Návrh aplikace pro devizovou a online platbu

Bc. Jaroslav Kadleček

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jaroslav Kadleček**
Osobní číslo:	**A15363**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Návrh aplikace pro devizovou online platbu**
Téma anglicky:	**The Design of an Online Foreign Exchange Payment Application**

Zásady pro vypracování:

1. **V teoretické části popište technologie pro distribuci zpráv v režimu klient-server, výkon aplikace a multiplaformní řešení klienta.**
2. **Analyzujte možnosti a navrhněte řešení škálování výkonu, odolnosti proti výpadku a odolnosti proti útoku.**
3. **Popište možnosti a navrhněte řešení vícefaktorové autentizace.**
4. **Navrhněte ukázkovovou aplikaci umožňující aktualizaci vybraných kurzů v reálném čase, zadávání příkazů k transakci a zobrazení aktuálního stavu transakcí.**
5. **Vytvořte ukázkovovou aplikaci.**
6. **Popište klíčové částí ukázkové aplikace.**
7. **Formulujte závěry a demonstrujte výsledky.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **NAGEL, Christian. Professional C# 6 and .Net Core 1.0. Indianapolis, IN: Wrox, a Wiley brand, published by John Wiley & Sons, 2016. ISBN 111909660X.**
2. **ASP.NET SignalR [online]. 2017 [cit. 2017-01-27]. Dostupné z: http://signalr.net/**
3. **VELU, Vijay. Mobile Application Penetration Testing. Birmingham: Packt Publishing, 2016. ISBN 978-1785883378.**
4. **HERMES, Dan. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. Berkeley, CA: Apress, 2015. ISBN:978-1-4842-0215-9.**
5. **Https://www.owasp.org/ [online]. 2016 [cit. 2017-01-27]. Dostupné z: https://www.owasp.org**
6. **ANDERSON, Jonathan., John. MCREE a Robb. WILSON. Effective UI. Cambridge: O'Reilly, c2010. ISBN 059615478X.**
7. **COPELAND, Marshall. Microsoft Azure: planning, deploying, and managing your data center in the cloud. Berkeley, CA: Apress, 2015. Expert's voice in Microsoft Azure. ISBN 1484210441.**
8. **GRIGORIK, Ilya. High-performance browser networking. Sebastopol, CA: O'Reilly Media, Inc., 2013. ISBN 1449344763.**

Vedoucí diplomové práce: **Ing. Erik Král, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

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

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 11.5.2017 …….

podpis diplomanta

ABSTRAKT

Tato práce je zaměřena na návrh zabezpečení provozu aplikace pro online devizové platby. Pro realizaci plateb se na straně klienta využívá jak přístup přes standardní webový prohlížeč, tak i mobilní aplikace. Cílem je navrhnout taková opatření, aby byla zajištěna odolnost systému proti možným bezpečnostním hrozbám a vyhověla požadavkům kladeným legislativou i doporučením regulátora. V práci jsou probrána jednotlivá bezpečnostní rizika a možnosti jejich řešení. Další oblastí je pak zajištění kontinuity provozu systému, který může být ohrožen buď nedostatečnou kapacitou, nebo výpadky systému. Následně jsou navržena řešení infrastruktury systému tak, aby toto riziko bylo minimalizováno a současně aby toto řešení umožnilo poskytovali služby kontinuálně navyšovat výkon systému. Na závěr je předvedena ukázková aplikace, na které jsou demonstrovány výsledky.

Klíčová slova:

zabezpečení, internetové platby, webové služby, automatická škálovatelnost, vysoká dostupnost, vícefaktorové zabezpečení

ABSTRACT

The thesis deals with an online foreign exchange payment application design and its crucial attribute - secure operation. Clients use standard Web browser, and mobile applications for making payments. The aim is to propose measures to ensure the system's robustness against possible security threats and satisfy the legislative and regulatory requirements and recommendations. Various security risks and respective solutions are discussed. Furthermore, threats to business continuity - such as lack of capacity and system failures - are addressed. The infrastructure for minimizing risks and optimizing system performance through scalability is proposed. To conclude, the draft for payment transaction security with two factored authentications demonstrates the chosen solution.

Keywords:

web application security, online payments, webservices, auto-scaling, high availability, multi-factored authentication

Tímto bych rád poděkoval panu Ing. Erikovi Královi, Ph.D. za rady, připomínky a veškerý jeho čas, který mi věnoval při odborném vedení této práce. Rovněž bych rád poděkoval touto cestou i svým kolegům za cenné podněty.

OBSAH

ÚVOD

Internetové platby jsou dnes již běžné a uživatelé jsou na ně zvyklí. S nástupem mobilních technologií roste význam poskytovat platební služby i na těchto platformách. Konkrétně u plateb v cizí měně je pro uživatele velmi důležitá informace o aktuálním kurzu, nebo v případě nevýhodného kurzu chce být uživatel ihned informován, že kurz je již příznivý a může platbu provést.

Čím větší je tok peněz, tím více roste riziko, že na službu bude proveden útok s cílem ji buď zkompromitovat vyřazením z provozu a tím poškodit poskytovatele, nebo v horším případě získat nad transakcemi kontrolu, nezákonně se obohatit a poškodit tím uživatele. S takovými hrozbami je potřeba při návrhu platebních systémů počítat, protože pro bankovní instituce má jakékoliv selhání v otázkách bezpečnosti velmi významné dopady na důvěru klientů v poskytované služby.

V teoretické části této práce budou popsány požadavky na systém, oblast rizik a technologie, kterými je možné dané problémy vyřešit. V praktické části bude navržena ukázková aplikace, která umožní otestovat navržené řešení.

I. TEORETICKÁ ČÁST

1 SYSTÉM PRO DEVIZOVOU A ONLINE PLATBU

Systém je založen na spotovém obchodu. Spotový obchod je takový typ transakce, kde daný předmět obchodu je dodán okamžitě a transakce je vypořádána během spotového data. Aktéry v systému jsou: externí zdroj kurzů měnových párů, platební systém obchodníka a klient.

- **Externí zdroj kurzů měnových párů**
  Poskytuje subjektům, se kterými má uzavřen smluvní vztah, aktualizaci kurzů měnových párů. Typicky aktualizace probíhá každou vteřinu.
- **Platební systém obchodníka**
  Zpracovává požadavky klientů, umožňuje výpis stavu devizových účtů klienta a historii transakcí. Umožňuje klientovi náhled na aktuální kurzy jednotlivých měnových párů.
- **Klient**
  Odesílá požadavky do platebního systému, potvrzuje platby, prohlíží si historii transakcí a stavy účtů.

Platební systém periodicky přebírá informace o aktuálním stavu kurzu měnových párů z externího systému a umožnuje jejich zobrazení klientům. Pokud klient zadá požadavek na uzavření obchodu, tak je předán do systému, který zajišťuje vypořádání obchodů. Po dobu trvání transakce je klientovi držen systémem nabídnutý kurz. Rychlost uzavření transakce je v zájmu obou zúčastněných stran. Systém vypořádání při vložení žádosti ověřuje, zda klient má na svém účtu dostatečnou hotovost na uzavření transakce a zda má dostatečnou likviditu na straně obchodníka. Pokud je všechno v pořádku, je transakce uzavřena. Na účet klienta je připsána hotovost v klientem požadované měně a současně je klientovi odečtena cena v měně, dle kurzu daného měnového páru. Odečtení proběhne buď na devizovém účtu inkasované měny. Všechny operace jsou prováděny v transakci. V případě selhání musí platební systém vrátit jednotlivé účty dotčené transakcí do původního stavu v jakém byly před započetím transakce [1].

V následujících kapitolách budou popsány požadavky na funkci systému z pohledu použitelnosti a bezpečnosti.

2 ZÁKLADNÍ POŽADAVKY NA SYSTÉM

V případě informačního systému pro organizace obchodující mimo finanční sektor, jsou požadavky dány společností, pro kterou je IS vytvářen. Typicky se jedná o základní požadavky na funkčnost, které vychází z obchodního zákona a zákona o ochraně osobních údajů [2]. Organizace musí dbát na to, aby osobní údaje sloužily pouze k účelu, pro který byly pořízeny a musí zamezit jejich zneužití. Platební systém, je určen pro bankovní instituce, které podléhají regulačnímu úřadu. Ten poskytuje směrnice, normy a doporučení a provádí u bankovních institucí audit. Pro Českou republiku je tímto orgánem ČNB, která vydala sadu doporučení pro bezpečnost internetových plateb [3].

2.1 Odolnost proti neoprávněnému přístupu

Jako prvním požadavkem, je zajištění systému proti neoprávněnému přístupu k osobním údajům. Existuje více způsobů, jak útočník zkouší získat neoprávněný přístup [4]. Mezi nejznámější patří:

- Sociální inženýrství
  Jedná se o podvodné techniky zmanipulování uživatele za účelem získání určité informace, za účelem jejího zneužití. Existuje více technik sociálního inženýrství, ale všechny jsou obecně založeny na chybě lidského úsudku.
- Útok hrubou silou
  Útočník se snaží uhádnout autentizační údaje uživatele metodou pokus omyl. Typicky na to používá síť zotročených počítačů (botnet)
- Útok na webovou/mobilní aplikaci
  V tomto případě se útočník snaží najít slabé místo v dané aplikaci, nebo knihovně kterou aplikace využívá.

Typicky k těmto technikám patří pokusy útočníka najít chybu v programu, kterou se snaží využít ve svůj prospěch. Tyto techniky jsou detailně popsány na stránkách OWASP [5], včetně postupů, jak aplikovat opatření k zamezení daných bezpečnostních rizik.

2.2 Odolnost proti DoS, DDoS

Jiným druhem technik, jak poškodit poskytovatele služby je útok, který způsobí výpadek celé služby. V praxi útočník používá opět botnet a velkým množstvím požadavků zahltí aplikační server. V krajním případě tento útok způsobí výpadek poskytované služby. Systémy bankovního sektoru jsou často vyhledávaným cílem. Názorným příkladem je případ z roku 2013, kdy proběhl úspěšný DDoS útok na platební webové portály několika českých bank současně, včetně webového portálu ČNB [6].

2.3 Odolnost proti přetížení systému

Ne vždy je každé přetížení systému způsobeno útokem. Může nastat situace, kdy systém je navržen pro určitý počet uživatelů, ale po letech provozu jejich počet dramaticky vzroste a systém je již není schopen svým výkonem obsloužit. Při návrhu systému je potřeba takovou situaci předvídat a mít připraveno řešení, jak zajistit dodatečné navýšení výkonu.

2.4 Odolnost proti výpadku

U každé služby je pro uživatele nepříjemné, když dojde k jejímu výpadku. U bankovních systémů již může docházet k významným reálným finančním ztrátám a ztrátě důvěryhodnosti u uživatelů. Systém je typicky složen z několika prvků, které jsou vzájemně propojeny. Architekturu je tedy potřeba navrhnout tak, aby byla odolnost proti výpadku jednotlivých částí maximální. S tím ale roste cena výsledného řešení, takže v konečném důsledku se nalezne kompromis mezi pořizovací cenou výsledného řešení a velikostí rizika finanční ztráty v případě výpadku.

2.5 Použitelnost systému pro uživatele

Významným a občas pomíjeným požadavkem je použitelnost pro koncového uživatele (UX). Koncovým uživatelem je zde myšlen zejména klient finanční instituce. Dobrá použitelnost systému se dá popsat těmito základními požadavky:

- Uživatel nalezne informace, které potřebuje
- Potřebnou akci se mu podaří provést v co nejkratším čase a bez velkého přemýšlení
- Potřebnou akci dokáže provést bez chyb

U aplikace se UX nedá přesně změřit, nicméně přibližný odhad se dá získat například počtem chyb, kterých se uživatel při používání aplikace dopustí. Základem dobrého návrhu UX, je

poznat uživatele, kteří aplikaci budou používat. Pokud je to jen trochu možné, tak je dobré provést u vybraného vzorku uživatelů průzkum, ve kterém mohou vyjádřit svá očekávání od nově navrhovaného systému. Pokud uživatel pomáhá již při tvorbě, tak při reálném provozu se mu bude systém snadno ovládat, systém bude pro něj intuitivní a uživatel udělá velmi málo chyb [7]. Důležitou částí je přitom optimální rozvržení prvků na uživatelském rozhraní tak, aby se minimalizoval pohyb očí po obrazovce, a aby umístění odpovídalo zvyklostem čtení v dané lokalitě. V evropských zemích je to tedy čtení zleva doprava a odshora dolů. Detailní popis požadavků a postupů na návrh uživatelského rozhraní je mimo rozsah této práce.

3 TECHNOLOGIE PRO UŽIVATELSKÉ ROZHRANÍ

Pro uživatele je systém nepoužitelný, když s ním nemůže komunikovat. Pro komunikaci je potřebné uživatelské rozhraní (UI). Problémem je, že uživatelé používají různý hardware, operační systémy atp. Poskytovatel služby by uživatele neměl nutit změnit své chování. Cílem poskytovatele tedy je používat takové technologie, aby bylo možné obsloužit co nejširší skupinu uživatelů, a aniž by je to nějak omezovalo.

3.1 Nativní aplikace

Nativní aplikace je napsána a optimalizována pro dané zařízení a operační systém, na kterém je provozována. Zkompilovaný kód je obtížné ze strany útočníka zkoumat. Nevýhodou je, že je nutno pro každou platformu napsat aplikaci znovu, což způsobuje špatnou udržovatelnost a vysoké náklady na vývoj. Nicméně existují prostředky pro vývoj nativních multiplatformních aplikací:

- **Native Script** [8]
  Umožnuje psát aplikace v Javascriptu, které se následně překompilují do nativní aplikace pro danou platformu. Nevýhodou je, že toto řešení je k dispozici pouze Android a IoS a nepodporuje platformu Windows.
- **React Native** [9]
  Umožňuje psát aplikace v Javascriptu a pomocí výkonného frameworku React. Výhodou je, že umožňuje zvolit nativní kód zkompilovat do Javy, nebo Objective C. Do kódu je možné vkládat i časti nativního kódu. Podporuje všechny platformy. Nevýhodou je opět nepodporovaná platforma Windows.
- **Xamarin** [10]
  Multiplatformní řešení umožňuje psát kód v C# a kompilovat do nativní aplikace pro každou z platforem. Existuje dvojí forma přístupu vývoje:
  - **Sdílený projekt**
    Sdílený projekt obsahuje byznys logiku aplikace. Rozhraní je potřeba implementovat na každou platformu zvlášť. Umožňuje efektivněji využít rozhraní hardware dané platformy. Nevýhodou je nutnost psát část kódu odděleně.
  - **Portovatelná knihovna (PCL)**
    Veškerý kód včetně uživatelského rozhraní je implementován v portovatelné knihovně. Funkce, které jsou specifické pro daný hardware nebo platformu

jsou volány přes PCL knihovny, které se doinstalují jak do PCL projektu, tak i do projektů dané platformy. Při kompilaci pak dojde k propojení jednotlivých rozhraní. Výhodou je, že kód aplikace je pro všechny platformy na jednom místě. Vývoj probíhá jen jednou. Nevýhoda je, že kvůli návrhu sdílení jednotlivých obrazovek pro všechny platformy dochází k porušování doporučení pro UX, které je pro každou platformu jiné. Tento problém se dá vyřešit použitím knihoven pro ovládací prvky, které rozdíly přístupu ovládaní na různých platformách řeší. Seznam možností rozšíření ovládacích prvků je uveden na stránkách dodavatele.

3.2 Webové rozhraní

Webové rozhraní je velmi populární, v současné době nejvíce rozšířené a uživatelé jsou na něj zvyklí. Velkou výhodou je, že není nutné na straně klienta instalovat žádný software, protože webový prohlížeč bývá dodáván jako součást operačního systému. Nevýhodou je otevřený zdrojový kód, který má uživatel k dispozici. To umožňuje útočníkovi kód analyzovat a hledat slabá místa v systému. Tento problém se dá částečně vyřešit zamaskováním kódu tak, aby útočníkovi byla analýza kódu velmi ztížena. Není to ale úplná ochrana. Při návrhu webové aplikace je potřeba dbát bezpečnostních doporučení [11] a veškerou byznys logiku umístit na serverovou část aplikace. Na uživatelském rozhraní pak aplikace pouze řeší načítání a odesílání dat mezi klientem a serverem a zobrazování dat. Detailněji bude problematika zabezpečení aplikací popsána v kapitole 5.

Další problematikou je výkon webové aplikace. Při komunikaci klienta se serverem dochází zejména při prvním přístupu k načtení velkého množství souborů, což zpomaluje dobu spuštění aplikace. Částečně tento problém prohlížeče řeší pomocí dočasné paměti.

Pro minimalizaci přenášených dat se dle doporučení [12] používají následující metody:

- Minifikace Javascriptu
- Minifikace CSS

Minifikace spočívá v odstranění veškerých redundantních znaků v kódu jako jsou například: konce řádků, komentáře, dlouhé názvy proměnných a metod jsou přepsány na kratší názvy, typicky 1-2 znaky. Protože webová stránka včetně Javascriptu je čitelný text pro všechny uživatele, tak pro produkční platformu je vhodné provést i obfuskaci kódu. Obfuskace Javascriptu v praxi znamená odstranění veškerých komentářů, přepsání názvů proměnných

a názvů tak, aby se kód pro uživatele maximálně znepřehlednil, ale současně aby byla zachována jeho správná funkce. Minifikace Javascriptového kódu se již dá považovat za částečnou obfuskaci, protože většinu těchto operací kvůli zmenšení velikosti souborů již dělá.

Další metodou, jak zrychlit načítání stránek je použít formu „jednostránkové aplikace“. Pro transformaci vícestránkové aplikace na jednostránkovou je k dispozici několik různých řešení. Mezi nejznámější patří:

- ReactJS [13]
  Jedná se o knihovnu pro vytváření webových komponent. V návrhovém vzoru MVC představuje vrstvu view. Má vysoce optimalizované vykreslovací funkce a jedná se o nejrychlejší framework[1]. Další výhodou je, že jako jediný framework v seznamu zde uvedených, umožňuje odhalit syntaktické chyby ihned při kompilaci. Nevýhodou je, že kvůli optimalizaci vykreslování nelze použít šablonový systém, který by odděloval model od view. To je realizováno programovým kódem, který v sobě obsahuje kousky HTML značek.
- AngularJS [14]
  Jedná se o robustní řešení, které obsahuje plnou implementaci návrhového vzoru MVC. Z pohledu rychlosti aktualizace dat v html je pomalejší než ReactJS. Robustnost systému je zejména pro velké projekty výhodou, nicméně velká knihovna působí výkonnostní problém na mobilních zařízeních.
- KnockoutJS [15]
  Princip práce se šablonami a obousměrným bindingem je podobná jako u AngularJS. Výhodou je velmi jednoduchá implementace a je vhodný zejména pro malé projekty. Ze seznamu zde uvedených frameworků má nejmenší programový kód. Nevýhodou je, že z pohledu výkonu, se jedná o nejpomalejší framework, což by mohlo působit problém při aktualizaci velkého počtu položek na stránce. Zejména pro větší projekty je nevhodný, protože s rostoucí velikostí projektu je složitější udržovatelnost kódu.

[1] Při porovnávání výkonu frameworků se testování typicky provádí pomocí měření rychlosti aktualizace hodnot na velkém seznamu položek na jedné HTML stránce. Délka seznamu položek je v řádu tisíců. Následně se pak porovnává rychlost jednotlivých frameworků.

3.3 Hybridní aplikace

Principem funkce je, že aplikace je napsána v HTML kódu a Javascriptu. Následně se zkompiluje do jednoduchého prohlížeče, který se jako nativní aplikace spustí v mobilním zařízení. Výhodou je velmi efektivní a rychlý vývoj pomocí jednoduchého HTML značkovacího jazyka. Výsledný projekt lze zkompilovat a provozovat pod více platformami. Nevýhodou je poměrně nízká bezpečnost, protože se jedná HTML stránky, které může útočník poměrně jednoduše z aplikace vykopírovat, upravit a pak zase zpětně zkompilovat a spustit. Toto řešení je tedy vhodné pro aplikace, kde bezpečnost nehraje důležitou roli.

