VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2020 Martin Gajdoš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Gajdoš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: slovenština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ITeuro, a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura, Ph.D.**

Konzultant bakalářské práce: Ing. David Prokop

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1016 OSTRAVA

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 30. apríla 2020 .....

Súhlasím so zverejnením tejto bakalárskej práce podľa požiadaviek čl . 26, odst. 9 Študijného a skúšobného poriadku pre štúdium v bakalárskych programoch VŠB-TU Ostrava.

V Ostrave 30. apríla 2020 .....

ITeuro, a.s., Hollarova 1124/14
702 00 Ostrava-Moravská Ostrava
DIČ: CZ25850245, tel.: 591 162 411

Touto cestou sa chcem poďakovať mojej rodine a blízkym za podporu a spoločnosti ITeuro, a.s. za vytvorenie výborných podmienok pre absolvovanie odbornej praxe. Špeciálne poďakovanie patrí konzultantovi práce, Ing. Davidovi Prokopovi, za konzultácie ohľadom realizácie. Rovnako sa chcem poďakovať aj môjmu vedúcemu práce, Ing. Janovi Gaurovi, Ph.D., za usmerňovanie a rady pri písaní bakalárskej práce.

Abstrakt

V tejto bakalárskej práci opisujem priebeh mojej bakalárskej praxe vo firme ITeuro, a.s. V rámci úvodnej kapitoly predstavujem tému, ktorej som sa na praxi venoval. Nasledujúca časť práce opisuje odborné zameranie firmy a technológie, s ktorými som pracoval. Hlavnou a zároveň nosnou časťou bakalárskej práce sú zadané úlohy, postupy pri ich riešení a vyjadrenie časovej náročnosti jednotlivých úloh. Záver obsahuje zhodnotenie dosiahnutých výsledkov, zhodnotenie znalostí nadobudnutých počas štúdia a zhodnotenie znalostí získaných v priebehu bakalárskej praxe.

Kľúčové slová: C#, automatizované testovanie, MSSQL, Selenium, Winium

Abstract

In this bachelor thesis I describe the course of my bachelor work in the company ITeuro, a.s. In the introductory chapter, I present a topic that I have been dealing with in practice. The following part describes the professional focus of the company and technology, which I have worked with. The main and the biggest part of the bachelor thesis are assigned tasks, procedures for solving them and expressing the time demands of individual tasks. The conclusion includes evaluation of achieved results, evaluation of knowledge acquired during study and evaluation of knowledge gained during the bachelor's practice.

Keywords: C#, automation testing, MSSQL, Selenium, Winium

Obsah

Zoznam použitých skratiek a symbolov

ERP	–	Enterprise Resource Planning
CRM	–	Customer Relationship Management
SQL	–	Structured Query Language
WPF	–	Windows Presentation Foundation
SSMS	–	SQL Server Management Studio
T-SQL	–	Transact-SQL
IDE	–	Integrated Development Environment
PC	–	Personal Computer
APS	–	Advanced Planning and Scheduling
IDO	–	Intelligent Data Objects

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov zdrojového kódu

1 Úvod

Cieľom tejto bakalárskej práce je opis priebehu mojej bakalárskej praxe vo firme ITeuro, a.s. Vypracovanie bakalárskeho projektu formou praxe som si zvolil z niekoľkých dôvodov. V prvom rade som chcel spoznať a vyskúšať si prácu v reálnej IT firme a zároveň byť súčasťou vývojového tímu. V druhom rade som chcel nadobudnúť nové poznatky, skúsenosti a rozšíriť si tak vedomosti.

Do firmy som nastúpil už na konci druhého ročníka na pozíciu programátora. Spočiatku bolo moje pracovné nasadenie o zoznamovaní sa so systémami Infor Cloud Industrial a InduStream, na ktorých firma pracuje. Jednotlivé systémy sú popísané nižšie v tomto dokumente. Podieľal som sa na vývoji modifikácii systémov podľa požiadaviek zákazníka, čo mi prehĺbilo znalosti systémov a rozšírilo programátorske schopnosti. Následne som dostal úlohu vytvoriť sadu automatických testov týchto aplikácii, čo je témou mojej bakalárskej praxe.

V úvodnej kapitole predstavujem odborné zameranie firmy a technológie, s ktorými som pracoval. Hlavnou a zároveň nosnou časťou bakalárskej práce sú zadané úlohy, postupy pri ich riešení a vyjadrenie časovej náročnosti jednotlivých úloh. Záver obsahuje zhodnotenie dosiahnutých výsledkov, zhodnotenie znalostí nadobúdnutých počas štúdia a zhodnotenie znalostí získaných v priebehu bakalárskej praxe.

2 Spoločnosť ITeuro, a. s.

ITeuro, a. s. je česká konzultačná a softwarová firma, zaoberajúca sa vývojom informačného riešenia pre výrobné firmy. Pojem informačné riešenie zahŕňa vývin podnikových aplikácií, poskytovanie služieb v oblasti návrhu efektívneho riešenia pre zvýšenie efektivity a produktivity podľa celosvetových štandardov, expertné poradenstvo pre optimalizáciu podnikových procesov, školenie uživateľov a v neposlednom rade poskytovanie podpory formou supportu pre rýchle riešenie problémov, s ktorými sa výrobné firmy pri práci s našim softwarom stretnú.

2.1 História

Korene firmy siahajú už do roku 1997, kedy sa firma Autel rozhodla rozšíriť dodávku a implementáciu ERP systému SyteLine. Spočiatku systém vlastnila americká firma Symix. Autel založilo dcérsku firmu Autel-IT. Postupom času si SyteLine vystriedalo hneď niekoľko vlastníckych spoločností. Dnes tento systém vlastní spoločnosť Infor, ktorého partnerom pre Českú republiku a Slovensko sa stalo od roku 2000 ITeuro ako dcérska firma, spoločnosti Autel-IT.

Spoločnosť následne vyvinula niekoľko vlastných doplnkov a českú lokalizáciu pre SyteLine, ktorý dnes poznáme pod názvom Infor CloudSuite Industrial. Postupným rastom sa firme podarilo taktiež vyvinúť vlastný ucelený software s názvom InduStream [1].

2.2 Infor CloudSuite Industrial (SyteLine)

Infor CloudSuite Industrial je ERP systém, ktorý slúži ako nástroj pre udržanie konkurenciaschopnosti v zložitom globálnom prostredí. Poskytuje komplexné riešnie spojené s firemnými procesmi, ako aj sady sofistikovaných funkcii. Konkrétne ide o funkcie v oblasti obchodu, nákupu, plánovania, výroby, účtovaníctva, workflow, tvorbu reportov, riadenia vzťahov so zákazníkmi (CRM) a mnoho ďalších funkcii, ktoré sú nevyhnutné pre rýchle a efektívne fungovanie výrobných podnikov [2].

Medzi veľkú výhodu SyteLinu patrí aj integrovaná funkcia APS (Advanced Planning and Scheduling), ktorá dokáže efektívne plánovať a rozvrhovať výrobné kapacity tovární. Pri prijatí zákazky dokáže overiť či je dodanie v požadovaný termín možné, zdôvodniť príčiny prípadneho časového sklzu ako aj stanoviť reálny termín dodania [3].

2.3 InduStream

InduStream je aplikácia, ktorá slúži pre pokrytie potrieb výrobnej firmy priamo v továrni. Pracovníci tak môžu dostávať konkrétne úlohy priamo na pracovisku a taktiež poskytujú spätnú väzbu o prevedenej práci priamo na pracovisku prostrerdníctvom pevných, či mobilných terminálov. Kombinunje funkcie jendoučelových nástrojov a zároveň využívania jedného softwaru, ktorý je prepojený s ľubovoľným ERP systémom. Systém je zložený z viacerých modulov ako

napríklad „Terminálový sběr dat“, ktorý slúži na zobrazanie a zadávanie informácii. Ďalším dôležitým modulom je napríklad „Bezpapírová dílna“, ktorá slúži na zobrazovanie a spracovávanie výrobnej a technologickej dokumentácie [4].

Obr. 1: Logo firmy ITeuro, a. s.

2.4 Moje pracovné zaradenie

Do firmy som nastúpil na pozíciu programátora s cieľom naučiť sa nové veci, spoznať ako chutí práca vo vývojovom tíme, ako funguje riadenie projektu a rozdeľovanie práce. Do práce som chodieval dva až trikrát do týždňa v závislosti na situáciu a časovú náročnosť školy. Firma od začiatku naplnila moje očakávania.

Prvé dni som sa v práci zaúčal. Spoznával som technológie s ktorými firma pracuje. Spätnou väzbou bolo týždenné review poznatkov, ktoré som sa naučil, s firemným konzultantom mojej bakalárskej práce a zároveň vedúcim oddelenia vývoja, Ing. Davidom Prokopom.

V druhom kroku procesu zaúčania bolo vyvíjanie modifikácii v rámci ERP systému Infor CloudSuite Industrial podľa požiadaviek zákaznika. Pri vývoji modifikácii som používal mnoho technológií. V prvom rade bola potrebná znalosť jazyka SQL, nakoľko systém využíva ako nosnú dátovú vrstvu databázu na MSSQL serveri. Okrem SQL tiež využíva ActiveX vo forme VB.NET alebo C#. Samostatnú kategóriu predstavuje editačný mód, ktorý umožňuje vyvíjať jednoduchšie modifikácie aj bez nutnej znalosti programátorskeho jazyka. Postupom času som sa začal venovať aj vývoju modifikácii pre InduStream. Systém je napojený na ERP systém SyteLine. Pre vývin modifikácii sa využíva interný nástroj InduStream Designer, ktorý pripomína editačný mód v SyteLine. Aplikácia je zložená z modulov, ktoré sú implementované na platforme .NET s využitím WPF. Znalosť získaná z vyvíjania a učenia sa o jednotlivých produktoch firmy mi pomohla v nasledujúcich iteráciach praxe.

Okrem iného som pracoval aj s technológiami ako je Git, interný nástroj Scripting a mnoho ďalších, ktoré si vysvetlíme v ďalšej kapitolej tohoto dokumentu.

3 Využité technológie v priebehu praxe

Práca na komplexnom informačnom systéme ale aj rôznorodosť projektov si vyžaduje znalosť viacerých technológií. V mojom prípade išlo hlavne o jazyky SQL, VB.NET alebo aj samotné editačné prostredie systému Infor CloudeSuite Industrial či Designer aplikácie InduStream. Pre vývoj automatizovaných testov som použil jazyk C# a testovací framework Winium.

Pri práci v tíme sme pre zlučovanie kódu softwaru využívali Git, ale aj rôzne interné nástroje, ktoré zefektívňujú tímový vývoj. V tejto kapitole čitateľovi priblížim jednotlivé nástroje a technológie.

