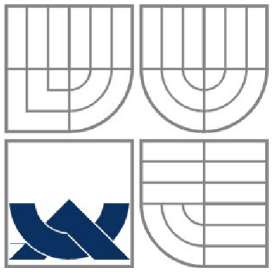
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

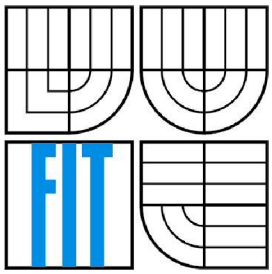
Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2016 Lukáš Novák



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SÍŤOVÁ KOMUNIKACE PRO VIRTUÁLNÍ ČEKÁRNU

NETWORK COMMUNICATIONS FOR VIRTUAL WAITING ROOM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ NOVÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN BREJCHA

BRNO 2016

Abstrakt

Jádrem této bakalářské práce je podrobně popsat návrh komunikačního protokolu pro komunikaci klient/server a vytvoření fungujícího prototypu serverové části. Pro tuto komunikaci bylo využito návrhového přístupu REST, který využívá zejména funkci GET a POST transportního protokolu HTTP. Prototyp serverové části aplikace je vytvořen v jazyce PHP za pomoci micro-frameworku Silex. Protokol je navržen jak pro mobilní klienty, ze kterých je prováděno zařazování do front, tak pro klienty, ze kterých se tato fronta řídí a spravuje. Na konci práce je popsáno testování navrženého protokolu s funkčním prototypem serverové části.

Abstract

The core of this thesis is to describe in detail the design of communication protocol for client/server communication and creation of a functioning server application prototype. REST design approach was used for this communication. This approach uses mainly GET and POST functions of HTTP transport protocol. The server application prototype is created in PHP with the help of Silex micro-framework. The protocol is designed for mobile clients, which are used for registering into queues and also for administrative clients, which can control and manage the queues. Testing of the designed protocol with server application prototype is described at the end of this thesis.

Klíčová slova

Virtuální čekárna, vyvolávací systém, Silex, REST API, HTTP, PHP, komunikační protokol.

Keywords

Virtual waiting room, queue management system, Silex, REST API, HTTP, PHP, communication protocol.

Citace

NOVÁK, Lukáš. *Síťová komunikace pro virtuální čekárnu* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Brejcha.

Síťová komunikace pro virtuální čekárnu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Brejchy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Novák Lukáš
18. května 2016

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu práce Ing. Janu Brejchovi za odborné rady a cenné připomínky k vypracování této bakalářské práce.

Obsah

Obsah 1
1 Úvod 3
2 Virtuální čekárna 4
2.1 Možné přístupy k návrhu protokolů 4
2.2 Obecný pohled 6
2.3 Role, klíčová slova a účastníci 6
3 Analýza požadavků 9
3.1 Funkční požadavky 9
3.1.1 Server 9
3.1.2 Klient mobilní 9
3.1.3 Klient v čekárně 11
3.1.4 Klient v provozovně 11
3.1.5 Hlavní a přepážkový display 12
3.2 Nefunkční požadavky 13
3.2.1 Server 14
3.2.2 Klient mobilní 14
3.2.3 Klient v provozovně 14
3.2.4 Hlavní a přepážkový display 14
4 Použité technologie 15
4.1 PHP 15
4.2 Silex 15
4.3 HTTP 15
4.4 REST 16
5 Návrh komunikace 17
5.1 CompanyController 17
5.1.1 Dotaz na provozovny v okolí zadaných souřadnic 17
5.1.2 Dotaz na veškeré provozovny 18
5.1.3 Dotaz na konkrétní provozovnu 18
5.2 MyQueuesController 19
5.2.1 Dotaz na moje fronty 19
5.2.2 Dotaz na stav fronty 20
5.2.3 Načtení konfigurace fronty 21
5.2.4 Požadavek na zařazení do fronty 22
5.2.5 Požadavek na vyřazení z fronty 22

5.2.6 Předání stavu zákazníka o příchodu do čekárny 23
5.3 WroomController 23
5.3.1 Načtení konfigurace Klienta v čekárně 23
5.3.2 Požadavek na zařazení zákazníka do fronty 24
5.3.3 Předání stavu Klienta v čekárně na server 25
5.4 CroomController 26
5.4.1 Připojení přepážky k serveru 26
5.4.2 Odpojení přepážky od serveru 26
5.4.3 Přihlášení obsluhujícího na přepážku 26
5.4.4 Odhlášení obsluhujícího z přepážky 27
5.4.5 Načtení konfigurace přepážky 27
5.4.6 Předání stavu přepážky na server 27
5.4.7 Přenesení fronty zákazníků na přepážku 28
5.4.8 Vyvolání zákazníka z konkrétní fronty 29
5.4.9 Vyvolání zákazníka mimo frontu 30
5.4.10 Opakované vyvolání konkrétního zákazníka 30
5.4.11 Ukončení obsluhy zákazníka 30
5.4.12 Přeložení obsluhovaného zákazníka na jinou přepážku 31
5.4.13 Přeložení obsluhovaného zákazníka do konkrétní fronty 31
5.4.14 Přeložení obsluhovaného zákazníka k obsluhujícímu 31
5.5 Chyba při vykonání některého z dotazů 32
6 Testování 33
Webový prohlížeč 33
Advanced REST client 33
7 Závěr 34
Možnost rozšíření práce 34
Možná optimalizace a vylepšení 34
Literatura 35
Seznam příloh 37

1 Úvod

V dnešní době jen zřídka potkáme člověka, který by nevlastnil mobilní telefon. Jako společnost jsme si zvykli, že přes mobilní telefon jsme schopni provádět bezhotovostní transakce, kupovat jízdenky na vlak či MHD a organizovat si celý svůj den. Ale pokud bychom chtěli zajít k holiči, nechat opravit automobil nebo se jen objednat na kontrolní prohlídku k lékaři, musíme obvykle stále zatelefonovat v úřední hodiny a domluvit se se sestřičkou nebo sekretářkou, která má většinou mnohem důležitější věci na práci. V tom horším případě se musíme dostavit osobně a vystát si klidně i frontu.

Nabízí se proto otázka, proč si neušetřit čas, který bezúčelně trávíme při čekání v čekárnách nebo frontách. Nebylo by efektivnější, méně stresující a hlavně rychlejší, pokud by se všichni ve frontě předem domluvili, kdo půjde kdy na řadu a podle toho všichni postupně přišli? Samozřejmě že ano. Bohužel tento model může fungovat pouze teoreticky nebo v malém kolektivu čekajících.

Již před několika lety se začaly objevovat tzv. „Vyvolávací systémy“. Každý nový zákazník, který přijde do čekárny, dostane pořadové číslo. Může se jít klidně posadit kamkoli v prostorách čekárny nebo si úplně odskočit a má jistotu, že ho nikdo nečestně nepředběhne. Tento způsob sice umožní zákazníkovi si zpříjemnit a zefektivnit čas, který by jinak musel strávit nepohodlným čekáním a kontrolováním, jestli ho někdo nepředbíhá, ale samotný čas čekání mu výrazně nezkrátí.

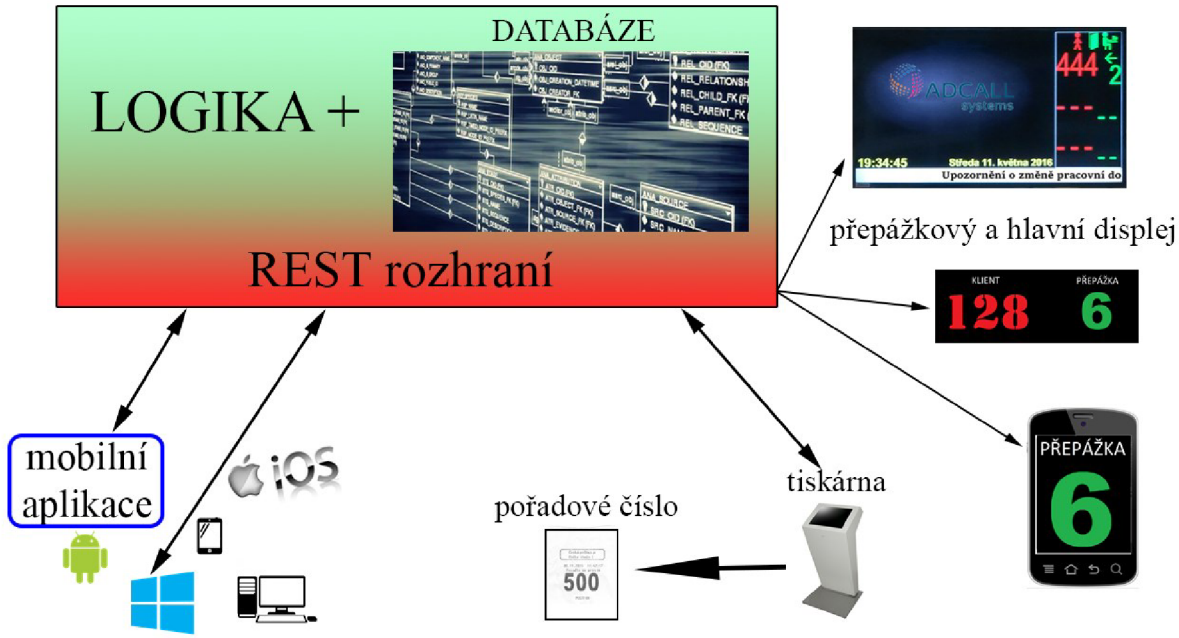
Dnešní doba by ale měla přeci umožňovat se do takové fronty zařadit ve chvíli, kdy je ještě relativně malá, aniž bychom se museli fyzicky dostavit do čekárny. Proto vznikl nápad „virtuálního“ čekání. Zákazník se může v libovolnou chvíli virtuálně zařadit do fronty a nechat za sebe čekat samotné pořadové číslo. Zákazník by mohl mít také přístup k informacím o velikosti stávající fronty, jak dlouho trvá průměrná obsluha jednoho zákazníka nebo odhad, kdy by mohl přijít na řadu on. Proto tedy „**Virtuální čekárna**“, která rozšiřuje klasický Vyvolávací systém mj. o objednání zákazníka dopředu na konkrétní termín a o možnost se zařadit do libovolné fronty kdykoliv a odkudkoliv.

Tato práce se zabývá návrhem protokolu komunikace mezi jednotlivými částmi systému, zejména mezi mobilními aplikacemi a serverem. Jako návrhový přístup pro výsledný protokol byl zvolen REST.

2 Virtuální čekárna

Nabízí možnost, aby se klient objednal nebo zařadil do konkrétní fronty na volný termín již v předstihu z pohodlí domova, nebo odkudkoliv pomocí aplikace v telefonu, tabletu či PC.

Virtuální čekárna může fungovat i jako samostatný vyvolávací systém, kdy klade minimální hardwarové požadavky na provozovatele s mnohem menší počáteční investicí než za klasické vyvolávací systémy. Obsluha provozovny si může vystačit s chytrým telefonem, na kterém běží aplikace, pomocí které si vyvolává zákazníky. Samotné vyvolání zákazníků je zajištěno pomocí notifikační zprávy pro klientovu aplikaci nebo informační obrazovku umístěnou v čekárně provozovny.



Obrázek 1:Virtuální čekárna

2.1 Možné přístupy k návrhu protokolů

V dnešní době je klíčové správně integrovat různé druhy aplikací, které běží na různých platformách, systémech a v jazycích. Existují různé přístupy pro návrh integrace poskytování webových služeb, z nich nejznámější jsou popsány níže.

CORBA

Prostředí CORBA (Common Object Request Broker Architecture) tvoří architekturu pro tvorbu distribuovaných, objektově orientovaných aplikací. Vývoj standardu CORBA sahá k počátku 90. let [1]. Jedná se spíše o historické řešení, ale díky své robustnosti a výkonosti má stále své místo ve specializovaných projektech (zejména při server/server komunikaci).

- Vhodná pro poskytování služeb v rozdílných jazycích a pro různé platformy.
- Vhodná pro rozsáhlé organizace, ve kterých mezi sebou komunikují rozdílné systémy.
- Poskytuje prostor pro vysoký výkon v reálném čase na úkor jednoduchosti. [2]

XML-RPC

Protokol, který používá XML pro přenos dotazů a metodu POST protokolu HTTP jako transportní mechanismus. RPC je iniciován klientem, který posílá zprávy s požadavkem pro vzdálený systém (server), k provedení určité metody pomocí předaných argumentů. XML-RPC je tedy na rozdíl od protokolů navržených podle REST specifikace určen pro přímé volání metod, kdežto REST přenáší reprezentaci zdrojů.