K dispozici je několik možností, jak lze hybridní aplikaci vytvořit:

- **PhoneGap** [16]
  Jedná se o framework, který umožňuje kompilaci do nativní aplikace. Neobsahuje prvky uživatelského rozhraní, ty je nutno doinstalovat zvlášť. Například Bootstrap [17].
- **IONIC**
  Podobně jako PhoneGap umožňuje kompilaci do nativní aplikace. Výhodou je, že součástí frameworku je i sada ovládacích prvků včetně přehledné dokumentace.
- **Mobile Angular UI** [18]
  HTML 5 framework, který používá Bootstrap 3 [17] a AngularJS [14]

3.4 Závěr a vyhodnocení

Z výše popsaných technologií lze usoudit, že nejvýhodnější je pro uživatelské rozhraní využít webové rozhraní. Responzivní design bude zajištěn pomocí knihovny Bootstrap [17] a web bude vytvořen jako jednostránková aplikace. Při rozhodování mezi jednotlivými frameworky byl zohledněn zejména problém, že aplikace pro finanční instituce je častým cílem útoků. Z toho důvodu je potřeba pravidelně revidovat vývoj nových prohlížečů, změn standardů v HTML, Javascriptu. Kontrolovat bezpečnostní chyby v knihovnách a provádět jejich pravidelnou aktualizaci, je poměrně náročný úkol. Cílem tohoto projektu je demonstrovat škálování, odolnost proti výpadku a zabezpečení pomocí vícefaktorové autentizace. byl zvolen framework KnockoutJS , aby programový kód se šablonami zůstal co nejvíce jednoduchý. Nízká komplexita frameworku umožní pozdější přechod na jiný šablonový systém, nebo jiný framework. Pro minifikaci byl zvolen projekt Madskristensen-BundlerMinifier [19], který funguje jako rozšíření pro Visual Studio.

Pro autentizaci druhým faktorem pak bude vhodnější využít nativní mobilní aplikaci, protože poskytne vyšší bezpečnost než hybridní aplikace. Pro vývoj byl zvolen framework Xamarin. Forma sdílení kódu byla zvolena PCL, protože umožní lepší udržovatelnost kódu.

4 KOMUNIKAČNÍ TECHNOLOGIE

Základem všech aplikací typu klient-server je jejich vzájemná komunikace. V následujících kapitolách budou popsány výhody a nevýhody jednotlivých protokolů typických pro aplikace komunikujících přes Internetovou síť. Velkým přínosem je možnost propojení heterogenních prostředí, protože komunikace je založena na nezávislých standardech.

4.1 Representational State Transfer (REST)

REST je architektura rozhraní, která je orientována datově. REST tedy určuje, jak se bude přistupovat k datům, nebo stavům aplikace, nikoliv procedurám a funkcím. REST implementuje čtyři základní metody, které jsou známé pod označením CRUD, což jsou funkce vytvoření, načtení, úpravu a odstranění objektu. Ve spojení s Javascript Object Notation (JSON) se stává standardem pro API webových služeb. Důvodem je hlavně nativní podpora serializace a deserializace přenášených dat v Javascriptu. REST umí přenášet i jiné formáty dat, ale ty už nemají tak velký význam. Ze své podstaty je komunikace jednosměrná. Spojení tedy vždy vyvolává klient zasláním požadavku směrem k serveru. Velkou výhodou tohoto rozhraní je snadná implementace klientské i serverové části. Existuje velké množství knihoven a obecně má REST velkou podporu vývojářské komunity. Nevýhodou je slabě typované rozhraní a implementátor se musí velmi spoléhat na dokumentaci. I tak ale bývá tento problém častým výskytem chyb. Tento problém se dá řešit serializací a deserializací pomocí silně typovaných objektů, které jsou sdílené mezi klientem i serverem. To zajistí mnohem vyšší odolnost před zanesením chyby jak ze strany programátora, tak ze strany případného útočníka.

4.2 Simple Object Access Protocol (SOAP)

SOAP je protokol pro posílání zpráv v XML formátu. Později byl rozšířen o standardy Web Service Description Language (WSDL) a Universal Description and Discovery Integration (UDDI). Ty rozšiřují jeho možnosti a usnadňují jeho použití. WSDL umožňuje při implementaci jednoduše vytvořit proxy třídy, které zajistí typově bezpečný přenos. Stejně jako REST popsaný v předchozí kapitole se jedná o jednostranně iniciované spojení. Server tedy nedokáže klientovi zaslat informace, dokud si o ně klient sám nepožádá. Další nevýhodou je mnohem vyšší náročnost na množství přenášených dat. Velký rozdíl je znát při rychlém posílání velkého množství objektů.

4.3 Network socket

Síťový soket je navázané spojení mezi dvěma aplikacemi pomocí IP technologie. Jedná se o nízko-úrovňový způsob komunikace. Velkou výhodou je obousměrná komunikace v reálném čase, trvalé spojení a velká efektivita z pohledu výkonu. Typicky se to využívá pro komunikaci klienta s SQL serverem. V podobě čistě soketového spojení je ale pro webové aplikace nepoužitelný. Nicméně webový standard HTML5 s sebou přinesl implementaci web soketu. Tím se otevřela možnost rozšíření této technologie do Internetu.

Webový soket umožňuje vytvořit obousměrné spojení mezi klientem a serverem. Umožňuje i zabezpečenou komunikaci přes Secure Socket Layer (SSL). Nabízí pouze tři události:

- Spojení otevřeno
- Příchozí zpráva
- Spojení zavřeno

Zpráva se odesílá příjemci v textovém formátu. Je to čistý text a struktura není dána standardem. Nevýhodou zatím je pouze to, že jako nová technologie byla implementována do prohlížečů od roku 2010 a zatím stále ještě existuje skupina uživatelů, kteří mají webové prohlížeče, které tuto technologii neumí. Tento problém se dá vyřešit v aplikaci pomocí hybridního spojení. Tedy klient a server se při úvodním navázání komunikace (handshake) dohodnou na nejlepším možném komunikačním kanálu a na tom začnou komunikovat.

4.4 Push Notification

Push notifikace je služba, která umožňuje pomocí jedné události odeslat zprávu na velké množství připojených klientů. Zásadním rozdílem proti předchozím technologiím je, že se jedná o asynchronní spojení. Doručení zprávy klientovi není garantované, odesílatel nedostane potvrzení o doručení a není garantován ani čas za který bude zpráva příjemci doručena. Dalším omezením je velikost zprávy. Většina push notifikačních služeb má velikost zprávy omezenu na maximálně několik set bytů. Službu není technicky možné využít na rychlé zasílání zpráv, kdy je vyžadováno potvrzení o doručení. Výhodou je nativní implementace push notifikační služby do mobilních telefonů. To umožňuje zasílat zprávy na mobilní telefony i když na nich zrovna není aplikace spuštěna a mobil má uzamčeno uživatelské rozhraní. Nevýhodou je, že každý mobilní operační systém používá jinou push notifikační službu a odesílatel dopředu neví, na jaké rozhraní má zprávu poslat. Řešením je využít

služby, která zajistí komunikaci se všemi notifikačními subsystémy pro nejrozšířenější mobilní platformy. Mobilní klient při registraci sdělí systému svůj operační systém a kam mají být přeposílány zprávy určené pro něj. Odesílatel zprávy už pak nemusí řešit kam zprávu pošle.

4.5 Queue

Queue, neboli fronta, umožňuje podobně jako push notifikace zasílání asynchronních zpráv mezi jednotlivými aplikacemi. Proti push notifikacím lze jednotlivým zprávám nastavit dobu platnosti a po tuto dobu je možné na serveru ověřit, zda byla zpráva doručena. Fronta pracuje na principu FIFO, tedy první zpráva uložena do fronty je jako první nabídnuta k přečtení. Aplikace, která zprávu z fronty přečte ji může upravit, odstranit, nebo přečíst další zprávu dle pořadí ve frontě. Velkou výhodou fronty je, že umožňuje škálovat výkon aplikací pomocí paralelního přístupu ke zpracovávaným požadavkům. Příkladem může být například několik služeb, které ve velkém množství generují požadavky na zpracování. Pokud server, který požadavky zpracovává již nestíhá požadavky řešit, tak stačí přidat několik dalších serverů se stejnou funkcí jako má původní server. Nevýhodou je asynchronní zpracování bez potvrzení přijetí.

4.6 Závěr a zhodnocení jednotlivých technologií

Plánovaný systém je poměrně komplexní, a proto není možné jednoznačně říci, že jedno konkrétní komunikační řešení je vhodné pro celý systém.

Pro vícefaktorovou autentizaci bude vhodné využít push notifikace, protože umožní zasílat zprávy na mobilní telefony i v případě, kdy aplikace není na mobilním zařízení spuštěna. Bude potřeba mít na zřeteli, že push notifikace bude nutné vyřešit pro několik různých platforem výrobců. Aplikace bude potřebovat pro svou funkci zasílat zprávy do systému vícefaktorové autentizace. Nejvhodnější bude použít nejjednodušší systém webových služeb REST, který umožní systému IPS efektivní monitorování provozu. Další výhodou je, že REST služba bude mít minimální nároky na výkon serveru.

Platební systém musí být odolný vůči výpadku služby. Musí být zajištěno, že všechny požadavky z platebních portálů budou obslouženy, přičemž není nutné mít zpracování v synchronním režimu. Dále by bylo výhodné pro platební systém mít rovněž horizontální škálování, které umožní rozdělení zátěže. K tomuto účelu bude nejvhodnější využít frontu. Ta umožní rovněž systému se vyrovnat s případným selháním platebního systému.

Platební portál bude nejvíce zatíženou částí systému, protože k němu budou přímo přistupovat uživatelé. Rovněž bude potřeba, aby komunikační kanál umožnil zasílat data ze serveru na klienta v co nejvyšší možné frekvenci. Uživatel může být připojen k portálu pomalým připojením a může mít na svém zařízení různou verzi operačního systému. Nejvhodnějším řešením bude využít webového klienta, který je multiplatformní a pro efektivní obousměrnou komunikaci bude vhodné využít webový soket se zachováním možnosti zpětné kompatibility se staršími zařízeními.

5 TYPY ÚTOKŮ A OCHRANA PROTI NIM

V této kapitole jsou navržena řešení ochrany proti jednotlivým typům útoků. Jsou zde uvedeny jen ty nejvýznamnější, protože ostatní okrajové typy jsou většinou využívají podobných technik a navržená ochrana tedy pokryje více typů současně. Ucelený seznam je uveden na webu OWASP [5].

5.1 DoS, DDoS

Na útok DDoS, který je v podstatě rozšířením DoS, neexistuje 100% funkční ochrana. Když se někdo rozhodne, že službu chce zastavit a má k dispozici dostatek prostředků, tak prostě službu zastaví. V případě DoS útoku, je útok veden z jednoho místa. Útočník typicky posílá po síti velké množství požadavků, kterými se snaží službu přetížit a tím vyřadit z provozu. Útok může být veden na všech vrstvách ISO/OSI modelu, a zahlcen může být i prvek, který je cíli předřazen, třeba firewall. Více je tato technika popsána na serveru OWASP [20]. Proti DoS útoku se dá účinně bránit. Pro webové služby je to například pomocí webového aplikačního firewallu, který umožňuje odfiltrovat škodlivý provoz a propustit k serveru jen legitimní uživatele. Pro menší objemy, řádově několik desítek Gbps, je možné využití in-line řešení, kdy zařízení stojí v cestě a data jím přímo prochází. V normálním stavu zařízení toky monitoruje a v případě útoků se snaží nasadit statistické filtry a škodlivé toky odstranit. Takovou službu je možné si objednat v datovém centru, kde je aplikační server hostován. Vetší problém je u DDoS útoků, které jsou vedeny z více míst a mnohem vyššími datovými toky. In-line řešení nezvládne takový tok a dojde k přetížení přístupové linky do datového centra. Cílem je zastavit útok co nejblíže k útočníkovi. Pro tyto případy je potřeba útoky odfiltrovat ještě před tím, než přijdou do datového centra. Typicky je to řešeno tak, že provoz je přesměrován už na úrovni hraničních routerů poskytovatele internetu (ISP) do tzv. scrubbing centra, kde je síťový provoz čištěn a dále je již přeposlán pouze regulérní provoz.

Výše uvedená problematika ukazuje, že ochrana proti DDoS na úrovni datového centra je neefektivní a nejvhodnější je využít služeb ochrany poskytovaných jednotlivými ISP na páteřních routerech.

5.2 Man in the Middle

Princip spočívá v tom, že útočník získá kontrolu nad routerem, který je v cestě mezi klientem a aplikačním serverem a je schopen pak spojení odposlouchávat, případně za běhu i měnit. Jednoduchou ochranou je použití protokolu HTTPS, který obsah přenášených zpráv šifruje.

Aby ochrana byla účinná, tak je potřeba zajistit, aby certifikát serveru byl podepsán důvěryhodnou certifikační autoritou a současně aby uživatel byl dostatečně seznámen s tím, jaký certifikát a od jaké certifikační autority je ten správný.

5.3 Trojan Horse

Trojský kůň je program, který se útočníkovi podařilo spustit na zařízení uživatele. Existuje více druhů těchto programů. V nejhorším případě získá útočník plnou kontrolu nad počítačem uživatele a dokáže číst i měnit uživatelem zadávané údaje. Aktivní ochranu ze strany poskytovatele služby lze implementovat pouze částečně, a to v podobě systému prediktivního chování. Systém monitoruje neobvyklé aktivity klienta a v případě, že klient začne vykazovat v aplikaci neobvyklé chování tak je možné takové akci zabránit. Takový systém je ale značně nespolehlivý, protože uživatel by měl mít možnost se chovat podle vlastních potřeb, a ne podle toho co predikuje systém, že uživatel udělá. Efektivnější je využít více faktorovou autentizaci. Ta umožní eliminovat riziko, že systém přijme od uživatel požadavek, který byl upraven útočníkem. Jako podpůrné opatření je edukace uživatelů ve formě doporučení, aby používali antivirovou ochranu.

5.4 Phishing

Phishing patří do skupiny podvodných útoků, kdy se útočník vydává za někoho jiného a snaží se z oběti vylákat citlivé osobní údaje. Typicky se jedná o čísla bankovních účtů, rodná čísla, hesla. Šíří se podvodnými emaily, nebo přesměrováním na falešnou stránku. Riziko je potřeba eliminovat opět edukací uživatelů ohledně kontroly validity certifikátů, kterými jsou data podepsána. Řešení je tedy obdobné jako kapitole 5.2, tedy využití certifikátu od důvěryhodné certifikační autority.

5.5 Pharming

Princip tohoto útoku je založen na napadení DNS a následném přepsání IP adresy cílového serveru. Nová stránka pak často vypadá k nerozeznání od původní. Když pak oklamaný uživatel zadá do této stránky přihlašovací údaje, tak je tím vlastně předá útočníkovi. Ochranu proti pharmingu je možné provést pouze edukací uživatelů, aby byli velmi obezřetní, na jaké stránky se přihlašují, a jestli se neděje něco nestandardního. Z pohledu uživatele toho nic víc udělat nejde. Poskytovatel kromě toho, že si zabezpečí data a infrastrukturu, aby nedošlo ke zneužití, tak ještě může zvýšit bezpečnost uživatele zavedením vícefaktorové autentizace.

Útok nemusí být veden jen na DNS, ale také na službu Windows, která překlad adres zajišťuje. V tomto případě je možné riziko minimalizovat kvalitním antivirovým programem, který kromě ochrany proti malware, bude ještě kontrolovat stav a hlídat změny u kritických systémových souborů, které bývají k pharmingu zneužívány.

5.6 Cross-Site Scripting

Útočník se v tomto případě snaží využít chybného návrhu kódu tak, aby do stránky podvrhl vlastní programový kód. Ochrana proti útoku musí být již přímo v návrhu aplikace. Do základních pravidel ochrany patří, že každý vstup od uživatele bude považován za nedůvěryhodný a proběhne vždy jeho validace na povolené vstupní znaky. Například když je ve formuláři očekávána číselná hodnota, tak je povoleno vkládání pouze pro numerické znaky.

5.7 SQL Injection

Tato technika umožňuje útočníkovi nežádoucím způsobem modifikovat data. Ochrana je podobná jako v předchozí kapitole. Navíc je pro komunikaci aplikace s SQL serverem využito standardních knihoven a uživatelem zadávaná data předávat do SQL parametricky, tím dochází k efektivní validaci vstupních dat [21].

5.8 Brute Force

Brute Force, neboli útok hrubou silou se od DDoS útoku popsaného v kapitole 5.1 liší tím, že síťový provoz je ve své podstatě v pořádku. Pouze je potřeba odfiltrovat nežádoucí aktivity, jako jsou například opakované neúspěšné pokusy o autentizaci. Jednoduchou základní ochranu je možné zajistit implementací přímo v aplikační části, kdy uživatelský účet po několika neúspěšných pokusech bude dočasně uzamčen. Další možností je použít systém CAPTCHA[2]. Sofistikovanější ochranou je využití specializovaných zařízení, jako je například webový aplikační filtr. Systém může být dále vylepšen tím, že aplikace bude aktivně filtru poskytovat data o podezřelém chování uživatelů. Zařízení pak vyhodnocuje podle nastavených pravidel četnost a řídí pro jednotlivé uživatele omezení přístupu ke službě.

[2] Jedná se o Turingův test, který se používá pro odlišení počítačů od lidí

II. PRAKTICKÁ ČÁST

6 UKÁZKOVÁ APLIKACE

V následujících kapitolách byla na základě požadavků navržena aplikace. Byly popsány funkce jednotlivých částí systému i jejich vzájemná interakce. Bylo navrženo řešení vícefaktorové autentizace i řešení podpory pro horizontální škálování. Následně byla vybrána infrastruktura, na které bylo možné efektivně vyvinout a testovat prototyp ukázkové aplikace. V ukázkové aplikaci byla provedena implementace platebního portálu, který umožňuje přihlášení uživatele pomocí dvoufaktorové autentizace. Byl vytvořen systém vícefaktorové autentizace, který odesílá požadavky uživateli pomocí push notifikační služby a přijímá zpět od uživatele potvrzení, nebo zamítnutí přístupu do systému. Jako uživatelské rozhraní pro autentizaci druhým faktorem byla vytvořena multiplatformní nativní mobilní aplikace. Autentizace pomocí biometrie implementována nebyla, protože v době vývoje aplikace by bylo finančně náročné zajistit potřebný hardware. Pro streamování kurzů byla vyvinuta služba, která aktualizuje data v reálném čase na platebním portálu. Platební portál umožňuje uživateli zobrazit přehled bankovních účtů, aktuální přehled kurzů měnových párů a možnost objednat směnu.

6.1 Požadavky

Požadavky na systém jsou rozděleny do funkčních a nefunkčních. Do funkčních byly vypsány požadavky na funkcionalitu systému vyplývající z požadavků popsaných v kapitole 1. Mezi nefunkční požadavky patří výkon, škálovatelnost, spolehlivost, bezpečnost, rozšiřitelnost a udržitelnost. Z těchto požadavků byly upřednostněny ty, které zajistí výkon systému, spolehlivost a bezpečnost.

Funkční požadavky

- Uživatel musí mít možnost se do systému přihlásit vícefaktorové
- Uživatel musí mít možnost výběru metody ověření druhým faktorem
- Uživatel si může prohlédnout historií transakcí
- Uživatel si může prohlédnout aktuální stav zůstatků na účtech
- Uživatel má k dispozici aktuální přehled vybraných kurzů
- Uživatel může zadat transakci na nákup, nebo prodej měnového páru

Nefunkční požadavky

- Systém musí umožnit navyšování výkonu horizontálním škálováním

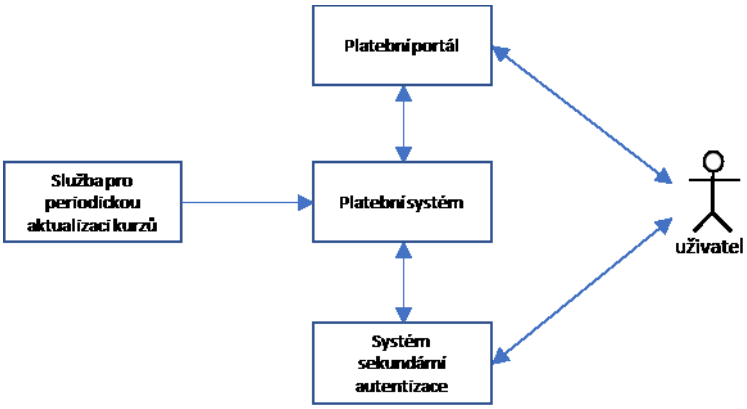
- Systém musí být odolný proti výpadku
- Systém musí být odolný proti opakovanému zadávání neplatných přístupových údajů
- Systém musí být odolný proti odcizení přihlášeného sezení
- Systém musí být odolný proti útoku DDoS

6.2 Schéma a popis jednotlivých částí systému

Na základě upřesněných požadavků, byli v systému identifikováni následující aktéři:

- Uživatel
- Platební portál
- Systém sekundární autentizace
- Služba pro periodickou aktualizaci kurzů
- Platební systém

Uživatel je klient platební instituce, který zadává požadavky do platebního portálu. Platební portál požadavky klienta předává do platebního systému. Platební systém zajišťuje ověření identity uživatele a zpracování transakcí a transfer peněz mezi účty klienta a finanční instituce. Služba pro periodickou aktualizaci kurzů plní data do platebního systému, který je pak následně distribuuje uživateli.