3.1 Vývojové prostredia

V priebehu praxe som využíval hneď niekoľko vývojových prostredí. Pri vývoji modifikácii na jednotlivých projektoch som prevažne pracoval vo vývojovom prostredí SSMS (SQL Server Management Studio) pre prácu s jazykom SQL a jeho programátorskou nadstavbou T-SQL. Druhým vývojovým prostredím, ktoré som používal bolo Microsoft Visual Studio pre programovanie na platforme .NET v jazyku Visual Basic alebo pri vývoji automatizovaných testov v jazyku C#.

3.1.1 SQL Server Management Studio (SSMS)

SSMS je vývojové prostredie vhodné pre správu akejkoľvek SQL infraštruktúry. Správou rozumieme prístup na server, konfiguráciu, vývoj komponentov SQL Serveru a mnoho ďalších. SSMS predstavuje komplexný nástroj, ktorý kombinuje širokú skupinu grafických nástrojov so skriptovacími nástrojmi, ktoré umožňujú napríklad:

- Lepšiu a prehľadnejšiu optimalizáciu dotazov (Display Estimated Execution plan).
- Pohodlnejšie písanie kódu (IntelliSense).
- Nástroj na ladenie (SQL Server Profiler) [5].

Toto prostredie som využíval s použitím jazyka SQL a jeho programátorskou nadstavbou T-SQL. Prevažne som jazyk SQL používal na dotazovanie sa na databázu, pre dohľadávanie potrebných dát alebo pre programovanie funkcionality prostredníctvom uložených procedúr, funkcí a pod. Pre ladenie som používal tool SQL Server Profiler, ktorý zaznamenáva príkazy ktoré sa v postupnosti poslali na databázu pri jednotlivých procesoch.

3.1.2 Microsoft Visual Studio

Microsoft Visual Studio je komplexné vývojové prostredie, ktoré je vhodné pre prácu v rôznych oblastiach vývoja softwaru. Ide o Integrated Development Environment (IDE), ktoré obsahuje veľké množstvo vylepšení urýchľujúce prácu programátora. Okrem štandardného editora a debuggeru, ktoré v dnešnej dobe ponúka mnoho IDE obsahuje kompilery pre preklady zdrojových kódov, nástroje pre nápovedu pri písaní programu (IntelliSense) či grafické dizajnery [6].

3.1.3 Interné nástroje

Pre zrýchlenie a zvýšenie efektivity vývoja softwaru sme využívali niekoľko interných nástrojov. Konkŕetne išlo o nástroje ako skriptovací nástroj alebo aplikačný nástroj.

Skriptovací nástroj slúži pre vytvorenie SQL skriptov, ktoré implementujú uživateľsky definované verzie formulárov do aktuálnej verzie, a tak ich sprístupňujú pre ostatných uživateľov. Tento proces bol dôležitý pri vývoji modifikácii. Nástroj funguje tak, že porovnáva uživateľské zmeny formulára s jeho pôvodnou verziou. Následne zobrazí tieto zmeny programátorovi, ktorý si môže skontrolovať, či sú všetky zmeny v poriadku. V prípade sporu môže zmenu daného komponentu zrušiť. Po kontrole zmien je možné vygenerovať SQL script.

Aplikačný nástroj je využívaný pre aplikáciu veľkého množstva scriptov do databáz. Najčastejšie sa využíva pre aplikáciu softwaru na stranu zákaznika, či aplikaciu do rôznych interných databáz.

3.2 Structured Query Language (SQL)

Jazyk, ktorý slúži na organizáciu dát relačných databáz. Užívatelia sú tak schopní dotazovať sa na dáta z databázových tabuliek, ktoré môžu byť prepojené prostredníctvom vzťahov.

3.3 .NET

Bezplatná vývojová platforma s otvoreným zdrojovým kódom (open source), ktorá sa využíva na vývoj webových, mobilných či desktopových aplikácii. Aplikácie .NET platformy je možné písať v programovacích jazykoch C#, F# alebo Visual Basic.

C# je jednoduchý, moderný, objektovo-orientovaný a typovo-bezpečný programovací jazyk.

F# je funkcionálny programovací jazyk využitelný naprieč všetky platformy. Charakterizuje ho aj otvorený zdrojový kód, ktorý tiež obsahuje prvky pre imperatívne a objektovo-orientované programovanie.

Visual Basic je jazyk s jednouchou syntaxou pre vytváranie typovo-bezpečných, objektovo-orientovaných aplikácii [7].

3.4 Git

Git je nástroj, ktorý je priam nevyhnutný pri tímovom vývoji projektov. Má otvorený zdrojový kód, je zadarmo a je ho možné používať prostredníctvom rôznych klientov. V mojom prípade to bol klient SmartGit.

Hlavnou úlohou gitu je verzovanie projektu. Verzovanie je realizované prostredníctvom vetví v stromovej štruktúre. Tieto vetvy môžu byť lokálne - uložené na PC programátora, alebo globálne - uložené na serveri (origin). Aktuálnu verziu projektu predstavuje hlavná vetva. Programátor si vytvorí lokálnu vetu z hlavnej vetvy, čím si zabezpečí, že má aktuálnu verziu projektu, a tým

pádom môže realizovať svoju časť projektu bez toho, aby zasahoval do hlavnej vetvy. Vďaka tomu je hlavná vetva izolovaná a znižuje sa riziko jej poškodenia.

Počas práce na svojej projektovej časti v tíme sa však stáva, že do hlavnej vetvy iný programátor vloží svoju časť projektu, ktorú dokončil. V momente keď sa to stane, programátor, ktorý si pred tým vytvoril vetvu z vtedajšej aktuálnej verzie, stráca aktuálnu verziu. Tento problém riešia operácie, ktoré git ponúka. Medzi základné operácie gitu patria `pull`, `push`, `commit` a `merge`.

Operácia `commit` predstavuje akési potvrednie a uloženie doterajších zmien v projekte na lokálnej vetve. Pred operáciou `commit` tak programátor vidí, čo všetko zmenil a pred samotným uložením/potvrdením jeho zmien si môže všetky zmeny skontrolovať, či sú v poriadku.

Operácia `pull` umožňuje stiahnutie verzí do lokálnej vetvy, ktoré boli pridané do originu v čase po vytvorení lokálnej vetvy. Vďaka tejto operácii môžu mať všetci vývojári aktuálnu verziu projetku.

Operácia `push` nahráva `commitnuté` verzie projektu, na ktorom vývojár pracuje z lokálnej vetvy do originu. Zabezpečuje sprístupnenie lokálnych verzií projektu celému tímu.

Najrizikovejšou a paradoxne najpotrebnejšou operáciou v gite je `merge`. Táto operácia zabezpečuje zlučovanie jednotlivých verzií projektu do jednej vetvy a tak vytvára jeden ucelený software. Pri tejto operácií môže pri nesprávnom postupe dôjsť ku konfliktu, ktorý je spôsobený najčastejšie stretom 2 vetví na jednom a tom istom riadku. Tento problém musí programátor vyriešiť dôsledným porovnávaním a ručným zlučovaním verzií daného kódu, aby nedošlo ku zbytočnému narušeniu kódu.

3.5 Winium

Winium je framework, ktorý je určený pre automatizované testovanie. Je postavený na testovacom frameworku Selenium. Kým Selenium slúži na testovanie webových aplikácii, jeho nadstavba Winium slúži na testovanie desktopových aplikácii. Tento framework nie je zavislý na jednom programátorskom jazyku. Je kompatibilný s jazykmi ako Java, Objective-C, JavaScript s Node.js, Python, PHP, Ruby a pod. Ja som tento framework používal v jazyku C# [8].

4 Zadané úlohy v priebehu praxe a ich časová náročnosť

V priebehu bakalárskej praxe som dostal niekoľko úloh. Na začiatku išlo o zoznámenie sa s aplikáciami Infor CloudSuite Industrial a InduStream. Následne som sa musel naučiť ako funguje testovanie softwaru a ako pracovať s Winium frameworkom. Potom čo som sa naučil ako Winium funguje som prešiel na vytváranie testovacích scenárov pre aplikácie Industream a Infor CloudeSuite Industrial. Najdlhšiu časť predstavoval návrh implementácie a samotná implementácia jednotlivých testovacích scenárov. Poslednou úlohou bola implementácia reportu o výsledku priebehu testov. V tejto kapitole čitateľovi priblížim, ako jednotlivé úlohy prebiehali a zároveň vyjadrím časovú náročnosť jednotlivých úloh.

4.1 Zoznámenie sa s aplikáciou Infor CloudSuite Industrial a InduStream

Na to aby som bol schopný navrhnúť testovacie scenáre rôznych procesov v aplikáciach Infor CloudeSuite Industrial a InduStream, bolo potrebné sa s jednotlivými aplikáciami zoznámiť. Zoznámenie so samotnými aplikáciami prebiehalo spočiatku formou samoštúdia. Neskôr som začal pracovať na vývoji modifikácií podľa požiadaviek zákaznika, vďaka čomu som sa s jednotlivými procesmi oboznámil v bežnej praxi.

4.1.1 Zoznámenie sa s aplikáciou Infor CloudSuite Industrial

Na začiatku som sa zoznámil s architektúrou softwaru. Jeho architektúra je rozložená na 3 vrstvy:

1. **Databázová vrstva** - základný prvok systému, zabezpečujúci korektné uloženie dát. Súčasťou tejto vrstvy sú dve hlavné databázy - aplikačná a formulárová. Aplikačná databáza obsahuje dáta nutné k fungovaniu systému - uložené procedúry. Formulárová databáza uchvováva informácie o uživateľskom rozhraní.

2. **Stredná vrstva** - obsahuje prvky, ktoré zabezpečujú komunikáciu medzi klientskou vrstvou a databázovou vrstvou. Tieto prvky sa označujú ako Intelligent Data Objects (IDO). Okrem toho sa na klientskej vrstve nachádzajú aj komponenty, ktoré slúžia pre interpretáciu údajov o uživateľskom rozhraní - Form server.

3. **Klient** - zabezpečuje komunikáciu užívateľa so strednou vrstvou. Jedná sa o jednoduchý nástroj, vďaka ktorému si môže užívateľ zobrazovať jednotlivé formuláre a manipulovať s príslušnými dátami.

