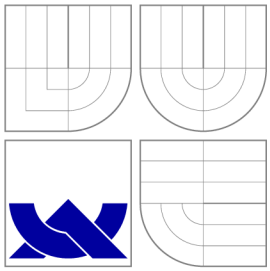
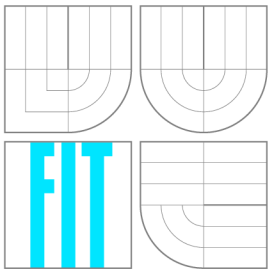


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYTÉM PRO VÝKONNOSTNÍ A ZÁTĚŽOVÉ TESTOVÁNÍ

PERFORMACE AND STRESS TESTING TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DANIEL JAVORSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FRANTIŠEK ŠČUGLÍK, Ph.D.

BRNO 2016

Abstrakt

Tato práce se zaměřuje na výkonnostní a zátěžové testování aplikačního serveru Xtend společnosti Xura, Inc. Vývoj softwaru a teoretické znalosti týkající se testování jsou rozebrány v úvodních kapitolách, stejně jako jednotlivé služby, které poskytuje tento aplikační server. Důraz je kladen převážně na implementaci systému na výkonnostní a zátěžové testování tohoto serveru . Tento systém umožňuje vytvářet dlouhodobé i krátkodobé testovací scénáře a jeho výstup slouží vývojářům tohoto aplikačního serveru. Součástí práce je i popis jednotlivých služeb, které poskytuje aplikační server Xtend a výsledky testování tohoto serveru.

Abstract

This thesis is concerned about performance and stress testing of Xtend product developed by Xura, Inc. Software development knowledge, theoretical knowledge of testing and testing tools are described in opening chapters together with key features and services provided by Xtend. Emphasis was put on implementation of performance and stress testing tool, which focuses on short-term and long-term testing scenarios and output of this tool serves Xtend developers. Part of this thesis also focuses on results of stress and performance tests.

Klíčová slova

výkonnostní testování, zátěžové testování, Xura, Xtend, vývoj softwaru

Keywords

performance testing, stress testing, Xura, Xtend, software development

Citace

JAVORSKÝ, Daniel. *Sytém pro výkonnostní a zátěžové testování.* Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ščuglík František.

Sytém pro výkonnostní a zátěžové testování

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ščuglíka.

.....
Daniel Javorský
25. května 2016

Poděkování

Chtěl bych převážně poděkovat svému vedoucímu diplomové práce za odborné vedení při vypracovávání této práce. Dále bych chtěl poděkovat rodině a všem známým, kteří mně při studiu podporovali.

Obsah

Kapitola 1

Úvod

Na úvod této diplomové práce budou vysvětleny základní důvody k provádění výkonnostního a zátěžového testování. Pokud zákazník objednává určitý produkt bez bezprostředního setkání se s výrobcem, nebo dokonce i tehdy, tak mezi první věci, které by měl daný zákazník zjistit, patří primárně všechny prerekvizity a potřebný materiál (v případě softwaru je to hardware). Mnoho zákazníku si kupuje zboží bez toho, aniž by zjistili, jestli je vůbec využijí. Toto neplatí pouze u běžných věcí, jako je oblečení, sportovní potřeby apod., ale také u softwarových produktů. Prodejce, kterému záleží na spokojenosti zákazníka a chce si udržet svou klientelu by měl mít přehled o tom, co daný zákazník opravdu využije. Názorným příkladem může být poté software. Pokud nebude mít zákazník dostatečně výkonný hardware a nainstalovány všechny podpůrné programy a knihovny, tak mu daný produkt poběží s problémy, nebo nebude vůbec fungovat. Toto je velmi důležitá část v této době, protože může potenciálně zvýšit zisky a pomoc přimět zákazníka vybrat a koupit si daný produkt, což zvýší zisky firmě, která jej poskytuje.

Při prodeji dražších a komplexnějších softwarů je tedy nutností přidat i seznam podporovaného a doporučeného hardwaru a systémových požadavků. Pokud má ale zákazník již existující hardware a nemá možnost nebo finance si zakoupit optimální hardware, který je doporučen výrobcem, tak se právě zajímá o výkonnostní a zátěžové testy. Tyto testy jsou prováděny na referenčním stroji s předem danými nainstalovanými knihovnami a programy. Po navrhnutí, provedení a vyhodnocení testů je možné poté zákazníkovi navrhnout hardware, popř. služby daného softwaru tak, aby vyhovoval jeho potřebám a byl s ním spokojený. Tato diplomová práce se zabývá testováním a vývojem systému pro automatizované testování aplikačního serveru Xtend od společnosti Xura, Inc. Výše zmíněné důvody jsou také důvodem, proč tato diplomová práce vznikla, nicméně existují i další.

Samotné výkonnostní a zátěžové testování aplikačního serveru Xtend musí proběhnout v laboratorních podmínkách, protože zvýšená zátěž u zákazníků by nebyla žádoucí. K tomuto účelu slouží virtuální stroje, které ovšem nedostahují takových rychlostí a parametrů, jako samostatný hardware. Z tohoto důvodu se bude daná diplomová práce zaobírat také testováním na virtuálních strojích s různým počtem jader, pamětí a místem na disku. Kromě virtuálních strojů bude provedena i analýza výsledků na hardwaru, který bude obsahovat produkt Xtend a všechny potřebné knihovny k jeho běhu. U koncových zákazníku produkt Xtendu běží převážně na samostatném hardwaru, ale pro testovací účely je důležité znát i výsledky výkonnostních a zátěžových testů na virtuálních strojích.

Aby bylo možné provést výkonnostní a zátěžové testy, je potřeba znát alespoň základní teorii, která je potřebná k samotnému vývoji softwaru, proto v této práci budou představeny základy problematiky vývoje softwaru s různými typy testování, stejně tak jako různé

metodiky vývoje softwaru, ať už se jedná o agilní metody nebo ne.

Pro výkonnostní a zátěžové testy tohoto aplikačního serveru se použíjí speciální programy, které simulují zátěž v síti a pomocí speciálních systémových utilit se poté sbírají relevantní data k tomuto typu testování. Jednotlivé scénáře budou nastíněny v následujících kapitolách, stejně jako služby, které tento aplikační server poskytuje a které jsou hodně využívány u zahraničních operátorů. Aplikace, která bude generovat provoz na daném serveru, bude také popsána společně s detailním popisem webové prezentace. Aplikace, která bude vyvinuta bude zahrnovat plánování krátkodobých i dlouhodobých testovacích scénářů. Tato prezentace bude schopná zobrazovat uživatelům výsledky jednotlivých testovacích scénářů a bude podporovat tvorbu uživatelů a uživatelskou správu.

Kapitola 2

Vývoj softwaru a testování

Tato kapitola se zabývá nutnými body z vlastního vývoje softwaru, jehož základy jsou důležité pro pochopení samotného testování a informací s tím spjatými. Po zkráceném úvodu o vývoji softwaru následuje teorie o testování softwaru. V této části budou také vyřčeny důležité body testování a klíčové body, které se ho týkají. V dané části této kapitoly bude uveden přehled jednotlivých testovacích metod, kromě zátěžového a výkonostního testování, kterému bude věnována celá sekce 2.3.

2.1 Vývoj softwaru

Pro vývoj softwaru se v dnešní době používají různé metodologie. Postupem času vývoj softwaru bez jakýchkoliv pravidel přestal být žádoucím. Hlavními důvody bylo samozřejmě zefektivnění samotné tvorby softwaru. Pod zefektnivněním tvorby softwaru si lze představit kromě lepší výsledné kvality vyvýjeného softwaru, také jeho větší znovupoužitelnost, spravovatelnost, rychlost, stabilitu, velikost, cenu, bezpečnost a v neposlední řadě také menší počet chyb na určitý počet řádků kódu. Toto nejsou všechny parametry, které zahrnují kvalitu softwaru, ale jsou jedny z nejdůležitějších. Z těchto důvodů se začaly vymýšlet různé metodiky vývoje softwaru. Na toto téma vznikla již celá řada publikací a tzv. "best-practices". Odvětví, které se těmito věcmi zabývá se jmenuje *softwarové inženýrství*. Softwarové inženýrství se samozřejmě nezabývá pouze metodikami vývoje softwaru, které jsou nezbytné pro vývoj kvalitního softwaru, ale jedná se o jeho důležitou část. Metodiky vývoje softwaru se dělí na lehké (agilní) a těžké, podle toho, na kterou skupinu vývojářů se zaměřují. V případě agilních metodik se mluví věsmes o menších týmech a menších projektech, kde je kladen velký důraz na přínos jednotlivce. Naproti tomu druhou skupinu tvoří velké týmy v korporátních společnostech. Některé důležité metodiky vývoj softwaru se nazývají následovně [26]:

- Vodopádový model
- Prototypování
- Inkrementální vývoj

Tyto tři výše vybrané metodiky budou popsány, jelikož jsou nejčastěji používanými a to i v případě velkých firem, které vyvýjejí rozsáhlý a komplexní software.

2.1.1 Vodopádový model

Vodopádový model je sekvenčním modelem, kdy jednotlivé fáze následují za sebou. Je akceptovatelné překrývání se některých fází v průběhu vývoje. Během vývoje softwaru je kladen důraz na dobrou dokumentaci projektu a ověřování správnosti navržené architektury a jejích jednotlivých částí. Nejprve se začíná jednotlivými požadavky. Tyto požadavky jsou přezkoumány a na jejich základě se poté stanovuje doba trvání samotného vývoje a cena za danou službu nebo produkt. Tato fáze je velice důležitá, jelikož se od ní odvíjí poté i celková doba vývoje a kvalita výrobku a mnoho dalšího.

Po této fázi následuje navržení struktury produktu a jeho jednotlivých funkčních části. Implementace jednotlivých částí a ověření správnosti produktu následuje ihned poté.

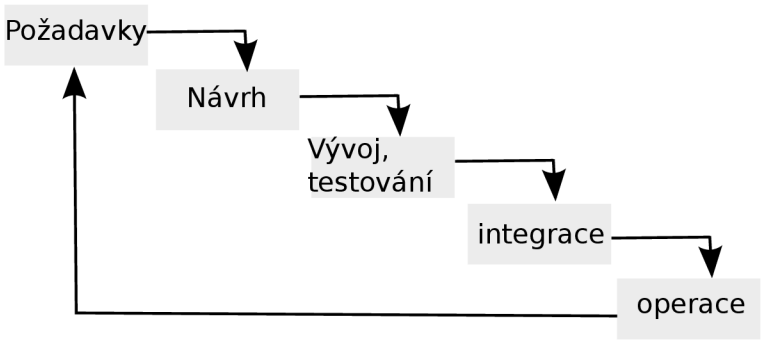
2.1.2 Prototypování

Základem prototypování je vytváření nekompletní verze, která se poskytne zákazníkovi během vývoje. Prototyp typicky zahrnuje pouze několik utilit, které budou později použity ve finálním produktu. Prototyp se také většinou typicky liší od výsledného produktu. Na základě připomínek k jednotlivým prototypům od zákazníka se upravuje vyvíjený produkt. Další výhodou vyvíjení prototypů je zpětná vazba od zákazníka ohledně správnosti produktu. Nevýhodou ovšem je práce, která je potřebná k opakovanému vytváření prototypů.

2.1.3 Inkrementální vývoj

V této metodologii prochází vytváření softwaru několika kroky, které se pravidelně opakují. Ve své podstatě probíhá podobně jako již popsaný vodopádový model, ale v kratších intervalech. Po každém intervalu se provádí testování, které má odhalit chyby již v počátcích vývoje, aby se minimalizovaly náklady spojené s její opravou v pozdějších fázích, tak jak je tomu u vodopádového modelu, kdy se provádí testování až výsledného produktu. Další výhodou je bezesporu také možnost zákazníka ovlivnit vývoj, protože po jednotlivách fázích vidí, jak je produkt vyvíjen a v jakém je aktuálně stavu. Nevýhoda tohoto modelu spočívá v náročnosti samotné organizace.

Produkt Xtend vyvíjený společností Xura, Inc. bude představen v další části textu, ale je nutno zmínit, že je právě vyvíjen převážně pomocí inkrementálního modelu, který je zde popsán a který je zjendodušeně zachycen na 2.1.



Obrázek 2.1: Ukázka inkrementálního modelu.

2.2 Standardní testování

Testování je v současné době klíčovým faktorem ve vývoji a následném prodeji softwaru. Testování, jako pojem, byl definován organizací IEEE, která se zabývá standardy, následovně: "*Testování je proces analýzy části softwaru sloužící ke zkoumání rozdílů mezi existujícími a požadovanými vlastnostmi a k vyhodnocení vlastností softwaru*"[3]. Cílem této diplomové práce je se zaměřit na výkonnostní a zátěžové testování produktu Xtend a vytvořit systém, který tuto práci automatizuje. Samotné testování je důležité z několika důvodů. Výkonostnímu a zátěžovému testování se bude věnovat sekce 2.3.

Testování jako takové poté zahrnuje i validaci a verifikaci. Při validaci a verifikaci je nutno položit pár základních otázek a to:

- Validace - Vytváří se správný produkt?
- Verifikace - Vytváříme správně daný produkt?

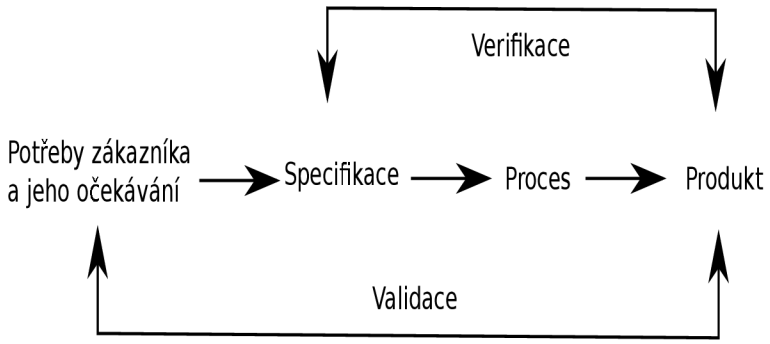
Validace podle standardu, který byl definován v [4], zjišťuje, jestli software a jeho asociované produkty a procesy:

- uspokojují požadavky na systém specifikované na konci každého vývojového cyklu.
- řeší zadaný problém (např. správně modelují daný fyzikální zákon).
- splňují požadovaný úmysl a uživatelské potřeby.

Na druhou stranu, pokud se mluví o verifikaci, tak ta byla definována v [4] následovně: Verifikační proces poskytuje objektivní důkazy, jestli software a jeho produkty a procesy:

- splňují požadavky (např. správnost, úplnost, konzistenci, přesnost) během všech vývojových fází.
- splňují standardy, požadavky a konvence během všech vývojových fází.
- úspěšně ukončí jednotlivé vývojové cykly a splňují podmínky pro započetí následujících vývojových cyklů.

Validace a verifikace se prolínají a vzájemně využívají svoje výsledky k lepší specifikaci analýzy, vyhodnocení a inspekce daného softwaru a jeho služeb a procesů. Názorná ukázka procesu validace a verifikace je znázorněn na 2.2.



Obrázek 2.2: Proces validace a verifikace v vývojovém cyklu.

Testování určuje samotnou kvalitu softwaru. První testování obvykle probíhá, jakmile existuje nějaká spustitelná verze daného softwaru. Poskytuje objektivní, nezávislý pohled na software, který je velice důležitý z mnoha hledisek. Jedním z nejběžnějších důvodů testování je snaha odhalit chyby v softwaru. Platí pravidlo, že pokud se odhalí chyba v návrhu softwaru, případně v implementaci, tak cena za její nápravu je mnohonásobně nižší, než v případě, že se odhalí chyby až při integraci nebo nasazení v reálném provoze [26]. Z hlediska testování softwaru jsou důležité ještě následující tři termíny:

- Chyba - chybný nebo chybící funkce v zdrojových souborech.
- Selhání - nefunkční modul nebo kód během spuštění.
- Funkční poruchy - funkce neprovádí stanovenou činnost.

2.2.1 Typy testování softwaru

Testování softwaru se rozděluje podle mnoha kritérií. Prvním rozdělením, které lze aplikovat na testování softwaru, je podle spustitelnosti samotného produktu. Z tohoto hlediska tedy posuzujeme [29]:

- statické - toto testování nepotřebuje, aby byl program spuštěn. Může tedy probíhat v jakémkoliv stádiu procesu vývoje softwaru. Během tohoto testování je velmi podstatná a důležitá nejen analýza zdrojových souborů, ale také přezkoumání dostupné dokumentace k danému softwaru. Při analýze zdrojových souborů je tedy snaha odhalit syntaktické chyby, popř. chyby, které zahrnují nesprávnou funkci produktu, jako je práce s pamětí, či špatná práce s proměnnými. Jako špatnou práci s proměnnými lze považovat jejich neinicializaci, špatné přiřazení datových typů apod. Lze zde využít také nástroje, které pomáhají s daným testováním jako jsou IDE (*Integrated Development Environment*), které zvýrazňují syntax.
- dynamické - dynamické testování oproti statickému vyžaduje spustitelnou verzi produktu. Při jeho testování dochází k interakci s právě spuštěnou verzí produktu. K jeho analýze můžou být použity nástroje, které umožňují pozastavovat spuštěný program a krokovat jeho vykonávání. Často jsou tyto standardně používané nástroje nazývány *laděnkami*, které měří vytížení CPU (*central processing unit*), RAM (*random access memory*), I/O (*input/output*) a jiných systémových parametrů.

Obě dvě tyto metody se kombinují, aby byla zajištěna správná funkčnost výsledného produktu. Další typy testování softwaru se již nerozdělují podle spustitelnosti, ale podle jiných parametrů. Jako další bude uvedeno rozdělení podle automatičnosti testů, tudíž jestli jsou testy prováděny automaticky nebo manuálně.

Manuální testování zahrnuje zkoumání softwaru a provádění testů bez použití nástrojů či automatizovaných skriptů. Při tomto testování také dochází k interakci uživatele s daným softwarem [34]. Oproti tomu automatické testování zahrnuje tesotvání softwaru pomocí automatických skriptů a nevyžaduje interakci s uživatelem během samotného testování. Obvykle se používají při velkém počtu testů, které by vyžadovaly příliš dlouhou dobu interakce uživatele s testovaným softwarem nebo při příliš velkém množství dat. Typicky jsou výsledky automatizovaných testů analyzovány uživatelem po jejich skončení. Automatické testy zahrnují především regresní testování, které je velmi důležitou složkou většiny větších produktů. Také produkt Xtend využívá regresní testování, aby se odhalily možné chyby v nových nebo změněných procesech tohoto produktu. Samotné regresní testování

bude popsáno dále. Při automatickém testování je také důležité, aby byl uživatel spraven o jeho výsledcích, tudíž musí být implementována nějaka zpětná vazba z tohoto testování. Jedna z nejpoužívanějších zpětných vazeb jsou zaznamenané logy o samotném průběhu testů, které mohou být později lehce analyzovány. Kromě logů se také využívají například emailové notifikace [32].

V rámci automatického testování se používají typicky frameworky. Testovací framework je vlastně spustitelné prostředí pro automatizované testy. Testovací framework je zodpovědný za definování očekávaného výsledku, spuštění jednotlivých testů a reportování výsledků. Měl by být nezávislý na aplikaci, lehce spravovatelný, udržovatelný a jeho rozšíření by mělo být jednoduché. Používají se následující typy frameworků [2]:

- Data Driven Automation Framework - vstupní data a očekávané výsledky jsou uloženy v samostatných souborech, takže spouštějící skript může spustit všechny testy s různými daty.
- Keyword Driven Automation Framework - direktivy, které se zde nazývají klíčovými slovy, jsou také v speciálním souboru, stejně jako data a řídí samotný průběh testu.
- Modular Automation Framework - je postaven na abstrakci jednotlivých testů, které poté tvoří větší celky.
- Hybrid Automation Framework - kombinace výše zmíněných frameworků.

