Vysoká škola logistiky o.p.s.

Logistické procesy a zpracování testovacích dat v oblasti internetového bankovnictví

Přerov 2019 Jiří Ambroz, DiS.

Vysoká škola logistiky o.p.s.

Zadání bakalářské práce

student	**Jiří Ambroz, DiS.**
studijní program	Logistika
obor	Informační management

Vedoucí Katedry bakalářského studia Vám ve smyslu čl. 22 Studijního a zkušebního řádu Vysoké školy logistiky o.p.s. pro studium v bakalářském studijním programu určuje tuto bakalářskou práci:

Název tématu: **Logistické procesy a zpracování testovacích dat v oblasti internetového bankovnictví**

Cíl práce:

Cílem práce je analyzovat celý proces ukrývající se za distribucí dat a jejich zpracování/ zefektivnění v oblasti testování dat v aplikaci internetového bankovnictví.

Zásady pro vypracování:

Využijte teoretických východisek oboru logistika. Čerpejte z literatury doporučené vedoucím práce a při zpracování práce postupujte v souladu s pokyny VŠLG a doporučeními vedoucího práce. Části práce využívající neveřejné informace uveďte v samostatné příloze.

Bakalářskou práci zpracujte v těchto bodech:

Úvod
1. Logistické procesy v peněžních službách
2. Informační systémy v peněžnictví
3. Zabezpečení přenosu a zpracování dat
4. Testovací procedury
5. Identifikace slabých míst

Závěr

Rozsah práce: 35 – 40 normostran textu

Seznam odborné literatury:

Gros, I., Barančík, I., Čujan, Z.: Velká kniha logistiky. VŠCHT Praha, 2018, ISBN 978-80-7080-952-5

Vymětal, D.: Informační systémy v podnicích: teorie a praxe projektování. Grada 2009, ISBN 978-80-247-3046-2

Szarzecová, M.: Prostředky elektronického bankovnictví. DP MU Brno 2014. [on-line] dostupné z https://is.muni.cz/th/b7dne/Prostredky_elektronickeho_bankovnictvi.pdf [cit. 20.10.2018]

Vedoucí bakalářské práce: doc. Dr. Ing. Oldřich Kodym

Datum zadání bakalářské práce: 31. 10. 2018
Datum odevzdání bakalářské práce: 4. 5. 2019

Přerov 31. 10. 2018

Ing. et Ing. Iveta Dočkalíková, Ph.D.
vedoucí katedry

doc. Ing. Ivan Hlavoň, CSc.
rektor

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a že jsem ji vypracoval samostatně. Prohlašuji, že citace použitých pramenů je úplná a že jsem v práci neporušil autorská práva ve smyslu zákona č. 121/2000 Sb., o autorském právu, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

Prohlašuji, že jsem byl také seznámen s tím, že se na mou bakalářskou práci plně vztahuje zákon č. 121/2000 Sb., o právu autorském, právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména § 60 – školní dílo. Beru na vědomí, že Vysoká škola logistiky o.p.s. nezasahuje do mých autorských práv užitím mé bakalářské práce pro pedagogické, vědecké a prezentační účely školy. Užiji-li svou diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat před tím o této skutečnosti Vysokou školu logistiky o.p.s. prorektora pro vzdělávání.

Prohlašuji, že jsem byl poučen o tom, že bakalářská práce je veřejná ve smyslu zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, zejména § 47b. Taktéž dávám souhlas Vysoké škole logistiky o.p.s. ke zpřístupnění mnou zpracované bakalářské práce v její tištěné i elektronické verzi. Souhlasím s případným použitím této práce Vysokou školou logistiky o.p.s. pro pedagogické, vědecké a prezentační účely.

Prohlašuji, že odevzdaná tištěná verze bakalářské práce, elektronická verze na odevzdaném optickém médiu a verze nahraná do informačního systému jsou totožné.

V Přerově, dne 4. 5. 2019 .....

podpis

Poděkování

Tímto bych rád poděkoval doc. Dr. Ing. Oldřich Kodym za odborné vedení, podnětné a cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Anotace

Tato bakalářská práce je zaměřena na popsání celého procesu ukrývajícího se za distribucí dat a jejich následné zpracování či zefektivnění v oblasti testování dat v aplikaci internetového bankovnictví. V teoretické části je popsána historie internetového bankovnictví společně s definicí služby do které spadá a následně jsou vysvětleny jednotlivé komponenty používané v oblasti internetového bankovnictví společně se zabezpečením datového přenosu. Následně jsou v praktické části uvedeny zásady testování společně s návrhem vlastního zlepšení v oblasti automatického testování, což je společně s následným identifikováním slabých míst hlavním cílem této práce.

Klíčová slova

Internetové bankovnictví, testovací data, data, distribuce, komponenty, služba, proces, produkt

Annotation

This bachelor thesis is focused on describing the whole process behind data distribution and their subsequent processing or streamlining in the field of data testing in internet banking application. The theoretical part describes the history of internet banking together with the definition of the service it belongs to and then the individual components used in the area of internet banking are explained together with the data transmission security. Subsequently, in the practical part, the principles of testing are presented together with the suggestion of self improvement in the field of automatic testing, which together with the subsequent identification of weaknesses is the main aim of this work.

Keywords

Internet banking, test data, data, distribution, components, service, process, product

Obsah

Úvod

V dnešní moderní době se neustále hovoří o takzvané digitalizaci. Pod tímto pojmem se dá zjednodušeně představit, že co nejvíce administrativních či finančních úkonů by mělo jít vykonávat pomocí internetu. Jedna z prvních věcí, která se mezi ně dá zařadit jsou bankovní operace či služby. Tyto služby a operace vykonává internetové bankovnictví, které nám slouží jako prostředník mezi námi lidmi a bankovními institucemi. Lidé jsou schopni přes internetové bankovnictví zadávat platby, žádat o úvěry a mnoho dalšího. V tomto ohledu je internetové bankovnictví neoddělitelnou součástí naší moderní doby, ale málo kdo si uvědomuje jaké vývojové, logistické a testovací procesy se za tímto produktem stojí.

Cílem bakalářské práce je analyzovat celý proces ukrývající se za distribucí dat a jejich následné zpracování, zefektivnění v oblasti testování dat v aplikaci internetového bankovnictví.

Práce je rozdělena do pěti kapitol. V rámci první kapitoly je uvedena historie internetového bankovnictví společně s definováním služeb a správné zařazení internetového bankovnictví do nich. Další kapitola popisuje jednotlivé informační systémy používané v oblasti bankovních služeb, které jsou zde detailněji rozebrány společně s popisem komunikace, která probíhá mezi jednotlivými komponenty. Kapitola také obsahuje implementační schéma, je sestaveno za účelem zjednodušení bankovních procesů na projektu NYMBUS. Ve třetí kapitole je rozebráno zabezpečení přenosu dat pomocí protokolů, které mají za úkol chránit data proti případným hackerským útokům. Čtvrtá kapitola se věnuje popisu testovacích procedůr, rozdílu mezi manuálním a automatizovaným testováním společně s podrobným popisem jednotlivých postupů, které jsou použity na vytvoření automatizovaných testů a jejich následné porovnání s manuálními testy. Pátá kapitola je věnována identifikaci slabých míst, které se v průběhu tvorby automatizovaných testů vyskytli.

1 Logistické procesy v peněžních službách

Logistické procesy v peněžních službách je velmi široké téma, u kterého je nejdříve potřeba rozebrat samotnou službu a obecně si jí definovat společně s krátkou historií internetového bankovnictví a vývojem který mu předcházel. Poté je možné definovat jednotlivé logistické procesy, které se ukrývají uvnitř internetového bankovnictví a nejsou pro běžné každodenní uživatele či klienty viditelné. Všechny výše zmíněné pojmy jsou v rámci této práce podrobně popsány.

1.1 Historie internetového bankovnictví

Internetové bankovnictví si jako celek prošlo relativně dlouhým vývojem a i v dnešní době je stále potřeba přicházet s novými a inovativními přístupy jak uspokojit čím dál tím více náročnější klientelu. Samotné počátky internetového bankovnictví se datují k začátku 80. let, kdy čtyři největší banky v New Yorku přišli s produktem nazvaným Pronto, což byl produkt pro fyzické osoby (jednotlivce) a Pronto Business Banker, který byl určen pro malé až střední podniky zhruba v roce 1983. Za poplatek 12 dolarů měsíčně bylo zákazníkům umožněno využívat dial-up službu s možností spravovat elektronické záznamy, zjistit peněžní zůstatek na účtu atp. Tato služba spadá do kategorie "Telefonní bankovnictví". Tato kategorie je brána jako předchůdce internetového bankovnictví společně s platebními kartami, které zde byli ještě před touto službou.

Pokud se jedná o internetové bankovnictví jako určitý typ služby, kterou dnes používá přes 90 procent lidí po celém světě, tak její vznik se datuje v roce 1995, kdy na trh vstupuje Security First Network Bank, která je historicky první čistě internetová světová banka. Tento koncept si následně osvojila prakticky každá bankovní instituce a v dnešní době se těší obrovskému zájmu jak klientů, tak i u bank samotných. Jedním z hlavní důvodu takové obliby na straně bankovních institucí je snížení provozních nákladu. Pro klienta zase možnost připojit se k bankovnímu účtu prakticky z jakéhokoliv místa, pokud má k dispozici dostatečně kvalitní připojením k internetu, což je v této době bráno jako standard.

1.2 Služba

Definování služby nebo vysvětlení toho, co to vlastně služba je, se v dnešní době nejčastěji používá definice jednoho z předních amerických autorů Kotlera a Armstronga [1, s. 710] „*Služba je jakákoliv aktivita nebo výhoda, kterou může jedna strana nabídnout druhé, je v zásadě nehmotná a nepřináší vlastnictví. Její produkce může, ale nemusí být spojena s fyzickým výrobkem.“* Ovšem pokud se jedná o historii, tak v průběhu času vznikali nejrůznější názory či definice, které se týkaly především podstaty služeb a uspokojování samotných potřeb jednotlivce, tedy člověka. Prvního člověka, který definoval služby, je možné najít již v roce 1776, jedná se o ekonoma skotského původu A. Smith, jenž byl přesvědčen o tom, že služby jsou definovatelné jako statky, které neprodukují žádnou hodnotu. V důsledku jeho úvahy o dělení práce na produktivní a neproduktivní byla služba zařazena do kategorie neproduktivní, to ve výsledku znamená, že je spotřebována ve stejný okamžik, ve kterém je vytvořena, tedy nepřináší žádný kapitál, nezhmotňuje se v žádném předmětu a jedná se čistě o užitkovou hodnotu. Smithovy názory týkající se služeb byly takříkajíc přelomové a v podstatě inspirovali další jiné ekonomy z dané doby, například Jeana Baptisty Saye, Johna Stuarta, a dalších.

Obecně se dá pojem služba chápat jako typ hospodářské činnosti, která zajišťuje uspokojení určité potřeby, přičemž výsledkem je užitek, nikoli hmotný statek neboli výrobek.

1.2.1 Vlastnosti služeb

Existuje zde 5 specifických vlastností, které se dají označit za nejvýznamnější [2]:

- nehmotnost,
- neoddělitelnost,
- heterogenita,
- zničitelnost,
- nemožnost vlastnictví.

Nehmotnost je jednoznačně nejlépe vystihující vlastností služeb a odvíjejí se z ní další vlastnosti. Čistou službu není možné za žádných okolností vyhodnotit fyzickým vnímáním či smysly. Některé prvky, které definují kvalitu služby, jako například osobní přístup poskytovatele, jistota, spolehlivost apod., lze ověřit až při samotném nákupu a následné spotřebě. Díky tomuto vzniká určitá míra nejistoty zákazníků při poskytování služby bez ohledu, o jaký druh služby se jedná.

Neoddělitelnost je jednou z vlastností, kde se poskytovatel musí setkat v určitý čas na určitém místě se zákazníkem, aby došlo k naplnění služby či služeb. Služba je tedy vytvářena v reálném čase v přítomnosti zákazníka, proto se může zákazník označit za neoddělitelnou součástí produkce. Zákazník taková služba tudíž nezavazuje k tomu být přítomen po celý čas po kterou se daná služba poskytuje. Jako příklad z profesionálního prostředí služeb se dá použít běžná praxe z advokacie, kdy zákazník nemusí být nutně přítomen jednání a je zastoupen advokátem. Ve specifických případech existuje možnost nahradit poskytovatele v podobě fyzické osoby pomocí stroje, typickým příkladem je prodejní automat či bankomat.

Heterogenita je spíše známá pod pojmem variabilita či dynamičnost je specifická vlastnost. V procesu poskytování služby či služeb jsou přítomni potencionální zákazníci čili lidé, samotní zákazníci a poskytovatelé služby. Chování již zmíněných lidí nelze jednoduše předvídat, v případě zákazníku je vskutku těžké odhadnout určité normy chování. Navzdory všem těmto zmíněným aspektům jsou u mnoha druhů služeb normy stanoveny. Jako příklad se dá uvést chování zákazníků v restauraci nebo pravidla chování osob v hromadné dopravě a mnoho dalších. Heterogenita služeb a větší množství lidí při procesu poskytování služeb vede k tomu, že vstup na trh služeb je mnohem jednoduší a snadnější. To má za následek zvýšený výskyt konkurence, což je zapříčiněno díky nižší potřebě vstupního kapitálu společně s menší možnosti patentování.

Zničitelnost nehmotnost služeb má za následek, že službu obecně nelze uchovávat nebo skladovat či jiným způsobem prodávat. Příkladem je sedadlo v kině, místenka v letadle nebo obecně znalosti lidí v určitých profesích jako je lékařství, advokacie atp. V těchto případech pokud služby nejsou využity v určitém čase, ve kterém jsou nabízeny, stávají se pro daný okamžik zničenými. Takovou tu službu prakticky není možno vracet. Dalším příkladem může být špatně ostříhané vousy, které bohužel vrátit opravdu nelze.

Nemožnost vlastnictví je vlastnost, která velice úzce souvisí s předchozími vlastnostmi, kterými jsou nehmotnost a zničitelnost služby. Při samotném nákupu zboží přechází na zákazníka právo zakoupené zboží vlastnit. Naopak je tomu u poskytování služby, kde při koupi služby získává odběratel pouze právo na její poskytnutí nebo lépe řečeno využití, nejedná se tedy v tomto případě o právo vlastnit. Příkladem může být zakoupení přístupu do cloudové služby nebo využití prostředku veřejné dopravy. Tato vlastnost má za následek změnu z pohledu distribuční kanálu, který se využívá k čerpání služby zákazníkem. Tento kanál bývá relativně krátký, přímý a z pohledu samotné logistiky jednodušší oproti kanálu, kterým prochází fyzické zboží příkladem jsou sklady.

1.2.2 Členění služeb

Služby se obecně dají rozčlenit dle značného množství různých hledisek a hlavně do různých skupin. Česká republika patří do evropské unie, ve které se služby dělí na:

1. Služby liberalizované
2. Služby v obecném zájmu
    - Služby v obecném hospodářském zájmu:
    - Služby v obecném nehospodářském zájmu

Služby liberalizované neboli tržní služby jsou jako výkony poskytované za úplatu. Výše této úplaty se odvíjí od nabídky a poptávky, jelikož tyto služby nemají svého tzv. garanta, tudíž musí sám zákazník dohlížet na kvalitu poskytované služby. Příkladem takovéto služby je například pronájem vrtulníku, poradenství a mnoho dalších.

Služby v obecném zájmu sice jsou poskytovaný za účelem dosažení zisku, ale s tím rozdílem, že musí plnit určité úkoly. Tato služba již má svého garanta, který mimo jiné dohlíží na kvalitu poskytovaných služeb a určuje regulovanou cenu. Tyto služby se dále dělí na služby v obecném hospodářském a nehospodářském zájmu. Služby v obecném hospodářském zájmu jsou poskytovány za úplatu, ale musí z nich plynout veřejný užitek. Typickým příkladem jsou informační, dopravní či poštovní služby. Služby v obecném nehospodářském zájmu také plní určitý veřejný zájem s tím rozdílem, že jsou poskytovány bezplatně, příkladem je kultura nebo sociální služby, které poskytuje stát. Oblast internetového bankovnictví spadá do oblasti liberalizované služby.