- Jednoduchý na vývoj a používání.
- Umožňuje použití pouze jedné serializační metody.

SOAP

Jedná se o modifikovanou verzi XML-RPC, která je mnohem komplexnější (silnější). Metoda SOAP[1] je založena na WSDL[2] (Web Services Description Language) a UDDI[3] (Universal Description Discovery and Integration). Jedná se o univerzální řešení, které lze použít nad různými transportními protokoly jako je například SMTP nebo HTTP [3]. Ve srovnání s REST má nespočetné výhody týkající se komplexnosti celého protokolu, které jsou ale v některých případech zbytečné a zatěžující:

- Podporuje větší možnosti zabezpečení.
- Dokáže implementovat ACID transakce (REST je limitován HTTP protokolem).
- V protokolu má implementované standardizované zprávy pro klientovo vyrovnání se s chybami v komunikaci.

„*SOAP je protokol pro výměnu informací v decentralizovaném a distribuovaném prostředí. Jedná se o protokol založený na XML, který se skládá ze třech částí: obálka, která definuje rámec pro popis toho, co je ve zprávě a jak ji zpracovat, sadu pravidel pro vyjádření instancí definovaných aplikací a konvence pro volání vzdálených procedur a jejich odezvy.*“ cit. [3]

REST

REST není protokol, ale architektonický přístup. Může být použit přenos informací v XML nebo JSON formátu. Mezi hlavní výhody webových služeb navržených pomocí REST oproti metodě SOAP patří:

- Společně s daty se nepřenáší mnoho XML dat navíc.
- Výsledky jsou čitelné pro člověka.
- Žádné nástroje pro vývoj nejsou potřebné.

Pro potřeby komunikace Virtuální čekárny byl zvolen méně komplexní, za to jednoduší REST přístup. Požadavky na dále popisovaný protokol pro Virtuální čekárnu nejsou na velkou bezpečnost nebo snad transakční zpracování jako by to například bylo požadováno u protokolu pro aplikaci na bankovní platby, pro kterou by bylo SOAP řešení vhodnější. Podrobnější popis REST přístupu naleznete v kapitole *4 Použité technologie*.

[1] https://www.w3.org/TR/soap/
[2] https://www.w3.org/TR/wsdl/
[3] http://searchsoa.techtarget.com/definition/UDDI/

2.2 Obecný pohled

Následující kapitola dle zadání přehledně definuje části navrhovaného systému Virtuální čekárny. Požadavky na výsledný protokol a způsob komunikace vycházejí ze vztahů mezi jednotlivými částmi systému. Z obecného pohledu se celý projekt dá rozdělit na pět hlavních částí:

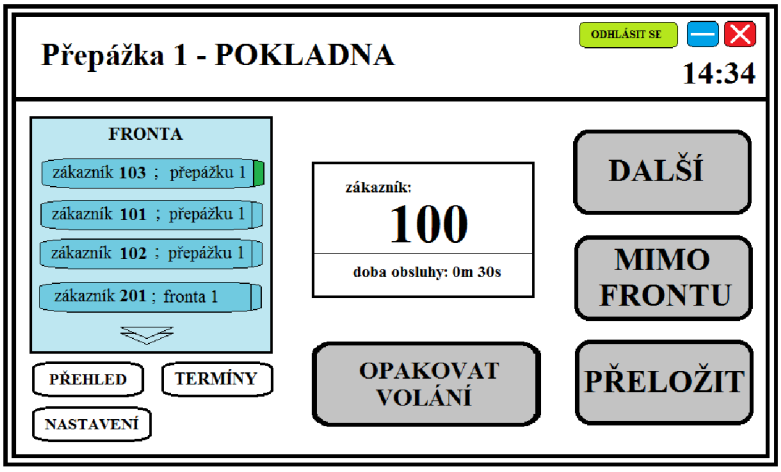
- **Serverová část:**
  - Serverová aplikace zajišťující logiku.
  - Databáze, ve které jsou uložené kompletní informace o zákaznících, provozovnách, frontách a všech potřebných nastaveních.
- **Komunikační protokol:**
  - HTTP bezstavová komunikace [4].
- **Aplikace pro zákazníka:**
  - Klient mobilní, který je spuštěn na zákazníkově mobilním telefonu, tabletu, PC či jiném podporovaném zařízením.
    - Obvykle bezdrátové připojení k internetu.
- **Aplikace pro obsluhujícího:**
  - Klient v provozovně, který je spuštěn na stolním PC, tabletu, …
    - Obvykle drátové připojení k intranetu.
- **Informační prvky a HW zařízení v čekárně:**
  - Klient v čekárně, který je spuštěn na vhodném terminálu, může být osazen tiskárnou na pořadová čísla.
  - Informační klient v čekárně, který je spuštěn na mini počítači. Ten je připojen k obrazovce připevněné na viditelném místě v prostorách provozovny.
  - Hlavní a přepážkový display s integrovaným zvukovým upozorněním.

2.3 Role, klíčová slova a účastníci

Zde definujeme klíčové pojmy, které jsou použity v následujícím textu. Jsou zde popsány i role jednotlivých uživatelů, kteří přicházejí do styku se systémem Virtuální čekárny. Ke správnému porozumění následujícího textu definuji následující pojmy:

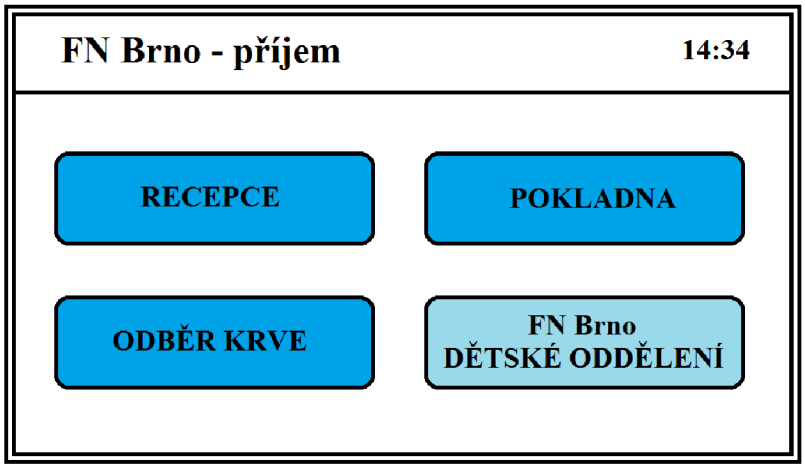
Zákazník: například pacient, který chce navštívit lékaře.
Obsluhující: proškolený personál, například lékař nebo úřednice na poště.
Provozovna: například pošta nebo oddělení v nemocnici.
Činnost: služba, kterou poskytuje konkrétní přepážka. Podle ní si vybíráme, ke které přepážce, frontě nebo obsluhujícímu chceme. Například odběr krve nebo podání balíků na poště.
Přepážka: například ordinace praktického lékaře nebo jedna z mnoha přepážek na poště. Přepážky si mezi sebou mohou vypomáhat nebo přeřazovat klienty. Každý obsluhující může obsluhovat 0 až N přepážek. Přepážka může poskytovat 0 až M činností.
Fronta: vede k 1 až N přepážkám. Každá fronta se dělí na **normální** a **prioritní** část. Normální část fronty si můžeme představit jako FIFO frontu uživatelů řazených podle času příchodu do provozovny. Prioritní fronta se využívá k akutním případům nebo prioritním činnostem. Tu také použijeme pro objednané zákazníky, u kterých bude poznámka, na jaký čas jsou objednaní. Dle toho se také budou řadit v prioritní frontě.

SW: softwarový klient v podobě aplikace nebo programu. Umožňuje počítači provádět specifické úkony prostřednictvím fyzických komponentů (hardware). Pro správné fungování je nutné spustit SW na podporovaném fyzickém zařízení, na kterém běží vhodné prostředí (operační systém).
HW: fyzický komponent (hardware), který díky nahranému jednoúčelovému firmwaru vykonává specifické úkony.
Klient v provozovně: (SW), používá ho obsluhující na přepážce. Ten má PC nebo tablet, na kterém je spuštěný Klient v provozovně, který je vázaný na konkrétní přepážku. Může se k němu přihlásit libovolný obsluhující pomocí přiděleného hesla. Vyvolává si klienty z fronty dle pořadí nebo i mimo ni. Vytváří, přeřazuje a ruší uživatele z front provozovny. Objednává a přeobjednává registrované uživatele na volné termíny.



Obrázek 2: Návrh Klienta v provozovně

Klient mobilní: (SW), používá ho zákazník. Může to být mobilní aplikace nebo i webová stránka pro zákazníky s klasickým PC. Zákazník přes něho vyhledává provozovny a jejich fronty, do kterých se může registrovat na volné termíny.
Klient v čekárně: (SW), používá ho zákazník po příchodu do čekárny. Využívá se k ohlášení, že se zákazník dostavil do čekárny. Dle zvolené provozovny a činnosti (fronty) vydá zákazníkovi pořadové číslo, podle kterého ho může obsluhující vyvolat, viz obrázek 3. Klient v čekárně je spuštěn na fyzickém zařízení (dotykový tablet či zabudovaný mini PC s dotykovou obrazovkou) na vhodném místě v provozovně, aby se k němu zákazník lehce dostal.



Obrázek 3: Návrh Klienta v čekárně – úvodní obrazovka např. pro FN Brno

Tiskárna: rozšíření Klienta v čekárně o tiskárnu lístků pořadových čísel, čtečku QR kódů či jiných HW periferií na snímání informací z různých karet a průkazů.

Informační klient v čekárně: (SW), zákazníkovi jsou prostřednictvím něho zobrazovány aktuální informace o provozovně, především o vyvolaných pořadových číslech. Informační klient v čekárně je spuštěn na fyzickém zařízení (dotykový tablet, smartTV či mini PC připojený k obrazovce viz obrázek 5). Může také zobrazovat na displeji jiná komerční sdělení nebo informace.

Hlavní display: jednou z možností, jak informovat zákazníka o vyvolaných pořadových číslech přepážkami, je hardwarový (HW) hlavní display, viz obrázek 4. Je osazen LED diodami, které zobrazují číslo vyvolaného zákazníka, číslo přepážky, která ho vyvolala a volitelně i směr k dané přepážce pomocí šipky. Dále se využívá kombinace hardware a SW řešení v podobě fyzického zařízení (např. mini PC připojený k obrazovce viz obrázek 5), na kterém je spuštěn Informační klient v čekárně. Na obrazovce se může navíc oproti hardwarovému řešení objevit mediální obsah s doplňujícími informacemi jako je například změna pracovní doby (obrázek 5).

Obrázek 4: Hlavní display firmy Kadlec - elektronika, s.r.o.

Obrázek 5: Hlavní display s mediálním obsahem firmy Kadlec - elektronika, s.r.o.

Přepážkový display: HW řešení (dále jako „PD“). Jednořádkový display umístěný v blízkosti přepážky (před dveřmi, nad okénkem s obsluhou) pomocí LED diod zobrazí číslo vyvolaného zákazníka, viz obrázek 6. Je možné ho nahradit Informačním klientem v čekárně s menší obrazovkou, jelikož PD zobrazuje především pouze pořadové číslo vyvolaného zákazníka ke konkrétní přepážce.

Obrázek 6: Přepážkový display firmy Kadlec - elektronika, s.r.o.

3 Analýza požadavků

Následující kapitola bude zaměřena na analýzu práce a specifikaci požadavků, které byly uvedeny na začátku. Analýza požadavků je rozděluje do dvou hlavních částí - funkční a nefunkční požadavky. Funkční požadavky jsou specifikovány v Use-case diagramech [5], které modelují případy užití. Jeden případ užití je chápán jako funkce, kterou systém vykonává jménem jednotlivých účastníků nebo v jejich prospěch.