Podle [34] jsou následující typy testů prováděny během testovací části ve vývojovém cyklu softwaru manuálně [27]:

- Akceptační testování - toto testování se volí v případě, že chceme zjistit, jestli vytvořený produkt splnil požadavky specifikované v předchozích fázích vývojového cyklu. Pokud jsou zadané podmínky splněny, tak je daný software připraven na své vydání a doručení koncovým uživatelům. Existuje několik typů tohoto testování, mezi které spadá: uživatelské, business, Alfa a Beta testování. Mezi parametry, které podléhají akceptačnímu testování se řadí: integrita a konverze dat, použitelnost, výkonnost, škálovatelnost, validita softwaru, dokumentace a mnoho dalších. Výsledkem tohoto typu testování je poté rozhodnutí o přijmutí produktu, tak jak byl definován na začátku, jeho odmítnutí nebo přijetí za splnění určitých modifikací daného softwaru.
- Testování bílou skříňkou - pomocí této metody se examinují datové struktury a samotná logika se zdrojovými soubory. Mezi její hlavní výhody řadíme odhalení chyb v skrytých částech kódu a odhalení kódu, který se nepoužívá a je tudíž zbytečný. Naopak nevýhoda může být relativně dlouhá doba examinace zdrojových souborů a nepozornost při kontrole programátorem.
- Testování černou skříňkou - narozdíl od předchozího testování zde nedochází k zkoumání kódu. V tomto případě jsou zajímavé pouze výsledky, které musí odpovídat specifikaci. Software se otestuje na určitých vstupech a výstupy by měly být shodné s očekávanými výsledky.
- Unit testy - využívány samotnými programátory k otestování jednotlivých částí modulů. Díky Unit testům jsou případné chyby odhaleny již v brzkých fázích, vylepšují samostatný návrh a refaktorizace kódu je mnohonásobně jednodušší.

- Systémové testování - využívá testování pomočí černé skříňky, které bylo definováno dříve. Na tomto testování se nejčastěji podílí nezávislý tým, který se neúčastnil samotného vývoje. Zkoumají se nejenom funkcionální závislosti, ale také nefunkcionální.
- Integrační testování - po proběhnutí Unit testů, které byly také objasněny v této kapitole, nastává období, kdy se testují moduly společně. Toto testování má za cíl odhalit případné chyby při samotné integraci jednotlivých modulů.
- Výkonnostní testování - toto testování slouží k určení výkonu samotného produktu a také k ověření základních a nezbytných parametrů stroje, který poté bude hostit danou aplikaci. Tento typ testování bude popsán dále podrobněji a důkladněji.
- Zátěžové testování - hlavním účelem provádění tohoto testování je určit limity výsledné aplikace. Stejně jako výkonnostní testování, tak i toto bude detailněji popsáno v dalších částech této práce.

Velmi důležitou složku při vývoji softwaru také hraje regresní testování. Ve velkých firmách je běžné a spouští se pravidelně, např. každý den. Slouží k periodickému otestování softwaru. Pokud jsou upraveny jednotlivé moduly, které již v systému existovaly, tak právě toto testování odhalí případné chyby, které mohly být zaneseny, a to jak v upravených částech, tak i v jiných modulech, které mohou být přímo navázány na modul, který byl upraven. Produkt Xtend je každodenně testován regresními testy a denně proběhnou stovky těchto testů, aby odhalily případné chyby. Regresní testy tedy provádějí verifikaci produktu.

Při předávání vyvinutého produktu zákazníkovi jsou důležité instalační testy. Tyto testy jsou prováděny kvůli kompatibilitě cílového prostředí s daným softwarem. Jedním z nejdůležitějších parametrů u těchto testů jsou samozřejmě hardwarové nároky a nároky na operační systém. Typicky bývají provedeny na různých platformách s různými parametry. Pro správný a hladký průběh instalačního testování by měly být provedeny i odinstalační testy [33].

V neposlední řadě bude v této části prodiskutováno ještě testování použitelnosti. Toto testování slouží k získání zpětné vazby od koncových uživatelů o produktu. Nejzajímavější a nejpodstatnější je zjistit, jak se cílovým uživatelům s tímto softwarem pracuje a jak složité je naučit se jej ovládat. Koncovým uživatelům bývají předloženy podklady ve formě dokumentace a zkoumá se nejen jejich učící křivka, ale také případné zotavení z chyby. Výsledky tohoto testování jsou poté analyzovány a případné změny na základě analýzy výsledku jsou zapracovány do stávajícího řešení.

2.3 Výkonostní a zátěžové testování

V této kapitole bude popsáno samotné výkonnostní a zátěžové testování, definovány základní pojmy a parametry, které budou sloužit k provedení tohoto testování a popsány jednotlivé systémové utility, které slouží k sběru užitečných dat o daném systému.

2.3.1 Teorie výkonnostního a zátěžového testování

Jedná se o testování, které má určit jak se daný systém chová v určitých situacích. Některé situace, které se vyskytují při výkonnostním a zátěžovém testování jsou dosaženy i v reálném světě a některé jsou pouze nasimulovány v laboratorním prostředí, aby se zjistilo, jak tento systém zvládá mezní situace. Systém může být také podroben tomuto testování, aby se

odhalily a odladily případné chyby a také aby došlo k validaci a verifikaci systému jako celku a v neposlední řade, aby byla otestována systémová škálovatelnost, spolehlivost a využití jednotlivých zdrojů systému [25].

Mezi nejčastější otázky zákazníků směřované na výrobce softwaru bezesporu patří: "Bude to dostatečně rychlé?", "Budou podporováni všichni moji klienti, nebo se to nějakých nedotkne?", "Co se stane pokud daný produkt selže? Jak to ovlivní moje klienty?"a "Co musím udělat, abych případně zvýšil počet uživatelů produktu?". Mezi lidmi bez většího vzdělání v oblasti výkonnostních a zátěžových testů jsou asociovány následující spojení [5]:

- dostatečná rychlost = výkonnostní testování.
- stačí na pokrytí dosavadního nebo očekávaného počtu zákazníků = zkušební testování.
- došlo k chybě a jak ji napravit = zátěžové testování.
- budoucí růst v počtu aktivních uživatelů = kapacitní testování.

Existuje několik typů samotného výkonnostního a zátěžového testování, nejběžnější typy tohoto typu testování byly zmíněny dříve v této kapitole a jejich detailní popis se nachází v následujících podkapitolách .

Výkonnostní testování

V angličtině také známé pod pojmem *Performance testing* je technika, která slouží k určení nebo validování rychlosti, škálovatelnosti a stability.

Toto testování tudíž také slouží k propagaci vyvíjeného produktu, jelikož zákazníci chtějí vědět, jak rychlý a stabilní jejich potenciální produkt je. Kromě tohoto účelu však také slouží k zjištění, jak moc se realita odlišuje od původního návrhu a očekávání z hlediska výše zmíněných vlastností.

Toto testování ovšem nepokrývá objevování chyb, které se vyskytují pod určitou zátěží. Pokud také není dobře naplánováno, tak to může vést ke zkreslujícím výsledkům, například z důvodu malého počtu testovacích scénářů vzhledem k celému produktu. V neposlední řadě se také liší výsledky v závislosti na použitém testovacím harwaru. Tomuto se lze vyhnout pouze testováním na cílovém hardwaru, protože při testování a porovnávání referenčního stroje dojde vždy k jistému stupni odchylky [5].

Zkušební testování

Zkušební testování nebo také *Load Testing* se provádí k verifikaci chování při normální a extrémní zátěži. Je prováděno, aby se verifikovalo, že daná aplikace nebo server splňuje všechny požadované vlastnosti. Tyto vlastnosti bývají velmi často specifikovány v SLA (*Service Level Agreement*). Tento typ testu tedy umožňuje měření času odpovědi od serveru nebo aplikace, průchodnost a využití jednotlivých alokovatelných zdrojů a v neposlední řadě slouží také k identifikaci slabého místa. Mezi nevýhody samotného zkušebního testování určitě patří nutnost porovnání výsledků těchto testů mezi sebou.

Endurance testing neboli odolnostní testování je podmnožinou zkušebního zatížení. Ve své podstatě to je výkonnostní testování, které je zaměřeno na validaci výkonnostních charakteristik daného produktu, který je pod určitou zátěží, která je očekávaná v reálném provoze. Toto testování trvá zpravidla delší časový úsek. Výstupy tohoto testování poté mohou být veličiny určující čas mezi výpadky (*Mean Time Between Failure - MTBF*) a čas do výpadku (*Mean Time To Failure - MTTF*) [5].

Zátěžové testování

Anglickým názvem *Stress Test* slouží k otestování produktu pokud je na něj vyvýjena větší zátěž, než je očekávaná v reálném provoze a to dokonce nad nejvýše očekávané limity. Cílem tohoto testování je odhalit chyby, které nastanou pouze při takto velké zátěži. Mezi příčiny, které mohou způsobit chyby, které se právě projevují pouze při velké zátěži, se řadí synchronizace, úniky paměti či tzv. závody o zdroje systému nebo produktu. Pomáhá odhalit slabé místa daného produktu a ukazuje samotné chování produktu při nadstandardně velké zátěži. Jelikož ale zátěž, která se generuje při tomto testování není primárně očekávána v normálním provoze, tak může dojít k narušení infrastruktury zákazníka, pokud od ní není testovací prostředí odděleno. Někteří obchodníci, kteří uvažují o daném produktu nemusí mít dokonce vůbec zájem o tento typ testování, jelikož se nepředpokládá, že by byl daný produkt vystaven takové zátěži.

Vrcholové testování neboli Spike Testing je poté podmnožinou zátěžového testování. V tomto případě dochází k opakovanému zvětšování zátěže na krátky časový úsek nad určitou hranici [5].

Kapacitní testování

Anglickým názvem *Capacity Test* slouží ke zjištění, kolik uživatelů, entit nebo transakcí může být obsluhováno a systém jako takový musí stále splňovat předem vytyčené vlastnosti. Společně s plánováním kapacity, které je prováděno na začátku a které slouží k pozdějšímu potenciálnímu růstu v počtu uživatelů nebo transakcí v systému, je provedeno kapacitní testování. Jedním z příkladů může být zjištění kolik dalších výpočetních jednotek či pamětí je potřeba pro růst uživatelů nad předem stanovenou hranici. Další výhodou tohoto testování je také zjištění škálovatelnosti produktu a jestli se májí rozměry daného produktu zvýšit nebo naopak snížit. Na druhou stranu je bohužel velmi těžké navrhnout správně tyto testy, protože bývají velmi komplexní. Hlavním nedostatkem přesto bývá neschopnost validovat všechny aspekty kapacitního testování pomocí samotných testů v čase, kdy jsou nejvíce potřeba [5].

2.3.2 Parametry testování aplikačního serveru Xtend

Jak již bylo zmíněno tak při samotném výkonnostním a zátěžovém testování je systém podroben zátěži. Jelikož aplikační server Xtend slouží k manipulaci s SMS zprávami, tak je v jeho případě generován provoz, který má simulovat reálné použití tohoto aplikačního serveru. Dochází tedy ke generování SMS zpráv s využitím jedné z předem definovaných služeb. Samotné testovací případy budou nastíněny v dalších kapitolách. Základním parametrem, který tento aplikační server musí splňovat, je počet průchozích zpráv za sekundu. Následující pojmy jsou klíčové pro porozumění samotným výkonnostním a zátěžovým testům aplikačního serveru Xtend:

- Uzel (Node) - uzel je typicky hardware nebo virtuální stroj na kterém běží daná komponenta nebo komponenty.
- Referenční server (Reference server) - fyzický server, který byl zvolen pro testování. Většina serverů má označení SPECint2006 a SPECint_rate2006 přidělené podle [28]. Tyto servery mohou sloužit k určení škálovatelnosti na hardwaru, na kterém nebylo prováděno samotné testování, ale je to požadováno od zákazníka.

- Komponenta (Component) - komponenta v kontextu výkonnostního a zátěžového testování aplikačního serveru Xtend je software, který se může samostatně nainstalovat na daný hardware a může být spuštěn samostatně. Typickým příkladem takovéto komponenty může být MCO.
- Služba (Service) - služba je role nebo použití, které poskytuje komponenta koncovému uživateli nebo jiné komponentě. Komponenta může mít několik služeb a tyto služby běží typicky samostatně a mohou být vypnuty.
- Rys nebo vlastnost (Feature) - je specifická vlastnost služby, která může být vypnuta nebo zapnuta, podle požadavků a potřeb zákazníka. Rys nemůže fungovat samostatně a je vždy závislý na dané službě. Příkladem rysu může být tzv. Antiflood (ochrana proti zahlcení).
- Testovací scénář (Test Case) - testovací scénář určuje jakou službu, s jakými rysy a na jakém zařízení testovat.
- Agregovaný výsledek testu (Aggregate Test Result) - tyto výsledky obsahují již spočtenou propustnost a náročnost na zdroje. Výsledky, které jsou zahrnuty, pocházejí z daného scénáře, který byl otestován. Nejmenší doba a také nejčastější, kterou daný test poběží je právě jedna hodina a tudíž výsledky zahrnují využití zdrojů během této hoiny.
- Propustnost (Throughput) - nejvyšší možný počet poslaných zpráv za sekundu, kterého je možno dosáhnout během daného scénáře, aniž by to ovlivnilo funkčnost dané komponenty.
- Zdroje (Resources) - sdílená součást systému, která je využívaná během testu komponentou. Měřenými hodnotami jsou využití CPU, RAM, disku a vytížení sítě.
- CPU - v kontextu o CPU se rozlišuje zapnutí nebo vypnutí hyper-threadingu. Pokud je hyper threading zapnutý, tak při referenci o CPU se myslí jedno hardwarové vlákno a pokud je hyper threading vypnutý, tak se myslí samotné jádro.
- Dimenzování (Sizing) - je kalkulace zahrnující fyzické servery nebo virtuální mašiny a jejich zdroje na daný produkt podle požadavků zákazníka.

Při samotném testování samozřejmě dochází ke zjišťování slabých míst, které jsou také zaznamenány v jednotlivých testech. Při posílení těchto slabých míst by poté došlo k zvětšení propustnosti systému.

Testovací strategie

Z hlediska výkonnostního a zátěžového testování aplikačního serveru Xtend se rozlišují tři přístupy k samotným testům. Způsob testování závisí na náročnosti a přizpůsobení samotné konfigurace dané služby tohoto produktu. Tyto typy testování jsou popsány níže:

- Generický profil - ve své podstatě to znamená, že test dané služby proběhne s základní konfigurací, která je typicky určena zákazníkem. Tento způsob testování se uplatní, pokud většina zákazníků používá danou službu se stejnými nebo velice podobnými rysy. Mezi výhody této strategie bezesporu patří její jednoduchost na konfiguraci a změření jak výkonnu samotné služby tak jejích rysů. Naopak mezi nevýhody patří neznalost dopadu jednotlivých rysů na danou službu.

- Testování rysů - toto testování nejprve testuje danou službu bez jakýchkoliv dalších zapnutých rysů. Tyto rysy se později postupně zapínají a měří se znova náročnost samotného rysu s danou službou. Tato technika se používá, pokud zákazníci používají stejnou službu, ale obecně s jinými zapnutými rysy. Výhodou tohoto testování jsou přesnější výsledky, ale nevýhodou je náročnost konfigurace samotného produktu pro individuální nastavení.

- Kombinovaná strategie - tato strategie kombinuje předešlé dvě, které byly zmíněny. Jejím principem je testování více rysů najednou. Rysy služby, které jsou náročné na provádění se testují zase zvlášť.

Měřené hodnoty

Rozlišují se dva druhy měření hodnot a to statické a dynamické. Pokud je řeč o staticky měřených hodnotách, pak jsou to následující zdroje:

- Využití RAM - pokud nejsou posílány žádné zprávy a daný produkt je spuštěn, pak využití RAM je právě minimum potřebné k běhu samotného systému.

- Velikost nainstalovaného systému - zákadní verze produktu s konfiguračními a spustitelnými soubory.

Kromě těchto dvou staticky měřených hodnot jsou další hodnoty měřeny dynamicky. Pro samotné dimenzování je použita interní aplikace firmy Xura,Inc. zvaná *Crystal*, která vyžaduje data zobrazená v tabulce 2.3 pro svůj výpočet.

Měření	Jednotka
Propustnost	zprávy/s
Využití CPU	%
Využití RAM	GiB
Velikost disku	GiB
Disk IOPS	IOPS
Rychlost čtení disku	MiB/s
Rychlost zápisu disku	MiB/s
Vytížení sítě - přijaté	Mibit/s
Vytížení sítě - odeslané	Mibit/s

Obrázek 2.3: Potřebná data s jednotkami pro nástroj Crystal.

2.3.3 Interpretace výsledků

Výsledky získané z jednotlivých testovacích scénářů, které budou popsány dále, budou převedeny do takové formy, aby bylo možné je prezentovat. Výsledky budou tedy převedeny do XML(*Extensible Markup Language*) a JSON (*JavaScript Object Notation*) formátů a tyto soubory budou poté sloužit jako vstupní data pro produkt Crystal (XML soubory) který je také vyvíjen společností Xura, Inc a pro systém na výkonnostní a zátěžové testování (JSON soubory). Převod a automatické nahrávání XML souborů nebude součástí systému na výkonnostní a zátěžové testování, jelikož jejich periodické nahrávání na server není žádoucí. JSON soubory budou generovány právě systémem, který bude implementován.

Crystal slouží ke změření, zaznamenání a následné interpretaci výsledků výkonnostních a zátěžových testů. Kromě aplikačního serveru Xtend, který bude používat Crystal k interpretaci výsledků, jej také používají další produkty firmy Xura, Inc. Produkt Crystal požaduje vstup právě ve formátu XML a je nutností mu na vstup dát i typ scénáře a použité hardwarové vybavení, pokud se jedná o testování na samostatném stroji. Výstupy nástroje Crystal jsou tedy použity pro specifikaci serverů u zákazníků a pro určení škálovatelnosti interních produktů.

Kapitola 3

Produkt Xtend

Tato diplomová práce se zabývá výkonnostním a zátěžovým testováním produktu Xtend od firmy Xura, Inc. V této kapitole bude popsána činnost firmy Xura, její jednotlivé produkty důležité pro tuto práci a bude představen produkt Xtend, na kterém budou prováděny výkonnostní a zátěžové testy. V rámci představení daného produktu budou uvedeny i důležité a jednotlivé služby tohoto produktu. Detailněji budou popsány služby, které budou podrobeny daným testům.

3.1 Xura

Společnost Xura, Inc. vznikla spojením firem Comverse a Acision v roce 2015. Acision byla původně firma zabývající se převážně telekomunikací a poskytováním různých služeb spojených se světem SMS (*Short Message Service*) a messagingu. SMS jsou krátké zprávy, jejichž délka je omezena na 160 znaků a jejichž prostřednictvím je možná komunikace mezi dvěmi zařízeními. Firma spolupracuje s velkými telefoními operátory v zahraničí, více informací o této firmě je možné nalézt na jejich webových stránkách [36].

3.2 Xtend

Jak již bylo zmíněno na začátku této kapitoly, tato diplomová práce se zabývá výkonnostním a zátěžovým testováním produktu Xtend a jeho určitých částí. Co se týče samotného produktu, tak je vyvýjen relativně malým týmem na Brněnské pobočce firmy. Xtend slouží primárně pro manipulaci s SMS zprávami. Tento produkt se nasazuje u telefonních operátorů, kteří chtějí poskytovat nadstandardní SMS služby svým zákazníkům. Xtend poskytuje různé služby, které ke svojí činnosti vyžadují i jiné produkty. Některé z těchto produktů jsou také vytvořeny firmou Xura, Inc. Mezi příklady používaných produktů v rámci služeb, na kterých se podílí Xtend, můžeme řadit MCO (*Message Controller*), AAGP (*Acision Applictaion Gateway Provisioning*), DS(*Directory Service*) používající protokol LDAP (*Lightweight Directory Access Protocol*) a další. Jelikož je produkt Xtend vlastnictvím firmy Xura, Inc., tak zde budou popsány pouze stručně jednotlivé služby, bez detailního popisu, jak fungují a jaké všechny operace během nich probíhají. Typicky může být Xtend nakonfigurován, aby podporoval více než jednu sms službu, podle potřeb zákazníka.