2 Informační systémy v peněžnictví

V rámci této kapitoly je důležité si nejdříve definovat, co jsou to informační technologie, jedna z definic z ní následovně: „*Informační technologie (dále jen IT) chápeme jako množinu prostředků a metod sloužících k práci s daty a informacemi. Podle této definice je tedy pojem IT značně široký. Zahrnuje nejen techniky a technologie pořizování a zpracování dat, ale také prostředky jejich přenosu, ukládání, využívání a následného vyhodnocování. Pronikání informačních technologií do veškerých činností společnosti znamená vývoj do stavu, který řada autorů nazývá existencí informační společnosti. U informačních technologií rozeznáváme složky technickou, programovou (implementační) a informační.”*.[3, s. 15]

Z této definice se dá odvodit, že všechny zmíněné složky jsou v peněžních či lépe řečeno bankovních systémech ve větší či menší míře používány.

Pokud se jedná o jasné definování, co je bankovní neboli peněžní informační systém, tak se nedá jednoduše a snadno odpovědět. Klasická banka prakticky nemá jeden ucelený a jednotný informační systém jako řešení pro složitou bankovní problematiku. Bankovní informační systém se dá specifikovat jako několik desítek až stovek různých navzájem spojených systému nebo aplikací, kde hlavním komponentem bankovního systému je primární uložiště dat neboli databáze, která obvykle představuje tzv. core system(jádro) banky. Do tohoto jádrového systému proudí veškeré informace, data z okolních systémů, která jsou na něj přímým způsobem napojena. V samotném okolí databáze se obvykle nachází systémy, které jasně definovaným způsobem podporují jednotlivé pracovní procesy. Například procesy, které jsou prováděny na přepážkách se označují jako front-office systémy. Dále existují aplikace pro zpracování důležitých informací tzv. back-office systémy společně s dalšími podpůrnými systémy, které se starají o technickou problematiku a zázemí.

2.1 Informační systém banky

Rozvoj bankovního sektoru a souvisejících služeb společně se samotným řízením činností banky v tržních podmínkách vyžadují rychlé a naprosto přesné informace, které jsou dosažitelné pouze výkonovou technickou soustavou společně s vysokou úrovní programových prostředků a kvalitním informačním systémem samotné banky.

2.1.1 Význam informačního systému banky

V dnešní moderní době se žádná banka nemůže obejit bez kvalitního informačního systému zaměřeného na bankovní problematiku. Bez tohoto systému nelze zajistit správný chod banky, provádění bankovních operací, zavádění nejnovějších funkcionalit nebo produktů do elektronického bankovnictví. Nemluvě o dalších aspektech jako získávání informací pro činnost a řízení banky. Kvalita takového systému společně s jeho bezpečností jsou brány jako jeden z nejdůležitějších předpokladů konkurenceschopnosti banky.

Informační systém by měl splňovat následující základní cíle:

- Plně či co nejvíce automatizovat všechny základní operace banky,
- Vyloučit či co nejvíce omezit manuální zpracování,
- Zefektivnit datový tok údajů v bance společně se zefektivněním bankovního segmentu.

Informační systém (IS) plní v bance následující úkoly[4]**:**

- Automatizované zpracování bankovních činností
- Poskytuje integrovaný přístup ke všem komponentám, které jsou v bankovní hierarchii umístěny, dále dodává potřebné informace pro řízení a kontrolu v reálném času napříč všemi operacemi a na různých úrovních,
- Umožňuje vedení účetní evidence banky, poskytuje důležité informace majitelům banky, investorům, klientům a odborné veřejnosti,
- Tvoří databázi požadované centrální bankou v rámci daňové kontroly pro ministerstvo financí společně s požadovanými statistikami,
- Poskytuje veškeré informace pro vyšší management a zároveň je možné v něm budovat moderní IS manažerského typu,

- Zajišťuje obecnou komunikaci a propojení v oblasti bezhotovostního platebního styku, propojení se systémem SWIFT v rámci zahraničních bezhotovostních plateb,
- Zpracovává údaje v reálném čase,
- Poskytuje propojení centrály banky s dalšími pobočkami a vytváří specializovanou informační základnu, která je ve své podstatě obrovskou centrální databází.

Je nezbytně nutné, aby banka byla pomocí vlastního vnitřního informačního systému napojena na některé vnější informační systémy, příkladem je informační systém národní banky, informační systémy jiných bank a obchodníků.

V informačním systému banky lze rozlišit následující druhy informací [4, s. 3]:

- *externí – informace získané proto, aby externího uživatele /např. měsíční hlášení centrální bance nebo které pochází z externího zdroje,*
- *interní – informace, které se získávají uvnitř organizace pro interní uživatele např. interní výkaz zisk a ztrát,*
- *formální – ty, jejichž forma nebo způsob předání jsou bankou podepsány /např. formuláře, přehledy, ve firmě jednoznačně převažují,*
- *neformální – ty, které se nezískávají běžnými formálními cestami,*
- *faktické – informace o faktickém stavu budou v současném okamžiku nebo v některém konkrétním okamžiku v minulosti,*
- *budoucí – informace o stavu, který nastane v budoucnosti,*
- *příležitostní – ty, které se požadují na základ určité události,*
- *informace pro daný projekt - ty, které se poskytují pro potřeb určitého konkrétního projektu,*
- *informace o činnosti – mají sloužit k získání přehledu o všech pravidelně se opakujících procesech v bance i v jejím běžném provozu,*
- *informace o prostředí – pomáhají chápat události a trendy vně banky, mají strategický význam,*
- *kvantitativní – lze je číselně vyjádřit, je to převážná část informací,*

- *kvalitativní – nedají se vůbec nebo jen velmi těžko vyjádřit číselně, přenášejí se slovy, grafy, apod.*

2.2 Dodavatelský pohled na vývoj informačních systémů

Každý klient je svým způsobem specifický, má jiné požadavky a jiné představy o produktu, který chce. Proto je velmi těžké v dnešní době vytvářet produkty sériově. Informační systémy, které jsou vyráběny sériově, kolikrát způsobí více obtíží na straně klienta.

Informační systémy, které jsou vyvíjeny specifiky pro určitého klienta reprezentuje jedinečné řešení odrážející know-how samotného podniku. U takto zadané zakázky na vývoj specifického softwaru je 99% úspěšnost toho, že bude v cílovém prostředí fungovat, jelikož je vytvářen specificky podle jasně zadané analýzy potřeb zákazníka v tomto případě banky. Pokud by se jednalo o sériově vyráběnou softwarovou aplikaci, tak pravděpodobnost fungování v cílovém prostředí je značně nízká, protože se jedná o produkt, na který je implementována obecná problematika nikoliv problematika specifická.

Důležitým aspektem tohoto typu vývoje je možnost implementace dalších funkcionalit nebo technologií, které se dají v pozdější fázi jednoduše přidat. Což je obrovská výhoda zakázkového softwaru, který je pro tento další možný rozvoj přístupný. Tato vlastnost přináší nezanedbatelnou úsporu finančních prostředků, které by v případě jiného vývoje museli být investovány do úplně nových řešení. Pro bankovní sektor se software většinou vyvíjí specificky, ať se jedná o aplikace nižšího typu, kde tyto aplikace slouží spíše k podpůrné roli nebo o aplikace vyššího typu, kdy se bavíme o rozsáhlém zásahu do logiky samotných informačních systémů určených k řízení klíčových procesů.

2.3 Způsob vývoje softwaru (Agilní metodiky)

Jedná se o metody původně určené pouze pro vyvíjení softwaru založené na iterativním a inkrementálním vývoji. Tyto metodiky umožňují relativně rychlý vývoj softwaru s možností pružně reagovat na změnu požadavků v průběhu vyvíjení. Ověření správnosti dochází jedině pomocí rychlého předložení produktu zákazníkovi a následné

úpravy podle zpětné vazby. Agilní přístup se nejvíce používá v IT či marketingových firmách. Opakem je tzv. vodopádový model.

2.3.1 Agilní metody (agile methods)

Jedná se o interaktivní způsob řízení projektu. Základem je průběžné dodávání produktu nebo služby zákazníkovi na bázi aktivní spolupráce obou stran v tomto případě vývojářského týmu (dodavatele) a zákazníka (bankovní společnost). Tato spolupráce pomáhá zajistit rychlé reagování na jakoukoliv změnu z jedné či druhé strany, dokonce i z okolního prostředí. Mezi nejpopulárnější metody patří:

- SCRUM
- Kanban
- Extrémní programování (XP)

První zmíněná metoda je nejpopulárnější agilní metodika a někdy je používána jako synonymní označení pro celou skupinku výše zmíněných metod.

2.3.2 Priority agilního vývoje, programování

Tyto priority byly formulovány již v roce 2001 skupinkou vývojářů, kteří sepsali dokument s názvem Manifest agilního programování (Manifesto for Agile Software Development), kde definují následující principy [5, s. 1] :

„*Jednotlivci a interakce před procesy a nástroji*

Fungující software před vyčerpávající dokumentací

Spolupráce se zákazníkem před vyjednáváním o smlouvě

Reagování na změny před dodržováním plánu“

2.3.3 Principy agilních metod

Jakožto každá metoda i ta agilní metoda své určité principy, podle kterých by se měl tento vývoj řídit:

- Hlavní a nejpodstatnější prioritou je uspokojit zákazníka rychlým a průběžným dodáváním kvalitního softwaru.

- Změny požadavků v průběhu vývoje jsou do určité míry vítány. Agilními procesy lze rychle zpracovat tyto požadavky tak, aby přinesly konkurenční výhodu.
- Dodávání fungujícího softwaru často v domluvených intervalech v řádů týdnů až měsíců. Kratší intervaly jsou upřednostňovány.
- Všechny složky týmů musejí spolu spolupracovat na každodenní bázi během celého projektu.
- Sdílení informací pomocí osobních setkání.
- Měřítkem postupu vývoje je fungující software.
- Agilní procesy podporuji stabilní výkonnost. Uživatelé, vývojáři by měli být dosahovat těchto výkonů, dokud je třeba.
- V jednoduchosti je síla, umění co nejvíce práce vůbec nedělat.
- Nejlepší architektury vznikají v týmech, ve kterých se dokážou lidé organizovat samostatně.
- V pravidelných intervalech se vyhodnocuje práce celého týmu a upravují se postupy takovým způsobem, aby se dosahovalo co nejvyšší efektivnosti.

2.3.4 Základní struktura vývoje podle agilní metody

- Před zahájením vývoje se dělá tzv. product backlog, což je kompletní seznam všech úkolů, které mají být zhotoveny na konci projektu. Každý úkol má svojí prioritu, která je mu určená na samotném začátku projektu a podle této priority se přistupuje k jednotlivým úkolům (čím je priorita vyšší, tím spíš bude daný úkol dříve hotov). Existuje zde i metoda, která ukazuje, že úkol je hotov tzv. definitive of done (DOD).
- Pro lepší spolupráci a pochopení mezi jednotlivými členy týmu a zákazníkem v tomto případě bankou jsou vytvářeny user stories neboli use cases. Jedná se o popis správné funkčnosti v jednoduchých bodech společně s předem určenou rolí (např. admin) ve specifikovaném prostředí.
- Základem je určení délky sprintu (iterace), jedná se o opakující jednotku s časovým omezením. Udává se v rozmezí od dvou až tří týdnů, má tedy fixní

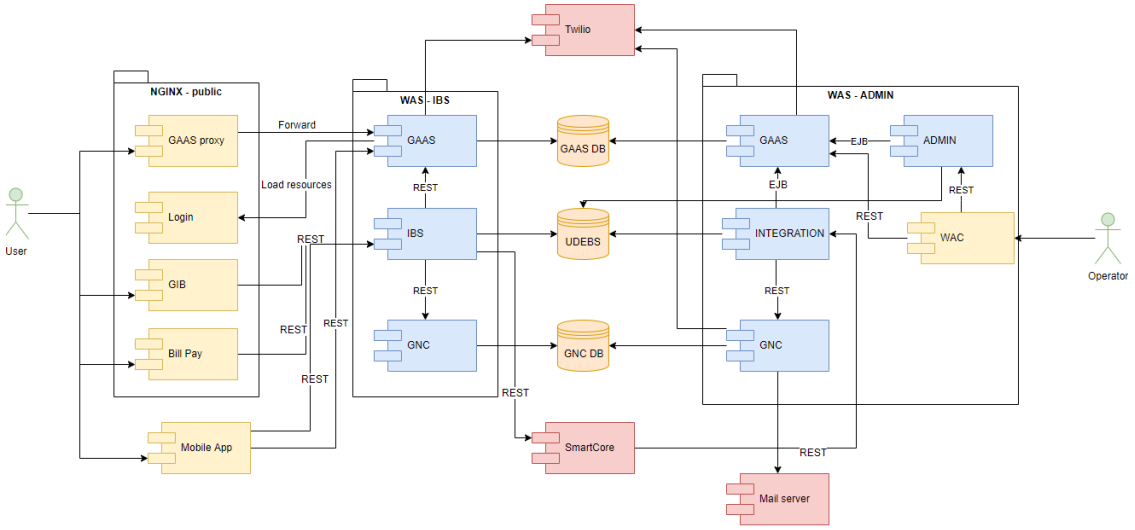
délku. Vývojový tým v tomto průběhu pracuje se zadaným sprint backlogem, což je počet úkolů, které by měly být v následujícím sprintu splněny. Tento seznam je sestaven na začátku na tzv. plánovacím meetingu, kde se odhaduje délka a náročnost práce.

- V průběhu sprintu se tým setkává každý den ve stejnou dobu, na stejném místě, a pokud možno ve stoje. Každý má za úkol zhruba ve dvou minutách, co udělal, na čem plánuje dělat dnes popřípadě, co se nepovedlo nebo co ho blokuje v jeho momentální situaci. Tento meeting se označuje jako daily stand-up a jeho maximální délka by měla být okolo 15 minut.
- Po skončení jednoho sprintu následuje tzv. sprint review, kde tým předvádí zákazníkovi novou funkcionalitu nebo produkt.
- Následuje restrospective meeting, který se koná po každém vykonaném sprintu a rozebírá se zde, co se podařilo nebo nepodařilo, a co je potřeba změnit do příštího sprintu.

2.4 Implementace internetového bankovnictví

Velký produkt jako je internetové bankovnictví je potřeba důkladně a precizně naplánovat a zaznamenat do vývojového diagramu. Tento vývojový diagram následně slouží pro všechny členy týmu jako prototyp správné komunikace mezi jednotlivými komponenty, která banka může, ale nemusí použít. Tato skutečnost je dána tím, že banka nemá jednotný informační systém, jak již bylo zmíněno v minulé kapitole. Některé komponenty, které bývají využity při vývoji ze strany dodavatele, má banka tendenci nahrazovat podobnou komponentou z vlastní informační struktury. Pokud takovou komponentu nemá, tak použije dodávanou.

Obr. 2.1 Kompletní implementační schéma



Zdroj: interní dokumentace [6]

Jednotlivé komponenty budou rozebrány postupně, ale prvně je potřeba definovat základní komunikaci, která je použita mezi komponenty uvedených na obrázku č. 2.1.

API (Application Programming Interface)

Techniky řečeno API označuje aplikační programové rozhraní. V dnešní době má většina velkých společností své vlastní API buď pro vnitřní použití ve firmě, nebo pro své zákazníky.

