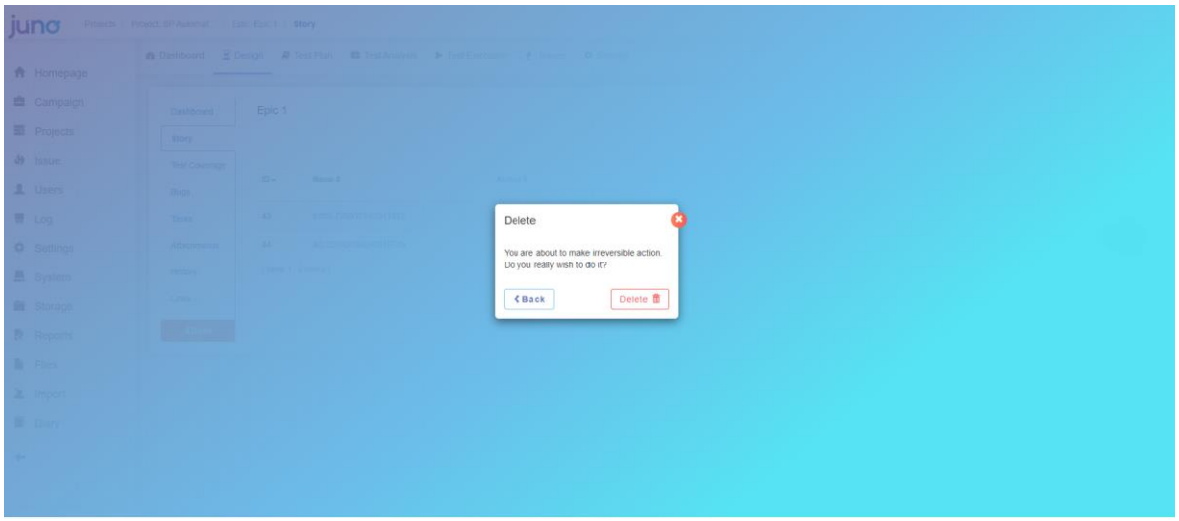
Před samotným otestováním procesu musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné vykonat test. Tento scénář je zaměřen na odstranění Story.

Prvním krokem testu je přihlášení do webové aplikace Juno. Po úspěšném přihlášení je dalším krokem uzavření uvítacího okna pro nové uživatele. Po jeho uzavření se uživatel nachází na Homepage.

Z této oblasti přejde do oblasti Projekt. Nyní se uživatel nachází na záložce obsahující projekty, po splnění předpokladu je zde již vytvořený projekt a uživatel ho otevře. Po jeho otevření se nachází na Dashboardu projektu.

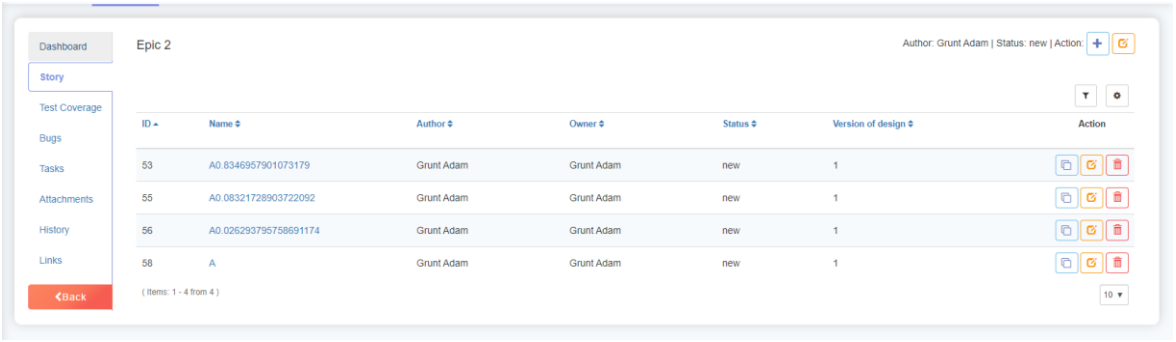
Uživatel otevře záložku Design, ve které se nacházejí objekty s názvem Epic. Otevře Epic, po jeho otevření se nachází na jeho Dashboardu.

Uživatel přejde do oblasti Story. Uživatel se nachází na záložce Story a zvolí možnost odstranit Story. Na následujícím obrázku je zobrazeno potvrzovací okno na odstranění Story.



Obrázek 20: Vyskakovací okno na Odstranění Story (vlastní zpracování autora)

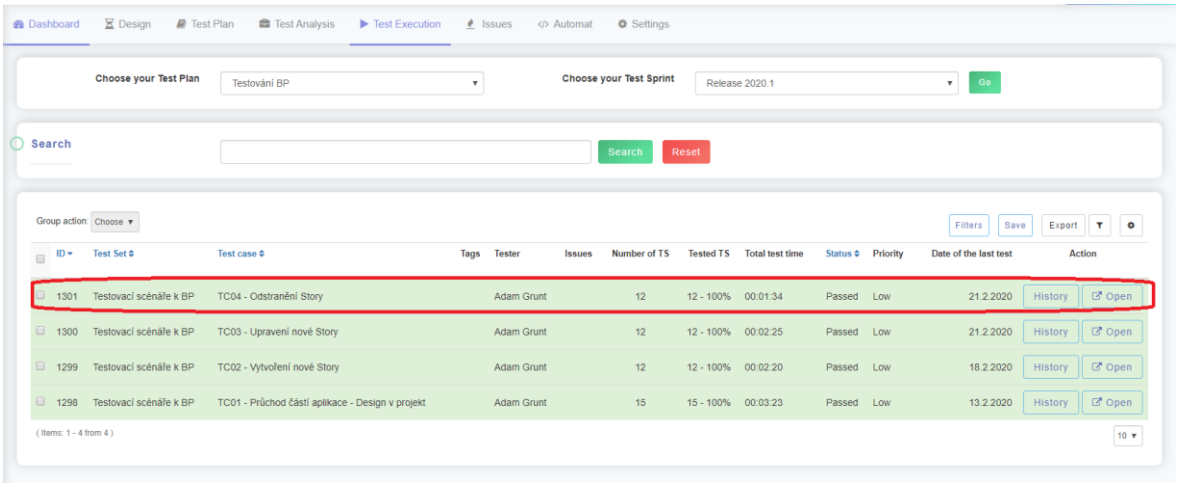
Uživatel zvolí na potvrzovacím oknu možnost Ano, potvrzovací okno se zavře a v tabulce všech Storek se již nezobrazuje Odstraněná Story. Na následujícím obrázku je zobrazena tabulka již bez odstraněné Story.



Obrázek 21: Tabulka Storek – Odstraněná Story (vlastní zpracování autora)

V poslední fázi uživatel přejde na Homepage a v pravém horním rohu se odhlásí z aplikace. Na následujícím obrázku je zobrazeno menu na Homepage sloužící pro odhlášení z aplikace.

Celý proces proběhl v pořádku a nikde se nezobrazila žádná chybová hláška, odstraněná Story se již nenachází v aplikaci a uživatel se korektně odhlásil ze systému. Test tedy bude v testovacím nástroji Juno vyhodnocen jako „Passed“.



Obrázek 22: Vyhodnocení testu Odstranění Story (vlastní zpracování autora)

Na obrázku číslo 22. je po provedení exekuce opět vidět pod názvem „Total test time“ (v překladu do češtiny Celková doba testování) celkový čas strávený testováním. Test Odstranění Story trval po dobu 1 minuty 34 vteřin.

V porovnání s prvním testem je test proveden rychleji, a to z důvodu menší rozsáhlosti oblasti testování.

V porovnání s testem druhým na Vytvoření nové Story a třetím na Upravení Story se časy testování trochu liší, jelikož test na Odstranění obsahuje pouze potvrzení odstranění ve vyskakovacím okně a není nutné zadávat nové údaje.

4.7 Tvorba automatizovaných testů

V této části bude obsažena konkrétní tvorba automatických testovacích scénářů podle případů užití, předpokladem pro jejich vytvoření je znalost webové aplikace, její grafické rozhraní, funkčnosti, jednotlivé možnosti pro vyvolání tíženého procesu. Tester by měl být seznámen s těmito funkčnosti do detailu, například z dokumentací aplikace, z funkčních diagramů, informací od vývojářů a jiné. Tester nemusí do detailu znát celou aplikaci, ale pouze ty oblasti, které se týkají tvořených automatizovaných testů, ale je velkou výhodou, pokud aplikaci do detailu zná.

Před samotným vytvořením testů, je důležité vyřešit nejprve samotné spuštění a načtení webové aplikace během testu. K tomu v Seleniu slouží tzv. WebDriver. Před spuštěním jakýkoliv kroků v testu, je potřeba tento WebDriver načíst z lokálního úložiště

a vytvořit jeho instanci, k tomu je vytvořena jednoduchá funkce. Použití této funkce pro načtení WebDriveru, který je umístěn ve složce worskpace\\Bp_Automat_Design na disku D:\\ vypadá následovně:

```
/* Function called BEFORE testing */
@Before
public void setup() throws MalformedURLException {


    File chromeDriver = new File("D:\\workspace\\BP_Automat_Design\\chromedriver.exe");
    System.setProperty("webdriver.chrome.driver", chromeDriver.getAbsolutePath()); // loading Webdriver
    driver = new ChromeDriver(); // driver instance

  /* Implicit wait - waiting for all elements to load, maximum of 5 seconds */
  driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

}
```

Obrázek 22: Načtení WebDriveru (vlastní zpracování autora)

Poté co je tento WebDriver načten může se přistoupit k načtení kokrétní testované webové aplikace a to Juno následujícím způsobem:

```
/* TEST */
@Test
public void main() throws InterruptedException, MalformedURLException, IOException
{
    File srcFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); // Screenshots
   /* First page initialization */
   driver.get("https://training.juno.one/en/admin/login");
   driver.manage().window().maximize();
```

Obrázek 23: Načtení webové aplikace (vlastní zpracování autora)

Pro vysvětlení kódu, příkaz ***driver.get(„https://training.juno.one/en/admin/login“);*** znamená, že instance driveru si načte a zavolá webovou aplikaci Juno, a příkaz ***driver.manage().window().maximize();*** zvětší okno na celou obrazovku.

Toto řešení je obecné a přece použitelné a to tak, že pokud by bylo potřeba odkázat se na jinou webovou aplikaci, stačí zaměnit volanou URL adresu za jinou požadovanou.

Po vyřešení spouštění aplikace přichází na řadu samotný testovací scénář. Aby bylo možné pracovat s objekty, je potřeba znát možnosti, jak tyto objekty identifikovat ve webové aplikaci. V Seleniu je několik možností, jak tyto prvky identifikovat. Pro příklad, zde budou uvedeny nejpoužívanější z nich:

1. Xpath
    - Relativní cesta
    - Absolutní cesta
2. ID

3. ClassName
4. Name
5. CSS selector
6. A jiné

Pro tvorbu testovacích scénářů byla zvolena volba identifikace pomocí Xpath absolutní cesta, která má své výhody i nevýhody, které budou rozebrány v diskusi.

