

# **Webová aplikace pro kompletní správu automatizovaných testů v agilním prostředí**

Bc. Ondřej Jakuba

---

Diplomová práce  
2019



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2018/2019

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Ondřej Jakuba**

Osobní číslo: **A16197**

Studijní program: **N3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Forma studia: **kombinovaná**

Téma práce: **Webová aplikace pro kompletní správu automatizovaných testů v agilním prostředí**

Téma anglicky: **A Web Application for the End-to-End Management of Automation Testing in Agile Environments**

Zásady pro vypracování:

1. Nastudujte danou problematiku související s automatizovanými testy a frameworky.
2. Vyberte vhodné metody a nástroje pro realizaci webové aplikace pro spouštění automatizovaných testů.
3. Vytvořte design a architekturu webové aplikace.
4. Implementujte webovou aplikaci za využití vámi vybraných metod a nástrojů.
5. Provéďte test aplikace a vhodně reprezentujte získané výsledky.



Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. SPAANJAARS, Imar. Beginning ASP.NET 4.5 in C# and VB. Indianapolis, IN: Wiley, c2013. Wrox beginning guides. ISBN 9781118311806
2. ŠOCHOVÁ, Zuzana a Eduard KUNCE. Agilní metody řízení projektů. 1. Environment. Brno: Computer Press, 2014, 175 s. ISBN 978-80-251-4194-6.
3. MYSLÍN, Josef. Scrum: průvodce agilním vývojem softwaru. 1. vydání. Brno: Computer Press, 2016, 167 s. ISBN 978-80-251-4650-7.
4. HUGGINS, Jason, Gheorghe Gheorghiu, C. Titus BROWN, An Introduction to Testing Web Applications with twill and Selenium
5. LUIS, Feroz Pearl, Gaurav Gupta, Mastering Mobile Test Automation

Vedoucí diplomové práce:

**Ing. Petr Žáček**

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

**26. července 2019**

Termín odevzdání diplomové práce:

**26. srpna 2019**

Ve Zlíně dne 2. srpna 2019

doc. Mgr. Milan Adámek, Ph.D.  
*děkan*



prof. Mgr. Roman Jašek, Ph.D.  
*ředitel ústavu*

## ABSTRAKT

Testování software neodmyslitelně patří k jeho samotnému vývoji. Automatizace testů v agilním vývoji velmi často šetří prostředky, nicméně někteří testéři tuto automatizaci nemusí zvládat snadno. Zvládají však většinou napsat vhodný testovací scénář.

Pro implementaci webové aplikace bylo využito ASP.NET MVC 5. Testovací aplikace byla napsána v .NET Framework 4.5, ve které byl použit nástroj Selenium.

Webová část slouží k tvorbě testů, zatím co testovací aplikace slouží ke spuštění testů na prostředí, ve kterém je spuštěna. Vytvořená aplikace pomůže testerům při správě svých testů. Je možné si testy také spouštět.

Klíčová slova:

Agilní vývoj, Testování, Selenium, ASP.NET, MVC, C#

## ABSTRACT

Software testing is inherent to software development. Test automation saves resources in agile development very often. However some testers are not able to create automation testing. They can usually write a suitable test scenario.

Web application is implemented in ASP.NET MVC 5. Test application is written in .NET Framework 4.5 and it use the Selenium tool.

Web application is using to creating tests. Test application is used to run tests on environment where the application is started. Created application can help to testers with test control. It is possible to run tests.

Keywords:

Agile development, Testing, Selenium, ASP.NET, MVC, C#

Nejprve bych chtěl poděkovat svému vedoucímu práce, kterým je Ing. Petr Žáček, za to, že mi umožnil pracovat na této diplomové práci. Dále bych chtěl poděkovat své rodině, která mi byla oporou po celé studium. Poděkování si zaslouží také BcA. Lívia Jančíková za pomoc s vytvořením ikoněk, které byly v této práci použity a také za to, že při mně celé studium stála. Další poděkování patří firmě CN Group CZ s.r.o., kde jsem získal spoustu cenných zkušeností a znalostí na pozici testera a umožnila mi získat certifikaci ISTQB.

Práce slouží čtenáři k získání několika informací o testování software. Lze se dovědět, jak lze přistupovat k vývoji SW a také poznát různé druhy testování, které se v praxi běžně používají. Téma testování jsem si vybral, protože jsem byl 3 roky zaměstnán jako inženýr testování software a získal jsem certifikáty ISTQB CTFL a ISTQB Agile Tester. V teorii se čtenář doví informace, o kterých by měl mít přehled každý tester a v praktické části se seznámí s návrhem a provedením aplikace, ve které bylo cílem usnadnit práci testerům, při tvorbě automatizovaných testů.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen přípouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 27.8.2019

.....  
podpis diplomanta

**OBSAH**

|  |           |
|--|-----------|
| <b>ÚVOD.....</b>   | <b>10</b> |
| <b>I TEORETICKÁ ČÁST.....</b>  | <b>12</b> |
| <b>1 VÝVOJ SOFTWARE .....</b>  | <b>13</b> |
| 1.1 PŘÍSTUP K VÝVOJI SW .....  | 13        |
| 1.1.1 Vodopádový model .....   | 13        |
| 1.1.2 Prototypový model .....  | 14        |
| 1.1.3 Inkrementální model .....  | 15        |
| 1.1.4 Spirálový model .....  | 16        |
| 1.1.5 Rapid Application Development (RAD).....                           | 17        |
| 1.1.6 Agilní vývoj .....   | 17        |
| <b>2 TESTOVÁNÍ .....</b>   | <b>19</b> |
| 2.1 WHITE/BLACK BOX.....   | 19        |
| 2.2 ÚROVNĚ TESTOVÁNÍ.....  | 19        |
| 2.2.1 UNIT testy.....  | 20        |
| 2.2.2 Integrované testy .....  | 20        |
| 2.2.3 Systémové testování.....   | 20        |
| 2.2.4 Akceptační testování .....   | 21        |
| 2.3 MANUÁLNÍ TESTOVÁNÍ .....   | 22        |
| 2.3.1 Exploratory testing .....  | 22        |
| 2.3.2 Usability testing .....  | 22        |
| 2.3.3 Ad-hoc testing .....   | 22        |
| 2.4 AUTOMATIZOVANÉ TESTOVÁNÍ.....  | 23        |
| 2.4.1 Regresní testování .....   | 23        |
| 2.4.2 Zátěžové testy.....  | 23        |
| 2.4.3 Testování výkonu .....   | 23        |
| <b>3 TESTOVÁNÍ V AGILNÍM PROSTŘEDÍ.....</b>                              | <b>24</b> |
| 3.1 SPOLUPRÁCE NA TVORBĚ UŽIVATELSKÝCH SCÉNÁŘŮ.....                      | 24        |
| 3.2 RETROSPEKTIVA.....   | 24        |
| 3.3 CONTINUOUS INTEGRATION .....   | 24        |
| <b>4 NÁSTROJE PRO AUTOMATIZACI TESTOVÁNÍ UŽIVATELSKÉHO ROZHRANÍ.....</b> | <b>26</b> |
| 4.1 TEST AUTOMATION FRAMEWORK .....                                      | 26        |
| 4.1.1 Lineární Skriptovací Framework .....                               | 26        |
| 4.1.2 Modální testovací Framework.....                                   | 26        |
| 4.1.3 Data-driven Framework .....  | 27        |
| 4.1.4 Framework pro testování klíčovými slovy.....                       | 27        |
| 4.1.5 Hybrid Driven Testing Framework .....                              | 27        |
| 4.1.6 Framework pro chování řízený vývoj testování.....                  | 27        |
| 4.2 POUŽÍVANÉ NÁSTROJE PRO AUTOMATIZOVANÉ TESTOVÁNÍ WEBU .....           | 28        |
| 4.2.1 Selenium.....  | 28        |
| 4.2.2 Katalon Studio.....  | 28        |

|                    |  |           |
|--------------------|--|-----------|
| 4.2.3              | Unified Functional Testing (UFT) .....     | 28        |
| 4.2.4              | TestComplete .....                         | 29        |
| 4.2.5              | SoapUI.....                                | 29        |
| 4.2.6              | Robot Framework.....                       | 29        |
| 4.2.7              | Ranorex .....                              | 29        |
| <b>II</b>          | <b>PRAKTICKÁ ČÁST .....</b>                | <b>30</b> |
| <b>5</b>           | <b>VÝBĚR NÁSTROJŮ A METOD .....</b>        | <b>31</b> |
| 5.1                | WEBOVÁ APLIKACE.....                       | 31        |
| 5.1.1              | ASP.NET.....                               | 31        |
| 5.1.2              | ASP.NET MVC Framework .....                | 32        |
| 5.2                | TESTOVACÍ FRAMEWORK .....                  | 33        |
| <b>6</b>           | <b>NÁVRH APLIKACE .....</b>                | <b>34</b> |
| 6.1                | IDENTIFIKACE UŽIVATELE .....               | 34        |
| 6.1.1              | Role .....                                 | 34        |
| 6.1.2              | Přihlášení a registrace .....              | 34        |
| 6.2                | NÁVRH STRUKTURY .....                      | 34        |
| 6.2.1              | Tvorba testů.....                          | 35        |
| 6.2.2              | Elementy .....                             | 35        |
| 6.2.3              | Testovací logy .....                       | 36        |
| 6.2.4              | Ukládání dat .....                         | 36        |
| 6.2.5              | Spouštění testů .....                      | 37        |
| 6.3                | DESIGN.....                                | 38        |
| <b>7</b>           | <b>IMPLEMENTACE APLIKACE .....</b>         | <b>39</b> |
| 7.1                | PROSTŘEDÍ PRO VÝVOJ .....                  | 39        |
| 7.2                | VYTVOŘENÍ PROJEKTU .....                   | 39        |
| 7.3                | INSTALACE BALÍČKŮ .....                    | 40        |
| 7.4                | MODELÝ.....                                | 41        |
| 7.5                | KONTROLÉRY .....                           | 42        |
| 7.6                | VIEW .....                                 | 45        |
| 7.7                | UPDATE DATABÁZE .....                      | 46        |
| <b>8</b>           | <b>NASAZENÍ APLIKACE .....</b>             | <b>48</b> |
| 8.1                | PARAMETRY SERVERU .....                    | 48        |
| 8.2                | ZVEŘEJNĚNÍ APLIKACE .....                  | 48        |
| <b>9</b>           | <b>UKÁZKA APLIKACE .....</b>               | <b>52</b> |
| 9.1                | ÚVODNÍ STRÁNKA .....                       | 52        |
| 9.2                | VYTVOŘENÍ TESTU .....                      | 52        |
| 9.3                | SPOUŠTĚNÍ A VÝSLEDEK TESTŮ .....           | 54        |
| 9.4                | PŘIDÁNÍ DALŠÍHO UŽIVATELE DO PROJEKTU..... | 55        |
| <b>ZÁVĚR .....</b> | <b>57</b>                                  |           |

---

|  |           |
|--|-----------|
| <b>SEZNAM POUŽITÉ LITERATURY.....</b>          | <b>58</b> |
| <b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b> | <b>62</b> |
| <b>SEZNAM OBRÁZKŮ .....</b>                    | <b>63</b> |
| <b>SEZNAM PŘÍLOH.....</b>                      | <b>64</b> |

## ÚVOD

V této diplomové práci je rozbor problematiky testování v agilním prostředí. Jde o téma v dnešní době velmi důležité, neboť mnoho firem pochopilo, že kvalita software je velice důležitá a dodávání kvalitního software zajišťuje vývojové firmě důležitou prestiž. Tomu odpovídá i trh práce, kde poptávka po testerech, kteří mají odpovídající profil je stále vysoká. A co víc? Dnešní doba přímo vyžaduje nespočet aplikací. Dostupnost k informačním technologiím je dnes natolik vysoká, že i starší lidé jsou schopni vyhledávat informace na internetu. Lidé stále více využívají webové služby k nákupům a jiným činnostem, neboť v tom sami pociťují jistou pohodlnost. Tato skutečnost tak tlačí především firmy, které chtějí být úspěšné k tomu, aby si nechali vytvořit také webové aplikace. Z tohoto důvodu se tedy v oboru IT protočí poměrně vysoká část financí.

Z pracovních zkušeností lze říci, že výrazná část testerů má problém s psaním kódu, což je omezovalo na manuální testování. Někteří možná jen nedostali příležitost, jiní mají problém s pochopením pro kódování a je pro ně náročné psát vlastní kusy kódu. Na základě tohoto problému vznikl nápad vytvořit něco, co by jim ve své práci pomohlo a bylo tak užitečné.

Někdo může namítnout, že již existuje nespočet možností na internetu a že je zbytečná snaha vytvářet něco, co už existuje. Tyto nástroje by však mohli být zastíněny jiným vhodnějším řešením. Mnoho dostupných nástrojů je především finančně nedostupné pro menší firmy. Některé nástroje mají své nedostatky, které brání uživateli být více produktivní a tak dále. Tyto nástroje pravděpodobně nejsou špatně navržené, nebo vytvořené, jen existuje několik způsobů přístupu, jakým lze testy vytvářet. Odborníci na téma tvorby testů ve svých pracích představují kvalitní doporučení, kterých se držet při tvorbě testovacích scénářů i ohledně výběru vhodného nástroje pro samotnou automatizaci. Je tedy třeba myslet na to, že každý projekt, na kterém může tester spolupracovat, je svým způsobem jedinečný a je třeba uvážit, zda nástroj, který se osvědčil na jednom projektu, bude vhodný i pro projekt další. Nejlepší volba je tedy velice individuální.

V první části je popis několika druhů přístupů k vývoji software. U těchto modelů popisují výhody a hlavní myšlenku. Nejvíce je pak popsán model pro agilní vývoj, neboť k tomuto modelu míří tato diplomová práce.

V druhé části je pojednáváno o testování software. Je zde vysvětleno, jaké přístupy k testování jsou používány. U testovacích úrovní popisují jejich význam a použití. Je zde

snaha poukázat i na chyby, které vznikají v této souvislosti. Také je zde uvedeno, kdo testování v jednotlivých úrovních provádí. Poté jsou zde uvedeny podmínky, kdy použít manuální a automatizované testování.

Ve třetí části je popis testování v agilním prostředí. Konkrétně tvorba uživatelských příběhů a důležitost role testera při jejich tvorbě, důvody k pravidelné retrospektivě a také výhody použití automatizovaných testů pro continuous integration.

Čtvrtá část se zabývá nástroji pro automatizaci testování. Je zde popis důvodů pro použití frameworků, jaké frameworky existují a k čemu slouží. Také jsou zde uvedeny některé existující nástroje, které lze použít k vytvoření automatizovaných testů.

Páté část se věnuje výběru nástrojů a metod. Především pak odůvodnění výběru technologie pro tuto diplomovou práci a popisem, jak pracuje.

V šesté části je popis vlastního návrhu aplikace. Jsou zde nastíněny aspekty, se kterými bylo pracováno a vysvětleny důvody k vytvoření návrhu, tak aby návrh odpovídal skutečným požadavkům. Jde především o prvky, které je nutné použít a které popisují data, která bude aplikace shromažďovat. Také pak návrh designu.

Sedmá část se již zabývá samotnou implementací aplikace. Popisuje důležité kroky, které jsou nezbytné pro vytvoření aplikace způsobem, který byl zvolen. Pomocí diagramů je popsána architektura aplikace. Mimo to je zde poukázáno na způsob tvorby aplikace a kroky, které vedly k úspěšnému dokončení aplikace.

Osmá část je již ukázkou výsledné aplikace. Popisuje postup pro práci s aplikací.

V rámci této diplomové práce měly být vyřešeny tyto dílčí úkoly:

- Nastudujte danou problematiku související s automatizovanými testy a frameworky
- Vyberte vhodné metody a nástroje pro realizaci webové aplikace pro spouštění automatizovaných testů
- Vytvořte design a architekturu webové aplikace
- Implementujte webovou aplikaci za využití vámi vybraných metod a nástrojů
- Proveďte test aplikace a vhodně reprezentujte získané výsledky.

## **I. TEORETICKÁ ČÁST**

## 1 VÝVOJ SOFTWARE

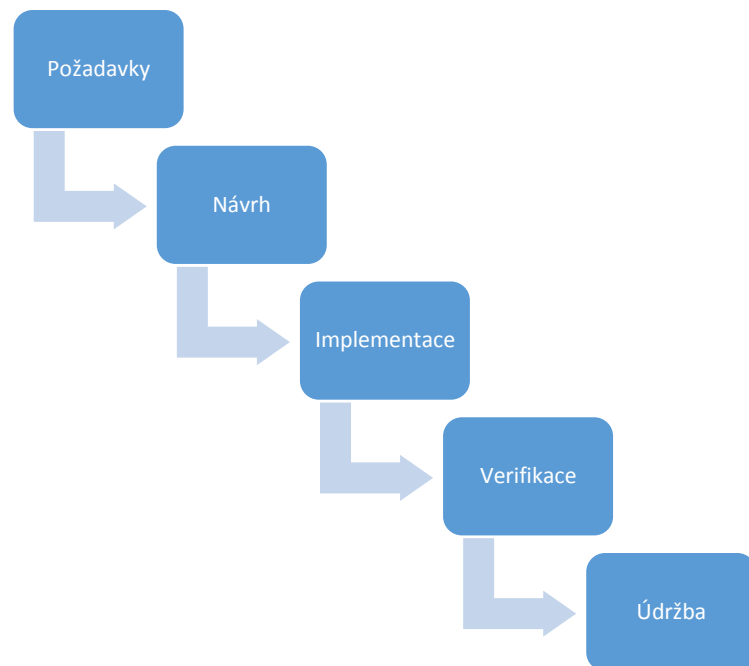
Ve vývoji software existuje spousta doporučení a pohledů na věc. Než člověk začne se samotným testováním software, měl by mít alespoň obecný přehled, jakým způsobem samotný software vzniká a následně pak se zabývat otázkou, proč vlastně software testujeme. Před samotným vývojem je důležité vytvořit dobrý návrh. Především pak, pokud vyvíjíme pro zákazníka, je potřeba důkladně prokonzultovat veškeré požadavky. Na základě požadavků je vhodné vytvoření modelu. K modelování můžou vhodně posloužit nástroje, jako je jazyk UML, CASE nástroje, ale i jiné. Při modelování jsou navrhovány k přehlednému zobrazení jednotlivých dílčích částí SW, k návrhu vztahů a spolupráce mezi nimi. Také k návrhu vstupů a výstupů jednotlivých částí. Modelování má pak dopad na samotný průběh vývoje. Vývoj je více organizovaný a předchází před tvorbou nedomyšlených kusů kódů.

### 1.1 Přístup k vývoji SW

K vývoji SW lze přistupovat několika způsoby. Jsou dva základní typy přístupů a to sekvenční (lineární) a iterativní. Hlavním modelem pro sekvenční přístup je model vodopádový. Zástupcem iterativního přístupu je prototypový model. Kombinací lineárního a iterativního přístupu pak vznikají modely jako inkrementální a spirálový.

#### 1.1.1 Vodopádový model

Hlavní myšlenkou je postup jedním směrem, z jednoho stádia do druhého bez možnosti cesty zpět, podobně jako tomu je u vodopádů, kde voda teče jen směrem dolů.

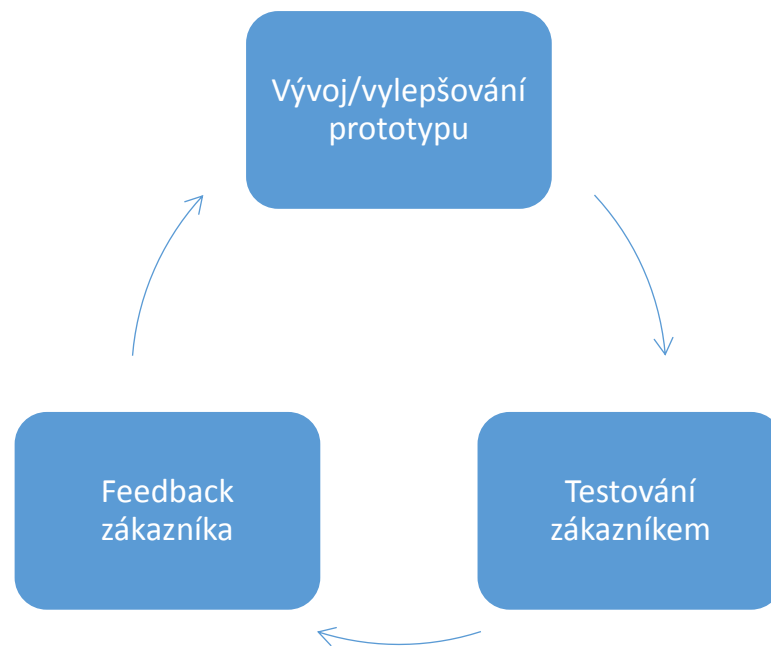


Obrázek 1: Vodopádový model

V tomto modelu je důležité plánování, přesné časové rozvržení a rozpočty. Celý vývoj SW se realizuje najednou.

### 1.1.2 Prototypový model

Jde o způsob vývoje SW, kde jsou vyvíjeny prototypy SW, který je testován zákazníkem a na základě zpětné vazby zákazníka je SW dále vyvíjen.