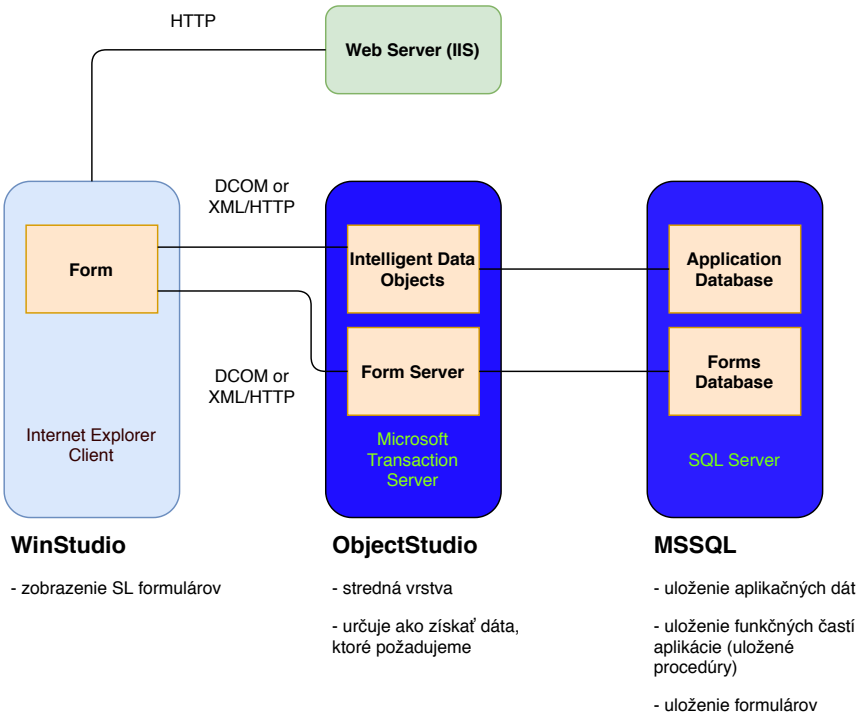
Následne som si vyskúšal vývoj modifikácie na základe série YouTube videí od spoločnosti Infor zvanou Hello world [9].

SyteLine je zložený z formulárov vďaka ktorým je užívateľ schopný realizovať jednotlivé procesy v oblastiach výroby, účtovníctva, logistiky a pod. Toto je možné v tzv. runtime móde.

Ak však chceme tieto formuláre modifikovať, musíme prejsť do takzvaného editačného módu. Následne nám aplikácia ponúkne výber Scopu, teda verzie formulára, v ktorej sa jednotlivé zmeny v editačnom móde prejavia. Tieto formuláre môžu byť v štyroch verziách:

1. Vendor
2. Site - default
3. Group
4. User

Verzia Vendor predstavuje štandardnú verziu formulárov, ktorú dodáva spoločnosť Infor. Verzia Site - default je kópia verzie Vendor a zároveň obsahuje modifikácie formulárov, ktoré sú dostupné všetkým užívateľom danej konfigurácie. Verzia Group obsahuje modifikácie formulárov spoločné pre užívateľov, ktorí sú súčasťou danej skupiny. Poslednou verziou je User verzia jedného užívateľa Sytelinu. V rámci praxe som na modifikáciach pracoval prostredníctvom svojho uživateľského konta, a tak všetky úpravy, ktoré som previedol spôsobili, že sa formuláre z verzie dostupnej všetkým užívateľom ‚Site - default, previedli do verzie User a boli dostupné iba pri prihlásení sa prostredníctvom môjho užívateĺského konta. Pre ostatných užívateĺov je teda viditelná verzia formulára Site - default.



Obr. 2: Architektura aplikácie Infor CloudSuite Industrial (SyteLine)

Samotný vývoj modifikácii prebiehal v niekoľkých fázach. Porces začínal štúdiom analýzy požiadaviek zákaznika. Analýzu spisuje konzultant, ktorý je so zákaznikom v priamom kontakte. Po preštudovaní analýzy som absolvoval stretnutie s konzultantom, ktorý mi celý proces predviedol na užívateľskej úrovni. To docielilo, že som sa postupne začal s jednotlivými procesmi zoznamovať.

Začiatok vývoja prebiehal tak, že som previedol operáciu `pull` na troch hlavných vetvách dev, test a live v aplikácii SmartGit, čím som si zabezpečil, aby som mal vo svojom lokálnom repozitári aktuálnu verziu týchto troch hlavných vetví. Každý zákazník ma svoj vlastný git repozitár s troma hlavnými vetvami, aby sme izolovali skripty jednotlivých zákazníkov a zároveň oddelili verzie systému, pretože každý zákazník požaduje niečo iné.

Vetva dev je určená pre vývoj a test modifikácie v rámci našej firmy. Vetva test je vetva, ktorá je určená na testovanie požadovaných modifikácii u našich zákazníkov. V tomto momente sa modifikácia ešte v riadnej výrobe nepoužíva. Vetva live následne obsahuje skripty modifikácii, ktoré boli schválené zákaznikom v procese testovania a tak aplikované pre riadne používanie vo výrobnej firme daného zákazníka.

Vývoj pokračuje vytvorením novej vetvy z hlavnej vetvy live, ktorá je najbližšia k aktuálne používanej verzii formulárov v riadnej výrobe. Novovytvorená vetva nesie názov danej modifikácie. Týmto krokom som si vytvoril vhodné izolované prostredie pre vývoj a mohol som sa vrhnúť do samotnej modifikácie.

Vývoj modifikácie prebieha v SyteLine v editačnom móde. Z pohľadu frontendu je možné v editačnom móde na formulár pridávať, odoberať a meniť jednotlivé komponenty formulára.

Data formulára sú prezentované pomocou business objektov, ktoré sa nazývajú Intelligent Data Object (IDO). Jednotlivé IDO sú súčasťou IDO projektu. IDO projekty sú uložené v objektovej databáze. Samotné dáta sú uložené v databázových tabuľkách, ktoré sú uložené v aplikačnej databáze. IDO formulára je referencované ku konkrétnej tabuľke, kde sú uložené dáta daného formulára a tak vytvára väzbu medzi objektovou a aplikačnou databázou.

Backend aplikácie je možné vyvíjať priamo v editačnom móde napríklad nastavovaním rôznych vlastností komponentov. Vytváranie component classov tento proces rozširuje a prenáša na viac komponentov. Ďalšou možnosťou ako obstarať chovanie formulára je vytváranie event handlerov. Event hadler je reprezentovaný sekvenciou funkcii. Tieto funkcie môžu byť rôzneho typu. Medzi najpoužívanejšie v mojom prípade by som zaradil volanie SQL uloženej procedúry so vstupnými a výstupnými parametrami, ktorý sa nazýva Method call. Druhým u mňa veľmi často používaným typom funkcie bolo volanie Form script method. Ide o .Net funkcie, ktoré sú písané v jazyku Visual Basic a nazývajú sa ActiveX. Každý formulár má svoj ActiveX script, ktorý sa nazýva aj Form Script. V rámci event handleru je možné volať na pomoc aj iný event handler. Vytvorené event handlery následne môžeme použiť ako primárnu udalosť, pri zmene dát, pri uložení, pri vytvorení nového záznamu, ako úlohu na pozadí a pod. Primárna udalosť a zmena dát sa spájajú s konkrétnymi komponentami alebo component classami. Event handler pri vytvorení nového záznamu formulára a event handler pri uložení dát formulára sú špeciálne

event handlery, ktoré niesú súčasťou atributu pre event na componente, ale sú volané pri operáciach uloženia alebo vytvorenia nového záznamu. Na ochranu vstupu ale aj na validovanie iných componentov som využil validátory.

Po dokončení vývoja vzniklo niekoľko mnou definovaných užívateľských verzii formulárov. Aby ich bolo možné testovať, musel som dostať moje uživateľské verzie formulárov do verzie Site - default a tak sprístupniť mnou modifikované formuláre všetkým používateľom danej konfigurácie systému u nás vo firme, a teda aj samotnému konzultantovi. Tento proces zabezpečoval interný nástroj Scripting, ktorý porovnáva jednotlivé verzie formulárov na základe výberu programátora. Jednotlivé zmeny zobrazí programátorovi, ktorý si ich skontroluje. Následne je možné pomocou nástroja tieto zmeny vyskriptovať. Po vyskriptovaní formulárov som ich aplikoval do danej konfigurácie a zaradil do vytvorenej vetvy spolu s ostatnými skriptami, ktoré vznikli v priebehu modifikácie. Skripty je nutné radiť do štandardne definovanej adresárovej štruktúry vetvy. Aplikačné scripty sa radia do adresára AppDB. Formulárové scripty, ktoré som vyskriptoval pomocou nástroja Scripting sa radia do adresára FormDB. Jednotlivé adresáre obsahujú ďalšie podadresáre ako napríklad DataTypes, ktoré obsahujú uživateľsky definované dátové typy, StoredProcedures, ktoré opsahujú uložené procedúry a mnoho ďalších na základe typu skriptu. Jednotlivé skripty počas vývoja postupne aplikujem do internej konfigurácie. Názvy a verzie konfigurácie Sytelinu sú zväčša pri tovrbe modifikácie pre zákaznika zhodné s názvom a verziou danej zákaznickej firmy. Formulárové skripty po vyskriptovaní pomocou interného nástroja Scripting taktiež aplikujem do tejto konfigurácie. Vďaka tomu sa z mnou uživateľsky definovaných formulárov stali forumláre verzie Site - default a porces testovania môže začať.

Test prevádza konzultant. Samotné ladenie prebieha formou komunikácie medzi konzultantom a programátorom. Ak konzultant narazí na chybu, oznámi ju programátorovi a ten sa ju snaží odstrániť. Následne ak celá modifikácia funguje ako má a konzultant potvrdí spravnosť modifikácie, prichádza na rad `merge` všetkých scriptov novej vetvy s hlavnými vetvami. V tomto štádiu vývoja sú to zatial iba konkrétne vetvy dev a test.

Ako prvé je nutné previesť na lokálnych hlavných vetvách dev a test operáciu `pull`, ktorá zabezpečí, že sa z originu natiahnú všetky zmeny danej vetvy, na ktorých pracovali iní členovia vývojového tímu a pridali skripty svojich modifikácii do týchto vetiev od momentu, kedy som si založil novú vetvu z vetvy live. Týmto krokom si programátor zabezbečí, že má na jeho lokálnych hlavných vetvách aktuálnu verziu. Následne prevediem `merge` modifikačnej vetvy s vetvou dev a s vetvou test.V tejto časti môže dôjsť ku konfliktom. Konflikty vznikajú napríklad ak počas vývoja modifikácie pridal do hlavnej vetvy (dev alebo test) iný programátor modifikačné skripty a prípadne modifikoval funkcionalitu v rovnakom skripte a v rovnakom riadku ako ja. V tomto prípade sa rieši konflikt buď ručne a to porovnávaním skriptov a následným zlučovaním bez straty funkcionality, alebo pomocou nástroja programu SmartGit, ktorý sa volá Resolver. Tento nástroj ponúkne riešenie konfliktu zobrazením troch skriptov. Prvý skript predstavuje verziu konfliktného skriptu, ktorá sa nachádza v hlavnej vetve. Druhý skript obsahuje verziu skriptu v modifikačnej vetve a tretí skript obsahuje návrh zlúčenie týchto dvoch skriptov. Ak programátor

potvrdí správne zlúčenie skriptov, môže použiť operáciu resolve, alebo upraviť skript podľa seba. Po úspešnom mergovaní s oboma vetvami prevedie programátor operáciu `push` na oboch hlavných vetvách. Tá zabezpečí, že sa všetky vykonané zmeny v skriptoch prevedú do originu (na server). Týmto krokom sa proces radenia skriptov končí a nasleduje aplikácia modifikácie do testu ku zákaznikovi.