Definice podle společnosti Apple:*„API neboli aplikační programové rozhraní, slouží k předávání dat mezi softwarovými aplikacemi formalizovaným způsobem. Mnohé služby nabízejí veřejná API, díky nimž může kdokoli do služby předávat obsah nebo ho z ní odebírat. API, která pracují přes internet prostřednictvím adres URL ve tvaru http://, se označují jako webová rozhraní API. Uživatelé na webu posílají rozhraní API své požadavky týkající se načtení nebo zveřejnění informací.“* [7, s. 1]

Když vezmeme web jako celek, tak se bavíme o velké serverové síti, která je navzájem propojená. Každá stránka na internetu je uložena někde na vzdáleném serveru. Tento vzdálený server je prakticky jen část vzdáleného počítače, který je optimalizován pro zpracování požadavků. Když uživatel do svého prohlížeče nyní zadá například stránku www.google.com, požadavek přejde na vzdálený server firmy Google. Jakmile prohlížeč uživatele obdrží odpověď, kód interpretuje a zobrazí stránku. To znamená, že pokaždé když uživatel navštíví jakoukoliv stránku na webu, tak komunikuje v podstatě s API vzdáleného serveru. Ovšem rozhraní API není stejné jako vzdálený server,

přesnější popis je, že se jedná o část serveru, který se stará o přijímání požadavků a odesílá zpět odpovědi.

Využití v praxi

Existují společnosti, které se zabývají tzv. balením API jako produktu. Příkladem je společnost Weather Underground, která prodává přístup k API pro data o počasí. Dalším scénářem může být například zákazník, který má webové stránky na které je formulář, který slouží k podepisování klientů ke schůzkám. Tento zákazník nebo lépe řečeno majitel chce dát svým klientům možnost automaticky vytvořit událost v Google kalendáři s podrobnými informacemi. V ten moment přichází na řadu použití API rozhraní. Cílem je, aby server této webové stránky mluvil přímo se serverem společnosti Google s požadavkem na vytvoření události s danými informacemi. Server majitele poté obdrží odpověď od serveru společnosti Google, zpracuje ji a odešle zpět relevantní informace pro prohlížeče, například zprávu o potvrzení vytvoření takové události. Hlavní rozdíl tkví v tom, že pokud by uživatel chtěl vykreslit celou webovou stránku, tak prohlížeč by očekával odpověď v jazyce HTML, zatímco volání rozhraní API kalendáře Google pouze vrátí data, která jsou většinou ve formátu JSON. Důležité je si uvědomit, že pokud server, na kterém běží webové stránky posílá požadavek na rozhraní API, tak je na něj pohlíženo jako na klienta (stejný princip jako je tomu u navigace skrz webové stránky). Z pohledu uživatelů jim API umožňuje dokončit akci, aniž by opustily stránky, na kterých se zrovna nacházejí. Většina moderních webových stránek využívá do určité míry API třetích stran. To je dáno hlavně z důvodu, že většina třetích stran má toto řešeno ve formě knihoven nebo služeb, které jsou brány jako jednodušší a spolehlivější řešení.

REST (Representational State Transfer)

Jedná se o styl softwarové architektury, který definuje sadu pravidel, která mají být použita pro vytváření webových služeb. Webové služby, které architektově odpovídají tomuto stylu zvaného REST se obecně označují jako RESTful Web services (dále jen RWS). Webové služby RWS umožňují požadujícím systémům přístup a následnou manipulaci s textovými reprezentacemi webových zdrojů pomocí jednotné a předdefinované sady operací.

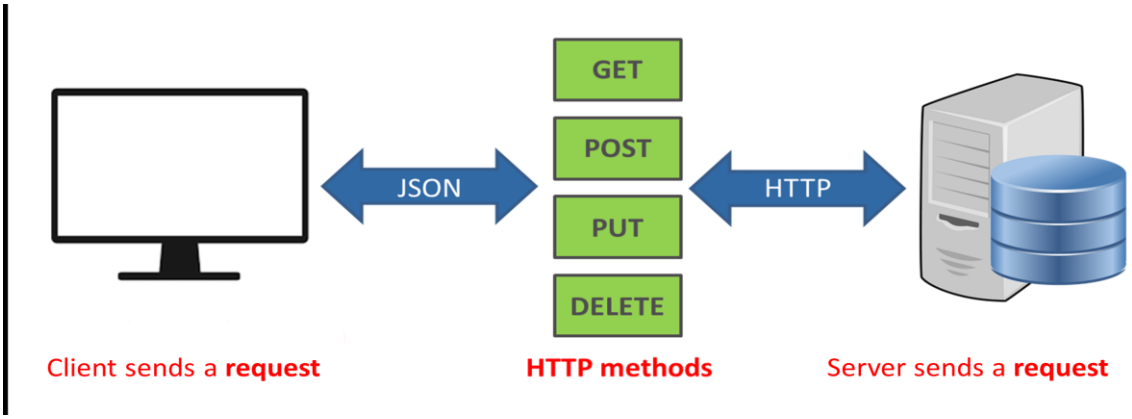
Tyto tzv. webové zdroje byly poprvé definovány na internetu jako dokumenty nebo soubory identifikované jejich adresami URL. Dnes však mají mnohem obecnější a

abstraktnější definici, která zahrnuje každou věc nebo entitu, která může být identifikována, pojmenována, oslovena nebo zpracována jakýmkoliv způsobem na webu. Ve RWS službě jsou požadavky, které jsou odeslané do URL identifikátoru zdroje zpracovány a z něho jsou vyvolány příslušné odpovědi. Obsah ukrytý v těchto odpovědích je obvykle formátován v HTML, XML nebo JSON souborech. Tato odpověď může potvrdit, že byla provedena nějaká změna. Pokud se k tomuto používá HTTP protokol, jak tomu bývá nejčastěji zvykem, tak jsou k dispozici níže zmíněné operace nebo lépe řečeno metody:

- GET, HEAD, POST, PUT, PATCH
- DELETE, CONNECT, OPTIONS, TRACE

Využívání HTTP protokolu, který nevyžaduje, aby server uchovával informace o relaci nebo stavu mezi komunikačními partnery po dobu trvání vícero požadavků má za následek to, že RWS systémy jsou výkonnostně velmi rychlé, spolehlivé a mají potenciál stále se rozšiřovat do budoucna. To je především zapříčiněno opětovným použitím komponent, které mohou být spravovány a aktualizovány bez ovlivnění systému jako celku, i když je spuštěn.

Obr. 2.2 REST API v praxi



Zdroj: [8]

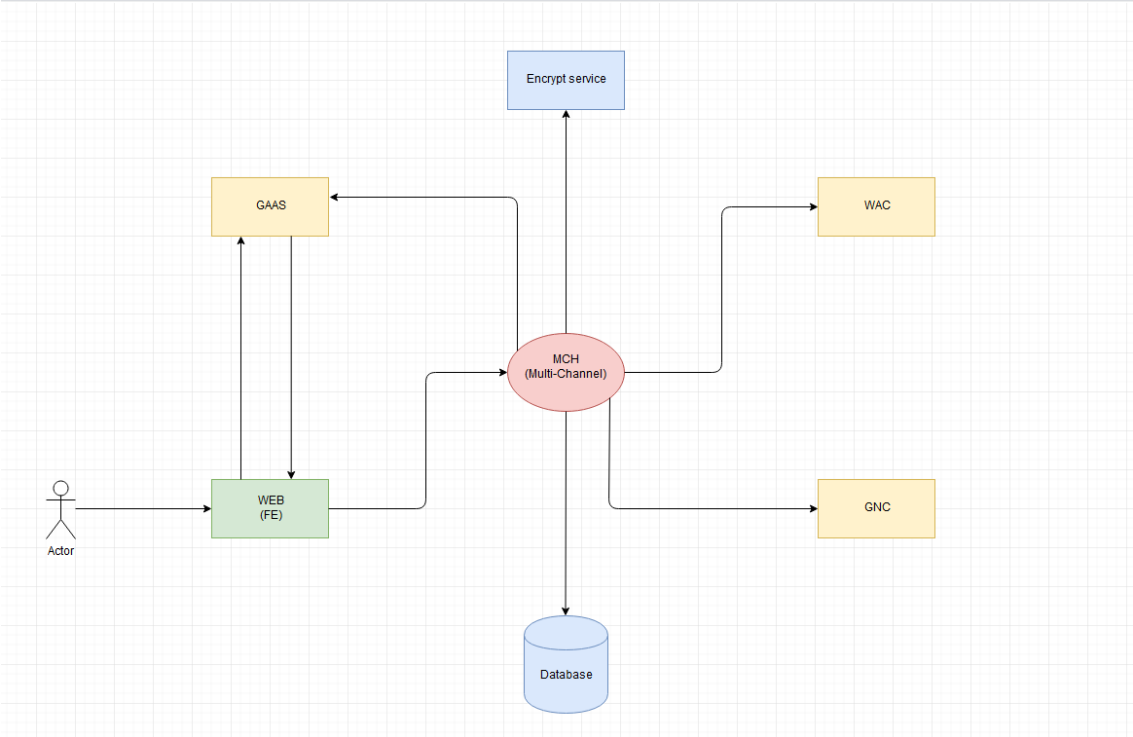
Využití v oblasti internetového bankovnictví

Není neobvyklé, že vývojové týmy rozdělují své aplikace na více serverů nebo komponent, které spolu navzájem komunikují prostřednictvím rozhraní API. Servery nebo komponenty, které vykonávají pomocné funkce pro hlavní aplikační server, jsou běžně označovány jako mikroservisy.

Specifikace komponent a jejich význam

Již několikrát bylo v textu zmíněno jádro bankovního systému takzvaný core nebo v některých případech smart core. Tento software je využíván pracovníky na pobočkách či call centrech pro vykonávání jednotlivých činností banky. Jednotlivé komponenty, které jsou napojeny na jádro banky, navzájem komunikují přes rozhraní REST API.

Obr. 2.3 Zjednodušené implementační schéma



Zdroj: vlastní zpracování, 2019

Multichannel (MCH) – Vícekanálová komponenta pro budování systémů internetového bankovnictví. Poskytuje prostor pro vytváření podnikových funkcí a sadu kompatibilních koncových bodů REST API. MCH implementuje následující funkce:

- Poskytování dat (klient, uživatel, účet)
- Správa oprávnění (správa pravého profilu, správa podpisových pravidel)
- Správa systému (správa katalogů, správa parametrů, prohlížeč časopisů)
- Správa operátorů MCH
- Správa objednávek oznámení
- Manuální správa požadavků uživatelů

- Zprávy - zasílání zpráv bankovním klientům, uživatelům. Příjem zpráv od nich

Z pohledu dodavatelské firmy je potřeba nasimulovat prostředí banky co možná do nejmenších detailů, proto se používá tzv. multichannel dále už jen MCH. Komponenta MCH simuluje reálné chování jádrového systému, který banka používá. Stejně jako je u tomu core systému banky, tak MCH komponenta se považuje za srdce neboli jádro, které komunikuje se všemi ostatními komponenty. Jeho primární cíl je zabezpečování správného toku dat mezi všemi komponenty.

GNC (GEMINI Notification center) – Komponenta vyvinuta speciálně pro shromaždování příchozích událostí z externích systémů s následným generováním oznámení uživateli. Všechny tyto oznámení jsou poté distribuována podle kanálu (obvykle SMS nebo e-mail), což si uživatel sám volí v internetovém bankovnictví. Příkladem je uživatel, který chce obdržet oznámení, když zůstatek na jeho účtu klesne pod určitou nastavenou hranici. GNC pracuje na bázi šablonovitého systému, kdy se vytvoří jedna šablona pro každý typ operace (oznámení o zůstatku, blokace peněz na kartě atp.).

GAAS (GEMINI Authentication and Authorization Server) - je centralizovaný autentizační server, který řídí přístup uživatelů k jiným (cílovým) systémům. Jeho hlavním účelem je identifikovat a autentizovat uživatele a udržovat jeho relaci ve všech systémech. Banky obvykle provozují řadu převážně nezávislých systémů či komponent a každý takový systém musí samozřejmě určitým způsobem určit a ověřit uživatele (získat jeho identitu a ověřit si ji), který má pracovat se systémem. V běžných řešeních řeší každý systém tento problém nezávisle, což přináší následující problémy:

- Obtížný nebo téměř nemožný přesun uživatele z jednoho systému do druhého.
- Související problémy s přenosem dat mezi těmito systémy (např. Inicializací platebního příkazu v internetovém bankovnictví z jiného systému).
- Potřeba několikrát řešit možné modifikace přihlašovacích mechanismů (pro každý systém zvlášť).
- Mnohem vyšší riziko chyb zabezpečení (potřeba otestovat několik nezávislých implementací).
- Je obtížné centralizovat bezpečnostní data.
- Obtížná centralizovaná správa (např. Blokování uživatele ve všech systémech).

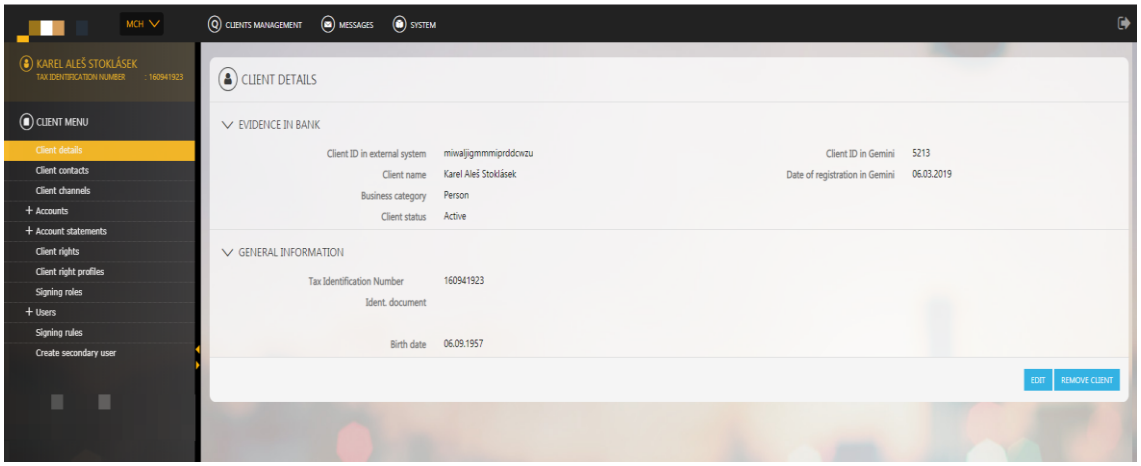
- Obtížné centrální monitorování (které uživatelé jsou aktuálně přihlášeni do kterého systému).

Výhody, které přináší GAAS, jsou odpovědi na výše uvedené problémy:

- Všechna bezpečnostní zařízení jsou umístěna v databázi GAAS (nebo systémech, které jsou připojeny pouze k GAAS), takže mohou být lépe chráněny.
- Existuje pouze jedna implementace ověřovacího procesu, kterou je třeba ověřit.
- Možné úpravy autentizačního procesu jsou implementovány pouze v GAAS.
- Po přihlášení uživatele do jednoho systému (přes GAAS) může uživatel volně přecházet do jiných připojených systémů.
- Centralizovaná správa uživatelů a bezpečnostních zařízení.
- Centralizovaná správa a monitoring (kdo je přihlášen, kde se odhlašují uživatelé ze všech systémů, přiřazování přístupu k jednotlivým systémům na základě pravidel).
- Jednorázové odhlášení (při odhlášení uživatele je odhlášen ze všech systémů připojených k systému GAAS).

WAC (Web admin console) je komponenta vyvinuta za účelem editování informací o uživatelích, nastavování práv a plní roli jakéhosi admina přes kterého se dá přistupovat k uživatelům uložených v databázi. Tato komponenta je využívána pouze pro kanál internetového bankovnictví z pohledu vývojáře, protože větší banky mývají tuto funkci zastoupenou ve svém jádrovém systému banky tedy v core systému. Dalším faktem zůstává, že malé banky nepotřebují do určité míry tolik zasahovat do práv uživatelů, tudíž není potřeba mít další komponentu. Přes WAC se mohou například nastavovat globální texty, které jsou nastavovány přes položku GUI (Global User Interface). Jedná se o texty, které uživatel vidí skrz celé internetové bankovnictví. Další položkou je možnost posílání zpráv, jedná se o oboustrannou komunikaci, kde uživatel je schopen poslat zprávu bance.