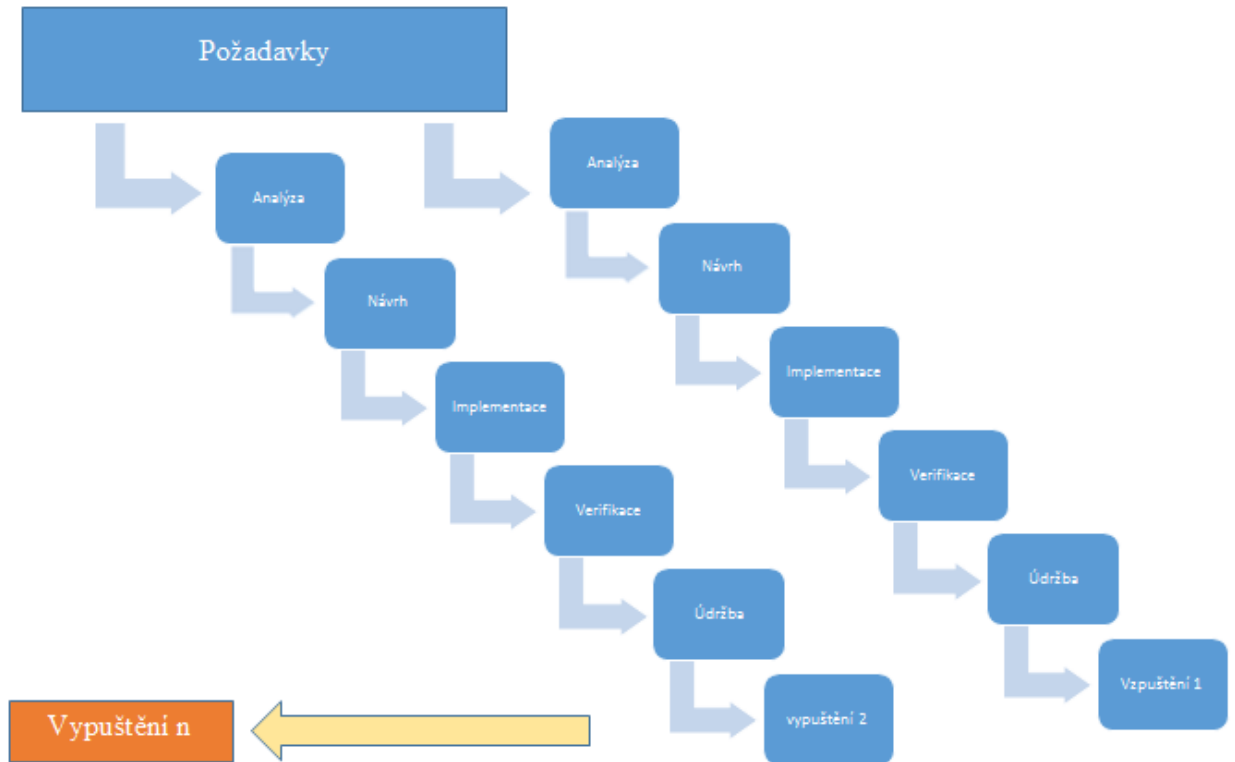


Obrázek 2: Prototypový model

Snahou tohoto modelu je snížení nebezpečí projektových rizik. Rozdělením vývoje na menší celky lze snadněji dosáhnout změny v průběhu vývoje. Další výhodou je, že je zákazník součástí vývoje a tak je větší pravděpodobnost uspokojení zákazníka. [1]

### 1.1.3 Inkrementální model

Inkrementální, tedy přírůstkový model funguje na principu nabalování přírůstků. Každý přírůstek je vyvíjen např. po vzoru modelu typu vodopád.

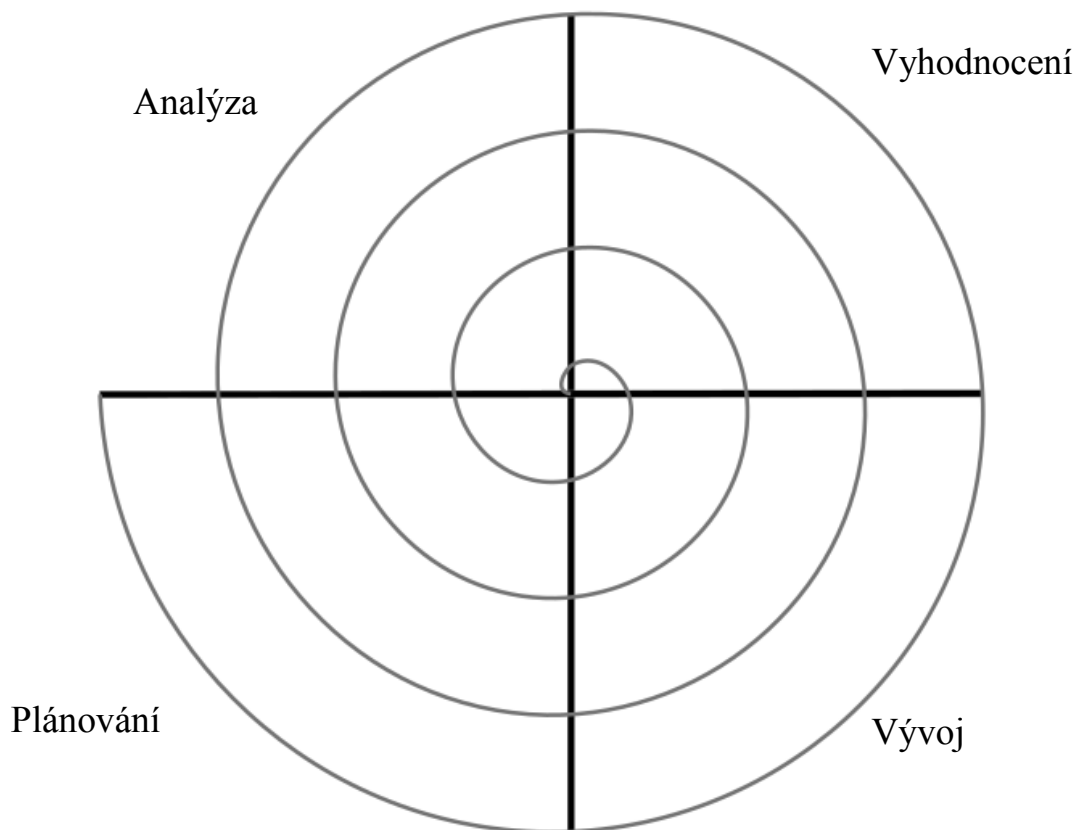


Obrázek 3: Inkrementální model

Cílem je omezit rizika selhání produktu rozdělením projektu na menší části. Výhodou může být to, že si požadavky rozebere více týmů a pracují na nich souběžně a přitom nezávisle. [2][3]

#### 1.1.4 Spirálový model

Spirálový model se prezentuje graficky jako spirála, u které se smyčka nazývá fáze vývoje. Počet těchto smyček se mění od projektu k projektu. Poloměr spirály značí vynaložené náklady na vývoj, zatím co úhlový rozměr odpovídá dosavadnímu pokroku. Každá fáze je rozdělena do čtyř kvadrantů (Analýza, vyhodnocení, vývoj a plánování). [4] [5]



Obrázek 4: Spirálový model

### 1.1.5 Rapid Application Development (RAD)

Důraz na rychlost a nízkou cenu vývoje. Jde-li produkt rozdělit na malé moduly, lze implementovat tento model tak, že jsou přiřazeny jednotlivé modely jednotlivým týmům, které na sebe nejsou závislé. Vývoj každého modulu pak zahrnuje kroky jako u vodopádového modelu. [5][6]

### 1.1.6 Agilní vývoj

Agilní vývoj se soustředí na usnadnění rychlého dokončení projektu. V agilním vývoji se neplýtvá časem a energií na činnostech, které nejsou nezbytné. Požadavky jsou rozděleny na menší části. Ty jsou pak vyvíjeny postupně. Agilní vývoj se zakládá na iterativním přístupu. Z pravidla jsou iterace malé a snadno docílitelné a to během několika dnů až týdnů. V každé iteraci se provádí plánování, vývoj a doručení zákazníkovi. V krocích tedy shromáždění požadavků, následně jejich analýza, návrh, samotné kódování, kde následují jednotkové testy a nakonec akceptační testy.

V agilním vývoji je nejdůležitější uspokojit zákazníka. To se snaží docílit rychlým a průběžným dodáváním kvalitního SW. Obrovskou výhodou může být to, že v agilním vývoji není tolik kladen důraz na dokumentaci, pokud nějaká dokumentace vůbec vzniká. Namísto dokumentace je důležitější fungující SW. Průběžným doručováním SW zákazníkovi lze včas reagovat na změnu. V agilním vývoji se počítá s tím, že zákazník většinou sám neví, co vlastně chce. Požadavky na změny jsou tedy vítány i v průběhu vývoje. Dalším parametrem spojeným se zmíněným je i to, že spolupráce mezi lidmi je důležitější než nástroje a procesy. Tato spolupráce probíhá skrz na skrz všemi podílejícími se na projektu. Tedy i lidi z businessu spolupracují s developery a to na každodenní bázi během celého vývoje. Důležité je mít motivované jedince. Doporučuje se, aby tým byl malý (5-9) osob. V malém týmu je větší potenciál, že se do komunikace zapojí všichni. Existuje zde idea, že informovanost celého týmu dosáhneme efektivněji osobním kontaktem, než předáváním formálních dokumentů.

Agilní vývoj zavádí i párové programování. To je založeno na ideji, že párové programování vyprodukuje kvalitnější program s méně chybami, než kdyby totéž měl programovat jednotlivec. Párové programování je založeno na střídání rolí dvou programátorů, kdy jeden píše kód a druhý dělá revizi.

Ve výsledku lze tedy říci, že jde o vývoj, který snižuje celkovou dobu vývoje a při tom má zástupce zákazníka přehled po celý vývoj o aktuálním stavu produktu, díky čemuž může kdykoliv pozměnit požadavky. Nevýhodou však může být absence formálních dokumentů a tak může docházet ke špatné interpretaci mezi členy týmu a po dokončení produktu, kdy vývojáři již pak jsou přeřazeni na jiný projekt, může být problematická údržba.

Nejznámějšími zástupci agilního modelu jsou SCRUM, Extreme programming (XP) a test driven development (vývoj řízen testy). [7] [8] [9]

## 2 TESTOVÁNÍ

Testování software je nedílnou součástí vývoje. Testováním lze dosáhnout zvýšení kvality. Testovat SW lze několika různými způsoby, které lze také kombinovat. Některé testy se zaměřují na bezpečnost programu, jiné na výkon a pak především na samotnou funkčnost. Způsob testování lze rozdělit několika způsoby.

### 2.1 White/black box

Základním rozdělením testování pohled na program, kde buď máme možnost vidět jen výsledný produkt z pohledu uživatele, nebo máme možnost číst kód samotného programu.

Black box testing – jedná se o způsob testování, kdy tester nezná samotnou strukturu programu a při testování vychází jen z požadavků a popisu o tom, jak by se měl výsledný produkt chovat. Při testování se tedy vychází z chování programu a to tak, že sledujeme jaký výsledek dostaneme po zadání vstupních parametrů. [10]

White box testing – v tomto případě má tester k dispozici zdrojový kód a má znalosti struktury programu. Díky tomu má tester více možnosti k otestování všech možných průchodů kódem a k otestování neočekávaných vstupních hodnot. [11]

Gray box testing – je založen na kombinaci obou předchozích pohledů. [12]

### 2.2 Úrovně testování

Testovací úrovně souvisí s životním cyklem vývoje. V sekvenčních modelech životního cyklu jsou úrovně definovány tak, že výstupní kritéria jedné úrovně jsou součástí vstupních kritérií pro další úroveň. Toto pravidlo ale neplatí pro některé iterativní modely. Během iterací každá daná user story bude typicky probíhat sekvenčně skrze následující aktivity:

- Unit testy, které jsou obvykle prováděny vývojáři
- Akceptační testy, které mohou být rozděleny do následujících dvou
  - Testy ověřující funkčnost, které jsou často automatizované, mohou být prováděny jak vývojáři, tak testery
  - Testy ověřující funkčnost, které jsou obvykle manuální. Můžou být prováděny vývojáři, testery, ale také obchodními partnery spolupracujícími na ur-

čení, zda je funkce vhodná k použití, ke zpřesnění dosaženého pokroku a aby poskytli zpětnou vazbu.

Současně se navíc provádí často i regresní testování, které zahrnuje opakované spouštění automatizovaných jednotkových testů a verifikační testy funkcí jak z aktuální iterace, tak i z iterací minulých. [13] [14] [15]

### 2.2.1 UNIT testy

Zaměřují se na testování komponent (nejmenší testovatelná část SW). Klíčovou roli hrají při ujištění se v projektech, kde se průběžně mění kód, že tyto změny neporušili stávající komponenty. UNIT testy mohou zamezit uniknutí chyb do vyšších testovacích vrstev. Nalezené chyby v této vrstvě jsou běžně opraveny, jakmile jsou nalezeny, bez vytváření oficiálních reportů. Oprava v této úrovni je v podstatě nejméně nákladná, jelikož k nalezení chyby stačí projít poslední napsané úpravy. Testy jsou psány obvykle vývojáři současně s psáním testovaného kódu. [14] [16]

### 2.2.2 Integroční testy

Integroční testy se zaměřují na testování závislostí mezi komponenty a systémy. Stejně jako u jednotkových testů, integrační testy ověřují, zda nedošlo k poškození stávajících rozhraní, komponent nebo systémů. Pokud moduly tvoří nedílný celek, testování by mělo být zaměřeno na testování komunikace, mezi jednotlivými moduly, ne na testování modulů samotných. V případě, že tvoří nedílný celek systémy, pak se zaměřuje na komunikaci mezi systémy. Samotnou funkčnost systémů testujeme v systémovém testování.

Integraci komponent testují většinou vývojáři, zatím co integraci systémů testují testeři. [14]

### 2.2.3 Systémové testování

Systémové testování se zaměřuje na celkové chování a vlastnosti systému jako celek. Tato úroveň testování také přináší jisté ověření, zda s novými změnami nedošlo k poškození existující vlastnosti. Jde o testování produktu z pohledu zákazníka. Provádí se před doručením k zákazníkovi. Výsledky z testování mohou pomoci rozhodnout zájmovým stranám o vydání SW. Testovací prostředí by mělo ideálně odpovídat tomu výslednému, či produkčnímu. Testování probíhá na základě testovacích scénářů, které popisují scénář,

který by mohl nastat v praxi. Po případném nalezení a opravení chyby je třeba tyto testy opakovat. Testování provádí obvykle testeři, kteří jsou nezávislí. Pokud specifikace nejsou dostatečné, či chybné, tester může očekávat jiné chování, než je vyžadováno. Tato skutečnost pak může způsobit situaci, že vzniknou plané výsledky (chybové i pozitivní). [14] [17]

#### **2.2.4 Akceptační testování**

Akceptační testy se zaměřují na chování a vlastnosti celého produktu, stejně jako u systémového testování. Nalezení chyb však není cílem. Akceptační testování mohou posloužit k posouzení o připravenosti produktu k nasazení a použití zákazníkem.

##### **2.2.4.1 Uživatelské akceptační testování**

Zaměřeno na ověření vhodnosti pro použití systému plánovanými uživateli v reálném nebo simulovaném provozním prostředí. Hlavním cílem je ověření, že systém plní potřeby uživatelů, plní požadavky a obchodní procesy s minimálními obtížemi, náklady a riziky.

##### **2.2.4.2 Funkční akceptační testování**

Akceptační testování je prováděno obsluhou nebo zaměstnanými správci v produkčním prostředí. Zaměřuje se na provozní aspekty, jako může být instalace, upgrade, zálohování, kontrola bezpečnosti a jiné. Hlavním cílem je ověření, že provozovatel nebo správce může udržovat uživatelům systém v pořádku v provozním prostředí.

##### **2.2.4.3 Smluvní a regulační akceptační testy**

Smluvní akceptační testy jsou prováděny vzhledem ke smluvním akceptačním kritériím pro vytvoření SW na zakázku. Tato akceptační kritéria by měla být vytvořena při uzavření spolupráce. Smluvní akceptační testy jsou často prováděny uživateli nebo nezávislými testery.

Regulační akceptační testy jsou prováděny vzhledem ke všem regulacím, které musí být dodrženy. Jsou to třeba vládní, právní nebo bezpečnostní předpisy. Jsou často prováděny uživateli, nebo nezávislými testery někdy s výsledky, které se mohou stát svědectvím nebo kontrolou pro regulační orgány.

#### 2.2.4.4 Alfa a Beta testování

Typicky je používáno komerčními vývojáři, kteří mají zájem získat zpětnou vazbu od potenciálních, nebo stávajících uživatelů před uvedením produktu na trh.

Alfa testování je prováděno na stránkách vývojové organizace, zatím co Beta testování je prováděno na prostředí uživatelů. Beta testování může přijít po Alfa testování, ale i bez něj.

Hlavním důvodem je ověřit, že lze systém používat za běžných podmínek a případné zjištění chyb, které jsou závislé na prostředí a situacích, za kterých je produkt užíván.

### 2.3 Manuální testování

Tester by měl představovat koncového uživatele a zkoušet všechny možné funkce fungují dle specifikací. Během testování tester provádí test case (testovací scénáře) a na základě výsledku produkuje reporty manuálně bez užití některých z automatizačních nástrojů.

#### 2.3.1 Exploratory testing

Manuální testování se využívá především při exploratory (ověřovacích) testech. Jedná se o testování založené na zkušenostech testera. Je tedy vhodné, aby bylo prováděno zkušenými odborníky. Testování se provádí bez znalostí požadavků. Zkoumány jsou funkce aplikace.

#### 2.3.2 Usability testing

Testování použitelnosti slouží k ověření, že je aplikace uživatelsky příjemná. Cílem je vyhodnotit, zda koncový uživatel může snadno pochopit prostředí a bude schopen aplikaci používat.

#### 2.3.3 Ad-hoc testing

Neformální způsob testování, kdy testeři náhodně testují aplikaci bez dodržování dokumentace a bez testovacích návrhů. Primárně je testování prováděno testery se znalostí aplikace.

## 2.4 Automatizované testování

Testování je prováděno za pomoci automatizačních nástrojů. Jsou spouštěny testovací skripty a výsledky jsou tvořeny automaticky automatizačními nástroji. Ne vždy je však automatizace testů vhodná. V zásadě je důležité, aby testy byly opakovatelné a to i výhledově do budoucna. Dále by měly testovat funkcionality, které jsou již dokončeny.

### 2.4.1 Regresní testování

Regresní testování je využíváno u agilního vývoje. Opakovaným spouštěním testů lze ověřit, zda postupným vývojem a úpravou programu nedošlo k poškození některé z části programu. Díky častému provádění změn v kódu a tedy častému spouštění totožného testu je regresní testování vhodným příkladem využití automatizovaného testování. Provádění regresních testů manuálně by bylo velmi časově náročné. [18]

### 2.4.2 Zátěžové testy

Klade si za úkol ověřit produkt, že zvládne očekávaný počet transakcí a normálně fungovat i v období, kdy dochází ke špičce v provozu. Automatizované testování je nejlepším způsobem, jak zátěžové testy provádět. [19]

### 2.4.3 Testování výkonu

Tento typ testování zjišťuje nebo ověřuje rychlost, škálovatelnost a stabilitu produktu. [19]

## 3 TESTOVÁNÍ V AGILNÍM PROSTŘEDÍ

### 3.1 Spolupráce na tvorbě uživatelských scénářů

Uživatelské scénáře (user stories) jsou v agilním vývoji psány k zachycení požadavků z perspektivy vývojářů, testerů a obchodních zástupců. Uživatelské scénáře musí obsahovat funkcionální i nefunkcionální charakteristiky. Každý příběh zahrnuje akceptační kritéria pro tyto charakteristiky. Tato kritéria by měla být definována ve spolupráci mezi vývojáři, testery a obchodními zástupci. V agilním týmu se považuje úkol za dokončený, když jsou akceptační kritéria splněna. Pohled z perspektivy testera může vylepšit uživatelské příběhy tím, že najde chybějící detaily nebo nefunkcionální požadavky. Tester tedy může položit doplňující otázky obchodním zástupcům a navrhne jakým způsobem otestovat tyto uživatelské příběhy a ověří akceptační kritéria.

### 3.2 Retrospektiva

Jde o setkání, které se koná na konci každé iterace. Cílem je diskutovat o tom, co bylo úspěšné, co by se dalo zlepšit a jak tato zlepšení začlenit a jak pokračovat s úspěchy v následujících iteracích. Výsledkem souvisejícím s testováním jsou rozhodnutí o zaměření se na účinnost testů, produktivitu testů, kvalitu test case a spokojenost týmu. Taktéž se řeší testovatelnost aplikací, uživatelských příběhů, funkcí, nebo systémového rozhraní. Analýza problémů může vylepšit řízení testu a vylepšit vývoj.

### 3.3 Continuous Integration

Na konci každého sprintu je od rozšíření produktu vyžadována spolehlivost, funkčnost a sjednocení software. Průběžná integrace tuto problematiku řeší průběžným slučováním všech vytvořených změn v SW, a sjednocuje všechny změněné komponenty průběžně alespoň jednou denně. Tím, že vývojáři neustále integrují své změny, mohou být průběžně sestaveny (vytvořen build) a následně otestovány. Díky tomu případné chyby v kódu jsou nalezeny daleko rychleji.

Nicméně kompilace a vydání nové verze je složitý proces. Z tohoto důvodu je vhodné využít nástroj k automatizaci buildu.

Průběžná integrace umožňuje agilním testerům spouštět automatizované testy pravidelně. V některých případech mohou být testy spouštěny v rámci samotného procesu prů-

běžného integrování. To poskytuje rychlou zpětnou vazbu o kvalitě kódu. Tyto výsledky mají možnost vidět všichni členové týmu. Tím, že jsou regresní testy automatizované, testéři se mohou věnovat manuálnímu testování týkajících se nových funkcí a implementačních změn.

## 4 NÁSTROJE PRO AUTOMATIZACI TESTOVÁNÍ UŽIVATELSKÉHO ROZHRANÍ

Pro ulehčení práce při vytváření testů lze využít nástroje k tomu určené. Těchto nástrojů existuje mnoho. Důležité je vybrat si ten správný.

### 4.1 Test Automation Framework

Obecně Framework slouží jako podpora při programování. Definuje sadu pravidel a nejlepších postupů, kterými se můžeme řídit pro dosažení požadovaných výsledků. Framework může obsahovat knihovny API, podporu pro návrhové vzory, podpůrné programy a doporučené postupy.

Framework pro automatizované testování je tedy v podstatě soubor pokynů sloužících k vytváření a navrhování testů. Použitím frameworku si tester může ulehčit spoustu práce. Odpadnou tak náležitosti spojené s vytvářením kódu, což vyžaduje jistý stupeň zkušeností a také, o který je třeba se starat. Tester se tak může soustředit na práci spojenou se samotným testováním. [20] [21] [22]

#### 4.1.1 Lineární Skriptovací Framework

Je to základní úroveň automatizačních frameworků, který je ve formě „Record and Playback“ lineárním způsobem. Tento typ je vhodný pro malé aplikace. Vytváření a spouštění testů se provádí pro každý test samostatně.