Dle zadání je cílem této práce vytvořit komunikační protokol pro sjednocení klient/server komunikace s efektivním přenosem dat, který bude jednoduše rozšiřitelný do budoucna. Komunikace má být bezstavová, tj. každý požadavek je zpracováván jako samostatná transakce, která nemá žádný vztah k předchozí nebo k budoucí komunikaci. Díky tomu server nemusí udržovat spojení otevřené po odeslání odpovědi, nebo si uchovávat dočasné informace o předchozích komunikacích s klienty. Jsou kladeny menší požadavky na paměť a výpočetní výkon serveru, než při využití ukládání dočasných souborů. Díky omezení se pouze na dotaz a odpověď bez nutnosti udržování otevřeného spojení po delší dobu se také sníží nároky na šířku pásma připojení. Tímto dosáhneme lepší škálovatelnosti systému, než při udržování otevřeného spojení se serverem, při kterém může dojít k chybám nebo výpadkům spojení. „*Škálovatelnost je schopnost softwarového systému přizpůsobit se novým požadavkům na velikost a rozsah*.“ cit. [6].

3.1 Funkční požadavky

„*Funkční požadavky popisují softwarovou funkcionalitu, kterou musí vývojáři do systému dostat, aby uživatelé mohli splnit své úkoly, a tím i podnikatelské požadavky. Funkčním požadavkům se někdy též říká behaviorální požadavky, neboli požadavky na chování.*“ cit. [7]

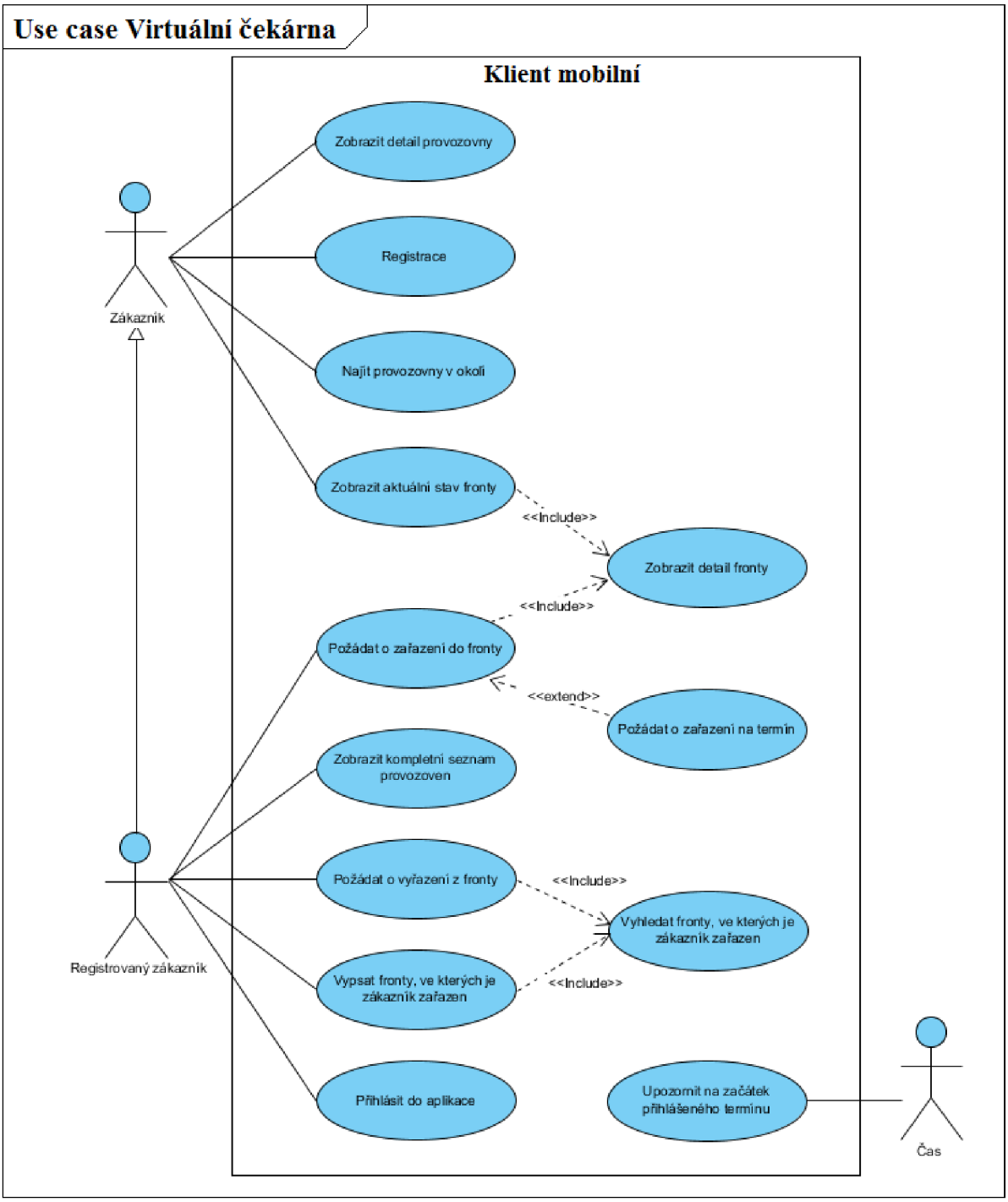
3.1.1 Server

- Kolize v termínech se řeší na straně serveru společně s vyhodnocením dotazů od klienta a zabezpečením.
- Server musí být schopen komunikovat bez ohledu na operační systém klienta.
- Server musí mít možnost nastavení automatických záloh a exportu záznamů o chybových hlášeních jak lokálně tak vzdáleně.
- Server musí být schopen generovat pro různé fronty pořadová čísla v předem zadaném rozsahu.

3.1.2 Klient mobilní

Případy užití mezi zákazníkem a Klientem mobilním (dále jako „aplikace“) jsou znázorněny na Use-case diagramu (obrázek 7). Tento diagram zobrazuje požadavky, které musí aplikace splňovat.

- Registrovat nového zákazníka a umožnit přihlášení již registrovaného zákazníka. Správu osobních údajů a možnost jejich aktualizace.
- Dotázat se na aktuální stav fronty, zažádat o zařazení či vyřazení ze zvoleného termínu.
- Musí podporovat jistou formu notifikací (například upozornění na jeho vyvolání k přepážce, nebo změnu přihlášeného termínu).

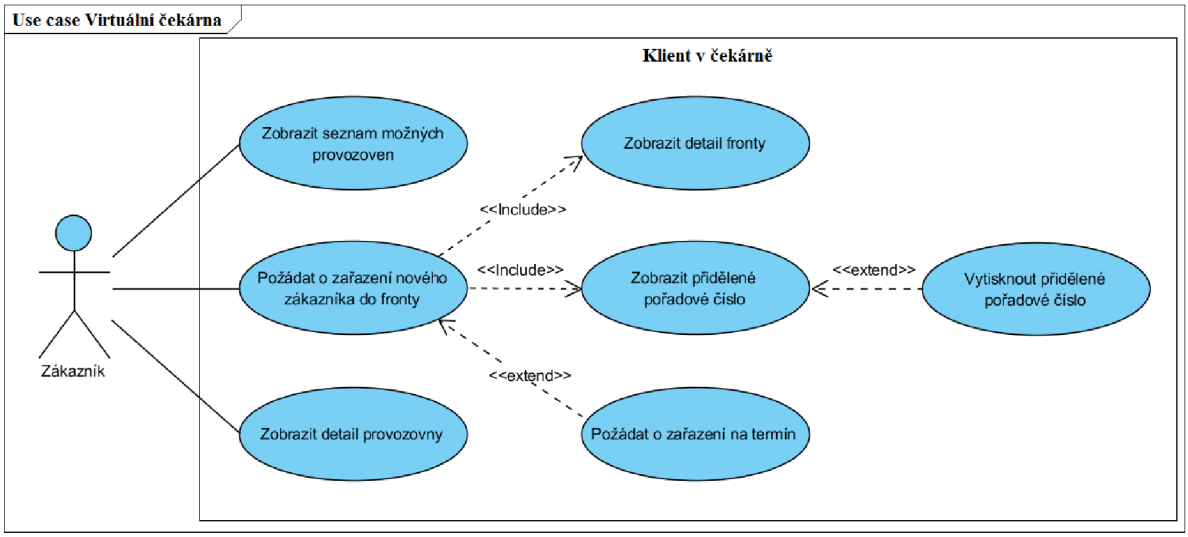


Obrázek 7: Use-case diagram pro Klient mobilní

3.1.3 Klient v čekárně

Případy užití mezi zákazníkem a Klientem v čekárně jsou znázorněny na Use-case diagramu (obrázek 8). Tento diagram zobrazuje požadavky, které musí aplikace splňovat.

- Klient v čekárně musí pravidelně posílat údaje o svém stavu. V kombinaci s tiskárnou například varování o nedostatku papíru nebo zaseklém papíru v tiskové hlavě.
- Na vyžádání serveru (popsané v odpovědi na požadavek z bodu: *5.4.6 Předání stavu přepážky na server*) si načíst novou konfiguraci o frontách, přepážkách a grafickém rozhraní.
- Vytvořit a požádat o zařazení nově příchozího zákazníka do fronty (jak prioritní tak normální), na určitou přepážku nebo přímo ke konkrétnímu obsluhujícímu.
- Zobrazit zákazníkovi jeho pořadové číslo nebo ve spojení s tiskárnou mu toto číslo vytisknout.



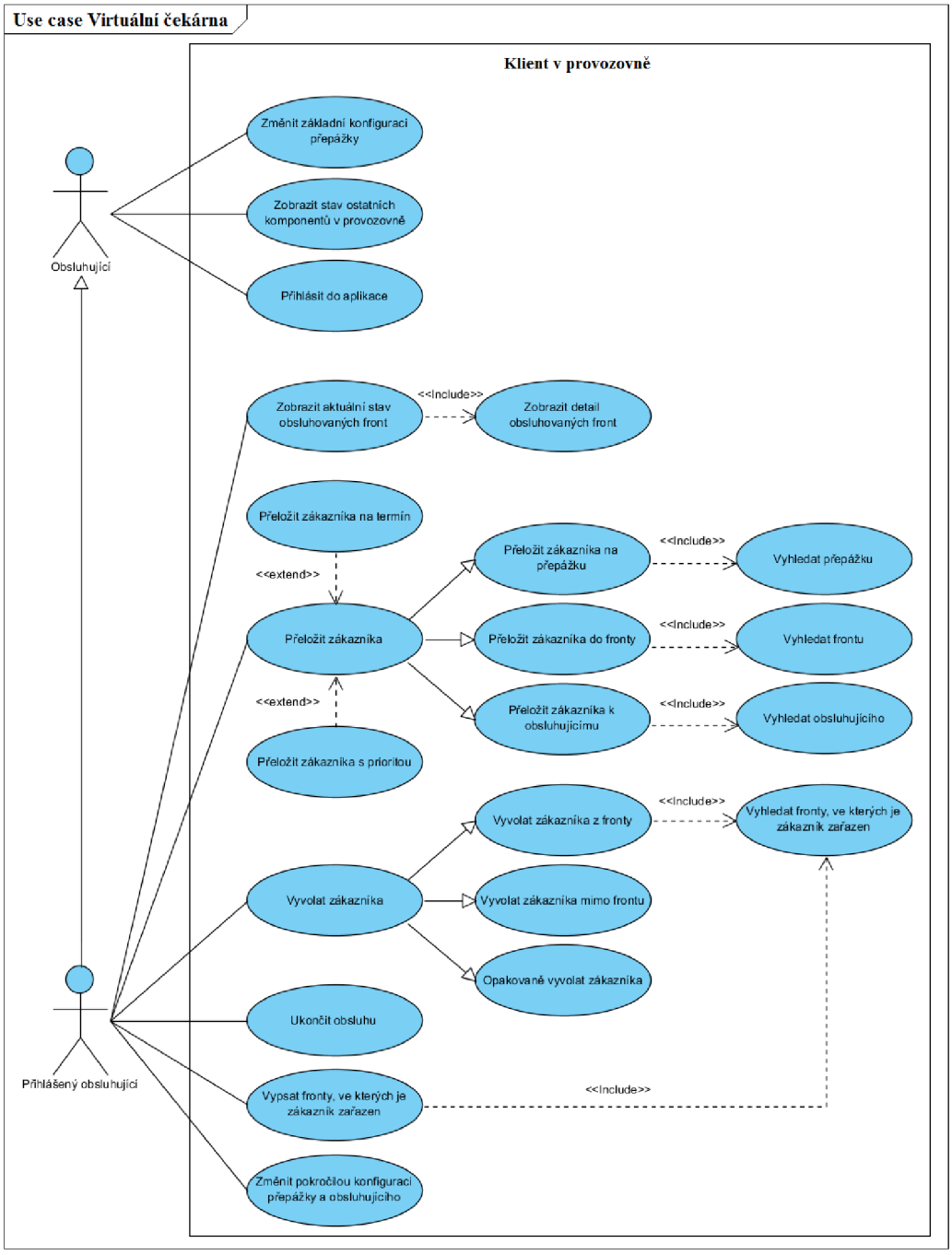
Obrázek 8: Use-case diagram pro Klienta v čekárně

3.1.4 Klient v provozovně

Případy užití mezi obsluhujícím a Klientem v provozovně (dále i jako „přepážka“) jsou znázorněny na Use-case diagramu (obrázek 9). Tento diagram znázorňuje požadavky, které musí přepážka splňovat.