Obr. 2.4 Web-admin console (WAC)



Zdroj: vlastní foto, 2019

Jak je možné vidět na obrázku č. 2.4 je zde hlavní nabídka, která obsahuje následující položky: Clients Management, Messages, SYSTEM.

Clients management – slouží pro vyhledání uživatele či klienta, který již existuje v databázovém systému banky.

Messages – slouží pro posílání zpráv uživateli, který má internetové bankovnictví.

SYSTÉM – tato záložka obsahuje možnost nastavení rolí, katalogy funkcí, systémové parametry a již zmíněné GUI texty.

Kromě již zmíněných položek je zde ještě přepínač sloužící k přepínání komunikace. Výchozí komunikace je nastavena na MCH (Multi-channel) komponentu, další možností je GAAS.

Databáze – slouží k uchování a uložení veškerých dat o uživatelích. Do databáze se přistupuje pomocí MCH komponenty, která s těmito daty sama manipuluje nebo distribuuje k dalším komponentám. Jedná se ve většině případů o relační databáze poskytovány společností Oracle, které jsou nabízeny v podobě cloudové služby.

Dohromady všechny tyto komponenty zajišťují tzv. BE (Back-End) komunikaci.

2.5 Nástroje pro řízení, sledování projektu (JIRA, Confluence)

Na tuto problematiku se dá pohlížet ze dvou perspektiv. Jednak ze strany dodavatele internetového bankovnictví, který dodává jasně definovaný produkt společně s technologickým know-how, nebo z pohledu zákazníka v tomto případě banky. Z hlediska dodavatele se jedná o určení finanční a časové náročnosti, proto se využívá softwarový nástroj JIRA, kde se zadávají data v podobě elementárních časových jednotek zaměstnanců, kteří zrovna pracují na daném projektu. Díky tomuto postupu je možnost určit finální nákladovost projektu.

2.5.1 Software JIRA

JIRA je nástroj vyvinutý australskou společností Atlassian. Používá se pro sledování chyb, sledování problémů a řízení projektů. Jméno "JIRA" je v podstatě odvozeno od japonského slova "Gojira" což znamená "Godzilla".

Základní použití tohoto nástroje tkví ve sledování problémů, chyb související se softwarem ať se jedná o desktop nebo mobilní aplikaci. Používá se také pro řízení projektů.

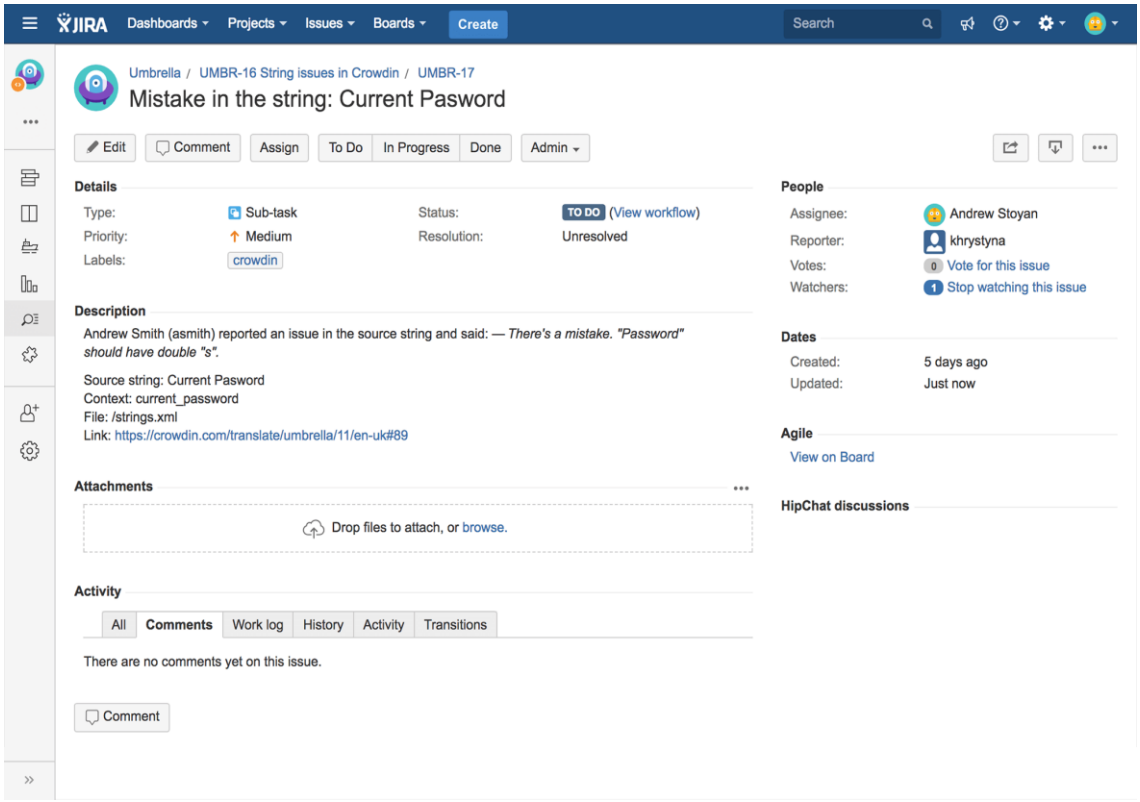
JIRA a její distribuce se dá rozdělit do tří kategorií:

- **JIRA CORE**
- **JIRA Software**
- **JIRA Services Desk**

JIRA Core je určen pro obecný projektový management. JIRA software nabízí základní software, který má v základu možnosti nastavení agilně projektového řízení. Poslední je JIRA Service Desk, který je určen pro používání IT pracovníky či obchodního oddělení.

Software je koncipovaný jako komerční produkt, který může být licencován jako cloud do kterého si kupuje firma přístup nebo jako vlastní serverová aplikace, tuto aplikaci si následně firma sama spravuje. Níže podle uvedeného obrázku budou rozebrány jednotlivé funkcionality, se kterými se uživatel může setkat při používání na projektu.

Obr. 2.5 Hlavní nabídka v program JIRA

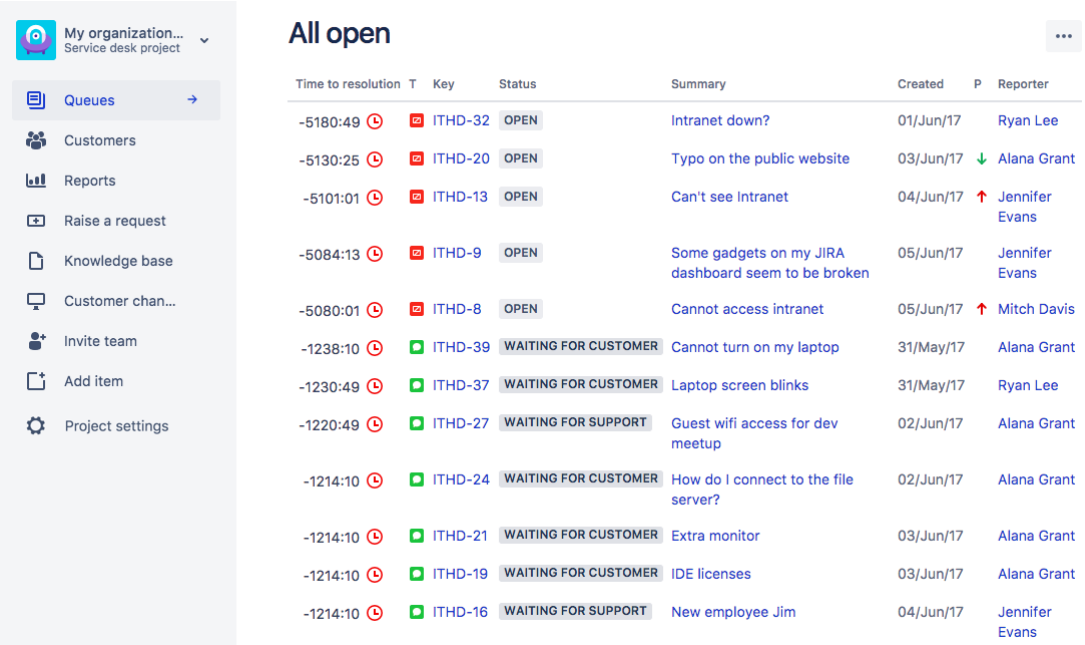


Zdroj: [9]

Dashboard – Tzv. deska se skládá z funkcionalit, které si uživatel sám zvolí buď z předem zadaných filtrů, které dodává sama firma Atlassian nebo si seskládá jednotlivé funkcionality podle svých subjektivních preferencí. Například funkce typu historie činností, sledované úkoly, rozvrh pracovních hodin nebo zadané odpracované hodiny jsou jen zlomek z možných funkcionalit, které si uživatel může zvolit.

Projects - Zde se nachází informace o probíhajících projektech. Uživatel může přepínat mezi vícero projekty naráz (pokud má přiřazeny práva). Po vybrání jednoho projektu je uživatel přesměrován do detailního náhledu, kde může vidět jednotlivé reporty, vydané verze nebo aktuální stav daného projektu.

Obr. 2.6 Detailní stránka projektu



Zdroj: [10]

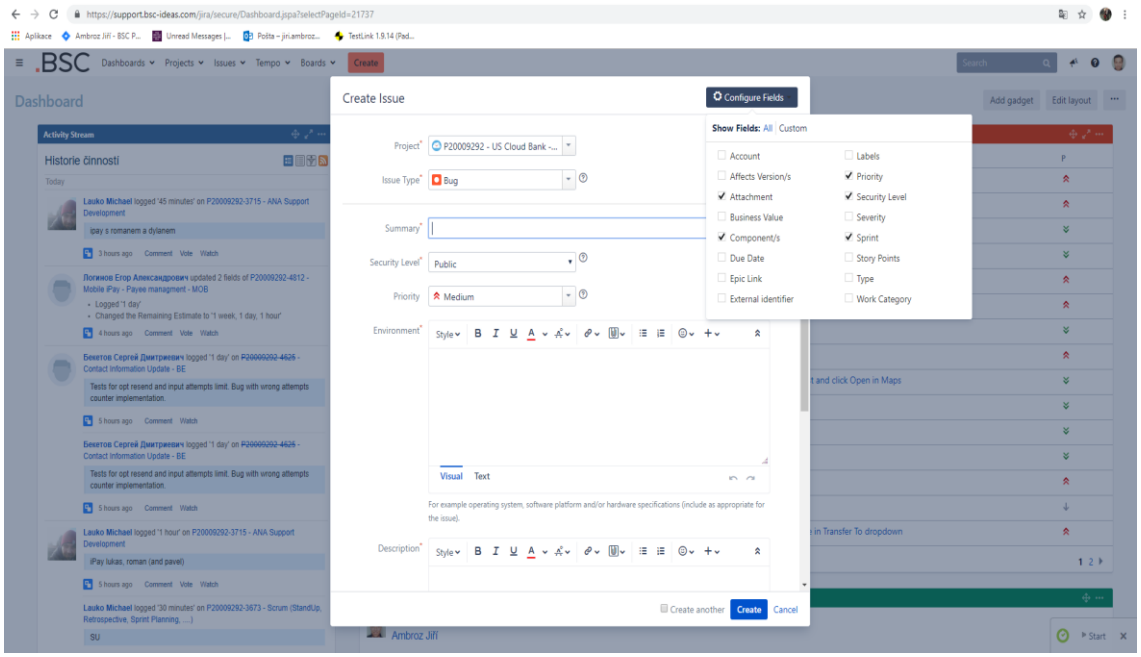
Issues - V záložce issues neboli v českém překladu problémy je možnost zobrazit aktuálně zadané úkoly, chyby. Uživatel má k dispozici velké množství přednastavených filtrů např. otevřené úlohy, úlohy zadané libovolnou osobou, všechny úlohy, kritické úlohy atp. Je zde kladen velký důraz na customizaci, což je jedna v mnoha přednosti softwaru JIRA.

Tempo - Záložka pro sledování pracování výkazu nebo vytváření exportních souborů s jednotlivými detaily o stráveném čase na úkolech. Dále se zde nachází přehled s informacemi o jednotlivých členech týmů společně s jejich zařazením a pozicemi na projektu.

Boards - Záložka je určena pro sledování aktivit týmu. Všichni členové zde mají zobrazený svůj aktuální úkol, na kterém pracují, společně s informací o jaký úkol se jedná. Důležitá je také informace o verzi produktu, na kterém se zrovna pracuje, ta se nachází v levém horním rohu společně s filtrovací lištou.

Create - Prakticky nejvíce využívanou funkcionalitou pro testera či projektového manažera je právě záložka create. V ní je možno si v pravém horním rohu nakonfigurovat požadované pole, které pomáhá s popsáním nalezeného problému nebo vytvoření úlohy pro vývojáře. Jak lze vidět z přiloženého obrázku níže.

Obr. 2.7 Zadávací formulář s konfigurovatelnými poli



Zdroj: vlastní foto, 2019

Nacházejí se zde pole od účtu po pracovní třídu. Opět záleží na daném projektu, kolik je potřeba využít popisných polí, protože někdy méně znamená více. Může zde lehce dojít k obrovskému přehlcení informacemi, které nejsou tak nutné pro přesné definování problému. Většina projektu si dokáže vystačit zhruba s 9 následujícími poli:

Attachment/Příloha - Slouží k nahrání obrázku, videí či jiných věcí, které jsou zapotřebí k rychlému pochopení a lepšímu porozumění daného problému.

Component/Komponenta - Určení komponenty slouží především k jednoduché distribuci problému k vývojářům, kteří jsou schopni podle předem nastaveného filtru rychle nalézt daný problém a ihned se zorientovat.

Priority/Priorita - Priorita slouží k určení přednosti. Priorita se dělí do 7 typů (Blocker,Blocker(Live) Critical, High, Medium, Normal, Low, Trivial). Speciálně stojí za zmínku právě oba typy s názvem blocker. Obvykle se jedná o problém, který je potřeba vyřešit v rámci hodin. Takovýto problém se obvykle vyskytne po velkém systémovém upgradu nebo vydání nové verze do bankovního prostředí.

Security Level/ Nastavení zabezpečenosti - Nastavení viditelnosti jednotlivých úkolu je důležitá součást agilního vývoje. Jsou zde v nabídce dvě možnosti a to public (veřejný) nebo private (soukromí).

Environment/Prostředí - Ve firmách se lze setkat dokonce s několika prostředími naráz. Vývojáři používají své vlastní lokální nebo společné prostředí, do kterého mají přístup pouze lidé s právem vývojáře. Další mají testeři, kteří zde testují poslední vydanou verzi daného produktu.

Description/Popis - V popisu se nacházejí detailní informace podle jakých má vývojář postupovat při reprodukování chyby, kterou nalezl tester. Další možností je přesný popis úkolu, který má vývojář splnit. Pokud se jedná o úkol, který nahlásil tester, tak by měl popis obsahovat jednotlivé kroky potřebné pro reprodukování chyby společně s použitými daty, které byly použity.