Výhodou je, že není třeba ztrácet mnoho času tvorbou testovacích skriptů. Nicméně má nedostatky jako je opakovatelnost, nebo natvrdo zapsaná data, takže není možné spouštět test na různých sadách dat. [22] [23]

#### 4.1.2 Modální testovací Framework

Zde testeři vytváří testovací skripty vhodným rozdělením celé aplikace na menší nezávislé testy. Testeři si tedy rozčlení aplikaci na menší moduly a testovací skripty vytváří individuálně. Z takto vytvořených skriptů lze jejich kombinací vytvářet větší testovací skripty za pomoci hlavního skriptu tak, aby byly pokryty požadované testovací scénáře. Testy se pak spouští za užití vytvořených hlavních testovacích skriptů. [22] [23]

### 4.1.3 Data-driven Framework

Daty řízený framework pro automatizaci testů se zaměřuje na rozdělení logiky testovacích skriptů a testovacích dat od sebe. Umožňuje to vytvářet skripty pro automatizované testy předáváním různých sad testovacích dat. Tyto sady testovacích dat jsou uchovávány v externích souborech, nebo ve zdrojích jako je XML soubor, Databáze, sešit Excelu a jiné. Tento Framework výrazně snižuje počet vytvořených skriptů na rozdíl od modálního testovacího Frameworku. [22] [23]

### 4.1.4 Framework pro testování klíčovými slovy

Využívá se tabulka k určení klíčových slov, slov popisujících akci pro každou funkci nebo metodu, kterou bychom mohli provádět. Pomocí tohoto Frameworku můžou testeři vytvářet jakýkoliv testovací skript pro automatizaci užitím těchto klíčových slov. Testeři, kteří mají menší znalosti v programování, jsou tak schopni také vytvářet testovací skripty. Nicméně počáteční práce a nastavení vyžaduje více odbornosti. [22] [23]

### 4.1.5 Hybrid Driven Testing Framework

Jde o kombinaci dvou a více uvedených frameworků. Snaží se využít více druhů výhod. [22] [23]

### 4.1.6 Framework pro chování řízený vývoj testování

Behavior Driven Testing (BDT) velmi uznávaná a běžná metodika v agilním prostředí. Testy se více zaměřují na chování uživatelů, než na technické funkce SW. BDT je nejlepší volbou, pokud chceme představit vlastní pohled na činnosti a požadavky v produktu. K tomu, abychom rozvíjeli myšlenky o produktu, používá se snadno srozumitelný jazyk, tak aby to bylo pro všechny jasně srozumitelné. A tak se můžou zapojit aktivně do procesu testování lidé, kteří mají na starosti obchodní analýzu a správu produktu.

Přestože nástroje BDT jsou speciálně vyvíjeny pro použití v projektech BDD (Behavior Driven Development), lze tyto nástroje brát jako speciální formu nástrojů, které pomáhají v TDD.

Dan North, který vyvinul BDD, zjistil, že jednotkové testy by měly být pojmenovány celými větami, které by měly začínat slovy „Měl by“ (z anglického slova should) a měly by být psány v pořadí dle byznys významu. Akceptační testy by měly být psány po vzoru uži-

vatelských příběhů agilního frameworku. Předloha: „*Jako [role uživatele] chci [nějaký výsledek] abych [nějaký důvod]*.“ Akceptační kritéria by měly být psané v podmínkách scénáře a měly by být implementovány jako třídy. Předloha: „*Vzhledem k [počáteční souvislost], pokud [nastane událost], tak [zajisti nějaké výsledky]*“. [24] [25]

## 4.2 Používané nástroje pro automatizované testování webu

Testovacích nástrojů existuje celá řada. Proto bych rád zmínil jen ty nejpoužívanější.

### 4.2.1 Selenium

Pro vývojáře a testery, kteří mají nějaké zkušenosti s programováním a skriptováním, Selenium poskytuje flexibilitu, která u mnoha ostatních nástrojů není. Testy lze psát pomocí širokého spektra programovacích jazyků. Selenium je podporováno v Javě, C#, Python, PHP, Ruby, Perl a Groovy. Spustit testy lze na Windows, Linux, Android i iOS. Prohlížeče, ve kterých je možnost testovat, je taktéž široký výběr. Jde o Chrome, IE, Firefox, Opera, Safari a jiné.

Selenium má však nevýhodu v tom, že použití tohoto nástroje vyžaduje pokročilé programovací znalosti a u začínajících projektů také určitý čas k vytvoření vhodného Frameworku pro svůj produkt. Existují však již vytvořené Frameworky, které lze použít. [26] [27]

### 4.2.2 Katalon Studio

Katalon umožňuje vytvářet automatizované testy skriptováním, manuálně vybíráním klíčových slov, nebo způsobem „nahraj a zopakuj“. Dále umí vytvářet testovací výstupy a reportovat jejich výsledky. Nástroj je zdarma ke stažení po registraci na jejich stránkách. Použití vypadá uživatelsky přívětivé a existují video tutoriály, podle kterých lze začít. [28]

### 4.2.3 Unified Functional Testing (UFT)

Jedná se o oblíbený komerční nástroj pro automatizované testování desktopových, webových a mobilních aplikací. Používat UFT může také neodborný tester. Poskytuje funkci „nahraj a proved““. K automatizaci aplikací používá VBScript. Umí velmi dobře

identifikovat objekty. Roční cena produktu je 3200 dolarů, což je v přepočtu asi 73 tisíc Kč. [26] [29] [30]

#### 4.2.4 TestComplete

TestComplete také podporuje automatizaci testů pro desktop, web a mobilní aplikace. Poskytuje funkci „nahraj a proved““. Testy lze použít na různých prohlížečích. Cena licence začíná v přepočtu kolem 72 tisíc Kč. [26] [31]

#### 4.2.5 SoapUI

Kromě testování webu a mobilních aplikací zvládá testovat API a služby. Nabízí se jak open-source, tak i pro verze. [26] [32]

#### 4.2.6 Robot Framework

Robot Framework se stal velmi populární díky své jedinečné syntaxi, která je velmi srozumitelná i pro méně zdatné jedince v programování.

Příklad syntaxe:

```
*** Test Cases ***
Valid Login
    Open Browser To Login Page
    Input Username      demo
    Input Password     mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]        Close Browser
```

Dále umí vytvářet přehled výsledků. Robot Framework je open source. [33]

#### 4.2.7 Ranorex

Podporuje automatizaci testů pro desktop, web a mobilní aplikace. Je také vhodný i pro začátečníky, díky možnosti nahrávání postupů. Cena licence začíná okolo 59 tisíc Kč. [34] [35]

## **II. PRAKTICKÁ ČÁST**

## 5 VÝBĚR NÁSTROJŮ A METOD

### 5.1 Webová aplikace

Na základě vlastních zkušeností byl vybrán programovací jazyk C#. Pro jazyk C# existuje ASP.NET Framework, který je určený k vytváření webových aplikací. ASP.NET je nástupcem ASP. Konkrétně byl vybrán ASP.NET 4.5 MVC Framework.

#### 5.1.1 ASP.NET

Jedná se o vyspělou architekturu. Poskytuje služby nezbytné k sestavení na byznys úrovni. Malou nevýhodou může být, že produkt napsaný v tomto Frameworku lze nasadit pouze na servery s Windows. V případě, že by byl požadavek, aby výsledná aplikace mohla běžet na serveru s Linuxem, nebo macOS, bylo by potřeba zvolit ASP.NET Core. Nicméně .NET Framework nabízí vytvoření webových formulářů a webových stránek. ASP.NET obsahuje knihovny, které výrazně ulehčí vývoj samotné aplikace. Tyto knihovny obsahují hotová řešení pro bezpečnost, autentifikaci uživatele, práci s databází a jiné. [36]

##### 5.1.1.1 *Dynamický web*

ASP.NET Framework poskytuje možnost vytvářet dynamické webové stránky. To funguje tak, že server ještě před tím, než odešle data uživateli, má možnost ještě stránku upravit. Oddělení logiky od prezentace je velmi populární. Usnadňuje to údržbu a zpřehledňuje aplikaci. Aktualizace informací na stránce je mnohem snadnější, než je tomu u statických webů.

Díky tomu lze vytvářet propracovanější grafické rozhraní. Lze například vytvářet rolovací menu, které reaguje na pohyb kurzoru, změnit, nebo rozšířit obsah stránky pro identifikované návštěvníky a jiné.

Zpracování takových stránek probíhá tak, že prohlížeč odešle požadavek na webový server o dynamickou stránku. Server stránku vyhledá a předá ji aplikačnímu serveru. Aplikační server vyhledá na stránce případné instrukce, které zpracuje, a stránku doplní, tak aby odpovídala požadavkům. Aplikační server dokončenou stránku předá webovému serveru, který ji pak následně předá prohlížeči, který si o ni požádal.

Dalším důležitým poznatkem je přístup k databázi. Aplikační server nemůže s databází komunikovat přímo. Je potřeba využít databázový ovladač. Pomocí tohoto ovladače

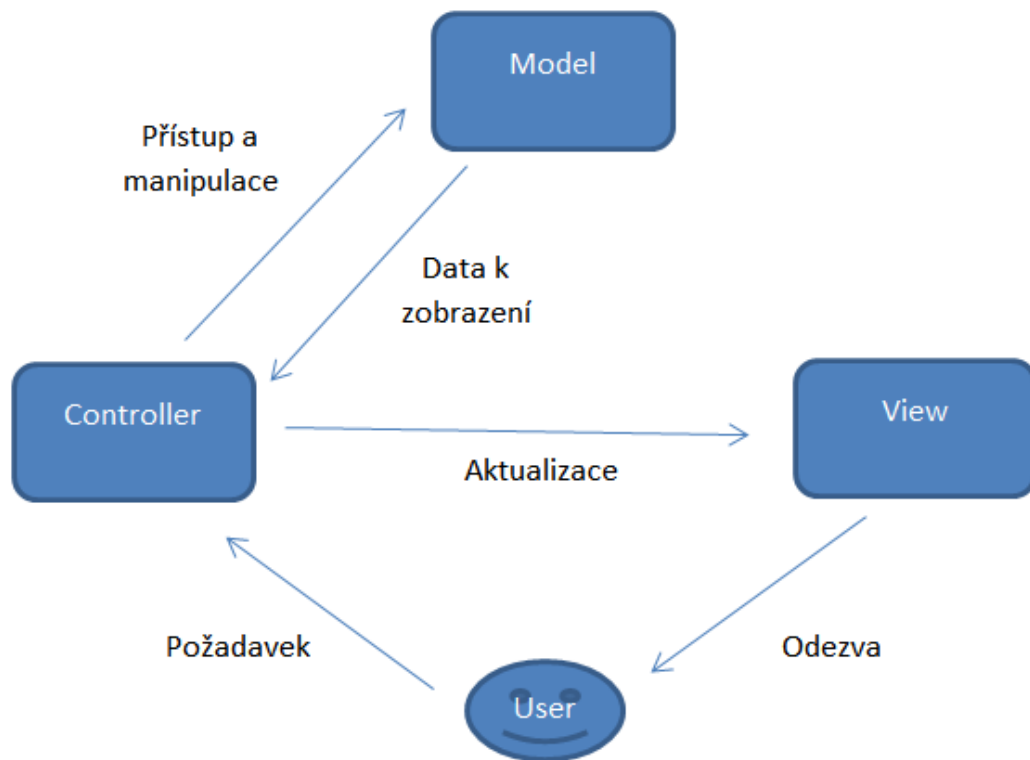
dače má možnost aplikační server vytvořit dotaz na databázi. Databáze dotaz zpracuje a vrátí sadu záznamů. V databázi lze shromažďovat data, která při zpracování dynamických stránek aplikační server později vyhledá v databázi a vrátí stránku s daty, která odpovídají požadavkům.

Vytváření rozhraní pro uživatele je tak daleko příjemnější. Stránky, které se ve výsledku zobrazí uživateli, lze vhodně rozdělit na několik částí. Tedy například navigační menu a hlavičky lze napsat do jednoho souboru, který popisuje rozvržení stránky, a obsah stránky, který se mění dle aktuální pozice na webu, lze napsat zvlášť do samostatných souborů, které obsahují jen jádro samotné stránky bez navigačního menu a hlaviček. Dále lze vytvářet jen části stránek, které lze pak použít na jednotlivých stránkách. Celý vývoj je pak daleko přehlednější a změny, které především v počátku probíhají (např. změna názvů), zaberou jen minimum úprav. [37]

### 5.1.2 ASP.NET MVC Framework

Model-View-Controller (MVC) je architektura, která rozděluje do tří samostatných, nezávislých komponent datový model aplikace, uživatelské rozhraní a řídicí logiku. Úprava jedné z komponent ovlivní zbylé jen minimálně.

Obecný princip MVC je vyobrazený na obrázku (Obrázek 5: MVC model). Uživatel provádí nějakou činnost v uživatelském rozhraní (View). Řadič o činnosti dostane informaci a přistupuje k modelu. Model na základě požadavku provede úkon s databází a vrátí do řadiče případná data. Řadič zpracuje získaná data a vytvoří uživatelské rozhraní. [38] [39]



Obrázek 5: MVC model

Model odpovídá za obchodní logiku, všechny funkce, které lze provádět s modelem a uchovává si stav aplikace. Pro složité uživatelské rozhraní se používá ViewModel (model pro zobrazení), který controller vytváří a plní obdrženými daty pro daný model.

Uživatelské rozhraní odpovídá za prezentovaný obsah. Aby bylo možné vkládat .NET kód do HTML, používá se zobrazovací modul Razor. Logika by zde měla být minimální a měla by se týkat pouze zobrazování obsahu.

Kontrolér odpovídá za zpracování činnosti uživatele, práce s modelem a za výběr zobrazení, které má být vykresleno. Kontrolér je v MVC vstupní bod, který vybírá typ modelu, se kterým se bude pracovat a které zobrazení se vykreslí. [40]

## 5.2 Testovací framework

Pro část testování byl zvolen Selenium, konkrétně Selenium WebDriver. Pro potřeby této diplomové práce nabízí všechny důležité funkce. Podpora pro tento nástroj je velmi obsáhlá a komunita používající tento nástroj vytvořila nespočet příspěvků, kde si lze některé užitečné rady nastudovat.

## 6 NÁVRH APLIKACE

Při návrhu aplikace byly zohledněny známé aspekty, které s problematikou souvisí, a také byl brán zřetel na to, které komponenty jsou nezbytně potřebné.

### 6.1 Identifikace uživatele

První část, bez které by se aplikace nedala vytvořit, je identifikace uživatele. Díky identifikaci lze dosáhnout jistého zabezpečení dat, tím, že data, která jsou jednoho uživatele, neuvidí ostatní.

#### 6.1.1 Role

Aby mohla vzniknout administrační sekce v aplikaci, je vhodné nastavit uživatelům role. Role umožňují skupinově rozhodovat o tom, kdo má povolení provádět jednotlivé akce. Jako základ jsou použity dvě role a to role pro admina a role pro uživatele, kteří jsou zaregistrovaní. Nepřihlášený uživatel bude bez skupiny a jeho práva budou tedy minimální.

#### 6.1.2 Přihlášení a registrace

K identifikaci uživatele je potřeba, aby měl uživatel možnost registrovat se a přihlásit. Pro ověřování je zapotřebí uchovat informace o uživateli včetně hesla. Heslo však musí být chráněno před hrozbou útoku, proto je potřeba jej vhodně zašifrovat. Vhodné je použít třeba funkci Hash. V projektu je použita hashovací funkce SHA-512.

Pokud by uživatel zapomenul heslo, může využít možnost, nechat si na email zaslat odkaz pro změnu hesla. Tento email však musí být nejdříve ověřený.

### 6.2 Návrh struktury

Aplikace se zabývá testováním. Hlavním zkoumaným prvkem je tedy test. Test jakožto takový má určitou strukturu. Tato struktura obsahuje název, podle kterého by uživatel mohl odvodit, co daný test provádí. Dále obsahuje popis, kde je přehledně popsáno, o jaký test jde. Každý test má požadovaný postup, tedy sled kroků, které určují proces testu. Těmito kroky jsou myšleny jednotlivé akce, které lze na webové stránce provádět (např. kliknutí na element). Každý test také začíná otevřením prohlížeče a to na určené stránce, proto je třeba v rámci struktury použít i odkaz.

Tyto testy náleží pod určitý projekt. Tímto projektem lze chápat třeba testovanou aplikaci. V rámci jednoho projektu se vytváří množství testů, tak aby bylo možné otestovat aplikaci co nejefektivněji. Mimo to v rámci jednoho projektu je běžné, že pracuje více testerů. Proto je zde možnost přidávat další členy týmu do projektu.

### 6.2.1 Tvorba testů

Z kroků v testu by měl být čitelný postup, co daný test provádí. V ideálním případě vytvořený test odpovídá samotnému test case (testovací scénář). Čímž odpadá nutnost udržovat ještě dokumentaci testovacích scénářů. Na příklad by to mohlo vypadat následně:

*Student UTB má přístup do STAG*

1. Otevřít stránku „<https://stag.utb.cz/portal>“
2. Napiš „*jmeno\_uzivatele*“ do „Uživatelské jméno:“
3. Napiš „*heslo\_uzivatele*“ do „Heslo:“
4. Klikni na „Přihlásit se“
5. Je zde viditelný element „Zobrazit můj rozvrh“

Po prozkoumání problematiky tvorby testů byl vytvořen návrh, jak uživateli poskytnout možnost vytvářet testovací kroky jednoduchým a přehledným postupem. Každý krok vykonává specifickou činnost. Tato činnost je prováděna nad určitým elementem. Příkladem je činnost „Kliknutí na tlačítko ‘Potvrdit’“. Některé akce vyžadují ještě další atribut. Například akce pro zapsání textu do elementu. Zde je atributem hodnota, kterou je požadováno vepsat do elementu. Tyto kroky musí být prováděny ve správném pořadí, což je také důležité zohlednit.

### 6.2.2 Elementy

U elementů je třeba uchovávat informace o identifikátoru. Selenium umí několika způsoby hledat elementy na stránce. Nejčastěji je však používán CSS selector a XPath.

Element by měl mít také pojmenování, které bude dávat smysl běžnému čtenáři. Jde o hodnotu, která se bude zobrazovat uživateli při výběru elementu, ale také v logování testů. Jako pojmenování lze použít například text obsažený v tomto elementu.

Element lze používat v rámci jednoho projektu a to v libovolném množství testů. Bylo by nevhodné vytvářet v každém testu selektor, který byl již jednou vytvořen.

V případě změny názvu, nebo selektoru by tak bylo třeba upravovat všechny testy, ve kterých byl element použitý.

### 6.2.3 Testovací logy

Užitečnou součástí automatizovaného testování jsou logy, tedy záznamy o průběhu testu, ze kterých lze vyčíst, jak test dopadl. V případě, že některý krok nemohl být proveden, bude tato skutečnost zaznamenána v tomto logu i s případnou zprávou. Díky logům má tester možnost analyzovat, co přesně se stalo v testu. Jestliže tedy test selže (nalezne chybu v testované aplikaci), je pro testera mnohem snazší reportovat tuto skutečnost. Tester nemusí vytvářet manuálně postup, jak reprodukovat chybu, ale stačí, aby předal vytvořený log. Vývojář pak může projít tento scénář vlastnoručně a analyzovat tak chybu, která při vývoji vznikla. V ideálním případě tester nemusí předávat tuto zprávu manuálně, ale vývojář má možnost sám přistoupit k těmto výsledkům, což zefektivní celý proces.

U logování je stejně důležité, jako u tvorby testů, aby byly logy čitelné a pochopitelné i pro osoby, které zastupují stranu zákazníka. Mnohdy jsou to lidé, kteří jsou bez technických znalostí.

Logy by také měly odpovídat 1:1 krokům v testovacím scénáři. Proto bylo navrženo, aby takový log měl stejnou strukturu jako vytváření testů, jen doplněné informací, zda byl krok úspěšný.

Dále je důležité zahrnout informace související se spuštěním testů. Konkrétně tedy informaci o čase, kdy test byl spuštěn. Tato informace je velmi důležitá především u projektů, které jsou velmi rychle vyvíjeny a i konkrétní minuta rozhoduje o tom, v které fázi jsme aplikaci testovali. Díky tomu je snazší určit, která změna měla za vinu rozbití funkcionality. Také je třeba udržovat informaci o tom, na kterém prostředí test proběhl. Chyby mohou být totiž závislé na prostředí. To co funguje například v prohlížeči Chrome na systému Windows 7, nemusí nutně fungovat v prohlížeči Edge na systému Windows 10.

### 6.2.4 Ukládání dat

Pro ukládání dat byla zvolena databáze. A to vzhledem k velkému množství dat a také k možnostem, které databáze nabízí. Například relace mezi daty, díky kterým lze vhodně a rychle dohledat potřebné informace, které jsou závislé na záznamech z jiných tabulek.

.NET technologie poskytuje možnost využít Entity Framework, který byl taktéž použit. Tento Framework umožňuje pracovat s databází s použitím objektů napsaných v .NET. Entity Framework mapuje relace objektů. Díky Entity Framework není třeba vytvářet většinu kódu pro přístup k datům, který by jinak vývojář musel psát. [41]

Pro vytváření tabulek byl zvolen pracovní postup Code First. Databáze je tedy modelována v závislosti na třídách entit, které si programátor vytvoří v projektu .NET.

### 6.2.5 Spouštění testů

Je nutné, aby testy byly spouštěny v připraveném prostředí. Často se pro tyto účely vytváří speciálně virtuální počítač ke vzdálenému připojení. Díky virtualizaci je možnost nasimulovat prostředí, které odpovídá reálnému produkčnímu prostředí. Na toto prostředí se tedy musí nainstalovat potřebné aplikace, aby bylo možné testy vzdáleně pouštět.

Pro spouštění testů na daném prostředí byla vytvořena desktopová aplikace. Tato aplikace bude stále aktivní na prostředí s nainstalovanými prohlížeči, aby bylo možné kdykoliv vyvolat akci, která testy spustí. Aplikace dostane informaci o tom, který test spustit. Následně si aplikace načte data, o tom, co se má a jak testovat. Poté spustí test a během testování bude odesílat průběžné výsledky k uložení do databáze.

Pro komunikaci mezi touto aplikací a severem, na kterém běží webová aplikace, byla navržena implementace modelu klient-server. V tomto případě bude serverem aplikace, která má na starosti spouštění testů. Na desktopovou aplikaci bude webová aplikace odesílat požadavky ke spuštění testu tak, že webová aplikace spustí klienta, který naváže komunikaci se serverem (aplikace ke spouštění testů) a odešle informaci. Po té se klient může ukončit. Neboť další komunikace již probíhat nemusí.

Pro ukládání informací o průběhu testu lze využít možnost odesílání požadavků na kontrolér přes API. Tento kontrolér již zpracuje požadavek a provede činnost vedoucí k uložení dat.