- Přepážka musí být schopna se připojit na server se svým interním identifikátorem a poté umožnit obsluhujícímu jeho přihlášení na server. Na jedné přepážce může být současně přihlášen pouze jeden obsluhující.
- Rozlišení jednotlivých obsluhujících provádí přepážka pouze pomocí unikátního vstupního kódu, který byl přidělen obsluhujícím při konfiguraci celého systému.
- Načíst si svoji aktuální konfiguraci jak v pravidelných intervalech, tak i na vyžádání serveru (předané v odpovědi požadavku viz *5.4.6 Předání stavu přepážky na server*).
- Předávat pravidelně svůj stav serveru a přijímat poté stavy ostatních přepážek společně s možnými požadavky serveru. Např. aby si přepážka aktualizovala frontu, konfiguraci, nebo aby se odpojila.
- Načíst fronty zákazníků a to buď pouze z front, které obsluhuje, nebo kompletní frontu provozovny.
- Požádat o zařazení nového zákazníka do fronty, vyvolat zákazníka z fronty na přepážku, vyvolat zákazníka mimo frontu na přepážku, opakovaně vyvolat zákazníka, pokud na výzvu nereaguje.

- Ukončit obsluhu zákazníka, nebo ho přeložit na jinou přepážku, frontu nebo k jinému obsluhujícímu. Přeložit zákazníka musí být možné na konec fronty, na konec fronty s prioritou, na konkrétní termín.



Obrázek 9: Use-case diagram pro Klienta v provozovně

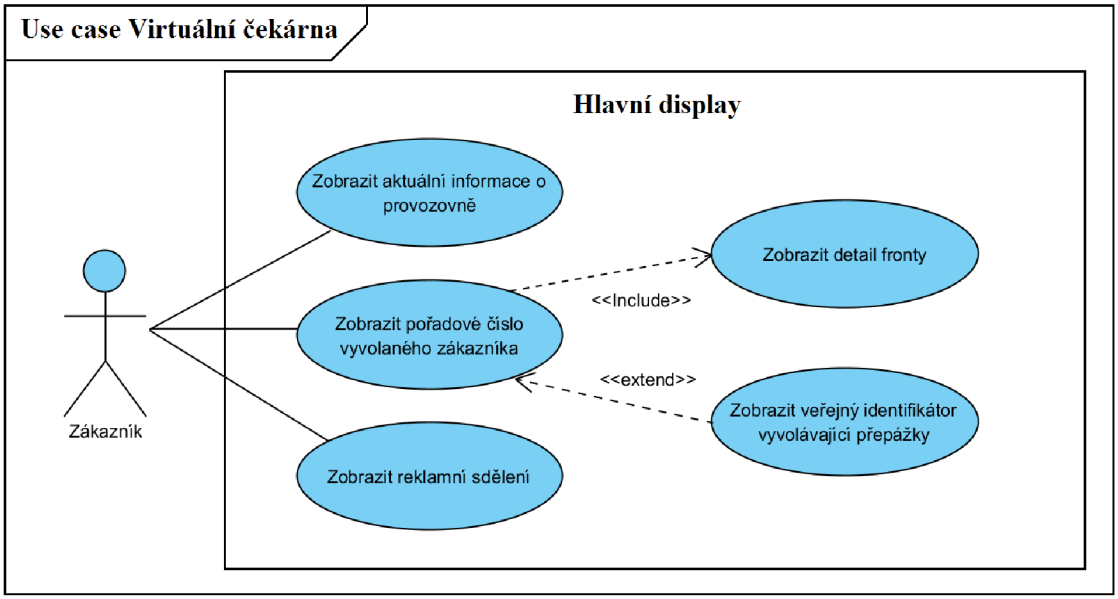
3.1.5 Hlavní a přepážkový display

Budou zde popsány požadavky především pro hlavní a přepážkový display v provedení SW, který je spuštěn na fyzickém zařízení. Informace se zobrazují na obrazovce, která je připojena k zařízení, na kterém běží Informační klient v čekárně (SW).

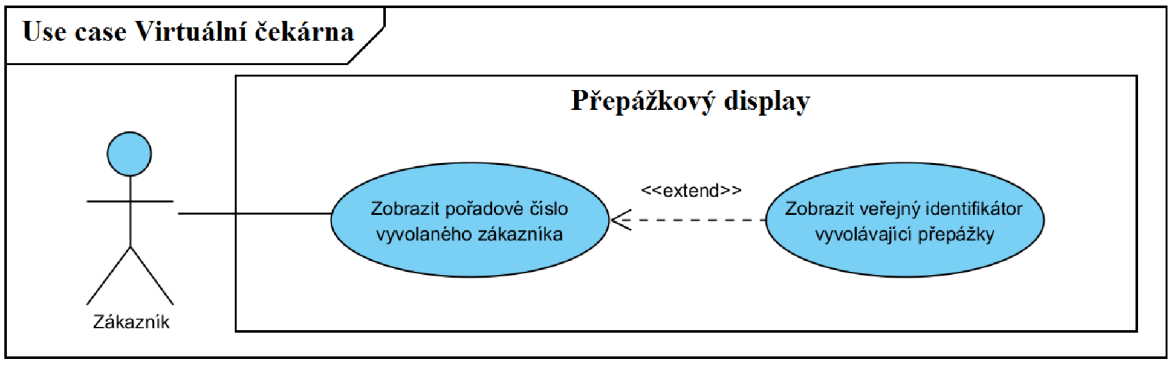
Při čekání zákazníka na vyvolání v čekárně provozovny dochází k interakci s informačními obrazovkami v provozovně. Tyto případy jsou znázorněny na Use-case diagramu (obrázek 10) pro

hlavní display a (obrázek 11) pro přepážkový display. Požadavky na jednotlivá zařízení se liší podle provedení (HW nebo SW) a účelu použití v provozovně.

- Zobrazovat informace o právě vyvolaném zákazníkovi a přepážce (Klientovi v provozovně), která zákazníka vyvolala. Například pořadové číslo vyvolaného zákazní 444 a číslo přepážky 2, ke které se má zákazník dostavit (obrázek 5).
- Podle konfigurace si ukládat omezenou historii vyvolaných zákazníků a zobrazovat ji na obrazovce (obrázek 4).
- Možnost zvukového upozornění na změnu informací na displeji.
- Zobrazovat aktuální informace o provozovně a jiná komerční a informační sdělení (obrázek 5).



Obrázek 10: Use-case diagram pro Hlavní display



Obrázek 11: Use-case diagram pro Přepážkový display

3.2 Nefunkční požadavky

Nefunkční požadavky jsou ty, které nejsou uvedeny v Use-case diagramech. Spíše než definovat, co aplikace dělá, se zaměřují na to, jak má aplikace poskytovat požadované funkce [8]. Popisují další vlastnosti systému, které uživatel nemusí přímo vidět při práci s aplikací.

3.2.1 Server

Server musí být schopen:

- obsluhy více klientů najednou
- běžet jak v uzavřené síti, tak i jako distribuovaná služba. „*Distribuovaný systém je takový, jehož komponenty (části) jsou umístěny v síti počítačů, které mezi sebou komunikují a koordinují svoje akce pouze pomocí zpráv, které si mezi sebou předávají. Tato definice vede k následujícím význačným vlastnostem distribuovaného systému: nepřítomnost centrálního časového mechanismu (hodin), absence sdílené paměti a zachování funkčnosti při výpadku jednoho nebo několika uzlů přesměrováním zátěže na ostatní funkční uzle*“ cit. [9].

3.2.2 Klient mobilní

Klient mobilní musí:

- zobrazit veškeré podstatné informace a ovládací prvky aplikace s důrazem na využití responsivního designu
- snížit frekvenci pravidelných dotazů při připojení přes mobilní internet

3.2.3 Klient v provozovně

- Responsivní design tak, aby okno spuštěné aplikace mohlo mít velikost jednoho tlačítka až po přehledné zobrazení všech prvků při velikosti okna přes celou obrazovku.

3.2.4 Hlavní a přepážkový display

- Všechny informační displeje musí být schopny komunikovat jak se serverem Virtuální čekárny, tak i s jinými servery, které poskytují chtěný mediální obsah nebo jiný druh informací.

4 Použité technologie

Charakteristika použitých knihoven, programů a protokolů pro tuto práci je stručně definována v této kapitole. Jedná se o popis zejména použitého micro-frameworku Silex společně s využitím rozhraní REST.

4.1 PHP

„*Zkratka PHP je rekurzivní akronym pro PHP: Hypertextový Preprocesor. PHP je velmi rozšířený a pro všeobecné užívání určený open source skriptovací jazyk, který je vhodný zvláště pro programování na webu a může být vnořen do HTML.*“ cit. [10]

PHP je skriptovací jazyk pracující na straně serveru, který je určen především pro programování dynamických internetových stránek.

V této práci je využit především díky své jednoduchosti na použití a pro možnost jednoduchého rozšíření systému do budoucna. Pro vývoj prototypu komunikační části serveru nebylo nutné klást velké nároky na rychlost nebo bezpečnost.

PHP se stal velmi oblíbeným především díky své jednoduchosti a použitelnosti. Podle statistiky na w3techs.com[4] ho k 15. 5. 2016 používá **82.2%** internetových stránek jako skriptovací jazyk na straně serveru.

4.2 Silex

Silex je micro-framework založený na kombinaci „Symphony“[5], „Pimple“[6] a inspirovaný programovacím jazykem „Sinatra“[7]. Micro-framework Silex poskytuje vhodné prostředky pro tvorbu jednoduchých REST-full aplikací.

- Jednoduše rozšiřitelný.
- Umožňuje vytvoření intuitivního a přehledného rozhraní API.
- Respektuje HTTP specifikaci a plně podporuje její řádné používání.
- Jednoduše testovatelný.

Stručně řečeno, lze vytvářet web s přehlednými adresami, které se dají lehce měnit a spravovat [11].

4.3 HTTP

HTTP (Hypertext Transfer Protocol) je protokol pracující na aplikační vrstvě, původně navržený k přenosu dokumentů ve formátu HTML, dnes se mu však dostává mnohem širšího využití. Klíčovou vlastností tohoto protokolu je bezstavovost. Hlavička dotazu mimo jiné obsahuje pole Content-Type, které obsahuje klientem požadovaný typ média [12]. Díky němu lze rozlišit, jestli klient žádá o odpověď v podobě XML[8] nebo JSON[9] řetězce viz kapitola *5 Návrh komunikace.* Součástí hlavičky odpovědi

[4] http://w3techs.com/technologies/details/pl-php/all/all
[5] http://symfony.com/
[6] http://pimple.sensiolabs.org/
[7] http://www.sinatrarb.com/
[8] http://tools.ietf.org/html/rfc3236
[9] http://tools.ietf.org/html/rfc4627

serveru je mimo jiné stavový kód, který upřesňuje, jak byla odpověď zpracována [4]. Po předání stavového kódu v odpovědi klient okamžitě zjistí, jestli byl jeho požadavek správně vyhodnocen, nebo v průběhu došlo k nějaké chybě, kterou přesně definuje daný stavový kód.

Protokol definuje celkem osm metod pro práci s daty objektu (dokumentu). Mezi nejdůležitější patří GET, POST, UPDATE a DELETE.

4.4 REST

REST (Representational State Transfer) - Jedná se o výraz, který představil Roy Fielding v roce 2000 ve své disertační práci [13], kde položil základy návrhového stylu REST softwaru. Podle [13] se jedná o sadu návrhových pravidel a doporučení, která omezují vztahy mezi jednotlivými prvky aplikace a jejich rolemi [14].

Rozhraní definované podle REST je použitelné pro snadný a jednotný přístup ke zdrojům aplikace. Zdrojem mohou být samotná data nebo stavy aplikace, které lze popsat konkrétně daty. REST rozhraní definuje čtyři základní metody pro přístup ke zdrojům pomocí jejich unikátních URI identifikátorů [15]. Tyto metody (Create, Retrieve, Update, Delete) jsou implementovány pomocí odpovídajících metod HTTP protokolu.