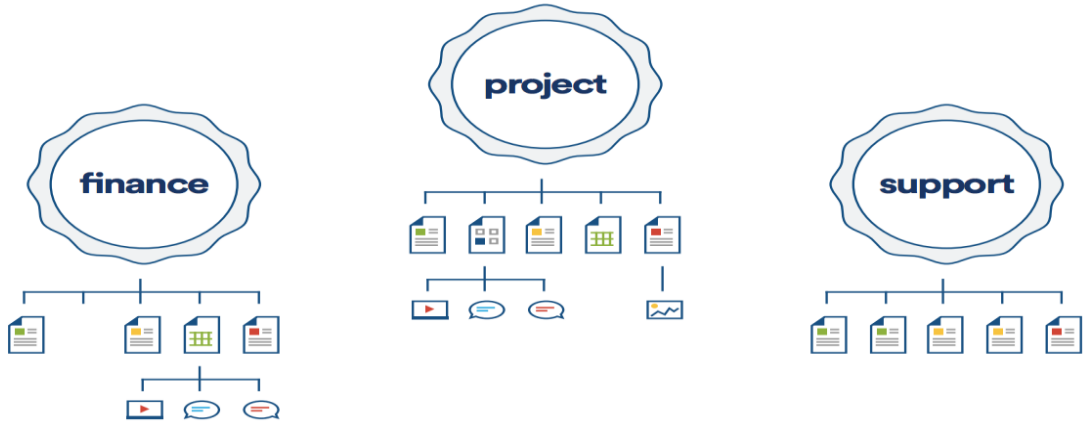
Project/Projekt - Název projektu do kterého se má úkol, chyba objevit.

Issue Type/Typ problem - Jak už ze samotného názvu vyplývá, jedná se o určení typu problému, například tester zde zadává chyby, analytik tu může zadávat takzvané change requesty neboli požadavky na změnu ve vývoji. Projektový manažer zde může založit úkol pro vývojáře atp.

2.5.2 Software Confluence

Confluence je nástroj určený pro spolupráci v oblasti obsahu, který pomáhá týmům efektivně spolupracovat a sdílet znalosti. S programem Confluence mohou uživatelé vytvářet stránky a blogy, které mohou komentovat a upravovat všichni členové týmu. Je zde například možnost snadno vytvořit plán, vytvořit poznámky obsahující kontrolní seznam, vytvořit znalostní databázi a vše centralizovat na jednom místě. Dále nabízí možnost připojit soubory různých typu (word, excel) a zobrazit je na stránce pro větší pohodlí. Confluence byl také navržen tak, aby se integroval s již dříve zmíněnou Jirou a má mnoho dalších integračních bodů, což uživatelům Confluence umožňuje zobrazit, komunikovat a odkazovat na problémy.

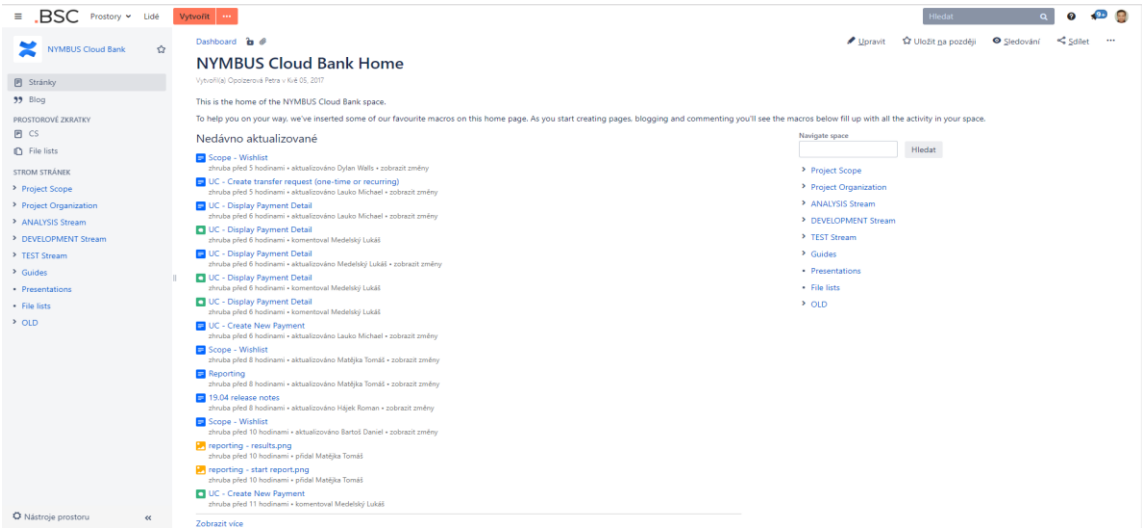
Obr. 2.8 Ilustrační obrázek rozděleného prostoru



Zdroj: [11]

V dnešní době internetu je zcela nezbytné, aby uživatele měli rychlý, efektivní způsob jak ukládat či sdílet své znalosti v reálném čase a nejlépe bez omezení. Proto je software Confluence v posledních letech absolutní špička v oblasti organizování online pracovního prostoru napříč světem. Není zde problém mít založeny takzvaný „prostor“, do kterého má přístup každý pracující člen týmu v daném projektu. Síla toho softwaru plyne právě z možnosti založit více takzvaných. „prostorů“, kde může mít prakticky celá firma uloženou interní dokumentaci a navzájem si aktualizovat nově nabité informace a zaznamenávat změny na jednotlivých projektech na kterých se zrovna pracuje, viz obrázek níže.

Obr. 2.9 NYMBUS Confluence stránka



Zdroj: vlastní foto, 2019

3 Zabezpečení přenosu a zpracování dat

K dnešnímu dni se uvádí zhruba 3 miliard uživatelů internetu, což vede k potřebě šifrovat data více než kdy jindy, aby byl přenos dat co nejvíce bezpečný. Přesně k tomu slouží protokol SSL (Secure Socket Layer). SSL okamžitě šifruje hesla nebo čísla kreditních karet do dat, které může dešifrovat pouze uživatel nebo určená webová stránka. Další činnost, kterou SSL zajišťuje je nezaměnitelnost dat společně s možností detekce potencionálních hrozeb a jejich eliminací v reálném časovém horizontu. Následující kapitola se bude věnovat protokolu SSL více do detailů společně s rozebráním

Na projektu specifikovaným v této práci se tento protokol využívá k zabezpečení komunikačního kanálu mezi jednotlivými komponentami.

3.1 Protokol SSL

SLL je standardní bezpečnostní protokol, který zajišťuje šifrovanou komunikaci mezi webovým serverem a prohlížečem. Toto spojení zaručuje, že veškerá data, která se posílají mezi webovým serverem a prohlížečem zůstávají zabezpečeny a šifrovány. SSL je v dnešní době brán jako průmyslový standard v oblasti IT a je používán miliony webových stránek jako hlavní bezpečnostní prvek pro online transakce mezi zákazníky.

Pro vytvoření SSL připojení je nutno, aby webový portál vlastnil SSL certifikát. Certifikát se dá získat, aktivovat po zodpovězení několik desítek otázek ohledně identity webového portálu společně s informacemi o vaší společnosti, potom se vytvoří dva zašifrované klíče: Private Key, Public Key.

Public Key (Veřejný klíč) - Veřejný klíč nemusí být tajný a je umístěn v požadavku CSR (Certificate Signing Request) datovém souboru, který obsahuje také vaše údaje. Poté byste měli předložit zprávu CSR. Během procesu žádosti o certifikát SSL certifikační autorita potvrdí vaše údaje a vydá certifikát SSL, který obsahuje vaše údaje a umožní vám používat protokol SSL. Váš webový server bude odpovídat vašemu vydanému SSL certifikátu vašemu soukromému klíči. Váš webový server pak bude moci vytvořit šifrované spojení mezi webovou stránkou a webovým prohlížečem vašeho zákazníka.

Komplexnost protokolu SSL zůstává pro zákazníky neviditelná. Jejich prohlížeče jim místo toho poskytují klíčový indikátor, který jim umožňuje vědět, že jsou v současné době chráněny šifrovanou relací SSL, ikona zámku v pravém dolním rohu, kliknutím na ikonu zámku zobrazí certifikát SSL a podrobnosti o něm. Všechny certifikáty SSL jsou vydávány buď společnostem, nebo právně odpovědným osobám.

Certifikát SSL bude obvykle obsahovat název domény, název společnosti, adresu, město, stát a zemi. Bude také obsahovat datum skončení platnosti certifikátu a údaje o certifikační autoritě odpovědné za vydání certifikátu. Když se prohlížeč připojí k zabezpečenému webu, načte certifikát SSL serveru a zkontroluje, že jeho platnost nevypršela, byla vydána certifikační autoritou důvěryhodnosti prohlížeče a že je používána webovou stránkou, pro kterou byla vydána. Pokud neuspěje žádná z těchto kontrol ověření, prohlížeč zobrazí varování koncovému uživateli, který jim umožňuje si uvědomit, že web není zabezpečen protokolem SSL.

Private Key (Soukromý klíč) - Soukromý klíč je textový soubor, který byl původně použit k vygenerování požadavku CSR (Certificate Signing Request) a později k zabezpečení a ověření připojení pomocí certifikátu vytvořeného na základě tohoto požadavku. Soukromý klíč se používá k vytvoření zabezpečeného digitálního podpisu. Z názvu může být dále odvozeno, že soukromý klíč by měl být pečlivě uschován a zabezpečen, protože každý kdo má přístup k soukromému klíči je schopen napáchat obrovské škody a využít tento přístup nekalím záměrům. Je důležité si uvědomovat, že soukromý klíč je sice pouze textový soubor, ale natolik důležitý, že je potřeba ho chránit odpovídajícím způsobem.

3.2 Protokol HTTPS

HTTPS je primárně navržen tak, aby poskytoval zvýšenou bezpečnostní vrstvu nad nezabezpečeným protokolem HTTP pro citlivá data, jako jsou transakce, fakturační údaje, transakce s kreditními kartami atp. HTTPS protokol šifruje každý datový paket v přechodu pomocí šifrovací techniky SSL nebo TLS, aby se zabránilo extrahování obsahu dat z řad hackerů a útočníků.

HTTPS je standardně nakonfigurován a podporován ve většině webových prohlížečů a automaticky inicializuje zabezpečené připojení, pokud přístupové webové servery

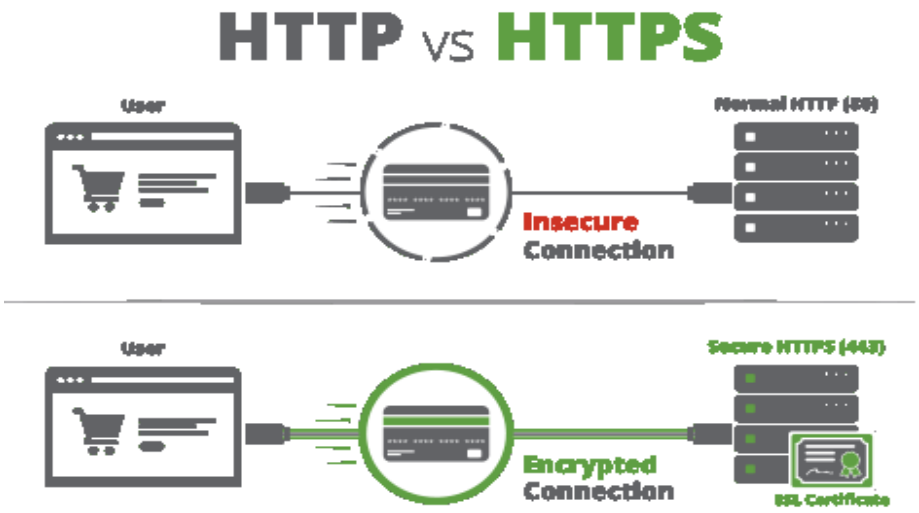
vyžadují zabezpečené připojení. HTTPS pracuje ve spolupráci s certifikačními autoritami, které vyhodnocují bezpečnostní certifikát přístupné webové stránky.

Na počátku se museli správci sítě domluvit a vymyslet postup, jak sdílet informace, které zveřejňují na internetu. Dohodli se na postupu ohledně výměny informací a nazvali ho HTTP (HyperText Transfer Protocol). Jakmile všichni věděli, jak si vyměňovat informace, tak pravděpodobnost zachycení takovéto informace někým cizím, kdo by jí správně neměl vidět, byla velmi vysoká. Vymyslel se tedy postup na ochranu informací, které se vyměňovali skrze internet. Tato ochrana se spoléhá na certifikát SSL pro šifrování online dat. Šifrování v tomto smyslu znamená, že odesilatel a příjemce souhlasí s tzv. kódem a převedou své dokumenty do náhodných řetězců a znaků. Procedura pro šifrování informací a jejich následná výměna se nazývá HTTPS (HyperText Transfer Protocol Secure).

Pokud kdokoliv mezi odesilatelem a příjemcem tuto zprávu otevře, tak této zprávě nemůže rozumět. Zprávu může rozluštit pouze odesílatel a její příjemce, kteří znají specifický kód. Lidé samozřejmě mohou zakódovat své vlastní dokumenty, ale počítače to umí dělat rychleji a efektivněji než lidé samotní. Počítač využívá toho, že na každém konci používá dokument nazvaný SSL certifikát obsahující řetězce znaků, které jsou klíčem k jejich tajným kódům.

Postup pro výměnu veřejných klíčů pomocí SSL certifikátu slouží pro povolení HTTPS, SSL a TLS protokolů se nazývá PKI (Public Key Infrastructure).

Obr. 3.1 HTTPS vs HTTP



Zdroj: [12]

4 Testovací procedury

Praktická část je zaměřena na zefektivnění testování pomocí automatizovaných testovacích nástrojů společně s popsáním manuálního testování a následným porovnáním manuálního a automatizovaného testování.

Na začátku této kapitoly je důležité si definovat, co pojem testování a testování softwaru znamená, a proč testování je nedílnou součástí vývoje.

Testování

„Je proces spouštění programu se záměrem nalézt chyby (ne dokázat jeho správnost). Testování ale neznamená pouze verifikaci běžícího programu, zahrnuje rovněž testování požadavků, revize dokumentace, inspekce kódu, statické analýzy, atd. Na testování programů je lépe nahlížet jako na destruktivní proces snažící se najít chyby (jejichž přítomnost se předpokládá) v programu“.[13 s. 1]

Testování softwaru

lze definovat jako odhalování chyb v aplikacích či programech. Třemi základními pilíři jsou v tomhle ohledu Analytici, vývojáři a testeři. Softwarový architekt nebo analytik specifikuje produkt společně se zákazníkem, který si určí, jak by měl výsledný softwarový produkt vypadat. Vývojáři jej naprogramují a testeři následně zkontrolují, jestli softwarový produkt splňuje zadání zákazníka, zda má veškerou potřebnou funkcionalitu, kterou zákazník vyžaduje a zda se bude uživatelům dobře používat. Všechny výše zmíněné pozice společně spolupracují a do určité míry testují v průběhu celého projektu.

Důležitost testování

Hlavním důvodem, proč testování je důležité, je fakt, že podcenění testování vede k nevyhnutelnému zvýšení prostředků či nákladů na vývoj samotný. Pokud se chyby nenajdou včas a objeví je až finální uživatel nebo zákazník, tak je nutné je v co nejkratší době opravit. Z toho plyne fakt, že náklady na opravu chyby, která se najde v rané fázi vývoje nebo ještě lépe při startu vývojového cyklu (sprintu), jsou daleko nižší než náklady na opravu produktu, který už je nasazen a používán zákazníkem. Faktem také je, že identifikování takovéto chyby společně s následnou opravou a dalším otestování, jestli tato oprava nezanesla do produktu potencionální problémy, je práce pro celý tým.

Obecně se dá konstatovat, že špatné otestování produktu vede k finančním ztrátám či ztrátě důvěryhodnosti. V minulosti se stalo několik výše zmíněných softwarových chyb např. firma Microsoft, při své první prezentaci operačního systému Windows 98 spustila instalaci ovladačů, a ty způsobily pád celého systému.

4.1 Hlavní fáze testovacího procesu

Jedná se v podstatě o komplexní logistický proces, který je potřeba si předem důkladně naplánovat. Pokud se jedná o oblast softwarového testování, tak tato oblast se dá rozdělit na tři části. [14]

- fáze příprav
- fáze provedení/testování
- fáze vyhodnocení

4.1.1 Fáze příprav

Požadavky managementu

- analýza dohodnutých požadavků,
- definování požadavků na testování,
- konkretizace funkčních a nefunkčních požadavků.