Pro práci s jednotlivými prvky na webové aplikace slouží několik základních příkazů. Na vývoj zvolených automatizovaných testů budou použity následující příkazy:

1. click() – metoda pro kliknutí na zvolený prvek
2. sendKeys() – metoda sloužící k napsání a odeslání textu
3. waitForElement() – metoda, která vyvolá časovou relaci na načtení prvku
4. findElement() – metoda na vyhledání prvku na stránce

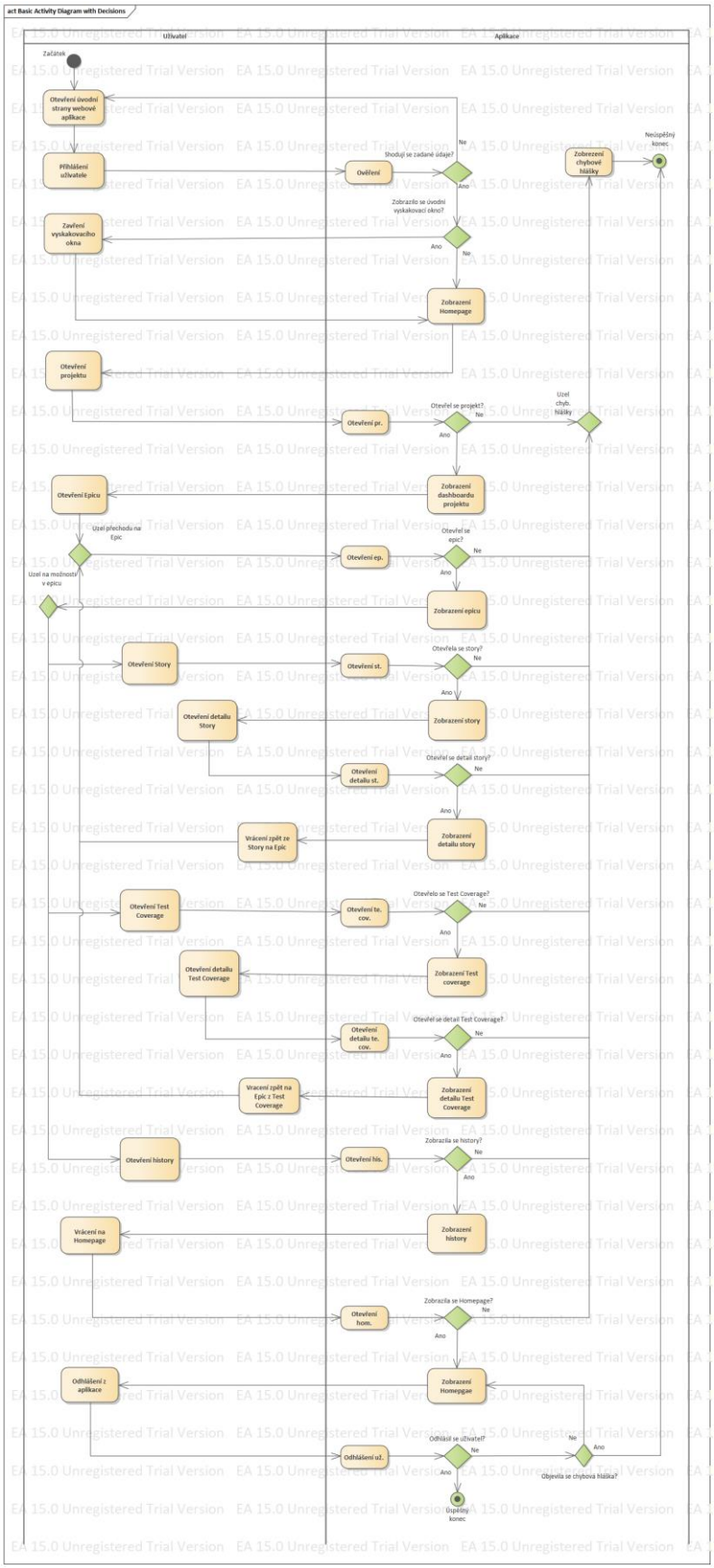
4.7.1 Automatizovaný test Průchod částí aplikace – Design v projektu

Na základě vytvořeného případu užití „Průchod částí aplikace – Design v projektu“ a po využití předešlých funkcí a příkazů byl vytvořen první automatizovaný test na průchod oblasti „Designu“ v aplikaci. Zdrojový kód je součástí přílohy (Příloha 1). Pro testovací scénář byl vytvořen grafický model, který znázorňuje jednotlivé akce uživatele a chování aplikace v reakci na danou akci. Automatizovaný test může skončit dvěma způsoby, úspěšně a neúspěšně.

V případě úspěšného průchodu testované oblasti, test v grafické podobě začíná v akci Začátek a po průchodu všemi částmi testu končí v bodě Úspěšný konec.

V případě, že test z důvodu nějaké chyby skončí neúspěšně, začíná takový test rovněž v bodě Začátek a v okamžiku výskytu chyby se přesune na cílový bod Neúspěšný konec.

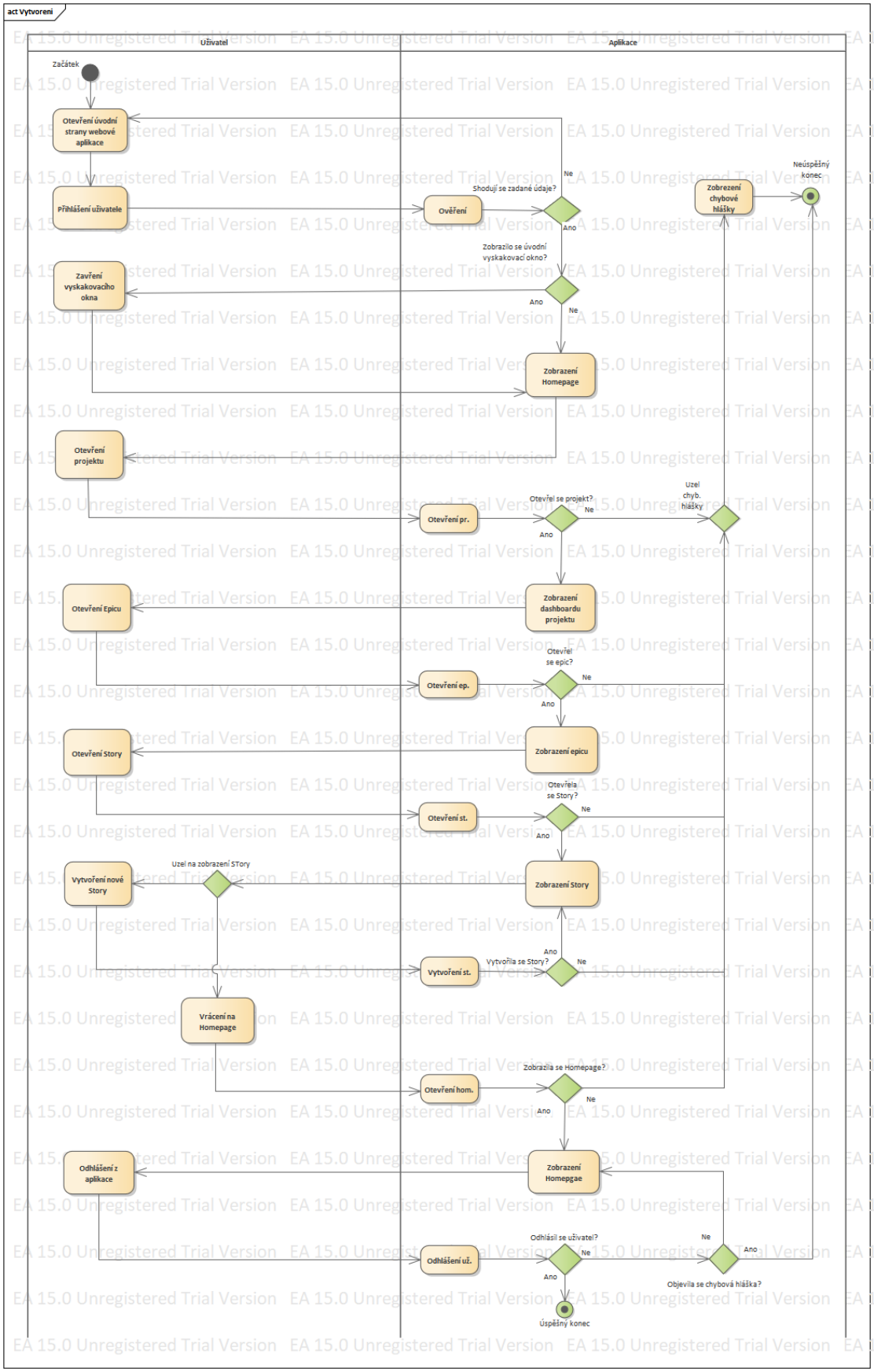
Tvorba automatizovaného testu „Průchod částí aplikace – Design v projektu“ trval 25 minut a 16 vteřin.



Obrázek 24: Vývojový diagram testu na průchod částí aplikace – Design v projektu (vlastní zpracování autora)

4.7.2 Automatizovaný test Vytvoření nové Story

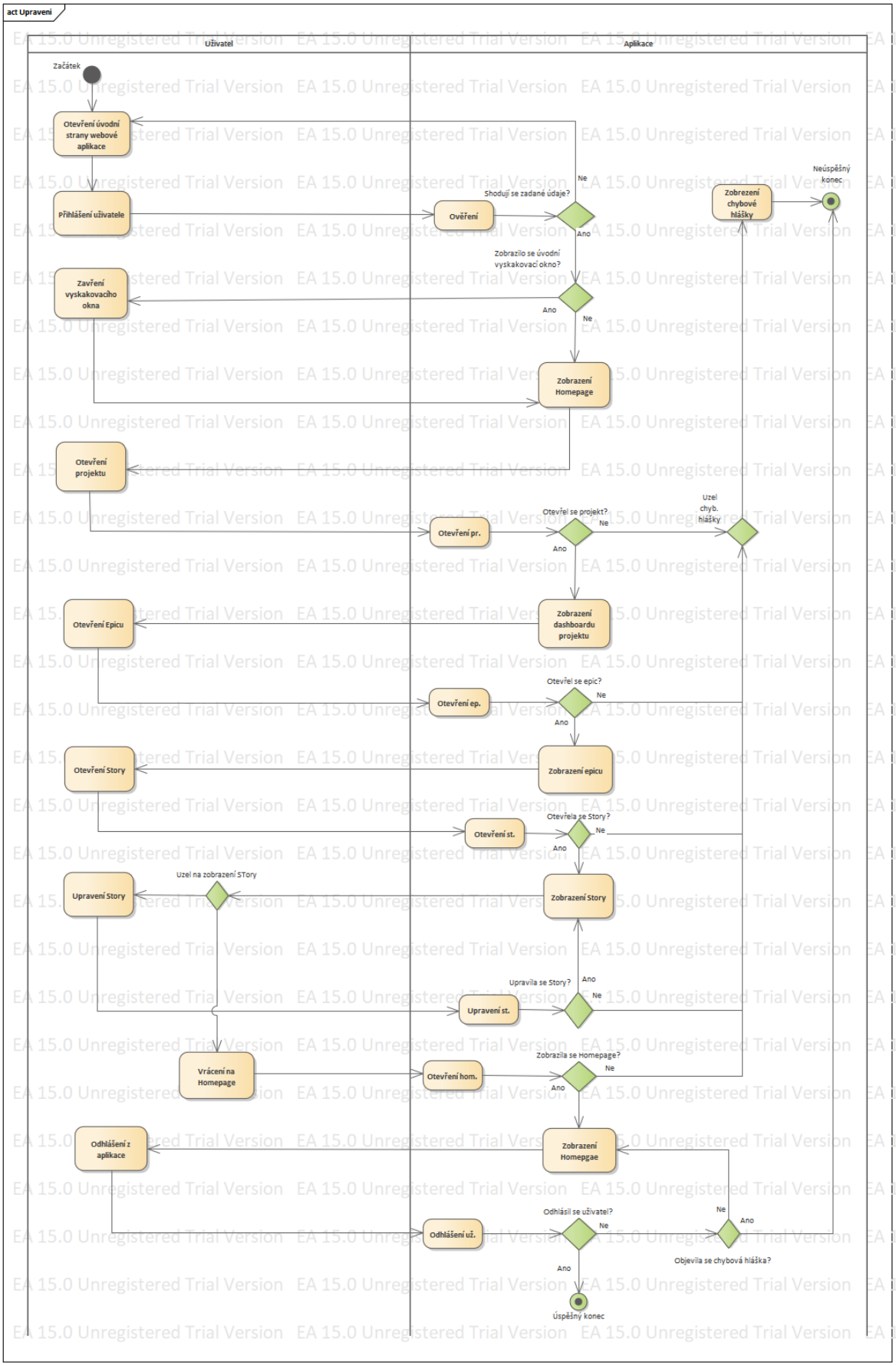
Na základě vytvořeného případu užití „Vytvoření nové Story“ byl vytvořen automatizovaný test. Zdrojový kód testu se nachází v příloze (Příloha č. 2). Grafické znázornění vytvořeného testu pomocí vývojového diagramu je znázorněno pomocí následujícího obrázku. Tvorba automatizovaného testu „Vytvoření nové Story“ trval 18 minut a 23 vteřin. Z důvodu využití již vytvořených částí skriptu z předešlého scénáře se čas tvorby zkrátil.