5 Návrh komunikace

V dále uvedených popisech budou pro přehlednost XML data rozepsána na více řádků a řádky budou odsazeny podle úrovně zanoření. Dvojitými lomítky jsou uvozovány komentáře, které se v samotném protokolu nepřenáší. Údaje ve složených závorkách budou nahrazeny aktuálními daty. Pokud je hodnota parametru řetězec, při odpovědi pomocí JSON řetězce, píše se do dvojitých uvozovek. Případná diakritika je kódovaná v UTF8. Server na dotazy odpovídá buď odpovědí popsanou níže, nebo chybovou zprávou s chybovým kódem uvedeným na konci této kapitoly.
Komunikační protokol byl zaměřen na komunikaci mezi serverem a výše popsanými klienty. Technicky je přenos požadavků a dotazů na server založen na dotazovací metodě GET a pro přenos dat jsou použity metody POST a UPDATE protokolu HTTP.

Díky možnosti rozlišit, jestli klient preferuje přijímat odpovědi od serveru pomocí XML nebo JSON řetězců vyplněním pole v hlavičce dotazu Content-type je použitelnost a variabilita celé komunikační vrstvy ještě navýšena.

Popis výsledného protokolu je rozdělen do následujících kapitol tak, že každá z kapitol popisuje jednotlivou část prototypové aplikace.

5.1 CompanyController

Umožňuje obsluhu především mobilních klientů při vyhledávání provozovny, kterou chce zákazník navštívit. Pro získání podrobných informací o provozovnách mohou využít následující dotazy.

Formát dotazu na server:

```
/public/index.php/company/…
```

5.1.1 Dotaz na provozovny v okolí zadaných souřadnic

Slouží pro zjištění provozoven, které používají virtuální čekárnu, ve kterých zákazník může využít jejich služby. Je to dotaz, který díky GPS souřadnicím {**GPS_coordinates**} a zadanému rádiusu {**radius**} v kilometrech dokáže zákazníkovi odpovědět seznam provozoven, které jsou v okolí zadaného bodu. Pro podrobnější informace se musí klient dotázat na tyto provozovny přímo.

Dotaz na server:

```
/public/index.php/company/{GPS_coordinates}/GPS/{radius}/
```

Odpověď serveru:

```
<response>
  <company>
    <com_id>{ID_company}</com_id>
    //unikátní identifikátor provozovny
    <uri>/public/index.php/company/id={ID_company}/</uri>
    //adresa dotazu pro získání podrobnějších informací o provozovně
    <gps>{COORDINATES}</gps>
    //GPS souřadnice
  </company>
  <company>…</company>
</response>
```

5.1.2 Dotaz na veškeré provozovny

Tento dotaz slouží pro získání seznamu všech provozoven, které poskytují virtuální čekárnu bez ohledu na jejich umístění.

Dotaz na server:

```
/public/index.php/company/
```

Odpověď serveru:

Formát odpovědi se shoduje s formátem odpovědi u předchozího dotazu 5.1.1.

5.1.3 Dotaz na konkrétní provozovnu

Slouží pro získání podrobnějších informací o provozovně {**ID_company**} (název, adresa, kontaktní údaje, poskytované služby případně její pobočky). Každá provozovna obsahuje minimálně jeden záznam o další provozovně (oddělení) `<department>` nebo frontě `<queues>`, kdy jejich počet není jinak omezen. Záznamy o přepážkách nejsou nijak limitovány. Záznamy o webové adrese `<web>`, doplňujících informacích `<info>` a přepážkách `<counters>` jsou nepovinné.

Získání podrobných informací pro jednu a více provozoven najednou je možné řetězením jejich identifikátorů ({**ID_company**}, {**ID_company**_2}, …) za sebou pomocí znaku „&“. Před každým identifikátorem je nutné uvést „id=“.

Dotaz na server:

```
/public/index.php/company/id={ID_company}&id={ID_company_2}
```

Odpověď serveru:

```
<response>
  <item key="{ID_company}">
    <uri>/public/index.php/company/id={ID_company}/</uri>
    //adresa dotazu pro tuto konkrétní provozovnu.
    <com_id>{ID_company}</com_id>  //unikátní identifikátor provozovny
    <name>{NAZEV}</name>           //jméno provozovny
    <address>{ADRESA}</address>    //adresa
    <contact>{KONTAKT}</contact>   //kontaktní informace (tel, e-mail)
    <gps>{COORDINATES}</gps>       //GPS souřadnice
//následují nepovinná data:
    <web>{WADR}</web>              //webová adresa
    <info>{DI}</info>              //doplňující informace
    <department>
      <id>{ID_department}</id>     //identifikátor podřazené provozovny
      <uri>/public/index.php/company/id={ID_department}/</uri>
      //adresa dotazu pro získání podrobnějších informací o provozovně
    </department>
    <department>…</department>     //další provozovny (oddělení)
    <queues>
      <id>{queue_ID}</id>          //identifikátor fronty
      <uri>/public/index.php/myqueues/id={queue_ID}/</uri>
      //adresa dotazu pro získání podrobnějších informací o frontě
    </queues>
    <queues>…<queues>              //další fronty
    <counters>
      <id>{num_counter}</id>            //interní pořadové číslo
```

```
      <cntr_id>{counter_ID}</cntr_id>    //identifikátor přepážky
    </counters>
    <counters>…</counters>               //další přepážky
  </item>
  <item key="{ID_company_2}">…</item>
</response>
```

5.2 MyQueuesController

Používá se pro obsluhu dotazů od Klienta mobilního a Klienta v čekárně včetně Informačního klienta v čekárně pro načtení konfigurace front. Pro dotazy od Klienta mobilního mimo dotazu na načtení konfigurace fronty a dotazu na stav fronty je nutné, aby zákazník byl registrovaný a přihlášený ke svému účtu.

Pro zjednodušení a oproštění se od konkrétního způsobu ověření zákazníka nejsou dotazy rozlišeny pro přihlášeného a nepřihlášeného zákazníka. Server proto odpoví na dotaz nepřihlášeného zákazníka tak, jako kdyby byl přihlášen a úspěšně ověřen a má oprávnění tento dotaz provést. Autorizace a autentifikace se tedy provádí na vyšších úrovních aplikace, než je samotný protokol.

Formát dotazu na server:

`/public/index.php/myqueues/…`

5.2.1 Dotaz na moje fronty

Slouží pro informaci o registrovaných termínech konkrétního zákazníka, jehož identifikátor {**customer_ID**} je shodný s identifikátorem přihlášeného uživatele. V atributu {**que_data**} se nachází počáteční a koncové datum požadovaného období ve tvaru: „since={**DATE**}&to={**DATE**}“, kdy atribut „since“ určuje počáteční datum a atribut „to“ určuje koncové datum období. Datum {**DATE**} je ve tvaru „DD/MM/RRRR“. Pin pro potvrzení příchodu do čekárny {**PPP**} slouží objednanému zákazníkovi pro potvrzení, že fyzicky přišel do čekárny a je ho tedy možné vyvolat k přepážce. Zákazník ho může přímo zadat po příchodu do Klienta v čekárně, nebo jeho Klient mobilní nabízí přímé potvrzení pomocí tlačítka u konkrétní fronty, do které je přihlášen. Tento pin je jedinečný v rámci jedné fronty a jednoho dne, na který je zákazník objednaný. Pro jednoduché zapamatování a zadání do Klienta v čekárně je pin generován z rozsahu čísel od 1000 do 9999.

Počet tagů `<myqueue>` je shodný s hodnotou {**PZF**}. Pokud je zákazník registrován na více termínů v rámci jedné fronty, je každý záznam zvlášť v tagu `<myqueue>`.

Zákazník se může nacházet v jednom ze čtyř stavů {**stavID**}:

- **‘0‘**: **není přítomen** – Zákazník je přiřazen do fronty, ale ještě nepřišel fyzicky do čekárny. Jeho stav z „0“ do „1“ se změní ve chvíli, kdy zadá svůj pin do Klienta v čekárně, nebo v aplikaci odsouhlasí, že již do čekárny přišel.
- **‘1‘**: **čeká** – Zákazník přišel fyzicky do čekárny. Potvrdil ve své aplikaci příchod nebo zadal do Klienta v čekárně přidělený pin a bylo mu vydáno pořadové číslo.
- **‘2‘**: **obsluhován** – Právě je zákazník obsluhován některou z přepážek. Přechod ze stavu „1“ do „2“ je proveden ve chvíli vyvolání zákazníka na přepážku.
- **‘3‘**: **obsloužen** – Zákazník již byl obsloužen. K přechodu ze stavu „2“ do stavu „3“ dochází po obsloužení zákazníka automaticky, když obsluhující vyvolá dalšího klienta, nebo přepážka ukončí obsluhu viz *5.4.11 Ukončení obsluhy zákazníka*

Dotaz na server:

```
/public/index.php/myqueues/getmyques/{customer_ID}/{que_data}/
```

Odpověď serveru:

```
<response>
  <ack>getmyques</ack>
  <num_queues>{PZF}</num_queues> //počet záznamů o přihlášených frontách
  <myqueue>
    <custom_id>{IDZ}</custom_id>      //jedinečný identifikátor zákazníka
    <queue_id>{IDF}</queue_id>        //identifikátor fronty
    <status>{stavID}</status>
      //stav, ve kterém se aktuálně nachází zákazník: 0;1;2;3
    <request_pin>{PPP}</request_pin>
      //pin pro potvrzení příchodu do čekárny
    <request_id>{JIDP}</request_id>
      //jedinečný identifikátor požadavku
//následují nepovinná data:
    <id_wroom>{PČ}</id_wroom>
      //pořadové číslo, pokud je přiřazeno
    <name>{JMENO}</name>
      //jméno, které se může tisknout na lístek s pořadovým číslem
    <insurance_number>{CISLO}</insurance_number>
      //skrytý identifikátor, který se nemůže tisknout na lístek s PČ
    <appointment_id>{IDT}</appointment_id>
      //ID termínu, na který je zákazník objednaný předem
  </myqueue>
  <myqueue>…</myqueue>                //další fronty uživatele
</response>
```

5.2.2 Dotaz na stav fronty

Slouží pro zjištění aktuálního stavu konkrétní fronty {**queue_ID**}. Pokud je zákazník přihlášený, podle nastavení dané fronty mu mohou být poskytnuty například podrobnější informace o dalších přihlášených zákaznících.

Význam jednotlivých dat zákazníka zůstává stejný jako u předchozího dotazu: *5.2.1 Dotaz na moje fronty*. Rozdíl je v podrobnosti vrácených informací, kdy se nemusí veškeré informace poskytovat. Zobrazení nepovinných dat nepřihlášeným zákazníkům záleží na nastavení politiky provozovny a na nastavení konkrétní fronty.

Dotaz na server:

```
/public/index.php/myqueues/getque/{queue_ID}/
```

Odpověď serveru:

```
<response>
  <ack>getmyques</ack>
  <stats>…</stats>           //aktuální statistiky a metriky fronty
  <customer>
    <queue_id>{IDF}</queue_id>
//nepovinná data:
    <id_wroom>{PČ}</id_wroom>
    <status>{stavID}</status>
    <name>{JMENO}</name>
```

```
    <appointment_id>{IDT}</appointment_id>
    <custom_id>{IDZ}</custom_id>      //pokud je zákazník registrovaný
  </customer>
  <customer>…</customer>
</response>
```

5.2.3 Načtení konfigurace fronty

Používá se pro načtení konfigurace konkrétní fronty {**queue_ID**} nebo načtení konfigurace z vybraných front, které splňují další parametry dotazu {**service**}. Konfigurace obsahuje především seznam frontou poskytovaných termínů v zadaném období. Platný dotaz musí obsahovat minimálně jeden identifikátor fronty. Na pořadí parametrů nezáleží.