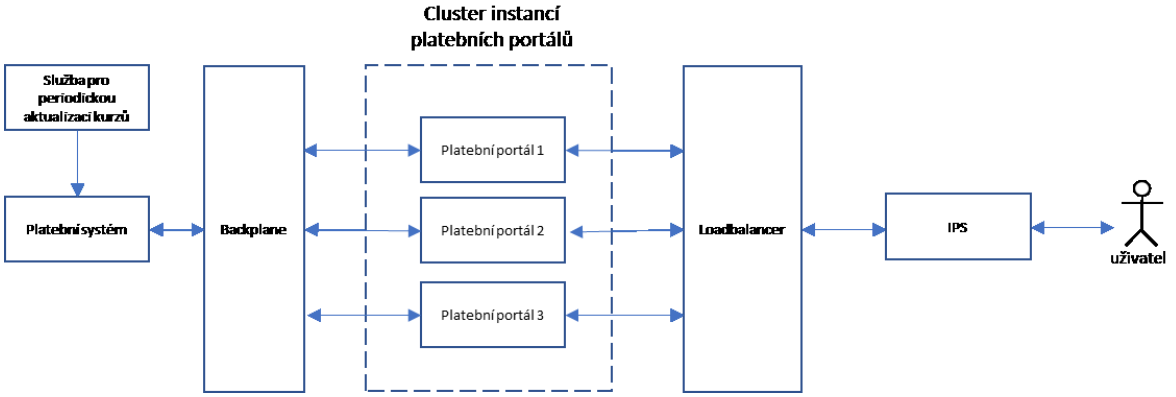


Obrázek 1. Základní schéma systému

Návrh systému předpokládá, že platební instituce, která bude systém používat již má stávající platební systém i službu pro načítání aktuálních kurzů. Tyto systémy budou v návrhu implementovány pouze v ukázkovém režimu, bez napojení na reálné služby. Budou mít však k dispozici rozhraní, pomocí kterého půjde jednoduše jednotlivé komponenty s minimálním úsilím vyměnit a napojit na vlastní systémy.

Systémy musí rovněž podporovat možnost horizontálního škálování a odolnost proti výpadku. Platební systém, platební portál i služba pro periodickou aktualizaci kurzů budou implementovány v režimu vysoké dostupnosti.

Naprostá většina požadavků od uživatele bude směrována na platební portál. Nejvíce požadavků se předpokládá na aktuální stav kurzu. Minimální množství požadavků pak bude na uskutečnění transakce, případně na zobrazení stavu transakcí, jejich historii a aktuální stav účtů. Pro zajištění potřebné funkcionality je potřeba mít v systému zapojeno paralelně několik platebních portálů. Uživatel pak bude na platební portál přistupovat přes loadbalancer, který bude zajišťovat rovnoměrné vyvažování zátěže klientů na platební portál. Loadbalancer periodicky zjišťuje stav provozuschopnosti a zátěže na všech instancích platebních portálů. V případě výpadku platebního portálu přesměruje automaticky a rovnoměrně připojené uživatele na zbývající instance platebních portálů. V případě přetížení je možné rozšířit systém o automatické horizontální škálování („auto-scaling“). To je služba, která bude zajišťovat spuštění dalších instancí platebních portálů, pokud dojde k přetížení stávajících instancí. Pokud zátěž klesne pod stanovenou úroveň, tak je postupně bude počet aktivních instancí snižovat. Dalším nutným prvkem je „backplane“. Tento prvek zajišťuje synchronizaci komunikace mezi platebním systémem a všemi instancemi platebních portálů tak, aby byly platební portály mezi sebou zaměnitelné a aby poskytovaly jak identické služby, tak i identické informace. Pokud se například uživatel připojí na jednu instanci platebního portálu a loadbalancer z důvodu výpadku nebo přetížení provoz přesměruje na jinou instanci, tak uživatel nepozná rozdíl. Mezi uživatele a loadbalancer je předřazen systém pro prevenci útoku (IPS), který umožní odfiltrovat škodlivý síťový provoz. Detail znázorňuje Obrázek 2.



Obrázek 2. Detail horizontálního škálování platebního portálu

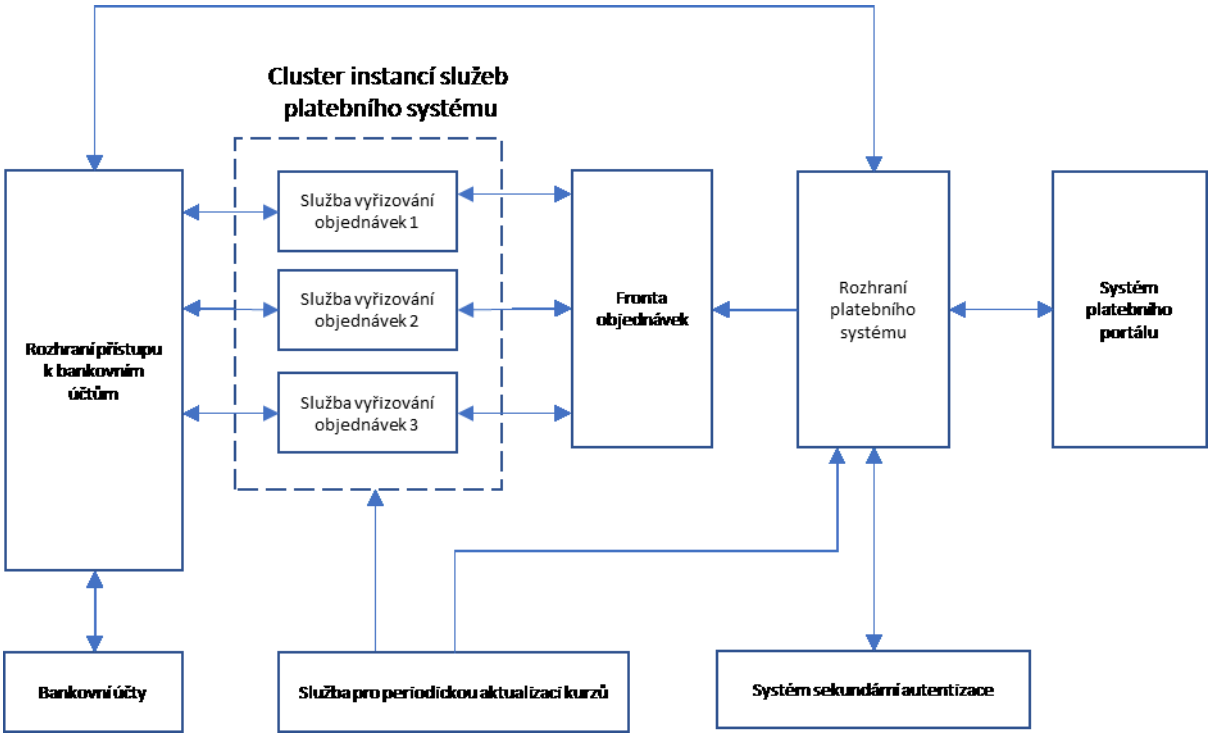
Platební systém přebírá požadavky z platebního portálu a skládá se ze třech částí:

- Rozhraní platebního systému

- Fronta objednávek
- Služba vyřizování objednávek
- Rozhraní přístupu k bankovním účtům

Rozhraní platebního systému slouží ke komunikaci platebního systému s platebním portálem. Vyřizuje požadavky na seznam účtů, jejich zůstatky a historii transakcí. V případě těchto operací komunikuje přímo s rozhraním přístupu k bankovním účtům. Řeší rovněž požadavek na přihlášení uživatele do platebního portálu, včetně komunikace se systémem sekundární autentizace.

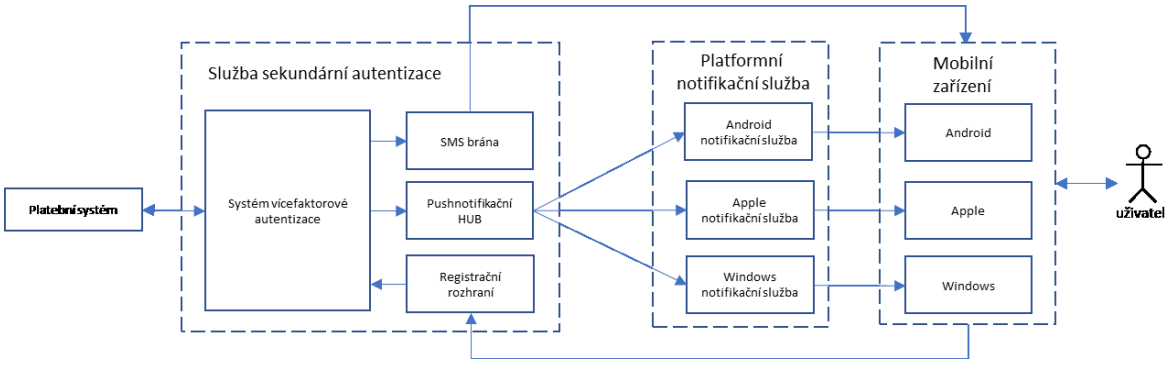
Fronta slouží pro evidenci objednávek a jejich stavů. Je určena pro vyrovnávání zátěže pro případ, že by služba vyřizování objednávek selhala. Služba pro vyřizování objednávek zpracovává objednávky z fronty a během zpracování dochází k zamykání záznamu objednávky ve frontě a zámek je uvolněn, jakmile je objednávka vyřešena. Během vyřizování objednávky běží operace služby v transakci. Tedy pokud v některém kroku služba selže, tak se vše vrátí do původního stavu před započetím transakce.



Obrázek 3. Detail horizontálního škálování platebního systému

Požadavek na vícefaktorovou autentizaci bude řešen pomocí push notifikací a SMS. Zasílání autentizačních kódů přes SMS zprávy je forma zastaralá, neekonomická a uživatelsky nepřívětivá, nicméně je ji potřeba zachovat kvůli zpětné kompatibilitě s klienty, kteří používají starší typy mobilních zařízení, nebo kteří nebudou ochotni si nainstalovat do svého mobilního zařízení novou aplikaci.

Při požadavku systému na autentizaci dalším faktorem zvolí uživatel druh druhého ověření. Pokud zvolí SMS kód, tak bude vygenerován kód s časově omezenou platností a odeslán na telefonní číslo mobilního zařízení, které je svázáno s registračními údaji, pod kterými se uživatel přihlásil při prvním faktoru autentizace. SMS kód uživatel opíše do formuláře a odešle jej. Platební systém ověří platnost kódu a následně prohlásí požadavek uživatele za ověřený dvěma faktory. Pokud uživatel zvolí jako další faktor push notifikaci, tak se očekává, že má na svém mobilu nainstalovánu autentizační aplikaci. Aplikace bude multiplatformní. To tedy umožní instalaci na zařízení na platformě Android, Apple i Windows. Problémem ale je, že každá z platforem má vlastní systém pro push notifikace. Formát zpráv, které telefony různých platforem přijímají, se rovněž liší. Bude potřeba vytvořit notifikační hub, který zajistí platformě nezávislé zasílání zpráv na notifikační služby jednotlivých platforem. Aplikace při spuštění si vyžádá autentizaci. Uživatelem zadané údaje jsou předány na registrační rozhraní. Systém provede ověření, pokud je vše v pořádku, tak je ID zařízení svázáno s uživatelem. Uloží se rovněž informace o typu zařízení a vybere se vhodný formát notifikačních zpráv v závislosti na platformě zařízení. Dále je na platformní notifikační službě provedena registrace zařízení pro příjem push notifikací. Detail řešení znázorňuje Obrázek 4



Obrázek 4. Detail systému pro vícefaktorovou autentizaci

6.3 Případy užití

V této kapitole budou popsány scénáře případů užití. Jedná se pouze o minimalistické řešení základních požadavků. V praktickém použití bude potřeba dopracovat další scénáře, které více zvýší užitnou hodnotu a komfort pro uživatele.

6.3.1 UC1 – Přihlášení uživatele do systému pomocí mobilní aplikace

Uživateli je umožněno přihlášení do systému. Vícefaktorová autentizace je provedena push notifikací do mobilní aplikace.

Aktéři

- Uživatel
- Platební portál
- Platební systém
- Systém sekundární autentizace
- Mobilní aplikace

Podmínky pro spuštění

- Uživatel má zaregistrován přístupový účet, který je aktivní, není zablokovaný a nepřihlášený uživatel vstoupil na úvodní stránku platebního portálu.

Základní tok

1. IPS provede kontrolu přístupu.
2. Systém zobrazí formulář pro zadání přihlašovacích údajů: číslo smlouvy a heslo.
3. POKUD je počítadlo neplatných pokusů o přihlášení větší než 2.
   3.1. Systém zobrazí výzvu pro vyplnění CAPTCHA kódu. Složitost kódu se zvyšuje s počtem neplatných pokusů.
4. Uživatel vyplní do přihlašovacího formuláře ID smlouvy, heslo a zvolí typ druhé autentizační metody.
5. Systém provede validaci dat od uživatele.
6. POKUD uživatel vybral autentizaci SMS, tak systém vygeneruje jednorázový kód a pošle jej na telefonní číslo svázané s autentizačními údaji.
   6.1. Systém zobrazí výzvu pro zadání SMS kódu a tlačítko „zaslat nový SMS kód“.
   6.2. POKUD uživatel zvolí akci „zaslat nový SMS kód“.
      6.2.1. tok pokračuje krokem č.5 základního toku.

6.3. Uživatel vepíše SMS kód do formuláře.

6.4. Platební systém provede ověření SMS kódu.

7. POKUD uživatel vybral autentizaci „mobilní aplikace“
   7.1. Systém odešle push notifikaci do push notifikačního hubu, který zprávu doručí do mobilního zařízení.
   7.2. Mobilní aplikace zobrazí uživateli výzvu k potvrzení autentizace druhým faktorem.
   7.3. Uživatel potvrdí žádost.
   7.4. Mobilní aplikace odešle potvrzení do systému sekundární autentizace.
   7.5. Systém sekundární autentizace předá informaci platebnímu systému o úspěšně provedené autentizaci.
8. Platební systém provede přihlášení uživatele. Vytvoří záznam o přihlášeném sezení a uzamkne sezení k dané IP adrese klienta, ze které se klient přihlásil. Vynuluje počítadlo neplatných přihlášení.
9. Platební portál zobrazí klientovi přehled stavu jednotlivých účtů klienta.

Alternativní tok 1

1.1 Pokud je identifikační údaj uživatele na blacklistu, tak systém zobrazí uživateli informaci „služba je dočasně nedostupná“.

Alternativní tok 2

5.1 Pokud uživatel zadal neplatný vstup, tak se zvýší počítadlo neplatných pokusů.

5.2 Doba platnosti hodnoty počítadla je nastavena na 10minut.

5.3 Tok pokračuje na kroku č.1 základního toku.

Alternativní tok 3

6.4.1 Pokud není SMS kód platný, tak se zvýší počet neplatných pokusů u daného SMS kódu.

6.4.2 Pokud je počet neplatných pokusů u SMS kódu roven 3, tak se SMS kód zneplatní a je zobrazena notifikace uživateli, aby si vyžádal zaslání nového kódu.

6.4.3 Tok pokračuje na kroku č. 6.2 základního toku.

6.3.2 UC2 – Zobrazení aktuálního stavu kurzů uživateli

Uživateli je zobrazen aktuální stav kurzů vybraných měn.

Aktéři

- Uživatel
- Platební portál
- Platební systém

Podmínky pro spuštění

- Uživatel je přihlášen na platebním portálu.

Základní tok

1. Uživatel zvolí zobrazení přehledu kurzu měnových párů.
2. Platební portál si načte z platebního systému aktuální přehled kurzů.
3. Platební systém převezme aktuální přehled kurzů z mezipaměti.
4. Platební portál zobrazí aktuální přehled kurzů a očekává událost aktualizace kurzu.
5. INCLUDE UC3 – periodická aktualizace kurzů ze zdroje na platební portál.

6.3.3 UC3 – Periodická aktualizace kurzů ze zdroje na platební portál

Na platební portál je doručena informace o aktualizaci kurzu měnového páru.

Aktéři

- Systém pro periodickou aktualizaci kurzů
- Platební systém
- Platební portál

Podmínky pro spuštění

- Uživatel je přihlášen na platební portál a ve zdroji dat kurzů došlo ke změně.

Základní tok

1. Systém pro periodickou aktualizaci předá událost do platebního systému.
2. Platební systém uloží změnu do tabulky mezipaměti kurzů.
3. Systém pro periodickou aktualizaci nastaví stav systému na „OK“.
4. POKUD došlo ke změně stavu Chyba⇨OK, tak uloží informaci do logu a pošle informaci administrátorovi, že služba je opět v provozu. Platební systém povolí uzavírání nových transakcí. KONEC POKUD

5. Platební systém předa událost pomocí backplane do všech aktivních instancí platebních portálu.
6. Platební portál zaktualizuje zobrazované údaje kurzů měnových párů.

Alternativní tok 1

3.1. Pokud se platebnímu systému nepodaří změnu zapsat do mezipaměti měnových kurzů, tak oznámí neúspěch službě pro periodickou aktualizaci kurzů.

3.2. POKUD došlo ke změně stavu služby z OK⇨Chyba, tak platební systém uloží informaci do logu a pošle informaci administrátorovi, že došlo k výpadku služby.

Platební systém pozastaví uzavírání nových transakcí. KONEC POKUD

3.3. Tok pokračuje krokem 1 základního toku.

Výjimka

Přerušení funkce z důvodu selhání služby periodické aktualizace kurzů

1. Systém se pokouší provést automatické znovu spuštění služby.
2. Po opětovném spuštění po selhání pokračuje tok krokem 1 základního toku.

6.3.4 UC4 – Zobrazení historie transakcí

Uživateli je zobrazena historie transakcí

Aktéři

- Uživatel
- Platební portál
- Platební systém

Podmínky pro spuštění

- Uživatel je přihlášen na platebním portálu.

Základní tok

1. Uživatel zvolí zobrazení přehledu historie transakcí.
2. Platební portál si načte z platebního systému aktuální přehled historie transakcí.
3. Platební portál zobrazí aktuální přehled kurzů a očekává z platebního systému událost aktualizace historie transakcí.
4. POKUD došlo k události změna stavu historie transakcí.

4.1. Platební portál provede aktualizaci stavu transakcí.

4.2. Tok pokračuje na kroku č.4 základního toku.

6.3.5 UC5 – Zobrazení stavu účtů uživateli

Uživateli je zobrazen aktuální stav účtů uživatele

Aktéři

- Uživatel
- Platební portál
- Platební systém

Podmínky pro spuštění

- Uživatel je přihlášen na platebním portálu.