Obrázek 25: Vývojový diagram testu Vytvoření nové Story (vlastní zpracování autora)

4.7.3 Automatizovaný test Upravení Story

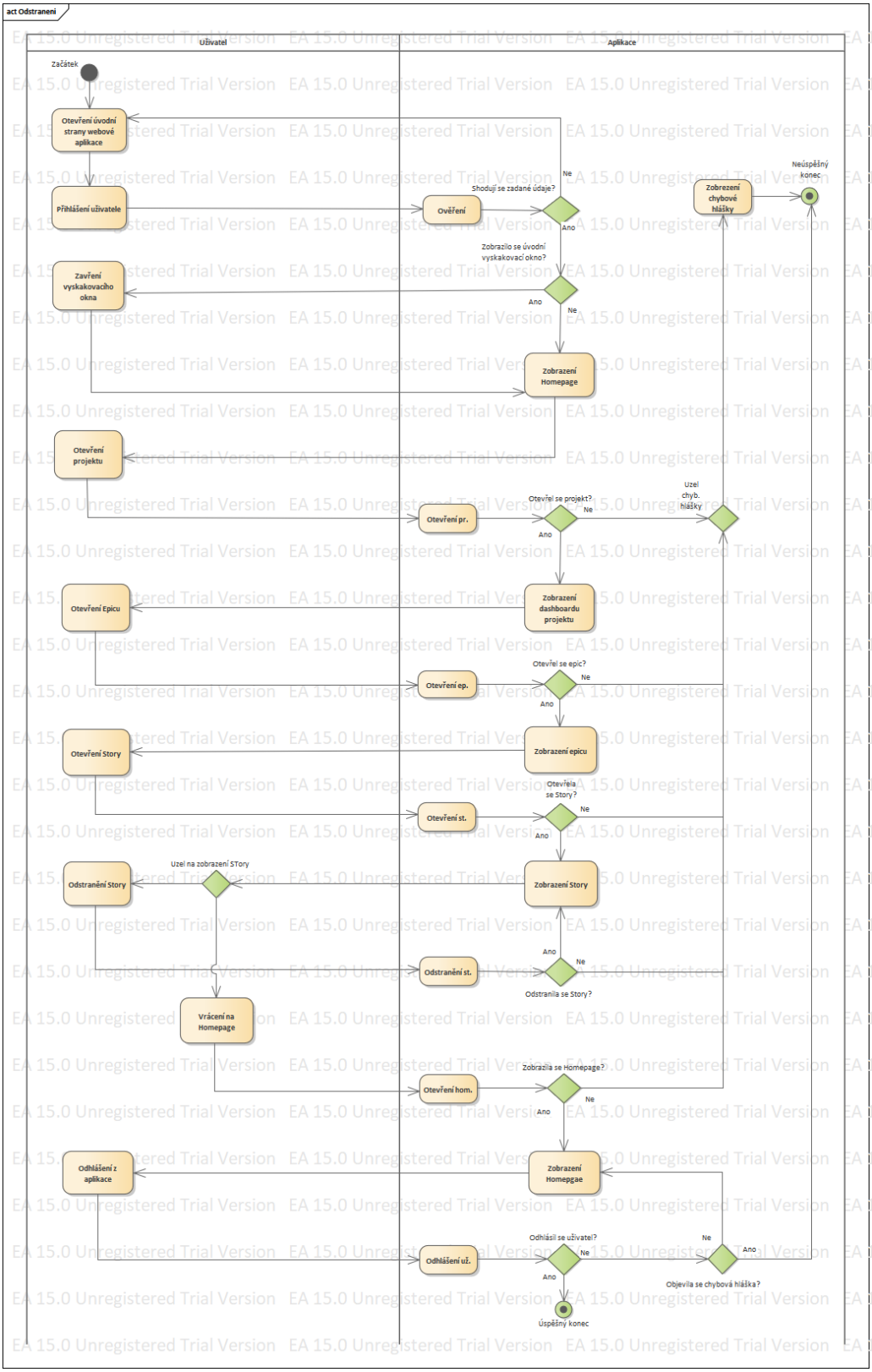
Na základě vytvořeného případu užití „Upravení Story“ byl vytvořen automatizovaný test. Zdrojový kód testu se nachází v příloze (Příloha č. 3). Grafické znázornění vytvořeného testu pomocí vývojového diagramu je znázorněno pomocí následujícího obrázku. Tvorba automatizovaného testu „Upravení Story“ trval 13 minut a 47 vteřin. Stejně jako u předešlého případu jej využito již vytvořených částí skriptu z předešlého scénáře se čas tvorby ještě více zkrátil.



Obrázek 26: Vývojový diagram testu Upravení Story (vlastní zpracování autora)

4.7.4 Automatizovaný test Odstranění story

Na základě vytvořeného případu užití „Odstranění Story“ byl vytvořen automatizovaný test. Zdrojový kód testu se nachází v příloze (Příloha č. 4). Grafické znázornění vytvořeného testu pomocí vývojového diagramu je znázorněno pomocí následujícího obrázku. Tvorba automatizovaného testu „Odstranění Story“ trval 8 minut a 13 vteřin. Z důvodu jednoduššího procesu a využití již vytvořených částí skriptu z předešlých scénářů se čas tvorby zkrátil.



Obrázek 27: Vývojový diagram testu Odstranění Story (vlastní zpracování autora)

4.8 Exekuce automatizovaných testů

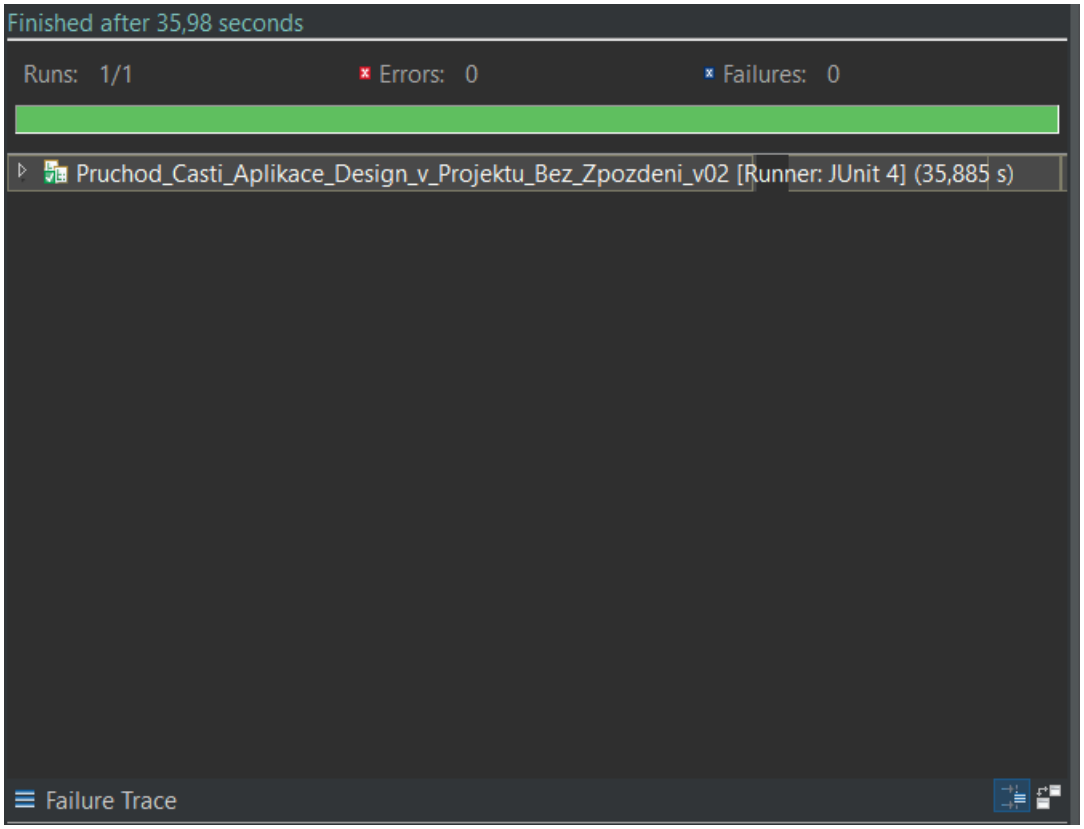
Exekucí testů je myšlena činnost pro dosažení stanoveného cíle. Po vytvoření automatizovaných testů přišlo jejich samotné spuštění.

Nástroj pro spuštění automatizovaných testů známe z kapitoly 4.3 Základní popis využitého nástroje pro tvorbu automatizovaných testů. Testy jsou vytvořeny a uchovány v tomto nástroji, autor na základě vlastní zkušenosti a urychlení procesu vykonání testů skrze znalost nástroje tuto možnost využil.

4.8.1 Exekuce automatizovaného testu Průchod částí aplikace – Design v projektu

Před spuštěním testu musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné spustit automatizovaný test. Tento scénář se zabývá průchodem částí aplikace – „Design“ v projektu.

Jednotlivé kroky testu přesně odpovídají vývojovému diagramu vyobrazenému na obrázku číslo 24. Z tohoto důvodu nebude v této kapitole detailně rozepsán postup jednotlivých kroků. Po úspěšném doběhnutí všech kroků v testu je stav testu „Passed“ a počet chyb roven 0.