3.2.1 Collect SMS

Ve světovém měřítku přes 75% lidí, kteří používají mobilní telefon a mají předplacený kredit, vyčerpá svůj kredit před datem jejich obvyklého dobíjení. Ve všech sítích, kde se vyskytují uživatelé s kreditem, je jejich počet signifikantní. Počet SMS zpráv, které nemohou být doručeny a zúčtovány je z tohoto důvodu velký. Pro tyto případy byla vymyšlena služba COllect SMS.

Collect SMS umožňuje operátorům generovat zisky právě z SMS zpráv, které by normálně nemusely být doručeny. Uživatelé, kteří nedisponují kreditem mohou požádat příjemce zprávy, aby zaplatil za SMS, která je mu určena místo jejího skutečného odesílatele. Collect SMS je tu tedy pro ty, kteří chtějí poslat SMS zprávu, ale nemají na ni dostatek financí. Typické příklady služby mohou být následující:

- Lidé bez kreditu, kteří se snaží ozvat přátelům
- Posílání zpráv týkající se práce z osobních zařízení
- Děti, které se snaží kontaktovat rodiče

V následujících odstavcích bude odesílatel někdy označován jako *A-party* a příjemce jako *B-party*. Služba Collect SMS je buďto spuštěna přidáním speciálního prefixu před číslo příjemce tzv. *Sender initiated*, nebo může být spuštěna druhým způsobem, který je nazván *Network initiated*. Oba tyto způsoby iniciování služby Collect SMS budou popsány dále.

Collect SMS služba poté zašle textovou zprávu příjemci, kde je dotázán, zda-li si přeje přijmout zprávu od odesílatele. Tato zpráva je poslána ze speciálního prefixu v rámci sítě a tudíž se nejeví jako číslo reprezentující A-party. V případě kladné odpovědi od B-party, je mu stržena částka, kterou by zaplatil odesílatel a je mu doručena zpráva, která mu byla původně zaslána od A-party. Tato služba je dostupná v rámci sítí GSM(*Groupe Spécial Mobile*) a CDMA (*Code Division Multiple Access*).

Sender initiated

Služba je iniciována odesílatelem, který před zvolené číslo příjemce umístí speciální prefix, například 5550612345678 (kde *555* je prefix). Prefix slouží k vyhnutí se zúčtování SMS zprávy na SCP/SMSC (*Service Control Point/Short Message Service Center*) a správnému nasměrování dané zprávy na aplikační server Xtend, který zajišťuje tuto službu. Xtend podle tohoto prefixu rozpozná o jakou službu se jedná a pošle zprávu týkající se možného přijetí zprávy a zaplacení místo odesílatele. Podle nastevení operátora poskytujícího telekomunikační služby může být poté poslána původnímu odesílateli ještě doručenka o zaslání zprávy.

Network initiated

Pokud si odesílatel není vědom nedostatku kreditu a pošle zprávu bez výše zmíněného prefixu, tak je služba Collect SMS automaticky spuštěna pomocí tzv. *in-network-trigger*, který se nachází na Xura SMSC. Odesílateli je poté nabídnuta možnost zaslání zprávy pomocí služby Collect SMS. Pokud bude s touto možností souhlasit, tak ji stačí potvrdit bezplatnou zprávou SMS. V tomto případě dojde ke stejnému zacházení se zprávou jako u Sender initiated. V případě negativní odpovědi není zpráva samozřejmě doručena.

Služba Collect SMS podporuje další nastavení uživatelských údajů a preferencí, ale na základní pochopení této služby nejsou nezbytně nutné.

3.2.2 Group Messaging

Další službou, která bude otestována v rámci produktu Xtend, je Group Messaging. Jak již samotný název napovídá, jedná se o skupinovou komunikaci prostřednictvím SMS. Použitím se to neliší od jiných skupinových chatů na různých sociálních sítích (např. Facebook). I bez skupinových zpráv je možné odeslat jednu SMS tak, aby ji obdrželo více příjemců. V tomto případě se ale odesílá více zpráv z dané mobilní stanice až k SMSC, takže se vlastně jedná o jednu zprávu, ale odeslanou několikrát z mobilní stanice. V případě Group Messagingu se poté jedná o jednu zprávu, která se odešle na SMSC. Na SMSC se již rozdistribují dané zprávy a odešlou se cílovým adresátům. Podstatnou změnou také je, že daná zpráva není adresována přímo jednotlivým účastníkům, ale je adresovaná skupině, která má své specifické PSN (*Personal Short Number*). Narozdíl od Collect SMS, kde se používá speciální prefix.

V Group Messagingu se vytvářejí skupiny do kterých se poté pozývají lidé. Skupiny se mohou tématicky pojmenovat, tedy můžeme je pojmenovat jako "Rodina", "Kamarádi"a podobně. V daných skupinách je více rolí, kdy uživatel, který skupinu vytvořil je jejím administrátorem a má právo nastavovat různé parametry skupiny jako je : pozývání uživatelů, vyhazování ze skupiny apod. Podle uživatelských práv poté následují přímo jednotliví uživatelé, kteří mají práva nastaveny podle nastavení skupiny. Všechny tyto parametry skupiny se nastavují prostřednictvím SMS zpráv na dané PSN s přesně danou syntaxí. Další možností nastavení těchto parametrů je využití webového portálu, kde jsou tyto konfigurační změny také dostupné.

Stejně jako v případě Collest SMS, i zde bude následovat posloupnost jednotlivých kroků u Group Messagingu:

- Číslo A chce vytvořit skupinu s jménem Family, tudíž pošle SMS zprávu na specifické PSN (Personal Short Number).
- Číslo A obdrží potvrzovací SMS ohledně vytvoření skupiny a daná zpráva ještě obsahuje specifické PSN, které referencuje danou skupinu pro uživatele A.
- Číslo A poté pozve pomocí dalšího SMS příkazu jiné uživatele do nově vytvořené skupiny pomocí PSN.
- Číslo A obdrží potvrzovací zprávu ohledně pozývání jednotlivých uživatelů do skupiny.
- Číslo B obdrží zprávu ohledně pozvání do skupiny. Může buďto přijmout nebo odmítnout pozvání odpovědí na PSN z kterého SMS přišla.
- Pokud číslo B přijme danou pozvánku, obdrží PSN a potvrzovací SMS ohledně přijetí do skupiny.
- Jakmile číslo B přijme pozvánku, může začít komunikovat pomocí Group Messagingu s ostatními členy skupiny.

Group messaging využívá služeb tzv. Group Messaging servru, který obsahuje databázi a samotnou logiku dané služby. Tento server funguje na bázi *dotaz-odpověď* a odpovídá na předem definované dotazy od Xtendu.

3.2.3 Personalized Messaging

Poslední testovanou službou bude Personalized messaging. Tato služba zpříjemňuje uživateli používání SMS, protože stejně jako u emailů, tak i s podporou této služby má možnost automaticky modifikovat standardně posílané zprávy na různé čísla. Personalized Messaging umožňuje nastavit následující SMS modifikace:

- Black list na straně odesílatele / příjemce - na tyto čísla bude posílání zpráv zablokované
- Whitelist na straně odesílatele / příjemce - na tyto čísla bude posílání zpráv povoleno
- Kopie zprávy na straně odesílatele / příjemce - na čísla specifikované v odpovídající položce v LDAP databázi budou přeposílány jednotlivé zprávy.
- Multi-SIM na straně odesílatele - Odesílatel se může rozhodnout ke změně čísla, ze kterého byla daná zpráva odeslána. V tomto případě tedy dochází ke změně odesílatelova čísla.
- Podpis na straně odesílatele - Ke zprávě je automaticky přídán určitý podpis, stejně jako tomu může být u emailů. Pokud podpis překročí povolenou délku zprávy, tak se zpráva rozdělí na více segmentů.
- Automatická odpověď na straně příjemce - Automatická odpověď slouží v době nepřítomnosti k uvědomnění odesílatele SMS zprávy o případné nedostupnosti či aktuálním stavu příjemce. Může být nastaveno více potenciálních odpovědí.
- Divert na straně příjemce - dovoluje ještě před přijetím zprávy na daném čísle přeposlat zprávu na jedno či více jiných čísel, tedy tak, že původní příjemce danou zprávu na svém čísle neobdrží.

Oproti Group Messagingu a Collect SMS se zde nepoužívá žádný prefix nebo PSN a jednotlivé služby se mohou zapínat pomocí SMS příkazů, které jsou odeslány na speciální číslo. Tyto příkazy poté nastavují potřebné služby na straně zákazníka. V rámci Xtendu je zapnutí či vypnutí dané služby pro zákazníka popsáno specifickým příznakem u jednotlivých položek v LDAP databázi. Personalized Messaging nepotřebuje ani oddělený server, jelikož jeho funkcionalita je zanesena v Xtendu.

Kapitola 4

Dostupné nástroje na měření výkonu

V této části budou uvedeny některé existující nástroje na měření výkonnosti, které jsou dostupné na trhu. Jednotlivé nástroje budou probrány a zhodnoceny. Některé z těchto nástrojů jsou placené, některé jsou volně šiřitelné.

4.1 Open source nástroje

Jak již název této sekce napovídá, tak se v této části zaobírám využitím volně šiřitelných nástrojů pro výkonnostní a zátěžové testování. Na trhu existuje velká řada těchto nástrojů a proto je důležité vybrat takový nástroj, který by splňoval všechny požadavky, které na něj mohou být kladeny v rámci výkonnostního a zátěžového testovaní produktu Xtend.

4.1.1 Apache JMeter

Apache JMeter je napsán v programovacím jazyce Java. Kromě využití k provedení výkonnostních a zátěžových testů, může být tento nástroj použit také k vytvoření testovacího plánu. Podporuje take vytváření funkcionálních testů a testovacích plánů. Apache JMeter může být nahrán na server a poté může být pomocí tohoto nástroje prováděna analýza chování systému [6].

Bohužel tento nástroj podporuje primárně testování Java aplikací a z tohoto důvodu se proto nehodí k testování aplikačního server Xtend.

4.1.2 The Grinder

The Grinder je další z řady volně dostupných nástrojů na testování. Tento nástroj je stejně jako Apache JMeter založený na použití Javy. Umožňuje flexibilní parametrizaci testovacích scénářů, což znamená, že I během testu je možné měnit testovací data. Podporuje také použití externích databází a souborů jako zdrojů dat. Distribuované testování je samozřejmosti pro tento nástroj společně s přístupem k jednotlivým výsledkům testů. The Grinder podporuje take škálování aplikací [7].

Tento nástroj se ale take nehodí k výkonnostnímu a zátěžovému testování aplikačního server Xtend kvůli jeho nemožnosti použití protokolu SMPP.

4.1.3 Tsung

Poslední ze zkoumaných nástrojů je Tsung. Tento nástroj, jako jediný ze zmíněných v této sekci, nepoužívá Javu, ale jeho základy nalezneme v programovacím jazyce Erlang. Pro využití tohoto nástroje je tedy nezbytné nainstalovat Erlang. Tento nástroj nepodporuje zadávání testovacích scénářů pomocí grafického rozhraní, ale jednotlivé specifikace testovacích scénářů jsou vytvářeny v terminálu. Nicméně pro zobrazení výsledků a grafů nalezneme grafickou podporu. Tsung podporuje také vice protokolů, podobně jako The Grinder. Nicméně SMPP protocol, který je nezbytností k výkonnostnímu a zátěžovému testování zde nenalezneme [7].

4.1.4 SMPPSim

SMPPSim je nástroj na testování SMS aplikací. Z výše uvedených jako jediný podporuje použití SMPP protokolu. Nicméně k tomuto nástroji jsem nenalezl žádné funkční GUI a také tento nástroj nepodporuje nastavení specifických scénářů, které by odpovídaly službám Collect SMS a Group Messaging [23].

4.2 Placené nástroje

V této části se budu zaobírat vybranými placenými webovými nástroji, které podporují výkonnostní a zátěžové testování. Oproti open source verzím, které byly popsány dříve, obsahují standardně online podporu a v některých případech zahrnují take dodatečnou funkcionalitu.

4.2.1 LoadRunner

Tento nástroj je prodávan společností HP. O tomto nástroji se traduje, že dokáže otestovat skoro jakýkoliv software nebo databázi. Nicméně tento nástroj je velice robustní. Tento nástroj je velmi často využíván velkými korporáty na otestování jejich aplikací a výrobků. Bohužel při zkoumání tohoto nástroje jsem narazil na nemožnost otestování aplikací využívající SMPP a primárně byla zmíněná velmi dlouhá učící se křivka s tímto placeným softwarem [17].

4.2.2 Silk Performer

Tento nástroj je také určen k využití ve velkých korporátních společnostech. Stejně jako LoadRunner i Silk Performer poskytuje mnoho funkcí a možností specifikace testovacích scénářů. Tento nástroj poskytuje 45 denní období, kdy je zdarma bez jakýchkoliv poplatků [17].

4.3 Zhodnocení dostupných nástrojů

V této kapitole byly uvedeny nástroje, které jsou dostupné a využívané pro výkonnostní a zátěžové testování různých aplikací. Byly probrány jak open source nástroje, tak i placené nástroje. Jelikož je tato diplomová práce psaná pro firmu, tak byly speciální požadavky kladeny na tento nástroj a většina z výše uvedených nástrojů má buďto dlouhou učící se křívku nebo je v nich velice těžké, či nemožné nakonfigurovat a vytvořit prvky dle vlastních

potřeb a specifikací. Z těchto důvodů vznikne tedy system pro výkonnostní a zátěžové testování, který bude napsán kompletně od základů.

Kapitola 5

Návrh Aplikace

V této kapitole bude popsána aplikace, za jejíž pomocí budou zhotoveny a vykonány jednotlivé testovací scénáře. Tato aplikace bude sloužit jako systém pro správu a zadávání krátkodobých i dlouhodobých výkonnostních a zátěžových testů aplikačního serveru Xtend. V jednotlivých sekcích této kapitoly budou probrány uživatelské i serverové vlastnosti této aplikace, která bude obsluhována z webové prezentace. Všechny texty ve webové prezentaci jsou napsány anglickým jazykem, pokud se nejedná o hlášky prohlížeče, v tom případě jsou popisky v nativním jazyce prohlížeče.

5.1 Webová prezentace

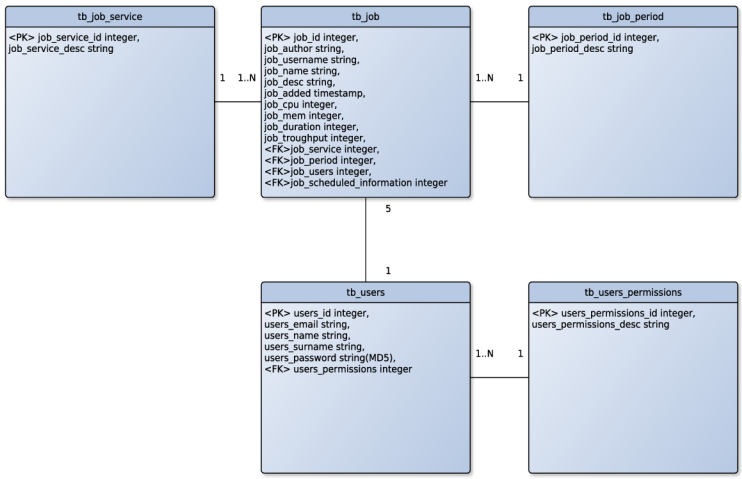
V této části bude popsána jednoduchá webová prezentace, která slouží k vizualizaci výsledků výkonnostního a zátěžového testování aplikačního serveru Xtend. Grafy, které tato webová prezentace bude poskytovat slouží k zpřehlednění výsledků a informovanosti uživatele o celkovém využití zdrojů a propustnosti systému, tzn. počtu zpráv za sekundu. Cílem této aplikace bude zjednodušenní náhledu na výsledky testů. Jako vstupní soubor bude sloužit soubor formátu JSON. Tento soubor bude obsahovat podobné data, jako soubory, které slouží pro škálování produktů pomocí aplikace Crystal. Struktura toho JSON souboru bude popsána dále.

Instalace jednotlivých klíčových balíčků při vývoji aplikace bude zajištěna pomocí nástroje *npm*, který je v dnešní době nejpoužívanějším prostředníkem pro instalaci javascriptových knihoven a balíčků. Pomocí tohoto nástroje budou tedy stahovány závislosti a balíčky ve výsledném systému.

Samotná webová prezentace by měla být co nejjednodušší na ovládání a dobře rozšiřitelná pro případné přidání další funkiconality, či dalším projektů v rámci firmy. Systém pro výkonnostní a zátěžové testování z hlediska serverové části, ale i uživatelské bude do detailu popsán v kapitole zabývající se implementací systému.

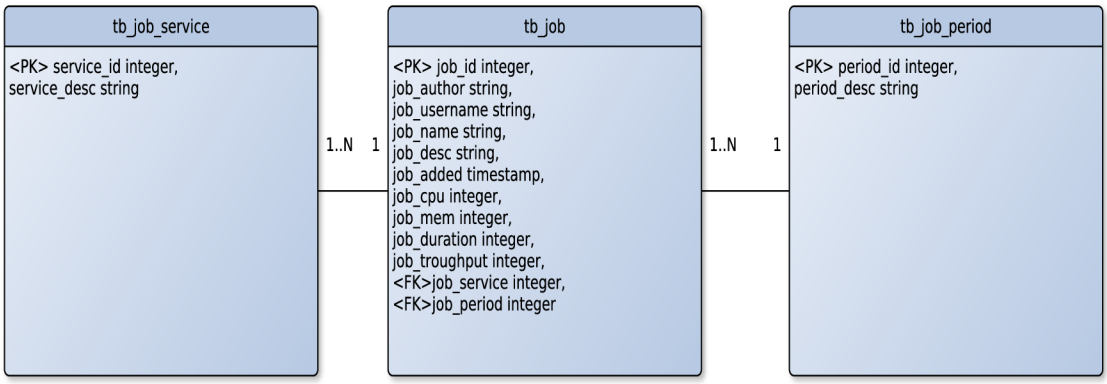
5.2 Databázové schéma

Pro začátek vývoje aplikace je třeba navrhnout databázové schéma, které bude následně v aplikaci použito. Význam jednotlivých tabulek a funkčnost systému bude popsána v následujících kapitolách. Jak již tomu u většiny aplikací je, tak i toto databázové schéma prošlo vývojem. Na počátku stálo databázové schéma, které je zobrazeno na obrázku 5.1.



Obrázek 5.1: Původní návrh databázového schéma.

Systém pro výkonnostní a zátěžové testování bezesporu musí obsahovat možnost vytvářet, modifikovat a mazat uživatele. Nicméně tato funkčnost může být zajištěna i jinými nástroji. Při hledání řešení, které by splňovalo požadavky na bezpečnost, ale i na uživatelskou použitelnost, jsem narazil na projekt Stormpath [30]. Tento projekt je volně k použití, při určitém provozu, který systém na výkonnostní a zátěžové testování dozajista splní. Při větším provozu by tento nástoj byl zpoplatněn a tudíž by již nepřipadalo v úvahu. Odklonění od původního návrhu databáze, kdy by uživatelská registrace a s tím spojené věci byly řešeny samotným systémem, bylo také způsobeno snahou o prozkoumání nástrojů na trhu a řešení Stormpath se jeví jako velmi kvalitní a spolehlivé. Schéma, které bylo zobrazeno na obrázku 5.1 se tedy transformovalo na schéma zachycené na obrázku 5.2.