Základní tok

1. Uživatel zvolí zobrazení přehledu účtů.
2. Platební portál si načte z platebního systému aktuální přehled účtů a jejich stav zůstatku a disponibilní částku.
3. Platební portál zobrazí aktuální stav účtů a očekává z platebního systému událost aktualizace změny stavu účtů.
4. POKUD došlo k události změna stavu účtů.
   4.1. Platební portál provede aktualizaci stavu účtů.
   4.2. Tok pokračuje na kroku č.4 základního toku

6.3.6 UC6 – Vložení nové transakce uživatelem

Uživatel zadá požadavek na nákup/prodej měnového páru

Aktéři

- Uživatel
- Platební portál
- Rozhraní platebního systému
- Služba platebního systému

Podmínky pro spuštění

- Uživatel je přihlášen na platebním portálu a má zobrazenu obrazovku přehled kurzů měnových párů
- Na platebním portálu probíhá periodická aktualizace kurzů měnových párů

Základní tok

1. Uživatel vybere měnový pár a druh transakce.
2. Platební portál zobrazí aktuální disponibilní zůstatek na účtu klienta a formulář pro zadání transakce.
3. Uživatel vyplní do formuláře požadovanou částku, kterou chce koupit
4. Uživatel zvolí způsob vypořádání (D+0 až D+4). Čím je delší doba vypořádání, tím výhodnější kurz platební systém nabídne.
5. Platební portál pravidelně přepočítává potřebnou částku pro uhrazení transakce proti aktuálnímu kurzu a dle zvolené doby vypořádání. (Je to nezávazná orientační informace, kurz se na obrazovce zatím stále mění dle změny kurzovního lístku. Aktualizaci řeší UC3).
6. Uživatel odešle údaje tlačítkem „Rekapitulace“ do rozhraní platebního portálu.
7. Rozhraní platebního portálu vytvoří objednávku, do které si z tabulky mezipaměti uloží aktuální kurz, dobu vypořádání a přepočítá částky dle aktuálních údajů.
8. POKUD je potřebná částka na úhradu vyšší než disponibilní zůstatek.
   8.1. Platební portál zobrazí varování, že obchod nelze uzavřít a objednávka je zrušena.
   8.2. Tok pokračuje krokem č.2 základního toku.
9. Platební portál zobrazí rekapitulaci objednávky. Kurz je v tuto dobu již pro tuto objednávku neměnný. Na stránce rekapitulace běží časový odpočet platnosti nabídky.
10. POKUD je obrazovka s rekapitulací více jak 10 vteřin bez potvrzení nabídky
    10.1. Rozhraní platebního systému zruší nabídku.
    10.2. Platební portál zobrazí informaci o skončení platnosti nabídky.
    10.3. Tok pokračuje krokem č.2 základního toku.
11. Uživatel potvrdí objednávku tlačítkem „Potvrdit“.
12. Platební portál předá požadavek objednávky do rozhraní platebního systému.
13. Rozhraní platebního systém uloží objednávku do fronty.
14. Služba platebního systému načte z fronty novou transakci a přepne ji do stavu „právě zpracovávaná“. Tím je uzamčena pro případné duplicitní zpracování paralelním procesem.
15. Služba platebního systému načte aktuální disponibilní zůstatek účtu klienta.

16. POKUD je potřebná částka na úhradu vyšší než disponibilní zůstatek.
    16.1. Služba platebního systému zamítne objednávku. Nastaví ji do stavu zrušeno. Odstraní ji z fronty.
    16.2. Služba platebního systému odešle oznámení o zamítnutí do platebního portálu.
    16.3. Platební portál zobrazí informaci, že objednávka byla zamítnuta z důvodu nedostatečného krytí na účtu klienta.
    16.4. Tok pokračuje krokem č.2 základního toku.
17. Platební systém zablokuje danou částku na disponibilním zůstatku na účtu klienta.
18. Platební systém provede ověření na likviditu požadavku na účtu platební instituce.
19. POKUD platební systém zjistí nedostatečné zajištění na straně finanční instituce.
    19.1. Platební systém vrátí disponibilní zůstatek na účtu klienta do původního stavu.
    19.2. Platební systém zruší transakci, odstraní objednávku z fronty.
    19.3. Platební systém pošle zprávu do platebního portálu o zamítnutí transakce.
    19.4. Platební portál zobrazí informaci, že objednávka byla zamítnuta.
    19.5. Tok pokračuje krokem č.2 základního toku.
20. Platební systém odečte danou částku z disponibilního zůstatku na účtu platební instituce.
21. Platební systém odečte požadovanou částku v měně A z účtu klienta a přičte ji na účet platební instituce.
22. Platební systém přičte na účet klienta požadovanou částku v měně B na účet klienta a odečte ji z účtu platební instituce.
23. Platební systém uzavře transakci a uloží záznam do historie transakcí.
24. Platební systém pošle událost do platebního portálu o aktualizaci stavu zůstatků na účtech klienta.
25. Platební systém pošle událost do platebního portálu o aktualizaci stavu historie transakcí.
26. Platební portál zobrazí informaci o uzavření transakce.

Výjimka

Přerušení transakce z důvodu selhání platebního systému při převodu peněz.

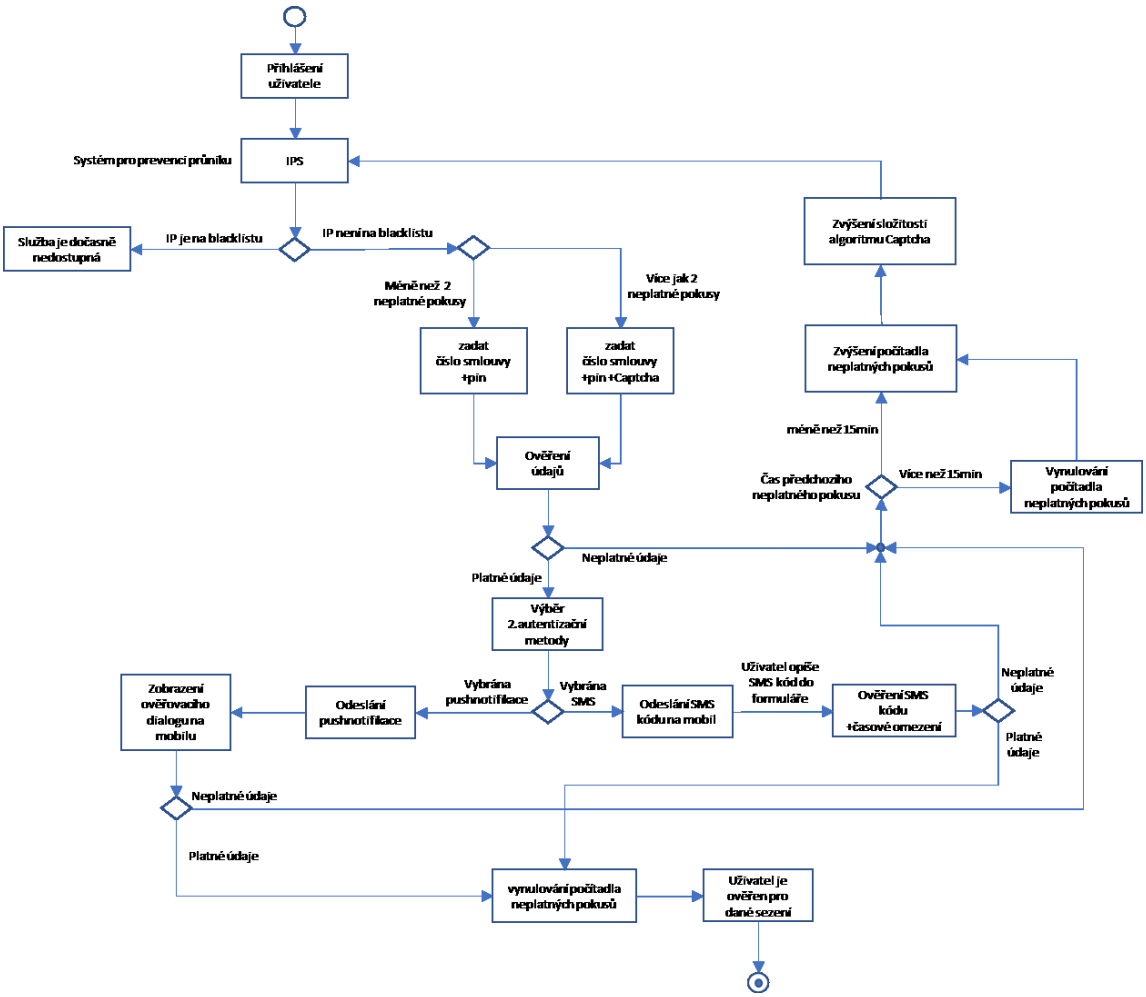
3. Platební systém vrátí stav účtů klienta do původního stavu
4. Platební systém vrátí stav účtů platební instituce do původního stavu
5. Platební systém objednávku ve frontě uvolní pro opakování zpracování.
6. Tok pokračuje krokem 14 základního toku.

6.4 Diagramy aktivit

V následujících podkapitolách jsou detailněji znázorněny vybrané procesy pomocí diagramu aktivit. Ty umožní lépe znázornit procedurální logiku, byznys procesy a pracovních postupy popsané v uživatelských scénářích. Plavecké dráhy, které oddělují zodpovědnosti účastníků za jednotlivé aktivity jsou vynechány. Bez nich je model v těchto případech přehlednější a odpovědnosti aktérů již byly popsány v předchozí kapitole 6.3

6.4.1 Přihlášení do systému

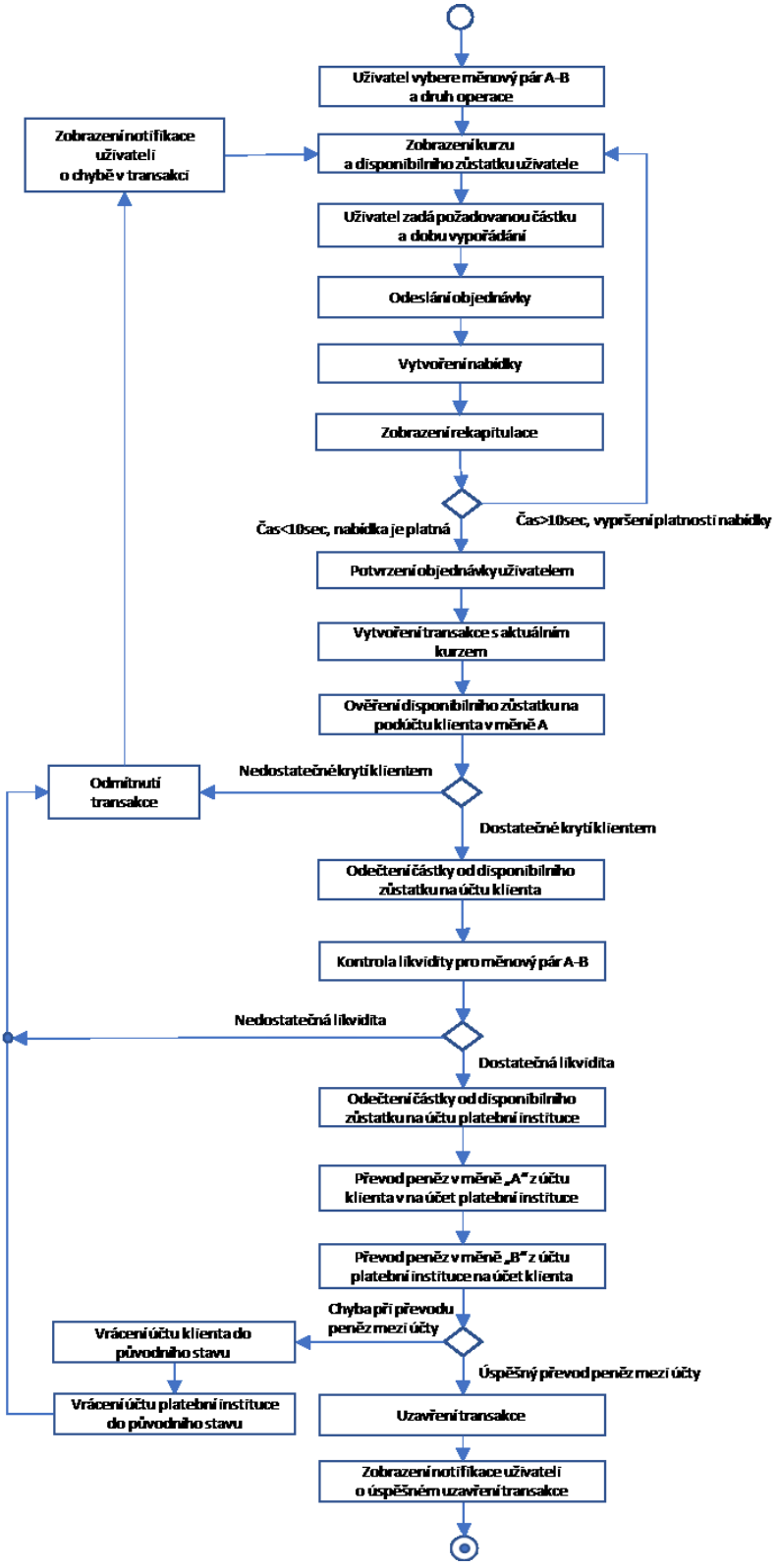
Přihlášení uživatele do systému bylo popsáno pomocí UC1 v kapitole 6.3.1. Událost přihlášení je velmi důležitá z pohledu bezpečnosti, protože je to vstup neznámého uživatele do systému. Diagram (viz Obrázek 4) umožnuje snadnou orientaci v daném procesu.



Obrázek 5. Diagram vícefaktorové autentizace

6.4.2 Vložení nové transakce uživatelem

Vložení požadavku na směnu je proces, který musí být uzavřen v transakci. Není přípustné, aby proces skončil někde „uprostřed“. Detail znázorňuje Obrázek 6.

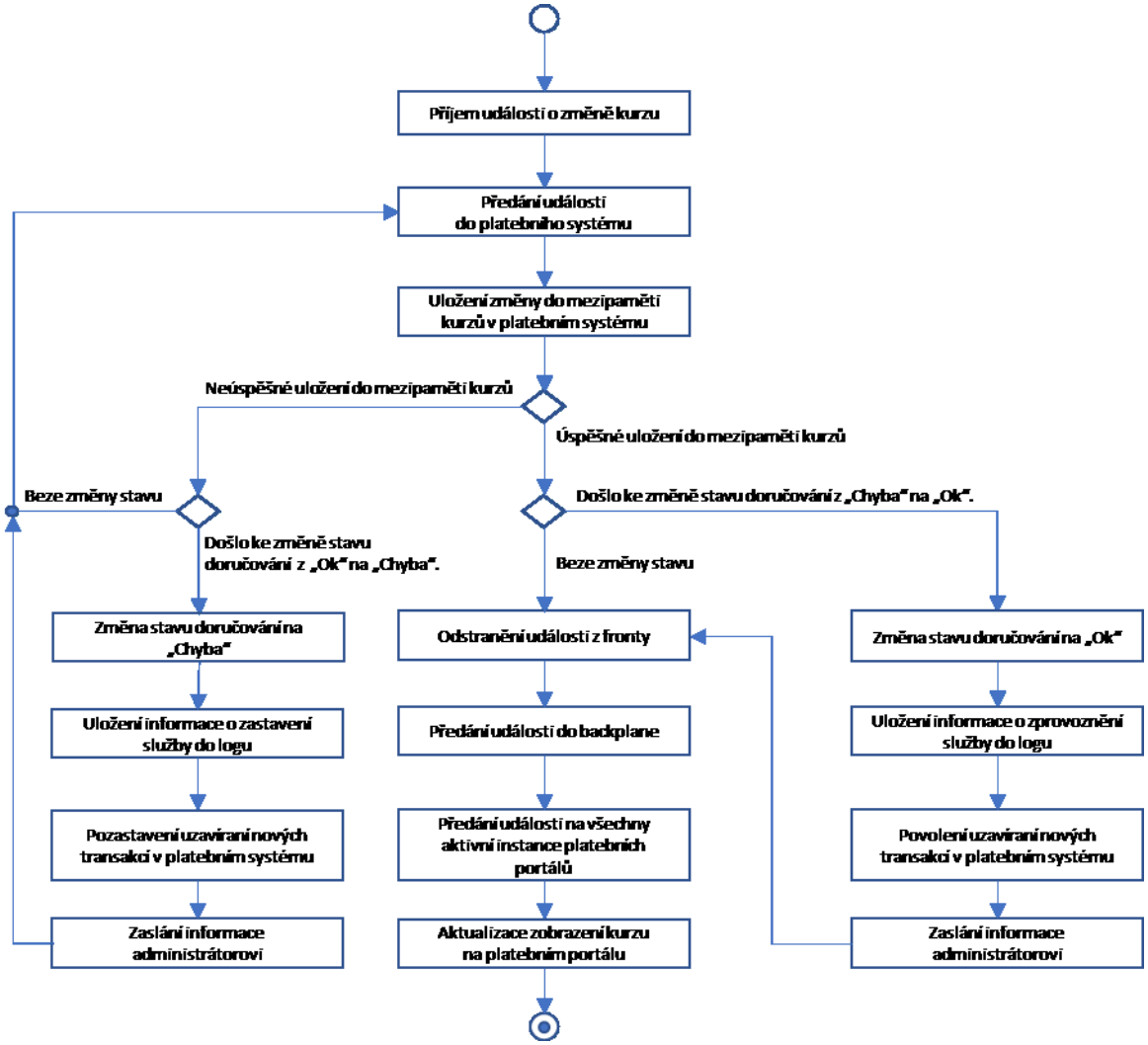


Obrázek 6. Vložení nové transakce uživatelem

Vždy musí být provedeny všechny operace v transakci, nebo žádná. Proces musí rovněž umožňovat paralelní zpracování transakcí, aby bylo možné systém horizontálně škálovat. To je zajištěno zpracováním transakcí pomocí fronty. Platební portál odešle objednávku do platebního systému. Ten zařadí záznam do fronty. Další část platebního systému pak načítá objednávky z fronty a provádí jejich vyřízení. Jakmile je objednávka vyřízena (kladně, nebo záporně), tak je z fronty odstraněna. Toto řešení zajistí, že systém zpracování objednávek je možné podle potřeby spustit paralelně vícekrát.

6.4.3 Periodická aktualizace kurzů měnových párů

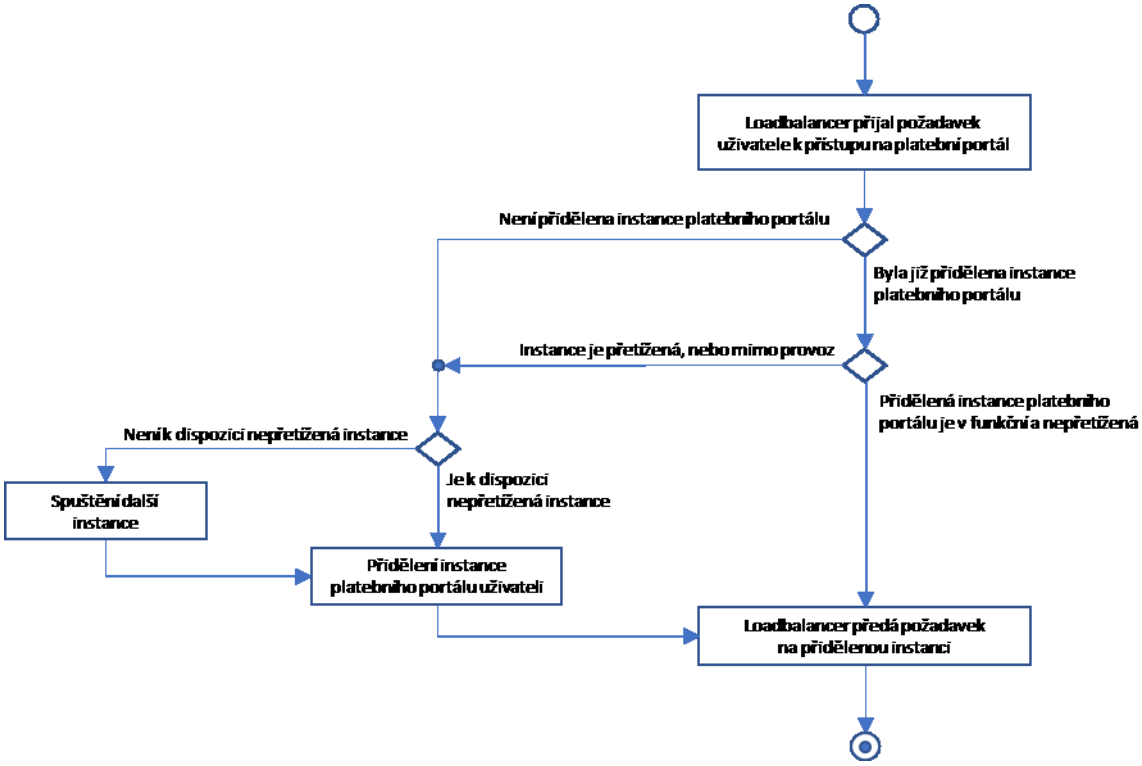
Poměrně náročnou operací z pohledu rychlosti komunikace a rizikovou operací z pohledu obchodu je systém pro periodickou aktualizaci kurzů měnových párů. Kurzy se mění ve velmi krátkých intervalech. Typicky se jedná o frekvenci do jedné vteřiny. Proto je potřeba, aby systém umožňoval efektivní a rychlé doručování zpráv do ostatních systémů. Je potřeba, aby byl, pokud možno odolný proti výpadku. Nicméně protože dodavatelem služby bude systém třetí strany, tak nelze vyloučit, že k výpadku může dojít. V tomto případě pak musí systém zastavit uzavírání nových objednávek, dokud není doručování aktuálních kurzů opět obnoveno. Detail řešení znázorňuje Obrázek 7.