Jelikož průběh testu je navržen tak, aby byly ukládány jako posloupnost kroků, bylo třeba navrhnout desktopovou aplikaci, aby uměla tato data zpracovat a řídit běh testů přesně podle požadavků. Zde bylo třeba navrhnout architekturu tak, aby bylo možné předávat parametry absolutně univerzálně.

### 6.3 Design

Tato část se zabývá rozložením prvků na stránkách a také vhodným uspořádáním stránek.

K navržení designu byl použit nástroj Draw.io, který je online a je zdarma k použití. Bylo zde navrženo několik stránek, které se zdály být nezbytné. Tyto návrhy pomohly promyslet všechny důležité kroky, které vedly k výslednému designu. Cílem bylo vytvořit design jednoduchého stylu, který bude přehledný a uživatelsky přívětivý. Webová aplikace by měla být intuitivní, aby nevyžadovala speciální zaškolení, nebo tvorbu dokumentu, dle kterého by se uživatel musel řídit. Snahou bylo navrhnout stránky tak, aby bylo i začátečníkům, nezkušeným uživatelům, jasné, co je od uživatele vyžadováno a aby se mohl snadno zorientovat. Také aby byly reporty a výsledky testů pro něj snadno čitelné a pochopitelné.

Návrh designu, který byl vytvořen v Draw.io, byl přidán do přílohy (PŘÍLOHA P I: NÁVRH DESIGNU).

## 7 IMPLEMENTACE APLIKACE

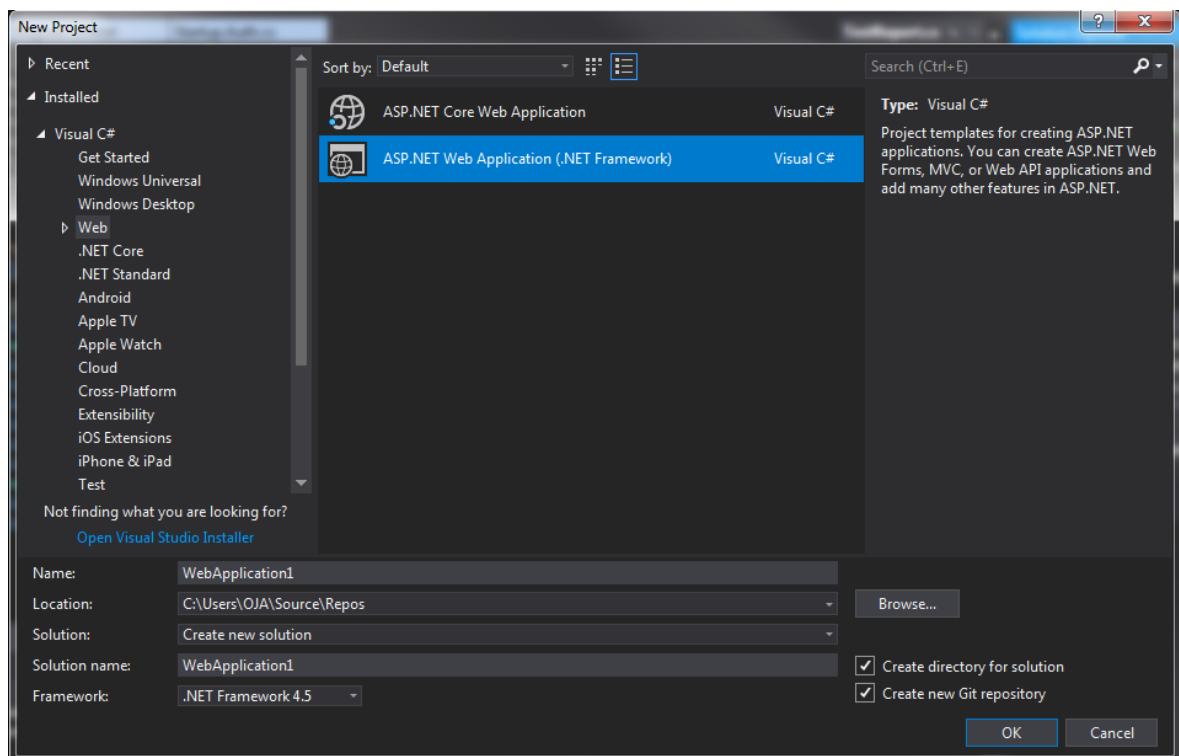
### 7.1 Prostředí pro vývoj

Pro tvorbu aplikací na platformě .NET je potřeba prostředí Microsoft Visual Studio. Pro vývoj byla použita verze Microsoft Visual Studio Community 2017, která je zdarma při dodržení licenčních podmínek uvedených na stránkách Microsoft (<https://visualstudio.microsoft.com/cs/license-terms/mlt553321/>). Tato licence osahuje zcela všechny funkce a nástroje, které pro vývoj aplikace byla potřeba.

Před vytvořením samotného projektu, bylo třeba doinstalovat některé komponenty do Visual Studia. K tomu slouží Visual Studio Installer. Jedná se o vývojové nástroje pro ASP.NET a web, .NET Framework, ve kterém chcete pracovat (pro projekt byl zvolen 4.5) a ASP.NET MVC.

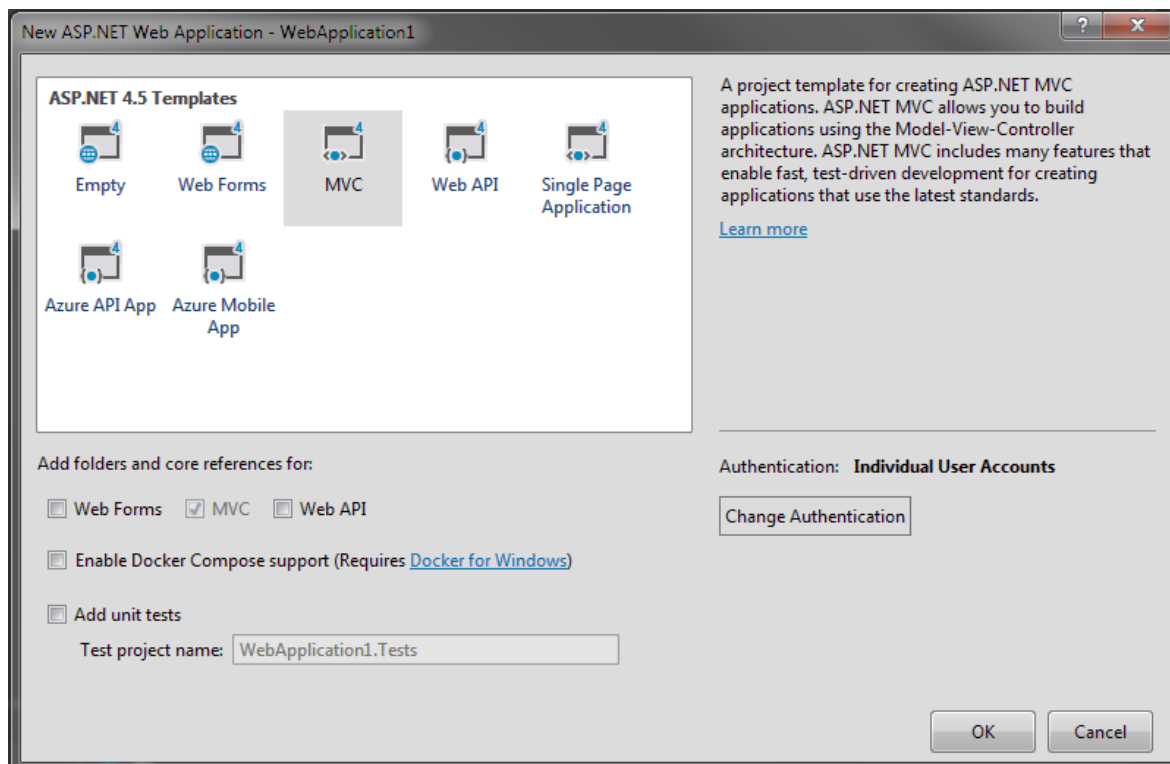
### 7.2 Vytvoření projektu

Pro vytvoření nového projektu v aplikaci byla vybrána šablona ASP.NET Web Application (.NET Framework) (viz Obrázek 6: Výběr šablony pro vytvoření projektu).



Obrázek 6: Výběr šablony pro vytvoření projektu

Po potvrzení výběru šablony se objeví ještě jedno okno pro výběr šablony ASP.NET 4.5. Zde bylo zvoleno MVC a autentizace byla změněna na „Individual User Accounts“ (viz. Obrázek 7:Výběr šablony MVC), aby bylo možné si vytvořit přihlašování uživatelů.



Obrázek 7:Výběr šablony MVC

Visual Studio vytvoří projekt, který je již možné spustit. Již neupravená šablona vypadá celkem přijatelně i z designového pohledu. Tato šablona byla tedy upravena dle požadavků.

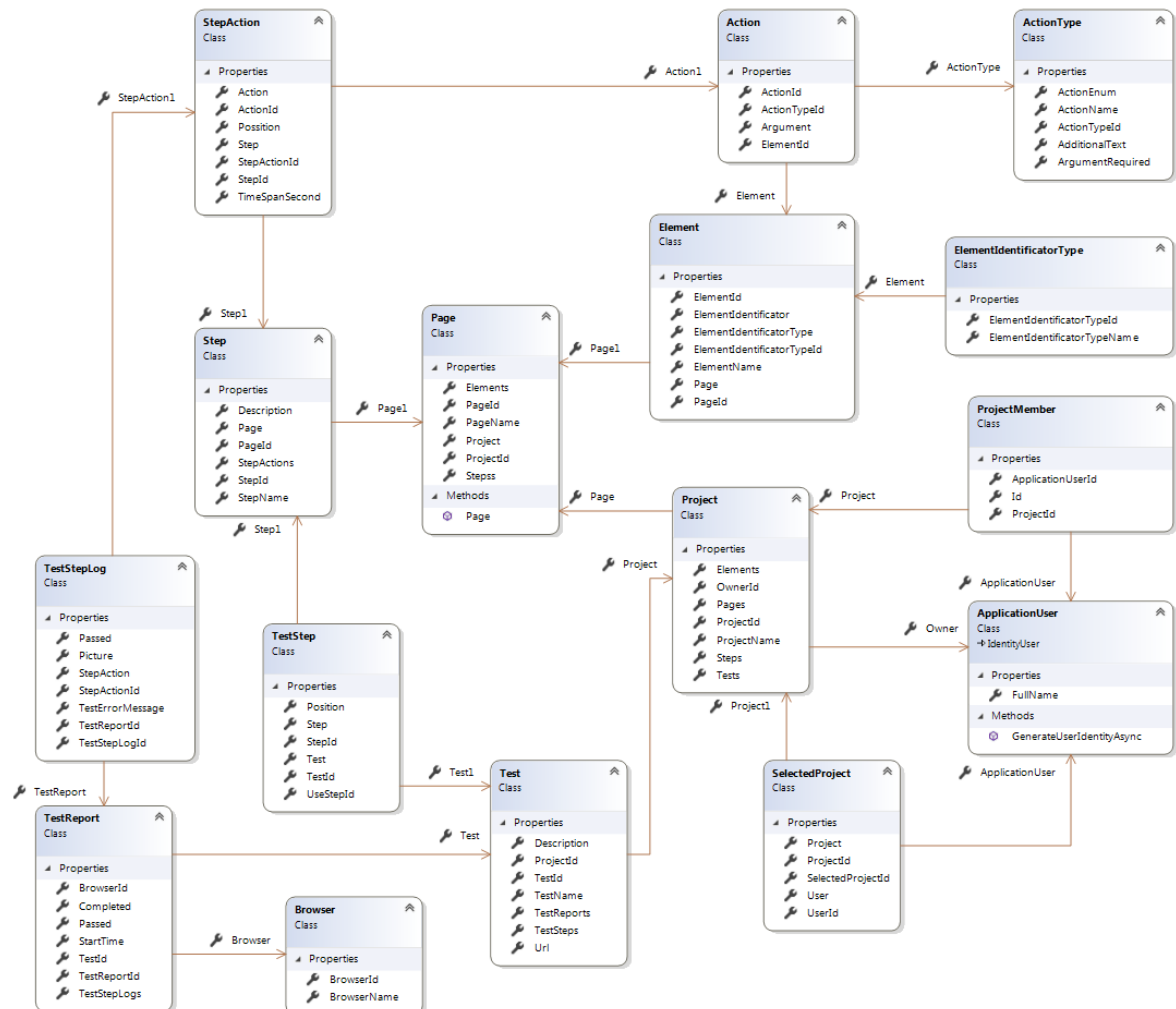
MVC bylo zvoleno, jelikož poskytuje výhody popsané v teorii. Při implementaci požadovaných funkcí, poskytuje MVC možnost oddělit návrh modelu, business logiku a vzhled stránky.

### 7.3 Instalace balíčků

Aby bylo možné využít knihovny, které pomáhají při vývoji aplikací, je třeba si nainstalovat je v rámci balíčků NuGet. První balíček, který byl potřeba, byl EntityFramework od Microsoft. Tento balíček umožňuje využít možnosti Entity Frameworku pro práci s databází.

## 7.4 Modely

Na základě návrhu byly vytvořeny objekty, které jsou potřeba pro správné fungování aplikace. Tyto modely jsem zobrazil v Obrázek 8: Diagram tříd použitých modelů.



Obrázek 8: Diagram tříd použitých modelů

U modelů je důležité myslet na to, aby obsahovaly všechny důležité parametry včetně ID, jelikož by Entity Framework nemohl vytvořit relaci s databází. Dále je potřeba, aby modely, které mají vztah s jiným modelem (např. model *Step* má vlastnost *Page*, což je další model) měli i vlastnost ID pro vytvoření vazby k modelu, se kterým takový vztah tvoří. Jako příklad použiju Obrázek 9: Třída *Step*, kde jsou okomentovány řádky, vedoucí k pochopení problematiky.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace Diplomka.Models
{
    [Table("Step")] //Atribut pro vytvoření databázové tabulky s názvem 'Steps'
    public class Step
    {
        [Key] // Atribut pro vytvoření primárního klíče
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)] //Atribut definující, že ID bude generováno automaticky
        public int StepId { get; set; }

        [MaxLength(120), Display(Name = "Step Name"), Required]
        public string StepName { get; set; }

        public string Description { get; set; }

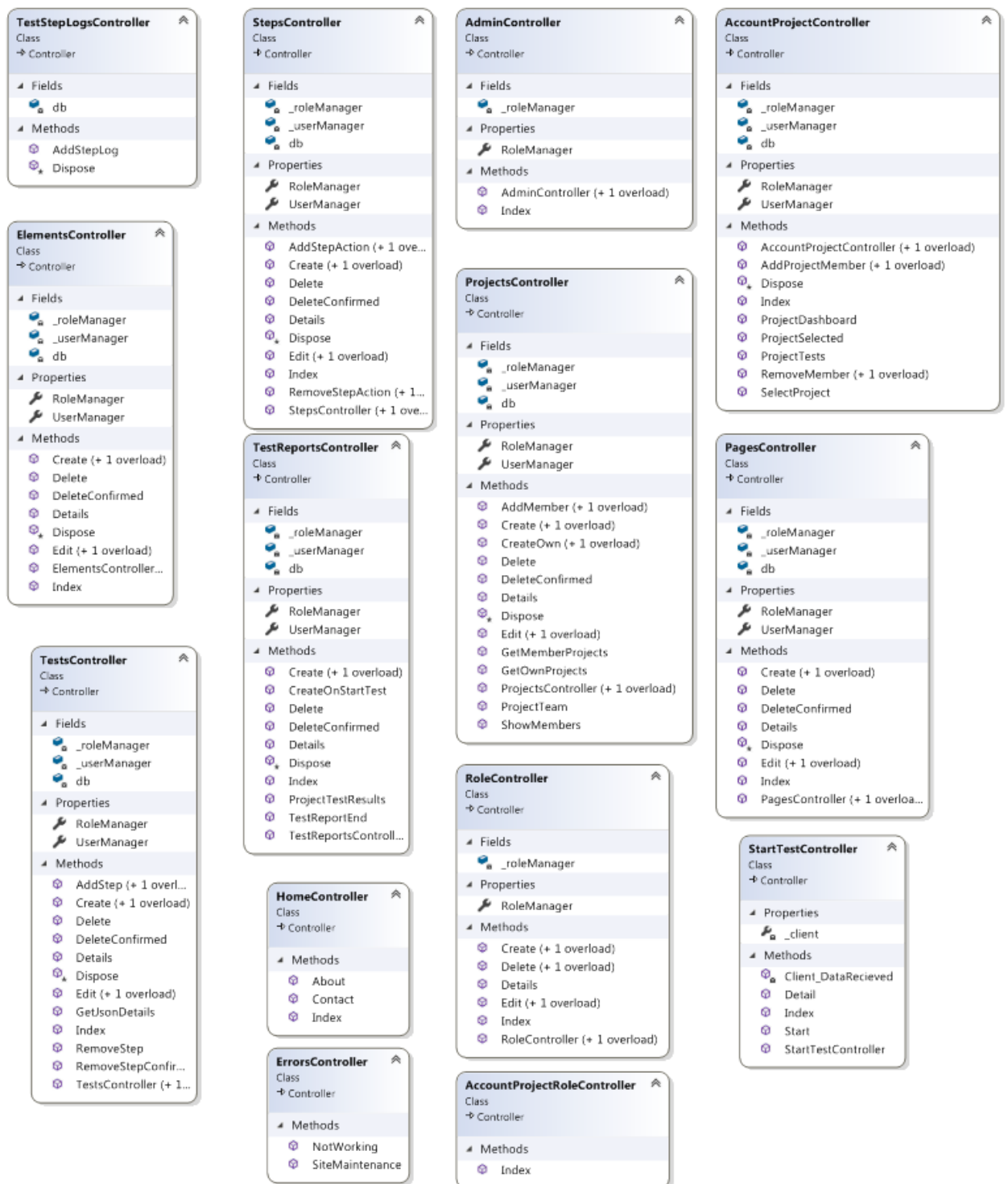
        [ForeignKey("Page"), Required] //Atribut pro vytvoření cizího klíče, který je povinný a vztahuje se k vlastnosti 'Page'
        public int PageId { get; set; }
        public virtual Page Page { get; set; } //Vlastnost Page je pouze virtuální, jelikož v databázi bude použit cizí klíč k identifikaci testu

        public virtual ICollection<StepAction> StepActions { get; set; }
    }
}
```

Obrázek 9: Třída Step

## 7.5 Kontroléry

Následně pro implementaci byznys logiky byly vytvořeny kontroléry. Vytvořené kontroléry si lze prohlédnout níže viz Obrázek 10: Diagram tříd kontrolérů.



Obrázek 10: Diagram tříd kontrolérů

Zde je nejen třeba zohlednit byznys logiku, ale je třeba myslet i na to, co je umožněno uživateli ve výsledku, aby viděl na stránkách (uživatelské rozhraní). Někdy je potřeba vytvořit prezentační modely. Tyto modely pak umožňují v uživatelském rozhraní (View) pracovat s daty, které byly v kontroléru připraveny. Jedním takovým příkladem je v této

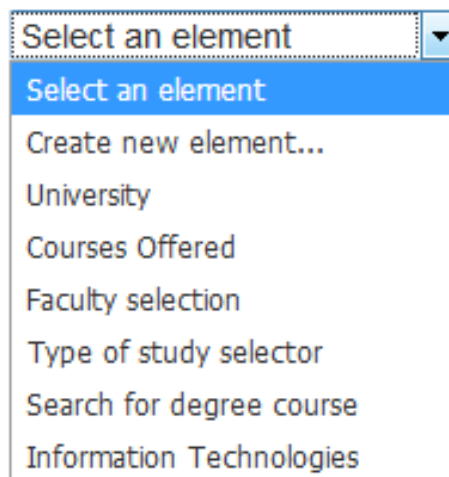
práci třeba *ElementSelectionViewModel* (viz Obrázek 11: Třída *ElementSelectionViewModel*).

```
namespace Diplomka.Models
{
    public class ElementSelectionViewModel
    {
        public ElementSelectionViewModel() { }
        public ElementSelectionViewModel(List<Element> elements)
        {
            _elements = elements;
        }
        private readonly List<Element> _elements;

        [Display(Name = "Selected Element")]
        public int? SelectedElementId { get; set; }
        public IEnumerable<SelectListItem> ElementItems
        {
            get
            {
                var items = new SelectList(_elements, "ElementId", "ElementName");
                return DefaultElementItems.Concat(items);
            }
        }
        private IEnumerable<SelectListItem> DefaultElementItems
        {
            get
            {
                return new List<SelectListItem>() {
                    new SelectListItem() { Text = "Select an element" },
                    new SelectListItem() { Text = "Create new element...", Value = "new element" } };
            }
        }
    }
}
```

Obrázek 11: Třída *ElementSelectionViewModel*

V tomto modelu byla vytvořena vlastnost *ElementItems*, která vrací prvky pro vytvoření listu s elementy. Zde bylo přidáno také privátní vlastnost *DefaultElementItems*, díky které je možné do listu přidat prvky do listu, o které lze tento list rozšířit. Výsledný prvek, který lze díky této vlastnosti vytvořit je na Obrázek 12: Výběr elementu.



Obrázek 12: Výběr elementu

V kontroléru, je třeba pro danou funkci tento model předat jako návratovou hodnotu zabalenou do objektu *View*, případně *PartialView*. V tomto případě bylo použito *PartialView*, neboť výsledné zobrazení pro tuto funkci bude pouze jako součást jiného zobrazení. Tomu odpovídá i návratový typ *PartialViewResult*.