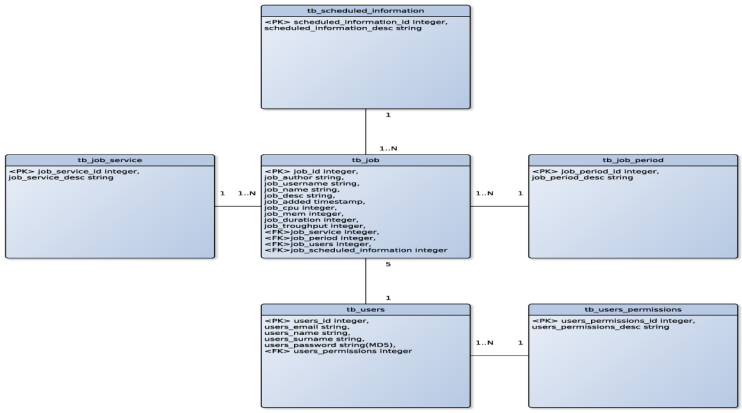
Obrázek 5.2: Návrh s použitím Stormpath.

Kromě utilit, které budou použity v aplikaci umožňuje Stormpath i přihlašování pomocí Facebooku a jiných sociálních sítí, nicméně tato funkce je v rámci vyvíjeného systému zbytečná. Pro unikátní identifikaci uživatele se používá registrační email. Mazání, nastavování uživatelských skupin a podobné záležitosti se řeší přes administrátorské rozhraní na portálu [30]. K účtu, který přísluší aplikaci bude mít přístup manažer produktu Xtend a hlavní

tester, který bude se systémem na výkonnostní a zátěžové testování každodenně pracovat. Ostatní členové týmu nebudou mít přístup k administrátorské sekci na zmíněném portále, nicméně si budou moct založit účet ve vyvíjeném systému.

Nicméně toto schéma nakonec nebylo v produkčním prostředí použito, kvůli interním záležitostem firmy, takže i přes snahu zprovoznit autentizaci uživatelů pomocí výše zmíněného nástroje, je nakonec využitá uživatelská administrace, kdy jednotlivé údaje o uživatelích jsou uloženy v lokální databázi. V databázi je uložena zašifrovaná podoba hesla a vůči ní je poté i testováno připojení. Databázové schéma tedy vypadalo jako na obrázku 5.1.

V průběhu vývoje se ještě jednou změnilo databázové schéma kvůli přidání funkcionality. Při prezentaci návrhu všech možných utilit, který bude daný systém zajišťovat, vznikly nové požadavky a bylo tedy třeba dané schéma poupravit. Finální podoba je vidět na obrázku 5.3.



Obrázek 5.3: Finální podoba databázového schéma.

5.3 Overlord

Nástroj, který bude generovat provoz na jednotlivých virtuálních strojích se nazývá *Overlord*. Základ tohoto nástroje byl vyvinut v minulých letech při posledním výkonnostním a zátěžovém testování aplikačního serveru Xtend. Nástroj byl napsán v jazyce C. Jako vstupní parametr vyžaduje konfigurační soubor. Tento konfigurační soubor poté určuje posloupnost akcí, které se vykonají a posloupnost přechodů. V konfiguračním souboru lze také nalézt i následující důležité parametry:

- IP adresu serveru
- port potřebný ke spojení
- počet zpráv za sekundu
- velikost front pro odesílání, přijímání a buffrování zpráv, které čekají opakované odeslání
- čas odeslání zprávy
- číslo od kterého budou generovány zprávy (čísla se rotují)

Nástroj posílá zprávy a komunikuje pomocí protokolu SMPP (*Short Message Peer-to-Peer Protocol*), který bude popsán dále. Samotné zdrojové soubory aplikačního serveru Xtend nejsou součástí této diplomové práce a proto nebude popsáno detailně spojení mezi nástrojem Overlord a samotným aplikačním serverem. Stejně tak nebude v této diplomové práci popsán chod a interní komunikaci při zpracovávání zpráv aplikačního serveru Xtend. Část, která bude modifikována při této diplomové práci v rámci nástroje Overlord, se bude převážně týkat času odeslání jednotlivých zpráv. V současné verzi se čas odeslání počítá staticky. Mnou předělaná verze bude tedy více odpovídát reálné situaci podle statistických dat, kdy jednotliví uživatelé neodpovídají v předem daný čas. Ale odpověď na Collect SMS zprávy, stejně tak posílání zpráv do skupin, je věc proměnlivá a tudíž čas odpovědi od obdržení zprávy není nikdy stejný. Plánuji tedy změnit chod programu Overlord tak, aby čas odpovědí byl definován Gaussovým rozložením, což více odpovídá realitě. Uživatel bude mít možnost při testování si nakonfigurovat průměrný čas odpovědí a rozptyl, který bude brán v zřetel. Tato konfigurace bude probíhát v konfiguračním souboru, který je předáván programu Overlord.

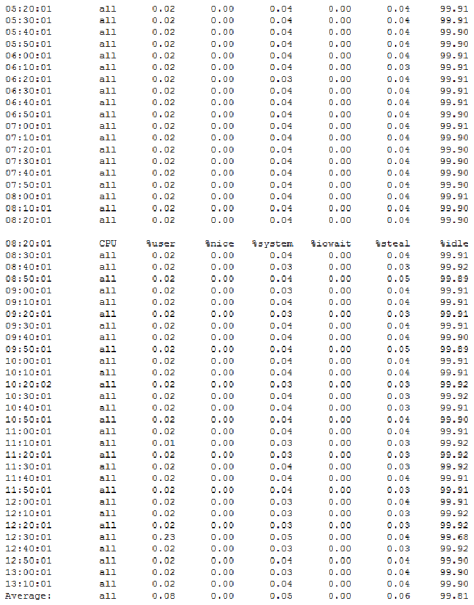
5.4 Utility pro sběr dat

V této části kapitoly budou popsány jednotlivé utility, které budou použity pro sběr a vyhodnocování jednotlivých výkonnostních a zátěžových testů.

SAR

Pro sběr užitečných dat, které se poté budou vyhodnocovat, bude využit nástroj *sar*. Jelikož server Xtend v laboratorních podmínkách poběží na systému od společnosti Red Hat, tak bylo možno využít tento nástroj. Podle [22] sar vypíše na standardní výstup obsahy vybraných kumulativních čítačů v operačním systému. Účtovací systém, který je založen na hodnotách v *count* a *interval* parameters, vypisuje jednotlivé hodnoty operačního systému po intervalech udaných v parametrech v sekundách. Pokud je *interval* roven 0, pak *sar* vypisuje průměrné hodnoty od spuštění systému. Pokud dojde k restartování daného systému, tak je tento restart také zaznamenán. V případě, že *count* je roven nule, tak jsou jednotlivé hodnoty vypysiovány průběžně. Pro uchovávání výsledků se používá speciálního parametru, který ukládá výstup z této utility do souboru, takovým parametrem je *o*, který vyžaduje další povinný arugment, kterým je jméno výstupního souboru. Pokud je jméno výstupního souboru vynecháno, tak se data ukládají do speciálního souboru */var/log/sa/sadd* , kde *dd* je aktuální den. Jeden z možných výstupů této utility je zobrazen na 5.4.

Všechny možnosti nastavení a použití utility *sar* jsou zveřejněny v manuálových stránkách, které jsou také dostupné pod [22].



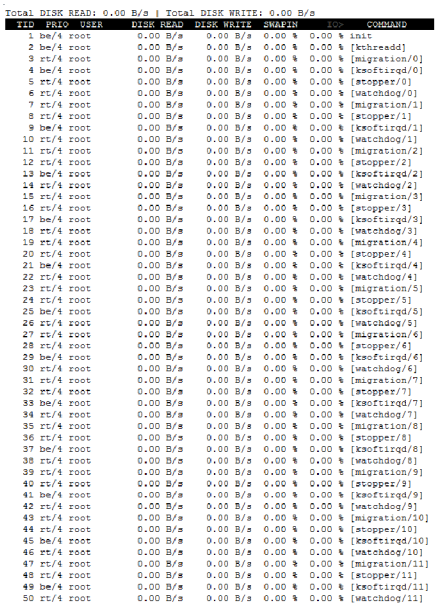
```
05:20:01        all      0.02      0.00      0.04      0.00      0.04     99.91
05:30:01        all      0.02      0.00      0.04      0.00      0.04     99.91
05:40:01        all      0.02      0.00      0.04      0.00      0.04     99.90
05:50:01        all      0.02      0.00      0.04      0.00      0.04     99.90
06:00:01        all      0.02      0.00      0.04      0.00      0.04     99.91
06:10:01        all      0.02      0.00      0.04      0.00      0.03     99.91
06:20:01        all      0.02      0.00      0.03      0.00      0.04     99.91
06:30:01        all      0.02      0.00      0.04      0.00      0.04     99.91
06:40:01        all      0.02      0.00      0.04      0.00      0.04     99.91
06:50:01        all      0.02      0.00      0.04      0.00      0.04     99.90
07:00:01        all      0.02      0.00      0.04      0.00      0.04     99.91
07:10:01        all      0.02      0.00      0.04      0.00      0.04     99.90
07:20:01        all      0.02      0.00      0.04      0.00      0.04     99.90
07:30:01        all      0.02      0.00      0.04      0.00      0.04     99.90
07:40:01        all      0.02      0.00      0.04      0.00      0.04     99.90
07:50:01        all      0.02      0.00      0.04      0.00      0.04     99.90
08:00:01        all      0.02      0.00      0.04      0.00      0.04     99.91
08:10:01        all      0.02      0.00      0.04      0.00      0.04     99.90
08:20:01        all      0.02      0.00      0.04      0.00      0.04     99.90

08:20:01        CPU     %user     %nice   %system   %iowait    %steal     %idle
08:30:01        all      0.02      0.00      0.04      0.00      0.04     99.91
08:40:01        all      0.02      0.00      0.03      0.00      0.03     99.92
08:50:01        all      0.02      0.00      0.04      0.00      0.05     99.89
09:00:01        all      0.02      0.00      0.03      0.00      0.04     99.91
09:10:01        all      0.02      0.00      0.04      0.00      0.04     99.91
09:20:01        all      0.02      0.00      0.03      0.00      0.03     99.91
09:30:01        all      0.02      0.00      0.04      0.00      0.04     99.91
09:40:01        all      0.02      0.00      0.04      0.00      0.04     99.90
09:50:01        all      0.02      0.00      0.04      0.00      0.05     99.89
10:00:01        all      0.02      0.00      0.04      0.00      0.04     99.91
10:10:01        all      0.02      0.00      0.04      0.00      0.04     99.91
10:20:02        all      0.02      0.00      0.03      0.00      0.03     99.92
10:30:01        all      0.02      0.00      0.04      0.00      0.03     99.92
10:40:01        all      0.02      0.00      0.04      0.00      0.03     99.91
10:50:01        all      0.02      0.00      0.04      0.00      0.04     99.90
11:00:01        all      0.02      0.00      0.04      0.00      0.04     99.91
11:10:01        all      0.01      0.00      0.03      0.00      0.03     99.92
11:20:01        all      0.02      0.00      0.03      0.00      0.03     99.92
11:30:01        all      0.02      0.00      0.04      0.00      0.03     99.92
11:40:01        all      0.02      0.00      0.04      0.00      0.04     99.91
11:50:01        all      0.02      0.00      0.04      0.00      0.03     99.91
12:00:01        all      0.02      0.00      0.03      0.00      0.04     99.91
12:10:01        all      0.02      0.00      0.03      0.00      0.03     99.92
12:20:01        all      0.02      0.00      0.03      0.00      0.03     99.92
12:30:01        all      0.23      0.00      0.05      0.00      0.04     99.68
12:40:01        all      0.02      0.00      0.03      0.00      0.03     99.92
12:50:01        all      0.02      0.00      0.04      0.00      0.04     99.90
13:00:01        all      0.02      0.00      0.03      0.00      0.04     99.90
13:10:01        all      0.02      0.00      0.04      0.00      0.04     99.90
Average:        all      0.08      0.00      0.05      0.00      0.06     99.81
```

Obrázek 5.4: Ukázka výpisu utility *sar*.

Iotop

Tato systémová utilita bude využita kvůli náročnosti výkonnostních a zátěžových testů na vstupní/výstupní rozhraní. Při různých nastaveních produktu Xtend může docházet k vyčerpání všech alokovatelných zdrojů. Pokud k tomuto bude docházet, tak samotné měření bude velice nepřesné a tento nástroj bude pomáhat odhalovat chyby na I/O. K problematice *iotop* je věnována následující manuálová stránka [13]. Podle tohoto zdroje *iotop* zobrazuje tabulku, která poukazuje na aktuální využití vstupně/výstupních operací jednotlivými procesy nebo vlákny v systému. Aby tato utilita fungovala, je zapotřebí mít povolenou následující možnost při sestavování jádra: *CONFIG_TASK_DELAY_ACCT* a *CONFIG_TASK_IO_ACCOUNTING*. Všechny možnosti nastavení *iotop* je možné najít na [13].

Samotný výpis je znovu, obdobně jako u utility *sar*, vypočítáván a zobrazován na základě určité definované periody. Obrázek 5.5 zobrazuje možný výpis.

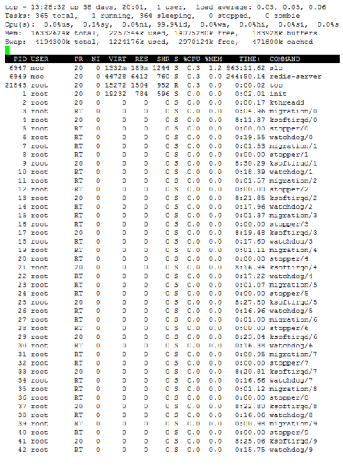


Obrázek 5.5: Ukázka výpisu utility *iotop*.

Top

Podobně jako *iotop*, tak i *top* ukazuje na systémové hodnoty. Tato utilita bude zvolena z důvodů monitorování využití CPU. Zobrazuje dynamicky měnící se hodnoty v reálném čase. Zobrazuje také jednotlivé procesy a využití systémových zdrojů těmito procesy. Kromě tohoto *top* poskytuje také limitovanou možnost interaktivity s jednotlivými procesy a konfiguraci sama sebe, kdy se může modifikovat i chování samotné utility [31].

Výpis takovéhoto nástroje může vypadat obdobně, jako je na obrázku 5.6.



Obrázek 5.6: Ukázka výpisu utility *top*.

Kapitola 6

Implementace aplikace

V této části bude popsán návrh aplikace, od protokolu, pomocí kterého se bude zajistovat posílání SMS zpráv po samotnou aplikaci.

6.1 SMPP

Následující popis protokolu SMPP, který slouží k transportu SMS zpráv mezi jednotlivými uzly sítě. SMPP je použit ve výsledných zdrojových souborech a proto je zde popsán. Tento protokol má více specifikací, v následujícím textu bude uvažována poslední dokumentovaná a oficiálně vydaná specifikace tohoto protokolu a to verze 5.0. Popis specifikace tohoto protokolu v této diplomové práci je převzat z dokumentu [1].

SMPP je otevřený standard a jak již bylo zmíněno, tak tento protokol slouží k transportu dat SMS zprávy mezi jednotlivými uzly telekomunikační sítě a to hlavně mezi ESME (*External Short Messaging Entities*) , RE (*Routing Entities*) a MC (*Message Centre*). SMPP umožňuje posílání také EMS (zahrnuje prvky SMS i MMS), notifikací o hlasových schránkách, MMS notifikací a dalších. Jelikož je SMPP všestrané a podporuje i protokoly, které nejsou založené na GSM, tak je to nejpoužívanější protokol na výměnu SMS zpráv mimo sítě SS7 (*Signalling System No.7*).

Tento protokol byl původně navržen k testování SMSC bez použití přístrojů a vybavení podporující SS7 pro posílání zpráv. Na tomto protokolu pracovalo mnoho firem a uskupení z nichž některé již neexistují a nyní je tento protokol vlastnictvím firmy Xura, původně Acision [24].

SMPP je založeno na modelu klient-server. Ve většině případů se právě MC v tomto modelu chová jako server a očekává připojení od ESME. Pokud je spojení mezi jednotlivými operátory na přímo a nejsou využívaná často velmi drahá zařízení třetích stran, tak se MC chová jako klient. Tento protokol je založen na výměně PDU (*protocol data unit* neboli packet) rámců, které se posílají na bázi dotaz-odpověď. Tyto PDU rámce se posílají v síti po čtvrté vrstvě ISO/OSI modelu přes TCP spojení. Pro tuto komunikaci byl vyhrazen i speciální port s číslem 2775. Nicméně typicky se mohou používat i jiné porty. Jako u většiny spojení v síti přes TCP, i zde se musí spojení mezi dvěma stranami nejprve navázat a to jak na transportní vrstvě (TCP), tak i na relační vrstvě (SMPP) tzv. navazovacím příkazem (*bind command*), který obsahuje krom dále popsaných parametrů také verzi SMPP protokolu. Navázání takovéhoto spojení určí která strana bude posílat užitečná data a která je bude přijímat. Typicky se rozlišují tři navazovací příkazy:

- *bind_transmitter* - pouze klienti budou moci odesílat zprávy na server.

- *bind_receiver* - klienti budou pouze přijímat zprávy.
- *bind_transceiver* - klienti mohou odesílat i přijímat zprávy.

Dalšími parametry, které se posílají v těchto navazovacích příkazech jsou: *system_id*, *system_type* a *password*. Tyto parametry jsou důležité zhlediska navázání spojení.

PDU rámce, které se posílají mají strukturu, která je zobrazena v tabulce 6.1. Tyto rámce mají povinnou hlavičku, ale jejich obsah není již povinný. Pro efektivní přenos informací jsou samozřejmě tyto rámce zakodovany binárně.

SMPP PDU				
PDU Header (mandatory)				PDU Body (Optional)
Command length	Command Id	Command Status	Sequence Id	PDU Body
4 octets	Length = (Command Length value - 4) octets			

Tabulka 6.1: Struktura PDU rámce [24].

Hlavička PDU rámce se skládá ze čtyř políček, kdy každé je dlouhé právě čtyři oktety. A jelikož PDU používá Big Endian, tak první oktet je nejvýznamnější, tzv. MSB (*Most Significant Byte*). Jednotlivé oktety mají následující významy:

- command_length - celková délka paketu v oktetech. Nejmenší možná délka, která je zaznamenaná v tomto políčku je šestnáct oktetů, jelikož samotná hlavička obsahuje minimálně šestnáct oktetů.
- command_id - identifikace samotné operace nebo smpp příkazu.
- command_status - pokud se jedná o dotaz, tak je vždy hodnota tohoto políčka rovna 0, pokud se jedná o odpověď na určitý dotaz, tak pak toto pole nese návratovou hodnotu.
- sequence_number - pomocí tohoto pole se párují jednotlivé dialogy.

Co se týče samotného těla zprávy, tak to může obsahovat kromě samotného textu zprávy také zdrojovou a cílovou adresu, typ služby, která je zrovna používaná, kodování dat, délku zprávy a další. Pokud se tedy posílají data skrze sít, tak jedno tělo PDU rámce může obsahovat následující parametry: *service_type*, *source_addr_ton*, *source_addr_npi*, *source_addr*, *dest_addr_ton*, *dest_addr_npi*, *dest_addr*, *esm_class*, *protocol_id*, *priority_flag*, *schedule_delivery_time*, *validity_period*, *registered_delivery*, *replace_if_present_flag*, *data_coding*, *sm_default_msg_id*, *sm_length*, *short_message*.

Pro samotné tělo zprávy, které je volitelné, se používá forma TLV (*Type-Length-Value*) a nachází se nakonci PDU.