Obrázek 7. Periodická aktualizace kurzů

6.4.4 Výpadek instance platebního portálu

Platební portál je propojen s platebním systémem přes backplane. To je služba, která umožní synchronizaci předávaných událostí mezi portály a platebním systémem tak, aby byl zachován nepřerušený provoz systému. Následující Obrázek 8 znázorňuje situaci vyřízeni „obecného" požadavku, s detailním řešením výjimek při výpadku platebního portálu.



Obrázek 8. Diagram systému pro horizontální škálování

6.5 Analýza infrastruktury

V předchozích kapitolách byly popsány požadavky na systém a všechna rizika, které se dají u takového systému očekávat. Následně je potřeba zvolit vhodnou infrastrukturu pro vývoj a provoz tohoto systému. Infrastrukturou se rozumí aplikační servery včetně sítových prvků, které budou zajišťovat provoz. Použití čistě podnikové infrastruktury pro finanční systém a zajistit přitom plnou ochranu je technicky proveditelné, ale v praxi ekonomicky velmi nevýhodné. Bylo by nutné podnikovou síť připojit vysokorychlostním připojením na páteřní internet. Dále by bylo nutné zajistit několik dalších nezávislých záložních připojení. Nejlépe dva až tři zahraniční spoje a minimálně jeden lokální peering v rámci státu. Dále by bylo nutné pořídit hardware, který bude schopen efektivně směrovat provoz a zajišťovat ochranu v případě útoku. Pokud je to zkomplikováno tím, že připojovaná lokalita je mimo hlavní páteřní trasy, tak to měsíční náklady ještě více zvýšilo.

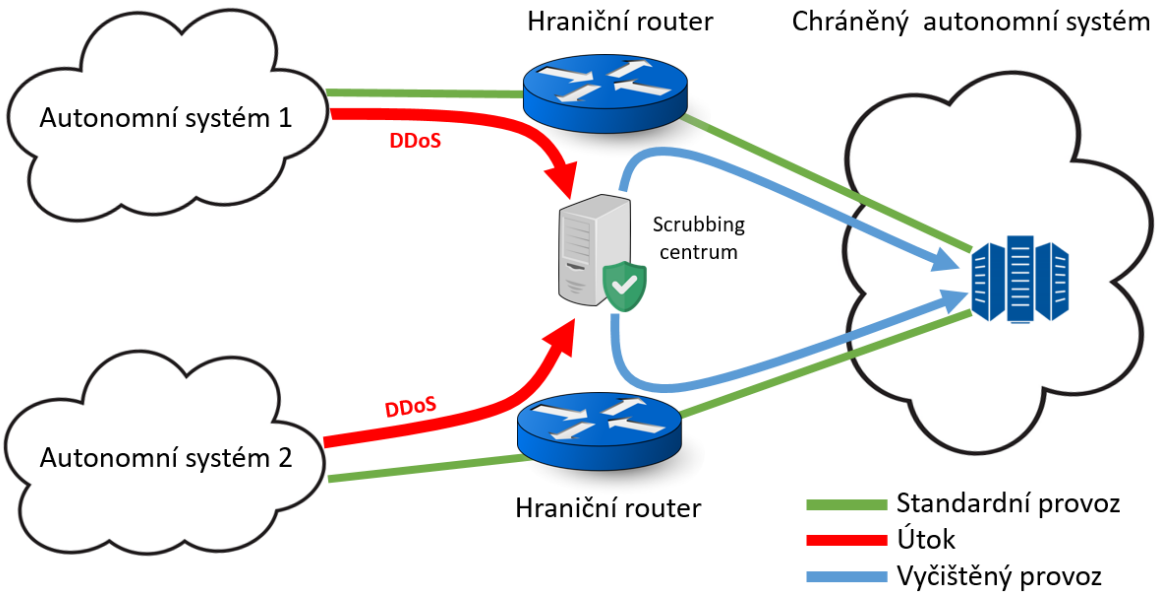
Z výše uvedeného lze vyvodit závěr, že výhodnější je zřídit infrastrukturu přímo v datovém centru, které již tyto prostředky má a pronajmout si je.

6.5.1 Systém pro prevenci proniknutí (IPS)

Datová centra již mají taková řešení pro své zákazníky připravena jako službu, která má pro webové servery již v sobě nakonfigurovánu optimální sadu pravidel. Mělo by stačit tedy pravidla s poskytovatelem služby zrevidovat a případně upravit podle potřeb navrhované aplikace. Jako příklad takových systémů je Next-Generation Intrusion System od společnosti Cisco Systems [22], nebo IPS systém od společnosti Fortinet [23]. Datová centra jsou schopna IPS službu poskytnout jak pro virtualizované servery, tak i pro fyzické servery.

6.5.2 Systém pro prevenci útoku DDoS

Součástí firewallu IPS bývá i základní ochrana proti DDoS útokům. Kvůli zapojení in-line je jejich schopnost omezena datovým tokem, který přes něj přímo protéká. V případě požadavku na zajištění vyšší dostupnosti je potřeba vybrat řešení Out-of-Path, které na základě analýzy provozu při zjištění DDoS útoku dokáže rekonfigurovat páteřní routery autonomních sítí pomocí protokolu BGP a odklonit síťový provoz do čistícího centra (na obrázku č.9 označeno jako “scrubbing centrum“), které zajišťuje očištění standardního provozu od útoku tak, aby byla dostupnost služby zachována. Prakticky tedy dojde k filtrování provozu už na hraničních routerech autonomních systémů. Toto řešení je v ČR již dostupné. Poskytuje ho například firma ČD-Telematika a jak poskytovatel na svých stránkách uvádí, tak řešení je vhodné i pro finanční instituce [24]



Obrázek 9. Schéma řešení ochrany proti DDoS

6.5.3 Systém pro automatické škálování výkonu

Výkon systému lze navyšovat dvěma způsoby, tzv. vertikálním a horizontálním škálováním.

Vertikální škálování představuje takové řešení, kdy výkon zařízení je navyšován množstvím procesorů a paměti. Problémem je, že u fyzických serverů se navyšování dělá velmi obtížně a velmi brzy se narazí na limit, kdy už dál navyšovat výpočetní kapacitu nelze.

Oproti tomu horizontální škálování využívá metody rovnoměrného rozložení zátěže na více souběžně pracujících serverech. Rovnoměrné rozložení zátěže zajišťuje loadbalancer. Loadbalancer může být řešen buď na úrovni DNS serveru, nebo pomocí hardware který řídí provoz na síťové vrstvě. Vyvažování zátěže pomocí DNS není vhodné pro systémy, kde vyžadujeme současně i vysokou dostupnost služeb. Proto bude vhodnější vybrat systém vyvažování na úrovni síťového provozu.

Detailněji je problematika vyvažování zátěže a vysoké dostupnosti popsána v knize plánování, nasazení a správa datového centra v cloudovém prostředí [25].

Další výhodou tohoto řešení je možnost konfigurace fail-over systému, který monitoruje stav jednotlivých serverů a v případě výpadku jej za běhu odpojí a provoz je automaticky přesměrován na jiný server. Další možností je využití automatického horizontálního škálování, kdy systém při zátěži automaticky připojuje další servery podle potřeby. Když pak zátěž klesne, tak nadbytečné servery zase opět deaktivuje. Tím se šetří výkon i energie. Automatické škálování je velmi efektivní, lze jej ale aplikovat pouze ve virtualizovaném prostředí. Nevýhodou je vyšší cena a specifické požadavky na návrh architektury a SW implementace. Pokud bude při návrhu systému rozhodnuto, že se využije horizontální škálování, tak již dopředu musí být aplikační část připravena tak, aby splňovala potřebné požadavky tohoto řešení. Mezi základní požadavky na serverovou aplikační část patří:

- Na serverech, které zajišťují komunikační rozhraní s klienty nesmí být žádná uživatelská data. Všechna data od uživatele je potřeba ukládat na sdílené úložiště, SQL, cache, atp.
- Uživatelská session musí být sdílená.

Lze tedy vyvodit, že pro malé weby by stačilo zůstat u vertikálního škálování, ale pro weby kde hlavní prioritou je dostupnost služby, bude vhodné zvolit metodu horizontálního škálování. Datová centra bývají na tento požadavek připravena. Pro plánovanou aplikaci na online devizové platby, kde se očekává ze začátku velmi malý provoz, který postupně bude

růst, takže bude nejvhodnějším řešením pořídit dva aplikační servery a před ně bude předřazen load balancer, který bude zajišťovat rovnoměrné vytěžování serverů. Datová centra k této službě nabízí i službu geo-replikace do jiné lokality a disaster recovery plán [26].

6.5.4 Systém pro autentizaci klienta

Zvolení správného typu autentizace je jedna z velmi důležitých rozhodnutí, protože mají velký vliv na bezpečnost celého systému a současně musí být pro uživatele, pokud možno, co nejjednodušší na ovládání. Existuje několik možností, jak autentizaci u bankovních aplikací řešit.

Současným trendem u bank je využívání zasílání autorizační SMS v kombinaci s přístupovým loginem a heslem. Klient se přihlásí pomocí loginu a hesla. Zadá transakci a před jejím odesláním je klientovi zaslána SMS s kódem, který klient připíše k transakci a tím je transakce více faktorově potvrzena. Problémem tohoto systému je, že SMS zprávy jsou placené. Typicky banky zasílají tyto SMS zdarma, ale účtují za službu udržovací poplatek. Systém se tak stává pro poskytovatele nevýhodným, když systémem začne protékat spousta malých transakcí a udržovací poplatek na pokrytí výdajů nebude stačit. Alternativa k tomuto řešení je hardwarový generátor autentizačního kódu. Je velmi efektivní a rychlý na používání. Nevýhodou je pouze jeho pořizovací cena a omezená životnost.

Další metodou autentizace je využití elektronických certifikátů. Banka klientovi vystaví certifikát, který si klient buď uloží do počítače, nebo ho má uložený na čipové kartě. Nevýhodou jsou vícenáklady na vystavování certifikátů a při prodlužování jejich doby platnosti. Platební instituce musí za tyto operace platit certifikační autoritě poplatky, které potom v různé formě nakonec musí uhradit klient. V poslední době se začínají pro víceúrovňovou autentizaci více využívat mobilní zařízení. V praxi je to řešeno tak, že klient si nainstaluje na svůj mobil aplikaci a místo autorizační zprávy formou SMS je odeslána zpráva do této mobilní aplikace. Výhodou jsou pro klienta nulové pořizovací náklady a pro banku nulové provozní náklady. Alternativou k tomuto řešení je možnost autentizace formou kontroly čtyř očí. Toto řešení je vhodné pouze pro větší organizace, která má zaveden proces schvalování plateb více osobami. Jedna oprávněná osoba vloží do systému požadavek a potvrdí jej autentizačním klíčem, který byl odeslán na její zařízení. Následně je pak odeslána informace o požadavku na schválení transakce druhé oprávněné osobě. Jakmile druhá osoba potvrdí transakci, tak je

odeslán druhý autentizační kód, tentokrát na zařízení druhé oprávněné osoby. Po jejím zadání je tedy transakce potvrzena dvěma nezávislými subjekty v kombinaci s více faktorovou autentizací a je přijata bankou ke zpracování.

Trendem se v poslední době rovněž stává biometrie. To znamená, že k autentizaci by se využil klientův otisk prstu, hlas, sken oka atp. Technologie je sice stále v počátcích, ale v USA již velké banky tuto technologii podporují. Například tři miliony klientů Bank of America již používají pro přihlašování na své účty otisk prstu na svém mobilním telefonu [27]. Banka Well Fargo dokonce využívá mobilní zařízení pro sken očí [28]. Rovněž dle sdělení regulátora ČNB, je možné při autentizaci bankovních transakcí využít i biometrii [3].

Po posouzení parametrů jednotlivých typů autentizace popsaných v této kapitole se jeví jako nejlepší řešení využití biometrie. Platební instituce by získala konkurenční výhodu, protože tato technologie je dle průzkumu vnímána klienty pozitivně [29] a jiné banky biometrii buď vůbec nepoužívají, nebo s ním teprve začínají. Pro klienty používání tohoto typu autentizace znamená velký uživatelský zážitek. Nevýhodou je, že typy zařízení, které tento způsob autentizace podporují, jsou na trhu zatím stále málo rozšířené.

Z výše uvedeného tedy vyplývá, že do systému bude vhodné zakomponovat autentizaci pomocí biometrie a alternativně bude dostupná i tradiční metoda pomocí SMS kódu, případně přes mobilní aplikaci sloužící jako více faktorové ověření. Systém tak umožní přístup uživatelům, kteří nebudou schopni kvůli technologickému omezení svého zařízení novou metodu biometrie prozatím využít.

6.5.5 Infrastruktura pro vývoj a testování

Výše popsaná problematika zahrnuje velmi komplexní systém a využívá velké množství specializovaného hardware. Pro vývoj a otestování je vhodnější nejprve zrealizovat projekt formou prototypu (Proof of Concept), který umožní zjistit nejvhodnější konfiguraci systému pro finální řešení. Velmi vhodné je zřídit si požadované prostředí formou „infrastruktura jako služba“ (IaaS). Umožní to flexibilně měnit parametry prostředí podle potřeby, testovat zátěž, přidávat a snižovat výkon systému, simulovat výpadky. Pronajaté prostředí umožňuje platit pouze za prostředky, které se právě využívají, takže mimo dobu vývoje je možné prostředky uvolnit a ušetřit tak náklady. Další výhodou je, že se nemusí platit pořizovací náklady.

Infrastrukturu produkčního prostředí je pak možné postavit podle požadavků, které vyplynou ze závěrů testů na vývojovém prostředí.

Pro produkční řešení je potřeba zohlednit požadavky a doporučení regulátora finančního trhu. Zatím tedy v současné době je nejvhodnější provozovat infrastrukturu v datovém centru s tím, že není doporučeno využít nadnárodní cloud, či decentralizaci dat formou georeplikace.

6.6 Implementace

V implementační části je popsán postup praktické realizace jednotlivých částí systému. Jako programovací jazyk byl zvolen C#.NET, který umožní tento projekt realizovat v jednom vývojovém prostředí. Umožní efektivně zrealizovat jak systémové služby, tak i multiplatformní uživatelské rozhraní. Jako vývojové prostředí bylo použito Visual Studio 2015. V době psaní této práce byla k dispozici novější verze Visual Studio 2017 RC1, ale obsahovala chyby v knihovnách pro vývoj mobilních aplikací. Verze 2015 byla pro požadovaný vývoj naprosto postačující. Jako infrastruktura byla zvolena platforma Microsoft Azure, přestože Amazon poskytuje srovnatelné cloudové služby. Důvodem byla očekávaná lepší integrace a podpora služeb MS Azure v knihovnách .NETu. Pro aktivaci podpory služeb MS Azure přímo z prostředí Visual Studia, stačí nainstalovat MS Azure SDK for .NET., které je k dispozici ke stažení na adrese https://azure.microsoft.com/cs-cz/downloads .

6.6.1 Ochrana proti DDoS

Jak bylo popsáno v kapitole 5.1, tak pro produkční prostředí bude vhodné službu objednat přímo u poskytovatele služeb datového centra. Preferovaná by měla být ta datová centra, která umí DDoS ochranu na úrovni GPG směrování. Například: Dial Telecom, a.s., T-Mobile Czech Republic a.s. Na vývojovém prostředí tato funkce nebude implementována z důvodu vysokých nároků na realizaci.

6.6.2 Fronta

Fronta bude využita platebním systémem pro zvýšení odolnosti proti selhání a pro možnost navyšování výkonu paralelním zpracováním objednávek. Rozhraní platebního systému převezme potvrzení objednávky z platebního portálu a vloží objednávku do fronty. Služba platebního systému, která může být spuštěna ve více instancích bude načítat nové objednávky

z fronty a provádět jejich vyřízení. Na MS Azure byla služba zřízena s následujícími parametry:

- Name: queueexchangerates
- Deployment model: Resource manager
- Account kind: General purpose
- Performance: Standard
- Replication: Locally-redundant storage (LRS)
- Resource group: RG_queue

6.6.3 Push notifikační služba

Služba bude sloužit pro zasílání notifikací ze systému vícefaktorové autentizace. Služba byla zřízena s následujícími parametry:

- Notification Hub: exchangeratespush
- Create new namespace: exchangerates
- Location: West Europe
- Resource Group: RG_pushnotification
- Pricing tier: Free[3]

6.6.4 Instance platebního portálu a systém horizontálního škálování

Platební portály musí podporovat automatické horizontální škálování, vyvažování zátěže a vysokou dostupnost. Pro tyto požadavky je nejvhodnější využít službu „Virtual Machine Scale Sets“. Při konfiguraci služby se vytvoří jeden virtuální stroj, který současně slouží jako vzor pro kopie dalších instancí. Systém má předřazen loadbalancer ve vysoké dostupnosti, který kromě vyvažování zátěže zajišťuje i automatické spouštění a opětovné vypínání dalších kopií virtuálních serverů tak, aby zátěž jednotlivých instancí nepřekračovala stanovenou mez. Při zřízení služby byla zvolena kategorie A1 v2, která je doporučena pro vývoj, testy a současně již v sobě obsahuje i loadbalancer. Při vytvoření byly použity následující parametry:

- Virtual machine scale set name: ScaleSetPlatebniPortal

[3] Služba v režimu zdarma obsahuje 1mil. notifikací, 500 aktivně připojených zařízení

- OS type: Windows
- Limit to a single placemen group: true
- Resource group: create new
- Resource group name: RG_platebniPortal
- Location: West Europe
- Public IP address: new
- Domain name label: platebniportal[4]
- Operating system disk image: 2016-Datacenter
- Instance count: 2
- Disk Type: managed
- Scale set virtual machine size: Standard D1_V2 [5]
- Autoscale: enabled
    - Minimum number of VMs: 1
    - Autoscale maximum number of VMs: 3
    - Scale out CPU percentage threshold: 75 (výchozí hodnota)
    - Number of VMs to increase by on scale out: 1
    - Scale in CPU percentage threshold: 1
    - Number of VMs to increase by on scale in: 1

Cenový kalkulátor vypočítal za službu horizontálního škálování cenu 0,12 Eur za 1 hodinu provozu za provoz jedné instance virtuálního serveru (tj. 87,84 EUR za 1 měsíc). Po dobu zátěže nad 75% CPU, bude systém postupně zapínat další instance. Cena bude účtována za každou další běžící instanci zvlášť. Systém po snížení zátěže opět automaticky sám vypne nepotřebné instance, když zátěž klesne pod 25% CPU.

6.6.5 Backplane a SignalR

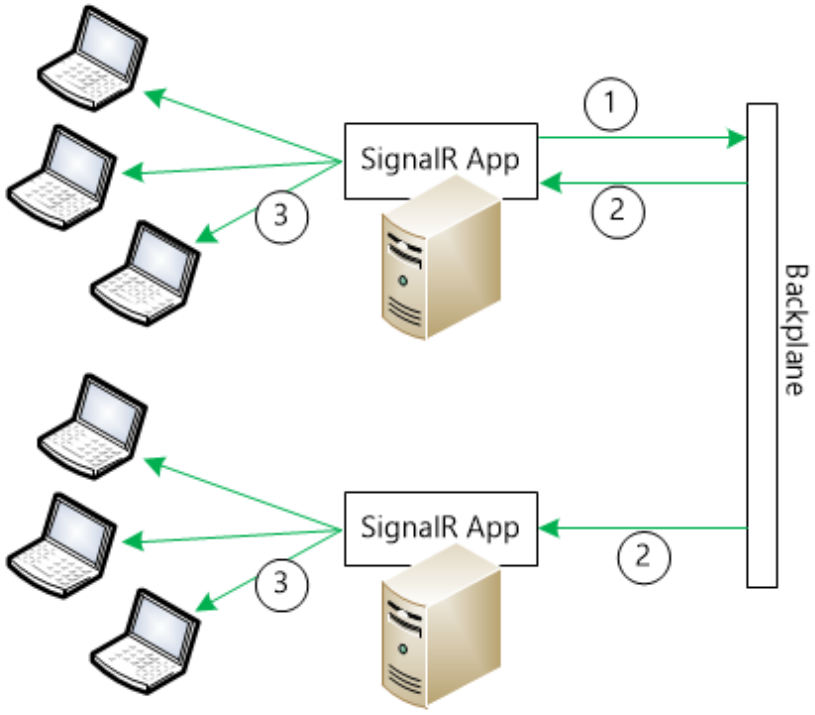
Protože v požadavcích na rozhraní bylo, že musí umožňovat rychlou a obousměrnou aktualizaci dat, tak byla vybrána technologie SignalR [30]. SignalR je knihovna pro ASP.NET, která zjednodušuje proces implementace funkce doručování zpráv rychlostí blízké reálnému

[4] Slouží jako veřejná adresa loadbalanceru (platebniportal.cloudapp.azure.com)
[5] Konfigurace pro jednotlivé instance: 1x CPU, 3.5GB RAM,2x SSD 50GB, Load balancing

času. Velkou výhodou proti jiným technologiím je podpora zpětné kompatibility se staršími typy webových prohlížečů, které ještě nepodporují webový soket[6]. SignalR umožnuje kromě webového soketu komunikovat i pomocí serverem zasílaných událostí[7] a Comet transportu[8]. Funkce zpětné kompatibility je implementována tak, že při inicializaci spojení si klient i server mezi sebou domluví nejrychlejší možný protokol výměny dat.