Plánování testování

- určení velikosti týmu (počet testerů na projekt),
- navrhnutí časového rozvrhu v souladu s předpokládaným vydáním softwaru.

Příprava/Vývoj testů:

- Příprava testovacích scénářů,
- Příprava testovacích případů,
- Příprava testovacích dat,
- Údržba testovacího prostředí.

4.1.2 Fáze provedení/testování

Provedení funkcionálních testů:

- Otestování zpětné kompatibility (starší data testované na novější verzi),
- Manuální nebo automatizované testování podle zadaných scénářů.

Provedení nefunkčních testů:

- provádění výkonových testů pro ověření výkonu systému a stability systému,
- provádění bezpečnostních testů pro simulaci zranitelnosti aplikace různými typy bezpečnostních útoků (XSS, SQL Injection, ...),
- ověření autorizačních a autentizačních logistických procesů,
- provádění specializovaných bezpečnostních testů.

4.1.3 Fáze vyhodnocení

Vyhodnocení testů:

- Analyzování výsledků z uskutečněných testů
- Podání reportu (zprávy).
- Sumarizace nalezených defektů.

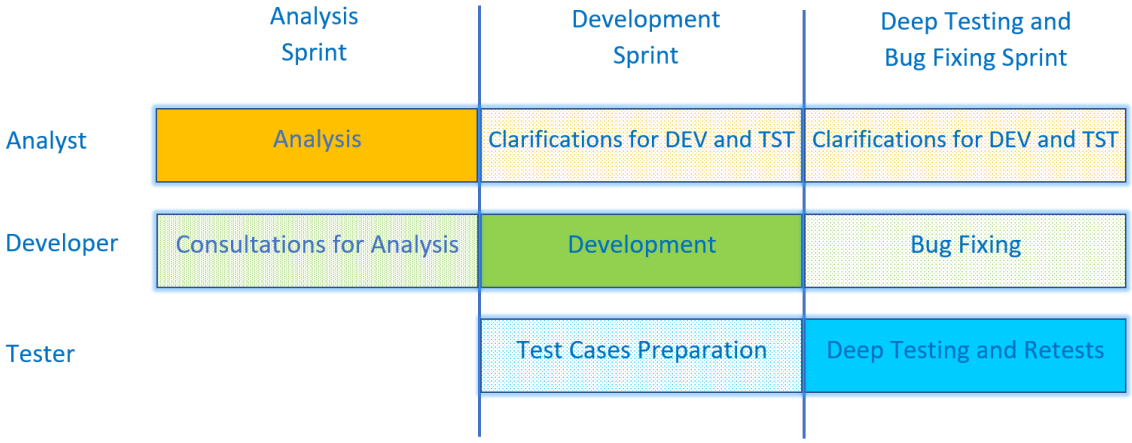
Jednotlivé fáze se dají přizpůsobit projektu na míru. Například je možné ve fázi testování úplně přeskočit výkonové testy pokud se jedná například o malou změnu v systému, která by neměla mít dopad na zvýšení zátěže, tudíž není potřeba využít tuto formu testování a ušetřený čas se může investovat do jiných např. funkcionálních testů.

4.2 Průběh vývojového sprintu (Quality Assurance)

Každému vývojovému sprintu předchází vytváření analýzy podle požadavku banky. Tento logistický proces je tvořen jak interním analytikem dodavatelské firmy, tak zadavatelskou firmou v tomto případě bankovní institucí. Po vytvoření a předání analýzy vývojovému týmu následuje fáze vývoje. Ve stejnou dobu nastupuje tzv. Quality Assurance neboli QA oddělení, které má za úkol připravit detailní testovací scénáře podle kterých bude probíhat testování. Analytik po vytvoření a předání analýzy v mezidobí vypomáhá vývojovému týmu a QA týmu s objasněním potencionálních

nejasností. Po ukončení vývojového cyklu nastává fáze důkladného testování celé aplikace pomocí manuálních či automatizovaných testů.

Obr. 4.1 Grafické znázornění vývojového sprintu



Zdroj: interní dokumentace [15]

Tento celý postup je definován v interním dokumentu do tří na sebe navazující fáze:

- Analýza
- Vývoj
- Testování

Analýza - na konci období určené pro vytvoření analýzy předá analytik dokumenty vývojářům a testerům. Předání se provádí na schůzce, kdy analytik vysvětluje hlavní funkčnosti. Ukazuje, kde najít všechny analytické dokumenty a snaží odpovědět na potencionální otázky ohledně funkčnosti z řad vývojářů a testerů.

Vývoj - zahájení vývoje podle předané analýzy. Jako součást vývojového sprintu si vývojáři píši vlastní testy (jednotkové testy, integrační testy.). Na konci vývojového sprintu je vyvinutá funkcionalita prezentována celému týmu.

Testování - Tester připravuje testovací scénáře. Pokud to je nutné, tak analytik objasní veškeré podrobnosti ohledně funkčnosti testerům. Testerský tým tyto vytvořené testovací scénáře předloží k přezkoumání analytikovi, jestli splňují kritéria domluvené při předání analýzy. Výsledkem je dohodnutý testovací plán včetně určení testovacích platforem a časové dotace nutné pro testování. Na konci důkladného testování testeři oznámí výsledky svých testů projektovému manažerovi. Pokud se vyskytnou

nevyřešené chyby, tak podle vyhodnocené priority jsou buď ihned opravovány, nebo je jejich oprava naplánována do dalšího vývojového sprint.

4.3 Manuální testování

Jak již bylo zmíněno v předchozích kapitolách, klíčovým konceptem manuálního neboli ručního testování je zajistit, aby aplikace byla bezchybná a fungovalo v souladu se stanovenými funkčními požadavky. Bohužel, ve většině případu nelze zajistit bezchybnost aplikace, jelikož existuje možnost, že takový scénář je krajně extrémní a nelze jej předpokládat. Testovací sady nebo tzv.cases (případy) jsou navrženy během již zmíněné testovací fáze a měly by stoprocentně pokrývat testovanou funkcionalitu v internetovém bankovnictví. Manuální testování dále zajišťuje také to, že ohlášené závady, chyby, které opraví vývojář, jsou opětovně otestovány. Toto testování v podstatě kontroluje kvalitu systému a snaží se dodávat zákazníkovi produkt, který se co nejvíce přibližuje k tzv. bug-free produktu.

Na druhy manuálních testů se dá pohlížet dvěma způsoby:

- Vývojářský přístup
- Testerský přístup

4.3.1 Vývojářský přístup

Vývojáři mají k dispozici následující 3 druhy testů, které si samostatně tvoří nezávisle na testerech.

Unit testy – jedná se o testy, které jsou napsány samotnými programátory nebo vývojáři a jedná se o kontrolu základních prvků v kódu společně s jeho funkčnostmi.

Modul testy – rozdíl toho testu oproti unit testu spočívá v jeho velikosti. Využití modul testů je spíše preferováno na velkých projektech, kde se snaží vývojář ověřit funkčnost modulu nebo jiné komponenty. Prakticky se od unit testu příliš neodlišuje a využívají se například pro testování celých knihoven.

Integrační testy – testy pro ověření, zda jednotlivé moduly, komponenty jsou schopny fungovat dohromady jako celek.

4.3.2 Testerský přístup

Z pohledu testera se jedná především o tzv. funkční testy, které se dělí do čtyř skupin:

Smoke test - je druh testování softwaru prováděného při nové iteraci produktu, aby se zjistilo, že kritické funkce programu fungují správně. Tento test se provádí před jakýmkoliv podrobným funkčním nebo regresním testováním. Účelem tohoto testu je odmítnout takto rozbitou aplikaci, aby QA tým nemusel ztrácet čas instalací a testováním aplikace, která od počátku nefunguje. Při tomto testování se použijí scénáře, které pokryjí nejdůležitější funkce a části systému/produktu. Cílem není provádět vyčerpávající testování, ale ověřit, zda kritické funkce systému fungují správně. Například se jedná o ověření, zda se aplikace či produkt spouští úspěšně, zda je možné se přihlásit do aplikace atp.

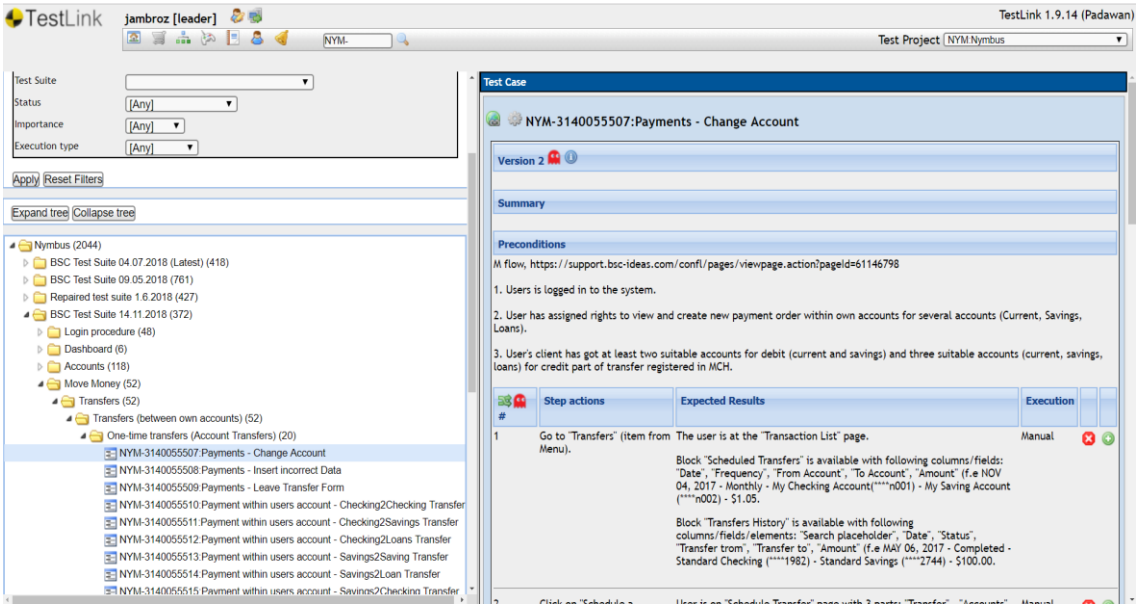
Sanity test – Jedná se o typ testování, kdy po obdržení nové verze by měly být opraveny chyby, které se v předchozích verzích objevili a byly nahlášeny QA oddělením. Cílem je zjistit, zda tyto chyby v důsledku změn nejsou příčinou zavedení dalších problémů. Nejedná se tedy o žádné důkladné ověření nové funkcionality, ale spíše o kontrolu tzv. bug fixování, které provádí vývojáři.

Regresní testy – testování, kdy se ověřuje, zda dříve vyvinuta a fungující funkcionalita, která byla otestována a prošla celým procesem, se chová stále stejně i po menší či větším zásahu do systému nebo propojení s nějakou další funkcionalitou či softwarem. Těmito změnami se myslí jakékoliv zlepšení softwaru, opravy či změny konfigurace. Během tohoto testování se někdy podaří odhalit nové chyby, které se objeví až pří těchto výše zmíněných změn.

V praxi to znamená, že smoke testy a sanity testy jsou vykonávány prakticky s každou novou verzi produktu, v tomto případě internetového bankovnictví. Je to z důvodu toho, že každá nová verze produktu sebou nese určité riziko neočekávaných chyb, a proto je potřeba otestovat veškeré hlavní funkce. Zákazník si tyto funkce může ověřit sám formou akceptačních testů, ale je potřeba říct, že se jedná spíše o finalizaci než o důkladné testování. Cílem těchto akceptačních testů je totiž ověřit, že produkt splňuje očekávané funkce, kvalitu na jiném než dodavatelském prostředí.

Všechny tyto typy scénářů se zapisují do programu TestLink. Jedná se o program, kde se skladují veškeré testovací scénáře, společně s verzemi na kterých se tyto scénáře testovali.

Obr. 4.2 Testovací scénáře v programu TestLink.



Zdroj: vlastní foto, 2019

Na obrázku číslo 4.2 je možné vidět na levé straně nabídku s již napsanými scénáři. Tyto scénáře se seřazují podle funkcionalit. Uprostřed lze vidět detail jednoho testovacího scénáře obsahující jednotlivé kroky vedoucí k otestování dané funkce. Důležité je poznamenat, že veškeré tyto testovací scénáře jsou psány podle analýzy, která je uložena na webové stránce Confluence. Proto je nutné uvést odkaz do podmínek, aby bylo možné zjistit, z čeho tester vycházel při psaní testovacího scénáře a také pro zpětnou kontrolu.

Všechny tyto zmíněné testy ať už patří do smoke či sanity kategorie jsou dále rozdělovány podle následujících popisů:

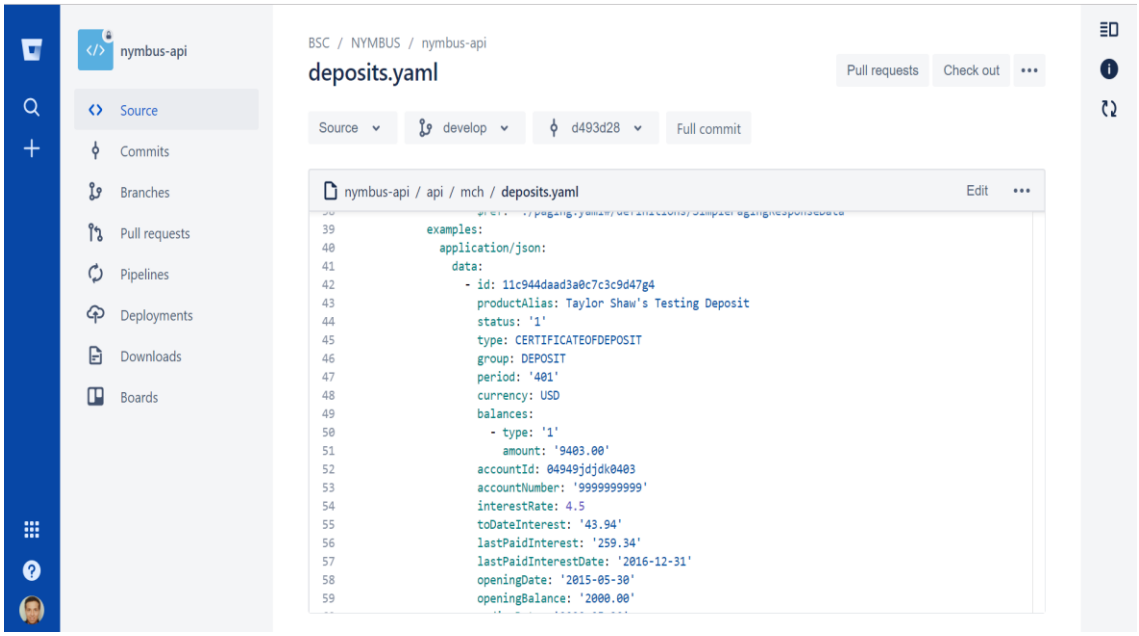
Negativní scénář – správný výstup tohoto scénáře je ve formě negace. Tedy očekávám, že pomocí toho scénáře nastane negativní výsledek a tudíž je výsledek správný. Příkladem je zákazník, který se snaží poslat více peněz než na svém účtu doopravdy má a tudíž mu internetové bankovnictví zakáže tuto operaci provést. Pokud se tak stane, je negativní scénář brán jako úspěšný.

Validační scénář – validace slouží k informování uživatele pro lepší orientování v aplikaci (co se děje, nebo co právě udělal uživatel). Příkladem je zaslání jednorázové platby na jiný účet, kdy po zadání všech údajů a odeslání požadavku je uživatel informován o úspěšném odeslání platby pomocí krátké zprávy. Nebo dalším příkladem je nesprávné zadání údajů, kdy je uživatel informován obvykle červeným textem s nápovědou, co by mělo dané pole obsahovat a v jaké formě.