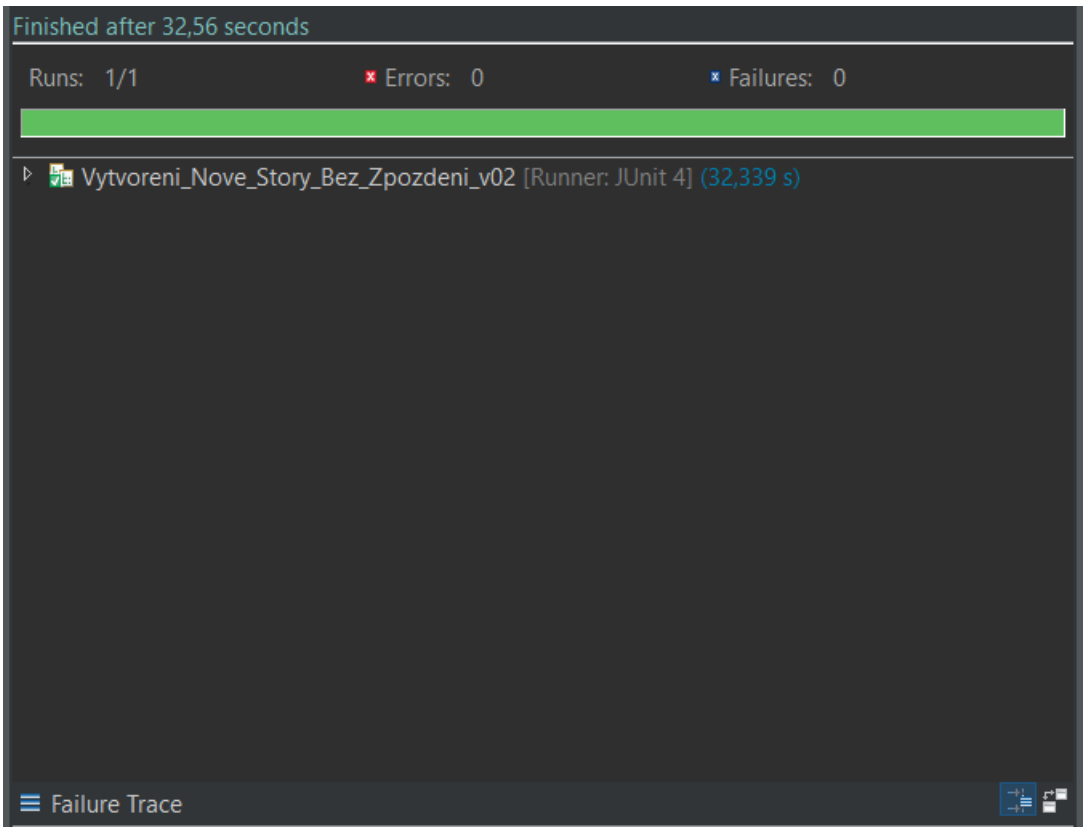
Obrázek 28: Vyhodnocení automatizovaného testu Průchod částí aplikace – Design v projektu (vlastní zpracování autora)

Na obrázku číslo 28. je po spuštění automatizovaného testu „Průchod částí aplikace – Design v projektu“ zobrazen celkový čas strávený testováním. Test „Průchod částí aplikace – Design v projektu“ trval po dobu necelých 36 vteřin.

4.8.2 Exekuce automatizovaného testu Vytvoření nové Story

Před spuštěním testu musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné spustit automatizovaný test. Tento scénář se zabývá vytvořením nové „Story“.

Jednotlivé kroky testu přesně odpovídají vývojovému diagramu vyobrazenému na obrázku číslo 25. Z tohoto důvodu nebude v této kapitole detailně rozepsán postup jednotlivých kroků. Po úspěšném doběhnutí všech kroků v testu je stav testu „Passed“ a počet chyb roven 0.



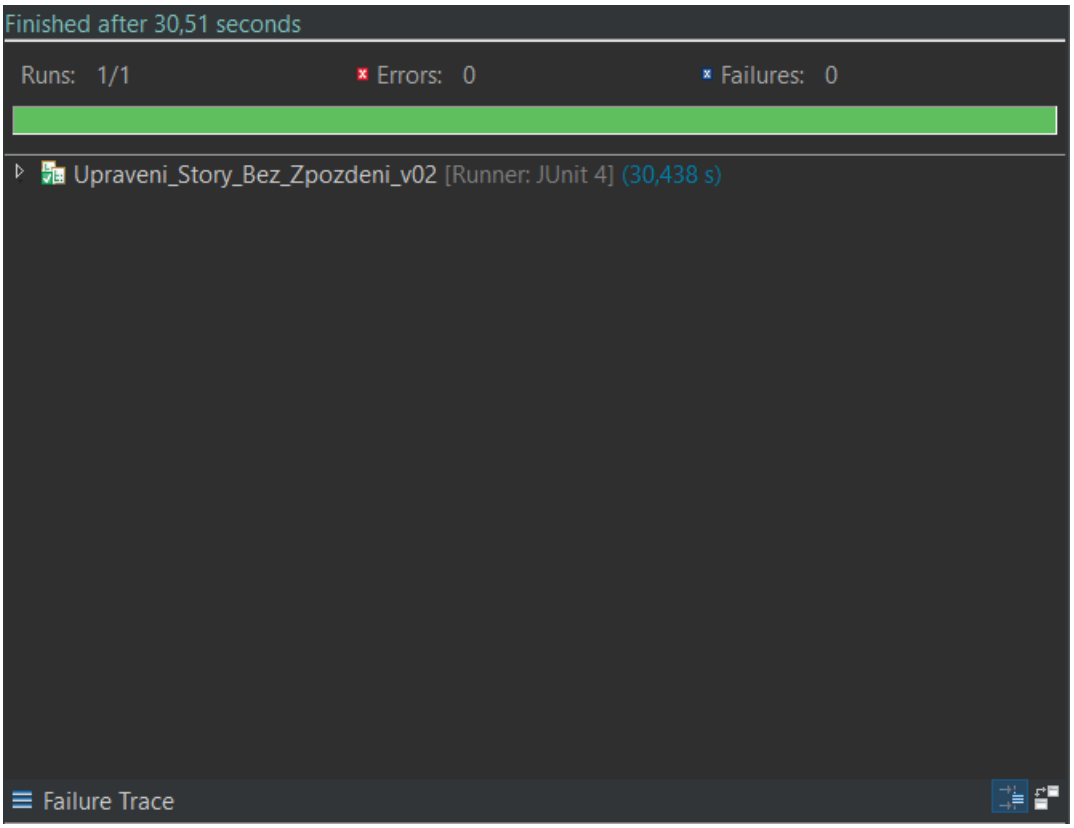
Obrázek 29: Vyhodnocení automatizovaného testu Vytvoření nové Story (vlastní zpracování autora)

Na obrázku číslo 29. je po spuštění automatizovaného testu „Vytvoření nové Story“ zobrazen celkový čas strávený testováním. Test „Vytvoření nové Story“ trval po dobu necelých 33 vteřin.

4.8.3 Exekuce automatizovaného testu Upravení Story

Před spuštěním testu musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné spustit automatizovaný test. Tento scénář se zabývá upravením „Story“.

Jednotlivé kroky testu přesně odpovídají vývojovému diagramu vyobrazenému na obrázku číslo 26. Z tohoto důvodu nebude v této kapitole detailně rozepsán postup jednotlivých kroků. Po úspěšném doběhnutí všech kroků v testu je stav testu „Passed“ a počet chyb roven 0.



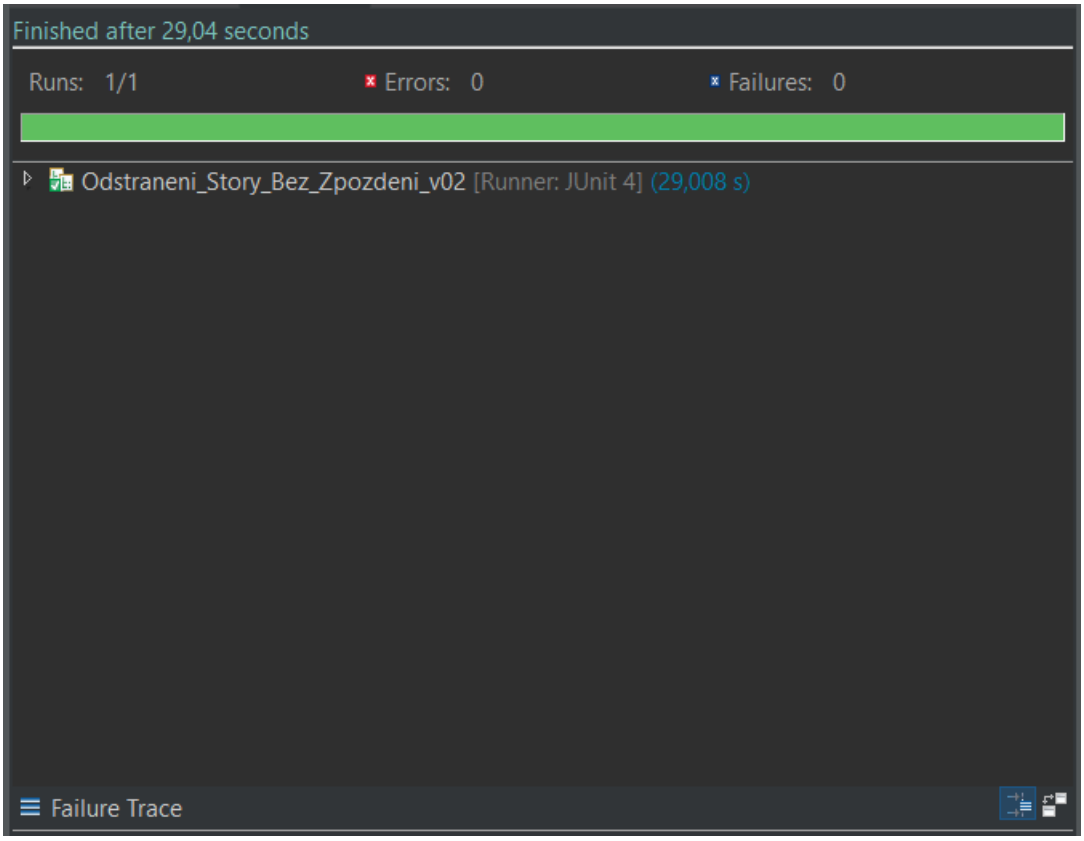
Obrázek 30: Vyhodnocení automatizovaného testu Upravení Story (vlastní zpracování autora)

Na obrázku číslo 30. je po spuštění automatizovaného testu „Upravení Story“ zobrazen celkový čas strávený testováním. Test „Upravení Story“ trval po dobu necelých 31 vteřin.

4.8.4 Exekuce automatizovaného testu Odstranění Story

Před spuštěním testu musí být splněny veškeré předpoklady a vytvořeny testovací data. Po zajištění těchto podmínek je možné spustit automatizovaný test. Tento scénář se zabývá odstraněním „Story“.

Jednotlivé kroky testu přesně odpovídají vývojovému diagramu vyobrazenému na obrázku číslo 27. Z tohoto důvodu nebude v této kapitole detailně rozepsán postup jednotlivých kroků. Po úspěšném doběhnutí všech kroků v testu je stav testu „Passed“ a počet chyb roven 0.



Obrázek 31: Vyhodnocení automatizovaného testu Odstranění Story (vlastní zpracování autora)

Na obrázku číslo 31. je po spuštění automatizovaného testu „Odstranění Story“ zobrazen celkový čas strávený testováním. Test „Odstranění Story“ trval po dobu 29 vteřin.