```
public PartialViewResult ProjectElements(int id)
{
    var elements = db.Elements.Where(e => e.ProjectId == id).ToList();
    var list = new ElementSelectionViewModel(elements);
    ViewBag.ProjectId = id;
    return PartialView(list);
}
```

Obrázek 13: Funkce pro získání elementů pro projekt, který je identifikovaný vstupní hodnotou id

## 7.6 View

Soubory pro grafická rozhraní v ASP.NET MVC Framework nesou příponu „.cshtml“. Tyto soubory jsou známé pod názvem Razor stránky. Syntaxe Razor umožňuje použít při vytváření html kódu pro výsledný vzhled stránky také funkce C#, díky čemuž lze vkládat a předpřipravit výsledné HTML stránky, které se odešlou uživateli k zobrazení.

Pokud je požadováno vložit funkci, nebo proměnnou, musí být použit klíčový znak „@“. Za tímto znakem pak následuje požadovaný výraz. Například pro výpočet 1+1 by byl použit výraz „@(1+1)“ a výsledkem by bylo, že by se před odesláním výsledného HTML souboru dosadila hodnota „2“ namísto použitého výrazu.

Pokud má být na Razor stránce použit výstup z kontroleru, je potřeba zahrnout definici modelu na této stránce. To lze provést za pomoci klíčového slova „@model“, za kterým bude následovat reference na model. Pro lepší pochopení je zde uveden příklad z této práce.

V kontroléru *ElementsController.cs* je metoda *Create*, která vrací model *Element*. Na Razor stránce proto je zde použito „@model Diplomka.Models.Element“.

Grafické rozhraní se nachází ve složce Views, ve které se nachází další složky. Tyto další složky nesou vždy název odpovídající kontroleru. Jmenuje-li se kontroler *StepsController.cs*, název složky bude nést název odpovídající tomuto kontroleru jen s rozdílem, že se vynechává část „Controller.cs“, tedy složka grafických rozhraní pro zvolený kontroler bude mít název Steps. Cesta ke grafickému rozhraní je dána vzorem *View/[název kontroléru]/[název metody]*. Cesta ke grafickému rozhraní pro metodu *TestSteps*, která se nachází v kontroleru s názvem *StepsController.cs*, je tedy *View/Steps/TestSteps.cshtml*.

Pokud však Razor stránka pro danou metodu ještě nebyla vytvořena, lze vytvořit pohled tak, že se na metodu klikne pravým tlačítkem myši a z nabídky se vybere možnost „Add View...“. Následně vyskočí okno, kde je možnost volby pro vybrání šablony. Díky této možnosti se vygeneruje soubor, který je připraven ihned k použití bez dalších úprav. Pokud je požadováno design dále přizpůsobit, není problém tak učinit. Dále v tomto okně lze zvolit možnosti, jako je vytvoření souboru, který bude použitelný jen jako částečné zobrazení. V jiném případě lze použít možnost, že stránka bude používat layout stránku.

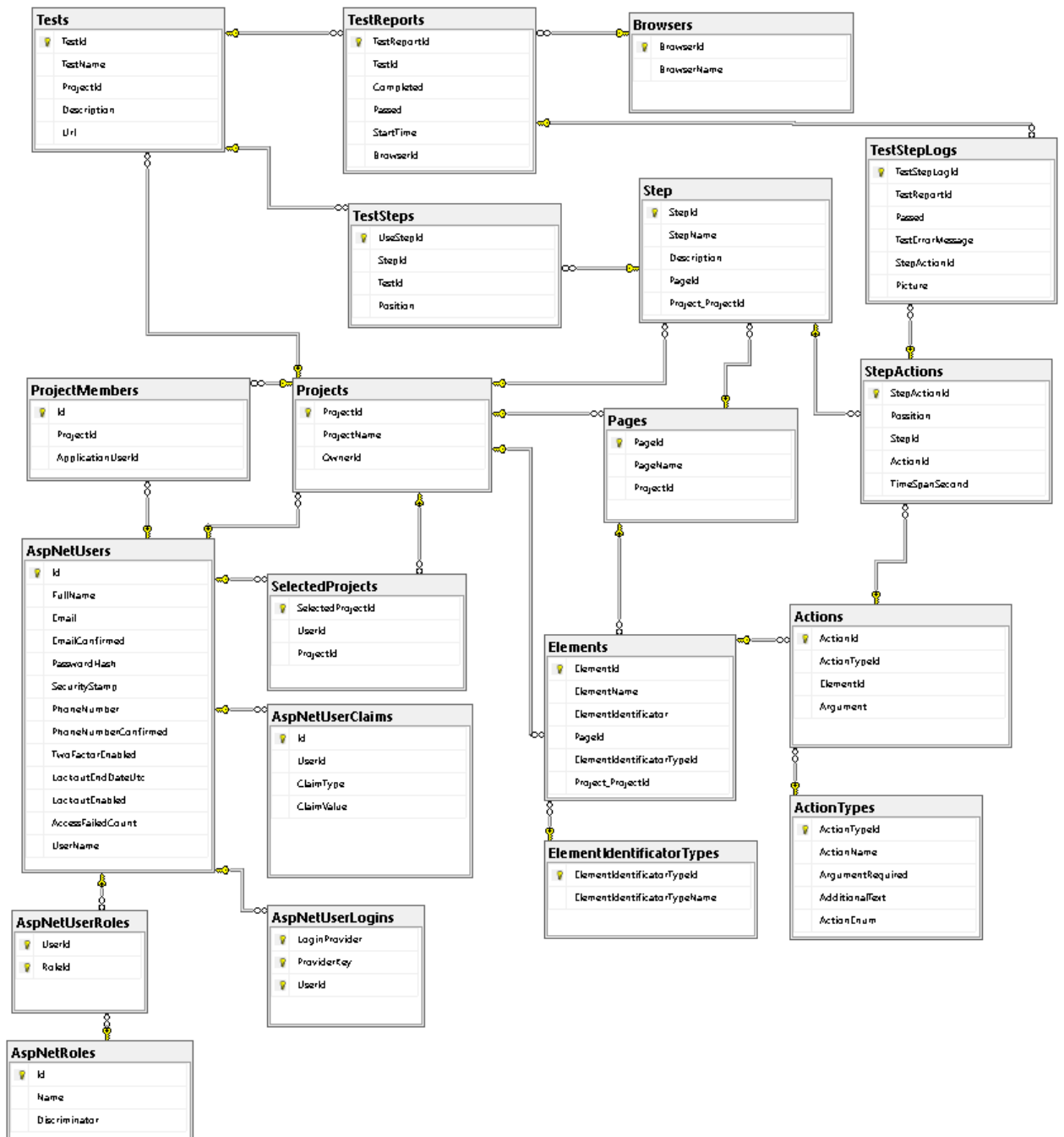
## 7.7 Update databáze

Před samotným spuštěním napsané aplikace je však potřeba vytvořit migrace pro nacheystání databáze k použití a následně tuto databázi updatovat, aby se tato migrace provedla. Pro tento krok bude potřeba *Konzole správce balíčků*. Tu lze najít ve VS v menu *Nástroje > Správce balíčků NuGet > Konzola správce balíčků*.

V této konzoli je nejprve potřeba povolit migrování. To lze provést zadáním příkazu *enable-migration*. Následně je potřeba vytvořit migraci a pojmenovat ji. Pro vytvoření *Initial* migrace se používá příkaz *add-migration Initial*. Tento příkaz automaticky vygeneruje soubor, který obsahuje instrukce popisující činnosti, které se mají provést při update databáze. V některých případech je potřeba udělat v tomto souboru úpravy. Další krok je update databáze. Pro tuto akci se používá příkaz *update-database*. Že byl update databáze

proveden úspěšně, lze poznat z konzole, kde by se měla zobrazit zpráva „Running Seed method.“.

Níže je možnost vidět diagram databáze, která byla vytvořena pro tuto aplikaci (Obrázek 14: Diagram databáze).



Obrázek 14: Diagram databáze

## 8 NASAZENÍ APLIKACE

Webová i desktopová aplikace byla nasazena na vlastní server. Díky přidělené statické IP adrese poskytovatelem internetového připojení bylo možno zveřejnit projekt na internetu. Adresa webových stránek byla nastavena na *http://192.168.31.106:50505/DiplomkaTest*.

### 8.1 Parametry serveru

Jako server byl použit starší notebook (Hewlett-Packard HP 620), na který byl nainstalován Windows Server 2016. Notebook má RAM 4GB a procesor Intel® Core™2 Duo T6570 2,1GHz.

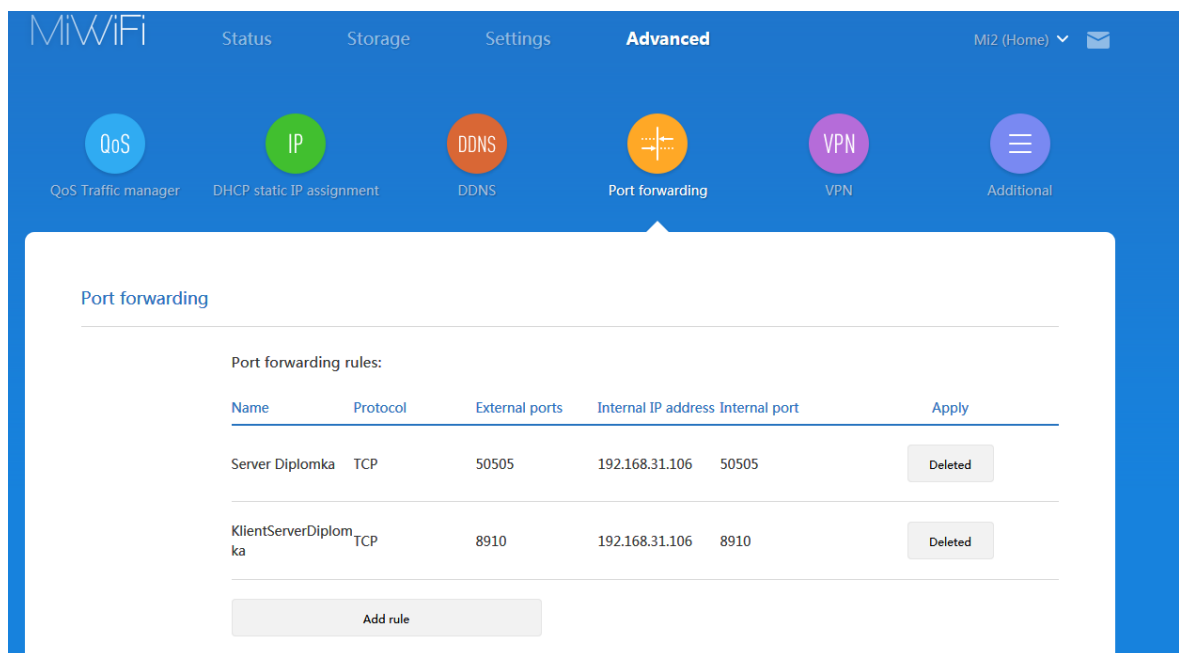
Server je připojen do místní sítě pomocí LAN kabelu k WiFi routeru Shiaomi (Mi Router 3(R3)). Připojení internetu poskytuje INTERNEXT 2000, s.r.o. a to o rychlosti stahování až 100Mbps a rychlosti nahrávání až 25Mbps.

### 8.2 Zveřejnění aplikace

Na serveru byl vytvořen databázový server, který program využívá k ukládání dat. Dále zde byla přenesena složka s hotovým programem. Ten byl nasazen pomocí služby IIS (viz příloha P 2).

Na serveru byla potřeba povolit použitý port pro aplikaci ve službě firewall. Postup, který byl zvolen, je popsán v příloze P 3. Tento krok měl za následek, že web byl přístupný v rámci sítě LAN. K tomu, aby byl web dostupný v rámci internetu, bylo třeba vykonat ještě jeden krok.

Posledním krokem je nastavení WiFi routeru, kde bylo potřeba povolit port aplikace, jako přístupný z veřejné sítě. Pro použitý router bylo třeba otevřít adresu routeru v prohlížeči, kde je možno tyto nastavení provádět. Po přihlášení bylo třeba přejít do sekce „Advanced“ a poté do „Port forwarding“ (viz Obrázek 15: Nastavení routeru).



Obrázek 15: Nastavení routeru

Po kliknutí na možnost „Add rule“ se objevilo okno (viz Obrázek 16: Vytvoření nového pravidla pro povolení portu z internetu), kde je třeba vyplnit údaje k povolení a přístupu k webu. Je třeba vyplnit název, vybrat TCP protokol. Jako Externí port je třeba vybrat port, pod kterým bude možné z internetu přistupovat. V mém případě jsem Interní i Externí číslo portu zanechal totožné. Pro vyplnění Interní IP adresy je třeba mít zjištěnou IP adresu serveru, na které se nachází a tuto vložit.

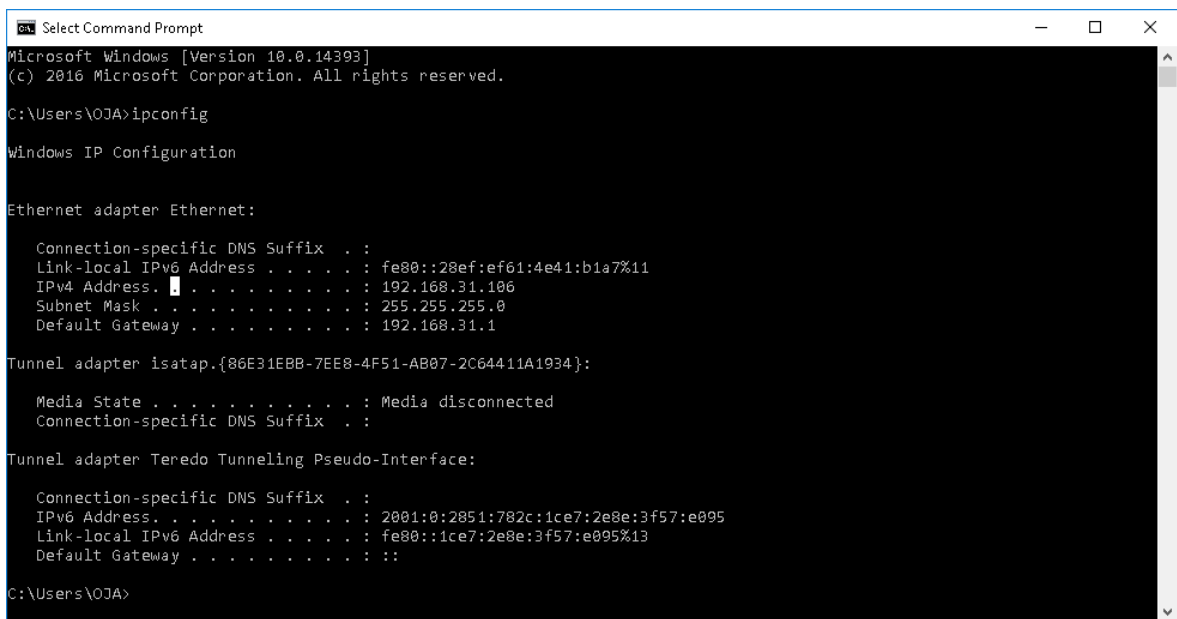
✕Create port forwarding rules

|                                     |   |
|-------------------------------------|---|
| Web                                 | Name  |
| TCP                                 | Protocol <span style="font-size: 0.8em;">▼</span> |
| 50505                               | External ports                                    |
| Internal IP address:<br>192.168.31. | 106   |
| 50505                               | Internal port                                     |

Install

Obrázek 16: Vytvoření nového pravidla pro povolení portu z internetu

Pro zjištění IP adresy slouží příkaz „ipconfig“ v příkazové řádce (viz Obrázek 17: ipconfig), kterou je třeba spustit na požadovaném zařízení. Zde je řádek s hodnotou IPv4 Address, která bude použita.



```
Select Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\VOJA>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::28ef:ef61:4e41:b1a7%11
    IPv4 Address. . . . . : 192.168.31.106
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.31.1

Tunnel adapter isatap.{86E31EBB-7EE8-4F51-AB07-2C64411A1934}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Teredo Tunneling Pseudo-Interface:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2001:0:2851:782c:1ce7:2e8e:3f57:e095
    Link-local IPv6 Address . . . . . : fe80::1ce7:2e8e:3f57:e095%13
    Default Gateway . . . . . : ::