Aplikácia skriptov sa preváza s použitím všetkých skriptov z danej modifikácie z vetvy test. Je to z dôvodu zachovania funkcionality, ktorá sa v konifigurácii test nachádza. Vetva test obsahuje zlúčené skripty so všetkými modifikáciami pre konfiguráciu test a preto je to navhodnejšie prostredie, odkiaľ ich možno vziať. Na to mi slúži v aplikácii SmartGit dodatková funkcionalita - Save changes between two commits - ktorá umožňuje tieto skripty z vetvy test izolovať a uložiť na mnou vybrané miesto. Výhodou tejto dodatkovej funkcionality je aj uloženie skriptov do vybraného adresára v štandardnej adresárovej štruktúre. Zobrazením stromovej štruktúry vývojovej vetvy test a následným odfiltrovaním si zobrazíme graficky všetky `commity` modifikácie. Následne označím prvý a posledný `commit` a pomocou operácie Save changes between two commits uložím všetky zmeny medzi týmito dvoma `commitmi`. Skripty sú tak pripravené na aplikáciu ku zákazníkovi. Prihlásime sa cez pripojenie k vzdialenej ploche ku zákazníkovi na server a aplikujeme všetky skripty pomocou interného nástroja pre aplikáciu skriptov. Tento nástroj prevádza aplikáciu skriptov na základe štandardnej stromovej štruktúry.

Po aplikácii na zákazníckej konfigurácii test testuje modifikáciu zákazník. Po úspešnom otestovaní a schválení modifikácie zákaznikom prichádza na rad zlučovanie skriptov modifikácie s vetvou live a ich aplikáciou do zákazníckej konfigurácie live. Od tohoto momentu sa modifikácia používa v riadnej výrobe.

4.1.2 Zoznámenie sa s aplikáciou InduStream

Vývoj InduStreamu prebieha prostredníctvom metodiky vývoja softwaru SCRUM. Dva-krát do týždňa sa vývojový tím schádza na standupe, kde si hovoríme, čo všetko sme realizovali a čo ešte treba realizovať do nasledujúceho šprintu. Šprint prebieha každé 2 týždne.

Jednotlivé modifikácie sa realizujú prostredníctvom interného nástroja - Designer, ktorý je podobný editačnému módu SyteLinu. Industream je zložený z takzvaných modulov, ktoré tvoria formuláre. V moduloch je pomocou Designera možné obdobne ako v SyteLine pridávať, odoberať alebo meniť komponenty z pohľadu frontendu. Backend je možné vyvíjať programovaním modulu na platforme .Net. Nástroj Designer umožňuje zobrazenie zdrojového kódu modulu.

InduStream je možné napojiť na ERP systém SyteLine, ktorý sprostredkuje dáta pomocou IDO kolekcii. V rámci IDO môžeme pridávať aj uložené procedúry pre podporu vývoja funkcionality nad dátami.

4.2 Štúdium testovania softwaru

Pred tým ako som sa pustil do automatizovania procesu testovania aplikácii Infor CloudeSuite Industrial a InduStream vo firme, bolo nutné aby som si niečo o testovaní naštudoval. V tejto kapitola sa budem snažiť čitateľovi priblížiť mnou získané poznatky o testovaní softwaru.

4.2.1 Čo je testovanie softwaru

Testovanie softwaru je noddelitelnou súčasťou procesov pri výrobe softwarervého produktu pre zaručenie kvality softwaru. Ide o proces, ktorého hlavným cieľom je vyhodnotiť, či softwárový produkt splňuje stanovené požiadavky a v neposlednom rade identifikovať chyby aby sme zabezpečili vývoj bezchybného a kvalitného softwarového produktu [10] [11] [12] [13].

Podľa štandardu ANSI/IEEE 1059 je testovanie softwaru definované ako proces analýzy softwarového produktu pre zistenie rozdielov medzi existujúcimi a požadovanými podmienkami pre vyhodnotenie funkcií softwarového produktu.

Štandard IEEE 610.12-1990 definuje testovanie ako proces prevádzkovania systému alebo komponentu za špecifických podmienok, sledovanie alebo zaznamenávanie výsledkov a vyhodnocovanie niektorých aspektov daného systému alebo komponentu.

4.2.2 Základné pojmy

V rámci testovania softwaru existuje niekoľko základných pojmov:

- **Error** - chyba, ktorá je spôsobená ľudským omylom. Tieto chyby nazývame aj *mistakes*. Ak sa jedná o chybu ktorú človek spôsobí počas programovania, túto chybu nazývame *bug*.
- **Fault** - porucha, ktorá je reprezentovaná ako výsledok erroru. Existujú dva typy porúch. Porucha, ktorá vzniká po vytvorení nekorektnej reprezentácie softwaru sa nazýva porucha provízie. Na druhú stranu, porucha, ktorá je spôsobená vytvorením korektnej reprezentácie softwaru sa nazýva porucha opomenutia. Druhý typ poruchy je mnohonásobne ťažšie detekovateľný a vyriešitelný.
- **Failure** - stav, ktorý nastáva po vzniku poruchy(fault).
- **Incident** - príznak spojený s failure, ktorý upozorňuje na výskyt failure.
- **Test** - akt skúšania softwaru pomocou testovacích prípadov (test-cases). Test ma 2 jednoznačné ciele: násjť chybu alebo preukázať správne konanie.
- **Test case** - testovací prípad, ktorý je jedinečný a reprezentuje určité chovanie testovaného programu. Taktiež obsahuje sadu vstupov a očakávaných výstupov.

4.2.3 Typy testovania softwaru

Testovanie softwaru môže byť rôzne. V tejto časti sa budem snažiť vysvetliť antonimicky odlišné typy testovania:

- Manuálne vs. automatizované testovanie
- Statické vs. dynamické testovanie
- Black-box vs. white-box testovanies

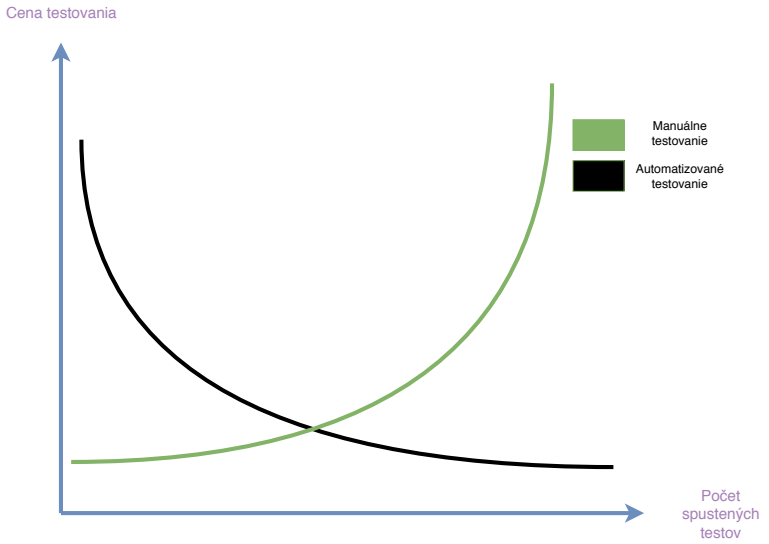
4.2.3.1 Manuálne vs. automatizované testovanie

Manuálne testovanie - proces testovania softwaru manuálne, teda ručne, ktorý slúži pre odhalenie toho, čo v aplikácii funguje či nefunguje. Tester pri manuálnom testovaní určitého procesu v rámci aplikácie využíva testovacie vstupy a kontroluje testovacie výstupy na základe požiadaviek zákazníka. Okrem toho môžeme manuálne testovanie využiť aj na spoznávanie systému. Pri manuálnom testovaní je nutné, aby sa tester vžil do role zákazníka, ktorý bude so systémom neskôr pracovať a snažil sa simulovať jeho chovanie.

Automatizované testovanie - proces testovania softwaru prostredníctvom testovacieho nástroja pre hľadanie chýb v systéme. Vďaka rôznym testovacím nástrojom je možné simulovať manuálne testovanie, nieje však možné úplné nahradenie manuálneho testovania z dôvodu testovacích prípadov, kedy nastanú neočakávané zmeny v aplikácii.

V procese automatizácie testovania testeri spúšťajú testovacie skripty a generujú výsledky jednotlivých testov automaticky prostredníctvom testovacích nástrojov. Využíva sa hlavne pri neustále sa opakujúcich testovaných scenárov aplikácie a taktiež ako náhrada veľmi rozsiahlých maunálnych testov. Jeho výhodou je hlavne rýchlosť, úspora ľudského faktoru a zvýšenie kvality softwarového produktu.

V rámci firmy ITeuro, a.s. sa využíva manuálne testovanie systémov Infor CloudeSuite Industrial a InduStream prostredníctvom konzultantov. Veľa procesov sa v rámci testovacích scenárov opakuje. Myšlienka zmenšenia nákladov na testovanie, úspora času konzultantov a tým pádom väčšie množstvo času pre pokrytie iných procesov ako napríklad kominikáciu so zákazníkmi, predstavuje v najbližších rokoch pre firmu jasný cieľ - zavedenie automatizovaného testovania.



Obr. 3: Graf znázorňujúci porovnanie manuálneho a automatizovaného testovania v závislosti na cene a počte testov [14].

4.2.3.2 Statické vs. dynamické testovanie

Statické testovanie - pri tomto druhu testovania nedochádza k spúšťaniu testovaného programu. Testovanie je prevádzané formou analýzy a overovania zdrojového kódu na logickej úrovni, pretože teoreticky je možné dokázať, že sa program správa tak ako má. Využitie statického testovania je hlavne v malých moduloch, na významných miestach, ktroých funkčnosť a správnosť je kľúčová (Tim A. Majchrzak to prirovnáva ku kontrole jadrovej elektrárne).