5 Výsledky a diskuse

Pro určení výhodnosti testů zde budou detailně rozebrány naměřené výsledky manuálních a automatizovaných testů. Nejprve zde budou analyzovány testy manuální, a to jak pro jednotlivé testy, tak pro celou sadu všech 4 testů. Dále zde budou analyzovány výsledky testů automatizovaných, také pro celou sadu a jednotlivé testy.

Hlavní údaj, na který se bude nahlížet s největší prioritou je časové hledisko vytvoření a náročnost na provedení exekuce, poté pracnost z pohledu testera.

V naměřených hodnotách není započten čas strávený analýzou aplikace a její dokumentace.

Po provedení analýzy bude provedeno doporučení, které testy využít pro případné situace.

5.1 Výsledky manuálních testů

V této části bude provedena analýza celé sady manuálních testů, výsledky budou zapsány v hlavní tabulce pro všechny provedené testy, tabulka bude obsahovat časovou náročnost, pracnost a výslednou časovou náročnost.

V následujících podkapitolách budou uvedeny výsledky jednotlivých manuálních testů, které budou podkladem pro hlavní tabulku v této kapitole výsledků manuálních testů. V následují tabulce jsou uvedeny a vypočteny výsledné hodnoty pro manuální testy.

	Časová náročnost pro vytvoření (sekundy)	**Časová náročnost pro exekuci (sekundy)**	**Celková pracnost z pohledu testera**	**Výsledná časová náročnost pro vytvoření a 1 běh testů (sekundy)**	**Výsledná časová náročnost vytvoření a 10 běhů testů (sekundy)**
Průchod částí aplikace – Design v projektu	643	203	střední	846	2 673
Vytvoření nové Story	501	140	malá	641	1 901
Upravení Story	281	145	malá	426	1 731
Odstranění Story	260	94	malá	354	1 200
∑	= 1 685	= 582	malá	= 2 267	= 7 505

Tabulka 1: Výsledná tabulka naměřených hodnot pro sadu manuálních testů (vlastní zpracování autora)

5.1.1 Výsledky testu Průchod částí aplikace – Design v projektu

V následující tabulce jsou hodnoty naměřené pro manuální test „Průchod částí aplikace – Design v Projektu“. Hodnoty mohou být ovlivněny zkušenostmi autora s testováním, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Průchod částí aplikace – Design v projektu	643	203	střední

Tabulka 2: Naměřené hodnoty pro manuální test Průchod částí aplikace – Design v projektu (vlastní zpracování autora)

5.1.2 Výsledky testu Vytvoření nové Story

V následující tabulce jsou hodnoty naměřené pro manuální test „Vytvoření nové Story“. Hodnoty mohou být ovlivněny zkušenostmi autora s testováním, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Vytvoření nové Story	501	140	malá

Tabulka 3: Naměřené hodnoty pro manuální test Vytvoření nové Story (vlastní zpracování autora)

5.1.3 Výsledky testu Upravení Story

V následující tabulce jsou hodnoty naměřené pro manuální test „Upravení Story“. Hodnoty mohou být ovlivněny zkušenostmi autora s testováním, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Upravení Story	281	145	malá

Tabulka 4: Naměřené hodnoty pro manuální test Upravení Story (vlastní zpracování autora)

5.1.4 Výsledky testu Odstranění Story

V následující tabulce jsou hodnoty naměřené pro manuální test „Odstranění Story“. Hodnoty mohou být ovlivněny zkušenostmi autora s testováním, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Odstranění Story	260	94	malá

Tabulka 5: Naměřené hodnoty pro manuální test Odstranění Story (vlastní zpracování autora)

5.2 Výsledky automatizovaných testů

V této části bude provedena analýza naměřených hodnot celé sady automatizovaných testů, výsledky budou zapsány v hlavní tabulce pro všechny provedené testy, tabulka bude obsahovat časovou náročnost, pracnost a výslednou časovou náročnost.

V následujících podkapitolách budou uvedeny výsledky jednotlivých automatizovaných testů, které budou podkladem pro hlavní tabulku v této kapitole výsledků automatizovaných testů. V následují tabulce jsou uvedeny a vypočteny výsledné hodnoty pro automatizované testy.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera	Výsledná časová náročnost pro vytvoření a 1 běh testů (sekundy)	Výsledná časová náročnost pro vytvoření a 10 běhů testů (sekundy)
Průchod částí aplikace – Design v projektu	1 516	36	vysoká	1 552	1 876
Vytvoření nové Story	1 080	33	střední	1 113	1 410
Upravení Story	827	31	střední	858	1 137
Odstranění Story	493	29	malá	522	783
∑	= 3 916	= 129	střední	= 4 045	= 5 206

Tabulka 6: Výsledná tabulka naměřených hodnot pro sadu automatizovaných testů (vlastní zpracování autora)

5.2.1 Výsledky aut. testu Průchod částí aplikace – Design v projektu

V následující tabulce jsou hodnoty naměřené pro automatizovaný test „Průchod částí aplikace – Design v projektu“. Hodnoty mohou být ovlivněny zpožděním odpovědi testované aplikace, nástrojem „Eclipse“ a nastaveným zpožděním ve scriptu z důvodu jeho

úspěšného provedení, rychlostí internetu zkušenostmi autora s vytvářením aut. testů, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Průchod částí aplikace – Design v projektu	1 516	36	vysoká

Tabulka 7: Naměřené hodnoty pro aut. test Průchod částí aplikace – Design v projektu (vlastní zpracování autora)

5.2.2 Výsledky aut. testu Vytvoření nové Story

V následující tabulce jsou hodnoty naměřené pro automatizovaný test „Vytvoření nové Story“. Hodnoty mohou být ovlivněny zpožděním odpovědi testované aplikace, nástrojem „Eclipse“ a nastaveným zpožděním ve scriptu z důvodu jeho úspěšného provedení, rychlostí internetu zkušenostmi autora s vytvářením aut. testů, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Vytvoření nové Story	1 080	33	střední

Tabulka 8: Naměřené hodnoty pro aut. test Vytvoření nové Story (vlastní zpracování autora)

5.2.3 Výsledky aut. testu Upravení Story

V následující tabulce jsou hodnoty naměřené pro automatizovaný test „Upravení Story“. Hodnoty mohou být ovlivněny zpožděním odpovědi testované aplikace, nástrojem „Eclipse“ a nastaveným zpožděním ve scriptu z důvodu jeho úspěšného provedení, rychlostí internetu zkušenostmi autora s vytvářením aut. testů, s testovanou aplikací, a s využitými nástroji.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Upravení Story	827	31	střední

Tabulka 9: Naměřené hodnoty pro aut. test Upravení Story (vlastní zpracování autora)

5.2.4 Výsledky aut. testu Odstranění Story

V následující tabulce jsou hodnoty naměřené pro automatizovaný test „Odstranění Story“. Hodnoty mohou být ovlivněny zpožděním odpovědi testované aplikace, nástrojem „Eclipse“ a nastaveným zpožděním ve scriptu z důvodu jeho úspěšného provedení, rychlostí internetu zkušenostmi autora s vytvářením aut. testů, s testovanou aplikací, a s využitými nástroji.

.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera
Odstranění Story	493	29	malá

Tabulka 10: Naměřené hodnoty pro aut. test Odstranění Story (vlastní zpracování autora)

5.3 Vyhodnocení naměřených hodnot

Po vytvoření a provedené exekuce manuálních a automatizovaných testů vznikla následující tabulka obsahující naměřené hodnoty pro celou sadu 4 testů.

- Průchod částí aplikace – Design v projektu
- Vytvoření nové Story
- Upravení Story
- Odstranění Story

Následující tabulka obsahuje údaje o Časové náročnosti pro vytvoření testů, Časové náročnosti 1 exekuce testů, Celková pracnost z pohledu testera, Výslednou náročnost pro vytvoření testů a 1 exekuci a Výslednou časovou náročnost pro vytvoření testů a 10 exekucí.

	Časová náročnost pro vytvoření (sekundy)	Časová náročnost pro exekuci (sekundy)	Celková pracnost z pohledu testera	Výsledná časová náročnost pro vytvoření a 1 běh testů (sekundy)	Výsledná časová náročnost pro vytvoření a 5 běhů testů (sekundy)	Výsledná časová náročnost pro vytvoření a 10 běhů testů (sekundy)
Manuální testy	1 685	582	malá	2 267	4 595	7 505
Automatizované testy	3 916	129	střední	4 045	4 561	5 206

Tabulka 11: Výsledná tabulka pro porovnání manuálních a automatizovaných testů (vlastní zpracování autora)

Z tabulky lze jasně vyčíst, jaké testy použít. Aby bylo možné zvolit některou variantu, je potřebné se zamyslet, jak se bude k testování aplikace přihlížet v budoucnu, zdali se jedná pouze o 1 kolo testů nebo se testy budou využívat vícekrát.

Z naměřených hodnot je tedy vidět, že pokud budou testy spuštěny pouze 1krát výhodnější po všech stránkách bude možnost manuálních testů s časovou náročností 2 267 vteřin a malou pracností. Což je oproti výsledným hodnotám aut. testů s časovou náročností 4 045 a celkovou střední pracností zhruba 2krát rychlejší a tedy i výhodnější.

Pokud ovšem budou testy využívány opakovaně, výhodnějšími se stanou testy automatizované. Bod zlomu přichází v 5 opakování exekuce testů, zde už se stávají výhodnější automatizované testy. Čím více provedení exekucí testů, tím výhodnější budou automatizované testy. Například pro vytvoření testů a jejich 10 spuštění vzroste rozdílná hodnota v časové náročnosti a výhodnosti automatizovaných testů na 7 505 – 5 206 = **2 299**.

5.4 Doporučení

Na základě předešlé analýzy výsledků z tabulky číslo 11 z kapitoly 5.3 je provedeno následující doporučení na využívání manuálních a automatizovaných testů.

Časová analýza náročnosti sady testů odhalila vhodnost využití automatizovaného testování při alespoň 5 opakování exekuce. Přesně v tomto počtu opakování dochází ke zlomu, kdy se přestávají vyplácet testy manuální, a naopak vyplácet testy automatizované. Při každé další exekuci se zvětšuje časová náročnost manuálních testů, z toho plyne, že čím více exekucí bude, tím více se nám vyplácí automatizované testy.

V naměřených hodnotách pro tyto výpočty nejsou započítány další časové náklady z důvodu údržby testů, ta se provádí, pokud systém provede nějaké změny v oblastech, které testy pokrývají. Údržba se ovšem provádí jak u manuálních, tak automatizovaných testů, tudíž nezapočtení těchto časů nám nijak výrazně měření nezkreslí.

Z výsledných tabulek pro jednotlivé testy je patrné, že ve všech 4 případech se vyplatí využívat automatizované testy při jejich časté exekuci.

Není to však pravidlem a je zde možnost, že jiný test by mohl přinést odlišné hodnoty a být výhodnější v manuální verzi.

Hlavní kritérium pro rozhodnutí vhodnosti využití automatizovaného testování je počet vykonání testů. Počty pro minimální počet exekucí testů jsou již známé z tabulky č. 11, avšak potřeby exekucí závisí na rozhodnutí zodpovědné osoby.