C:\Users\VOJA>
```

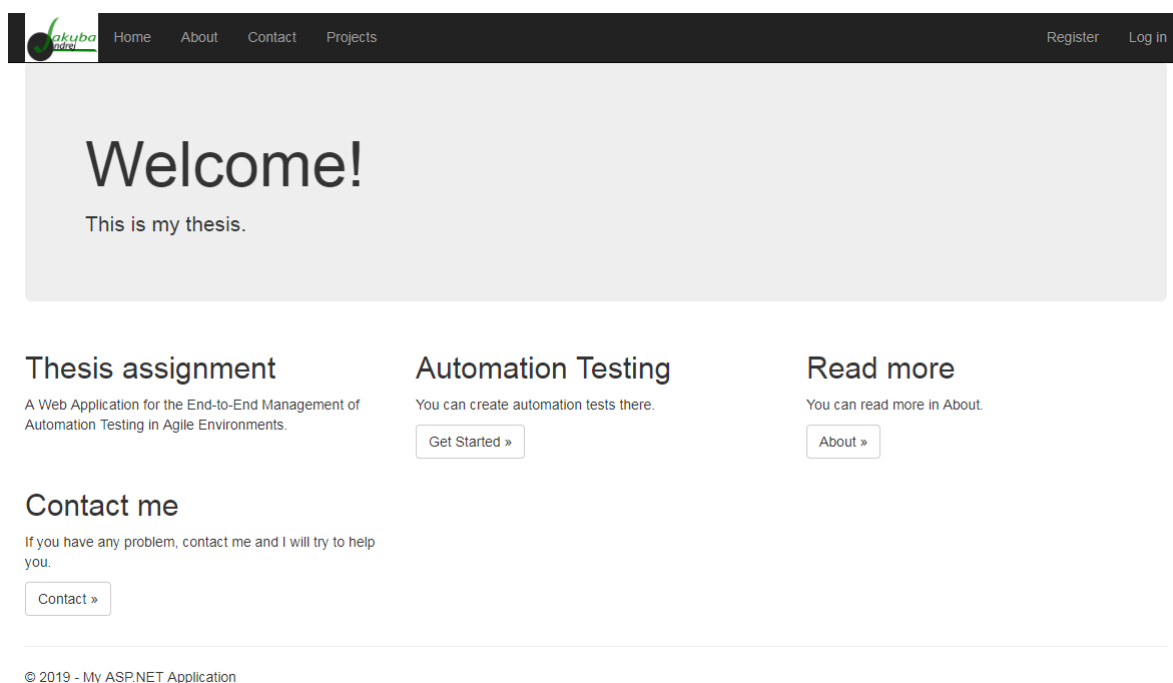
Obrázek 17: ipconfig

## 9 UKÁZKA APLIKACE

V této části lze nalézt názornou ukázkou, jak aplikace funguje. Webová aplikace je dostupná na adrese <http://192.168.31.106:50505/DiplomkaTest>. Tutoriál pro tvorbu testů lze nalézt v příloze.

### 9.1 Úvodní stránka

Po otevření aplikace se uživatel ocitne na úvodní stránce, kde nalezne základní odkazy. Pomocí nabídky je uživatel schopen proklikat k vytváření testů.



Obrázek 18: Úvodní stránka

### 9.2 Vytvoření testu

Test je vytvořen vždy v rámci projektu. Test může být vytvořen například i business zástupcem, který klade požadavky na kvalitu. Především jde o osobu, která poskytuje informace o tom, jak by měl produkt fungovat. Tuto představu může tedy formulovat při vytváření testu (viz Obrázek 19: Přidání testu do projektu).

Dashboard Projects Show Tests Team Hello test1! Log off

## Create Test

**Test Name** Search people

**Description** As a guest I want to search a person

**Uri** https://www.utb.cz/en

Create

[Back to Test List](#)

Obrázek 19: Přidání testu do projektu

To že byl test úspěšně přidán, pozná uživatel tak, že je přesměrován na stránku pro přidávání kroků. Dále jej uvidí v přehledu testů (viz Obrázek 20). Nyní je třeba, aby uživatel do testu přidal kroky, které má test provádět. Bez těchto kroků test pouze otevře stránku a tím test končí.

Dashboard Projects Tests Steps Pages Elements Team Hello Ojal Log off

## Tests

[Create New](#)

| Project Name | Test Name                                |  |
|--------------|--|--|
| UTB          | Proklikání ke zboží "Triko černé Be In"  | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>   <a href="#">Run test</a> |
| UTB          | Student wants to go to Wi-Fi síť eduroam | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>   <a href="#">Run test</a> |

Obrázek 20: Přehled testů v projektu

Pro příklad byl vytvořen test stránek UTB, kde je cílem proklikat se ke zboží „Triko černé Be In“ (Obrázek 21: Detail testu).

Dashboard Projects Tests Steps Pages Elements Team Hello Ojal Log off

## Details Test

**Test Name** Proklikání ke zboží "Triko černé Be In"

**Description** Jako návštěvník webu UTB, bych měl mít možnost dohledat zboží pod názvem "Triko černé Be In".

**Uri** http://www.utb.cz/

Steps:

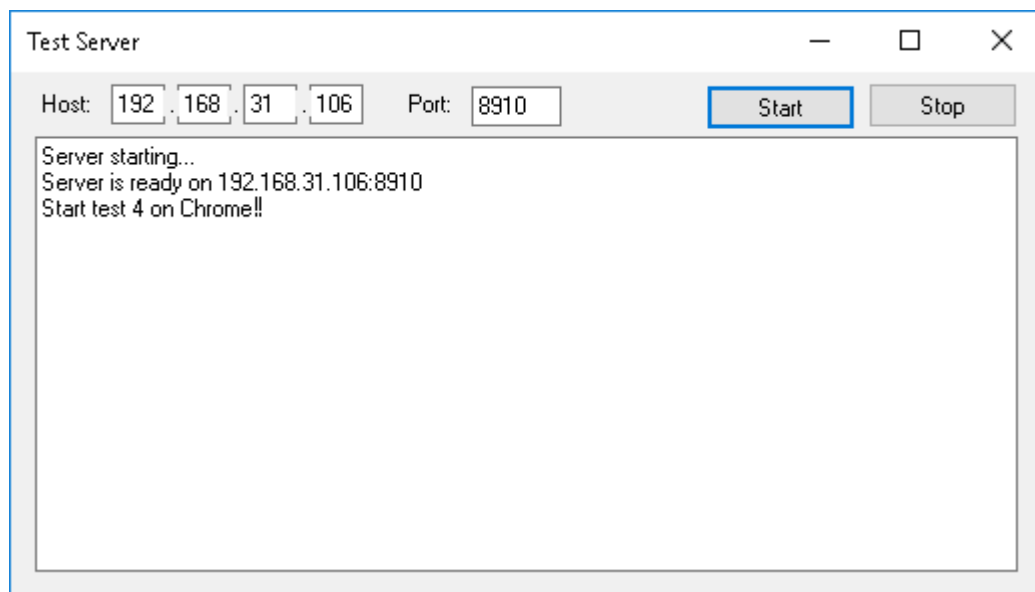
| Step name                           | Actions                |
|-------------------------------------|------------------------|
| Výběr sekce "Student"               | <a href="#">Remove</a> |
| Přejít na "Mikiny a trička"         | <a href="#">Remove</a> |
| Přejít na "Stylové mikiny a trička" | <a href="#">Remove</a> |
| Zobraz produkt "Tričko černé be in" | <a href="#">Remove</a> |
| <a href="#">Add step</a>            |                        |

[Edit](#) | [Back to List](#)

Obrázek 21: Detail testu

### 9.3 Spouštění a výsledek testů

Spuštění testu je možné ze sekce „Tests“ (viz Obrázek 20: Přehled testů v projektu), kde po kliknutí na možnost „Run test“ se objeví stránka pro výběr prohlížeče, na kterém test má být spuštěn. Po potvrzení proběhne spojení a komunikace s aplikací pro spuštění testů (Obrázek 22: Aplikace pro spuštění testů). Tato aplikace na základě obdržené zprávy o spuštění testu, provede test.



Obrázek 22: Aplikace pro spuštění testů

Po zahájení testu se vytvoří report, který lze i v průběhu testu sledovat, a mít tak přehled o tom, jak test běží. Výsledek testu se nachází v sekci „Dashboard“ (viz Obrázek 23: DashboardObrázek 24).

| Browser Name | Test Name                               | Completed | Passed | StartTime           |  |
|--------------|---|-----------|--------|---------------------|--|
| Chrome       | Proklikání ke zboží "Triko černé Be In" | ✘         | ✘      | 26/08/2019 19:27:54 | <a href="#">Details</a>   <a href="#">Delete</a> |

Obrázek 23: Dashboard

Po rozkliknutí možnosti „Details“ si lze prohlédnout výsledek krok po kroku (viz Obrázek 24: Výsledek spuštěného testu).

**Dashboard** Projects Tests Steps Pages Elements Team

## Test Report

[Back to List](#)

Proklikání ke zboží "Triko černé Be In"

|              |                     |
|--------------|---------------------|
| Browser Name | IE                  |
| Completed    | ✓                   |
| Passed       | ✓                   |
| StartTime    | 26/08/2019 02:33:58 |

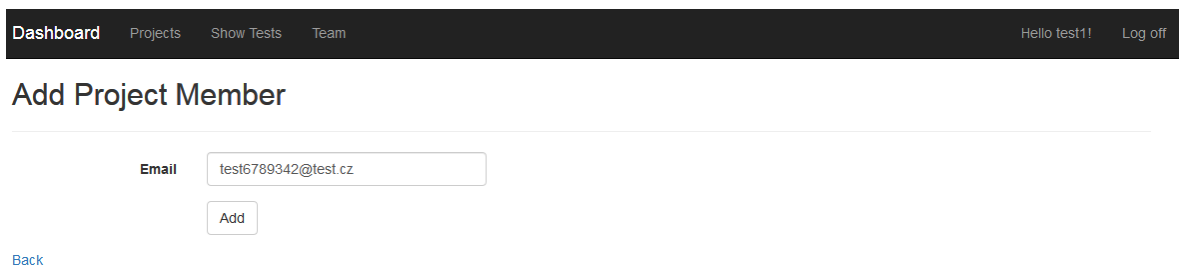
---

- ✓ **Výběr sekce "Student"**
  - ✓ Assert that element is displayed | Element: **Student**
  - ✓ Click on **Student**
- ✓ **Přejít na "Mikiny a trička"**
  - ✓ Assert that element is displayed | Element: **Mikiny a trička**
  - ✓ Move mouse on **Mikiny a trička**
  - ✓ Click on **Mikiny a trička**
- ✓ **Přejít na "Stylové mikiny a trička"**
  - ✓ Move mouse on **STYLOVÉ OBLEČENÍ**
  - ✓ Click on **STYLOVÉ OBLEČENÍ**
- ✓ **Zobraz produkt "Tričko černé be in"**
  - ✓ Click on **Triko černé Be In**

Obrázek 24: Výsledek spuštěného testu

## 9.4 Přidání dalšího uživatele do projektu

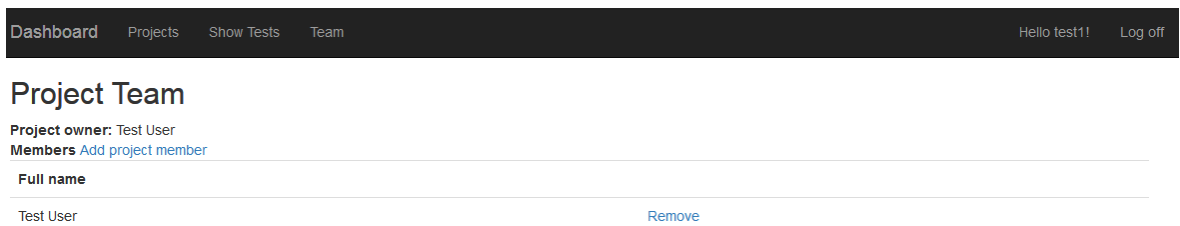
Pro přidání dalšího uživatele do projektu je potřeba kliknout na *Team* v horním menu (uživatel musí mít vybraný projekt). Zde je možnost *Add project member*, na kterou je potřeba kliknout. Aby bylo možné přidat dalšího uživatele, je potřeba, aby tento uživatel byl zaregistrovaný v této aplikaci. Je-li uživatel registrovaný, lze jeho přidání provést zadáním jeho emailu a stiskem tlačítka *Add* (viz Obrázek 25).



The screenshot shows a web application interface with a dark navigation bar at the top containing 'Dashboard', 'Projects', 'Show Tests', and 'Team'. On the right side of the bar, it says 'Hello test1!' and 'Log off'. Below the navigation bar, the main heading is 'Add Project Member'. There is a form with an 'Email' label and a text input field containing 'test6789342@test.cz'. Below the input field is an 'Add' button. A 'Back' link is located to the left of the form.

Obrázek 25: Přidání uživatele do projektu

O tom, že je uživatel již součástí týmu se lze ujistit tak, že jej lze vidět mezi členy týmu (viz Obrázek 26).



The screenshot shows a web application interface with a dark navigation bar at the top containing 'Dashboard', 'Projects', 'Show Tests', and 'Team'. On the right side of the bar, it says 'Hello test1!' and 'Log off'. Below the navigation bar, the main heading is 'Project Team'. Underneath, it says 'Project owner: Test User' and 'Members Add project member'. There is a table with one row. The table has a header row with 'Full name' and a 'Remove' button. The row below contains 'Test User' and 'Remove'.

Obrázek 26: Tým na projektu

## ZÁVĚR

Motivací pro tuto diplomovou práci bylo získat zkušenosti s tvorbou webové aplikace v technologii ASP.NET a možnost vytvořit aplikaci, na která bude mít přínos. Cílem bylo, aby aplikace pomohla při práci testerům, kteří nemají mnoho zkušeností s kódováním a umožnila jim tak testovat aplikace automatizovaně.

Při vývoji software je nedílnou součástí i testování kvality vyvíjeného software. Díky testování si lze být více jistý výsledkem, který má být prezentován zákazníkům. Odhalení chyby zákazníkem sice může být přínosné pro obě strany, nicméně pokud zákazník objeví chybu až příliš mnoho, byl by značně znepokojen.

V agilním prostředí tedy pro testování software použijeme automatizovaných testů k pokrytí funkcionalit v dostatečné míře, aby tak testy šetřili čas i práci. Nelze však zapomenout i na testy manuální, které jsou vhodné například při testování nových funkcionalit.

Nástroje, které lze pro automatizaci testů použít, existuje mnoho. Důležité je shrnout si informace o tom, jak a co je potřeba testovat. Vhodnou volbou nástroje totiž lze výrazně ovlivnit efektivitu práce.

Implementace aplikace se částečně liší od návrhu. Tato skutečnost je dána poznatkami, které přicházely až s tvorbou samotné aplikace. V zásadě to ale výsledek nijak neovlivňuje. Aplikace je napsaná v anglickém jazyce, neboť jde o jazyk, který je světově rozšířený a aplikaci tak může používat širší spektrum lidí.

Pro tvorbu testů byly vybrány akce, které by měly pokrýt většinu testů. Pokud by byl požadavek na nějaké další akce, bylo by možné aplikaci doplnit. Testy se spouští na prostředí, které je společné pro webovou aplikaci i databázový server. Pro umožnění testovat na více různorodém prostředí by bylo možné obstarat další zařízení, na které by se aplikace také spustila. Údržba testů v této aplikaci se zdá být poměrně snadná, což v projektech, kde trvá vývoj delší dobu a probíhá v nich poměrně dost změn, je určitě důležité.

S výhledem do budoucna by šlo dodělat vychytávky, jako je streamování průběhu testu. Dále by mohla být přidána možnost nechat si posílat výsledky testů na email. Dashboard v budoucnu může obsahovat i grafy pro lepší přehled.

## SEZNAM POUŽITÉ LITERATURY

1. Software Engineering | Prototyping Model. *Geeks for geeks*. [Online] <https://www.geeksforgeeks.org/software-engineering-prototyping-model/>.
2. Software Engineering | Incremental process model. *Geeks for geeks*. [Online] <https://www.geeksforgeeks.org/software-engineering-incremental-process-model/>.
3. **Ghahrai, Amir**. Incremental Model. *Testing Excellence*. [Online] 2. 12 2018. <https://www.testingexcellence.com/incremental-model/>.
4. **PAL, SAYAN KUMAR**. Software Engineering | Spiral Model. *Geeks for geeks*. [Online] <https://www.geeksforgeeks.org/software-engineering-spiral-model/>.
5. **Mgr. Jiří MARTINŮ, doc.Ing.Petr ČERMÁK, Ph.D.** *METODIKY VÝVOJE SOFTWARE*. Olomouc : Moravská vysoká škola Olomouc, o. p. s., 2018.
6. Software Engineering | Rapid application development model (RAD). *Geeks for geeks*. [Online] <https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/>.
7. **Beck, Kent, a další, a další**. Manifest Agilního vývoje software. *Agile manifesto*. [Online] 2001. <http://agilemanifesto.org>.
8. **PAL, SAYAN KUMAR**. Software Engineering | Agile Development Models. *Geeks for geeks*. [Online] <https://www.geeksforgeeks.org/software-engineering-agile-development-models/>.
9. **ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ**. *Řízení kvality softwaru: průvodce testováním*. Brno : Computer Pres, 2013. ISBN 978-80-251-3816-8.
10. Black Box Testing. *Software Testing Fundamentals*. [Online] <http://softwaretestingfundamentals.com/black-box-testing/>.
11. White Box Testing. *Software Testing Fundamentals*. [Online] <http://softwaretestingfundamentals.com/white-box-testing/>.
12. **Hlava, Tomáš**. Testování bílé a černé skříňky. *Testování softwaru*. [Online] 20. 8 2011. <http://testovanisoftwaru.cz/tag/white-box/>.
13. **Rex Black, Anders Claesson, Gerry Coleman, Bertrand Cornanguer, Istvan Forgacs, Alon Linetzki, Tilo Linz, Leo van der Aalst, Marie Walsh, and Stephan**

**Weber.** Foundation Level Extension Syllabus Agile Tester. *ISTQB*. [Online] 2014. <https://www.istqb.org/downloads/send/5-agile-tester-extension-documents/41-agile-tester-extension-syllabus.html>.

14. **Klaus Olsen (chair), Tauhida Parveen (vice chair), Rex Black (project manager), Debra Friedenberg, Matthias Hamburg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh, and Eshraka Zakaria.** Foundation Level Syllabus. *ISTQB*. [Online] 2018. <https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018-syllabus.html>.

15. **ISTQB – Test Levels. Get Software Services.** [Online] <https://www.getsoftwareservice.com/481/>.

16. **Unit Testing. Software Testing Fundamentals.** [Online] <http://softwaretestingfundamentals.com/unit-testing/>.

17. **Hlava, Tomáš.** Fáze a úrovně provádění testů. *Testování softwaru*. [Online] 21. 8 2011. <http://testovanisoftwaru.cz/metodika-testovani/druhy-typy-a-kategorie-testu/faze-testu/>.

18. **What is Regression Testing? Smartbear.** [Online] <https://smartbear.com/learn/automated-testing/what-is-regression-testing/>.

19. **SM, Rajkumar.** Automation Testing Vs Manual Testing | SoftwareTestingMaterial. *Software Testing Material*. [Online] 16. 3 2019. <https://www.softwaretestingmaterial.com/automation-testing-vs-manual-testing/>.

20. **Test Automation Framework. Techopedia.** [Online] <https://www.techopedia.com/definition/30670/test-automation-framework>.

21. **SM, Rajkumar.** Most Popular Test Automation Framework Interview Questions. *Software testing material*. [Online] 22. 4 2019. <https://www.softwaretestingmaterial.com/test-automation-framework-interview-questions/>.

22. **Rajkumar.** Types of Test Automation Frameworks | Software Testing Material. *Software testing material*. [Online] 2. 3 2019. <https://www.softwaretestingmaterial.com/types-test-automation-frameworks/#What-is-a-framework>.

23. **Types of Automation Frameworks. 3Qi Labs.** [Online] 19. 9 2014. <http://3qilabs.com/types-of-automation-frameworks/>.

24. Behavior Driven Testing in Automated testing . *Katalon*. [Online] <https://www.katalon.com/sa/behavior-driven-testing/>.
25. **Kagathara, Mehul**. Automated Testing using BDT (Behavior Driven Testing). *Infostretch*. [Online] 13. 2 2013. <https://www.infostretch.com/blog/automated-testing-using-bdt-behavior-driven-testing/>.
26. **Brian**. Best Automation Testing Tools for 2019 (Top 10 reviews). *Medium*. [Online] 26. 11 2017. <https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2>.
27. *SeleniumHQ*. [Online] <https://www.seleniumhq.org>.
28. Simplify API, Web, Mobile Automation Tests. *Katalon*. [Online] <https://www.katalon.com/>.
29. Unified Functional Testing (UFT). *Micro Focus*. [Online] <https://www.microfocus.com/en-us/products/unified-functional-automated-testing/overview>.
30. What is QTP/UFT Automation Testing Tool? *Guru99*. [Online] <https://www.guru99.com/uft-qtptest-automation-testing.html>.
31. TestComplete. *SMARTBEAR*. [Online] <https://smartbear.com/product/testcomplete/overview/>.
32. SoapUI. [Online] <https://www.soapui.org/>.
33. *Robot Framework*. [Online] <https://robotframework.org>.
34. *Ranorex*. [Online] <https://www.ranorex.com>.
35. Top 20 Best Automation Testing Tools in 2019 (Comprehensive List). *Software Testing Help*. [Online] 25. 4 2019. <https://www.softwaretestinghelp.com/top-20-automation-testing-tools/>.