Aby bylo možné provozovat tento komunikační systém v clusteru více instancí platebních portálů, tak je nutné použít komponentu Backplane[9]. Princip funkce spočívá ve společném úložišti pro ukládání a distribuci zpráv, které jsou zasílány mezi serverem a klientem. Zpráva ze serveru je tedy uložena do sdílené paměti, odkud je rozeslána na všechny připojené instance aplikace SignalR (viz Obrázek 10). Nevýhodou je zmenšení propustnosti maximálního počtu přenesených zpráv. Je to z toho důvodu, že Backplane přeposílá každou zprávu na všechny připojené instance aplikací SignalR. Dle doporučení dodavatele [30] je vhodné Backplane použít v případě, že server vysílá zprávy na velké množství klientských aplikací. Důvodem je, že server má možnost řídit rychlost zasílání zpráv, a to je případ i této aplikace.

[6] Webový soket – umožňuje plnohodnotnou a plně duplexní komunikaci. Podporován na prohlížečích s podporou HTML5.
[7] Server Sent Events – je podporován všemi prohlížeči s výjimkou MS Exploreru.
[8] Jedná se o komunikaci pomocí dlouhých rámců. Tzv. Long-polling. Komunikace je podporována na všech platformách a verzích prohlížečů, ale je nejméně efektivní.
[9] Backplane umožňuje synchronizaci mezi jednotlivými instancemi systémů, které jsou k ní připojené.



Obrázek 10. Princip synchronizace SignalR pomocí Backplane [30]

SignalR přímo podporuje tři metody implementace Backplane:

- **ServiceBUS**

  Z pohledu rychlosti a efektivity nevýkonnější a nejvíce variabilní. Umožňuje zasílat zprávy v režimu klient-klient, nebo klient-více klientů. Lze ji ale použít pouze v cloudu MS Azure.

- **Redis**

  Redis úložiště je velmi rychlá NoSQL cache. Ukládá data ve struktuře klíč-hodnota. Data jsou kvůli výkonu ukládána přímo do paměti ve struktuře klíč-hodnota. Výhodou Redisu je možnost navyšovat výkon horizontálním škálováním. Rovněž podporuje replikace master-slave, takže při výpadku nodu se data neztratí, a navíc se s daty dá pracovat transakčním způsobem. Výhodou je vyšší výkon ve srovnání s implementací pro SQL Server. Nevýhodou je nutnost rozšířit systém o další prvek, čímž vzrůstají vícenáklady.

- **SQL Server**
  Umožňuje pro ukládání události využít SQL server. SQL server umožňuje replikaci master-slave. Výhodou je velmi snadná konfigurace a implementace. Typicky v systémech bývá vždy přítomen SQL server pro ukládání dat, a tak jeho další využití i pro Backplane umožní sdílení prostředků a tím snížení provozních nákladů. Nevýhodou je, že se jedná o nejpomalejší z podporovaných technologií.

Po zvážení jednotlivých vlastností byla zvolena implementace pro SQL Server s tím, že později bude jednoduché vyměnit implementaci Backplane SQL za Redis, pokud to bude vyžadováno z pohledu výkonu. Při zřízení podpory služby na SQL serveru služby byl proveden následující postup:

- Vytvoření nové databáze s názvem db_backplane
- Vytvoření uživatele v SQL s právy log-in, create schemas, create tables do databáze db_backplane
- Zapnutí služby Service Broker[10] spuštěním SQL příkazu:
  ```
  ALTER DATABASE db_backplane SET ENABLE_BROKER
  ```

Detailní popis je implementace podpory horizontálního škálování je popsán v dokumentaci dodavatele [30].

6.6.6 Sdílené datové struktury

Jako první je vhodné navrhnout pro vzájemně komunikující systémy společné datové struktury, které umožní efektivní serializaci a deserializaci přenášených zpráv. Pro úsporu datového toku bylo při návrhu dbáno na minimalizaci délky názvu jednotlivých atributů [12]. Pro serializaci dat pro webový portál, kde probíhá komunikace s klientem pomocí datového formátu Javascript Object Notation (JSON) je navíc možné upravit systém tak, aby názvy při přenosu zkracoval. Originální název je přitom zachován. Pro serializaci byla použita knihovna NewtonsoftJson.

[10] Pokud to SQL Serveru podporuje, tak je vhodné službu zapnout. Služba umožňuje výrazně zvýšit efektivitu výměny zpráv mezi SQL serverem a klientem. Není nutné ji zapínat, systém funguje i bez toho.

Ukázka třídy pro přenos změny kurzu měnového páru:

```
public class CurrencyPair
    {
        [JsonProperty("i")]
        public int Id { get; set; }

        [JsonProperty("b")]
        public decimal Buy { get; set; }

        [JsonProperty("s")]
        public decimal Sell { get; set; }
    }
```

6.6.7 SQL datové struktury

V případě, že finanční instituce, která bude chtít toto řešení využít, budou některé části nahrazeny stávajícím řešením. Zejména se jedná o transakční platební systém a rozhraní pro přístup k bankovním účtům. Ukázková data jsou ukládána na SQL serveru. Pro demonstraci základních funkcí, byly implementovány jen nejnutnější datové struktury. Pro přístup k SQL serveru byl zvolen EntityFramework, který je dodavatelem technologie .NET doporučován pro přístup k datům [31]. Jedná se o objektově relační vrstvu, který vývojářům umožňuje přistupovat k relačním datům a současně výrazně zjednodušuje syntaktický zápis programového kódu.

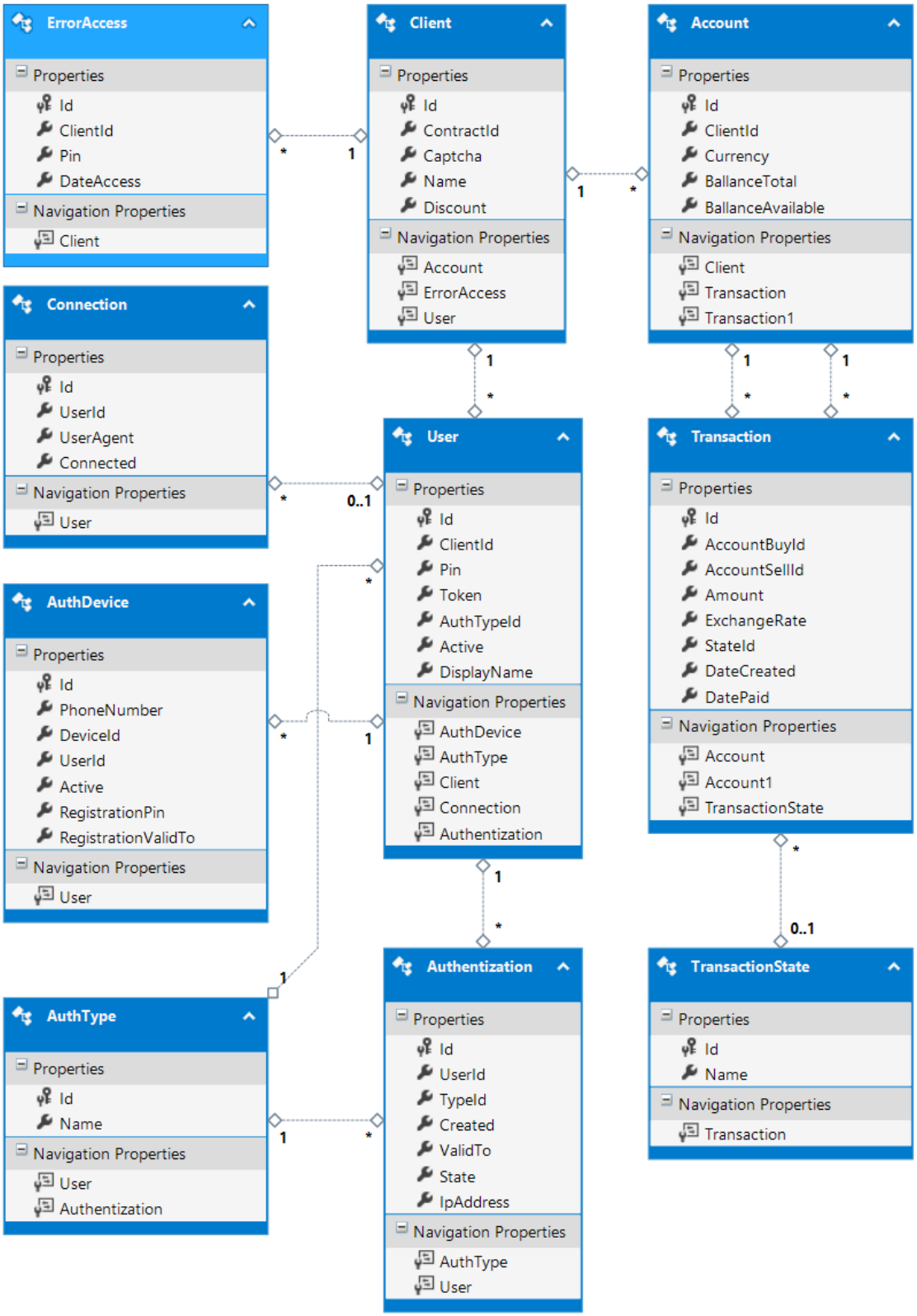
Na SQL serveru byly vytvořeny následující tabulky:

- **Client**
  Číselník klientů. U klienta se eviduje unikátní Id, název a velikost slevy.
- **User**
  Číselník oprávněných osob. Každá osoba má svůj osobní PIN, který slouží společně s Id klienta pro identifikaci prvním faktorem. U uživatele se eviduje, jaký typ autentizace úspěšně provedl při posledním přihlášení.
- **Authentizatior**
  Evidence stavu žádostí uživatelů pro schválení druhým faktorem.
- **AuthDevice**
  Číselník registrovaných autentizačních zařízení oprávněné osoby. Zařízení smí být registrováno pouze jednou. Není povoleno registrovat více zařízení se stejným číslem, nebo stejným DeviceId. Aktivaci mobilní aplikace lze pro zařízení provést rovněž pouze jednou.
- **AuthType**

Číselník typů autentizace (SMS, APP).

- **Connection**
  Evidence aktivních připojení uživatelů do systému. Úspěšně provedená autentizace umožňuje přes přidělený jednorázový token vytvořit pouze jedno aktivní připojení.
- **Account**
  Číselník účtů klientů. Eviduje se měna, celkový zůstatek a disponibilní zůstatek.
- **ErrorAccess**
  Evidence neúspěšných pokusů o přihlášení. Umožňuje zvyšovat obtížnost CAPTCHA kódu při opakovaném vkládání neplatných údajů. Cílem je ochrana proti robotům.
- **Transaction**
  Evidence transakcí v systému. U každé transakce se eviduje prodejní a nákupní účet klienta, kurz, částka, datum vytvoření, datum vypořádání a stav transakce.
- **TransactionState**
  Číselník jednotlivých stavů pro transakce (NEW, RUNNING, SUCCESS, FAILED).

Tabulky slouží pro ukládání údajů platebního systému a systému pro vícefaktorovou autentizaci.



Obrázek 11. Struktura SQL tabulek pro platební systém

6.6.8 Serverová část systému

Serverové části systému jsou aplikace, které nemají přímou interakci s uživatelem, a tak u nich není potřeba implementovat uživatelské rozhraní. Nicméně komunikují s jinými systémy a je potřeba vhodně navrhnout společné rozhraní tak, aby komunikace byla co nejvíce efektivní a bezpečná.

6.6.8.1 Služba periodické aktualizace kurzů měnových párů

Tato služba bude v provozu nepřetržitě a bude připojena na externí zdroj událostí změn kurzů měnových párů. Tento externí zdroj bude v praxi dodán dodavatelem třetí strany. Zde byla pouze pro názornost provedena minimální implementace. Zdrojem dat jsou náhodně generovaná data. Cílem bylo demonstrovat schopnost přenášet data do platebního systému.

Projekt byl kvůli jednoduššímu ladění vytvořen jako konzolová aplikace. Pro produkční prostředí je ale nezbytné, aby byl řešen jako Windows služba. To umožní efektivní správu běhu a zotavení v případě výpadku. Služba tedy je permanentně připojena na rozhraní platebního systému, do kterého zasílá zprávy o změnách kurzu voláním metody **ExRatesChanged(),** které parametrem předá pole změněných kurzů. V případě výpadku zavolá na rozhraní platebního systému funkci **ExchangeRateStopped().** V případě obnovení provozu zavolá funkci **ExchangeRateStarted()**.

6.6.8.2 Rozhraní pro obsluhu bankovních účtů

Jedná se o sdílenou knihovnu, která umožnuje komunikovat se systémem bankovních účtů.

Pro testovací účely postačuje, že metody vrací fiktivní data. Pro ovládání stavu bankovního účtu bylo pro zvýšení efektu reálného provozu přidáno zpoždění v řádu 1-5sec. Implementovány byly metody:

- **GetAccountHistory()**
  Načte seznam probíhajících a vypořádaných transakcí pro daný účet dle filtru.
- **GetAccountBalance()**
  Načte celkový a disponibilní zůstatek na účtu.
- **PutTransaction()**
  Umožní transfer peněz z účtu A na účet B. Pracuje v transakci. V případě chyby vrátí funkce výjimku a interně si provede rollback. Funkce si interně zjišťuje stav disponibilních zůstatků na účtech, likviditu trhy a provádí transfer peněz.

6.6.8.3 Služba pro vyřizování objednávek

Tato služba bude permanentně připojena na frontu, ze které bude načítat nové zprávy pomocí metody GetMessage(). Načtená zpráva bude deserializována podle následující struktury:

```
public class Order
    {
        public long Id { get; set; }
        public int AccountBuyId { get; set; }
        public int AccountSellId { get; set; }
        public decimal ExchangeRate { get; set; }
        public decimal Amount { get; set; }
        public StateEntity State { get; set; }
        public DateTime DateCreated { get; set; }
        public DateTime? DatePaid { get; set; }
    }

public enum StateEntity
    {
      NEW,
      RUNNING,
      SUCCESS,
      FAILED
    }
```

Zde optimalizace na množství přenášených dat není potřeba. Limit pro jednu zprávu ve frontě je 64KB a data budou přenášena pouze v rámci datového centra. Služba bude v rozhraní bankovních účtů volat metodu **PutTransaction()**. Objednávce nastaví stav "RUNNING" pomocí metody **UpdateMessage()** a objednávka je uzamčena. V případě úspěšného zpracování nastaví objednávce stav „SUCCESS". Následně odešle oznámení do platebního portálu o změně stavu objednávky metodou **OrderChanged()** a objednávku z fronty metodou **DeleteMessage()** odstraní. V případě, že nedojde k úspěšnému vyřízení transakce, nastaví objednávce stav „FAILED", odešle notifikaci do platebního portálu metodou **OrderChanged()** a zprávu z fronty odstraní metodou **DeleteMessage().**

6.6.8.4 Rozhraní platebního systému

Rozhraní je připojeno na systém platebního portálu přes Backplane. V případě požadavku nové objednávky, založí nový záznam do fronty objednávek metodou **AddMessage().**

V rozhraní jsou implementovány následující metody:

- **AddRequest()**

  Požadavek klienta na nákup nebo prodej cizí měny.
- **ConfirmRequest()**

  Potvrzení požadavku klientem

- **GetAccountHistory()**
  Načte z platebního rozhraní seznam transakcí pro daný účet dle filtru.
- **GetAccountBalance()**
  Načte celkový a disponibilní zůstatek na účtu z rozhraní platebního systému.
- **OnRequestChanged()**
  Notifikace klienta o vytvoření nové nabídky na základě zadaného požadavku.
- **OnOrderChanged()**
  Notifikace klienta o změně stavu objednávky.
- **OnAccountBalanceChanged()**
  Notifikace klienta o změně stavu zůstatku na účtu.
- **OnAccountHistoryChanged()**
  Notifikace o změně výpisu historie účtu.
- **OnExRatesChanged()**
  Notifikace o změně kurzu měn.
- **OnExchangeRateStopped()**
  Notifikace o pozastavení uzavírání nových obchodů z důvodu nedostupnosti aktuálních kurzů.
- **OnExchangeRateStarted()**
  Notifikace o spuštění uzavírání nových obchodů z důvodu opětovné dostupnosti aktuálních kurzů.
- **UserLogIn()**
  Přihlášení uživatele do systému prvním faktorem. Synchronní funkce. Pomocí parametru je možné zvolit typ sekundární autentizace klientem.
- **OnUserAuthFinished()**
  Notifikace klienta o výsledném stavu sekundární autentizace.

6.6.8.5 Subsystém ochrany proti útoku hrubou silou

Útok hrubou silou není útokem DoS. Jedná se ve své podstatě o legitimní provoz, kdy útočník opakovaně zkouší zadat přihlašovací údaje. Tato část ochrany systému je velmi citlivá, protože nevhodně navržená ochrana by mohla omezit přístup i legitimnímu uživateli. Nejvhodnější bude tedy umožnit pro první přístup uživateli zadat přihlašovací údaje, pokud možno co nejjednodušeji. V případě že údaje nesouhlasí, tak se pro každý další pokus již

zaktivuje režim kontroly ochrany proti robotům. Uživatel je kromě autentizačních údajů požádán o vyplnění CAPTCHA kódu. S každým dalším chybným pokusem o přihlášení se složitost postupně zvyšuje. Po 20-ti neúspěšných pokusech bude rovněž prodloužena doba pro ověření platnosti údajů. Režim ochrany se automaticky opět vypne po 15-ti minutách nečinnosti, nebo po úspěšném přihlášení. Systém pro generování obrázku CAPTCHA byl vyřešen přidáním web handleru Captcha.ashx. Má jediný vstupní parametr c, ve kterém očekává id smlouvy, které uživatel zadal do autentizačního formuláře. Následně vygeneruje obrázek, který obsahuje 5 znaků. Obrázek je proložen šumovými prvky. Aby byla zachována podpora horizontálního škálování, tak pro předávání parametru nebyla použita session ani cookies.

6.6.8.6 Subsystém vícefaktorové autentizace

Dle návrhu vícefaktorové autentizace (viz Obrázek 9) se system skládá z těchto částí:

- Registrační rozhraní
- Push notifikační HUB
- SMS Brána

Registrační rozhraní bylo řešeno formou webové služby postavené na architektuře REST. Do projektu byl přidána třída RestControler.cs, ve které jsou uloženy všechny metody webové služby. která umožňuje provést prvotní registraci klienta. Pro mapování směrování byla vybrána technika automatického mapování směrování, která umožní pomocí návrhového vzoru Dekorátor nastavovat směrování přímo u metod webové služby. Což zlepšuje přehlednost a udržovatelnost kódu. Implementovány byly funkce pro autentizaci prvním faktorem, tedy pomocí čísla smlouvy a PINu. V případě že uživatel zvolí autentizaci přes mobilní aplikaci, tak je zavolána metoda GetAuthToken, která ověří platnost zadaných údajů. V případě, že je přihlášení úspěšné, tak platební portál vygeneruje token, který pak klient může použít pro připojení na SignalR rozhraní. Token lze využít pouze pro jedno aktivní připojení.

Push notifikační HUB byl realizován ve třídě PushNotification.cs. Odeslání notifikace na mobilní zařízení daného uživatele řeší metoda SendMessage. Při implementaci bylo potřeba ještě zajistit správný formát odesílané zprávy, který se pro každou platformu liší. Knihovna má na toto řešení v podobě šablonového systému. Ten funguje tak, že při registraci zařízení je zaevidován typ platformy. Při odesílání notifikace je pak zpráva naformátována podle této

šablony tak, aby na zařízení klienta došlo ke správnému formátu. Tato část ale nebyla hlavním cílem práce, a tak byla implementována jen základní notifikace bez šablon, aby bylo možné provést alespoň základní test vícefaktorové autentizace.

Autentizace pomocí SMS kódu implementována nebyla, nicméně její použití je obdobné jako u CAPTCHA kódu. Po vygenerování kódu je kód odeslán formou SMS. Klient opíše kód do formuláře v prohlížeči a následně dojde k jeho ověření.