6.2 Výběr prostředí

Po představení si protokolu, který bude zajišťovat komunikaci mezi testovaným softwarem a nástrojem určeným k testování, bylo třeba zvolit platformu, na které budou testované virtualní stroje spouštěny a provozovány. Jelikož se ve firmě, pro kterou je tato diplomová práce, používají tzv. *cloudové* systémy od více poskytovatelů, tak toto rozhodnutí nebylo snadné, protože všechny mají svoje výhody a nevýhody.

V současné době se ve firmě Xura používají platformy pro služby v tzv. *cloudu* od následujících poskytovatelů:

- NaviSite
- NTT
- Amazon

A také se používá open-source projekt *OpenNebula*. Řešení, které se nabízejí k použití pro výkonnostní a zátěžové testování jsou tedy privátní, veřejný a hybridní cloud. I výběr tohoto systému byl důležitý. Jednotlivé typy a zvolení poskytovale budou popsány v následujících podkapitolách [19].

6.2.1 Privátní cloud

Tento typ je hodně využíván společnostmi kvůli jeho spolehlivosti a bezpečnosti. Jednotlivé vlastnosti jsou popsány níže:

- Dedikovaný výpočetní výkon
- Škálovatelnost
- Bezpečnost
- Flexibilita
- Vyšší cena - i oproti ostatním řešením od NaviSite
- Rozšíření celkového řešení

6.2.2 Veřejný cloud

Pokud zahrneme veřejný cloud, který je také hojně používaný, hlavně díky jeho nižší cenové náročnosti, tak jeho výhody a nevýhody jsou následující:

- Jednoduchost použití
- Nižší cena oproti privátním cloudům
- Správa externí firmou
- Flexibilní
- Nespolehlivý
- Potenciálně nebezpečný

6.2.3 Hybridní cloud

Poslední zmíněné řešení je tzv. hybridní cloud. Tento cloud poskytuje rovnováhu mezi spolehlivostí, bezpečností a jednoduchosti použití. V roce 2015 mnoho firem začalo využívat právě tento model. Hybridní cloudy bohužel postrádají některé výhody privátních či veřejných cloudů, nicméně výhody jsou shrnuty v následujících bodech:

- Flexibilní
- Škálovatelný
- Cena pohybující se někde mezi privátním a veřejným cloudem
- Jednoduší migrace mezi jednotlivými řešeními

6.2.4 Poskytovatel prostředí

Jelikož většina cloudových operací je zpoplatněna poskytovatelem za využité zdroje, tak volba poskytovatele nebyla jednoduchá. Pravidelné výkonnostní a zátěžové testování je jedním z ukazatelů kvality produktu a proto i placené řešení by mělo potenciálně smysl. Nicméně jsem vybral platformu OpenNebula, která je open-source a není tudiž zpoplatněn. Důvodem ke zvolení této varianty byla hlavně možnost spravovat si cloud samostatně. Vývojáři mají možnost si jej sami spravovat, což přináší výhody v možnosti si upravit virtuální stroje dle potřeby a preferencí. Další výhodou je rovněž uživatelská znalost tohoto prostředí, která dovoluje spravovat jednotlivé virtuální stroje bez větších problémů. Z výše vypsaných typů cloudových prostředí jsem zvolil privátní cloud, převážně díky jeho dedikovanému výkonu, který je nezbytný pro výkonnostní a zátěžové testy.

6.3 Databáze

Jakmile bylo zvoleno prostředí a poskytovatel virtuálního prostředí, na kterém budou pouštěny výkonnostní a zátěžové testy, tak bylo třeba přejít k zvolení databáze. Naskytovaly se různé možnosti databází, které se ve firmě využívají a jsou uživatelsky i administrátorsky příjemné. Všechny tyto možné databáze budou podrobněji popsány v následujících podkapitolách. Shrnutí a vybrání jedné z níže popsaných databází bude následovat.

6.3.1 MySQL

Jedna z nejpopulárnějších databází v rámci celého světa. MySQL je taktéž open-source projekt. MySQL je relační typ databáze a je hojně využíván ve webových aplikacích. MySQL beží na všech vetších platformách. Stejně jako Kyoto, tak je i MySQL napsáno v jazyce C a C++. MySQL je distribuováno ve dvou verzích. Jedna z těchto verzí je dostupná volně ke stažení a využití. Druhá verze je placená a obsahuje mnohem více funkcí a hlavně obsahuje podporu. MySQL podporuje databázové kurzory, zanořené dotazy, indexování a mnoho dalšího. [15]

6.3.2 MongoDB

Oproti MySQL a také PostgreSQL (popsána níže) je tato databáze typu NoSQL. Což znamená, že místo tradicních relacních databází, které využívají tabulky, používá dokumenty ve tvaru JSONu a také dynamické databázové schéma. Toto vede k rychlejšímu vytvárení

a integraci dat v aplikacích. MongoDB je open-source databáze a řadí se mezi nejpopulárnejší NoSQL databáze. I mnoho větších projektů využívá tento typ databáze namísto klasických relačních databází jako jsou MySQL a PostgreSQL [14].

MongoDB má mnoho funkcí, které zrychlují práci s databází a některé z nich nenajdeme v relačních databázích. Mezi některé tyto funkce patří následující:

- Ad hoc dotazy - vyhledávání podle polí či regulárních výrazů.
- Dokumentačně orientované - nevyužívá rozsáhle tabulkové struktury, ale může si uchovávat rozmanité informace, které by byly roztroušeny po tabulkách v jednom objektu.
- Souborový systém - MongoDB muže být také použit jako souborový systém, který rozprostírá data po více uzlech a muže také sloužit k rozdelení zátěže.
- MapReduce - dávkové zpracování dat a agregace, tato funkce se používá prevážne u velkého objemu dat.

6.3.3 Kyoto

Kyoto databáze je již starší projekt a v současné době již nemá podporu. Kyoto databáze je implementována převážně v jazyku *C* a z tohoto důvodu je rychlá. Samotná práce s Kyoto databází zahrnuje i psaní skriptů v *lua* jazyce. V současné době je ve firmě Xura používaná v několika produktech, hlavně díky její rychlosti. Uložení dat může být perzistentní, ale i volatilní. Pokud je řeč o perzistentních datech, tak ty jsou uloženy na disku a pro aplikaci, která vychází z této diplomové práce, je tedy nezbytná. Jelikož volatilní databáze se uchovává v RAM, tak je tedy rychlejší z hlediska přístupu k datům. Nicméně po restartu systému by se její data ztratily a to je pro tuto aplikaci nežádoucí. Co se týče celé Kyoto databáze, tak složitost nastavení a všeobecné používání je mnohem vyšší, než-li u ostatních vyjmenovaných databází a z tohoto důvodu nebude tato databáze v aplikaci použita. [9]

6.3.4 Redis

Redis neboli Remote Dictionary Server muže také sloužit jako databáze. Jedná se o NoSQL databázi, která je populární. Perzistence v této databázi je zajištena dvěma zpusoby. Obdobne jako Kyoto muže být tato databáze použita jako *on-memory database*. Tento pojem byl již vysvetlen dříve u Kyoto databáze. Co se týče konzistentní databáze, tak ta je zajištěna následovně [21]:

- Snapshotting - semi-persistentní mód, který zajišťuje uchovávání dat na disku po určitých časových intervalech.
- Žurnálování - tento druhý způsob perzistentního uchování dat na disk využívá soubor, do kterého se může pouze připisovat. Redis zajišťuje na pozadí přepisování tohoto souboru, aby nedocházelo k jeho nekonečnému růstu.

6.3.5 PostgreSQL

PostgreSQL je open-source databáze využívající relační schéma. V rámci firmy je používaná ve více projektech a její použití a administrace je jednoduchá. Stejně jako MySQL databáze, tak je PostgreSQL databáze velmi rozšířená a existují stovky návodů jak s ní zacházet a spravovat. Jak PostgreSQL tak i MySQL databáze jsou již velmi dlouhotrvající

projekty. Počet uživatelů MySQL databází je sice vetší, než-li počet uživatelů využívající PostgreSQL, ale v celosvětovém měřítku je toto nesignifikantní důvod k využití MySQL databáze oproti PostgreSQL. Oproti MySQL využívá PostgreSQL lehce odlišnou syntax dotazovacího jazyka. Jedním z rozdílů, který se pozná při bežné praxi s MySQL a PostgreSQL je automatická inkrementace jednoznačných identifikátorů řádků v tabulce, neboli primárního klíče. V PostgreSQL se k tomuto úcelu využívá typ *serial*, který automaticky inkrementuje číselnou hodnotu o jedna. V MySQL se používá *AUTO_INCREMENT*, který plní stejnou funkci, nicméně při vytváření tabulky je nutno ješte u daného sloupce zadat datový typ.

Kvůli open-source projektu PostgreSQL a také kvuli předchozí zkušenosti s tímto typem databáze, jsem jej zvolil místo MySQL a MongoDB, které se také k této aplikaci hodí. Důvodem nepoužití MongoDB byl fakt, že v soucasné době není používaná lidmi, kteří budou se systémem na výkonnostní a zátežové testy pracovat. Ostatní typy databází byly zavrhnuty kvůli jejich složité administraci, manipulaci a nevhodnosti použití pro webové aplikace [18].

6.4 Použité jazyky

Při vzniku jakékoliv aplikace je nezbytné vybrat jazyky, které budou použity ve vývoji. Pro udržování a potenciální vývoj v budoucnu je potřeba vybrat moderní jazyky a technologie, které budou i za několik let stále používány. V této části budou tedy popsány použité jazyky a co stálo za jejich výběrem.

6.4.1 ReactJS

Kromě klasicky používaných jazyků, jako jsou HTML5 a CSS, pro vytváření uživatelských prostředí ve webových aplikacích jsem použil ReactJS, což je vlastně knihovna postavená nad javascriptem. ReactJS je novou technologií na poli vývoje front-endu. V poslední době je vysoce prosazován společností Facebook, inc. Jedná se o formu javascriptu, který by se dal zařadit jako *view* neboli pohledová vrstva v rámci MVC(*Model, View, Controller*) architektury. Oproti ostatním jazykům nebo rozšířením, které se používají pro vývoj front-endu, se ReactJS vyznačuje používáním virtuálního *DOMu* (*Document Object Model*).

Document Object Model je reprezentace strukturovaného dokumentu skrze objekty nezávisle na platformě. Problém, který řeší ReactJS je právě neoptimalizovanost výše zmíněného modelu při vytváření dynamických uživatelských rozhraní. Virtuální DOM stále využívá standardní model, ale snaží se ho používat co nejméně a co nejefektivněji. Tedy místo práce s klasickým DOMem používáme virtuální verzi. Jelikož je při práci s dynamickým uživatelským rozhraním nejpomalejší část zobrazení u klienta, tak se tedy aplikují pouze změny mezi standardním DOMem a virtuálním [35].

Hlavním důvodem vybrání ReactJS pro implementaci front-endu byla skutečnost, že je velice rychlý, moderní a jelikož je prosazován společností Facebook, Inc., která vlastní v současné době jeden z nejnavštěvovanějších webových portálů, tak předpokládám, že se bude věhlas této knihovny rozrůstat.

6.4.2 NodeJS

NodeJS je v dnešní době stále více používaným jazykem, který se spoléhá na asynchronní komunikaci. Je založen na javascriptu a používá neblokující I/O model. Existuje mnoho

frameworků, například *ExpressJS*, který bude využit v systému na výkonnostní a zátěžové testování. NodeJS implementuje serverovou část webových aplikací. ExpressJS je využíván na routování a potřebné operace na pozadí, které jsou nezbytné pro správné zobrazení uživatelských dat.

6.4.3 Bash

Bash je jeden z unixových shellů, který je běžně používaný. V systému na výkonnostní a zátěžové testování bude tento jazyk použit právě na automatické spouštění skriptů, které vedou k samotnému testování a ke shromažďování výsledků. Toto spouštění společně se shromažďováním výsledků bude dále popsáno v popisu samotného systému. Jako alternativu bashe lze využít *python*, *perl*, *PHP* a podobné skriptovací jazyky, nicméně bash byl vybrán kvůli jeho plošnému využití a pohodlnější práci se soubory. V ostatních skriptovacích jazycích by implementace jednotlivých skriptů byla ale také možná.

6.4.4 PHP

Použití PHP ve výsledném systému bylo pouze z hlediska jednodušší komunikace s databází PostgreSQL, jenž byla popsána výše. Sice se jedná o skriptovací jazyk, ale jak již bylo zmíněné, tak celková implementace v PHP bez použití výše zmíněného bashe by byla nerozumná a zbytečně komplikovaná.

6.5 Tvorba prostředí pro vývoj

Pro spuštění a běh serveru jsem se rozhodl využít nástroj *webpack*. Webpack je tzv. *module bundler*, který se stará o generování výsledné podoby stránek. Webpack tedy bere v úvahu všechny definované soubory, závislosti a akce, které se mají provést při spuštění. Jeho hlavní výhodou je možné přizpůsobení, relativně krátký čas spuštění webového serveru s definovaným obsahem a lehké doplnění dalších modulů, nebo softwaru třetích stran. Kromě výše uvedených vlastností se dá také nakonfigurovat s použitím dalších knihoven tak, aby hlídal soubory, které jsou definované, na změny a automaticky prováděl znovu načtení stránky při změně daného souboru. Tato vlastnost se hodí převážně během vlastního vývoje, protože šetří šas. Jak již bylo zmíněno na dříve, tak pro instalaci závislostí se použije nástroj npm. Popis a použití toho nástroje lze najít na [16]. Pro mapování závislostí a potřebných balíčků pro běh bude použit soubor *package.json*.

6.6 Uživatelská část

Kromě výše zmíněných jazyků ve vývoji jsou ještě použity HTML5 a kaskádové styly, neboli CSS. Všechny tyto zahrnuté prvky jsou standardními nástroji na správu a vytváření webových prezentací. Ve vývoji uživatelské části jsem se snažil o co nejintuitivnější a nejjednoduší ovládání celé aplikace. Jednotlivé prvky jsem se snažil skloubit ve firemním stylu. Inspirací pro použití grafiky jsem nalezl na stránkách společnosti [36].

Ve většině případů jsem tedy používal barvy fialové, modré a bílé. Obrázky, které jsou použity jsou buďto firemním vlastnictvím, stejně jako logo společnosti, nebo jsou volně dostupné pod odpovídající licencí na internetu a mohou být použity i ke komerčním účelům.

Grafy, které zobrazují samotné chování systému pod určitou zátěží jsou generovány knihovnou *react-highcharts* [20]. Tato knihovna vychází z knihovny *Highcharts* [12]. Použitá knihovna v ReactJS obsahuje pouze grafovou část knihovny Highcharts. Osatní prvky Highcharts nejsou v systému na výkonnostní a zátěžové testování použity. V dalších částech této práce budou zobrazeny grafy vygenerované touto knihovnou reprezentující jednotlivé systémové vlastnosti při určité zátěži. Ve všech grafech je poté možno přibližovat, vypínat/-zapínat různé datové části, rozdrobit graf o více datových linkách na jednotlivé grafy blíže specifikující dané měření, stáhnout reprezentaci grafu v JSON souboru, aby bylo možné měření archivovat i na jiných systémech a dalších mnoho věcí.

Výsledná webová prezentace podporuje pouze novější prohlížeče. Podpora pro Internet Explorer verze 9 a níže není z důvodů nekompatability použitých knihoven a jednotlivých prvků na stránce. Podporované verze této webové aplikace tedy jsou:

- Mozzila Firefox
- Chrome
- Internet Explorer verze 10 a výš
- Edge

U ostatních webových prohlížečů není zaručené výsledné rozložení a design.

6.6.1 Úvodní obrazovka

V úvodní obrazovce je možno nalézt informace týkající se přihlášení či registrace do systému pro výkonnostní a zátěžové testy. Celková správa uživatelů probíhá pomocí autentizace vůči lokální databázi, jak již bylo dříve popsáno. Pokud není ještě uživatel přihlášen, jsou mu nabídnuty následující možnosti:

- **Registration** - registrace uživatelů do systému na testování aplikačního serveru Xtend
- **Login** - přihlášení do systému
- **Forgot Password** - zaslání nově vygenerovaného hesla na registrační email, pokud uživatel zapomene své heslo

Při registraci je nutné vložit email a další povinné údaje, které jsou vyžadovány a podle kterých jsou jednotliví uživatelé rozpoznáváni. Pokud tuto obrazovku navštíví již zaregistrovaný uživatel, jehož údaje jsou uloženy v prohlížeči a tudíž je přihlášen, tak následuje automatické přesměrování na obrazovku, která již poskytuje možnosti zaregistrovaného uživatele. Údaje o uživateli po přihlášení jsou uloženy v tzv. *Session Store*.

6.6.2 Hlavní obrazovka

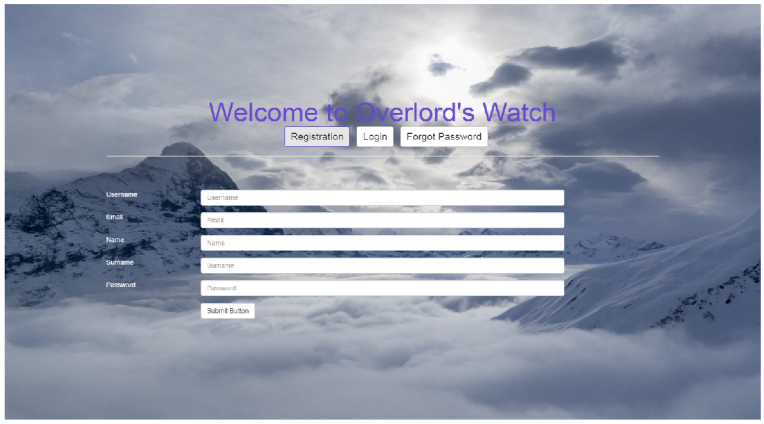
V dnešní době je velmi velký trend webových prezentací na jedné stránce, která se posouvá pouze nahoru a dolů. Stále častěji se můžeme s těmito stránkami setkávat a uživatelé si na ně postupně zvykají a podle malého průzkumu se jim i líbí více, než-li stránky na kterých musí uživatel překlikávat mezi jednotlivými částmi. V tomto duchu jsem se snažil vytvořit výsledný systém. Jelikož díky technologii, která byla použita, je možné takovouto stránku provozovat s minimálním zpožděním při provádění jednotlivých akcí, tak jsem zvolil právě

tuto možnost. Na celý koncept této hlavní stránky byla použita knihovna *react-fullpage*, která byla jako ostatní knihovny a moduly nainstalována pomocí zmíněného nástroje npm.

React-fullpage knihovna se stará o zobrazování jednotlivých částí vždy na jednu obrazovku uživatele. Navigace je možná pomocí šipek na klávesnici, stejně jako pomocí kolečka u myše. Při posouvání se tedy mění jednotlivé sekce tohoto systému. Všechny důležité části jsou právě zobrazeny na hlavní stránce v jednotlivých sekcích. Pro jednodušší možnost pohybu po hlavní stránce je v horní části lišty připevněno menu, které obsahuje následující odkazy (jednotlivé sekce lze také nalézt v příloze této diplomové práce):

- **What is this?** - zobrazení úvodní obrazovky přihlášeného uživatele na hlavní stránce s názvem systému a logem firmy
- **How it works?** - popis jednotlivých částí a možností systému
- **Upload file** - sekce pro jednorázové nahrání souboru do systému
- **User management** - sekce pro jednotlivé uživatelské nastavení a přehled vykonávaných operací
- **Home(*email*)** - kliknutím na tento odkaz se uživatel dostane na Úvodní obrazovku, která byla popsána v předcházející podkapitole

Popis jednotlivých sekcí je v dalších částech této diplomové práce. Co se týče rozložení stránky, tak na to byla použita knihovna *react-bootstrap*. Tato knihovna umožňuje definování jednotlivých tříd pro prvky na stránce a tyto třídy jim pak přiřazují CSS styly. V dnešní době je velmi používaná, protože díky ní se ulehčuje stylování dokumentu jako celku a pomáhá také při stylování stránky na různých rozlišeních a zařízeních. Zobrazení hlavní obrazovky lze vidět na obrázku 6.1.