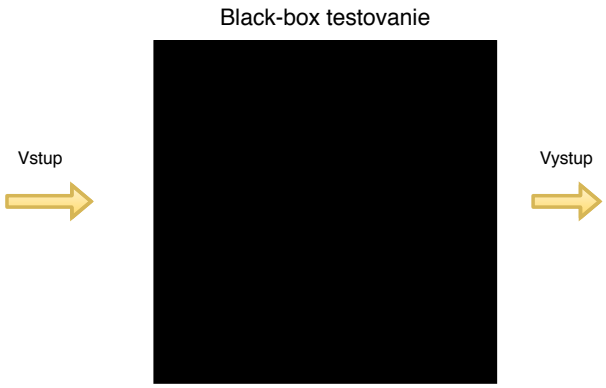
Dynamické testovanie - oproti statickému testovaniu sa proces testovania vykonáva počas spustenej aplikácie v testovacom prostredí, kde sa kotroluje očakávané správanie testovanej aplikácie na základe špecifikácie a hľadajú sa chyby.

4.2.3.3 Black-box vs. white-box testovanie

Black-box testovanie - pri tomto druhu testovania testorovi záleží na tom, či testovaná aplikácia funguje na základe funkčnej špecifikácie. Špecifikom black-box testovania je, že tester nepozná zdrojový kód testovanej aplikácie a teda vyhodnocuje správnosť fungovania aplikácie na základe zadaných vstupov a očakávaných výstupov.

Keďže tester nemusí poznať zdrojový kód testovacieho produktu, nemusí tak ani vôbec poznať programovací jazyk, na ktorom je systém postavený. Testy tak môžu byť vykonávané bez prítomnosti programátorov a preto môžu byť objektívnejšie.

Nevýhodou black-box testovania je obmedzenie pre testovanie väčšieho množstva validných vstupov.

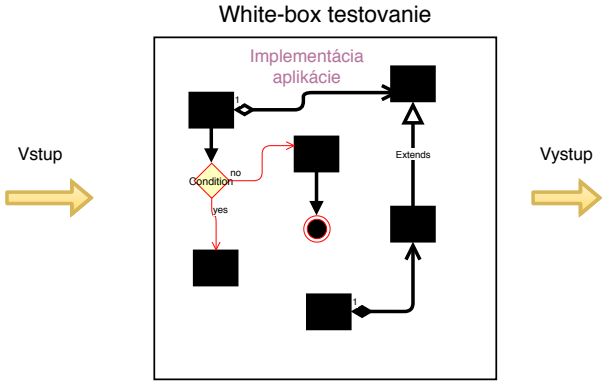


Obr. 4: Grafické znázornenie black-box testovania [15].

White-box testovanie - nazývané tiež „Glass-box testing“ je testovanie, pri ktorom tester pozná zdrojový kód testovanej aplikácie. Vďaka tejto skutočnosti vie tak tester lepšie odhaliť, kde v kóde aplikácie mohla nastať potencionálna chyba. Samotný návrh testovacích scenárov môže prísť už v momente, kedy ešte neexistuje uživateľské rozhranie aplikácie.

Výhodou tohoto testovania je, že tester môže otestovať aplikáciu za asistencie širšieho množstva vstupov rôznej validity a tak dosiahnuť tú najvyššiu kvalitu aplikácie.

Medzi nevýhody sa radí nutnosť znalosti programovacieho jazyka.



Obr. 5: Grafické znázornenie white-box testovania [15].

Gray-box testovanie - špeciálny druh testovania, pri ktorom dochádza ku kombinácii white-box a black-box testovania. Tester v tomto prípade nepozná zdrojový kód ale pozná vnútornú logiku systému. Tento typ testovania som navrhol aj v rámci mojej bakalárskej praxe. Samotná

demonštrácia gray-box testovania je preukázaná kontrolou požadovaných vstupov alebo výstupov dotazom do databáze.

4.2.4 Levely testovania softwaru

Pre zaručenie čo najväčšej kvality a efektívnosti testovania softwaru je treba pokryť čo najväčšie množstvo testovacích prípadov rôznych druhov. Podľa toho z akého uhla pohľadu a do akej hĺbky testujeme softwarový produkt, rozlišujeme niekoľko levelov testovania softwaru. V tejto časti sa budem snažiť priblížiť základné levely testovania softwaru.

4.2.4.1 Unit testy

Jedná sa o testy častí kódu. V objektovo orientovanom programovaní môžeme chápať časť kódu ako metódy jednotlivých tried, kde sa prostredníctvom vstupných parametrov posielajú do metódy vstupy a po vykonaní metódy kontrolujú výstupy. Tieto testy prevádza programátor, čo má za následok, že úroveň špecifikácie testovaných prípadov nemusí byť tak vysoká. Je to v dôsledku toho, že vývojari vykonávajú white-box testing (poprípade gray-box testing), čo znamená že poznajú svoj vlastný kód. Tento druh testovania býva dosť často podceňovaný.

4.2.4.2 Integračné testy

Testy, ktoré sa vykonávajú v procese integrácie alebo zlučovania softwaru. Hlavnou úlohou týchto testov je otestovať ako jednotlivé časti systému na seba vplývajú, či je ich vzájomná komunikácia v súlade so špecifikáciou a nespôsobujú tak neočakávané chovanie softwaru. Medzi tieto testy sa radia okrem integrácie kódu aj integrácia hardwaru, alebo integrácia rôznych operačných systémov pre danú aplikáciu.

4.2.4.3 Systémové testy

V tejto fáze dochádza k testovaniu softwarového produktu ako takého z pohľadu užívateľa. Tester nepozná zdrojový kód aplikácie a teda vykonáva black-box testing. Z toho dôvodu je nutná kvalitná dokumentácia systému a vytvorenie podrobných testovacích scenárov. V rámci tejto fázy sú z aplikácie odstránené mockovacie objekty, ktoré simulovali správanie chýbajúcich komponentov z predchádzajúcich fází, pretože ide o poslednú fázu pred odovzdaním aplikácie zákazníkovi.

4.2.4.4 Akceptačné testy

Konečná fáza testovania softwaru prostredníctvom zákazníka. Ide o schválenie aplikácie na základe testu zákazníkom. Aby sme mohli pristúpiť k tejto fázy, je nutné, aby sme mali splnené predchádzajúce fázy testovania.

4.2.4.5 Performance testy

Testy, ktoré skúmajú výkon aplikácie prostredníctvom záťaže systému. Je vhodné ich vykonávať po integračných testoch, aby sme mali zaručenú integráciu všetkých komponentov. Vďaka týmto testom dokážeme odhaliť chyby aplikácie, ktoré môžu vzniknúť pri potencionálne reálnej záťaži v prevádzke, čím zabezpečíme väčšiu stabilitu systému.

4.3 Vytvorenie testovacích scenárov aplikácii SyteLine a InduStream

Pred začiatkom implementácie automatizovaných testov bolo potrebné vytvoriť testovacie scenáre, podľa ktorých sa budú automatizované testy realizovať. Toto riešenie som riešil spolu s konzultantmi, ktorí prevádzajú testy aplikácii manuálne. S ich pomocou sme navrhli štyri najčastejšie testované procesy v aplikáciach SyteLine a Industream, ktoré by bolo vhodné automatizovať a tak zefektívniť proces testovania vo firme:

1. Test zahájenia a ukončenie práce v InduStreame.
2. Vytvorenie faktury prijatej v SyteLine.
3. Test zadania a zobrazenia poznámok v InduStreame.
4. Vytvorenie faktury vydanej v SyteLine.

V tejto časti zoznámim čitateľa so spomínanými testovanými scenármi.

4.3.1 Test zahájenia a ukončenie práce v InduStreame

Tento tento testovací scenár slúži v systéme pre evidenciu zahájenia a ukončenia práce rôznych operácii. Skladá sa z nasledujúcich krokov:

1. Uživateľ otvorí aplikáciu InduStream, prihlási zadaním užívateľského mena a hesla (tento krok sa prevádza iba pri prvom spustení aplikácie na danom zariadení) a ďalej sa prihlási ako zamestnanec zadaním alebo kliknutím čisla 2 a klikne na tlačidlo prihlásiť.
2. Otvorí sa modul „Bezpapírová dílna“, kde na formulári „Plán práce“ užívateľ vyhľadá a klikne na riadok v gride splňujúci nasledujúce podmienky:
   - V stĺpci „Stav“ sa nenachádza žltá šípka, ktorá indikuje zahájenú operáciu.
   - V stĺpci „Lze donončit“ je hodnota väčsia ako nula.
   - V stĺpci „Zpětné hlášení“ je hodnota rovná nule.

   Výber v gride je indikovaný označením vybraného riadku červeným obdĺžnikom.
3. Užívateľ klikne na tlačidlo „Zahájit“ a otvorí sa formulár pre transakciu „Zahájení práce“, kde sa zvaliduje pole „Výrobný príkaz (VP)“, „Operácia“ a zobrazí sa „Popis položky“ na základe výberu riadka v gride.

4. Užívateľ klikne na tlačidlo „Zahájit práci“ - objaví sa hláška „[Začátek běhu] bylo úspěšné.“ a klikne na tlačidlo „OK“. Hláška zmizne a objaví sa opäť základný formulár modulu „Bezpapírová dílna“ - „Plán práce“.

5. Na formulári je riadok, ktorý bol zahájený, označený žltou šípkou v stĺpci „Status“, označenie zostáva na tomto riadku a na mieste, kde bolo tlačidlo „Zahájit“ sa teraz obajvuje tlačidlo „Ukončit“.

6. Uživateľ klikne na tlačidlo „Rozpracováno“ - otvorí sa formulár, kde sa nachádzajú všetky zahájené operácie v gride a teda aj náš testovaný záznam.

7. Užívateľ otvorí SyteLine, prihlási sa prostredníctvom užívateľskeho mena a hesla a otvorí formulár „Zpracování chyb VP“.

8. Na tomto formulári sa zobrazuje nová transakcia typu „Zač. běhu“. V poli „Terminál“ je „JOB“ a v poli zamenstnanec je 2. VP, suffix a operace a je ten ktorý sme v InduStreame zahájili. „Čas zahájení“ odpovedá času, kedy bolo stlačené tlačidlo „Zahájit práci“ v InduStreame. V poli „Stav“ je hodnota „Čekající“. V poli „Datum transakce“ je dátum, kedy užívateľ stlačil tlačidlo „Zahájit práci“ v InduStreame. Sú vyplnené polia „Mzdová sazba“ a „Sazba nákladů VP“. V poli „Celkem hodin“ je nula a pole „Čas ukončení“ je prázdne.

9. Užívateľ sa vráti späť do InduStreamu, kde stojí na riadku u ktorého zahájil prácu na formulári „Plán práce“. Následne klikne na tlačidlo „Ukončit“ v hornej lište, ktoré je na pozíci, kde bolo tlačidlo „Zahájit“.

10. Objaví sa formulár transakcie „Ukončení práce“, v ktorej sa predvyplní „VP“, „Operace“ a „Položka“. Zároveň sa zobrazia aj informácie o hodnotách „Uvolněno“ a „Lze dokončit“.