6.6.8.7 Podpora automatického škálování

Podpora automatického škálování byla zaktivována u systému platebního portálu. Služba vyřizování objednávek je spuštěna prozatím v jedné instanci. Podporu automatického škálování je možno jednoduše spustit podobně, jako bylo popsáno v kapitole 6.6.4. Systém je na to připraven.

6.6.9 Klientská část

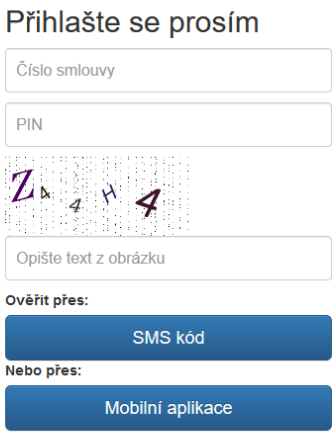
Klientské rozhraní jsou dvě. Platební portál a mobilní aplikace sloužící pro autentizaci druhým faktorem. Při návrhu rozhraní byl kladen důraz na univerzálnost řešení, tedy aby výslednou aplikaci mohl využít bez omezení libovolný uživatel. Systém musí tedy být multiplatformní a responzivní.

6.6.9.1 Platební portál

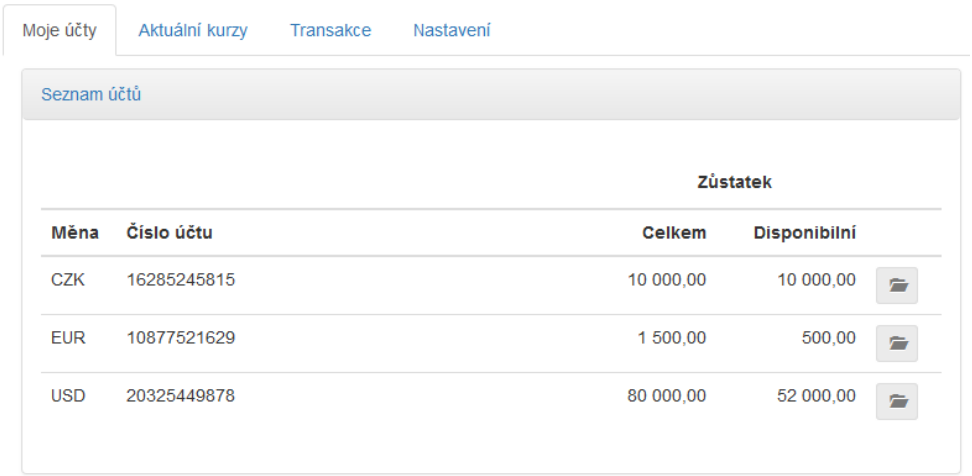
Platební portál je postaven na webové technologii ASP.NET. Jako frontendový framework byl použit Bootstrap [17]. Jedná se o open source kolekci nástrojů, určených k vytváření responzivních webových aplikací. Díky tomu se výsledná webová stránka správně zobrazí na různě velkých displejích, od velkých až po malé mobilní telefony. Tímto bude zajištěn požadavek, že rozhraní bude multiplatformní. Komunikace mezi klientem a serverem byla řešena pomocí frameworku SignalR. Web byl rozdělen na menší funkční části. Dle požadavků a případů užití byly identifikovány následující podstránky:

- Přihlášení do systému
- Přehled účtů
- Přehled transakcí
- Kurzovní lístek
- Formulář uzavření nového obchodu
- Nastavení

Při prvním přístupu na web se zobrazí uživateli přihlašovací obrazovka. Uživatel je vyzván k zadání čísla smlouvy a PIN. PIN je unikátní číslo, které umožní identifikovat oprávněnou osobu k dané smlouvě. Jakmile proběhne ověření prvním faktorem, tak je odeslána výzva k ověření druhým faktorem a platební portál čeká na potvrzení z autentizačního systému. Po provedení autentizace druhým faktorem platební portál provede přihlášení uživatele, stránka se přesměruje na stránku s přehledem účtů a zobrazí jejich aktuální stav.



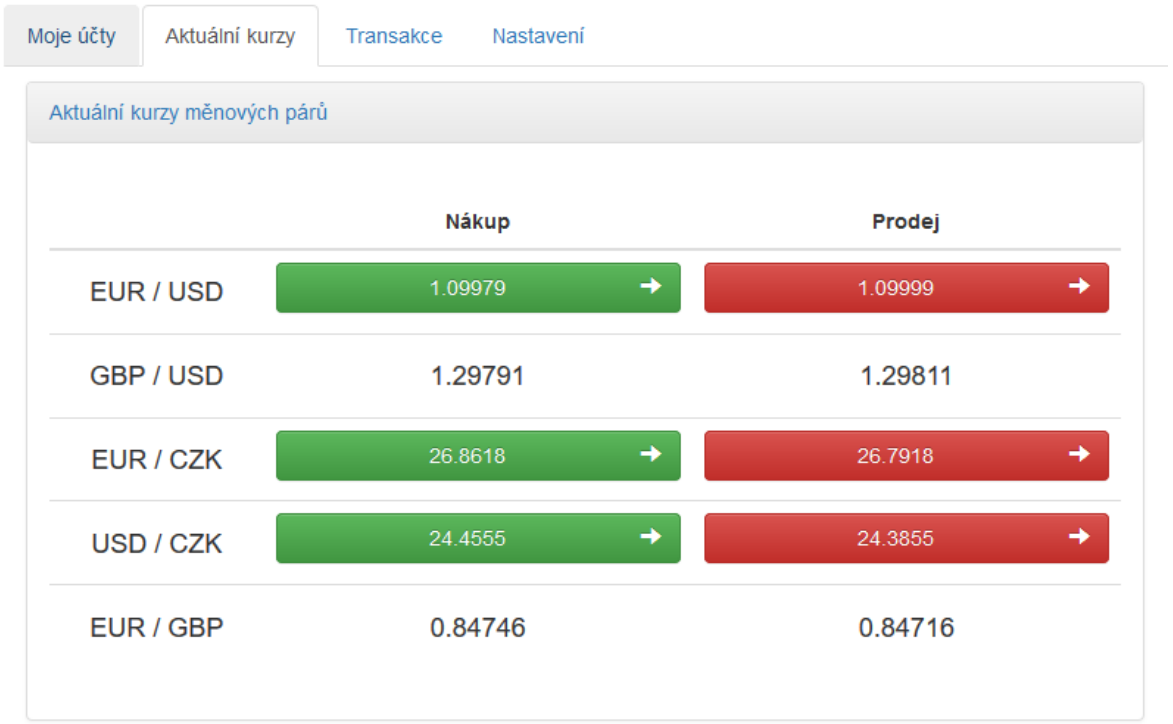
Obrázek 12. Platební portál – přihlášení



Obrázek 13. Platební portál – přehled účtů

Stránka seznamu účtů ukazuje aktuální stav. V případě, že dojde na účtu ke změně, tak se změna ihned promítne i na webovém prohlížeči. Tento přístup je pro uživatele mnohem příjemnější než klikat na tlačítko pro obnovení obsahu stránky.

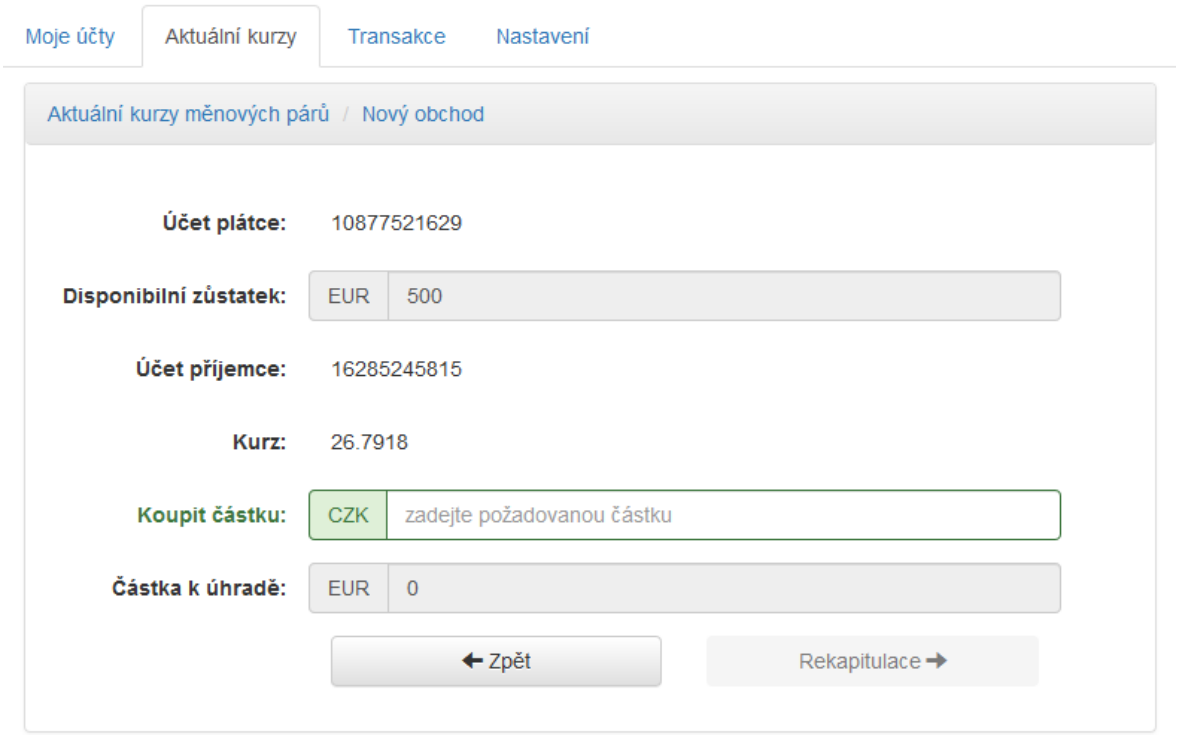
Stránka s přehledem aktuálních kurzů zobrazuje pouze ty měnové páry, které uživatel má oprávnění zobchodovat. Zobchodovat může pouze ty měny, ve kterých má vedeny účty. Pokud tedy má účet CZK, EUR a USD, tak má možnost obchodovat s měnovými páry CZK-EUR, CZK-USD a EUR-USD.



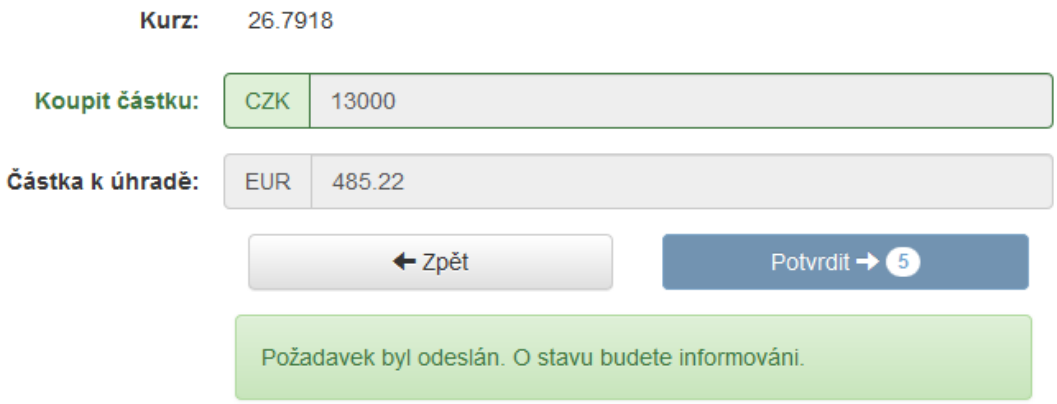
Obrázek 14. Platební portál – aktuální kurzy měn

Po kliknutí na tlačítko s nákupním, nebo prodejním kurzem, se zobrazí formulář pro zadání objednávky (viz Obrázek 15). Uživateli je zde zobrazen aktuální disponibilní zůstatek na svém účtu, ze kterého se budou odesílat peníze a aktuální kurz. Jakmile zadá částku, kterou chce zobchodovat, tak se provede kontrola vůči disponibilnímu zůstatku. V případě, že uživatel překročí disponibilní zůstatek, tak je zobrazeno upozornění a není možné formulář odeslat. Po odeslání formuláře se zobrazí obrazovka s rekapitulací objednávky (viz Obrázek 16), na které běží časový limit 10 vteřin. Po tuto dobu je systémem držen domluvený kurz. Pokud uživatel v časovém limitu klikne na tlačítko „Potvrzuji“, tak dojde k odeslání objednávky do systému. Uživatel je následně přesměrován na stránku přehledu aktuálních kurzů.

V případě, že uživatel v limitu nestihne potvrdit objednávku, tak dojde k přesměrování zpět na formulář objednávky a kurz se začne zase aktualizovat.

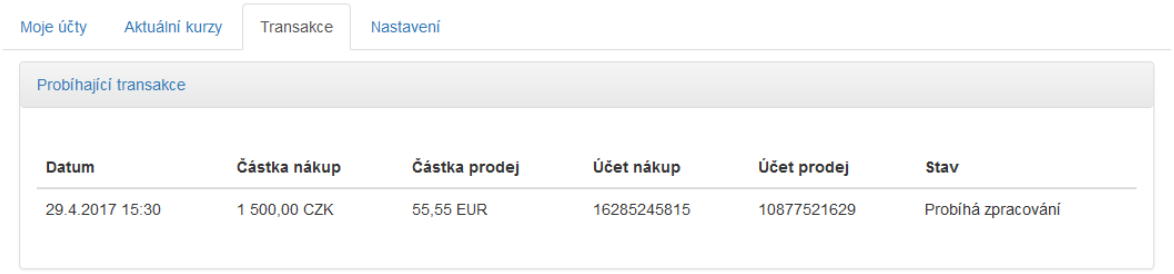


Obrázek 15. Platební portál – formulář pro uzavření nového obchodu



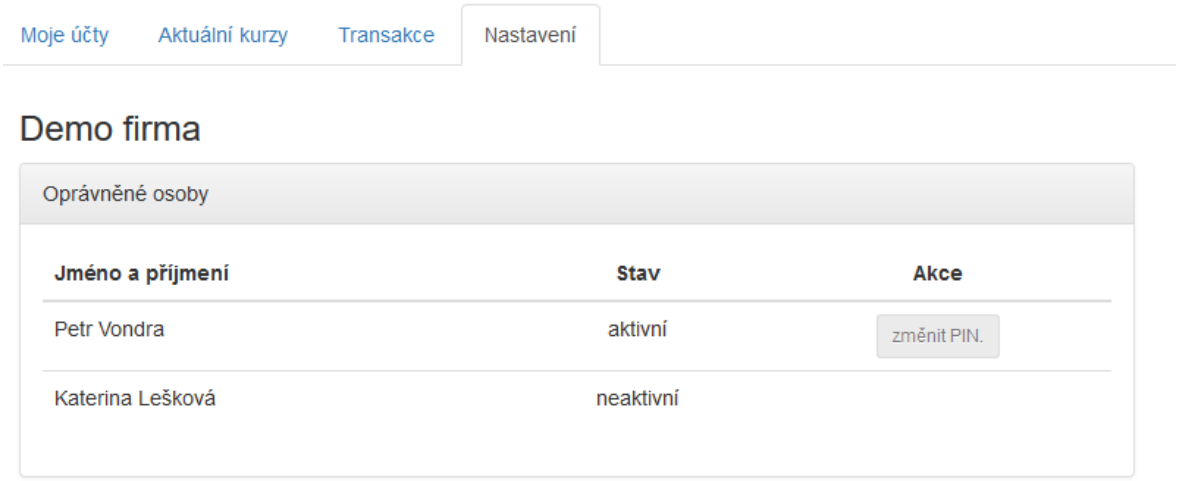
Obrázek 16. Platební portál – rekapitulace objednávky

Na stráce přehledu transakcí se zobrazuje aktuální stav posledních transakcí. V případě, že dojde k chybě při vypořádání, tak je transakce zrušena. Stav transakce uživatel uvidí v přehledu (Obrázek 17).



Obrázek 17. Platební portál – přehled transakcí

Na stránce nastavení (viz Obrázek 18), je zobrazen seznam oprávněných osob. Uživatel má možnost si u svého přístupu změnit PIN, který slouží pro přihlášení prvním faktorem.



Obrázek 18. Platební portál – stránka nastavení

6.6.9.2 Mobilní aplikace

Mobilní aplikace bude sloužit zejména pro vícefaktorovou autentizaci. Bude přijímat push notifikace a uživatel bude pomocí ní schvalovat, nebo zamítat žádosti o přístup.

Dle požadavků byla aplikace rozdělena na obrazovky:

- Seznam požadavků ke schválení (viz Obrázek 19)

- Detail požadavku na schválení (viz Obrázek 20)

Aplikace je poměrně jednoduchá. Při příchodu push notifikace se zobrazí na mobilu oznámení o příchodu zprávy. Uživatel na zprávu klikne a tím se mu spustí aplikace pro schvalování. Pokud je aplikace zaregistrována, tak proběhne automaticky přihlášení na rozhraní platebního systému. Aplikace si stáhne seznam požadavků na schválení, který zobrazí (viz Obrázek 19)

Obrázek 19. Mobilní aplikace

seznam žádostí k odsouhlasení

Uživatel klikne na vybranou položku v seznamu a zobrazí se její detail. Na stránce s detailem jsou úplné informace o žádosti a tlačítka souhlasím „ANO“ / „NE“. Uživatel klikne na tlačítko, žádost je vyřízena a zobrazí se opět obrazovka se seznamem žádostí. Seznam žádostí se zaktualizuje a vyřízené žádosti ze seznamu zmizí.

Obrázek 20. Mobilní aplikace

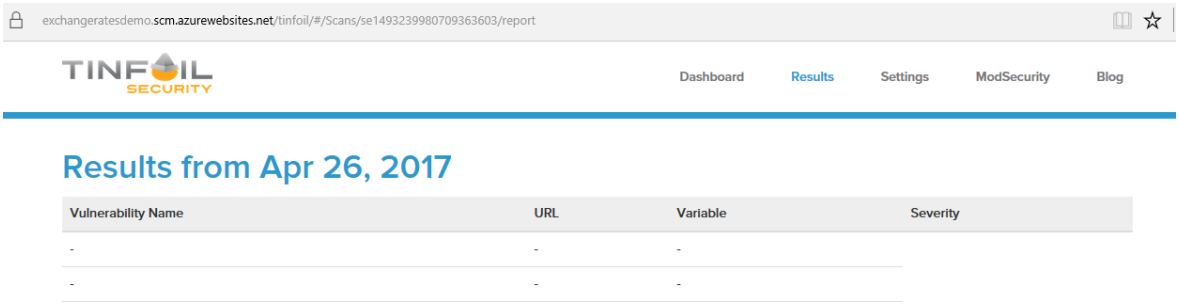
detail žádostí k odsouhlasení

6.7 Testy

6.7.1 Odolnost platebního portálu proti průniku

Odolnost byla otestována pomocí produktu Web Vulnerability Scanning for Azure App Service, které poskytuje společnost Tinfoil Security. Tato služba patří k základním, které podporují testování založené na doporučení v OWASP Top 10 Project pro webové aplikace [11]. Výhodou je, že je přímo integrována do platformy MS Azure, takže ji lze jednoduše zaktivovat a ovládat bez nutnosti složité instalace a konfigurace. Přímé propojení služby s MS Azure rovněž zaručuje, že penetrační testy prováděné touto službou není nutné oznamovat předem na bezpečnostním centru služby MS Azure. Podrobný postup aktivace služby a její

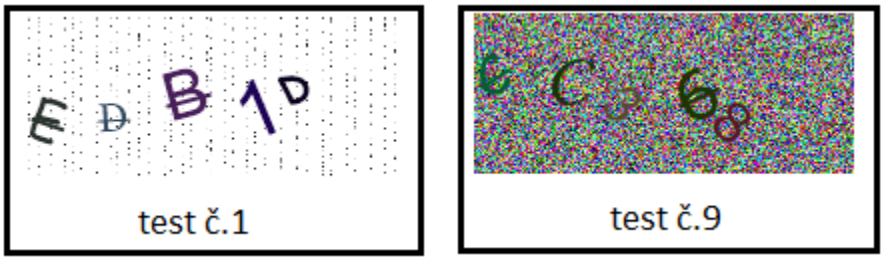
spuštění je popsáno na stránkách MS Azure [32]. Provedený test nenašel žádný problém (viz Obrázek 21).