Toto rozhodnutí záleží na tom, jak často se bude měnit verze testované aplikace, pokud se bude jednat o novou aplikaci, je zde vysoká pravděpodobnost nasazování nových verzí potřeby otestování, a tedy vhodnost využití automatizovaných testů. Pokud ovšem se bude jednat o aplikaci, kde bude nasazena například 1 verze do půl roku, automatizované testy se nevyplácí. Z toho lze usoudit, že automatizované testy je vhodné využít pro menší testy jako jsou například smoke nebo regresní testy. Pro složité E2E testy napříč více aplikacemi, není úplně vhodné využití automatizovaných testů z důvodu velké složitosti a malé iterace.

5.5 Diskuse

5.5.1 Problém s celkovým E2E testem a vázaným ID v testované webové aplikaci

Prvotní myšlenkou práce bylo vymyslet a naprogramovat jeden E2E test, který projde celým procesem od přihlášení po otevření projektu vytvoření, upravení, odstranění záznamů a odhlášení. Test byl původně navržen tak, že po jeho doběhnutí odstraní nově vytvořené záznamy, čímž zaručí integritu dat a funkčnost dalších nebo více běhů testů najednou.

Aplikace je ovšem postavena na unikátních ID identifikátorech jednotlivých záznamů, tím pádem není možnost v jednom běhu celého testu zachytit ID nově vytvořeného záznamu a ihned poté ten samý záznam upravit nebo odstranit. Veškeré prvky jako tlačítka, selectboxy a jiné objekty jsou totiž tímto ID vázané.

Z tohoto důvodu na konci běhu E2E testu není možné začistit databázi a odstranit prvky vytvořené tímto testem. Tudíž se nejedná o celkový funkční E2E test. Bylo pouze možné vytvoření nových záznamů, ale jejich úprava a odstranění nelze v testu provést.

Prvním možným řešením bylo rozdělení testu na více částí, respektive více testů.

To znamená, že vznikly testy:

- Na průchod částí aplikace (přihlášení do aplikace, průchod jednotlivých oblastí a odhlášení)
- Na Vytvoření nového záznamu (přihlášení do aplikace, vytvoření záznamu a odhlášení).
- Na Upravení záznamu (přihlášení do aplikace, upravení záznamu a odhlášení).
- Na Odstranění záznamu (přihlášení do aplikace, odstranění záznamu a odhlášení).

Tímto se částečně vyřešil problém se 3. testem na úpravu záznamů. V databázi již byl vytvořený záznam z předešlého testu, jehož prvky obsahovaly konkrétní ID, na který bylo možné tento test namapovat. Tímto se zaručilo, že test vždy doběhl ve stavu Passed.

Problém se 4. testem na odstranění záznamů zůstal i nadále. Opět se vyskytl problém s unikátností ID záznamu, po spuštění testu, sice test na odstranění záznamu doběhl v pořádku, ale pouze při prvním jeho exekuci. Záznam se odstranil a test již nebylo možné znovu spustit se stejně namapovaným ID, jelikož databáze po provedení testu již neobsahovala toto konkrétní ID, které bylo ve scriptu testu namapováno.

Po různých pokusech, jak vyřešit problém s unikátností ID, autora napadlo podívat se na celý problém z jiného úhlu. Řešení je poměrně komplikované a nemusí být vždy vhodné. Ze znalostí z teoretické části o testovacím prostředí je možné vyřešit problém následujícím způsobem.

Tento problém je možné vyřešit vytvořením unikátního testovacího prostředí, které bude sloužit pouze pro účely těchto automatizovaných testů. Toto testovací prostředí bude namapováno na vlastní databázi, která bude obsahovat předem vytvořené záznamy(předpoklady). Pomocí externích nástrojů Jenkins se vytvoří tzv. „Job“. „Job“ bude obsahovat tuto sadu automatizovaných testů a script do databáze.

Řešení se týká problému dat v databázi a jsou možné 2 řešení, buď před každým spuštěním sady testů, nebo ihned po jejím skončení, spustit skript na vrácení tzv.

„rollback“ databáze do výchozího stavu, který obsahoval již vytvořené záznamy (předpoklady) na jejichž ID jsou testy ve scriptu namapovány. Tímto je zaručena připravenost dat pro jednotlivé spouštění exekucí sady testů.

Při tomto řešení je potřeba hledět na několik aspektů a podrobně zanalyzovat, zdali se toto řešení vůbec vyplatí. Z pohledu autora má tohle řešení smysl pouze, pokud jsou vytvořené hromadné automatizované testy ve velkém počtu pro komplexní testování aplikace. Toto řešení má ovšem další výhodu, a to stabilitu prostředí. Z popisu testování víme, že se nevyplatí automatizovat v systému, kde probíhá vývoj či změnové požadavky a není zaručena jeho stabilita. Na tomto prostředí pro automatizované testy bude stále stabilní verze a teprve po dokončení vývoje na developerském prostředí pro vývojáře je možno změnit („update“) verze i na testovacím prostředí pro automaty a provádět tyto testy.

Dalším, ovšem méně praktickým řešením je před každým spuštěním testu ručně měnit ID ve scriptu testu na odstranění. Pokud by měl být test proveden 100krát, je zde nutnost 100krát zjistit a změnit ID ve scriptu tohoto testu.

5.5.2 Nevýhody využití absolutní Xpath

Ve vytvořených scriptech pro automatizované testy jsou využity absolutní Xpathy. Absolutní Xpathy jsou využity z důvodu jejich jednoduchého získání z konzole prohlížeče a jednoduchého vykopírování konkrétního záznamu. Také není nutná znalost tvoření a syntaxe relativních Xpath. Nevýhodou jejich využívání je možnost jejich nefunkčnosti a nutnosti častějších úprav ve scriptech, například při malé úpravě designu aplikace a posunutím některého prvku. Z tohoto důvodu je pro komplikovanější aplikace lepší mít znalost relativní Xpath pro její využití.

5.5.3 Nevyužití žádné konkrétní sofistikované metody pro analýzu vhodnosti

Z důvodu pouze časových kritérií není vhodné využít některou z vícekriteriálních metod pro určení vhodnosti. Hlavní kritérium, podle kterého je výsledná analýza provedena je časová náročnost a pracnost. Z výsledné tabulky automatizovaných a manuálních testů je zřejmé, jaké testy jsou vhodnější pro konkrétní situace. Tato analýza je zdůvodněna v sekci Výsledky, Doporučení a Závěr.

6 Závěr

Teoretická část práce byla zaměřena na problematiku manuálního a automatizovaného testování. Práce rozebrala celý životní cyklus vývoje softwaru a vybrané modely představující různé využití týkající se tohoto tématu, agilní metodiku a její typy. Další část se zaměřila na testování samotné, proč je vůbec testování důležité, jaké jsou jeho cíle a přiblížila téma životní cyklus testování softwaru a jeho fáze, poté rozdělení jednotlivých typu testů podle několika hledisek jako účel a úroveň testování. V poslední části charakterizovala průběh a důvody využití manuálního a automatizovaného způsobů testování.

Praktická část práce obsahuje základní popisy testované aplikace a nástrojů využitých pro vytvoření a exekuci manuálních a automatizovaných testů. Byly zde vytvořeny jednotlivé případy užití, které dále posloužily jako model pro vytvoření testovacích scénářů. V práci byly určeny čtyři případy užití. Manuální testy byly realizovány v nástroji Juno, automatizované testy byly provedeny a vytvořeny pomocí programovacího jazyku Java za pomocí frameworku Selenium WebDriver v nástroji Eclipse IDE. Bylo provedeno měření časové náročnosti na vytvoření jednotlivých manuálních testů a poté dobu trvání jejich exekuce. Posléze bylo provedeno obdobné měření časové náročnosti pro tvorbu a exekuci jednotlivých automatizovaných testů. Hodnoty byly naměřeny pro počty až 10 opakování. Výsledky byly zaevidovány jak pro jednotlivé testy, tak pro celou sadu 4 testů, ze kterých jsou stanoveny celkové výsledky a doporučení. Analýza vhodnosti je provedena z hlediska časové náročnosti, práce tedy poskytuje řešení z pohledu času a způsobu využívání testů s ohledem na počty opakování.

V práci byly navrženy, vytvořeny a provedeny exekuce pro všechny určené testy, byla provedena analýza vhodnosti využití manuálního a automatizovaného způsobu testování na základě časové náročnosti s ohledem na počet opakování. Vypracováním práce a zkoumáním odborných prací autor nabyl nových zkušeností i přes to, že se věnuje testovaní v praxi již několik let.

7 Seznam použitých zdrojů

1. SDLC (Software Development Life Cycle). *Guru99* [online]. [cit. 2020-02-24]. Dostupné z: https://www.guru99.com/software-development-life-cycle-tutorial.html
2. SDLC (Software Development Life Cycle) Phases, Methodologies, Process, And Models. *Software Testing Help* [online]. 10.11.2019 [cit. 2020-02-24]. Dostupné z: https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/
3. What is SDLC? Understand the Software Development Life Cycle. *Stackify* [online]. 06.04.2017 [cit. 2020-02-24]. Dostupné z: https://stackify.com/what-is-sdlc/
4. ALWAN, Motea. What is System Development Life Cycle? *Airbrake* [online]. 09.01.2015 [cit. 2020-02-24]. Dostupné z: https://airbrake.io/blog/sdlc/what-is-system-development-life-cycle
5. ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. Řízení kvality softwaru: průvodce testováním. Brno: Computer Press, 2013. ISBN 978-802-5138-168
6. HLAVA, Tomáš. Modely životního cyklu softwaru. *Testování softwaru* [online]. [cit. 2020-02-25]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/
7. PATTON, Ron. *Testování softwaru*. Praha: Computer Press, 2002. ISBN 978-807-2266-364.
8. SVOZILOVÁ, Alena. *Projektový management: Systémový přístup k řízení projektů*. 3. akt. vyd. Praha: Grada Publishing, 2016. ISBN 978-80-271-0075-0.
9. Agilní metodiky řízení vývoje software. *Management Media* [online]. [cit. 2020-02-26]. Dostupné z: https://managementmania.com/cs/agilni-metodiky-rizeni-vyvoje-software
10. KNESL, Jiří. *Co je Scrum?* [online]. 23.03.2016 [cit. 2020-02-26]. Dostupné z: http://www.knesl.com/co-je-scrum
11. BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. Praha: Grada Publishing, 2005. Management v informační společnosti. ISBN 80-247-1075-7.
12. LUHAN, Igor. Metodika DSDM. *Projektový underground* [online]. 4.12.2019 [cit. 2020-02-26]. Dostupné z: https://mypmi.eu/2019/12/04/metodika-dsdm
13. MANNOVÁ, Božena a Karel VOSÁTKA. *Řízení softwarových projektů*. Praha: Česká technika – nakladatelství ČVUT, 2005. ISBN 80-010-3297-3.
14. STLC – Software Testing Life Cycle Phases & Entry, Exit Criteria. *Guru99* [online]. [cit. 2020-02-27]. Dostupné z: https://www.guru99.com/software-testing-life-cycle.html

15. HAMBLING, Brian a Angelina SAMAROO. *Software testing: An ISEB Intermediate certificate*. BCS Learning & Development Limited, 2009. ISBN 978-1-906124-76-2.
16. Frontend Testing Vs. Backend Testing: What's the Difference? *Guru99* [online]. [cit. 2020-02-27]. Dostupné z: https://www.guru99.com/frontend-testing-vs-backend-testing.html
17. HLAVA, Tomáš. Fáze a úrovně provádění testů. *Testování softwaru* [online]. 21.8.2011 [cit. 2020-02-27]. Dostupné z: http://testovanisoftwaru.cz/category/metodika-testovani/druhy-typy-a-kategorie-testu/
18. Druhy testování v procesu vývoje SW. *Testování software* [online]. [cit. 2020-02-27]. Dostupné z: http://test.swtestovani.cz/index.php?option=com_content&view=article&id=18:druhy-testovani-v-procesu-vyvoje-sw&catid=3:zaklady&Itemid=11
19. End-to-End Test. *Techopedia* [online]. 21.2.2017 [cit. 2020-02-28]. Dostupné z: https://www.techopedia.com/definition/7035/end-to-end-test
20. A. Leitner, I. Ciupa, B. Meyer and M. Howard, "Reconciling Manual and Automated Testing: The AutoTest Experience," *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, Waikoloa, HI, 2007, pp. 261a-261a.