Jednotlivé parametry dotazu:

- „**id**“: určuje konkrétní frontu, která zákazníka zajímá. Při zadání více identifikátorů vzniká množina front, ze kterých se dle dalších zadaných parametrů vybírají fronty, jejichž konfigurace se poté pošle zákazníkovi.
  - `/id={queue_ID}&id={queue_ID2}…`
- „**attr**“: název služby, kterou zákazník požaduje, aby výsledná fronta poskytovala.
  - `/id={queue_ID}&attr={service}&attr={service#2}…`
- „**since**“ a „**to**“: počáteční a koncové datum vyhledávání termínů front. Bez platného zadání je zákazníkovi vrácen pouze předdefinovaný počet termínů dle politiky provozovny a nastavení konkrétní fronty.
  - `/id={queue_ID}&to={DATE}&since={DATE}…`

Dotaz na server:

```
/public/index.php/myqueues/id={queue_ID}&id={queue_ID2}&attr={service}&to={DATE}&since={DATE}/
```

Odpověď serveru:

```
<response>
  <item key="{queue_ID}">
    <name>{NAZEV}</name>            //název fronty zobrazovaný zákazníkovi
    <q_id>{queue_ID}</q_id>         //identifikátor fronty
    <uri>/public/index.php/myqueues/id={queue_ID}/</uri>
      //přímý odkaz na danou frontu
    <service>{service}</service>    //název poskytované služby
    <service>…</service>            //další poskytované služby - volitelné
    <dates>
      <id>{IDT}</id>                //identifikátor termínu
      <since>{TIME+DATE}</since>    //začátek termínu
      <to>{TIME+DATE}</to>          //konec termínu
      <act_fill>{NUM}</act_fill>    //počet přihlášených zákazníků
      <max_fill>{NUM}</max_fill>    //maximální počet zákazníků
      <info>{DI}</info>             //dostupné informace k termínu
    </dates>
    <dates>…</dates>                //další termíny
  </item>
</response>
```

5.2.4 Požadavek na zařazení do fronty

Pro úspěšné vykonání požadavku musí být zákazník přihlášen pod stejným identifikátorem, jako je identifikátor zákazníka {**customer_ID**} v požadavku. Pokud ověření proběhne úspěšně, tak se server pokusí přihlásit zákazníka do fronty {**queue_ID**} a na termín {**appointment_ID**}. Server do odpovědi zanese mj. i informace o frontě, termínu a přesném čase, na který byl zákazník skutečně přihlášen. Tyto informace se mohou lišit od informací v požadavku. Ke změně může dojít v důsledku optimalizace. Její míra využití závisí na politice provozovny a nastavení fronty. Pin pro potvrzení příchodu do čekárny {**PPP**} se v odpovědi serveru vyskytuje pouze v případě, že požadavek provedl Klient mobilní.

{**PN**}: Požadavek může podléhat ručnímu potvrzení od obsluhy (provozovatele). Pokud podle politiky provozovny a nastavení fronty toto potvrzení není vyžadováno, je požadavek potvrzen automaticky při zpracování.

Dotaz na server:

```
/public/index.php/myqueues/{queue_ID}/appointment/{appointment_ID}/{customer_ID}/
```

Odpověď serveru:

```
<response>
  <ack>appointment</ack>
  <request_id>{JIDP}</request_id>
    //jedinečný identifikátor požadavku
  <state>{PN}</state>
    //potvrzený nebo nepotvrzený požadavek
  <queue_id>{IDF}</queue_id>
    //identifikátor fronty, na kterou je zákazník přihlášen
  <appointment_id>{IDT}</appointment_id>
    //identifikátor termínu, na který je zákazník přihlášen
  <since>{TIME+DATE}</since>
    //začátek termínu
  <to>{TIME+DATE}</to>
    //konec termínu
//nepovinná data:
  <request_pin>{PPP}</request_pin>
    //pin pro potvrzení příchodu do čekárny
  <id_wroom>{PČ}</id_wroom>  //pořadové číslo, pokud je již přiřazeno
</response>
```

5.2.5 Požadavek na vyřazení z fronty

Požadavek na vyřazení zákazníka z fronty musí splňovat stejné požadavky na ověření zákazníka jako pro zařazení do fronty. Navíc musí přihlášený zákazník znát jedinečný identifikátor požadavku {**JIDP**} a pro kontrolu i identifikátor fronty {**IDF**}, do které je zákazník přihlášen.

Stav požadavku {**PNO**} v odpovědi serveru se předává jako jeden znak a může nabývat těchto třech hodnot:

- „**P**“: **přijato** – Požadavek na vyřazení zákazníka byl úspěšně přijat a zákazník byl úspěšně z fronty vyřazen.
- „**N**“: **nepotvrzeno** – Požadavek podléhá ručnímu potvrzení ze strany obsluhy (provozovatele). O jeho přijetí nebo odmítnutí rozhodne obsluha (provozovatel) později.

- „**O**“: **odmítnuto** – Požadavek nebyl přijat. Například se dle politiky provozovny a nastavení fronty není možno odhlásit v bezprostřední blízkosti začátku přihlášeného termínu.

Dotaz na server:

```
/public/index.php/myqueues/{IDF}/refuseapp/{JIDP}/{customer_ID}/
```

Odpověď serveru:

```
<response>
  <ack>refuseapp</ack>
  <state>{PNO}</state>        //přijato; nepotvrzeno; odmítnuto
  <msg></msg>                 //zpráva s více informacemi pro zákazníka
  <request_id>{JIDP}</request_id>
</response>
```

5.2.6 Předání stavu zákazníka o příchodu do čekárny

Předem objednaný zákazník po příchodu do provozovny zadá pin {**PPP**} do Klienta v čekárně, nebo potvrdí příchod Klientem mobilním. Po úspěšném předání požadavku je zákazníkovi přiřazeno pořadové číslo {**PČ**} a jeho stav se změní na „1“ – čeká. Od chvíle přidělení pořadového čísla lze zákazníka z fronty vyvolat přepážkou.

Dotaz na server:

```
/public/index.php/myqueues/{queue_ID}/customarrival/{PPP}/
```

Odpověď serveru:

```
<response>
  <ack>customarrival</ack>
  <id_wroom>{PČ}</id_wroom>        //pořadové číslo zákazníka
</response>
```

5.3 WroomController

Dotazy popsané v následující kapitole využívá zejména Klient v čekárně. Pro načítání konfigurace, předání svého stavu serveru a požádání o zařazení nového zákazníka do fronty.

Formát dotazu na server:

```
/public/index.php/waitingroom/…
```

5.3.1 Načtení konfigurace Klienta v čekárně

Klient v čekárně se pokusí vždy po svém zapnutí načíst konfiguraci ze serveru pomocí identifikátoru provozovny {**company_ID**} a svého pořadového čísla {**printer_NUM**}, které je jedinečné v rámci jedné provozovny. Každý Klient v čekárně má svoji vlastní konfiguraci uloženou na serveru a poslední funkční verzi konfigurace uloženou v lokální paměti. Při neúspěšném provedení tohoto požadavku si Klient v čekárně načte lokálně uloženou konfiguraci.

Dotaz na server:

```
/public/index.php/waitingroom/{company_ID}/printer/{printer_NUM}/
```

Odpověď serveru:

```
<response>
  <ack>getcfgWroom</ack>
  <printer>
    <p_num>{IPČ}</p_num>      //interní pořadové číslo
    <p_id>{printer_ID}</p_id> //identifikátor klienta pro další komunikaci
    <uri>www</uri>
    <buttons>
      <b_id>{IPČT}</b_id>     //interní pořadové číslo tlačítka
      <name>{JMENO}</name>    //název tlačítka zobrazovaný zákazníkovi
      <q_id>{IDF}</q_id>
//nepovinná data:
      <buttons>…</buttons>
      //tlačítka druhé až N úrovně
      //pokud se nejedná o tlačítko N úrovně.
      <reg_uri>{APPZSP}</reg_uri>
      //adresa požadavku pro přímé zařazení s parametry
      //pokud se jedná o tlačítko N úrovně (neobsahuje další tlačítka)
    </buttons>
    <buttons>…</buttons>     //další tlačítka první úrovně
  </printer>
</response>
```

5.3.2 Požadavek na zařazení zákazníka do fronty

Slouží pro zařazení nového zákazníka do konkrétní fronty {**queue_ID**}. Pro doplňující informace o zákazníkovi nebo o upřesnění termínu, na který se zákazník chce přihlásit, slouží {**usr_data**}. Informace o zákazníkovi {**usr_data**} mohou obsahovat atributy popsané v bodě 5.2.1 nebo 5.4.7, které jsou nepovinné. Přiřazené pořadové číslo vrátí server v kladné odpovědi na tento požadavek a Klient v čekárně ho zobrazí zákazníkovi, nebo ve spojení s tiskárnou mu ho vytiskne.

Zákazníka lze zařadit do fronty několika způsoby, které jsou závislé na množství o něm vyplněných nepovinných informací.

- **Anonymně** – není nastaven atribut priority ani identifikátor termínu.
  - {usr_data} => `null`
- **Anonymně s prioritou** – je nastaven atribut priority na hodnotu 1.
  - {usr_data} => `prior=1`
- **Na konec fronty** – je nastaven alespoň jeden z atributů pro jméno „`name`“ nebo skrytý identifikátor „`insurance_number`“.
  - {usr_data} => `name={JMENO}&insurance_number={NUMBER}`
- **Na konec fronty s prioritou** – je nastaven navíc atribut priority na hodnotu 1.
  - {usr_data} => `name={JMENO}&insurance_number={NUMBER}&prior=1`
- **Na termín** – je nastaven identifikátor termínu společně s rozšiřujícími údaji o termínu, které jsou nepovinné.
  - {usr_data} => `name={JMENO}&id={IDT}&since={DATUM}&to={DATUM}…`

Dotaz na server:

`/public/index.php/waitingroom/{queue_ID}/regque/{usr_data}/`

Odpověď serveru:

```
<response>
      <ack>regque</ack>
      <id_wroom>{PČ}</id_wroom>     //přiřazené pořadové číslo zákazníka
//nepovinná data:
      <msg>…</msg>      //doplňující sdělení, které se vytiskne na lístek
</response>
```

5.3.3 Předání stavu Klienta v čekárně na server

Klient v čekárně {**printer_ID**} pravidelně předává serveru svůj stav {**status_ID**} tímto požadavkem. Komunikace se serverem by měla probíhat v krátkých intervalech, alespoň jednou za 10s nebo častěji. Server v odpovědi předává různé požadavky, například aby si klient aktualizoval konfiguraci nebo se restartoval. Tyto požadavky jsou nepovinné a nemusí se v odpovědi vyskytnout. Požadavek je nastaven na hodnotu 1 jen v případě, že je server vyžaduje. Jako další nepovinná část odpovědi může být textová zpráva, která se má zobrazit na displeji. Tato zpráva bude obsahovat různé informace pro zákazníka o aktuálním stavu či reklamní sdělení.

Stav přepážky {**status_ID**} v dotazu a v odpovědi serveru se předává jako jeden znak a může nabývat těchto hodnot:

- „**D**“: **příprava** – Klient v čekárně komunikuje, nemá ještě načtenou konfiguraci, tudíž není připraven obsluhovat zákazníka.
- „**R**“: **připraven** – Klient v čekárně komunikuje, má úspěšně načtenou konfiguraci a je připraven k obsloužení zákazníka.
- „**P**“: **tiskne** – zákazníkovi je právě jedním ze způsobů vydáváno pořadové číslo, není připraven k obsloužení dalšího zákazníka.
- „**W**“: **varování** – Klient v čekárně nemůže vydat zákazníkovi jeho pořadové číslo. Například tiskárně došel papír.
- „.“: **nekomunikuje** – Klient v čekárně nekomunikuje (vyskytuje se pouze v odpovědi serveru).

Dotaz na server:

```
/public/index.php/waitingroom/{printer_ID}/status/{status_ID}/
```

Odpověď serveru:

```
<response>
  <ack>status</ack>
  <restart_printer>1</restart_printer> //restartování Klienta v čekárně
  <close_printer>1</close_printer>     //ukončení obsluhy zákazníků
  <getcfg>1</getcfg>                   //Klient si má přečíst konfiguraci
  <msg>{MSG}</msg>                     //zpráva pro zákazníka
</response>
```

5.4 CroomController

Umožňuje obsluhu Klienta v provozovně, tedy připojení a odpojení přepážky, přihlášení a odhlášení obsluhujícího, načtení fronty zákazníků, vyvolání konkrétního zákazníka na přepážku, ukončení obsluhy nebo přeložení zákazníka jinam.

Formát dotazu na server:

`/public/index.php/counter/…`

5.4.1 Připojení přepážky k serveru

Jako první se vždy musí přepážka připojit k serveru identifikátorem provozovny {**company_ID**} a pod svým interním pořadovým číslem {**num_counter**}. Bez úspěšného připojení přepážky nebude server vyhodnocovat žádné další dotazy. V úspěšné odpovědi na připojení přepážka obdrží své unikátní ID, kterým se bude při pozdější komunikaci identifikovat a uživatelské číslo přepážky, které se bude zobrazovat na informačních obrazovkách. Takto připojená přepážka může vyčíst svoji konfiguraci ze serveru a frontu zákazníků.