11. Užívateľ zadá do poľa „Hotovo“ hodnotu 1 a prejde na ďalšie pole pomocou tabulátoru alebo myši. Týmto krokom sa skopíruje hodnota z poľa „Hotovo“ do poľa „Přesun“. Tým sa sprístupní tiež tlačidlo „Ukončit práci“

12. Užívateľ klikne na tlačidlo „Ukončit práci“. Objaví sa hláška „[Konec běhu] bylo úspěšné.“ Užívateľ potvrdí hlášku stlačením tlačidla „OK“. Následne hláška a formulár transakcie „Ukončení práce“ zmiznú a objaví sa základný formulár modulu „Bezpapírová dílna“ „Plán práce“. Žltá šípka zmizne a tlačidlo „Ukončit“ je opäť nahradené tlačidlom „Zahájit“. V tomto momente sa daný riadok už nezobrazuje v gride na formulári „Rozpracováno“.

13. Opäť užívateľ otvorí SyteLine, prihlási sa a otvorí formulár „Zpracování chyb VP“.

14. Na tomto formulári sa zobrazuje pôvodná transakcia, ktorá ma teraz typ transkacie „Konec běhu“ a bola novo doplnená hodnota do poľa „Čas ukončení“, ktorá odpovedá času, kedy

bolo kliknuté na tlačidlo „Ukončit práci“ v InduStreame. V poli „Hotovo“ a „Přesunuto“ je hodnota 1.

15. Užívateľ klikne na tlačidlo „Převod“ a transakcia zo zoznamu zmizne.

4.3.2 Vytvorenie faktury prijatej v SyteLine

Tento testovací scenár slúži v systéme pre vytváranie faktúr prijatých. Postupnosť krokov pre správnu realizáciu procesu vytvorenia faktúry prijatej je nasledujúca:

1. Užívateľ otvorí SyteLine, prihlási sa a otvorí formulár „Faktury přijaté“.

2. Otvorí sa formulár „Faktury přijaté“ v lokálnom filtri(nezobrazujú sa dáta). Užívateľ následne klikne na tlačidlo pre filtráciu a po odfiltrovaní klikne na tlačidlo pre založenie nového záznamu.

3. Pri zakladaní nového záznamu sa na formulár „Faktury přijaté“ predvyplnia polia „Datum faktury“ a „Datum pro DPH“ aktuálnym dátumom. Následne užívateľ vyberie typ faktúry prijatej výberom z comboboxu v poli „Typ“.

4. Po výbere typu faktúry prijatej sa na základe výberu tohto typu dotiahnú hodnoty do polí „Kontace“, „Datum splatnosti“, „Měna“, „Způsob platby“, „Konst. symbol“, „DIČ“, „Účet“, „Text před položkou“ a „Text za položkou“.

5. Následne užívateľ zadá do poľa „Evindenční číslo“ hodnotu „123456“. Systém na to reaguje dotianhnutím hodnoty zadaného evidenčného čísla do poľa „Variabilní symbol“.

6. Užívateľ zadá do poľa „Datum přiznání daně“ dnešný dátum, vybere partnera v poli „Partner“, do poľa „Částka v MM“ zadá hodnotu 240 a klikne na tlačidlo pre uloženie záznamu.

7. Zobrazí sa varovná správa, ktorú užívateľ potvrdí. Vznikne nová faktúra prijatá, čo je potvrdené validáciou v poli „Č. faktury“ jedinečným číslom faktúry prijatej. Užívateľ následne klikne na tlačidlo „Řádek“.

8. Otvorí sa prepojený formulár „Faktury přijaté - Řádky“, kde sa dotiahnú hodnoty „Partner“, „Název“, „Částka v MM“, „Částka v CM“, „Měna“, „Typ“, „Číslo dokladu“, „Kontace“, „Účet“ z hlavičky formulára „Faktury přijaté“. Okrem toho sa predvyplnia aj polia „Č. řádku“ poradovým číslom nového riadku, „Kód DPH“, „Účet MD“, „Účet DPH“ a pole „Účet odchylky“. Užívateľ zadá do poľa „Částka bez DPH“ hodnotu 100, na čo systém reaguje validáciou poľa „Částka DPH“. Užívateĺ takto vytvorí 2 riadky pre danú faktúru a klikne na tlačidlo pre uloženie záznamov.

9. Po uložení vytvorených záznamov sa do polí „Částka celkem“ v každom riadku faktúry uloží hodnota súčtu polí „Částka bez DPH“ a „Částka DPH“. V hlavičke formulára sa v poli „Součet ř. v měně“ uloží súčet polí „Částka celkem“ všetkých riadkov danej faktúry.

10. Užívateľ zatvorí formulár „Faktury přijaté - Řádky“ a vráti sa späť na formulár „Faktury přijaté“, kde sa nachádza na faktúre, ktorú vytvoril a následne jej pridal 2 riadky. V poli „Základ v měne“ je hodnota súčtu polí „Částka bez DPH“ všetkých riadkov danej faktúry. V poli „DPH v měně“ sa nachádza súčet polí „Částka DPH“ všetkých riadkov danej faktúry.

11. Následne užívateľ nastaví stav faktúry výberom z comboboxu v poli „Stav FA“ na „Schváleno“ a klikne na tlačidlo pre uloženie záznamu. Tým sa na formulári „Faktury přijaté“ sprístupní tlačidlo „Zaúčtovať“.

12. Užívateľ klikne na tlačidlo zaúčtovať. Zobrazí sa hláška „[Generovat zúčtování] bylo úspěšné“, ktorú užívateľ potvrdí tlačidlom „OK“. Týmto krokom sa všetky polia pre danú fakturu stanú needitovatelnými.

13. Následne užívateľ klikne na odúčtovať. Zobrazí sa hláška „[Odúčtování] bylo úspěšné“, ktorú užívateľ potvrdí stlačením tlačidla „OK“.

4.3.3 Test zadania a zobrazenia poznámok v InduStreame.

Tento testovací scenár slúži v systéme pre zobrazenie poznámok na výrobných príkazoch, ktoré sa zadávajú v SyteLine:

1. Užívateľ otvorí InduStream, prihlási sa a zadá číslo zamestnanca pre otvorenie modulu „Bezpapírová dílna“, kde si vyberie riadok v gride s výrobným príkazom, ktorého poznámky chce vidieť. Zapamätá si výrobný príkaz, suffix a operáciu.

2. Následne užívateľ otvorí SyteLine, prihlási sa a otvorí formulár „Operace VP“, kde vyhľadá zapamätaný výrobný príkaz zadaním „VP“, „Suffix“ a „Operace“.

3. Užívateľ klikne na tlačidlo pre filtráciu, čím sa vyhľadá príslušný výrobný príkaz. Následne užívateľ klikne na tlačidlo pre poznámky.

4. Otvorí sa formulár, ktorý obsahuje poznámky pre daný výrobný príkaz. Užívateľ vytvorý novú poznámku kliknutím na tlačidlo „Nový záznam“ v hornej lište a do poľa „Předmět“ zadá hodnotu „IteTestMG“. Následne do poľa „Poznámka“ zadá text poznámky a uloží poznámku.

5. Užívateľ otvorí InduStream, prihlási sa a zadá číslo zamestnanca pre otvorenie modulu „Bezpapírová dílna“, kde si vyberie riadok v gride s testovacím výrobným príkazom. Po kliknutí na riadok s konkrétnym výrobným príkazom sa pri tlačidle „Poznámky“ zvýši

číslo reprezentujúce počet poznámok. Následne užívateľ klikne na toto tlačidlo, čím sa otvorí formulár, ktorý pre daný výrobný príkaz a operáciu zobrazuje poznámky.

6. Na tomto formulári bude vidieť novo pridaná poznámka, ktorá bude označená v stĺpci „Operace/Materiál“ číslom operácie, čím dáva systém užívateľovi najavo, že ide o poznámku na operácii.

7. Následne užívateľ otvorí SyteLine, prihlási sa a otvorí formulár „Materiál na VP“, kde vyhľadá zapamätaný výrobný príkaz zadaním „VP“, „Suffix“ a „Operace“.

8. Užívateľ klikne na tlačidlo pre filtráciu, čím sa vyhľadá príslušný výrobný príkaz. Následne užívateľ klikne na tlačidlo pre poznámky.

9. Otvorí sa formulár, ktorý obsahuje poznámky pre daný výrobný príkaz. Užívateľ vytvorý novú poznámku kliknutím na tlačidlo „Nový záznam“ v hornej lište a do poľa „Předmět“ zadá hodnotu „IteTestMG“. Následne do poľa „Poznámka“ zadá text poznámky a uloží poznámku.

10. Užívateľ otvorí InduStream, prihlási sa a zadá číslo zamestnanca pre otvorenie modulu „Bezpapírová dílna“, kde si vyberie riadok v gride s testovacím výrobným príkazom. Po kliknutí na riadok s konkrétnym výrobným príkazom sa pri tlačidle „Poznámky“ zvýši číslo reprezentujúce počet poznámok. Následne užívateľ klikne na toto tlačidlo, čím sa otvorí formulár, ktorý pre daný výrobný príkaz a operáciu zobrazuje poznámky.

11. Na tomto formulári bude vidieť novo pridaná poznámka, ktorá bude označená v stĺpci „Operace/Materiál“ číslom materiálu, čím dáva systém užívateľovi najavo, že ide o poznámku na materiále.

12. Následne užívateľ otvorí SyteLine, prihlási sa a otvorí formulár „Operace VP“, kde vyhľadá zapamätaný výrobný príkaz zadaním „VP“, „Suffix“ a „Operace“.

13. Užívateľ klikne na tlačidlo pre filtráciu, čím sa vyhľadá príslušný výrobný príkaz. Následne užívateľ klikne na tlačidlo pre poznámky.

14. Otvorí sa formulár, ktorý obsahuje poznámky pre daný výrobný príkaz. Užívateľ vyhľadá zadaním hodnoty do poľa „Předmět“ „IteTestMG“ a následne klikom na tlačidlo filtrácie, ním pridanú poznámku, ktorú edituje a uloží.

15. Užívateľ otvorí InduStream, prihlási sa a zadá číslo zamestnanca pre otvorenie modulu „Bezpapírová dílna“, kde si vyberie riadok v gride s testovacím výrobným príkazom. Po kliknutí na riadok s konkrétnym výrobným príkazom sa pri tlačidle „Poznámky“ nemení číslo reprezentujúce počet poznámok. Následne užívateľ klikne na toto tlačidlo, čím sa otvorí formulár, ktorý pre daný výrobný príkaz a operáciu zobrazuje poznámky.

16. Na tomto formulári bude vidieť zmenu práve tej poznámky, ktorá bola označená v stĺpci „Operace/Materiál“ číslom operácie, čím dáva systém užívateľovi najavo, že ide o poznámku na operácii.