36. Volba mezi .NET Core a .NET Framework pro serverové aplikace. *Microsoft*. [Online] 19. June 2018. <https://docs.microsoft.com/cs-cz/dotnet/standard/choosing-core-framework-server?toc=%2Faspnet%2Fcore%2Ftoc.json&bc=%2Faspnet%2Fcore%2Fbreadcrumb%2Ftoc.json&view=aspnetcore-2.2>.
37. Informace o webových aplikacích. *Adobe*. [Online] <https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html>.

38. **Čápka, David.** MVC architektura. *ITnetwork.cz.* [Online] <https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor>.
39. **Bernard, Borek.** Úvod do architektury MVC. *zdrojak.cz.* [Online] 7. May 2009. <https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/>.
40. **Smith, Steve.** Přehled ASP.NET Core MVC. *Microsoft.* [Online] 8. January 2018. <https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-2.2>.
41. Dokumentace Entity Framework. *Microsoft.* [Online] <https://docs.microsoft.com/cs-cz/ef/#pivot=entityfmwk&panel=entityfmwk1>.
42. Code-Based Migration in Entity Framework 6. *Entity Framework Tutorial.* [Online] <https://www.entityframeworktutorial.net/code-first/code-based-migration-in-code-first.aspx>.
43. Seriál Začínáme s ASP.NET. *dotnetportal.cz.* [Online] <https://www.dotnetportal.cz/clanky/serial/6/Zaciname-s-ASP-NET>.
44. **SPAANJAARS, Imar.** *Beginning ASP.NET 4.5 in C# and VB.* [editor] 1. Indianapolis : Wrox;, 2013. ISBN 9781118311806.
45. **ŠOCHOVÁ, Zuzana a Eduard KUNCE.** *Agilní metody řízení projektů.* Brno : Computer Press, 2014. str. 175. ISBN 978-80-251-4194-6.
46. **MYSLÍN, Josef.** *Scrum: průvodce agilním vývojem softwaru.* 1. Brno : Computer Press, 2016. str. 167. ISBN 978-80-251-4650-7.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

API Rozhraní pro programování aplikací (angl. Application Programming Interface)

ASP Active Server Pages

BDD Behavior Driven Development

BDT Behavior Driven Testing

SW Software

TDD Test Driven Development

**SEZNAM OBRÁZKŮ**

|  |    |
|--|----|
| Obrázek 1: Vodopádový model .....  | 14 |
| Obrázek 2: Prototypový model .....   | 15 |
| Obrázek 3: Inkrementální model .....   | 16 |
| Obrázek 4: Spirálový model .....   | 17 |
| Obrázek 5: MVC model.....  | 33 |
| Obrázek 6: Výběr šablony pro vytvoření projektu .....  | 39 |
| Obrázek 7:Výběr šablony MVC .....  | 40 |
| Obrázek 8: Diagram tříd použitých modelů .....   | 41 |
| Obrázek 9: Třída Step .....  | 42 |
| Obrázek 10: Diagram tříd kontrolérů.....   | 43 |
| Obrázek 11: Třída ElementSelectionViewModel .....  | 44 |
| Obrázek 12: Výběr elementu .....   | 45 |
| Obrázek 13: Funkce pro získání elementů pro projekt, který je identifikovaný vstupní<br>hodnotou id..... | 45 |
| Obrázek 14: Diagram databáze.....  | 47 |
| Obrázek 16: Nastavení routeru .....  | 49 |
| Obrázek 17: Vytvoření nového pravidla pro povolení portu z internetu .....                               | 50 |
| Obrázek 18: ipconfig .....   | 51 |
| Obrázek 19: Úvodní stránka .....   | 52 |
| Obrázek 20: Přidání testu do projektu .....  | 53 |
| Obrázek 21: Přehled testů v projektu.....  | 53 |
| Obrázek 22: Detail testu .....   | 53 |
| Obrázek 23: Aplikace pro spouštění testů .....   | 54 |
| Obrázek 24: Dashboard .....  | 54 |
| Obrázek 25: Výsledek spuštěného testu .....  | 55 |
| Obrázek 26: Přidání uživatele do projektu .....  | 56 |
| Obrázek 27: Tým na projektu .....  | 56 |

## **SEZNAM PŘÍLOH**

PŘÍLOHA P I: NÁVRH DESIGNU

PŘÍLOHA P II: SPUŠTĚNÍ PROJEKTU NA WINDOWS SERVERU

PŘÍLOHA P III: POVOLENÍ PORTU NA WINDOWS SERVERU

PŘÍLOHA P IV: TUTORIÁL

CD

# PŘÍLOHA P I: NÁVRH DESIGNU

LOGO

Home About

## Home

text

Sign In

User Name:

Password:

**SIGN IN**

[Forgot Password?](#)

---

New User

**SIGN UP**

LOGO

[Home](#) [About](#)

\*\*\*

Email

Password

Password again

Sign up

LOGO

[Tests](#) [Reports](#) [Profile](#)

\*\*\*

## Profile

Email

Password

Password again

Save changes

LOGO

Tests Reports Profile

\*\*\*  
**Welcome**

text

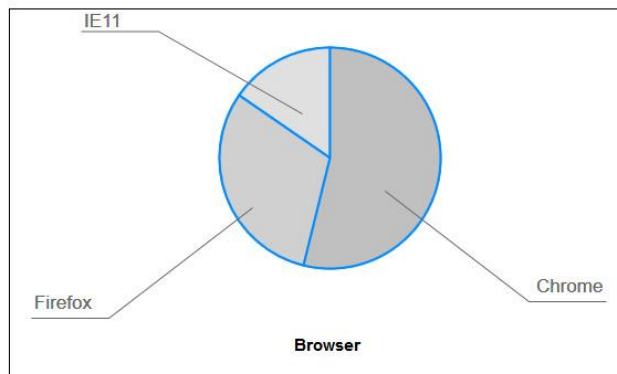
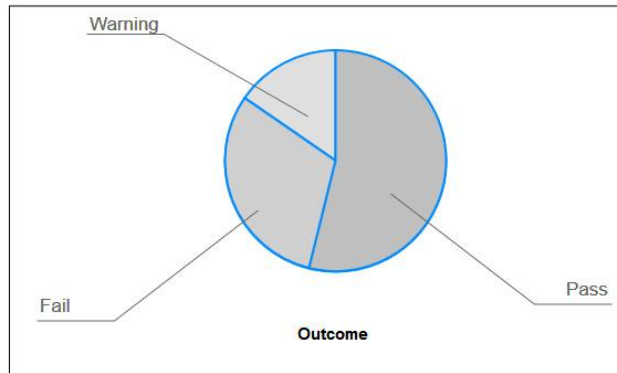
## Reports

### Last run

2018-10-16 15:42

### Logs

| Test name              | Outcome | Browser |
|------------------------|---------|---------|
| <a href="#">Test1</a>  | Passed  | Chrome  |
| <a href="#">Test2</a>  | Passed  | Chrome  |
| <a href="#">Test3</a>  | Passed  | IE11    |
| <a href="#">Test4</a>  | Failed  | Chrome  |
| <a href="#">Test5</a>  | Passed  | Firefox |
| <a href="#">Test6</a>  | Passed  | Chrome  |
| <a href="#">Test7</a>  | Passed  | Firefox |
| <a href="#">Test8</a>  | Passed  | Chrome  |
| <a href="#">Test9</a>  | Failed  | Chrome  |
| <a href="#">Test10</a> | Passed  | Firefox |
| <a href="#">Test11</a> | Passed  | Chrome  |
| <a href="#">Test12</a> | Warning | Chrome  |
| <a href="#">Test13</a> | Passed  | Chrome  |
| <a href="#">Test14</a> | Failed  | Chrome  |



## Test1 log

Started: 2018-10-17 17:32:16  
 Duration: 00:00:14  
 Browser: Chrome  
 Outcome: Pass

### Steps:

Open page *https://www.google.cz/*  
 Write *UTB* to *Find textbox*  
 Click on *Find by Google*  
 Assert text *UTB: Univerzita Tomáše Bati ve Zlíně* in *First result*

LOGO

[Tests](#) [Reports](#) [Profile](#)

---

## Tests

| ID | Test Name | Outcome | Last run   |                     |                      |                        |
|----|-----------|---------|------------|---------------------|----------------------|------------------------|
| 1  | Test1     | Failed  | 2016-08-21 | <a href="#">Run</a> | <a href="#">Edit</a> | <a href="#">Delete</a> |
| 2  | Test2     | Passed  | 2018-10-08 | <a href="#">Run</a> | <a href="#">Edit</a> | <a href="#">Delete</a> |
| 3  | Test3     | empty   | never      | <a href="#">Run</a> | <a href="#">Edit</a> | <a href="#">Delete</a> |

[Add new test](#)

LOGO

[Tests](#) [Reports](#) [Profile](#)

---

**Test name**

**Browser**

[Save](#)

---

### Test case steps

## PŘÍLOHA P II: SPUŠTĚNÍ PROJEKTU NA WINDOWS SERVERU

Jak postupovat při publikaci ASP.NET projektu, existuje i video-tutoriál dostupný pod odkazem <https://www.youtube.com/watch?v=PPaqVyBkwMk&t=295s>.

Kroky, které je třeba vykonat, ke spuštění projektu jsou následující:

1. Ve visual studiu kliknout pravým tlačítkem myši na ASP.NET projekt a vybrat z nabídky možnost „Publish...“.
2. Pro zveřejnění programu do složky je třeba, vybrat možnost „Folder“ a zvolit si cestu, do které složky tuto akci požadujeme.
3. Kliknutím na tlačítko „Publish“ proběhne operace sestavení projektu a jeho zveřejnění do požadované složky. O úspěšnosti procesu se lze dočíst v okně „Output“ (v programu VS).
4. Na Windows serveru je potřeba mít nainstalované balíčky „.NET Framework“ a „Internet Information Service (IIS)“. Tyto balíčky lze nainstalovat v „Control Panel\Programs\Programs and Features“ zvolením „Turn Windows features on or off“ v levé nabídce.

Add Roles and Features Wizard

### Select server roles

Before You Begin  
Installation Type  
Server Selection  
Server Roles  
Features  
Confirmation  
Results

Select one or more roles to install on the selected server.

#### Roles

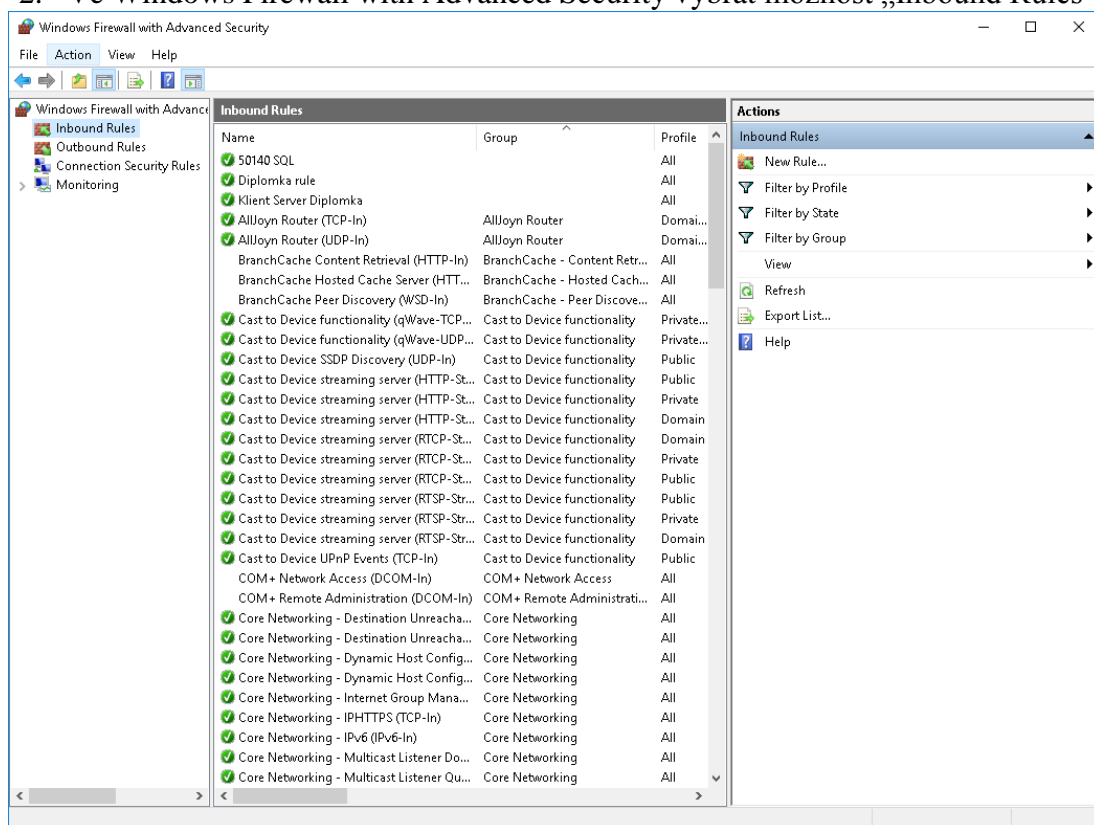
- Remote Access
- Remote Desktop Services
- Volume Activation Services
- Web Server (IIS) (18 of 43 installed)
  - Web Server (15 of 34 installed)
    - Common HTTP Features (4 of 6 installed)
      - Default Document (Installed)
      - Directory Browsing (Installed)
      - HTTP Errors (Installed)
      - Static Content (Installed)
      - HTTP Redirection
      - WebDAV Publishing
    - Health and Diagnostics (4 of 6 installed)
      - HTTP Logging (Installed)
        - Custom Logging
      - Logging Tools (Installed)
        - ODBC Logging
      - Request Monitor (Installed)
      - Tracing (Installed)
    - Performance (1 of 2 installed)
      - Static Content Compression (Installed)
      - Dynamic Content Compression
    - Security (1 of 9 installed)
      - Request Filtering (Installed)
      - Basic Authentication
      - Centralized SSL Certificate Support
      - Client Certificate Mapping Authentication
      - Digest Authentication
      - IIS Client Certificate Mapping Authentication
      - IP and Domain Restrictions
      - URL Authorization
      - Windows Authentication
    - Application Development (5 of 11 installed)
      - .NET Extensibility 3.5
      - .NET Extensibility 4.6 (Installed)
      - Application Initialization
      - ASP
      - ASP.NET 3.5
      - ASP.NET 4.6 (Installed)
      - CGI (Installed)
      - ISAPI Extensions (Installed)
      - ISAPI Filters (Installed)
      - Server Side Includes
      - WebSocket Protocol
  - FTP Server
  - Management Tools (3 of 7 installed)
    - IIS Management Console (Installed)
    - IIS 6 Management Compatibility
    - IIS Management Scripts and Tools (Installed)
    - Management Service (Installed)

5. Otevřít program „Internet Information Services (IIS) Manager“
6. Pravým tlačítkem myši kliknout na možnost „Sites“ v levém menu „Connections“ a zvolit možnost „Add website“.
7. V okně „Add website“ je potřeba vyplnit název sítě, zvolit umístění složky, ve které se nachází program, který jsme si ve VS zveřejnili. Dále kliknutím na tlačítko „Connect as...“ u cesty ke složce se objeví okno pro zadání jména a hesla pro účet, kterýmá potřebné oprávnění k této složce. V možnosti „Binding“ je potřeba nastavit „Port“ na hodnotu, která je na serveru volná. Kliknutím na tlačítko „OK“ se vytvoří nová síť na serveru.
8. Po označení vytvořené sítě v levém menu „Connections“ se zobrazí nabídka „Actions“ v pravé části okna. Zde je možnost otevřít stránku v prohlížeči volbou „Browse Website“

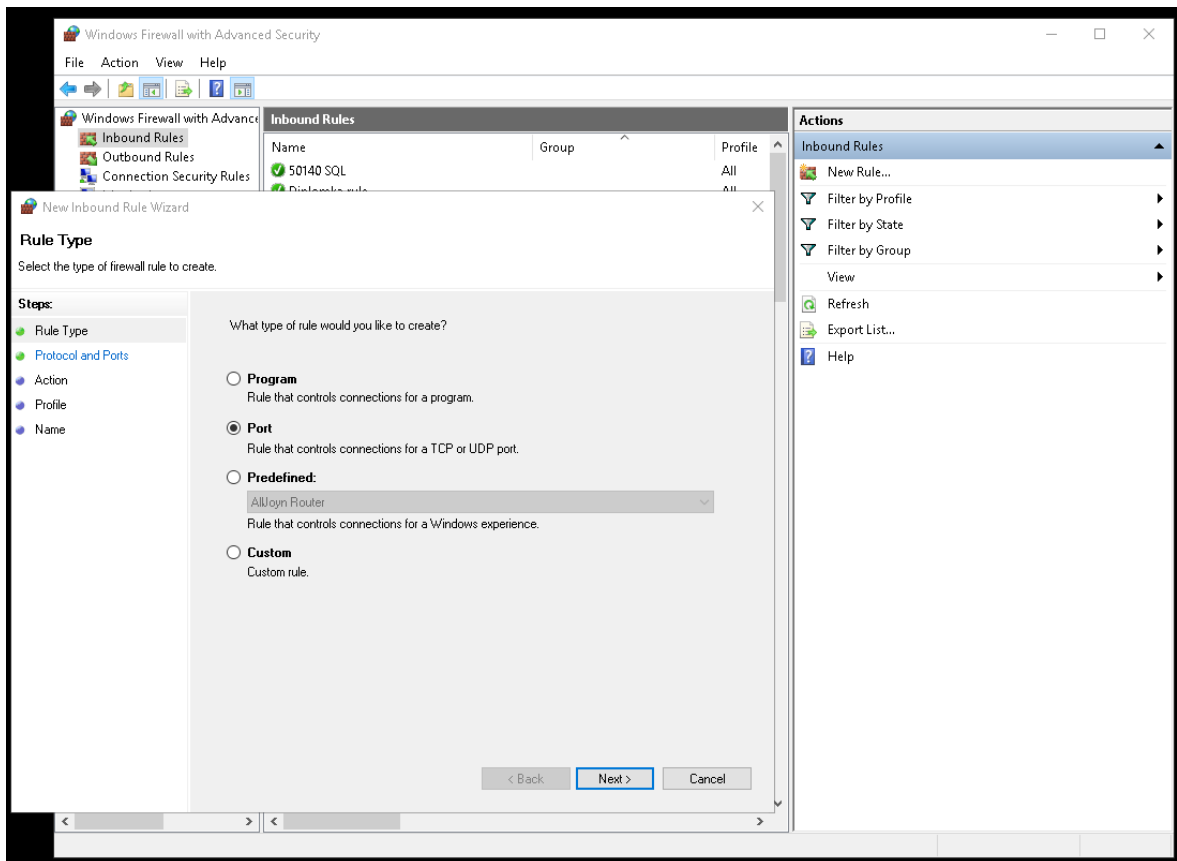
## PŘÍLOHA P III: POVOLENÍ PORTU NA WINDOWS SERVERU

Nasazený projekt (webovou aplikaci) není přístupný z jiného zařízení, pokud tato možnost není povolena ve Windows Firewall. Pro povolení komunikace z jiného zařízení je možné využít následující postup:

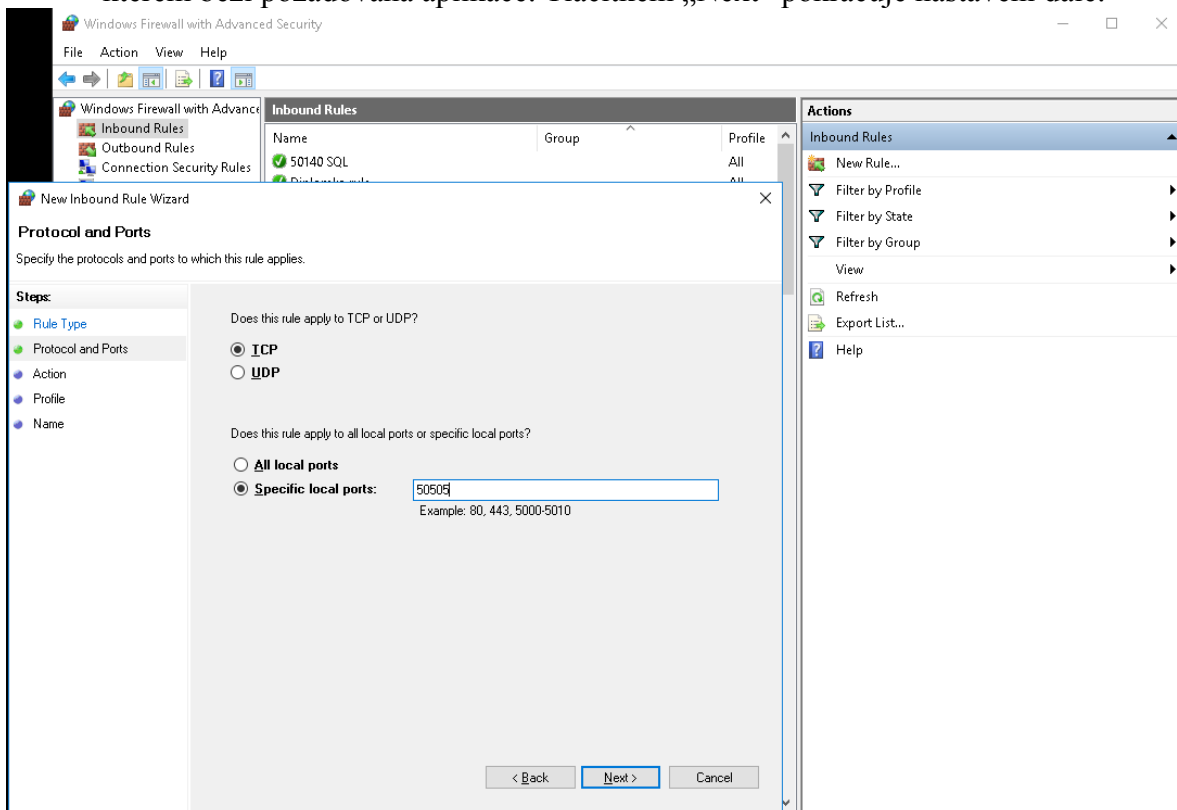
1. Otevřít Windows Firewall with Advanced Security
2. Ve Windows Firewall with Advanced Security vybrat možnost „Inbound Rules“



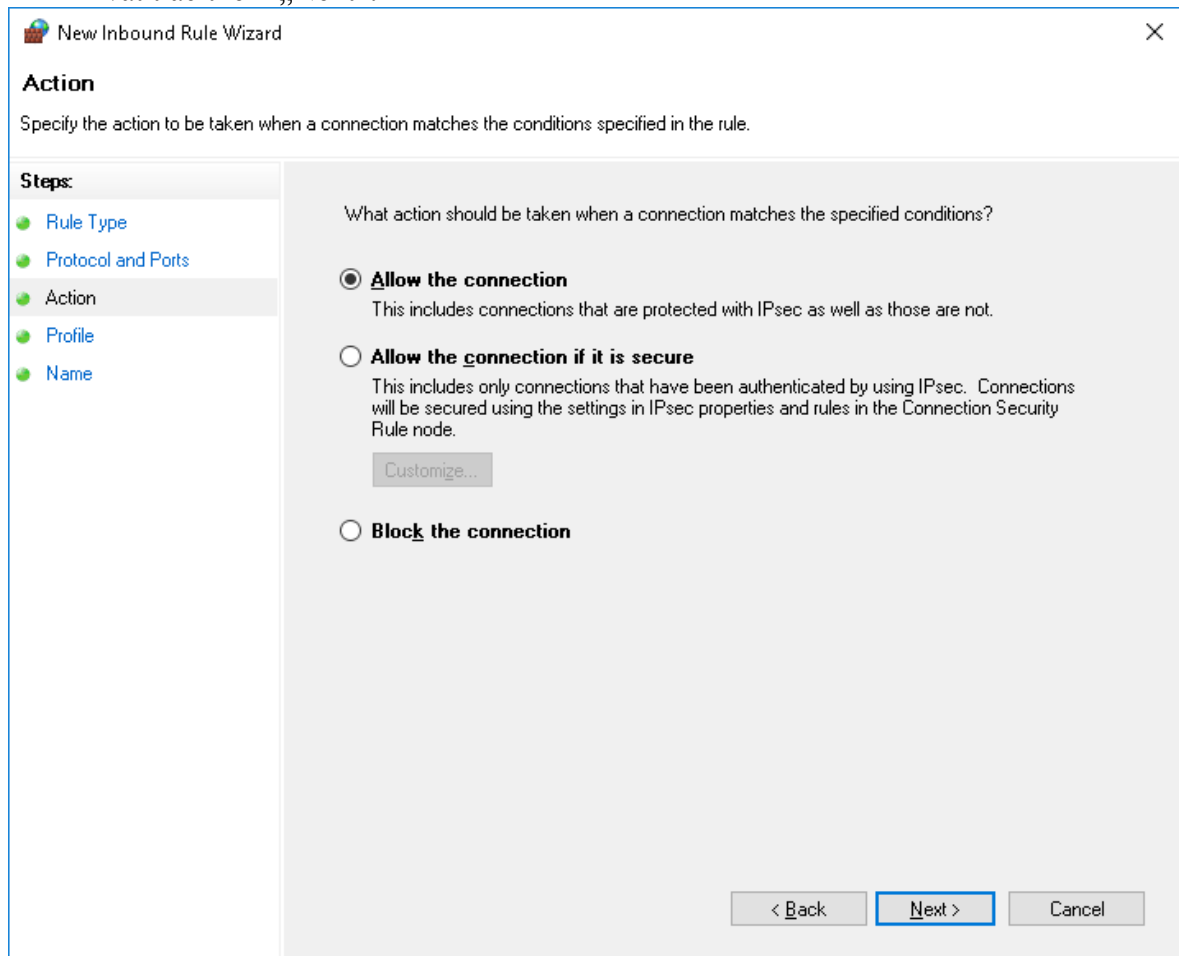
3. V nabídce Actions rozkliknout možnost „New Rule...“
4. V okně (New Inbound Rule Wizard), které se zobrazilo, je třeba provést výběr možnosti „Port“ a kliknout na tlačítko „Next“. Typ pravidla „Port“ je pro komunikaci s aplikací ASP.NET vyhovující, neboť komunikace probíhá na zvoleném portu, který byl nastaven ke komunikaci při spuštění aplikace v IIS.



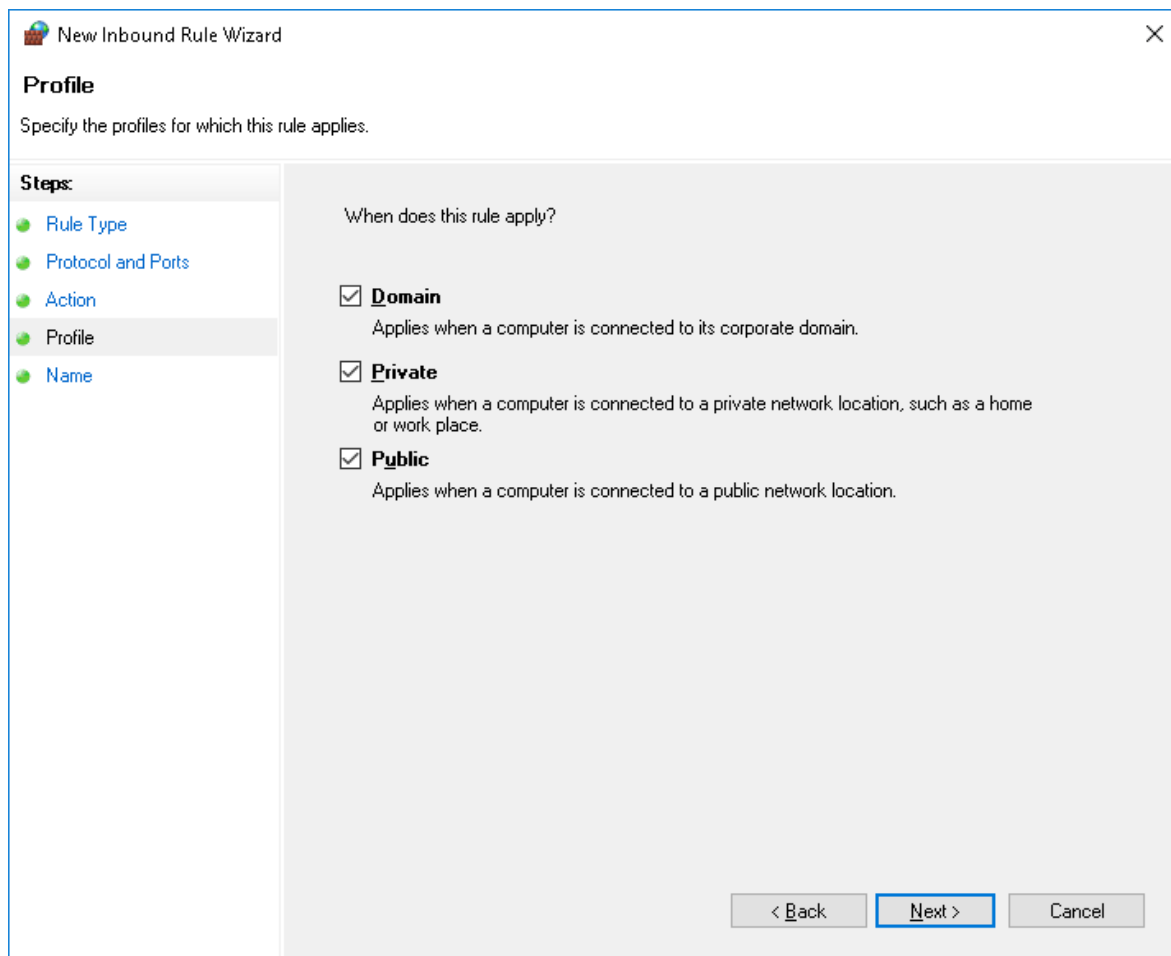
5. Na další obrazovce „Protocol and Ports“ je třeba zvolit možnost „TCP“ a vybrat „Specific local ports“, kde do editačního pole je třeba vepsat hodnotu pro port, na kterém běží požadovaná aplikace. Tlačítkem „Next“ pokračuje nastavení dále.



6. Na obrazovce „Action“ je třeba vybrat možnost „Allow the connection“ a pokračovat tlačítkem „Next“.



7. Na obrazovce „Profile“ je třeba vybrat všechny možnosti (Domain, Private i Public) a pokračovat tlačítkem „Next“.



8. Poslední krok slouží k pojmenování vytvářeného pravidla. Po kliknutí na tlačítko „Finish“ se pravidlo uloží.

New Inbound Rule Wizard ×

**Name**

Specify the name and description of this rule.

**Steps:**

- Rule Type
- Protocol and Ports
- Action
- Profile
- **Name**

Name:

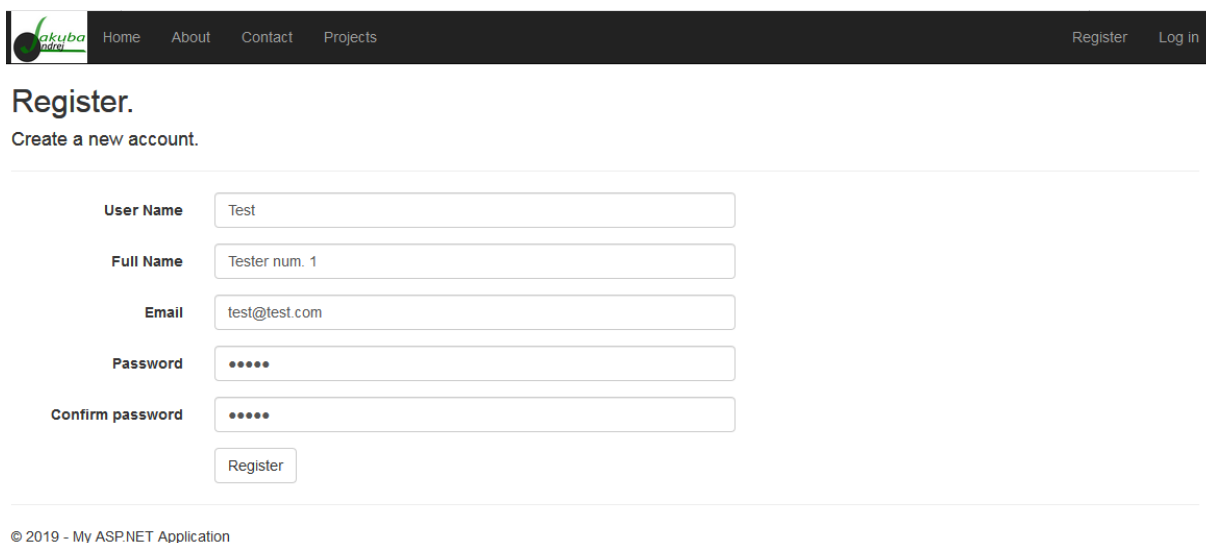
Description (optional):

## PŘÍLOHA P IV: TUTORIÁL

### Registrace

Email může být použit pouze jednou. Je potřeba pamatovat na to, že na tento email bude zaslán aktivační odkaz. Ujistěte se tedy, že zadaný email je opravdu správný.

Heslo musí obsahovat alespoň jedno číslo, malé písmeno a jedno velké písmeno. Délka hesla musí být alespoň 6 znaků.



The screenshot shows a web application interface for registration. At the top, there is a navigation bar with a logo on the left and links for 'Home', 'About', 'Contact', and 'Projects' in the center. On the right side of the navigation bar are links for 'Register' and 'Log in'. Below the navigation bar, the heading 'Register.' is displayed, followed by the text 'Create a new account.'. The registration form consists of several input fields: 'User Name' with the value 'Test', 'Full Name' with 'Tester num. 1', 'Email' with 'test@test.com', 'Password' with five dots, and 'Confirm password' with five dots. A 'Register' button is located below the 'Confirm password' field. At the bottom of the page, there is a copyright notice: '© 2019 - My ASPNET Application'.

### Zapomenuté heslo

Pokud uživatel zapomenul heslo, je možnost nechat si poslat na email odkaz pro změnu hesla. Pro zaslání emailu musí uživatel jít do sekce „Log in“ (<http://192.168.31.106:50505/DiplomkaTest/Account/Login>) a kliknout na odkaz „Forgot your password?“ (<http://192.168.31.106:50505/DiplomkaTest/Account/ForgotPassword>).

## Log in.

Use a local account to log in.

**User Name**

**Password**

Remember me?

[Register as a new user](#)

[Forgot your password?](#)

Use another service to log in

Sorry, there are no external authentication services configured at the moment.

© 2019 - My ASP.NET Application

Na následující stránce je potřeba vepsat vlastní email a kliknout na tlačítko „Email Link“.

Do emailu přijde zpráva, jako je na obrázku zde:

### Reset Password

qajakuba@gmail.com

Sent: so 24.8.2019 22:05

To: [REDACTED]

Please reset your password by clicking [here](#)

Otevřením odkazu se otevře následující stránka, kde je potřeba zadat email a nové heslo.

## Reset password.

Reset your password.

**Email**

**Password**

**Confirm password**

© 2019 - My ASP.NET Application

## Vytvoření projektu

V horní liště je možnost „Projects“ (Z hlavní stránky), na kterou je potřeba kliknout.

The screenshot shows the top navigation bar with the 'akuba' logo and links for Home, About, Contact, and Projects. The 'Projects' link is highlighted. On the right, there are links for 'Hello Oja!' and 'Log off'. Below the navigation bar is a large grey banner with the text 'Welcome!' and 'This is my thesis.' Underneath, there are three columns of content: 'Thesis assignment' with a description and a 'Get Start »' button; 'Automation Testing' with a description and an 'About »' button; and 'Contact me' with a description and a 'Contact »' button. At the bottom left, there is a copyright notice: '© 2019 - My ASP.NET Application'.