8 Přílohy

Příloha 1: Aut. script testu Průchod částí aplikace – Design v projektu (vlastní zpracování autora) 78
Příloha 2: Aut. script testu Vytvoření nové Story (vlastní zpracování autora) 80
Příloha 3: Aut. script testu Upravení Story (vlastní zpracování autora) 82
Příloha 4: Aut. script Odstranění Story (vlastní zpracování autora) 84

```
package package1;
import static org.junit.Assert.*;
import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import org.openqa.selenium.Keys;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.apache.commons.io.*;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

@SuppressWarnings("unused")
public class Pruchod_Casti_Aplikace_Design_v_Projektu_Bez_Zpozdeni_v02
{
    /* Defining elements used in tests*/
    public static WebDriver driver;
    WebElement username;
    WebElement password;
    WebElement button;
    WebElement storyName;
    String testString;
    double i =  Math.random();
    /* Function called BEFORE testing */
    @Before
    public void setup() throws MalformedURLException {
                File chromeDriver = new File("D:\\workspace\\BP_Automat_Design\\chromedriver.exe");
                System.setProperty("webdriver.chrome.driver", chromeDriver.getAbsolutePath()); // loading Webdriver
from PC
                driver = new ChromeDriver(); // driver instance
      /* Implicit wait - waiting for all elements to load, maximum of 5 seconds */
                                driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
    }
    /* TEST */
    @Test
    public void main() throws InterruptedException, MalformedURLException, IOException
    {
                File srcFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); // Screenshots
       /* First page initialization */
                driver.get("https://training.juno.one/en/admin/login");
                 driver.manage().window().maximize();
         try {
        /* Filling user credentials and clicking the login button, then wait for 2 secs*/
                username = driver.findElement(By.id("frm-loginForm-e_mail"));
                username.sendKeys("agrunt@denevy.eu");
                password = driver.findElement(By.id("frm-loginForm-password"));
                password.sendKeys("Heslo.123");
                button = driver.findElement(By.name("_submit"));
                button.click();
         } catch (Exception e) {
                FileUtils.copyFile(srcFile, new
File("C://Users//Adam//Desktop//Screeny//AutomatyBP//NefukcniLogin.png"));
                JOptionPane.showMessageDialog(new JFrame(), "Nedostupnost aplikace, nebo provolání špatné stránky!!
Systém se nedokázal                                    přihlásit."); // vyskakovací chybová hláška

                driver.quit();
                e.printStackTrace();
         }
        /* Control - Close Welcome window */
                if (driver.findElements(By.xpath("//*[@id=\"welcomeInfo\"]/div/div/div[3]/button")).size() == 1){

                button = driver.findElement(By.xpath("//*[@id='welcomeInfo']/div/div/div[1]/button"));
                button.click();
         }
         /* Assert test for successful login */
                testString =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a/div/div/p")).getText();
                assertTrue(testString.contains("Adam Grunt"));
                Thread.sleep(500);
// ------------------------------ TEST STARTS HERE ---------------------------------------------------------


         /* Click on the "Project" module */
                button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[3]/a/span[2]"));
```

```
                button.click();
        /* Open project*/
                button = driver.findElement(By.xpath("//*[@id='snippet-projectGrid-item-17']/td[5]/a"));
                button.click();
         /* Assert test for successful open the right project*/
                testString = driver.findElement(By.xpath("
//*[@id='wrapper']/div[2]/div[2]/div[2]/div/div/div/h3")).getText();
                assertTrue(testString.contains("BP Automat - Nesahat"));
        /* Open Design*/
                button = driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[1]/h2/ul/li[2]/a"));
                button.click();
        /* Open Folder in Design tab*/
                button = driver.findElement(By.xpath("//*[@id='item-14']/div[1]/div/p/a[3]"));
                button.click();
        /* Open Epic in Design tab*/
                button = driver.findElement(By.xpath("//*[@id='snippet-defaultGrido-14-item-
42_Requirement']/div[1]/div[2]/div[1]/div[1]/a"));
                button.click();
// ------------------------------ Story ----------------------------------------------------------
        /* Open Story in Epic*/
                button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[2]/div[1]/div/ul/li[2]/a"));
                button.click();
        /* Open Story detail in Epic*/
                button = driver.findElement(By.xpath("//*[@id='snippet-childrenGrid-item-43']/td[2]/a"));
                button.click();
        /* Back from Story detail to Epic */
                button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div/div[1]/div/div/div/div/div/a[1]"));
                button.click();
// ------------------------------ TEST COVERAGE ----------------------------------------------------
        /* Open Test Coverage in Epic*/
                button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[2]/div/div/ul/li[3]/a"));
                button.click();
        /* Show Test Coverage in Epic*/
                button = driver.findElement(By.xpath("//*[@id='snippet-testCasesGrid-item-44']/td[7]/div/a"));
                button.click();
        /* Back to Test Coverage in Epic*/
                button = driver.findElement(By.xpath("//*[@id='a']/div/div[1]/h4/a"));
                button.click();
// ------------------------------ History ----------------------------------------------------------
        /* Open History in Epic*/
                button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[2]/div/div/ul/li[7]/a"));
                button.click();
// ------------------------------ Homepage ----------------------------------------------------------
        /* Open Homepage in Epic*/
                button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[1]/a"));
                button.click();
// ------------------------------ Logout ----------------------------------------------------------
        /* Open Logout Menu in Epic*/
                button = driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a"));
                button.click();
        /* Logout*/
                button =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/ul/li[7]/a"));
                button.click();
        /* Assert test for successful open the right project*/
                testString = driver.findElement(By.xpath("/html/body/div[2]/div/div/div[2]/div/a")).getText();
                assertTrue(testString.contains("www.denevy.eu"));
                System.out.println("Test proběhl v pořádku.");

    }
    /* Function called AFTER testing */
    @After
    public void teardown() throws InterruptedException
    {
        /* Wait 3 seconds and then close the browser*/
                Thread.sleep(3000);
                driver.quit();
    }
}
```

Příloha 1: Aut. script testu Průchod částí aplikace – Design v projektu (vlastní zpracování autora)

```
package package1;
import static org.junit.Assert.*;
import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.Keys;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

@SuppressWarnings("unused")
public class Vytvoreni_Nove_Story_Bez_Zpozdeni_v02
{
    /* Defining elements used in tests*/
    public static WebDriver driver;
    WebElement username;
    WebElement password;
    WebElement button;
    WebElement storyName;
    WebElement selectBox;
    String testString;
    double i =  Math.random();
    /* Function called BEFORE testing */
    @Before
    public void setup() throws MalformedURLException {
                File chromeDriver = new File("D:\\workspace\\BP_Automat_Design\\chromedriver.exe");
                System.setProperty("webdriver.chrome.driver", chromeDriver.getAbsolutePath()); // loading Webdriver
from PC
                driver = new ChromeDriver(); // driver instance
      /* Implicit wait - waiting for all elements to load, maximum of 5 seconds */
                driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
    }
    /* TEST */
    @Test
    public void main() throws InterruptedException, MalformedURLException
    {
        /* First page initialization */
                driver.get("https://training.juno.one/en/admin/login");
                driver.manage().window().maximize();
        /* Filling  user credentials and clicking the login button, then wait for 2 secs*/
                username = driver.findElement(By.id("frm-loginForm-e_mail"));
                username.sendKeys("agrunt@denevy.eu");
                password = driver.findElement(By.id("frm-loginForm-password"));
                password.sendKeys("Heslo.123");
                button = driver.findElement(By.name("_submit"));
                button.click();
        /* Control - Close Welcome window */
                if (driver.findElements(By.xpath("//*[@id=\"welcomeInfo\"]/div/div/div[3]/button")).size() == 1){

                button = driver.findElement(By.xpath("//*[@id='welcomeInfo']/div/div/div[1]/button"));
                button.click();
        }
        /* Assert test for successful login */
                testString =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a/div/div/p")).getText();
                assertTrue(testString.contains("Adam Grunt"));
                Thread.sleep(500);
// ------------------------------ TEST STARTS HERE -----------------------------------------------------------

        /* Click on the "Project" module */
                button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[3]/a/span[2]"));
                button.click();
        /* Open project*/
                button = driver.findElement(By.xpath("//*[@id='snippet-projectGrid-item-17']/td[5]/a"));
                button.click();
         /* Assert test for successful open the right project*/
                testString = driver.findElement(By.xpath("
//*[@id='wrapper']/div[2]/div[2]/div[2]/div/div/div/h3")).getText();
                assertTrue(testString.contains("BP Automat - Nesahat"));
        /* Open Design*/
                button = driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[1]/h2/ul/li[2]/a"));
                button.click();
        /* Open Folder in Design tab*/
                button = driver.findElement(By.xpath("//*[@id='item-14']/div[1]/div/p/a[3]"));
                button.click();
        /* Open Epic in Design tab*/
                button = driver.findElement(By.xpath("//*[@id='snippet-defaultGrido-14-item-
                52_Requirement']/div[1]/div[2]/div[1]/div[1]/a"));
```

```
                button.click();
        /* Open Story in Epic*/
                button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[2]/div[1]/div/ul/li[2]/a"));
                button.click();
// ------------------------------ Create new Story ----------------------------------------------------------

        /* Create new story in Epic */
                button = driver.findElement(By.xpath("//*[@id='a']/div/div[1]/div[2]/p/a[1]"));
                button.click();
        /* Add Name to new story in Epic*/
                storyName = driver.findElement(By.xpath("//*[@id='frm-requirementForm-name']"));
                storyName.sendKeys("A" + i);
        /* Click on status field */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-status']"));
                selectBox.click();
        /* Choose status NEW */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-status']/option[2]"));
                selectBox.click();
        /* Click on Owner field */
                selectBox = driver.findElement(By.xpath("//*[@id='select2-frm-requirementForm-owner-container']"));
                selectBox.click();
        /* Choose Owner Adam Grunt */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-owner']/option[3]"));
                selectBox.click();
        /* Click on Author field */
                selectBox = driver.findElement(By.xpath("//*[@id='select2-frm-requirementForm-author-
container']"));
                selectBox.click();
        /* Choose Author Adam Grunt */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-author']/option[3]"));
                selectBox.click();
        /* Submit new story */
                button = driver.findElement(By.xpath("//*[@id='add-
requirement']/div/div/div[1]/div/div/div/div/div/button"));
                button.click();
// ------------------------------ Homepage ----------------------------------------------------------
        /* Open Homepage in Epic*/
                button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[1]/a"));
                button.click();
// ------------------------------ Logout ----------------------------------------------------------
        /* Open Logout Menu in Epic*/
                button = driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a"));
                button.click();
        /* Logout*/
                button =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/ul/li[7]/a"));
                button.click();
        /* Assert test for successful open the right project*/
                testString = driver.findElement(By.xpath("/html/body/div[2]/div/div/div[2]/div/a")).getText();
                assertTrue(testString.contains("www.denevy.eu"));
                System.out.println("Test proběhl v pořádku.");
    }
        /* Function called AFTER testing */
    @After
    public void teardown() throws InterruptedException
    {
        /* Wait 3 seconds and then close the browser*/
                 Thread.sleep(500);
                driver.quit();
    }
}
```