4.3.4 Vytvorenie faktury vydanej v SyteLine

Tento testovací scenár slúži v systéme pre vytváranie faktúr vydaných. Proces vytvorenia faktúry vydanej je opísaný v nasledujúcich krokoch:

1. Uživateľ otvorí SyteLine, prihlási sa a otvorí formulár „Faktury vydané“.
2. Otvorí sa formulár „Faktury vydané“ v lokálnom filtri(nezobrazujú sa dáta). Užívateľ následne klikne na tlačidlo pre filtráciu a po odfiltrovaní klikne na tlačidlo pre založenie nového záznamu.
3. Pri zakladaní nového záznamu sa na formulár „Faktury vydané“ predvyplnia polia „Datum vystavení“ a „Datum pro DPH“ aktuálnym dátumom. Následne užívateľ vyberie typ faktúry vydanej výberom z comboboxu v poli „Typ“.
4. Po výbere typu faktúry vydanej sa na základe výberu tohto typu dotiahnú hodnoty do polí „Kontace“, „Datum splatnosti“, „Měna“, „Způsob platby“, „Konst. symbol“, „DIČ“, „Účet“, „Text před položkou“ a „Text za položkou“. Následne užívateľ vyberie partnera z comboboxu v poli „Partner“ a klikne na tlačidlo pre uloženie záznamu.
5. Zobrazí sa varovná správa, ktorú užívateľ potvrdí. Vznikne nová faktúra vydaná, čo je potvrdené validáciou v poliach „Č. faktury“, „Evidenční číslo“ a „Variabilní symbol“ jedinečným číslom faktúry. Užívateľ následne klikne na tlačidlo „Řádek“.
6. Otvorí sa prepojený formulár „Faktury vydané - Řádky“, kde sa dotiahnú hodnoty „Partner“, „Název“, „Částka v MM“, „Částka v CM“, „Měna“, „Typ“, „Číslo dokladu“, „Kontace“, „Účet“ z hlavičky formulára „Faktury vydané“. Okrem toho sa predvyplnia aj polia „Č. řádku“ poradovým číslom nového riadku, „Kód DPH“, „Účet DAL“ a pole „Účet DPH“. Užívateľ zadá do poľa „Částka bez DPH“ hodnotu 100.
7. Do poľa „Částka DPH“ sa zvaliduje hodnota čiastky DPH. Užívateľ týmto spôsobom vytvorí 20 riadkov pre danú faktúru a uloží ich.
8. Po uložení riadkov sa do polí „Částka celkem“ v každom riadku uloží hodnota súčtu polí „Částka bez DPH“ a „Částka DPH“. V hlavičke formulára sa v poli „Částka v MM“ a „Částka v CM“ uloží súčet polí „Částka celkem“ všetkých riadkov danej faktúry.
9. Užívateľ zatvorí formulár „Faktury vydané - Řádky“ a vráti sa späť na formulár „Faktury vydané“, kde sa nachádza na faktúre, ktorú vytvoril a následne jej pridal 20 riadkov. V poli „Částka v MM“, „Částka v CM“ a „Částka k úhradě“ sa nachádza súčet polí „Částka celkem“ všetkých riadkov danej faktúry.

10. Užívateľ klikne na tlačidlo zaúčtovať. Zobrazí sa hláška „[Generovat zúčtování] bylo úspěšné“, ktorú užívateľ potvrdí tlačidlom „OK“. Týmto krokom sa všetky polia pre danú faktúru stanú needitovatelnými.

11. Následne užívateľ klikne na tlačidlo odúčtovať. Zobrazí sa hláška „[Odúčtování] bylo úspěšné“, ktorú užívateľ potvrdí stlačením tlačidla „OK“.

4.4 Implementácia testovacích scenárov aplikácii SyteLine a InduStream

Po úspešnom návrhu testovacích scenárov som sa pustil do vývoja sady testov, ktoré budú tieto scenáre automaticky simulovať. V tejto časti sa budem snažiť priblížiť čitateľovi priebeh vývoja automatizovaných testov v rámci praxe.

4.4.1 Návrh implementácie

Pred samotnou implementáciou som potreboval navrhnúť, akým spôsobom budem samotné testy implementovať. Návrh som si rozdelil do troch častí.

Prvá časť bol výber typu projektu vo Visual Studiu. Keďže bolo mojou úlohou vytvoriť sadu automatizovaných testov, rozhodol som sa využiť podporu vytvárania automatizovaných testov v rámci Visual Studia. Preto som zvolil základný testovací projekt „Unit Test Project“, ktorý je založený na .Net Frameworku. Základom tohto projektu je testovacia trieda, ktorá je označená atribútom *[TestClass]* . Testovacia trieda ďalej obsahuje testovacie metódy, ktoré sú označené atribútom *[TestMethod]*. Jedenotlivé testy je teda možné spúšťať v rámci *Test Exploreru*, ktorý po simulácii testu zobrazí testerovi jeho výsledok, teda či prešiel (bol úspešný) alebo neprešiel (nastala chyba).

Druhou časťou návrhu implementácie bol spôsob, ako budú jednotlivé testovacie scenáre implementované. Jednotlivé testovacie scenáre musia byť navzájom od seba nezávislé. Preto som sa rozhodol, že v rámci testovacej triedy bude jedna testovacia metóda predstavovať jeden testovací scenár. V rámci testovacej metódy môžu nastať teda dve možnosti. Ak bude priebeh simulácie testovacieho scenára bezchybný, je vyhodnotený ako úspešný. Ak nastane v rámci simulácie chyba, test nebude dokončený, spustená aplikácia sa vypne a test sa vyhodnotí ako neúspešný. Týmto zabezpečíme, že nebudú jednotlivé testy na sebe závislé.

Treťou časťou návrhu bol výber návrhového vzoru pre implementáciu jednotlivých testov. V mojom prípade to bol návrhový vzor Page Object Model (POM). Podrobnejšiu špecifikáciu návrhového vzoru POM priblížim v nasledujúcej časti tohto dokumentu.

4.4.2 Page Object Model

Návrhový vzor POM, ktorý som si vybral pre implementáciu simulácie testovacích scenárov, je charakteristický tým, že jedna trieda obsahuje testovaciu implementáciu jedného formulára (page). Vytvára sa teda akýsi objektový repozitár pre objekty typu *IWebElement*, ktoré predstavujú jednotlivé elementy formulára. Okrem elementov obsahuje tento repozitár metódy, ktoré

reprezentujú operácie nad týmito elementami. Sú to hlavne základné operácie nad elementami ako:

- nájdenie elementu,
- získanie textu daného elementu,
- prevedenie operácie kliku na daný element,
- vloženie textu do elementu a pod.

Zlučovaním základných operácii nad elementami vznikajú aj zložitejšie metódy, ktoré implementujú funkcionalitu v rámci testovacieho scenára.

Výhodou POM je redukcia duplikácii kódu, pretože môžeme pri opakovaní nejakej operácie nad formulárom len znovu využiť danú metódu a nie všetok kód, ktorý sa v nej nachádza. Ďalšou výhodou je udržateľnosť kódu. Predstavme si situáciu, že sa implementácia nejakého formulára zmení. Ak sa zmení implementácia formulára, je dosť pravdepodobné, že sa zmení aj jeho testovacia implementácia. Ak by sme v rámci testovacej implementácie nevyužívali POM, tak by sme riskovali zmenu kódu na viacerých miestach, čo je väčších projektoch veľmi náročné a neefektívne. Vďaka POM túto zmenu prevedieme len na jednom mieste. POM teda zabezpečuje v kóde lepšiu čitateľnosť, znovuvyužitelnosť a udržateľnosť.

```
public class LoginSLPage
{
        private Driver driver;
        private IWebElement userNameEdit;
        private IWebElement passwordEdit;

        public LoginSLPage(Driver driver)
        {
            this.driver = driver;
            this.userNameEdit = this.driver.findWithWaitElementById("
                userNameEdit");
            this.passwordEdit = this.driver.findWithWaitElementById("
                passwordEdit");
        }

        public void setLoginAndPassword(string login,string password)
        {
            this.userNameEdit.Clear();
            this.userNameEdit.SendKeys(login);
            this.passwordEdit.Clear();
```

```
            this.passwordEdit.SendKeys(password);
        }
}
```

Výpis 1: Príklad implementácie návrhového vzoru Page Object Model

4.4.3 Trieda Driver.cs

Na to aby bolo možné simulovať jednotlivé testy, bolo nutné vytvoriť triedu **Driver.cs**. Táto trieda zabezpečuje niekoľko základných procedúr potrebných pre fungovanie testov. Medzi ne patrí:

- inicializácia winium driveru,
- zapínanie/vypínanie príslušných aplikácii,
- hľadanie elementov,
- špeciálne akcie nad elementami.

Pred samotnou inicializáciou winium driveru bolo nutné do projektu pridať NuGet, ktorý nainštaluje a nareferencuje všetky knižnice potrebné pre prácu s frameworkom winium. Tento NuGet sa nazýva **Winium.Elements.Desktop**. Inicializácia winium driveru zahrňuje vytvorenie objektu typu **WiniumDriverService**, ktorému nastavíme cestu k winium driveru. Tým vytvoríme službu, ktorá nám sprístupní winium framework. Následne vytvoríme v rámci metódy pre zapnutie konkrétnej aplikácie objekt typu **DesktopOptions**, ktrému nastavíme atribut *ApplicationPath* na cestu k aplikácii, ktorú chceme testovať. Na koniec vytvoríme objekt typu **WiniumDriver**, ktorému v atributoch konštruktoru pošleme vytvorenú službu a nastavenia v predchádzajúcich krokoch. Tým vznikne objekt, pomocou ktorého sme schopný plne využívať winium framework.

```
public Driver()
{
   this.service = WiniumDriverService.CreateDesktopService(CoreConstants.
       WiniumDriverPath);
}
public void Launch()
{
   this.options = new DesktopOptions { ApplicationPath = CoreConstants.
       InduStreamPath };
   this.winiumDriver = new WiniumDriver(this.service, this.options);
}
```

Výpis 2: Inicializácia winium driveru

Dôležitou súčasťou automatizovaných testov pre zabezpečenie nezávislosti jednotlivých testovacích scenárov je okrem zapínania testovanej aplikácie aj jej vypínanie. Vypínanie aplikácie je zabezpečené zavolaním metody **Kill** na proces behu aplikácie. Celý testovací scenár začína spustením testovacej aplikácie. Následne simulácia scenára beží v **try/catch** bloku, ktorý zabzpečí odchytenie chyby. Chyba nastáva, ak bol evidovaný neočakávaný výstup zachytený metodou **Assert**, alebo ak došlo k nenájdeniu hľadaného elementu, čo indikuje nevyžadujúce správanie aplikácie. Ak dôjde k odhaleniu chyby, tok programu prejde do bloku catch, kde dôjde k vypnutiu aplikácie. Tým zabezpečíme nezávislosť od ostatných testovacích scenárov.

```
public void Quit()
{
        if (this.winiumDriver != null)
        {
              Process[] processes = Process.GetProcessesByName("InduStream");
              foreach (var item in processes)
              {
                  item.Kill();
              }
              this.service.Dispose();
        }
}
```

Výpis 3: Vypnutie aplikácie

Trieda Driver.cs zabezpečuju aj hľadnie elementov, ktoré je pri simulovaní testovacích scenárov veľmi dôležité. Existuje niekoľko spôsobov hľadania elementu pomocou winium driveru. Najefektívnejší a najbezpečnejší spôsom hľadania elementu je pomocou jeho jedinečného idetntifikátoru (AutomationId). Niektoré elementy však neobsahujú jedinečný identifikátor, alebo obsahujú generovaný identifikátor, ktorý môže byť pri každom nájdení elemntu iný. Preto som bol častokrát nútený použiť hľadanie elementu pomocou mena alebo xpathu. Najpomalšie hľadanie elementu v rámci automatizovaného testovania bolo pomocou mena. Najneefektívnejším ale zároveň najrýchlejším hľadaním elementu bolo pomocou xpathu.