Tvorba dat

Další nedílnou součástí je tvorba testovacích dat. Data jsou tvořeny podle API specifikace, která je uložena v interních dokumentech. Podle této specifikace je možné si vytvořit svůj vlastní JSON soubor, který se použije na provolání databáze, kde se vytvoří námi zvolené údaje. Na obrázku je vidět specifikace deposit účtu, který je něco na způsob spořicího účtu.

Obr. 4.3 Definice požadovaných příkazů



Zdroj: vlastní foto, 2019

Takový JSON soubor ovšem nedisponuje pouze jedním příkazem, ale je to vícero příkazů spojených dohromady. JSON soubor poté obsahuje např. příkaz pro vytvoření uživatele, vytvoření platebních příkazů, všech typů účtu atp. Poté se přistoupí k samotnému provolání pomocí příkazového řádku, kde zadáme jednoduchý příkaz, který provolá veškeré věci a uloží údaje do databáze a tester k těmto údajům může později přistupovat.

Obr. 4.4 JSON soubor pro vytvoření dat

```
        ]
      },
      {
        "externalCustomerId": "${customer1}",
        "externalAccountId": "${account_ira_saving_1}",
        "externalAccountType": "${CBS_ACCOUNT_TYPE_MAPPING.IRA_SAVINGS}",
        "relatedUsers": [
          {
            "externalCustomerId": "${customer1}",
            "relationshipType": "${CLIENT_ACCOUNT_RELATION_TYPES.PRIMARY_OWNER}"
          }
        ]
      },
      {
        "externalCustomerId": "${customer1}",
        "externalAccountId": "${account_ira_saving_2}",
        "externalAccountType": "${CBS_ACCOUNT_TYPE_MAPPING.IRA_SAVINGS}",
        "relatedUsers": [
          {
            "externalCustomerId": "${customer1}",
            "relationshipType": "${CLIENT_ACCOUNT_RELATION_TYPES.PRIMARY_OWNER}"
          }
        ]
      }
    ],
    "balanceImports": [
      {
        "externalAccountId": "${account_checking_1}"
      },
      {
        "externalAccountId": "${account_checking_2}"
      },
      {
        "externalAccountId": "${account_checking_3}"
```

Zdroj: vlastní foto, 2019

Pomocí těchto JSON souboru je možné vytvářet jakýkoliv druh dat pomocí kterých je tester schopen otestovat široké spektrum scénářů. Výhodou této metody vytváření dat je jednoduchá možnost reprodukování nalezené chyby v aplikaci. Vývojář je schopen si vzít tento JSON a reprodukovat nalezenou chybu se stejnými daty jako tester, což vede k zvýšení efektivity v oblasti oprav chyb.

Jak se dá logicky odvodit, takové testování vyžaduje obvykle velkou časovou dotaci, a pokud taková časová dotace není testerům umožněna, tak to vede k potencionálním finančním ztrátám, proto se spíše jedná o způsob testování, který je na ústupu a je pomalu nahazováno automatizovaným testováním. Faktem ale zůstává, že stále bude vždy potřeba manuálně testovat určité oblasti, takže není možné, aby zcela zmizelo.

4.4 Automatizované testování

V oblasti testování zažívá automatizované testování enormní nárůst popularity jak v řadách vývojářů, testerů tak i samotných bank. Automatizované testování je proces, který ověřuje, zda software správně funguje stejně jak je tomu například u smoke testování, ale s tím rozdílem, že se používají skriptované sekvence, které jsou prováděny testovacími nástroji. V dnešní době se používá zhruba 5 nejznámějších automatizovaných testovacích nástrojů: Selenium, IBM Rational Functional Tester (RFT), Cucumber, TestComplete, eggPlant. Absolutní špičkou je v dnešní době Selenium, díky obrovskému potencionálu, který se v něm skrývá, tudíž zde bude rozebrán.

Selenium – je open source nástroj, který se používá pro automatizaci testů prováděných na webových prohlížečích (webové aplikace jsou testovány pomocí libovolného webového prohlížeče). Vzhledem k tomu, že selenium je open-source, tudíž nevznikají žádnéé licenční náklady, což je jedna z hlavních výhod proč se těší tak velkému zájmu ve světě. Dalšími důvody, proč je selenium tak populární jsou:

- Testovací skripty mohou být napsány v některém z těchto programovacích jazyků: Java, Python, C#, PHP, Ruby
- Testy se mohou provádět na jakémkoliv operačním systému (Windows, MAC, Linux)
- Testy lze provádět pomocí libovolného prohlížeče (Mozilla Firefox, Internet Explorer, Google, Chrome, Safari nebo Opera)
- Selenium může být integrována do Jenkinks nebo Docker aplikace a dosáhnout tím kontinuálního testování.

Automatizované testy vs manuální testy

Automatizované testy poráží ty manuální snad ve všech ohledech. To je primárně dáno vysokou rychlostí a potřebnou investicí do lidských zdrojů, která je oproti manuálnímu testování relativně nízká. Dalším důvodem je menší náchylnost k chybám a možnost častěji provádět testování. Jako příklad se může uvést přihlašování do systému. Předpokladem je existující přihlašovací stránka a tester musí ověřit, zda jsou všechny pokusy o přihlášení úspěšné, pak je mnohem snazší napsat kus skriptu, který bude platit pro všechny pokusy o přihlášení a informovat nás o výsledku. Tyto testy lze navíc

konfigurovat tak, aby byly testovány v různých prostředích a webových prohlížečích. Klíčovým bodem každopádně zůstává, že automatizované testování dělá práci testerů mnohem jednodušší a časově efektivnější.

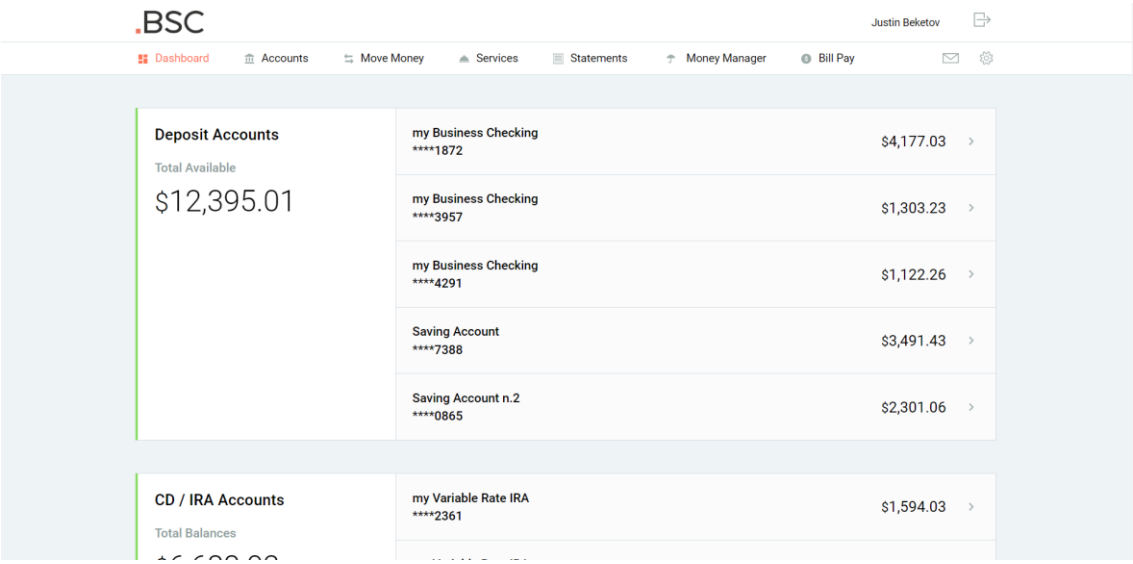
4.5 Použití automatizovaných testů na projektu

Na projektu NYMBUS se prozatím testuje především manuální cestou, ale s přibývající funkcionalitou a složitostí začala vyvstávat otázka, zdali není lepší aspoň část manuálního testování převést do automatizované podoby a zefektivnit celý logistický proces.

Použité testovací nástroje

Po delší úvaze se rozhodlo využít open-source nástroj Katalon Studio. Jedná se o software, který je schopen pomocí předem naprogramovatelných skriptů prozkoumat jak celou webovou aplikaci, tak i mobilní. Z důvodu časové náročnosti je zde uvedeno testování pouze webové aplikaci takzvaný FE (Front-End), který je asi nejpodstatnější část pro uživatele, kteří skrz webovou aplikaci ovládají celé internetové bankovnictví.

Obr. 4.5 Internetové bankovnictví z pohledu uživatele



Zdroj: vlastní foto, 2019

Na obrázku lze vidět hlavní menu společně s účty, které jsou pro daného uživatele k dispozici. Tyto data jsou vytvořeny pomoci již dříve zmiňovaného JSON souboru. Časová dotace k otestování základních 150 – 200 scénářů je pomocí manuálního

testování odhadována kolem 2-4 dní, tento odhad je především ovlivňován počtem lidí zapojených do testování. Pro lepší pochopení zde bude uvedena tabulka s časovými intervaly.

Tab. 4.1 Časový odhad manuálního testování

Název testu	Počet testů	Odhadována délka testování na dny
Smoke test	150 – 200	2-4
Sanity test	200 - 250	4-6
Regresní test	380	7-10
Celkem	830	20

Zdroj: vlastní zpracování dle interních materiálů, 2019

Celkově se tedy jedná o 830 testovacích případů na 20 dní, to je relativně velké množství času. Z tabulky jasně vyplývá, že manuální testování je časově náročné. Tato časová náročnost se liší podle toho, v jakém stádiu se projekt nachází. Pokud je projekt na svém začátku, tak je možné mít otestováno celou aplikaci během 1-2 dní, ale postupem času se přidávají nové a nové funkcionality a navyšuje se potřebný čas pro testování, což vede nevyhnutelně buď k zapojení více lidí do testování a s tím spojené vyšší náklady na vývoj nebo přejití na automatizované testy, které sice vyžadují určitou časovou investici, ale s tím rozdílem, že ve výsledku se tento investovaný čas vrací v podobě rychlejšího a přesnějšího testování s nižšími náklady na údržbu.

Katalon Studio

Tento nástroj kombinuje několik programovacích jazyků či skriptů pomocí nich se dá jednoduše napsat většina testovacích scénářů kromě některých výjimek typu zaslání notifikace na mobilní zařízení přes třetí stranu atp. Na skriptovací sekvence se využívá jazyk Groovy a na programování pak JAVA. Kombinace těchto dvou jazyků má za výsledek přenesení značné části manuálních testů do automatizované podoby. Jednou z velkých výhod je také možnost si velkou část věcí manuálně nastavit pomocí kvalitně zpracovaného UI s následným převedením do programovacího jazyka. Toto je možné využít i naopak, kdy pomocí skriptu či programovacího jazyka se nám vytvoří UI schéma.

Přidání objektů, elementů, funkcí

Objekty jsou vytvářeny přímo v UI. Na levé straně při spuštění programu je záložka Tests Explorer, kde jsou následující položky:

- Test Cases – zde jsou uloženy veškeré testovací scénáře podle kterých se testuje.
- Object Repository – zde jsou ukládány objekty společně s detailním popisem cesty k nim. Tyto objekty jsou například pole na vyplnění účtu, zadání částky atp.
- Test Suites – slouží k nakonfigurování jednoho velkého testu, který se skládá z menších testovacích případů, které jsou uloženy v Test Cases.
- Data Files – pomocí této funkce je uživatel schopen uložit data, která si bude průběžně volat a používat ve vícero instancích.
- Checkpoints –kontrolní bod je snímek zkušebních dat pořízených v určitém čase. Kontrolní bod se používá k ověření, zda je aktuální stav zdroje dat odlišný od předchozího stavu
- Keywords – slouží k ukládání funkcí a jejich programování. Příkladem je například osvětlení položky, na které se uživatel v daný moment nachází.
- Test Listeners – uživatel si může zvolit testovací kroky, které jsou vykonány po zvoleném kroku.
- Reports – slouží k vytvoření reportů po ukončeném testovacím cyklu.
- Include – zde se nachází nastavení skriptovacího jazyka, log souborů

Prvním krokem je pomocí již definovaného testovacího případu, který je uložen v TestLinku nalézt veškeré objekty nutné k testování. Toto je možné udělat buď manuálně pomocí prozkoumání zdrojového kódu přímo v prohlížeči nebo pomocí tzv. Web Spy funkce, kdy se průběžně ukazují jednotlivé objekty a elementy na které zrovna uživatel svým kurzorem ukazuje. Cesty k objektům jsou definovány pomocí:

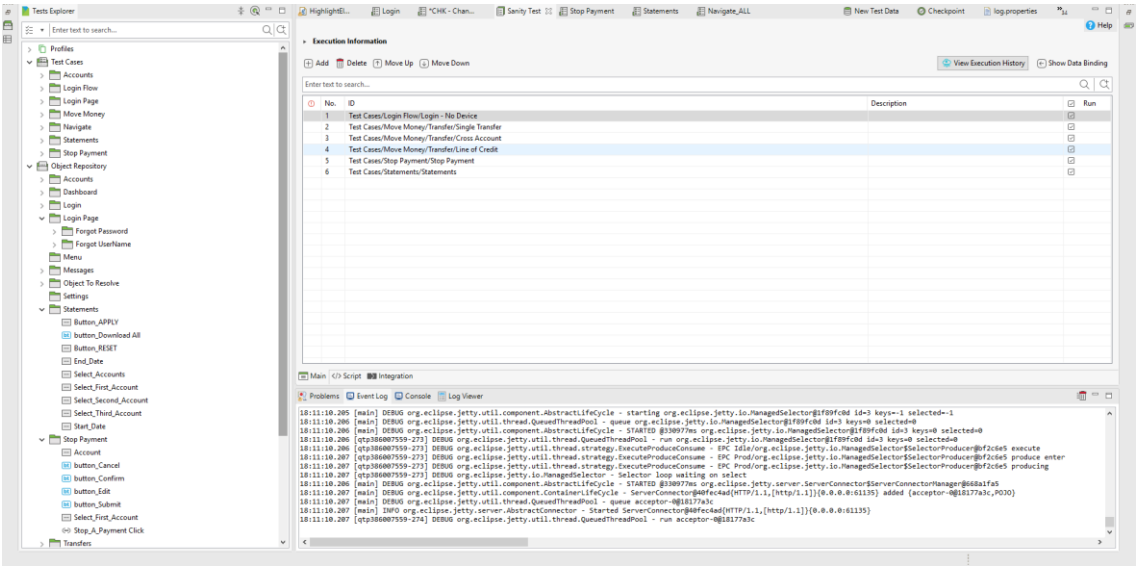
- xPath, id, class, name, text

Pokud jsou objekty správně nastaveny je možné přistoupit již k samotnému psaní testů.

Vytvoření automatizovaného testu a jeho porovnání

Pro vytvoření prázdného test casu stačí pravým klikem na záložku Test Cases zvolit nový test case a prázdný testovací scénář je vytvořen. V horní liště jsou záložky: Add, Recent keywords, Delete, Move up, Move down, Edit Tags pomocí nich vytváříme daný testovací případ neboli scénář. Podle scénáře z TestLinku si nastavíme jednotlivé komponenty společně s funkcemi, které se mají vykonat v jednotlivých krocích. Tím je myšleno například kliknutí na záložku platby, vybrání účtu atd. Důležité je také nastavení prohlížeče a prostředí, na kterém se bude scénář testovat, což se nastavuje pomocí záložky profiles, kde jsou také uchovány veškeré předem nadefinované údaje jako je právě URL.

Obr. 4.6 Vytvořené aut. testy v programu Katalon



Zdroj: vlastní foto, 2019

Pro účel této práce bylo zvoleno 6 jednoduchých scénářů, které porovnáme s klasickým manuálním testováním. Testováno je:

- přihlášení uživatele bez udání zařízení,
- jednorázová platba, platba do stejné banky,
- platba hypotéky,
- vyfiltrování výpisu,
- Zastavení platby.

Jedná se o scénáře, kdy se jedná o jednoduché projití položek a zkontrolování všech objektů a jejich správné zobrazování na stránce.