Dotaz na server:

/public/index.php/counter/reg/{**num_counter**}/compay/{**company_ID**}/

Odpověď serveru:

```
<response>
    <cntr_id>{counter_ID}</cntr_id>        //přidělení unikátního ID pro
                                           //další dotazy
    <usr_cntr_id>{ČP}</usr_cntr_id>        //uživatelské číslo přepážky
</response>
```

5.4.2 Odpojení přepážky od serveru

Dotaz na server:

/public/index.php/counter/unreg/{**counter_ID**}/

Odpověď serveru:

```
<response>
      <ack>unreg</ack>
</response>
```

5.4.3 Přihlášení obsluhujícího na přepážku

Pro obsluhu zákazníků (vytvoření, vyvolání, přeložení, ukončení) se musí na přepážku přihlásit obsluhující. Ten se přihlásí pomocí předem přiděleného hesla.

Dotaz na server:

/public/index.php/counter/login/{**counter_ID**}/code/{**opcode**}/

Odpověď serveru:

```
<response>
      <ack>login</ack>
      <opnumber>{IPO}</opnumber>    //identifikátor přihlášené obsluhy
</response>
```

5.4.4 Odhlášení obsluhujícího z přepážky

Používá se při výměně obsluhujících na přepážce. Aby se mohl nový obsluhující přihlásit, musí se nejprve přihlášený obsluhující odhlásit. Server má díky tomu aktuální informace o tom, kteří obsluhující již nejsou přihlášeni do Klienta v provozovně.

Dotaz na server:

/public/index.php/counter/logout/{**counter_ID**}/

Odpověď serveru:

```
<response>
      <ack>logout</ack>
</response>
```

5.4.5 Načtení konfigurace přepážky

Načtením konfigurace se přenese seznam front pro danou přepážku {**counter_ID**} a obsluhujícího {**opnumber**}. Pro každou frontu se přenese její identifikátor {**IDF**} a odkaz pro vyčtení podrobnějších informací o frontě.

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/opnumber/{opnumber}/
```

Odpověď serveru:

```
<response>
  <que>
    <id>{IDF}</id>                  //identifikátor fronty
    <uri> /public/index.php/myqueues/id={id}/</uri>
  </que>
  <que>…</que>                      //další fronty…
  <theop>
    <opnumber>{IDO}</opnumber>      //identifikátor obsluhy
    <name>{JMOBS}</name>            //jméno obsluhy
  </theop>
  <theop>…</theop>                  //další obsluhující…
</response>
```

5.4.6 Předání stavu přepážky na server

Tímto dotazem přepážka pravidelně předává serveru svůj stav. Komunikace se serverem musí probíhat v krátkých intervalech, alespoň jednou za 10s nebo častěji. Server v odpovědi předává stavy ostatních přepážek, všech Klientů v čekárně a různé požadavky (aby si přepážka aktualizovala frontu, konfiguraci, nebo aby se odpojila od serveru). Tyto požadavky jsou nepovinné a nemusí se v odpovědi vyskytnout. Požadavek je nastaven na hodnotu „1“ jen v případě, že je server vyžaduje. Jako další nepovinná část odpovědi může být textová zpráva, která se má zobrazit obsluhujícímu.

Stav přepážky {**status_ID**} v dotazu a v odpovědi serveru se předává jako jeden znak a může nabývat hodnot:

- „**S**“: **obsluhuje** – obsluhující přihlášen, zákazník je právě obsluhován.
- „**C**“: **zavřeno** – obsluhující přihlášen, ale žádný zákazník není právě obsluhován.
- „**L**“: **odhlášena** – obsluhující nepřihlášen, ale přepážka komunikuje.
- „.“: **nekomunikuje-** přepážka nekomunikuje (vyskytuje se pouze v odpovědi serveru).

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/status/{status_ID}/
```

Odpověď serveru:

```
<response>
  <ack>givecfg</ack>
  <statlist>
    <id_counter>{counter_ID}</id_counter>  //identifikátor přepážky
    <status>{status_ID}</status>           //stav přepážky S;C;L
  </statlist>
  <statlist>…</statlist>                   //další přepážky
  <shutdown_counter>1</shutdown_counter>   //odpojení přepážky od serveru
  <close_counter>1</close_counter>         //ukončení obsluhy zákazníka
  <getcfg>1</getcfg>                  //přepážka si má přečíst konfiguraci
  <getque>1</getque>                  //přepážka si má přečíst frontu
  <msg_op>{MSG}</msg_op>              //zpráva pro obsluhujícího
</response>
```

5.4.7 Přenesení fronty zákazníků na přepážku

Přenáší se zákazníci z front, které přepážka obsluhuje. U každého zákazníka se vždy přenese jeho pořadové číslo a identifikátor fronty. Ostatní parametry se přenesou, pokud jsou nastaveny. Zákazníci vyvolaní mimo frontu a přeložení ke konkrétní přepážce mají zvláštní identifikátor fronty.

Zákazník se může nacházet v jednom ze čtyř stavů {**stavID**}, které jsou již popsány z pohledu významu pro zákazníka v: *5.2.1 Dotaz na moje fronty.* Zde budou rozebrány z pohledu obsluhy, jaký pro ni mají jednotlivé stavy význam:

- „**0**“: **není přítomen** - Zákazník je přiřazen do fronty, nejspíše na konkrétní termín, ale ještě nepřišel fyzicky do čekárny. Jeho stav z „0“ do „1“ se změní ve chvíli, kdy zadá přidělený pin do Klienta v čekárně, nebo v aplikaci odsouhlasí, že již do čekárny přišel. Ve výpisu fronty se zákazník zobrazí, ale dokud se nebude nacházet ve stavu „1“, tak ho nelze vyvolat. Obsluhující ho ale může přeložit.
- „**1**“: **čeká** - Zákazník přišel fyzicky do čekárny. Registroval se přímo ve svojí aplikaci, nebo si v Klientovi v čekárně zvolil požadovanou frontu a bylo mu vydáno pořadové číslo. Zákazníka je možné vyvolat.
- „**2**“: **obsluhován** - Právě je zákazník obsluhován některou z přepážek. Nelze ho tudíž vyvolat k žádné jiné přepážce. Přechod ze stavu „1“ do „2“ je proveden ve chvíli vyvolání zákazníka přepážkou.
- „**3**“: **obsloužen** - Zákazník již byl obsloužen a není přeložen do další fronty. Záznam o zákazníkovi se v odpovědi serveru nemusí objevit, pokud byl do fronty přiřazen Klientem v čekárně bez předchozí registrace. Dále záleží na nastavení fronty.

Dotaz na server:

```
/public/index.php/counter/getque/{counter_ID}/
```

Odpověď serveru:

```
<response>
  <ack>getque</ack>
  <customer>
    <id_wroom>{PČ}</id_wroom>          //vyvolané pořadové číslo
    <queue_id>{IDF}</queue_id>         //id fronty, ve které čeká
```

```
//následují nepovinná data:
     <prior>1</prior>                        //určuje prioritního zákazníka
     <status>{stavID}</status>               //stav zákazníka: 0;1;2;3
     <name>{JMENO}</name>
       //jméno, které se může tisknout na lístek s pořadovým číslem
     <insurance_number>{CISLO}</insurance_number>
       //skrytý identifikátor, který se nemůže tisknout na lístek s PČ
     <appointment_id>{IDT}</appointment_id>
       //ID termínu, na který je zákazník objednaný předem
     <custom_id>{IDZ}</custom_id>
       //jedinečné ID zákazníka (pokud je registrovaný a předem objednaný)
  </customer>
  <customer>…</customer>                          //další zákazníci
</response>
```

5.4.8 Vyvolání zákazníka z konkrétní fronty

Aby přepážka mohla provést vyvolání zákazníka, musí na ní být přihlášen obsluhující. Přepážka předá pořadové číslo zákazníka {**SN**} serveru společně se svým identifikátorem přepážky {**counter_ID**} a identifikátorem fronty {**ID_queue**}. Pokud se v té konkrétní frontě zákazník nachází, tak je vyvolán. Pokud nelze zákazníka vyvolat, vrátí server chybovou odpověď (například byl zákazník již dříve vyvolán jinou přepážkou a nachází se stále ve stavu „2“). Po vyvolání zákazníka musí přepážka vždy přečíst aktualizovanou frontu zákazníků. Pokud přepážka ve chvíli úspěšného vyvolání obsluhuje jiného zákazníka, je při vyvolání tato obsluha automaticky ukončena (stav obsluhovaného zákazníka se automaticky změní na stav „3“ a k přepážce je vyvolán nový zákazník).

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/queue/{queue_ID}/sn/{SN}/
```

Odpověď serveru:

```
<response>
  <ack>qcall</ack>                        //qcall: vyvolání z fronty
  <customer>
    <id_wroom>{PČ}</id_wroom>             //vyvolané pořadové číslo
    <queue_id>{IDF}</queue_id>            //id fronty, ze které byl vyvolán
//následují nepovinná data:
     <status>{stavID}</status>
     //stav, ve kterém se aktuálně nachází zákazník: 0;1;2;3
     <name>{JMENO}</name>
     //jméno, které se může tisknout na lístek s pořadovým číslem
     <insurance_number>{CISLO}</insurance_number>
     //skrytý identifikátor, který se nemůže tisknout na lístek s PČ
     <appointment_id>{IDT}</appointment_id>
     //ID termínu, na který je zákazník objednaný předem
     <custom_id>{IDZ}</custom_id>
     //jedinečný identifikátor zákazníka
     // (pokud je registrovaný a předem objednaný)
  </customer>
</response>
```

5.4.9 Vyvolání zákazníka mimo frontu

Mimo čísel zákazníků, kteří jsou v námi známé frontě, může přepážka vyvolat libovolné číslo předáním pořadového čísla zákazníka {**SN**} serveru společně se svým identifikátorem {**counter_ID**}. Aby přepážka mohla provést vyvolání zákazníka, musí na ni být přihlášen obsluhující. Využívá se například při vyvolání zákazníka, jehož obsluha již byla ukončena a z nějakého důvodu nebyl přeložen jinam. Pokud je volané číslo nalezeno v obsluhované frontě přepážkou, je vyvoláno běžným způsobem z fronty (viz vyvolání zákazníka z konkrétní fronty). Všem zákazníkům volaným mimo frontu je přidělen zvláštní identifikátor zákazníka. Je to jeden ze způsobů, jak vytvořit nového zákazníka. Po jeho vyvolání ho může obsluhující přeložit jinam (do konkrétní fronty, přepážky nebo k obsluhujícímu).

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/sn/{SN}/
```

Odpověď serveru:

```
response>
  <ack>outcall</ack>                //outcall: vyvolání mimo frontu
  <customer>
    <id_wroom>{PČ}</id_wroom>
    <custom_id>{ZIDZ}</custom_id>  //zvláštní identifikátor zákazníka
                                   //určující vytvoření vyvoláním mimo frontu
  </customer>
</response>
```

5.4.10 Opakované vyvolání konkrétního zákazníka

Pokud se zákazník po vyvolání jeho pořadového čísla {**SN**} nedostaví k přepážce {**counter_ID**}, lze ho vyvolat opakovaně. Opakované vyvolání má za následek znovu rozblikání pořadového čísla na informačních tabulích. Opakovaně lze volat pouze přepážkou, která je ve stavu „S“ – obsluhuje a klienta, který je ve stavu „2“ – je obsluhován.

Dotaz na server:

```
/public/index.php/counter/recall/{counter_ID}/sn/{SN}/
```

Odpověď serveru:

```
<response>
  <ack>recall</ack>
  <customer>
    <id_wroom>{PČ}</id_wroom>
    <custom_id>{IDZ}</custom_id>   //jedinečný identifikátor zákazníka
                  // (pokud je zákazník registrovaný a předem objednaný)
  </customer>
</response>
```

5.4.11 Ukončení obsluhy zákazníka

Použije se, pokud zákazník fyzicky odejde od přepážky a obsluha nemůže nebo nechce vyvolávat dalšího klienta z fronty. Stav přepážky je poté změněn ze stavu „**S**“ – obsluhuje, na stav „**C**“ – zavřeno.