V rámci simulácie testovacích scenárov som narazil pár krát na rôzne problémy, kvoli ktorým som bol nútený vyvárať špeciálne operácie nad elementami. Ako príklad by som uviedol výber v comboboxe v aplikácii Infor CloudSuite Industrial. Ak išlo o combobox, ktorý bol v rámci componenty 4150formulára, winium framework poskytoval podporu prostredníctvom objektov **Elements.Desktop**, ktoré sú schopné vykonávať operácie nad základnými componentami. V rámci comboboxu išlo teda o vyhľadávanie konkrétnej položky po rozkliknutí comboboxu. V prí-

pade elementov, ktoré sa pridávajú na formuláre Sytelinu v editačnom móde sú však comboboxy identifikované ako componenty nazývané edit, ktoré slúžia na jednoduché zadanie vstupu. Tým pádom tieto componenty strácajú podporu frameworku winium pre vykonanie operácie hľadania elemntu v comboboxe. Preto som vytvoril špeciálnu operáciu hľadania elementu v comboboxe v SyteLine, ktorý funguje tak, že pomocou špeciálneho objectu **Actions** nasimulujeme metódu vyhľadania v comboboxe spojením akcie kliku na rozbaľovací button a následného hľadania požadovaného itemu v rozbalenom textboxe podľa mena.

```
public void clickOnComboDropdownSL(IWebElement element)
{
   Actions action = new Actions(this.winiumDriver);
   action.MoveToElement(element, element.Size.Width - 10, element.Size.Height -
        (element.Size.Height / 2)).Click().Build().Perform();
}
public IWebElement findAndSelectElemenetInComboSL(IWebElement element, string
    name)
{
   clickOnComboDropdownSL(element);
   Thread.Sleep(1000);
   var listView = this.findWithWaitElementById("listView");
   var result = listView.FindElement(By.Name(name));
        return result;
}
```

Výpis 4: Hľadanie elementu v comboboxe v SyteLine

4.4.4 Report výsledkov

Budúcnosť automatizovaných testov v rámci firmy ITeuro, a. s. smeruje k vytvoreniu testovacieho nástroja, ktorý bude služiť pre automatizovanie systémových alebo akceptačných testov. Za predpokladu absencie Test Exploreru v rámci visual studia nieje užívateľovi poskytnutý report výsledkov. Preto bolo mojou úlohou v rámci praxe vytvoriť reportovanie výsledkov testov.

Reportovanie výsledkov testov som realizoval pomocou knižnice **NLog**. Vytvoril som objekt typu **LoggingConfiguration**. Na tento objekt som zavolal medotu *AddRule*, v ktorej parametroch som poslal tri logovacie levely (Info, Error, Trace) a tri objekty typu **FileTarget**. Objekty typu *FileTarget* odkazujú na tri výstupné súbory: *logPassed.txt*, ktorý slúži na logovanie úspešných testov, *logFailed.txt*, ktorý slúži na logovanie chybných testov a v neposlednom rade *logTrace.txt*, ktorý loguje informácie o vyhľadávaní jednotlivých elementov. Posledný menovaný slúži testerovi pre zobrazenie chyby v prípade hľadania elementu.

```
var config = new NLog.Config.LoggingConfiguration();
```

```
var date = DateTime.Now.ToString("dd-MM-yyyy_HH_mm_ss");

var logfilePassed = new NLog.Targets.FileTarget("logfilePassed")
{
   FileName = "..\\..\\..\\..\\Results" + date + "\\logPassed.txt"
};
var logfileFail = new NLog.Targets.FileTarget("logfileFail")
{
   FileName = "..\\..\\..\\..\\Results" + date + "\\logFailed.txt"
};
var logfileTrace = new NLog.Targets.FileTarget("logfileTrace")
{
   FileName = "..\\..\\..\\..\\Results\\" + date + "\\logTrace.txt"
};

config.AddRule(LogLevel.Info, LogLevel.Info, logfilePassed);
config.AddRule(LogLevel.Error, LogLevel.Fatal, logfileFail);
config.AddRule(LogLevel.Trace, LogLevel.Trace, logfileTrace);

NLog.LogManager.Configuration = config;
```

Výpis 5: Konfigurácia reportovania výsledkov

4.4.5 Vyjadrenie časovej náročnosti jedntolivých úloh

Tabuľka 1: Tabuľka vyjadrenia časovej náročnosti jedntolivých úloh

Úloha	Počet dní
Zoznámenie sa s aplikáciami Infor CloudSuite Industrial a InduStream	17
Štúdium testovania softwaru	12
Vytvorenie testovacích scenárov aplikácii SyteLine a InduStream	7
Implementácia testovacích scenárov aplikácii SyteLine a InduStream	21

5 Záver

5.1 Vedomosti a skúsenosti získané počas štúdia

V rámci štúdia som získal obrovské množstvo poznatkov, ktoré som ako bývalý študent gymnázia oproti spolužiakom z technických stredných škôl nemal. Tieto poznatky tvorili základný stavebný pilier v rámci môjho pôsobenia na praxi.

Z pohľadu predmetov by som rád vyzdvihol predmety zamerané na databázové systémy, ktoré boli potrebné pri práci na vývoji aplikácii Infor CloudSuite Industrial a Industream. Konkrétne by som spomenul Úvod do databázových systémov a Databázové a informačné systémy. Okrem databáz som v rámci praxe využil aj poznatky z predmetov zameraných na programovanie a programovacie jazyky. Konkrétne by som vyzvihol Programovanie I a II, Algoritmy I a II, Programovacie jazyky I a II, Skriptovacie programovacie jazyky, či predmet Architektúra technológie .Net. Pri vytváraní a implementácii testovacích scenárov mi pomohli znalosti predmetov Softwárové inžinierstvo a Vývoj informačných systémov.

5.2 Vedomosti a skúsenosti nadobudnuté počas praxe

Počas praxe som nadobudol veľké množstvo poznatkov, ktoré boli počas štúdia spomenuté len okrajovo alebo neboli súčasťou učebných osnov. Z pohľadu vývoja v tíme som spoznal ako funguje réžia projektu. Medzi hlavné poznatky nadobúdnoté pri vývoji v tíme by som zaradil prácu s gitom, ktorá je pri tímovom vývoji dnes už nevyhnutná.Okrem toho som sa stretol s dnešnými využívanými programovacími sql štandardami v rámci dodávanej aplikácie SyteLine, čo mi pomohlo si prehĺbiť znalosti v rámci databáz. V neposlednom rade som získal podrobnejšie znalosti o testovaní, čo bolo v rámci štúdia spomenuté len okrajovo. Celkovo hodnotím bakalársku prax pozitívne, nakoľko som bol súčasťou skvelého kolektívu, veľa som sa naučil a zároveň som si vyskúšal reálnu programátorskú prax.

Literatura

1. *ITeuro, a. s. Profil spoločnosti ITeuro, a. s.* [online] [cit. 2020-02-17]. Dostupné z: https://www.iteuro.cz/profil-spolecnosti/.
2. *ITeuro, a. s. Infor CloudSuite Industrial (SyteLine) ERP* [online] [cit. 2020-02-17]. Dostupné z: https://www.iteuro.cz/produkty/infor-syteline/.
3. *ITeuro, a. s. Infor CloudSuite Industrial (SyteLine) APS* [online] [cit. 2020-02-17]. Dostupné z: https://www.iteuro.cz/produkty/aps-pokrocile-planovani/.
4. *ITeuro, a. s. InduStream* [online] [cit. 2020-02-17]. Dostupné z: https://www.iteuro.cz/produkty/industream/.
5. *Microsoft. SQL Server Management Studio (SSMS)* [online] [cit. 2020-02-19]. Dostupné z: https://docs.microsoft.com/cs-cz/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15.
6. *Microsoft. Microsoft Visual Studio* [online] [cit. 2020-02-20]. Dostupné z: https://docs.microsoft.com/sk-sk/visualstudio/get-started/visual-studio-ide?view=vs-2019.
7. *Microsoft. What is .Net framework* [online] [cit. 2020-02-21]. Dostupné z: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet.
8. *Winium* [online] [cit. 2020-02-21]. Dostupné z: https://github.com/2gis/Winium.
9. *YouTube. Infor Mongoose Hello World* [online] [cit. 2020-02-23]. Dostupné z: https://www.youtube.com/watch?v=MGXFfrZNbGI&t=599s.
10. MAJCHRZAK, Tim A. *Improving Software Testing: Technical and Organizational Developments*. Berlin Heidelberg: Springer-Verlag Berlin Heidelberg, 2012. ISBN 978-3-642-27464-0. ISSN 2192-4929.
11. GRAHAM, Dorothy; VAN VEENENDAAL, Erik; EVANS, Isabel; BLACK, Rex. *Foundations of software testing: ISTQB Certification*. London: Gaynor Redvers - Mutton, 2008. ISBN 978-1-84480-989-9.
12. *SoftwareTesting, Materials of software testing* [online] [cit. 2020-03-14]. Dostupné z: https://www.softwaretestingmaterial.com/software-testing/.
13. JORGENSEN, Paul C. *Software Testing: A Craftsman's Approach: Fourth Edition*. Auerbach Publications; 4 edition, 2013. ISBN 978-1-4665-6069-7.
14. *SoftwareTesting, Materials of software testing* [online] [cit. 2020-03-16]. Dostupné z: https://www.bdo.com/digital/insights/application-development/manual-testing-vs-automation-testing.
15. *SoftwareTesting, Materials of software testing* [online] [cit. 2020-03-17]. Dostupné z: https://www.codefirst.co.uk/blog/difference-black-white-box-testing/.