Obrázek 6.1: Úvodní obrazovka s registračním formulářem.

What is this?

Tato sekce slouží pouze jako úvod do systému a je na ní vidět pouze logo firmy a název systému. Ilustrační obrázek této sekce lze nalézt na obrázku A.1.

How it works?

Tato sekce slouží pouze jako vyjmenování jednotlivých funkcí celého systému. Tato sekce je rozdělena na dvě části, kdy v jedné z nich se hovoří o vlastnostech systému pro dlouhodobé plánování výkonnostních a zátěžových testů a druhá část se zabývá krátkodobým testováním. Její náhled je dostupný v příloze A.2.

Test from file

Tato sekce se zaobírá samotným nahráním souboru z disku uživatele. Dominuje jí pulsující tlačítko *Upload file*. Z důvodů přitáhnutí pozornosti a kladné uživatelské odezvě jsem nastavil styly tohoto tlačítko tak, aby pulsovalo. Náhled na tuto obrazovku je možný v příloze A.3 Při kliknutí na toto tlačítko se zobrazí modální okno, které je také řešeno pomocí react-bootstrap knihovny. V tomto okně uživatel smí nahrávat své soubory pomocí knihovny *react-dropzone* a to do maximálního počtu 6 souborů. Nahrávání souborů probíhá pomocí zvolení souboru z nabídky, nebo pomocí přetažení souboru do vyznačeného okna. Samozřejmostí je také nahrávání více souborů najednou. Při přidání souboru ve formátu JSON, který musí splňovat strukturu, která je znázorněná na úryvku 6.1:

```
{
        "Name" : "Test Graph",
        "Throughput" : "50",
        "Date" : "05-23-16",
        "Description" : "Test description",
        "Service" : "Collect SMS",
        "time": ["08:00:01","08:10:01","08:20:01",
                "08:30:01","08:40:01","08:50:01",
                "09:00:01","09:10:01","09:20:01",
                "09:30:01","09:40:01","09:50:01"],
        "data" : {
                "series1" : ["94.38","42.41","37.28",
                             "35.92","44.48","49.79",
                             "94.38","42.41","37.28",
                             "35.92","44.48","49.79"]
                }
}
```

Soubor 6.1: Struktura souborů akceptovaných systémem.

Obdobná struktura souboru je také prezentována v otevřeném modálním okně s popisem.

Při vložení souboru do systému se automaticky zobrazí výsledný graf na hlavním okně a modální okno s možným nahráním souborů zůstává stále otevřeno. Takto si uživatel může nahrát až 6 souborů, z čehož mu může vzniknout stejný počet grafů. Při nahrání více souborů jsou grafy zmenšovány, aby zůstal přehled zobrazených grafů. Pokud se uživateli nelíbí některý nahraný soubor, tak jej může pomocí ikonky ve tvaru křížku v tomto modálním okně zahodit. I při této akci dojde k přeškálování výsledného zobrazení. Pokud se chce uživatel podívat, jaké budou zobrazené veličiny v jednotlivých grafech, tak k tomu slouží náhled na soubor. Ten se provede prostým kliknutím na daný soubor a v tomtéž modálním okně se zobrazí jednotlivé statistiky, které budou posléze zobrazeny.

V případě, že dojde k vložení chybného souboru, tak je tato chyba detekována a oznámena uživateli. Může dojít k následujícím chybám, které budou prezentovány uživateli:

- **Time values do not correspong to values of *series_name*** - rozdílný počet časových údajů od užitečných dat
- **Not all values are numbers in *series_name*** - v datech se nachází nečíselný údaj
- **Cannot parse JSON file, check syntax** - chybně strukturovaný soubor

V těchto případech dochází k odmítnutí souboru a varování uživatele. Ve všech případech dochází k informování uživatele o stavu prováděné operace. V dolní části modálního okna, které je znázorněno na A.3, jsou potvrzovací nebo zamítací tlačítka. Při zavření okna již není dostupné původní tlačítko na nahrávání souborů, ale je nahrazeno třemi tlačítky na smazání jednotlivých grafů z výpisu, smazání všech grafů, nebo přidání dalšího grafu. I při tomto přidávání platí podmínka na maximálně 6 grafů na stránce.

User&Test

Nejzajímavější sekcí uživatelského prostředí systému je bezesporu tato sekce. V záhlaví se nacházejí tři tlačítka na zobrazení jednotlivých panelů, které jsou generovány pomocí knihovny react-bootstrap. Můžeme zde tedy vidět následující možnosti:

- **Edit User**
- **Add New Job**
- **Edit&Show Jobs**

Postupně budou jednotlivé tlačítka probrány a jejich význam a obsah přislušných panelů vysvětlen.

Jako první bych rád představil tlačítko *Edit User*. Toto tlačítko zobrazí panel s uživatelskými informacemi. Jedná se o informace o uživateli z lokální databáze. Uživatel má poté možnost si změnit své jméno, příjmení a heslo. Aktualizace při odeslání formuláře proběhne automaticky a uživatel okamžitě vidí své změněné údaje. Odelsání formuláře je třeba potvrdit heslem a jako ve zbytku aplikace, i tady platí že uživatel je obeznámen o výsledků své akce.

Dále bych se rád věnoval tlačítku *Add New Job*. Toto tlačítko v sobě obsahuje přidání pravidelného nebo jednorázového výkonnostního či zátěžového testu. Toto tlačítko je pouze aktivní v době, kdy uživatel nemá 5 pravidelných testů. Při větším počtu je toto tlačítko neaktivní. Důvody pro omezení počtu uživatelských testů jsou prosté. Virtuální prostředí, kde jsou jednotlivé virtuální stroje není neomezené a v případě většího počtu pravidelných jobů by příliš toto prostředí zatěžovalo a ostatní členové firmy, kteří pracují na tomto prostředí by neměli prostor potřebný k práci. Při vytváření nového zátěžového testu jsou volitelné následující položky:

- **Test Case Name** - jméno scénáře, pod tímto jménem potom budou vystupovat jednotlivé výsledky grafů.
- **Username from Xura** - podle tohoto parametru se budou vytvářet virtuální stroje v cloudu, je tedy nezbytné, aby jméno bylo uživatelské jméno z firmy Xura, v opačném případě nedojde k vytvoření virtuálního stroje

- **Test Case Description** - popis scénáře, který se bude testovat
- **Throughput** - počet zpráv za sekundu, které budou generovány během testu, pouze celočíselný parametr
- **Duration** - délka testovacího scénáře v hodinách, pouze celočíselný parametr v rozmezí 0 - 72
- **Number of CPUs** - počet virtuálních procesorů, které budou přiděleny tomuto stroji
- **Memory** - odpovídá velikosti paměti, která bude během testu použita
- **Services** - ma výběr je v současné době pouze Collect SMS a GroupMessaging, odpovídá testovanému scénáři.
- **Scheduled Run** - pro pravidelné testování aplikačního serveru Xtend je možné použít denní, týdenní nebo měsíční testování, pro krátkodobé testování se vybere možnost *No* a test se spustí jednorázově.
- **Choose ini file** - tato možnost se zde vyskytuje kvůli možnosti nahrání si jiného konfiguračního souboru, než který je vybírán, když uživatel nevyplní tuto položku, slouží převážně pro testovací účely jednotlivých vývojářů a testerů

Pokud formulář neobsauje chyby tak se vloží záznam do databáze a daný test je spuštěn vždy o půlnoci (kromě krátkodobého testování), aby se zbytečně nezabíralo místo v cloudu. Pokud formulář chyby obsahuje, tak je uživatel na dané chyby upozorněn a je mu řečeno, kterou položku by měl opravit. Krátkodobé testování probíhá vždy okamžitě.

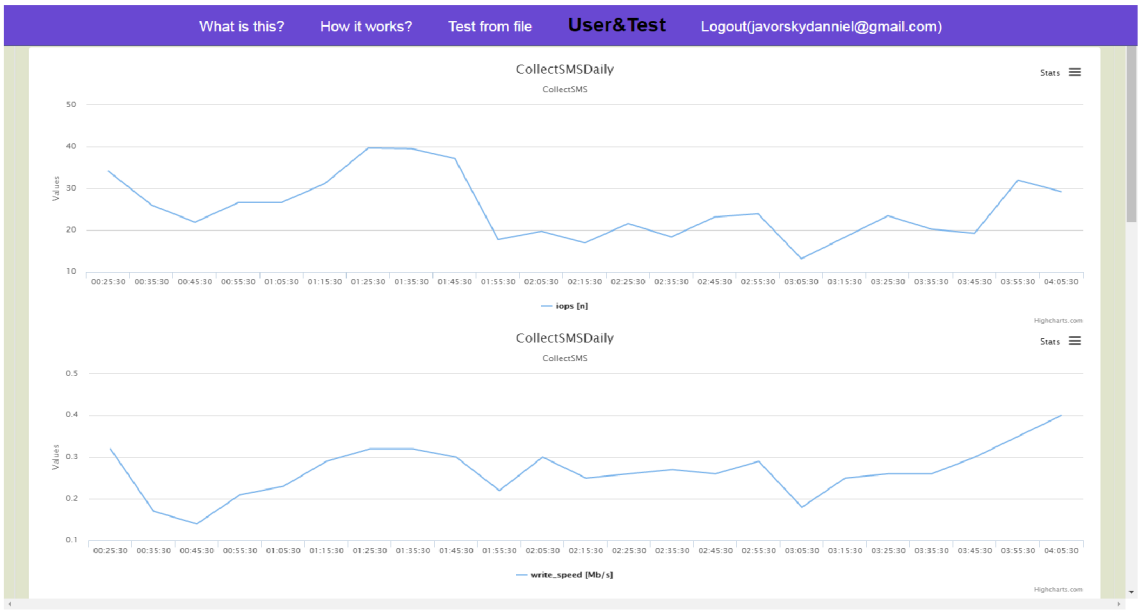
Poslední tlačítko, které se prozatím vyskytuje v nabídce běžného uživatele je *Edit&Show Jobs*. Pojmenování může být lehce matoucí, protože kromě úpravy již existujících testovacích scénářů, je zde také možnost si zobrazit daný testovací scénář se všemi jeho výsledkami v grafech. Ze začátku bych se tedy rád věnoval samotné úpravě testovacího scénáře. Samotná úprava je možná, pokud daný test právě neprobíhá. V případě, že probíhá, tak je možná úprava až v době, kdy doběhne. V hlavním panelu jsou tedy zobrazeny všechny testovací scénáře, které vytvořil daný uživatel. U každého testovacího scénáře je možná úprava kliknutím na jeho název. Úprava je obdobná jako smaotné vytváření testovacího scénáře a platí pro ní stejná pravidla. Dále se na řádku s názvem daného scénáře nachází také další dvě tlačítka a to následující:

- **Show Jobs** - toto tlačítko, pokud již daný scénář alespoň jednou doběhl (v případě dlouhodobých testů), zaručí zobrazení dalšího tlačítka v nabídce s popiskem názvu daného testovacího scénáře a zároveň se změní popisek z *Show Jobs* na *Hide Jobs*. Pokud testovací scénář ještě neproběhl, nebo pokud právě běží, tak je tlačítko *Show Jobs* zašedlé s informativním popiskem. Po kliknutí na nově otevřené tlačítko má uživatel možnost vidět dosavadní výsledky všech testů které k tomuto testovacímu scénáři proběhly. Stejně jako v případě samotného nahrávání souborů z lokálního systému platí, že grafy jsou škálovatelné a zobrazují se tedy přehledně i ve větším počtu. Uživatel má možnost skrýt dané vyskakovací okno pomocí tlačítka *Hide Jobs* v panelu *Edit Jobs* ve stejné sekci hlavní stránky. Grafy mají shodné vlastnosti jako dříve zmíněné grafy, které se nahrávaly pouze dočasně. Samozřejmostí je možnost otevření více paralelních oken s jednotlivými výsledky testů. Pokud se jedná o dlouhodobé měření,

tak v jednom grafu bude zobrazeno více dílčích měřených veličin. V případě krátkodobého měření, které proběhne pouze jednou, jsou jednotlivé dílčí měření prezentovány v samostnatných grafech pro větší přehlednost.

- **Delete** - toto tlačítko smaže daný scénář z databáze a nebudou tedy probíhat dále vákonnostní a zátěžové testy

Při dokončení jednotlivých testovacích scénářů je uživatel spraven o výsledku testu a o dostupnosti jednotlivých grafů pomocí automaticky zaslaného emailu na účet, který poskytl při registraci, více o této službě je k nalezení v odpovídající sekci v serverové části této diplomové práce. Rozložení popisované sekce v rámci stránky lze vidět v příloze A.4 a jeden test zobrazený v příslušném doplňujícím okně je poté k vidění na obárzku 6.2, který je přiložen níže.



Obrázek 6.2: Stránka s grafy z testování.

Co se týká administrátorského účtu, kterým disponuje manažer týmu, který vyvýjí produkt Xtend, tak to disponuje ještě tlačítkem *Remove Users*. Toto tlačítko zajišťuje možnost mazání uživatelských účtů.

Home

Toto je pouze odkaz na přihlašovací stránku do systému a tudíž to není sekce v rámci hlavní stránky. V menu, které je umístěno v horní liště je možnost vidět aktuálně přihlášeného uživatele.

6.7 Serverová část

Serverová část, neboli *backend*, bude implementována v rozšířeném a používaném jazyce NodeJS za použití ExpressJS frameworku [8]. Tento framework byl vybrán z důvodů jeho

rozšířenosti a robustnosti. V případě, že by byla potřeba tuto webovou prezentaci v budoucnosti rozšiřovat, tak to bude za použití tohoto frameworku mnohem snazší.

6.7.1 Zpracování uživatelských dat

V případě, že dojde k přidávání jednotlivých testovacích scénářů z webovehého rozhraní, které bylo popsáno dříve, tak se vloží záznam do databáze. Z této databáze jsou poté jednotlivé testovací scénáře vybírány. Jedná se o automatický proces, který je v případě plánovaných testovacích scénářů spouštěn vždy o půlnoci buďto každého dne, nebo vždy na začátku týdne, či měsíce podle zvoleného intervalu. V případě jednorázových testovacích scénářů je automaticky spouštěn a o výsledku je uživatel obeznámen prostřednictvím emailu, jak již bylo zmíněno výše. Samotné posílání a provádění testovacích scénářů bude popsáno dále.

Na serverové části existují dva skripty. Pro každou možnost (dlouhodobé/krátkodobé testování) je to tedy jeden. Jednotlivé skripty jsou následující:

- getRunOnlyJobs.php
- getScheduledJobs.php

Co se týče dlouhodobého testování, tak skript *getScheduledJobs.php* přijímá parametr, který určuje periodu spouštění.

Spouštění je iniciováno pomocí *crontabu*. Crontab je softwerový démon, který dovoluje automatické spouštění skriptů v určitý čas. Záznamy, které se přidávají do crontabu mají následující syntax:

```
0 0 * * * /root/overlord/server_tools/getScheduledJobs.php
```

Soubor 6.2: Struktura záznamu v crontabu.

Kdy jednotlivé pole mají následující význam:

- **0** - minuta (0-59)
- **0** - hodina (0-23)
- ***** - den (1-31)
- ***** - měsíc (1-12)
- ***** - den v týdnu (0-7, příp. anglické zkráceniny jednotlivých dnů)

Pokud se použijí místo některých čísel v odpovídajících polích zástupné znamky *, pak to znamená, že se to má vykonávat pro všechny případy, připomíná to tedy regulární výraz. Ukázka, která je použita k vysvětlení funkce a použití crontabu je aktuálně použita pro spouštění testovacích scénářů s periodou jednoho dne. Obdobně jsou poté nastaveny i další periodicky spouštěné testovací scénáře. Jelikož by toto nastavení nešlo uplatnit na jednorázové testovací scénáře, tak jsem v jejich případě použil obdobný skript, jako je skript, který spouští testovací scénáře s denní periodou. Nicméně v tomto případě dochází k předávání důležitých argumentů přímo přes parametry v příkazové řádce a nejsou získávány dynamicky z databáze během běhu skriptu.

6.7.2 Vytvoření virtuálního stroje

Jedním z prvních kroků každého skriptu, který je buďto spouštěn z crontabu nebo přímo z webového nástroje je vytvoření virtuálního stroje, který bude testován. Tento úkol zahrnuje především aktualizování instalačního skriptu pokaždé, když je spouštěn, aby docházelo k testování poslední stabilní verze softwaru. Stabilní verze se myslí aktuální verze, která je dostupná prostřednictvím verzovacího systému *GIT*. Tento nástroj slouží k verzování softwaru. Jelikož je to běžně používaný software a v systému na výkonnostní a zátěžové testování je pouze použit v tomto bodě, tak nebude dále rozebrán. Informace týkající se verzovacího systému je možné nalézt na webových stránkách [10].

Po aktualizaci skriptu následuje specifikace daného virtuálního stroje. Specifikací se myslí, na kterém hardwaru daný virtuální stroj poběží. Existuje více možností a preferovanou volbou je hardware přímo dedikovaný týmu, který se zabývá vývojem aplikačního serveru Xtend. Pokud na tomto hardwaru není dostatek výpočetních prostředků a místa, tak je virtuální stroj vytvořen na jiném stroji, který už je ovšem sdílen mezi týmy firmy. Z databáze pro odpovídající testovací scénář jsou vybrány parametry velikosti paměti a počtu virtuálních jader, které jsou důležité pro testování.

Proces vytváření virtuálního stroje na platformě OpenNebula je tedy automatizovaný úkon. Při vytváření se vytvoří dva virtuální stroje. Jeden představuje samotnou instalaci aplikačního serveru Xtend a druhý slouží jako server pro Group Messaging a ostatní služby. IP adresy jednotlivých strojů v privátní síti jsou uloženy do souborů a tyto soubory jsou poté použity jako zdroje jednotlivých IP adres. Po vytvoření daných strojů probíhá instalace aplikačního serveru Xtend a všech prerekvizit, které jsou potřebné pro jeho správný běh. Pomocí *ssh* jsou na dané servery automaticky nakopírovány jednotlivé instalační soubory a balíčky a poté proběhne jejich instalace.

Po dokončení instalace je zkontrolován stav daného aplikačního serveru. V dalších krocích probíhá nastavení samotného nástroje na generování Collect SMS dialogů a zpráv v Group Messagingu. Collect SMS služba byla popsána v teoretické části této diplomové práce. Nástroj byl původně vytvořen při dřívějším výkonnostním a zátěžovém testování aplikačního serveru Xtend. Nicméně při nynějším testování bylo třeba nástroj přepracovat. Abych mohl popsat jednotlivé změny, tak je třeba si trošku přiblížit nástroj na generování dialogů, který byl použit jako základ samotného testování.

6.7.3 Nastavení nástroje Overlord

Nastavení samotného aplikačního serveru probíhá překopírovaním konfiguračního souboru. V této části jsou dvě možnosti. Jedna z nich je překopírování standardního konfiguračního souboru, který byl vybrán a vytvořen pro potřeby tohoto testování. Druhou možností je nahrání vlastního konfiguračního souboru, který můžou využívat vývojáři a testeři tohoto produktu s vlastními definovanými kroky a přechody. Při této možnosti není zaručena správnost konfiguračního souboru a je tedy možné, že jednotlivé testovací scénáře nebudou odpovídat realitě a že jednotlivě posílané zprávy nebudou správně zpracovány na daném aplikačním serveru.

Mezi standardně měněné parametry při výkonnostním a zátěžovém testování pomocí webového nástroje patří IP adresa, port a počet generovaných zpráv za sekundu.

6.7.4 Možnosti spouštění nástroje Overlord

Jakmile jsou výše uvedené operace dokončeny, tak se přechází k samotnému spuštění nástroje Overlord. V této části je vhodné poznamenat existenci více možností samotného měření.