Otevře se stránka pro výběr projektu. Zde klikněte na odkaz „Create new“ (přímý odkaz: <http://192.168.31.106:50505/DiplomkaTest/Projects/CreateOwn>).

The screenshot shows the 'Select a project' page. The navigation bar is the same as in the previous screenshot. Below the navigation bar, there is a heading 'Select a project' and a link 'Create New' with a yellow arrow pointing to it. Below the link, there is a table with two columns: 'Project Name' and 'Owner'.

Napište název projektu a klikněte na tlačítko „Save“. Vlastníkem se stává automaticky uživatel, který tento projekt vytvoří.

The screenshot shows the 'Create Own Project' form. The navigation bar is the same as in the previous screenshot. Below the navigation bar, there is a heading 'Create Own Project' and a form with a 'Project Name' label and a text input field containing 'First project'. Below the input field is a 'Save' button.

## Výběr projektu

Výběr projektu lze provést na stránce <http://192.168.31.106:50505/DiplomkaTest/AccountProject>. Výběr projektu provedete

kliknutím na možnost „Select“, která se nachází ve stejném řádku, jako název projektu.

| Project Name  | Owner         |        |
|---------------|---------------|--------|
| UTB           | Ondřej Jakuba | Select |
| First project | Ondřej Jakuba | Select |

## Vytvoření stránky

Co je stránka? Jde o jednu z částí webu, která bude testována. Například na stránce <https://www.utb.cz/student/celozivotni-vzdelavani-a-u3v/sluzby-knihovny/> je sekce „Často hledáte“ (viz obrázek níže). Pro tuto sekci by bylo tedy vhodné vytvořit stránku s názvem „Často hledáte“. Používání stránek zlepšuje přehlednost, především pokud má testovaný projekt stránek velké množství.

The screenshot shows a web browser displaying the page <https://www.utb.cz/student/celozivotni-vzdelavani-a-u3v/sluzby-knihovny/>. The page features a navigation menu with items like 'E-příhláška', 'Portál IS/STAG', 'Office 365', 'Student', 'Zaměstnanec', 'Kontakty', and 'EN'. Below the navigation, there is a header for 'Univerzita Tomáše Bati ve Zlíně' and a list of categories: 'UNIVERZITA', 'UCHAZEČ', 'SPOLUPRÁCE', 'VĚDA A VÝZKUM', and 'ABSOLVENT'. The main content area has a large image of a modern building with a yellow arrow pointing to the title 'Často hledáte'. Below the image, there is a breadcrumb trail: 'Home > Student > Často hledáte'. A sidebar on the left contains a 'STUDENT' menu with sub-items: 'NASTAVENÍ WI-FI', 'E-MAIL UTB', 'MIKINY A TRIČKA UTB', and 'JIDELNÍČEK MENZY'. The main content area also has three columns of links: 'Nastavení Wi-Fi Mikiny a trička UTB', 'E-mail UTB Jídelníček menzy', and 'E-mail UTB Jídelníček menzy'.

Pro vytvoření stránky je třeba ve vybraném projektu přejít na sekci „Pages“ a zvolit možnost „Create new“ (přímý odkaz: <http://192.168.31.106:50505/DiplomkaTest/Pages/Create>).

Dashboard Projects Tests Steps **Pages** Elements Team Hello Ojal Log off

## Pages

Create New

| Page name               |   |
|-------------------------|---|
| Main                    | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Student                 | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| ESHOP UTB               | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Bestsellery - ESHOP UTB | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |

Na následující stránce napišete tedy vhodný název a potvrdíte kliknutím na tlačítko „Create“.

Dashboard Projects Tests Steps Pages Elements Team Hello Ojal Log off

## Create

### Page

Page Name

[Back to List](#)

## Vytvoření elementu

Na stránce, kterou je potřeba testovat se nachází několik elementů. Aby bylo možné s těmito elementy pracovat, je potřeba je vytvořit v projektu. Stránka „Často hledáte“ obsahuje například element „Nastavení Wi-Fi“.

E-příhláška Portál IS/STAG Office 365 Student Zaměstnanec Kontakty EN

Univerzita Tomáše Bati ve Zlíně FAKULTY A SOUČÁSTI UNIVERZITA UHAZEČ SPOLUPRÁCE VĚDA A VÝZKUM ABSOLVENT

# Často hledáte

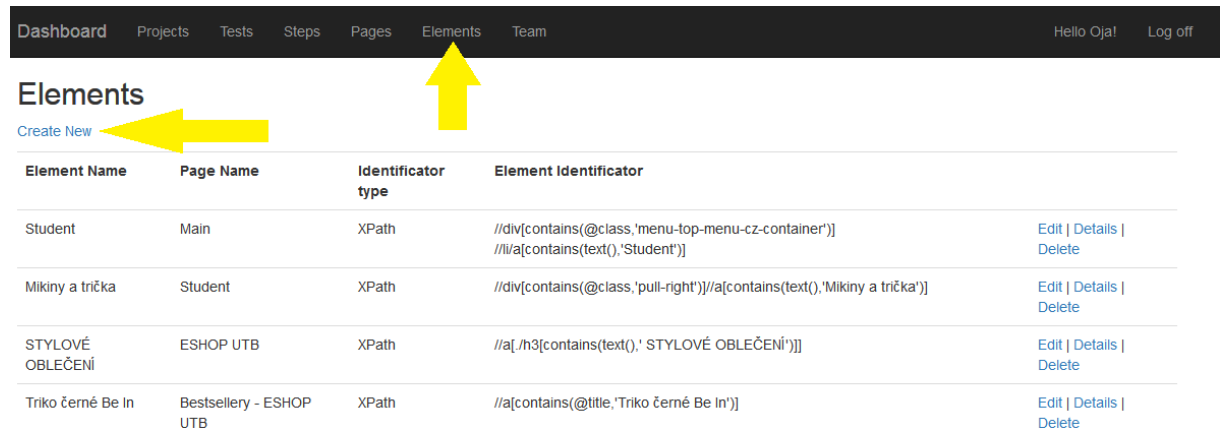
Home > Student > Často hledáte

- NASTAVENÍ WI-FI**
- E-MAIL UTB
- MIKINY A TRIČKA UTB
- JÍDELNÍČEK MENZY

**Nastavení Wi-Fi**  
Mikiny a trička UTB

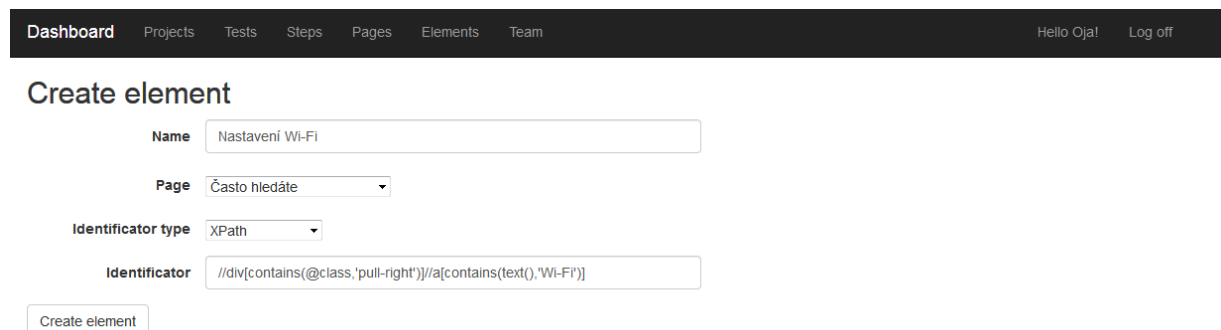
**E-mail UTB**  
Jídelníček menzy

Pro vytvoření elementu klikněte na sekci „Elements“ a poté na možnost „Create new“, nebo jděte rovnou na odkaz <http://192.168.31.106:50505/DiplomkaTest/Elements/Create>.



| Element Name      | Page Name               | Identifier type | Element Identifier   |   |
|-------------------|-------------------------|-----------------|--|---|
| Student           | Main                    | XPath           | //div[contains(@class,'menu-top-menu-cz-container')]<br>//li/a[contains(text(),'Student')] | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Mikiny a trička   | Student                 | XPath           | //div[contains(@class,'pull-right')]/a[contains(text(),'Mikiny a trička')]                 | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| STYLOVÉ OBLEČENÍ  | ESHOP UTB               | XPath           | //a[.h3[contains(text(),' STYLOVÉ OBLEČENÍ')]]   | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Triko černé Be In | Bestsellery - ESHOP UTB | XPath           | //a[contains(@title,'Triko černé Be In')]  | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |

Vepište vhodný název elementu (může být použit text, který element obsahuje). Zvolte stránku, na které se element nachází. Vytvořený identifikátor, kterým lze najít element, vložte jako „Identifier“. Podle druhu identifikátoru zvolte typ identifikátoru. Kliknutím na „Create element“ bude element vytvořen.



**Create element**

Name:

Page:

Identifier type:

Identifier:

## Vytvoření kroku

Krok na těchto stránkách je sérií akcí prováděných na jedné stránce, které tvoří komplexní celek požadovaného postupu. Například by to mohl být krok s názvem „Přihlásit se“, kde jako série kroků (vepsání jména, hesla a kliknutí na tlačítko přihlásit) bude odpovídat postupu, jakým je možné se dopracovat požadovanému výsledku.

Pro vytvoření kroku je potřeba přejít do sekce „Steps“ a vybrat možnost „Create new“, nebo přejít rovnou na <http://192.168.31.106:50505/DiplomkaTest/Steps/Create>.

| Page Name               | Step Name                           | Description   |
|-------------------------|-------------------------------------|---|
| Main                    | Výběr sekce "Student"               | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Student                 | Přejít na "Mikiny a trička"         | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| ESHOP UTB               | Přejít na "Stylové mikiny a trička" | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Bestsellery - ESHOP UTB | Zobraz produkt "Tričko černé be in" | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |

Vepište vhodný název, který dostatečně popisuje požadované sérii kroků. Vyberte stránku, se kterou se bude pracovat v tomto kroku. Popis je nepovinný, ale může pomoci seznámit i ostatní účastníky projektu o postupu, který tento krok bude vykonávat. Klikněte na tlačítko „Create step“.

| Name                   | Page          | Description   |
|------------------------|---------------|---|
| Jdi na Nastavení Wi-Fi | Často hledáte | knutím na možnost "Nastavení Wi-Fi" přejde na stránku "Wi-Fi síť eduroam" |

Create step

Po vytvoření kroku stránka přesměruje uživatele na stránku pro přidání akcí do tohoto kroku.

### Přidání akcí do kroku

Vyberte akci z nabídky. Pro některé akce je potřeba vložit text (například akce psaní). Vyberte element, se kterým tato akce bude pracovat (Na výběr jsou pouze elementy, které leží na stránce, pro kterou je krok tvořen). Posledním parametrem je časové rozpětí. Tato hodnota udává, v jakém časovém rozmezí musí být akce vykonána. Pokud například stránka nestihne element zobrazit během této doby, akce nebude úspěšná. Tato hodnota se zadává

v sekundách a musí být rovno 1 a víc.

Dashboard Projects Tests Steps Pages Elements Team Hello Oja! Log off

## Jdi na Nastavení Wi-Fi

**Page name** Často hledáte  
**Description** Kliknutím na možnost "Nastavení Wi-Fi" přejde na stránku "Wi-Fi síť eduroam"

[Edit](#)

**Actions**

| Action | Time span [s] |
|--------|---------------|
|--------|---------------|

Add step action

Select an action

Select an element

Time span in seconds

[Create](#)

Kliknutím na tlačítko „Create“ bude akce přidána. Dále je možno stejným způsobem pokračovat až do naplnění požadavků.

Dashboard Projects Tests Steps Pages Elements Team Hello Oja! Log off

## Jdi na Nastavení Wi-Fi

**Page name** Často hledáte  
**Description** Kliknutím na možnost "Nastavení Wi-Fi" přejde na stránku "Wi-Fi síť eduroam"

[Edit](#)

**Actions**

| Action                          | Time span [s] |                        |
|---------------------------------|---------------|------------------------|
| Click on <b>Nastavení Wi-Fi</b> | 5             | <a href="#">Delete</a> |

Add step action

Click

Nastavení Wi-Fi

Time span in seconds

[Create](#)

## Vytvoření testu

V sekci „Tests“ klikněte na možnost „Create new“, nebo přejděte přímo na <http://192.168.31.106:50505/DiplomkaTest/Tests/Create>.

Dashboard Projects **Tests** Steps Pages Elements Team Hello Oja! Log off

## Tests

[Create New](#)

| Project Name | Test Name                               |  |
|--------------|---|--|
| UTB          | Proklikání ke zboží "Triko černé Be In" | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>   <a href="#">Run test</a> |

Jako název testu používejte pár slov, které dostatečně přesně popisují cíl testu (Kdokoliv by měl z názvu vyčíst, co je testováno). Jako „Url“ použijte odkaz stránky, ze které má test začínat. Popis by měl obsahovat detailní popis testu. Není to povinné pole, ale může pomoci seznámit kohokoliv s detailním postupem. Test bude vytvořen kliknutím na „Create“.

The screenshot shows a web application interface with a dark navigation bar at the top containing 'Dashboard', 'Projects', 'Tests', 'Steps', 'Pages', 'Elements', and 'Team'. On the right side of the bar, it says 'Hello Oja!' and 'Log off'. Below the navigation bar, the page title is 'Create Test'. The form contains three input fields: 'Test Name' with the value 'Student wants to go to Wi-Fi síť eduroam', 'Url' with the value 'http://www.utb.cz/', and 'Description' with the text 'As a student I want to go to "Student section". In the Student section should be link to the WiFi setting. The link should open the page.'. Below the description field is a 'Create' button. At the bottom left of the form area is a link 'Back to Test List'.

Po vytvoření testu stránka uživatele přeměruje na stránku pro přidávání kroků do tohoto testu.

The screenshot shows the 'Add a step' page in the same web application. The navigation bar is identical. The page title is 'Add a step'. Below it, the test details are displayed: 'Test Name: Student wants to go to Wi-Fi síť eduroam', 'Description: As a student I want to go to "Student section". In the Student section should be link to the WiFi setting. The link should open the page.', and 'Url: http://www.utb.cz/'. There is an 'Edit' link below the test details. Under the heading 'Steps:', it says 'There is not any step.'. Below this, there are two dropdown menus: 'Page' and 'Step', both currently showing 'Select page'. A 'Save' button is located below the dropdowns. At the bottom left is a link 'Back to List'.

Vyberte stránku, na které se test bude nacházet a vyberte krok, který na ní chcete provádět. Poté klikněte na tlačítko „Save“.

## AddStep

### Test

**Test Name** Student wants to go to Wi-Fi síť eduroam

**Description** As a student I want to go to "Student section". In the Student section should be link to the WiFi setting. The link should open the page.

**Url** http://www.utb.cz/

[Edit](#)

### Steps:

| Step name             | Actions                |
|-----------------------|------------------------|
| Výběr sekce "Student" | <a href="#">Remove</a> |

**Page**

**Step**

[Back to List](#)

## Spuštění testu

Přejděte na sekci „Tests“ nebo odkazem <http://192.168.31.106:50505/DiplomkaTest/Tests> a klikněte na možnost „Run test“ v řádku, na kterém se nachází test, který má být spuštěn.

## Tests

[Create New](#)

| Project Name | Test Name                                |  |
|--------------|--|--|
| UTB          | Proklikání ke zboží "Triko černé Be In"  | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>   <a href="#">Run test</a> |
| UTB          | Student wants to go to Wi-Fi síť eduroam | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>   <a href="#">Run test</a> |

Na následující stránce musí být vybrán prohlížeč, na kterém test bude vykonán. Kliknutím na tlačítko „Start test“ se test spustí (testy běží na serveru s klientem).

## Detail

Select browser

## Výsledky testů

Výsledky se nachází v sekci „Dashboard“ (<http://192.168.31.106:50505/DiplomkaTest/AccountProject/ProjectDashboard>).

## Dashboard

### Last Results

| Browser Name | Test Name                               | Completed | Passed | StartTime           |  |
|--------------|---|-----------|--------|---------------------|--|
| Chrome       | Proklikání ke zboží "Triko černé Be In" |           |        | 26/08/2019 19:27:54 | <a href="#">Details</a>   <a href="#">Delete</a> |

Odsud si lze zobrazit výsledek testu detailně kliknutím na možnost „Details“, která se nachází na řádku s daným výsledkem. Příklad, jak detail testu může vypadat:

## Test Report

[Back to List](#)

### Proklikání ke zboží "Triko černé Be In"

|              |                     |
|--------------|---------------------|
| Browser Name | IE                  |
| Completed    |                     |
| Passed       |                     |
| StartTime    | 26/08/2019 12:33:58 |

- ✔ **Výběr sekce "Student"**
  - ✔ Assert that element is displayed | Element: Student
  - ✔ Click on Student
- ✔ **Přejít na "Mikiny a trička"**
  - ✔ Assert that element is displayed | Element: Mikiny a trička
  - ✔ Move mouse on Mikiny a trička
  - ✔ Click on Mikiny a trička
- ✔ **Přejít na "Stylové mikiny a trička"**
  - ✔ Move mouse on STYLOVÉ OBLEČENÍ
  - ✔ Click on STYLOVÉ OBLEČENÍ
- ✔ **Zobraz produkt "Tričko černé be in"**
  - ✔ Click on Triko černé Be In