Dotaz na server:

```
/public/index.php/counter/done/{counter_ID}/
```

Odpověď serveru:

```
<response>
      <ack>done</ack>
</response>
```

5.4.12 Přeložení obsluhovaného zákazníka na jinou přepážku

Ukončí obsluhu zákazníka na přepážce {**counter_ID**}, identifikátor zákazníka společně s dalšími informacemi o zákazníkovi {**usr_data**} vloží do fronty jiné konkrétní přepážky {**counter_ID_new**}. Pouze tato přepážka vidí daného zákazníka ve frontě a může ho z fronty vyvolat. Ostatní přepážky mohou tohoto zákazníka vyvolat pouze voláním mimo frontu.

Informace o zákazníkovi {**usr_data**} obsahují mimo povinného pořadového čísla i ostatní atributy z bodu číslo 5.4.7, které jsou v některých případech nepovinné.

Zákazníka lze přeložit několika způsoby, které jsou závislé na množství vyplněných nepovinných informací o zákazníkovi.

- **Na konec fronty** – není nastaven atribut priority ani identifikátor termínu.
  - {usr_data} => `/id_wroom={PČ}/`
- **Na konec prioritní fronty** – je nastaven atribut priority na hodnotu 1.
  - {usr_data} => `/id_wroom={PČ}&prior=1/`
- **Na termín** – je nastaven identifikátor fronty společně s příslušným identifikátorem termínu a rozšiřujícími údaji o termínu, které jsou nepovinné.
  - {usr_data} => `/id_wroom={PČ}&name={JMENO}&id={IDT}&since={DATUM}&to={DATUM}/`

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/tocntr/{counter_ID_new}/{usr_data}/
```

Odpověď serveru:

```
<response>
  <ack>tocntr</ack>
</response>
```

5.4.13 Přeložení obsluhovaného zákazníka do konkrétní fronty

Ukončí obsluhu zákazníka na přepážce {**counter_ID**}, identifikátor zákazníka společně s dalšími informacemi o zákazníkovi {**usr_data**} vloží do jiné konkrétní fronty {**queue_ID_new**}.

Zákazníka lze přeložit stejnými způsoby na konec fronty, s prioritou nebo na konkrétní čas (termín) viz bod: *5.4.12 Přeložení obsluhovaného zákazníka na jinou přepážku*.

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/toqueue/{queue_ID_new}/{usr_data}/
```

Odpověď serveru:

```
<response>
  <ack>toqueue</ack>
</response>
```

5.4.14 Přeložení obsluhovaného zákazníka k obsluhujícímu

Ukončí obsluhu zákazníka na přepážce {**counter_ID**}, identifikátor zákazníka společně s dalšími informacemi o zákazníkovi {**usr_data**} vloží do fronty se zvoleným číslem

obsluhujícího {**operator_ID_new**}. Pouze tento obsluhující, přihlášený na jedné z přepážek, pak může zákazníka vyvolat. Ostatní přepážky mohou tohoto zákazníka vyvolat pouze voláním mimo frontu.

Zákazníka lze přeložit stejnými způsoby na konec fronty, s prioritou nebo na konkrétní čas (termín) viz bod: *5.4.12 Přeložení obsluhovaného zákazníka na jinou přepážku*.

Dotaz na server:

```
/public/index.php/counter/{counter_ID}/toope/{operator_ID_new}/{usr_data}/
```

Odpověď serveru:

```
<response>
  <ack>toope</ack>
</response>
```

5.5 Chyba při vykonání některého z dotazů

Server posílá tuto zprávu v případě, že některý z předchozích dotazů nemohl z nějakého důvodu vyhodnotit společně s odpovídajícím stavovým kódem v hlavičce odpovědi viz bod *4.3 HTTP*. Chybový kód v tagu `<err_code>` je přiložen v odpovědi především pro upřesnění, kde daná chyba vznikla z pohledu návrhu a nemá žádný vztah ke druhu chyby.

Dotaz na server:

```
Libovolný dotaz uvedený výše.
```

Odpověď serveru:

```
<response>
  <ack>err</ack>
    //příznak, že dotaz nebyl korektně vyhodnocen
  <err_code>{CISLO}</err_code>
    //chybový kód, pro lepší dohledání příčiny poruchy nebo chyby
  <msg>{POPIS}</msg>
    //popis chyby, který má být zobrazen v aplikaci
</response>
```

6 Testování

Tato kapitola obsahuje popis použitých testovacích nástrojů, pomocí kterých byla ověřena funkčnost navrhovaného řešení protokolu.

Testování bylo prováděno v průběhu vývoje s pomocí běžného webového prohlížeče a jeho rozšíření na přehlednost a použitelnost dotazů společně s dostupností všech požadovaných informací při co nejmenším možném počtu dotazů.

Testování protokolů běžně probíhá formálními metodami za pomoci specializovaných software řešení (testování modelů, např. pomocí softwaru UPPAAL[10]), jejichž teorie překračuje hranici této bakalářské práce, která se zabývá návrhem síťové komunikace a ne přímo jejím testováním. Proto byl každý z výše popsaných požadavků otestován ručním zadáním do webového prohlížeče. Následně byla zkontrolována úplnost a relevantnost vrácených dat. Požadavky byly dále záměrně testovány s chybami, aby se ověřila reakce serverové aplikace na chybný dotaz nebo na dotaz správný, který se táže na chybná data. Další testování probíhalo ze strany autora mobilní aplikace (Klient mobilní), která byla vyvinuta autorem v rámci bakalářské práce [16].

Mezi nejvýznamnější výsledky testování patří:

- sjednocení názvů všech tagů v odpovědích serveru a přeložení do anglického jazyka
- doplnění podrobnějších informací o provozovně, především doplnění záznamů o přepážkách a jednotlivých odděleních provozovny
- přidání každému zákazníkovi stav `<status>`, ve kterém se právě nachází

Webový prohlížeč

Veškeré výše uvedené dotazy lze zadat do libovolného webového prohlížeče a získat požadovaná data v odpovědi serveru. Po přidání příslušné adresy serveru, na kterém je spuštěna serverová část pro přijímání dotazů, je vrácena odpověď viz obrázek 12.

Obrázek 12: Ukázka testování protokolu pomocí webového prohlížeče

Advanced REST client

Jedná se o rozšíření Google Chrome prohlížeče o REST klienta, který vytváří a testuje individuální HTTP požadavky. Především lze editovat hlavičku http požadavku například o položku Content-Type. Dokáže zobrazit kompletní hlavičku odpovědi a přehledně zobrazit výsledky dotazu jako prostý text nebo HTML, JSON a XML prohlížečem [17].

[10] http://www.uppaal.org

7 Závěr

Ke splnění požadavků zadání a demonstraci zvládnuté problematiky byla navržena ukázková serverová aplikace založená na PHP micro-frameworku Silex, který vychází z frameworku Symphony. Serverová část byla navržena společně s ukázkovými daty pro poskytování informací o struktuře provozoven, jejich frontách a zákaznících, kteří jsou registrováni, veškerých důležitých informacích pro budoucí rozšíření a napojení na mobilní aplikace.

Testování probíhalo společně s vývojem protokolu a ukázalo několik chyb v původním návrhu, které byly odstraněny.

Práce pro mě byla zajímavá a v mnoha ohledech poučná. Pomohla mi ke zpřehlednění veškerých již známých informací o komunikaci mezi klienty a serverem a navíc mi přinesla mnoho nových poznatků a zkušeností. Jako přínos této práce vidím její možné využití jako základ pro vytvoření komunikační vrstvy mezi klienty a serverem společně s integrací ostatních částí systému Virtuální čekárny.

Možnost rozšíření práce

Tato práce byla navržena jako otevřený protokol pro budoucí rozšíření o autentifikaci zákazníků

- Rozšířit samotný protokol o možnost registrace, autentifikaci a autorizaci jednotlivých dotazů vůči serveru.
- Podpora notifikací. Zvolit konkrétní formu notifikace zákazníka o změnách v termínech nebo samotného vyvolání k přepážce bez nutnosti využívání informačních obrazovek.
- Rozšířit protokol o komunikaci s externí tiskárnou pořadových čísel pro Klienta v čekárně společně s čtečkou zákaznických nebo jiných identifikačních karet či průkazů.
- Rozšířit protokol o vzdálený tisk lístků s pořadovým číslem. Pro předem objednané zákazníky, kteří potvrdí svůj příchod do čekárny pomocí aplikace (Klient mobilní) umožnit vytištění pořadového čísla na libovolné tiskárně pomocí načtení QR kódu nebo ručního zadání unikátního identifikátoru tiskárny.

Možná optimalizace a vylepšení

Jako optimalizaci protokolu navrhuji především vypuštění některých redundantních parametrů v dotazech, které by mohli být načteny z databáze na straně serveru. Díky tomu by se mj. zvýšila bezpečnost, kdy si například sice ověřený klient požádá o data, která mu nepřísluší a jsou pro jeho správné fungování nepotřebná.

Jako další optimalizaci navrhuji odstranění pravidelné komunikace Klienta v čekárně (viz *5.3.3 Předání stavu Klienta v čekárně na server*) například nahrazením notifikační zprávy ze strany serveru, že má Klient v čekárně zaslat údaje o svém stavu na server.

Literatura

[1] *IEEE communications magazine*. New York: Communications Society of Institute of Electrical and Electronics Engineers, 1979-. s46 – 55. ISSN 01636804.

[2] MAJTÁN, Martin. *Klient-server applikace založená na technologii CORBA*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, c2014. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=83887>

[3] SEELY, Scott. *SOAP: cross platform Web service development using XML*. Upper Saddle River, NJ: Prentice Hall PTR, 2002. vlastní překlad citace. ISBN 0130907634.

[4] FIELDING, Roy T. a kolektiv. *Hypertext Transfer Protocol -- HTTP/1.1* [online]. c1999-06 [cit. 2016-05-02]. Dostupné z: <http://tools.ietf.org/html/rfc2616>

[5] KŘENA, Bohuslav a Radek KOČÍ. *Úvod do softwarového inženýrství IUS*, Studijní opora [online]. c2010-12-21, s. 47-54. [cit. 2016-04-01]. Dostupné z: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IUS-IT/texts/IUS_opora.pdf>

[6] TAYLOR, Richard N., Nenad. MEDVIDOVIĆ a Eric M. DASHOFY. *Software architecture: foundations, theory, and practice*. Hoboken, NJ: John Wiley, c2010. vlastní překlad citace. ISBN 0470167742.

[7] WIEGERS, Karl Eugene. *Požadavky na software*. Brno: Computer Press, 2008. ISBN 9788025118771.

[8] GORTON, Ian. *Essential software architecture*. 2nd ed. Heidelberg: Springer, 2011. s. 6-11. ISBN 3642191754.

[9] COULOURIS, George F. *Distributed systems: concepts and design*. 5th ed. Boston: Addison-Wesley, 2012. vlastní překlad citace. ISBN 0132143011.

[10] LEISS, Oliver a Jasmin SCHMIDT. PHP v praxi: pro začátečníky a mírně pokročilé. Praha: Grada, 2010. Průvodce (Grada). ISBN 9788024730608.

[11] A PHP micro-framework standing on the shoulder of giants. *Silex - The PHP micro-framework based on Symfony2 Components* [online]. [cit. 2016-04-08]. Dostupné z: <http://silex.sensiolabs.org>

[12] FREED, Ned a kolektiv. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* [online]. c1996-11 [cit. 2016-05-02]. Dostupné z: <https://tools.ietf.org/html/rfc2045>

[13] FIELDING, Roy T. *Architectural Styles and the Design of Network-based Software Architectures* [online]. Doctoral dissertation, University of California, Irvine, c2000 [cit. 2016-05-15] Dostupné z: <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf >

[14] LI, L a W. CHOU. *Design and Describe REST API without Violating REST: A Petri Net Based Approach* [online]. Web Services (ICWS), 2011 IEEE International Conference on, Washington, DC, c2011, s. 508-515. [cit. 2016-05-15]. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6009431&isnumber=6009354>

[15] DRYTKIEWICZ, W. a kolektiv. *pREST: a REST-based protocol for pervasive systems* [online]. Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on, c2004, s. 340-348. [cit. 2016-05-15]. Dostupné z: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1392173&isnumber=30305>

[16] BUKOVSKÝ, Luděk. *Klient virtuální čekárny pro iOS,* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[17] PSZTYĆ, Pawel. *Advanced REST client for Google Chrome* [online]. c2004-01 [cit. 2016-04-13]. Dostupné z: <http://advancedrestclient.com>

Seznam příloh

Příloha 1. DVD s prototypem serverové aplikace