- Spouštění na virtuálním stroji, kde byl čerstvě nainstalován Xtend
- Spouštění na serveru, kde běží webový server

Obě tyto možnosti mají své limity a úskalí. Pokud bychom spouštěli nástroj na generování SMS zpráv na serveru s čerstvě nainstalovaným Xtendem, tak bychom se dostali do problémů s reálným využitím zdrojů. Aby měření bylo kompletní a správné a výsledné hodnoty nebyly zavádějící, tak měření nemůže proběhnout tímto způsobem z důvodů alokace zdrojů samotným nástrojem. Využití paměti by bylo mnohonásobně vyšší stejně jako využití procesoru. Při měření druhým možným způsobem tedy dostáváme jednoznačné výsledky, jelikož nebude žádný jiný provoz na vytvořeném serveru v daný okamžik. Výsledky jednotlivých měření se ale mohou lehce lišit v závislosti na aktuálním vytížení fyzického serveru, nicméně toto jsou prokazatelné výsledky, narozdíl od metody, která byla popsána jako první. Při spuštění nástroje Overlord dochází k simultálnímu připojení na server pomocí ssh a spuštění několika utility, které zaznamenávají aktuální vytížení zdrojů. Mezi těmito nástroji nalezneme sar a iostat, které byly popsány v teoretické části této diplomové práce. Jejich délka je přímo úměrná délce běhu nástroje Overlord. Perioda získávání výsledků pomocí těchto nástroje je poté 10 minut. Získávání výsledků v pravidelnějších intervalech by sice nemělo závažný vliv na výkon virtuálního stroje, ale z hlediska porovnávání výsledků není nezbytný.

6.7.5 Sběr výsledků

Důležitým krokem v celém procesu výkonnostního a zátěžového testování je samozřejmě sběr výsledků. V systému, který se stará o tento typ testování, je tato funkcionalita zajišťována pomocí bash skriptu. Tento script se spouští pomocí záznamu z démona cron. Záznam, kdy spustit tuto úlohu je vložen do cronu při startu nástroje Overlord. V této době dochází k dopočítání si přesného času, kdy doběhne samotné testování. Cron tedy kromě periodických testů, které byly popsány dříve, obsahuje také záznamy o spouštění skriptů na sběr výsledků. Pomocí ssh se script připojí na cílový stroj a následně zkopíruje výsledky z cílového stroje na server, odkud byl nástroj Overlord pouštěn.

6.7.6 Uložení výsledků

Jakmile jsou jednotlivé soubory s informacemi o systémovém chování pod určitou zátěží přeneseny na server, kde běží webový nástroj, tak dochází k jejich zpracování. Zpracování probíhá následujícím způsobem:

- získání obsahu souboru na *stdout* pomocí nástroje cat
- použití regulárního výrazu nebo nástroje *awk*
- uložení získaného výsledku do proměnné

Jakmile jsou všechny nezbytně nutné veličiny takto zpracovány, následuje vytvoření samotného souboru typu JSON. Do něj se vloží všechny výše získané proměnné společně s názvem grafu, počtem zpráv za sekundu, popisem testu, datem provedení testu, časovou hodnotou určující jednotlivé úseky měření a názvem testu.

6.7.7 Zobrazení výsledků

Výsledky, které jsou získány z jednotlivých měření musí být samozřejmě uloženy na disku tak, aby je bylo možné najít. Organizace uložiště výsledků plyne ze struktury adresářů. Jednotlivé výsledky testovacích scénářů jsou vloženy do adresáře s unikatním identifikátorem scénáře. Tato složka je poté vložena do složky s názvem služby, která se testovala. V době vzniku textu této diplomové práce to jsou pouze služby Collect SMS a Group Messaging. A jako hlavní identifikátor je použit email uživatele, který tento testovací scénář vložil přes webový nástroj do databáze. Jelikož se jedná o relativně komplexní způsob ukládání výsledků, tak je výsledná složka sestavena na základě výše zmíněných atributů. Jako ilustrace poslouží následující ukázka:

```
users/email.test@seznam.cz/CollectSMS/0/
```

Soubor 6.3: Uložiště výsledků.

Rozbor jednotlivých částí této adresářové struktury je následující:

- **users** - označení složky s relevantními výsledky testů
- **email.test@seznam.cz** - jednoznačný identifikátor uživatele, který vložil daný testovací scénář do databáze
- **CollectSMS** - označení služby
- **0** - id podle databáze testovacího scénáře

6.7.8 Smazání virtuálního stroje

Kdyby neprobíhalo automatické smazání virtuálního stroje, tak by brzy došlo k zaplnění místa a využití všech možných zdrojů. Proto je tedy použit nástroj na automatické mazání virtuálních strojů. Je spouštěn po doběhnutí výkonnostních a zátěžových testů. Společně se sběrem výsledků je tedy vkládán do Cronu. Čas jeho spuštění je krátce po sběru výsledků.

Kapitola 7

Testování webové aplikace

V této části bude popsáno testování systému, který byl navržen. V rámci této kapitoly budou poté probrány jednotlivé části testování a to jak manuální, tak automatické.

7.1 Automatické testování

Automatické testování může být prováděno více způsoby, jak již bylo popsáno v předcházejících kapitolách. Automatické testování webových systémů může být ale také prováděno specializovanými nástroji. Jedním z těchto nástrojů je poté *Selenium* [11]. Tento nástroj slouží pro automatizované testy, podporuje většinu globálně používaných prohlížečů a je využíván velkými společnostmi na testování jejich webových portálů. Pomocí Selenia je možné otestovat převážně velké webové systémy, ale jeho využití se najde i v testování menších systémů. Testování nástrojem Selenium se skládá z vlastnoručně napsaných a definovaných skriptů obsahujících testovací scénáře. Tyto skripty poté provádějí samotné testování. Jelikož je systém na výkonnostní a zátěžové testování menšího charakteru, tak Selenium není využito pro jeho testování.

7.2 Manuální testování

Jelikož není v aplikaci využito automatické testování, tak bylo třeba přijít s jiným řešením. Tímto řešením se právě stalo manuální testování. Tento typ testování probíhá definováním akcí a reakcí. Pokud tedy zmáčkneme na webové stránce tlačítko, tak očekáváme nějaký výsledek. Při vývoji probíhalo manuální testování neustále. V současné době byl systém na výkonnostní a zátěžové testování otestován i jednotlivými členy týmu, kteří s ním pracují. Pokud by se vyskytla chyba, která by ovlivňovala chod systému, tak by v současné době již byla odhalena.

V případě přidávání další funkcionality do stávajícího systému by proběhlo manuální testování znova. A při zjištění chyby by došlo k její nápravě.

Kapitola 8

Výsledky výkonnostního a zátěžového testování

V této kapitole se budu krátce věnovat samotným výsledkům výkonnostního a zátěžového testování. V první části budou prezentovány výsledky, kterých bylo dosaženo manuálním testováním a v další se budu věnovat výsledkům ze systému, který jsem implementoval.

8.1 Nastavení serveru Xtend

Manuální testování se primárně skládalo z testování jednotlivých nastavení, které jsou možné v rámci služeb Collect SMS a Group Messaging. Hlavně jsem se tedy zaměřil na Collect SMS, jelikož je tato služba používanější oproti Group Messagingu. Nastavování služby Collect SMS bylo zdlouhavé a vždy bylo třeba testovat daný scénář po určitý čas, aby se došlo k relevantním závěrům. Velký dopad na službu měl čas odeslání potvrzovací odpovědi a počet procent uživatelů, kteří neodpoví vůbec. Jak již bylo zmíněno dříve, po nezodpovězení se posílá ještě tzv. *reminder*. Nemusí být poslán vůbec, ale ve většině případů dochází k zaslání alespoň jednoho. Tento reminder připomíná uživateli, že mu byla zaslána SMS a čeká se, zda-li ji přijme nebo ne. Celou dobu musí být v databázi uchován kontext o této zprávě. Z databáze jsou kontexty zprávy mazány pouze v případě, že vyprší celkový limit na SMS nebo uživatel, který obdržel zprávu o zaplacení SMS odpoví negativně. Dalším signifikantním vlivem na samotné testování bylo nastavení doby pro přijetí potvrzovacího signálu. V reálných systémech k tomuto zaslání dochází v rámci sekund, nicméně při vysoce vytížené síti se toto potvrzení může zpozdit, proto bylo nutné upravit čas na potvrzení na jinou hodnotu, než je při standardní instalaci aplikačního serveru Xtend. Při vyšším počtu zpráv, které byly generovány bylo třeba také změnit počet spojení s databází a počet vláken k tomu přidružených. Při vyšším počtu spojení a vláken zvládal system větší zátěž. Co se týče samotné databáze, tak ta take prošla testováním. V rámci zjištění tzv. *bottlenecku* se použila on-memory database, která byla popsána již dříve, při výběru databáze vhodné pro system na výkonnostní a zátěžové testování. Nicméně z důvodů zachování dat není možné tuto databázi použít v produkčním prostředí, protože by při restartu došlo ke ztrátě dat. Samozřejmě při manuálním testování docházelo k vetšímu počtu změn, ale jelikož se jedná o produkt firmy, tak tyto změny zde nebudou probírány a diskutovány.

8.2 Prezentace výsledků

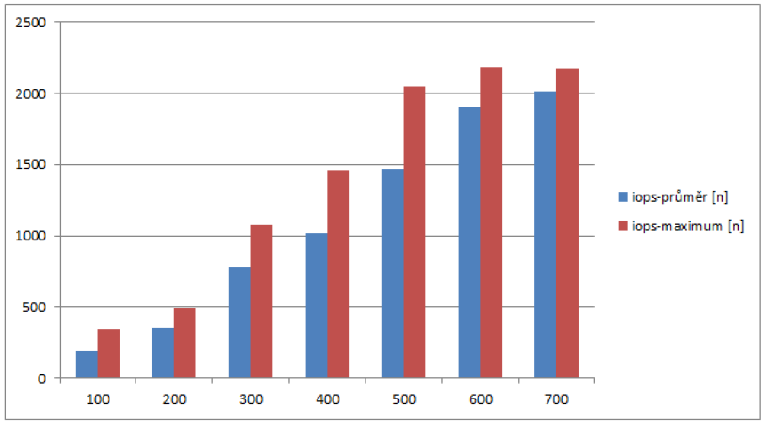
Výsledky výkonnostního a zátěžového testování byly prezentovány před týmem, který se zabývá vývojem dané aplikace. Všechny testovací scénáře, které jsou zde prezentovány mají následující konfiguraci:

Počet jader	16	28
Hyperthreading	ne	ano
Celková paměť v GB/	16	128
Velikost disku	40	600
Operační systém	CentOS (RHEL_R6.5-00)	CentOS (RHEL_R6.5-00)
Verze aplikačního serveru Xtend	Xtend_R2.0-01	Xtend_R2.0-01

Tabulka 8.1: Použité zdroje na testovaných serverech.

8.2.1 Rozdíly mezi virtuálním strojem a hardwarem

Rozdíly, které jsou patrné mezi virtuálním strojem a hardwarem mají příčinu v odlišné konfiguraci a počtu dostupných zdrojů. Virtuální servery také sdílejí své zdroje s ostatními servery spouštěnými na daném fyzickém stroji. Celková propustnost u hardwaru proto byla mnohonásobně vyšší, než je tomu u virtuálních strojů. Jednotlivé strovnání chování aplikačního serveru Xtend pod určitou zátěží je možné nalézt v níže uvedených grafech. Pro ukázku jsem vybral graf znázorňující chování systému při 100 - 700 zprávách za sekundu. V grafu 8.1 je také vidět, kde byl nalezen strop hardwaru pro Collect SMS s výše uvedenou hardwarovou konfigurací stroje.



Obrázek 8.1: IOPS pro hardware.

Na ose x je možné vidět počet zpráv za sekundu a na ose y je poté možné vidět jednotlivé hodnoty. V grafu jsou uvedeny měření průměrných a maximálních hodnot. V výše zmíněném grafu je tedy možné viděť, že stropem této konfigurace byl právě disk. Při použití on-memory databáze se tento strop mnohonásobně zvýšil, nicméně tento typ databáze nemůže být použit z důvodů neperzistence dat.

Všechny grafy, které jsou prezentovány v přílohách jsou prezentovány podobným způsobem, na ose x je počet zpráv generovaných a poslaných za sekundu a na ose y jsou hodnoty.

- **paměť** - grafy B.1 a B.2 znázorňují využití paměti
- **procesor** - grafy B.3 a B.4 znázorňují využití procesoru
- **rychlost zápisu na disk** - grafy B.5 a B.6 znázorňují rychlost zápisu na disk
- **odeslané pakety** - grafy B.7 a B.8 znázorňují počet odeslaných paketů
- **odeslané kilobajty** - grafy B.9 a B.10 znázorňují počet odeslaných kilobajtů
- **iops** - graf B.11 znázorňuje počet diskových operací na virtuálním stroji

Co se týče zaznamenávání zpráv pro reporting, tak to bylo vypnuto během těchto manuálních testů, které jsou zde prezentovány. Jejich zapnutí mělo samozřejmě dopad na výkon systému. Výsledky, které byly nahrány do interní aplikace Crystal, samozřejmě počítají se zapnutým zaznamenáváním zpráv pro reporting.

Kapitola 9

Závěr

Tato diplomová práce se zaobírala systémem na výkonnostní a zátěžové testování aplikačního server Xtend. Tento produkt je vlastnictvím firmy Xura, Inc. a hlavní vývoj tohoto produktu je soustředěn v Brně. Jelikož se jedná o produkt, který je komerčně používán, tak v této diplomové práci nebyly prezentovány všechny interní struktury a komunikace, která probíhá v tomto serveru.

V úvodní části byla nastíněna problematika výkonnostního a zátěžového testování spojená s všeobecným pohledem na testování. Kromě těchto důležitých věcí byly take prezentovány možnosti agilního i neagilního vývoje softwaru. Jelikož je produkt Xtend vyvíjen agilně, tak právě na tyto metody byl kladen důraz.

Samotná praktická část této diplomové práce zahrnovala vytvoření systému na výkonnostní a zátěžové testování. Pro samotný vývoj aplikace, která by toto zajišťovala, bylo potřeba navrhnout a vybrat prostředí ve kterém se bude dané testování provádět. Při výběru prostředí byly brány do úvahy různé parametry. Po zvážení pro a proti byla vybrána platforma OpenNebula a typ cloudu byl stanoven na privátní.

Stejně jako výběr platformy a typu cloudu, ve kterém poběží testované virtuální mašiny, tak i výběr databáze podléhal zhodnocení z několika hledisek. PostgreSQL byla nakonec tedy vybrána jako databáze, kterou bude tento projekt používat. Systém, který jsem navrhl umožňuje plánování dlouhodobých i krátkodobých testovacích scénářů, přehled výsledků ve formě grafů, ruční vkládání grafů z textových souborů, vlastní konfiguraci spouštěného virtuálního stroje, výběr testované služby, vlastní konfiguraci nástroje na generování SMS zpráv a v neposlední řadě změnu uživatelských údajů. Navržený system byl implementován jako webová prezentace, která byla optimalizována pro novější verze standardně používaných prohlížečů. Kromě podpory velkých stolních zařízení s vlastními monitory, tento systém podporuje také zobrazování na menších zařízeních a monitorech, které jsou zabudovány v noteboocích, tabletech a v neposlední řadě potpodurje také mobilní zařízení. V dnešní době stále častějšího používání těchto zařízení je to nutnost.

Jednotlivé výkonnostní a zátěžové testy byly prováděny pomocí nástroje Overlord. Skripty spouštěné na pozadí se starají o vytvoření virtuálního stroje, nastavení a konfiguraci nástroje Overlord, sběr výsledků a jejich následné zpracování. Body zadání, kterých měla dosáhnout tato diplomová práce byly splněny.

Co se týče možného rozšíření tohoto systému do budoucnosti, tak to je samozřejmě možné. Při návrhu jsem se snažil používat moderní jazyky a technologie, které umožňují budoucí rozšíření projektu. Projekt je psán v javascriptové modifikaci zvané NodeJS za pomocí knihovny ReactJS, která je stále více a více prosazována a toto řešení je v dnešní době velmi rychlé oproti ostatním používaným jazykům v oblasti vývoje uživatelské části. Sa-

motné rozšíření by mohlo zahrnovat implementaci vice firemních produktů na výkonnostní a zátěžové testování. Rozšíření z hlediska uživatelské části je velmi jednoduché kvůli použité structure databáze, která byla prezentována. Stejně jako rozšíření v oblasti serverové části. Jelikož je NodeJS častým nástrojem pro implementaci serverové části webových prezentací, tak jeho vývoj bude s největší pravděpodobností pokračovat i v budoucnu. Dále by se mohla webová prezentace rozšířit v podobě zahrnutí ostatních nabízených služeb v rámci produktu Xtend, na kterých ještě nebyly provedeny výkonnostní a zátěžové testy.

Literatura

[1] Forum, S.: Short Message Peer-to-Peer Protocol Specification. http://docs.nimta.com/smppv50.pdf.

[2] Ghanakota, G.: Testing frameworks. https://www.cs.colorado.edu/ kena/classes/5828/s12/presentation-materials/ghanakota

[3] IEEE: *IEEE Guide for Software Verification and Validation Plans*. IEEE Computer Society, 1993, iSBN 0-7381-0410-8.

[4] IEEE: *IEEE Standard for System and Software Verification and Validation*. IEEE Computer Society, 2005, iSBN 978-0-7381-7268-2.

[5] Meier, J.; Farre, C.; Bansode, P.; aj.: *Performance Testing Guidance for Web Applications*. Microsoft Corporation, 2007, iSBN 978-0735625709.

[6] WWW stránky: Apache JMeter. http://jmeter.apache.org/.

[7] WWW stránky: BlazeMeter. https://www.blazemeter.com/blog/open-source-load-testing-tools-which-one-should-you

[8] WWW stránky: Express. http://expressjs.com/.

[9] WWW stránky: Fundamental Specifications of Kyoto Cabinet Version 1. http://fallabs.com/kyotocabinet/spex.html.

[10] WWW stránky: GIT. https://git.wiki.kernel.org/index.php/Main_Page.

[11] WWW stránky: GIT. http://www.seleniumhq.org/.

[12] WWW stránky: Highcharts: Interactive JavaScript charts for your webpage. http://www.highcharts.com.

[13] WWW stránky: iotop(1) - Linux man page. http://linux.die.net/man/1/iotop.

[14] WWW stránky: MongoDB. https://cs.wikipedia.org/wiki/MongoDB.

[15] WWW stránky: MySQL. https://cs.wikipedia.org/wiki/MySQL.

[16] WWW stránky: The Node Package Manager. https://www.npmjs.com/.

[17] WWW stránky: Performance Testing Tools. http://performance-testing.org/content/performance-testing-tools.

[18] WWW stránky: PostgreSQL. http://www.postgresql.org/.

[19] WWW stránky: The pros and cons of public, private and hybrid clouds.
http://mashable.com/2015/04/02/public-private-hybrid-cloud/#sZhJIPCfC8q1.

[20] WWW stránky: React-highcharts.
https://github.com/kirjs/react-highcharts.

[21] WWW stránky: Redis. http://redis.io/.

[22] WWW stránky: sar(1) - Linux man page. http://linux.die.net/man/1/sar.

[23] WWW stránky: Selenium. http://www.seleniumsoftware.com/.

[24] WWW stránky: Short Message Peer-to-Peer Protocol.
https://en.wikipedia.org/wiki/Short_Message_Peer-to-Peer.

[25] WWW stránky: Software performance testing.
https://en.wikipedia.org/wiki/Software_performance_testing.

[26] WWW stránky: Software testing.
https://en.wikipedia.org/wiki/Software_testing.

[27] WWW stránky: Software testing dictionary.
http://www.tutorialspoint.com/software_testing_dictionary/.

[28] WWW stránky: Standard Performance Evaluation Corporation.
www.spec.org.