Příloha 2: Aut. script testu Vytvoření nové Story (vlastní zpracování autora)

```java
package package1;
import static org.junit.Assert.*;
import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.Keys;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

@SuppressWarnings("unused")
public class Upraveni_Story_Bez_Zpozdeni_v02
{
    /* Defining elements used in tests*/
    public static WebDriver driver;
    WebElement username;
    WebElement password;
    WebElement button;
    WebElement storyName;
    WebElement selectBox;
    String testString;
    double i =  Math.random();
    /* Function called BEFORE testing */
    @Before
    public void setup() throws MalformedURLException {
                    File chromeDriver = new File("D:\\workspace\\BP_Automat_Design\\chromedriver.exe");
                    System.setProperty("webdriver.chrome.driver", chromeDriver.getAbsolutePath()); // loading Webdriver
from PC
                    driver = new ChromeDriver(); // driver instance
      /* Implicit wait - waiting for all elements to load, maximum of 5 seconds */
                     driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
    }
    /* TEST */
    @Test
    public void main() throws InterruptedException, MalformedURLException
    {
       /* First page initialization */
                     driver.get("https://training.juno.one/en/admin/login");
                     driver.manage().window().maximize();
                    JavascriptExecutor js = (JavascriptExecutor) driver;
        /* Filling  user credentials and clicking the login button, then wait for 2 secs*/
                    username = driver.findElement(By.id("frm-loginForm-e_mail"));
                    username.sendKeys("agrunt@denevy.eu");
                    password = driver.findElement(By.id("frm-loginForm-password"));
                    password.sendKeys("Heslo.123");
                    button = driver.findElement(By.name("_submit"));
                    button.click();
        /* Control - Close Welcome window */
         if (driver.findElements(By.xpath("//*[@id=\"welcomeInfo\"]/div/div/div[3]/button")).size() == 1){

                    button = driver.findElement(By.xpath("//*[@id='welcomeInfo']/div/div/div[1]/button"));
                    button.click();
         }
         /* Assert test for successful login */
                     testString =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a/div/div/p")).getText();
                    assertTrue(testString.contains("Adam Grunt"));
                    Thread.sleep(500);
// ------------------------------ TEST STARTS HERE ---------------------------------------------------------

         /* Click on the "Project" module */
                    button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[3]/a/span[2]"));
                    button.click();
         /* Open project*/
                    button = driver.findElement(By.xpath("//*[@id='snippet-projectGrid-item-17']/td[5]/a"));
                    button.click();
         /* Assert test for successful open the right project*/
                     testString = driver.findElement(By.xpath("
//*[@id='wrapper']/div[2]/div[2]/div[2]/div/div/div/h3")).getText();
                    assertTrue(testString.contains("BP Automat - Nesahat"));
         /* Open Design*/
                    button = driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[1]/h2/ul/li[2]/a"));
                    button.click();
         /* Open Folder in Design tab*/
                    button = driver.findElement(By.xpath("//*[@id='item-14']/div[1]/div/p/a[3]"));
                    button.click();
         /* Open Epic in Design tab*/
```

```
                button = driver.findElement(By.xpath("//*[@id='snippet-defaultGrido-14-item-
42_Requirement']/div[1]/div[2]/div[1]/div[1]/a"));
                button.click();
        /* Open Story in Epic*/
                button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[2]/div[1]/div/ul/li[2]/a"));
                button.click();
// ------------------------------ Edit Story ------------------------------------------------------

        /* Edit story in Epic */
                button = driver.findElement(By.xpath("//*[@id='snippet-childrenGrid-item-43']/td[7]/div/a[2]"));
                button.click();
        /* Edit Name to new story in Epic*/
                storyName = driver.findElement(By.xpath("//*[@id='frm-requirementForm-name']"));
                storyName.clear();
                storyName.sendKeys("Edit" + i);
        /* Click on type of change */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-typeOfChange']"));
                selectBox.click();
        /* Choose type of change - intermediate */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-typeOfChange']/option[2]"));
                selectBox.click();
        /* Click on status field */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-status']"));
                selectBox.click();
        /* Choose status NEW */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-status']/option[3]"));
                selectBox.click();
                js.executeScript("window.scrollBy(0,-250)"); // scroll page up
        /* Click on Owner field */
                selectBox = driver.findElement(By.xpath("//*[@id='select2-frm-requirementForm-owner-container']"));
                selectBox.click();
        /* Choose Owner Kolarik Lubomir */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-owner']/option[5]"));
                selectBox.click();
        /* Click on Author field */
                selectBox = driver.findElement(By.xpath("//*[@id='select2-frm-requirementForm-author-
container']"));
                selectBox.click();
        /* Choose Author Vesela Katerina */
                selectBox = driver.findElement(By.xpath("//*[@id='frm-requirementForm-author']/option[6]"));
                selectBox.click();
        /* Edit story */
                button = driver.findElement(By.xpath("//*[@id='add-
requirement']/div/div/div[1]/div/div/div/div/div/button"));
                button.click();
// ------------------------------ Homepage ------------------------------------------------------
        /* Open Homepage in Epic*/
                button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[1]/a"));
                button.click();
// ------------------------------ Logout ------------------------------------------------------
        /* Open Logout Menu in Epic*/
                button = driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a"));
                button.click();
        /* Logout*/
                button =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/ul/li[7]/a"));
                button.click();
        /* Assert test for successful open the right project*/
                testString = driver.findElement(By.xpath("/html/body/div[2]/div/div/div[2]/div/a")).getText();
                assertTrue(testString.contains("www.denevy.eu"));
                System.out.println("Test proběhl v pořádku.");
    }
        /* Function called AFTER testing */
    @After
    public void teardown() throws InterruptedException
    {
        /* Wait 3 seconds and then close the browser*/
                Thread.sleep(500);
                driver.quit();
    }
}
```

Příloha 3: Aut. script testu Upravení Story (vlastní zpracování autora)

```
package package1;
import static org.junit.Assert.*;
import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.Keys;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

@SuppressWarnings("unused")
public class Odstraneni_Story_Bez_Zpozdeni_v02
{
    /* Defining elements used in tests*/
    public static WebDriver driver;
    WebElement username;
    WebElement password;
    WebElement button;
    WebElement storyName;
    String testString;
    double i =  Math.random();
    /* Function called BEFORE testing */
    @Before
    public void setup() throws MalformedURLException {
                File chromeDriver = new File("D:\\workspace\\BP_Automat_Design\\chromedriver.exe");
                System.setProperty("webdriver.chrome.driver", chromeDriver.getAbsolutePath()); // loading Webdriver
from PC
                driver = new ChromeDriver(); // driver instance
      /* Implicit wait - waiting for all elements to load, maximum of 5 seconds */
      driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
    }
    /* TEST */
    @Test
    public void main() throws InterruptedException, MalformedURLException
    {
       /* First page initialization */
                driver.get("https://training.juno.one/en/admin/login");
                driver.manage().window().maximize();
        /* Filling  user credentials and clicking the login button, then wait for 2 secs*/
                username = driver.findElement(By.id("frm-loginForm-e_mail"));
                username.sendKeys("agrunt@denevy.eu");
                password = driver.findElement(By.id("frm-loginForm-password"));
                password.sendKeys("Heslo.123");
                button = driver.findElement(By.name("_submit"));
                button.click();
        /* Control - Close Welcome window */
                if (driver.findElements(By.xpath("//*[@id=\"welcomeInfo\"]/div/div/div[3]/button")).size() == 1){
                button = driver.findElement(By.xpath("//*[@id='welcomeInfo']/div/div/div[1]/button"));
                button.click();
         }
         /* Assert test for successful login */
                testString =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a/div/div/p")).getText();
                assertTrue(testString.contains("Adam Grunt"));
                Thread.sleep(500);
// ------------------------------ TEST STARTS HERE -----------------------------------------------------------
         /* Click on the "Project" module */
                button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[3]/a/span[2]"));
                button.click();
         /* Open project*/
                button = driver.findElement(By.xpath("//*[@id='snippet-projectGrid-item-17']/td[5]/a"));
                button.click();
         /* Assert test for successful open the right project*/
         testString = driver.findElement(By.xpath("
//*[@id='wrapper']/div[2]/div[2]/div[2]/div/div/div/h3")).getText();
                 assertTrue(testString.contains("BP Automat - Nesahat"));
        /* Open Design*/
                button = driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[1]/h2/ul/li[2]/a"));
                button.click();
         /* Open Folder in Design tab*/
                button = driver.findElement(By.xpath("//*[@id='item-14']/div[1]/div/p/a[3]"));
                button.click();
         /* Open Epic in Design tab*/
                button = driver.findElement(By.xpath("//*[@id='snippet-defaultGrido-14-item-
42_Requirement']/div[1]/div[2]/div[1]/div[1]/a"));
                button.click();
         /* Open Story in Epic*/
```

```
                    button =
driver.findElement(By.xpath("//*[@id='wrapper']/div[2]/div[2]/div[2]/div[1]/div/ul/li[2]/a"));
                    button.click();
// ------------------------------ Delete Story ------------------------------------------------------------
          /* Delete  Story from Epic */
                    button = driver.findElement(By.xpath("//*[@id='snippet-childrenGrid-item-67']/td[7]/div/button"));
                    button.click();
                    button = driver.findElement(By.xpath("//*[@id='deleteAction']/div/div/div[3]/a"));
                    button.click();
// ------------------------------ Homepage ----------------------------------------------------------------
          /* Open Homepage in Epic*/
                    button = driver.findElement(By.xpath("//*[@id='mCSB_1_container']/ul/li[1]/a"));
                    button.click();
// ------------------------------ Logout ------------------------------------------------------------------
          /* Open Logout Menu in Epic*/
                    button = driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/a"));
                    button.click();
          /* Logout*/
                    button =
driver.findElement(By.xpath("//*[@id='body']/div[2]/div/div[2]/div/div[2]/ul/li/ul/li[7]/a"));
                    button.click();
          /* Assert test for successful open the right project*/
                    testString = driver.findElement(By.xpath("/html/body/div[2]/div/div/div[2]/div/a")).getText();
                    assertTrue(testString.contains("www.denevy.eu"));
                    System.out.println("Test proběhl v pořádku.");
     }
     /* Function called AFTER testing */
     @After
     public void teardown() throws InterruptedException
     {
          /* Wait 3 seconds and then close the browser*/
                    Thread.sleep(500);
                    driver.quit();
     }
}
```

Příloha 4: Aut. script Odstranění Story (vlastní zpracování autora)