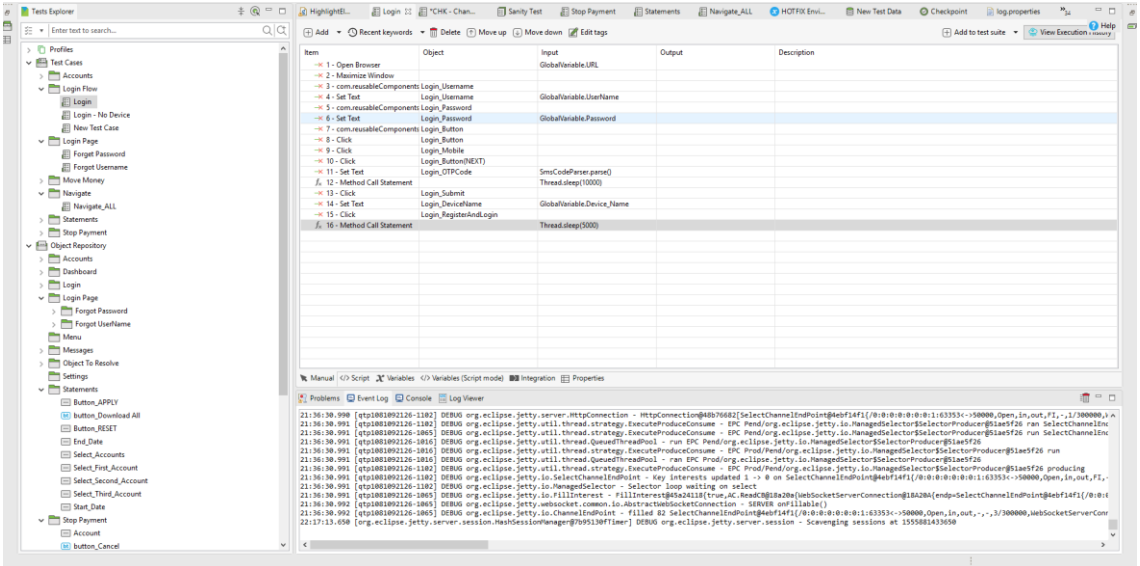
Tab. 4.2 Srovnání mezi manuálním a automatizovaným testováním

Způsob testování	Čas v minutách
Manuální	01:46,03
Automatizované	00:34:55

Zdroj: vlastní zpracování, 2019

Již při takto malém množství testovacích scénářů je čas prakticky více než poloviční. Díky této metodě je možné, aby tester prakticky pustil nadefinované scénáře a věnoval se například tvorbě specifických scénářům, které nelze otestovat pomocí tohoto nástroje nebo přípravě dalších dat. Tímto způsobem je možné ušetřit, až desítky hodin čistého času při takto velikém projektu, jakým je projekt NYMBUS. Zde je jen malá ukázka toho, co je pomocí automatizovaného testování dosáhnout.

Obr. 4.7 Detailní pohled na testovací scénář v Katalonu



Zdroj: vlastní foto, 2019

5 Identifikace slabých míst

V této kapitole jsou popsána slabá místa, která jsou obecně známá společně s identifikací slabých míst v oblasti testování, na které se postupem psaní této práce přišlo.

Šifrování jiných dat - projekt NYMBUS je specifický tím, že míří na americký trh, kde se žádají po dodavatelích jiné zvyklosti, než je tomu na evropském trhu. Základním bodem je přístup k šifrování dat. V Americe existují dva nejdůležitější údaje a těmi jsou tzv. SSN (Social Security Number) a Account Number. V Evropě nejsme zvyklí tyto údaje jakkoliv šifrovat, jelikož pomocí těchto čísel nejsme schopni udělat jakoukoliv nelegální činnost. Proto je potřeba tyto údaje šifrovat a zabezpečit proti případnému úniku.

Časová náročnost při vytváření automatizovaných testů - z pohledu testování a především automatizovaného testování se dají považovat za slabé místo některé nevyzpytatelné chybové hlášky, které se během psaní těchto testů mohou vyskytnout. Jedná se především o náhodné pády nebo nemožnost zacílit objekty nutné pro splnění daného scénáře atp., což se může brát jako důkaz toho, že pro vytvoření automatizovaných testů je potřeba investovat dostatečný čas. Tento čas se postupně navyšuje s přibývající funkcionalitou.

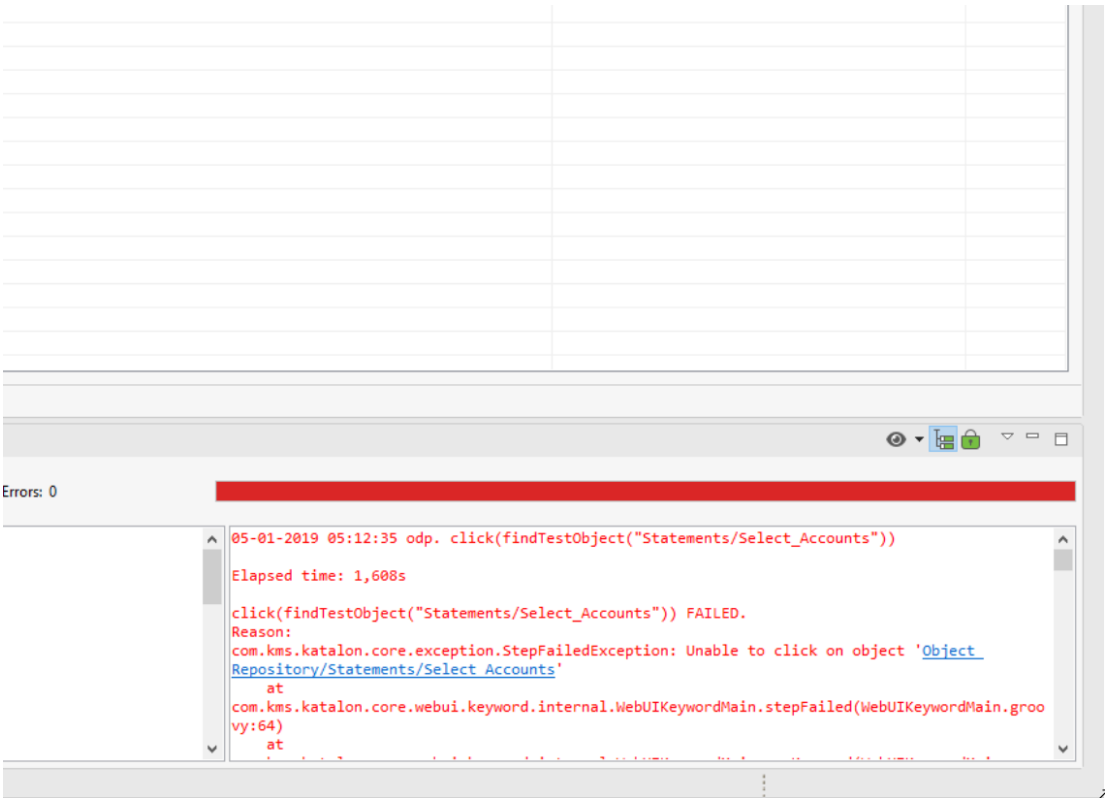
Programová vybavenost - pro spuštění automatizovaných testů ve čtvrté kapitole bylo potřeba například napsat funkci, která je schopna ze zabezpečené stránky vzít tzv. OTP kód, který se generuje při každém přihlášení do internetového bankovnictví a vložit jej po pole pro úspěšné přihlášení do aplikace. Bez znalosti programovacího jazyka by bylo prakticky nemožné se přes tento bod dostat, tudíž lze konstatovat, že bez určité znalosti programovacích nebo skriptovacích jazyků není možné psát automatizované testy.

Údržba testovacích scénářů - udržování testovacích scénářů je něco, kde je pravděpodobnost velkých časových ztrát. To je zapříčiněno do určité míry agilním vývojem, protože i přesto, že jsou na začátku projektu odsouhlaseny veškeré aspekty analýzy, tak se stává, že některé položky jsou dodatečně změněny a tudíž se v některých případech musí měnit testovací scénáře. Pokud jsou na projektu nasazeny automatizované testy a do toho se manuálně testuje, tak to ve výsledku znamená, že je

potřeba přepsat testovací scénáře v obou případech. Při malých položkách se nejedná o velké ztráty, ale při přepisování například 50 až 100 scénářů je už časová ztráta značná.

Lidský faktor – jak již bylo zmíněno několikrát v předchozích kapitolách zmíněno slabým místem bude vždy do určité míry člověk, proto je vhodné ho buď úplně nahradit, nebo co možná nejvíce omezit jeho vstup do procesu. V tomto případě se jedná o omezení manuálního testování a nahrazení pomocí automatizovaných testů, což má za následek vyšší pokrytí testovacích scénářů s menší procentuální chybovostí, ale na druhou stranu je zde potencionální riziko, že se například na grafickou chybu nepřijde dostatečně brzy nebo vůbec, to je ovšem dáno samotnou podstatou automatizovaného testování, jelikož automatizované testy spíše kontroluji existenci jednotlivých prvků na stránce společně s texty, které jsou na nich umístěny, tudíž se může stávat, že grafické chyby nemusí odhalit. Jedna z věcí, která se relativně běžně praktikuje pro co největší zefektivnění testování na projektech je kombinace manuálního a automatizovaného testování. To má za následek, že jakýkoliv problém s nepřívětivým UI (User-Interface) nebo se špatnou funkčností aplikace jsme schopni odhalit s relativně velkou přesností a časovým předstihem.

Obr. 5.1 Špatně zacílený objekt v Katalon studio



Zdroj: vlastní foto, 2019

Závěr

Většina z nás používá internetové bankovnictví na denní bázi, neboť se jedná o moderní produkt, díky kterému jsme schopni ušetřit obrovské množství času, který může člověk investovat do jiných oblastí svého života.

Cílem bakalářské práce bylo analyzovat celý proces ukrývající se za distribucí dat a následné zpracování těchto dat v oblasti testování v aplikaci internetového bankovnictví s návrhem na zefektivnění testovací procesu.

Práce byla rozdělena na teoretickou a praktickou část. V rámci teoretické části jsou především definovány základní bankovní systémy společně s popisem jednotlivých komponent, které se využívají na logistický přenos informací. Dále je popsáno implementační schéma internetového bankovnictví s následným postupem šifrování tohoto přenosu pomocí různých metod a protokolů. Praktická část byla zaměřena na definování pojmu testování a popsáním jednotlivých fází a procesů, které vedou k detekci nežádoucích chyb nebo neobvyklého chování v aplikaci internetového bankovnictví. Dalším krokem byla tvorba testovacích dat a jejich následné použití při realizaci automatizovaných testů, které byly poté porovnány oproti manuálním testům.

Pomocí automatizovaného testování bylo zjištěno, že už při malém počtu testovacích scénářů se dá konstatovat, že dochází k zefektivnění testovacího procesu díky časové úspoře, která je vyšší zhruba trojnásobně oproti testování manuálním přístupem.

Výsledným zjištěním provedeného testování bylo, že i přes dosažené výsledky nelze zautomatizovat veškeré testovací scénáře, což je především dáno velmi specifickými testovacími případy, které není možné převést do automatizované podoby jako například přijetí SMS upozornění skrze aplikace třetích stran nebo grafické chyby které není obvykle možné zachytit pouze manuálním testováním.

V rámci poslední kapitoly byly identifikovány a shrnuty obecná i specifické místa, která se dají označit za důvody proč použít nebo nepoužít automatizované testy na daném projektu.

Soupis bibliografických citací

[1] KOTLER, Philip et al. *Moderní marketing: 4. evropské vydání.* 1. Vyd.Praha: Grada, 2007. ISBN 978-80-247-1545-2.

[2] *Služby v obecném hospodářském zájmu v EU: Komparace České republiky a Německa* [online]. [cit. 2019-04-09]. Dostupné z: https://www.ekf.vsb.cz/export/sites/ekf/saei/cs/Text/SAEI_VOL07_ex.pdf

[3] VYMĚTAL, Dominik. *Informační systémy v podnicích: teorie a praxe projektování*. Praha: Grada, 2009. Průvodce (Grada). ISBN 978-80-247-3046-2.

[4] DOCPLAYER, *Informační systém banky* [online]. [cit. 2019-04-20]. Dostupné z: https://docplayer.cz/15738595-Informacni-system-banky.html

[5] Wikisofia, *Agilní metody projektování. Principy, role, organizace, nástroje* [online]. [cit. 2019-04-03]. Dostupné z: https://wikisofia.cz/wiki/Agiln%C3%AD_metody_projektov%C3%A1n%C3%AD._Principy,_role,_organizace,_n%C3%A1stroje

[6] *Nymbus architecture* [online]. [cit. 2019-04-10]. Dostupné z: https://bitbucket.org/bscideas/nymbus-guides/src/develop/

[7] Uživatelská příručka pro zkratky: Co je API? [online]. [cit. 2019-04-10]. Dostupné z: https://support.apple.com/cs-cz/guide/shortcuts/apd2e30c9d45/ios

[8] *Phpenthusiast: What is REST API? in plain English* [online]. [cit. 2019-04-30]. Dostupné z: https://phpenthusiast.com/blog/what-is-rest-api

[9] *Atlassian: JIRA* [online]. [cit. 2019-04-12]. Dostupné z: https://support.crowdin.com/assets/docs/integration_jira_subtask.png

[10] *Atlassian: Service Desk* [online]. [cit. 2019-04-12]. Dostupné z: https://confluence.atlassian.com/servicedeskcloud/files/945104207/945104208/2/1534304829803/JIRAServiceDesk-Queues.png

[11] *Atlassian: Confluence 101: organize your work in spaces* [online]. [cit. 2019-04-

[12]. Dostupné z: https://www.atlassian.com/collaboration/confluence-organize-work-in-spaces

[12] *Webcreator: HTTP vs HTTPS* [online]. [cit. 2019-04-13]. Dostupné z: http://webecreator.com/blog/wp-content/uploads/2018/06/17-sucuri-a-ssl-http-vs-https-chart@2.png

[13] *Význam testování, teminologie, testovací proces: Testování* [online]. [cit. 2019-04-]. Dostupné z: http://lucie.zolta.cz/index.php/statnice-vsb/166-vyznam-testovani-terminologie-testovaci-proces

[14] *Test Methodology: Methodical Specification* [online]. [cit. 2019-04-30]. Dostupné z: https://bscideas.sharepoint.com/SitePages/HR.aspx

[15] *Quality Assurance Principles: Sprint Life Cycle* [online]. [cit. 2019-04-13]. Dostupné z: https://support.bsc-ideas.com/confl/pages/viewpage.action?spaceKey=20009291&title=Quality+Assurance+Principles

Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek

Autor (vypracoval)	**Jiří Ambroz, DiS.**
Název BP	**Logistické procesy a zpracování testovacích dat v oblasti internetového bankovnictví**
Studijní obor	**Informační Management**
Rok obhajoby BP	**2019**
Počet stran	58
Počet příloh	0
Vedoucí BP	**doc. Dr. Ing. Oldřich Kodym**
Oponent BP	
Anotace	Tato bakalářská práce je zaměřena na popsání celého procesu ukrývajícího se za distribucí dat a jejich následné zpracování či zefektivnění v oblasti testování dat v aplikaci internetového bankovnictví. V teoretické části je popsána historie internetového bankovnictví společně s definicí služby, do které spadá, a následně jsou vysvětleny jednotlivé komponenty používané v oblasti internetového bankovnictví společně se zabezpečením datového přenosu. Následně jsou v praktické části uvedeny zásady testování společně s návrhem vlastního zlepšení v oblasti automatického testování, což je společně s následným identifikováním slabých míst hlavním cílem této práce.
Klíčová slova	Internetové bankovnictví, testovací data, data, distribuce, komponenty, služba, proces, produkt
Místo uložení	ITC (knihovna) Vysoké školy logistiky v Přerově
Signatura	