[29] WWW stránky: Static testing.
http://whatis.techtarget.com/definition/static-testing.

[30] WWW stránky: Stormpath. https://stormpath.com/.

[31] WWW stránky: top(1) - Linux man page. http://linux.die.net/man/1/top.

[32] WWW stránky: What is an Automated Software Testing?
http://www.tutorialspoint.com//software_testing_dictionary/automated_software_testin

[33] WWW stránky: What is Installation Testing?
http://www.softwaretestingclass.com/what-isinstallation-testing/.

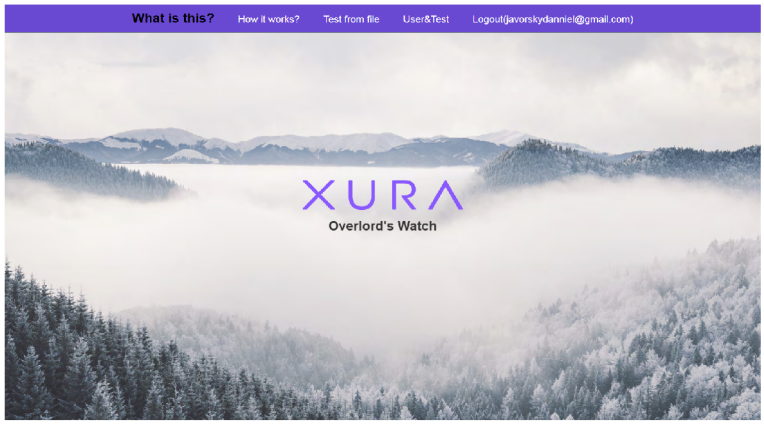
[34] WWW stránky: What is Manual Testing?
http://www.tutorialspoint.com/software_testing_dictionary/manual_testing.htm.

[35] WWW stránky: What is Virtual Dom.
http://tonyfreed.com/blog/what_is_virtual_dom.

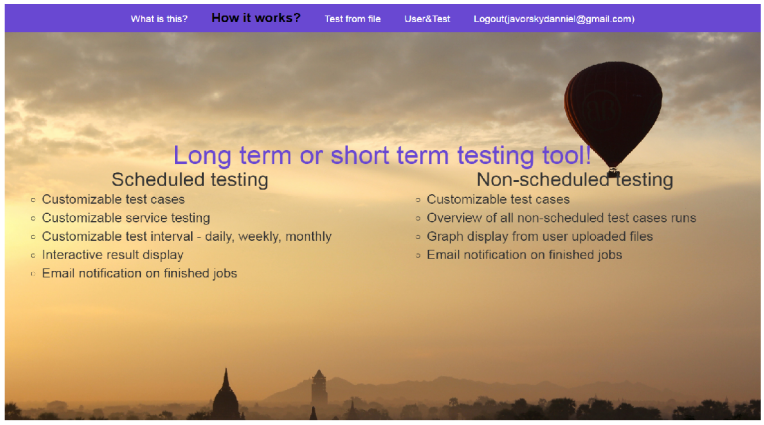
[36] WWW stránky: Xura. http://www.xura.com/.

Příloha A

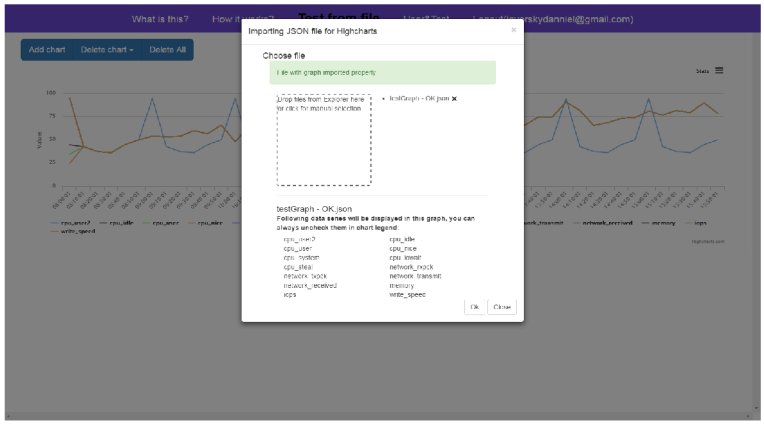
Obrázky uživatelského prostředí



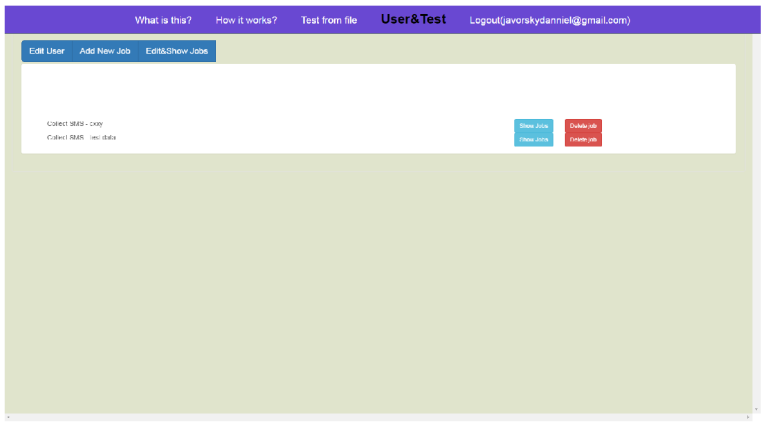
Obrázek A.1: Uvítací obrazovka po přihlášení.



Obrázek A.2: Oznámení možností nástroje.



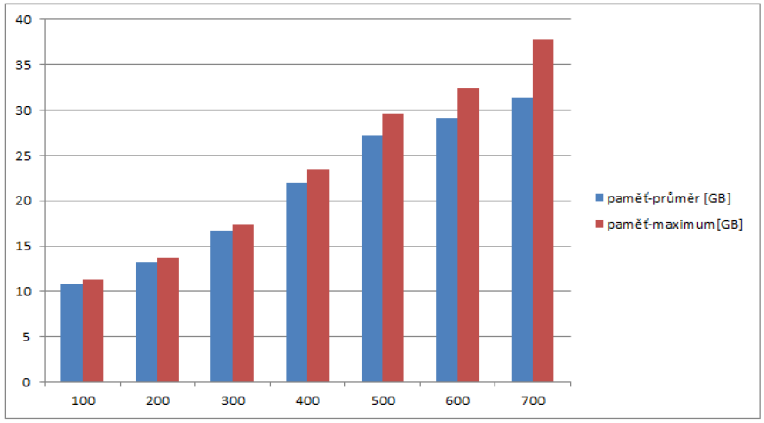
Obrázek A.3: Obrazovka s nahráním grafů ze souborů.



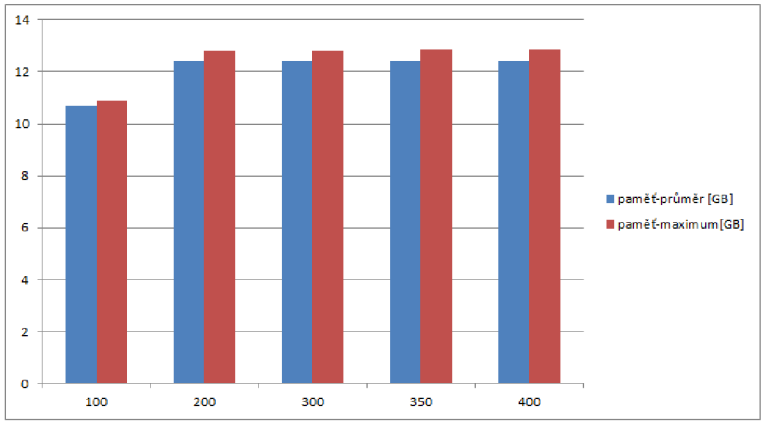
Obrázek A.4: Úprava uživatelských informací.

Příloha B

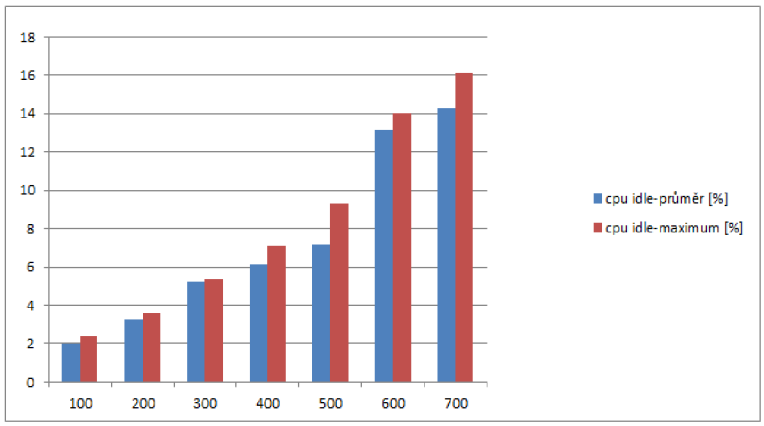
Měření aplikačního serveru Xtend



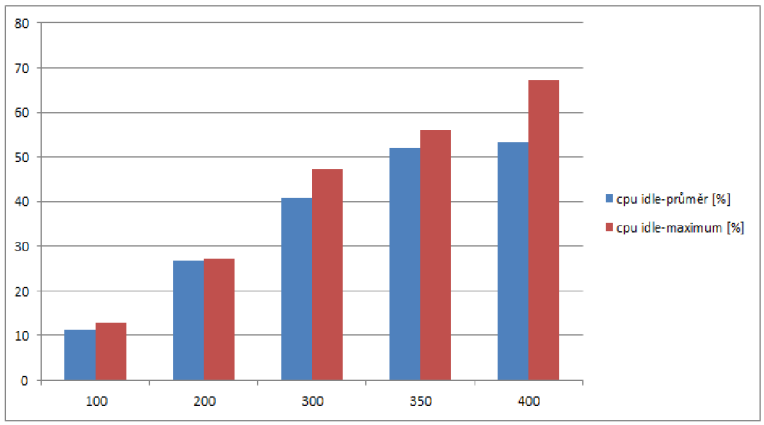
Obrázek B.1: Využití paměti pro hardware.



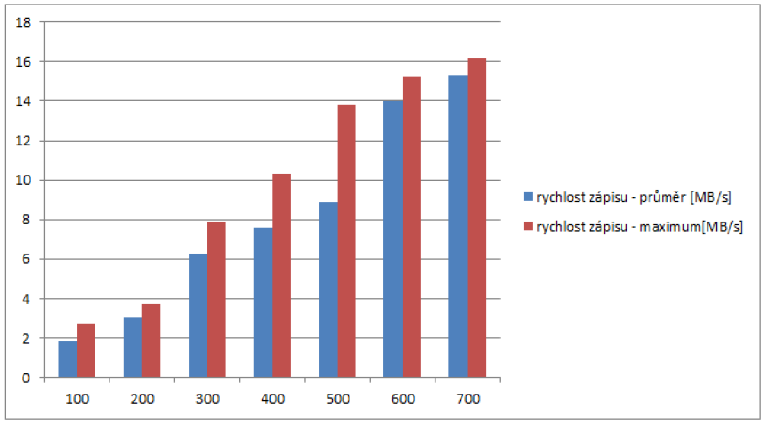
Obrázek B.2: Využití paměti pro virtuální stroj.



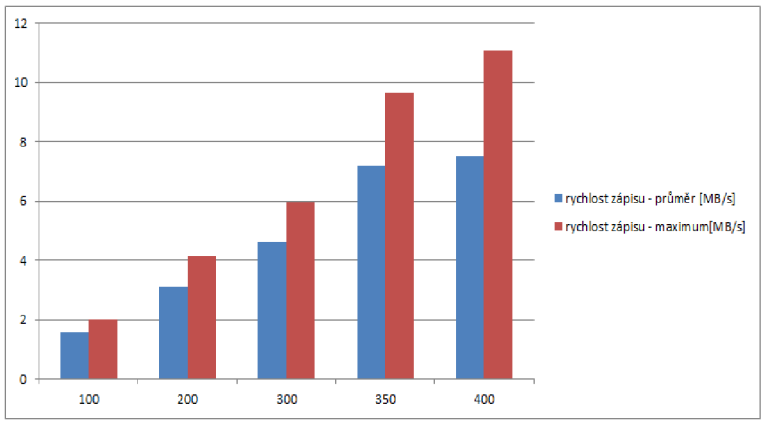
Obrázek B.3: Využití procesorů pro hardware.



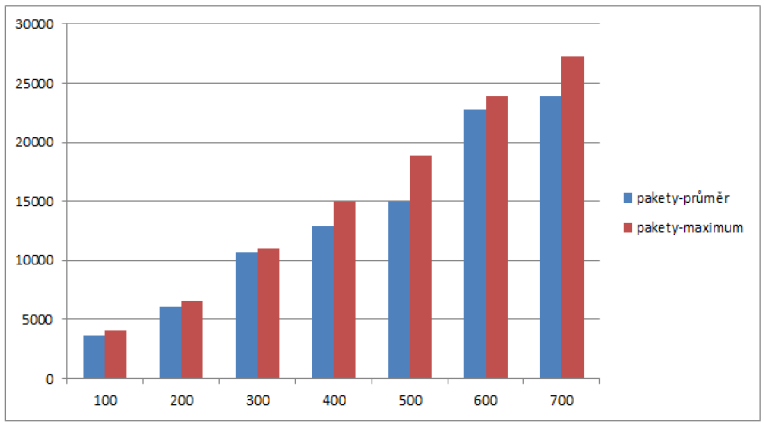
Obrázek B.4: Využití procesorů pro virtuální stroj.



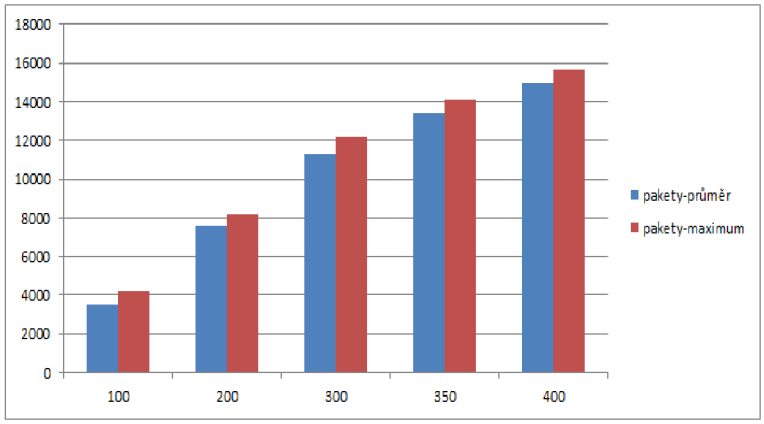
Obrázek B.5: Rychlost zápisu na disk na hardawru.



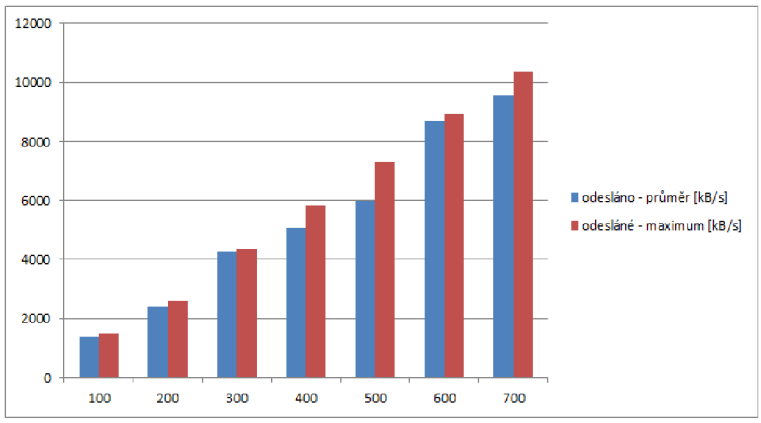
Obrázek B.6: Rychlost zápisu na disk na virtuálním stroji.



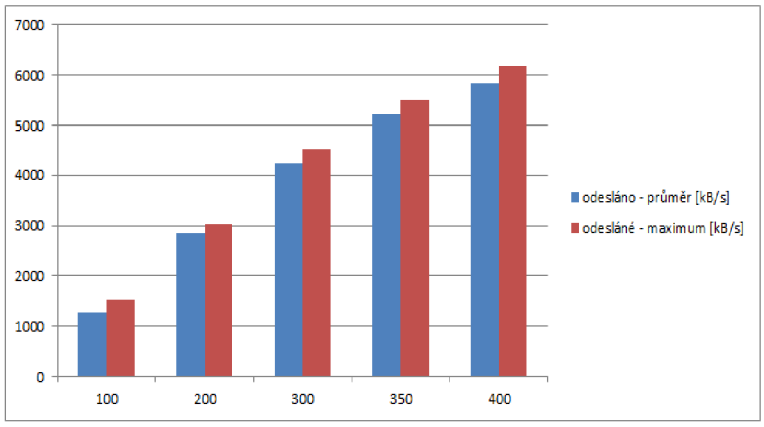
Obrázek B.7: Počet odeslaných paketů na hardwaru.



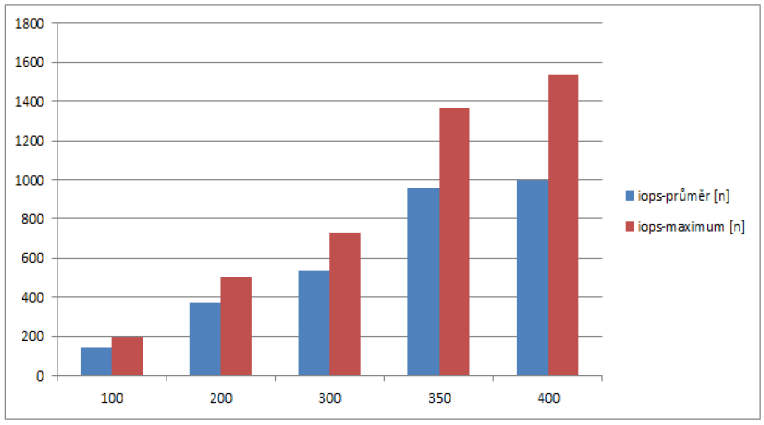
Obrázek B.8: Počet odeslaných paketů na virtuálním stroji.



Obrázek B.9: Počet odeslaných kilobajtů na hardwaru.



Obrázek B.10: Počet odeslaných kilobajtů na virtuálním stroji.



Obrázek B.11: Počet diskových operací na virtuálním stroji.

Příloha C

Obsah CD

V následující tabulce C.1 je uveden obsah přiloženého CD.

dp-xjavor14.pdf	Elektronická verze práce ve formátu PDF.
readme.txt	Obsah CD.
latex/	Zdrojový text technické zprávy.
src/	Zdrojové kódy systému na výkonnostní a zátěžové testování.

Tabulka C.1: Obsah přiloženého CD.

Příloha D

Seznam zkratek

AAGP = Acision Applictaion Gateway Provisioning
CMDA = Code Division Multiple Access
CPU = central processing unit
DS = Directory Service
EMS = Enhanced Messaging Service
ESME = External Short Messaging Entities
GSM = Groupe Spécial Mobile
IDE = Integrated Development Environment
IP = Internet Protocol
ISO/OSI = International Organization for Standardization/Open Systems Interconnection
I/O = input/output
JSON = JavaScript Object Notation
LDAP = Lightweight Directory Access Protocol
MC = Message Centre
MCO = Message Controller
MMS = Multimedia Messaging Service
MTBF = Mean Time Between Failure
MTTF = Mean Time To Failure
PDU = Protocol Data Unit
PSN = Personal Short Number
RAM = random access memory
RE = Routing Entities
SCP = Service Control Point
SLA = Service Level Agreement
SMPP = Short Message Peer-to-Peer Protocol
SMS = Short Message Service
SMSC = Short Message Service Center
SSH = Secure Shell
SS7 = Signalling System No.7
TCP = Transmission Control Protocol
TLV = Type-Length-Value
XML = Extensible Markup Language