Obrázek 21. Výsledek základního penetračního testu pomocí Tinfoil Security

Pro první indikaci problémů ve fázi vývoje je takový základní test postačující. Před nasazením do produkce bude žádoucí použít komplexnější nástroj. Jednou z možností je například produkt SecurityCenter SC od společnosti Tenable Network Security. Detailní popis funkcionality je dostupný na stránkách produktu [33].

Byl proveden test opakovaného pokusu o přihlášení s chybnými údaji. Postupně s každým neúspěšným pokusem o přihlášení do platebního portálu se zvýšila obtížnost CAPTCHA kódu (viz Obrázek 22).



Obrázek 22. Test zvyšování obtížnosti CAPTCHA

6.7.2 Odolnost mobilní aplikace proti zneužití

Jako první je potřeba provést identifikaci rizik. Jaké data může útočník získat a co s nimi může provést? Jakou největší škodu může provést? [34]. Mobilní aplikace slouží jako druhý faktor pro volbu autentizace přes mobilní aplikaci. Autentizace pomocí SMS není na mobilní aplikaci závislá. Při instalaci uživatel zadá jednorázové heslo, které umožní svázat unikátní ID mobilního zařízení s číslem smlouvy. Poté je jich heslo neplatné. Číslo smlouvy není

uloženo na mobilu. Přístupové údaje na webovou službu a do notifikační služby jsou uloženy na izolovaném datovém úložišti pevné paměti mobilu a zašifrováno pomocí RSA[11]. Obecně ale můžeme předpokládat, že útočník za vynaložení velkého úsilí, získal všechny výše uvedené informace. Mohl by s nimi provést pouze souhlas s přihlášením do systému přes webovou aplikaci. Na to by musel znát ještě číslo smlouvy a heslo. Oba tyto údaje v mobilu nikdy nebyly vyplněny, a tak je nepravděpodobné, že je získá. Nicméně pokud by přece jen získal i údaje z obou autentizačních faktorů, tak by získal pouze oprávnění provést směnu na jinou měnu za hotovost, která je na účtu klienta. Nemá žádnou možnost, jak peníze z účtu odeslat na jiný účet. V tomto krajním případě by mohla vzniknout škoda při výkyvu kurzu a na poplatcích za transakci, ale je to krajní případ a riziko je minimální.

Aplikace byla vytvořena jako multiplatformní. Každá z platforem má vlastní bezpečnostní pravidla a doporučení. Při vývoji bylo dbáno na následující:

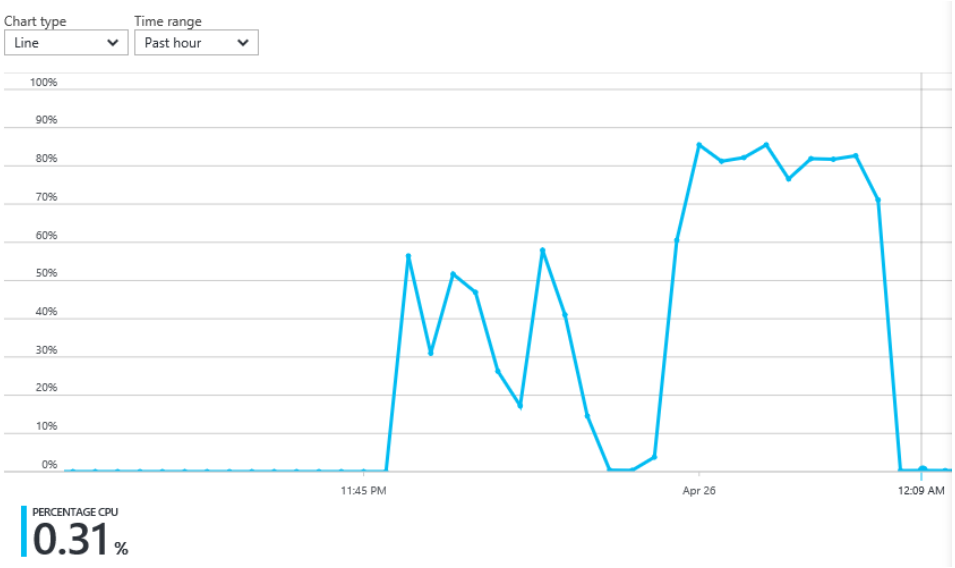
- Komunikace mezi aplikací a serverem je provozována na HTTPS protokolu a klient kontroluje validitu certifikátu serveru.
- Citlivá data jsou zkompilována v kódu, nebo na zašifrovaném a izolovaném úložišti
- V instalačním manifestu nastaveno omezení, že aplikaci není povoleno instalovat na SSD kartu.

6.7.3 Výkonové testy

Test automatického škálování byl proveden tak, že na vstup loadbalanceru byl spuštěn simulovaný provoz. Zátěž byla simulována pomocí aplikace httperf[12], která slouží pro generování zátěže na webové servery.

Aplikace byla nakonfigurována tak, aby provedla několik tisíc http dotazů na webové rozhraní. Dotazy byly proloženy pauzami. Výsledek zátěže byl monitorován portálu služby MS Azure. Služba má sekci „Metrics“ (Obrázek 23), která ukazuje zátěž systému. Nejedná se o zátěž na loadbalanceru, ale o zátěž CPU na jednotlivých instancích.

[11] RSA – metoda šifrování s pomocí veřejného a osobního klíče
[12] httpperf – šířen pod licencí GNU GPL. Zdrojové kódy jsou k dispozici na adrese https://github.com/httperf



Obrázek 23. Graf zátěže platebního portálu

Cílem testu bylo zjistit, zda automatické horizontální škálování funguje správně podle zvolené konfigurace. Funkci automatického škálování bylo možno ověřit v sekci „Activity log“ (viz Obrázek 24). V logu jsou vidět záznamy událostí zapínání a vypínání jednotlivých instancí. Z výpisu je vidět, že reakční doba systému při navyšování a snižování výkonu je 2 minuty. To je z pohledu požadavků na dostupnost přijatelné. V případě potřeby zajištění rychlejší reakce systému, by stačilo snížit práh maximálního povoleného zatížení instance. Nicméně doporučená hodnota 75 % je dostatečná, protože při testech zátěž CPU nedosáhla na 100% výkonu.

Obrázek 24. Výpis z logu služby automatického škálování

6.7.4 Odolnost proti selhání

Interní systémy jsou provozovány v režimu vysoké dostupnosti. Vysokou dostupností se rozumí takový režim, kdy je služba provozována na více serverech běžících paralelně vedle sebe s tím, že při výpadku převezme činnost jiný. V tomto režimu jsou provozovány tyto služby:

- Platební systém
- Vícefaktorová autentizace
- Služba pro periodickou aktualizaci kurzů

Na platformě MS Azure není možné simulovat výpadek hardware, který je v režimu vysoké dostupnosti, nicméně je možné provést simulované výpadky některých služeb, které komunikují s externími systémy. U nich se dá očekávat, že služba jako taková ve vysoké dostupnosti běží, ale nedá se připojit na systém třetí strany. Toto lze provést na těchto službách takto:

Vícefaktorová autentizace

Chyba: nelze odeslat autentizační kód vybranou metodou

Opatření: neproběhla autentizace, uživatel dostal možnost vybrat jinou metodu

Platební systém

Chyba: nepodařilo se spojit s rozhraním bankovních účtů u vypořádání pohledávek.

Opatření: transakce byla zrušena, uživatel byl vyzván, aby zkusil požadavek později

Systém periodické aktualizace kurzů

Chyba: selhalo připojení na zdroj dat

Opatření: zastavilo se uzavírání nových transakcí.

6.7.5 Celkové zhodnocení výsledků testů

Z testů vyplynulo, že řešení splnilo očekávání. Aplikace odolávala zvýšené zátěži i opakovaným pokusům o přihlášení. Jako další možností vylepšení je možnost rozšíření aplikace o hlášení podezřelé, nebo nežádoucí aktivity systému IPS.

6.8 Další možnosti zvyšování výkonu

Vyvinutý systém byl přizpůsoben tak, aby provoz nebyl finančně náročný, proto některé části nejsou výkonově optimální. V případě potřeby je však možné úpravou konfigurace výkon jednoduše zvýšit, aniž by bylo nutno měnit jádro systému:

- Server SQL je zapojen ve vysoké dostupnosti. Vyššího výkonu lze ale dosáhnout tak, že se spustí více instancí typu SLAVE, na které bude probíhat replikace. Tyto servery budou pouze pro čtení a umožní tak rozdělit zátěž. Každé službě bude pro čtení přiřazena jiná instance SQL serveru. Tímto způsobem sice poroste výkon, ale bude se současně snižovat odolnost proti selhání.
- Rekonfigurovat platební portál tak, aby backplane využíval REDIS místo SQL. To zvýší propustnost přenášených zpráv v platebním portálu.
- Spustit více instancí služby pro vyřizování objednávek, což zvýší množství paralelně probíhajících transakcí.

ZÁVĚR

Hlavním cílem práce bylo prozkoumat technologie, které by bylo možné využít pro implementaci škálování výkonu, odolnost proti výpadku a odolnost proti útoku. Byly upřesněny a formulovány požadavky tak, aby splňovaly uvedené cíle. Následně byly postupně prozkoumány technologie z oblasti uživatelského rozhraní, komunikační technologie a bezpečnostní rizika. Pro vývoj uživatelského rozhraní bylo zvoleno multiplatformní řešení, které umožňuje většině uživatelů bez omezení používat navržený systém. Jako uživatelské rozhraní bylo zvoleno webové rozhraní s responzivním designem a pro autentizaci druhým faktorem byla zvolena multiplatformní mobilní aplikace. Byly zhodnoceny výhody a nevýhody jednotlivých komunikačních technologií. Pro komunikaci částí, kde byl očekáván velký provoz, byla zvolena komunikace pomocí web soketu, který umožňuje plně duplexní provoz. Pro podporu škálování výkonu služby pro realizaci objednávek bylo zvoleno ukládání do fronty. Pro autentizaci druhým faktorem pak byla zvolena push notifikace, která umožňuje příjem zpráv na cílovém zařízení, i když aplikace není spuštěna. V závěru teoretické části byly probrány vybrané typy útoků a byla navržena ochrana proti nim.

V praktické části byla zvolena infrastruktura. Pro vývoj systému bylo využito cloudu, který umožňuje efektivně a rychle měnit konfiguraci systému podle potřeb vývoje. Jako výhoda byl zdůrazněn fakt, že prostředky jsou placeny formou pronájmu, což přináší velkou finanční úsporu. Drahé zařízení je možné aktivovat pouze na dobu nezbytně nutnou a cena za pronájem je tak minimální. V rámci návrhu infrastruktury byly pořízeny a nakonfigurovány jednotlivé komponenty, které byly využity při vývoji jednotlivých systémů. Pro produkční platformu bylo doporučeno řešení on-premise. Následně byl proveden návrh aplikace a implementace. Byly navrženy algoritmy procesů, které v systému probíhají a detailně byly popsány pomocí uživatelských scénářů. V závěru práce byly provedeny testy odolnosti proti přetížení a bylo navrženo několik možných řešení, jak výkon systému ještě více zvýšit. Bylo zhodnoceno ošetření bezpečnostních rizik a byly provedeny jednoduché penetrační testy.

Navržená aplikace splnila očekávání a teoretické i praktické výsledky bude možné využít při implementaci podobných systémů. Aplikace je napsána tak, aby ji bylo možné v budoucnu rozšířit, případně aby implementátor mohl tuto aplikaci jednoduše napojit na vlastní informační systém.

SEZNAM POUŽITÉ LITERATURY

[1] *Zákon: o platebním styku*. In: . Praha: Tiskárna Ministerstva vnitra, 2009, ročník 2009, částka 4174, číslo 284. Dostupné také z: https://www.cnb.cz/cs/legislativa/zakony/download/zakon_284_2009

[2] ČESKÁ REPUBLIKA. Zákon: o ochraně osobních údajů a o změně některých zákonů. In: *Sbírka zákonů*. Praha: Tiskárna Ministerstva vnitra, 2000, ročník 2000, částka 32, číslo 101. Dostupné také z: http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=3420

[3] ČESKÁ REPUBLIKA. *Sdělení ČNB o doporučení pro bezpečnost internetových plateb*. In: . Praha: Česká národní banka, 2013, ročník 2013, č. 6/1993. Dostupné také z: https://www.cnb.cz/cs/dohled_financni_trh/legislativni_zakladna/banky_a_zalozny/sdeleni_bezpecnost_internetovych_plateb.html

[4] *SystemOnLine: 7 kybernetických útoků, se kterými se setkáte* [online]. 2016, **2016**(9) [cit. 2017-04-11]. ISSN 1802-615X. Dostupné z: https://www.systemonline.cz/it-security/7-kybernetickych-utoku-se-kterymi-se-setkate.htm

[5] CategoryAttack - OWASP: common types of application security attacks. *OWASP* [online]. United States: OWASP, 2016 [cit. 2017-04-11]. Dostupné z: https://www.owasp.org/index.php/Category:Attack

[6] *Lupa.cz - server o českém Internetu: Weby českých bank ochromil DDoS útok* [online]. Praha: Internet Info, 2013, **2013**(6) [cit. 2017-04-11]. ISSN 1213-0702. Dostupné z: http://www.lupa.cz/clanky/web-ceske-sporitelny-neni-dostupny-vcetne-online-sluzeb-servis24/

[7] ANDERSON, Jonathan., John. MCREE a Robb. WILSON. *Effective UI*. 1st ed. Cambridge: O'Reilly, 2010. ISBN 059615478x.

[8] *Native Script: Build amazing iOS and Android apps with technology you already know* [online]. San Francisco: Progress Software Corporation, 2017 [cit. 2017-03-08]. Dostupné z: https://www.nativescript.org/

[9] *React Native: Build native mobile apps using JavaScript and React* [online]. San Francisco: Facebook, 2017 [cit. 2017-03-16]. Dostupné z: https://facebook.github.io/react-native/

[10] HERMES, Dan. *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals*. 1st. edition. New York: Springer Science, 2015. ISBN 978-1-4842-0215-9.

[11] OWASP Top Ten Project. *OWASP: the free and open software security community* [online]. United States: OWASP, 2017 [cit. 2017-04-26]. Dostupné z: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[12] GRIGORIK, Ilya. *High-performance browser networking*. First release. Sebastopol: O'Reilly, 2013. ISBN 9781449344764.

[13] *React: A JavaScript library for building user interfaces* [online]. San Francisco: Facebook, 2017 [cit. 2017-02-18]. Dostupné z: https://facebook.github.io/react/

[14] *AngularJS* [online]. Mountain View: Google, c2010-2017 [cit. 2017-02-19]. Dostupné z: https://angularjs.org

[15] *Knockout: Simplify dynamic JavaScript UIs with the Model-View-View Model (MVVM) pattern* [online]. San Francisco: Steven Sanderson, c2010-2017 [cit. 2017-02-27]. Dostupné z: http://knockoutjs.com

[16] *Adobe PhoneGap: Build amazing mobile apps powered by open web tech* [online]. San Jose: Adobe, 2016 [cit. 2017-02-19]. Dostupné z: http://phonegap.com/

[17] *Bootstrap: The world's most popular mobile-first and responsive front-end framework* [online]. San Francisco: Bardi Harborow, 2016 [cit. 2017-03-25]. Dostupné z: http://getbootstrap.com/

[18] *Mobile Angular UI: Build HTML5 Mobile Apps with Bootstrap and Angular JS* [online]. San Francisco: Maurizio Casimirri, 2016 [cit. 2017-02-20]. Dostupné z: http://mobileangularui.com/

[19] Madskristensen-BundlerMinifier: Visual Studio extension. *GitHub* [online]. San Francisco: Mads Kristensen, 2017 [cit. 2017-04-25]. Dostupné z: https://github.com/madskristensen/BundlerMinifier

[20] OWASP Automated Threats to Web Applications. *OWASP* [online]. United States: OWASP, 2016 [cit. 2017-03-11]. Dostupné z: https://www.owasp.org/index.php/OWASP_Automated_Threats_to_Web_Applications

[21] J.D. MEIER, Alex. How To: Protect From SQL Injection in ASP.NET. *Patterns & practices Developer Center*. 2005. Dostupné také z: https://msdn.microsoft.com/en-us/library/ff648339.aspx

[22] Cisco Next-Generation Intrusion Prevention System (NGIPS). *Cisco* [online]. San Jose: Cisco Systems, © 1992-2017 [cit. 2017-03-21]. Dostupné z: http://www.cisco.com/c/en/us/products/security/ngips/index.html

[23] FortiOS: Intrusion Prevention System. *Fortinet* [online]. Sunnyvale: Fortinet, 2014 [cit. 2017-03-21]. Dostupné z: http://docs.fortinet.com/uploaded/files/2028/inside-fortios-ips-52.pdf

[24] ČDT-ANTIDDOS. *ČD-Telematika* [online]. b.r. [cit. 2017-03-22]. Dostupné z: http://www.cdt.cz/cz/cdt-antiddos-1136/

[25] COPELAND, Marshall. *Microsoft Azure: planning, deploying, and managing your data center in the cloud*. 1st ed. Berkeley, CA: Apress, 2015. Expert's voice in Microsoft Azure. ISBN 1484210441.

[26] Clusterová řešení. *Master internet*. 2015. Dostupné také z: https://www.master.cz/clusterova-reseni/

[27] Bank of America Introduces Fingerprint and Touch ID Sign-in for Its Mobile Banking App. *Bank of America* [online]. Bank of America, 2017 [cit. 2017-04-21]. Dostupné z: http://newsroom.bankofamerica.com/press-releases/consumer-banking/bank-america-introduces-fingerprint-and-touch-id-sign-its-mobile-ban

[28] The new face of authentication: Yours: CEO Mobile® biometrics. *Wells Fargo* [online]. San Francisco: Wells Fargo Bank, 2017 [cit. 2017-04-21]. Dostupné z: http://wholesale.wellsfargobank.com/biometrics

[29] Evropané jsou připraveni na biometrické ověřování plateb. VISA [online]. *Visa Praha*. 2016. Dostupné také z: https://www.visa.cz/o-nas/tisk-media/-1482279

[30] Introduction to Scaleout in SignalR. *Microsoft Docs* [online]. Redmont: Microsoft, 2017 [cit. 2017-04-25]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/signalr/overview/performance/scaleout-in-signalr

[31] *Introduction to Entity Framework* [online]. Redmont: Microsoft, 2016 [cit. 2017-03-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx

[32] Web Vulnerability Scanning for Azure App Service powered by Tinfoil Security. *Microsoft Azure* [online]. Redmond: Microsoft, 2015 [cit. 2017-04-26]. Dostupné z: https://azure.microsoft.com/en-us/blog/web-vulnerability-scanning-for-azure-app-service-powered-by-tinfoil-security/

[33] Security Center SC: Vulnerability Management & Analytics. *Tenable* [online]. Dublin: tenable, 2017 [cit. 2017-04-26]. Dostupné z: https://www.tenable.com/products/securitycenter

[34] VELU, Vijaj. *Mobile Application Penetration Testing*. 1st ed. Birmigham: Pact Publishing, 2016. ISBN 978-1785883378.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
BGP	Border Gateway Protocol
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
ČNB	Česká národní banka
CPU	Central Processor Unit
DNS	Domain Name System
DoS	Denial of Service
DDoS	Distributed Denial of Service
HTML	Hypertext Markup Language
HTTPS	HyperText Transfer Protocol Secure
IaaS	Infrastructure as a Service
IP	Internet protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
ISO	International Organization for Standardization
JSON	Javascript Object Notation
OSI	Open Systems Interconnection
OWASP	Open Web Application Security Project
PCL	Portable Class Library
REST	Representational State Transfer
RSA	Rivest, Shamir, Adleman – asymetrická šifra
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SQL	Structured Query Language

SSL	Secure Socket Layer
UDDI	Universal Description and Discovery Integration
UI	User Interface
UX	User Experience
XML	Extensible Markup Language

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD-ROM

CD-ROM obsahuje následující adresáře:

- **ConfirmationApp**
  Mobilní aplikace pro autentizaci uživatele druhým faktorem
- **ExchangeRatesLib**
  Knihovna, která umožňuje sdílet kód datových struktur mezi jednotlivými aplikacemi
- **StreamServiceDemo**
  Služba pro streamování kurzů měnových párů
- **OrderService**
  Služba pro vyřizování objednávek
- **OrderService.Tests**
  Jednotkové testy pro OrderService
- **WebAPI**
  Webové rozhraní platebního portálu, registrační služba REST.
- **WebAPI.Tests**
  Jednotkové testy webového rozhraní