Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Aplikace pro vzdálené ovládání videokonferenčních zařízení z mobilních telefonů

Bc. Vojtěch Hlavačka

Vedoucí práce: Mgr. Soběslav Benda

5. května 2015

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu Mgr. Soběslavu Bendovi za jeho čas strávený vedením této práce a za cenné rady. Dále bych rád poděkoval svému konzultantovi Ing. Ondřejovi Procházkovi za veškeré rady a připomínky jak k technické části, tak k formální části práce. Dále bych rád poděkoval Bc. Kristýně Valdové a Ing. Lucii Myšičkové za testování aplikace a revizi textů. Nakonec bych chtěl poděkovat své rodině za podporu během celého mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 5. května 2015 .....

Odkaz na tuto práci

Hlavačka, Vojtěch. *Aplikace pro vzdálené ovládání videokonferenčních zařízení z mobilních telefonů* . Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zabývá návrhem a implementací aplikace, pomocí které lze ovládat videokonferenční zařízení od společnosti Cisco z mobilních telefonů. Součástí této práce je i návrh a implementace webového administračního rozhraní, které umožní definovat autorizaci přístupu uživatelů k videokonferenčním zařízením.

Klíčová slova Videokonference, videokonferenční zařízení, Cisco, ovládání, Android.

Abstract

This thesis describes the design and implementation of application that can control Cisco video conferencing equipment using mobile phones. Thesis also contains the design and implementation of web administration interface that allows users to define authorization of access to video conferencing equipment.

Keywords Video conferencing, video conferencing equipment, Cisco, control, Android.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Právě mezilidská komunikace je základním nástrojem sociální interakce a je tedy pro každého člověka žijícího ve společnosti nezbytná. A to jak v soukromém, tak i pracovním životě.

Nejlepším způsobem mezilidské komunikace je osobní kontakt. Tato možnost však není vždy dostupná, někdy je například nutné strávit hodiny na cestách. Díky moderním technologiím naší doby existují i jiné možnosti mezilidské komunikace, které ovšem nevyžadují lidský kontakt. Jelikož slova tvoří asi 10 % z komunikace, hlas (zvuky, tóny, hlasitost) 30 % a největší podíl má právě řeč našeho těla 60 % [2, str. 13], je pochopitelný velký rozmach videokonferenčních zařízení, která se snaží osobní kontakt co nejlépe nahradit. Aby byl tento způsob komunikace úspěšný, je zapotřebí nejen vysoká kvalita obrazu a zvuku, ale i velká uživatelská přívětivost zařízení, která tuto konferenci zprostředkovávají.

Společnost Cisco vlastní celou řadu videokonferenčních zařízení a příslušné infrastruktury. Škála produktů nabízejících video spojení je široká. Začíná u malých IP telefonů s možností videa přes zařízení v podobě stolního monitoru až po vybavení velkých konferenčních místností. Kvalita jejich obrazu a zvuku je na vysoké úrovni. Jejich ovládání je však různé, ne vždy uživatelsky přívětivé a cenově dostupné či adekvátní k danému produktu. Jelikož podle článku [3] má chytrý telefon každý pátý člověk na planetě, bude se tato práce zaměřovat na ovládání videokonferenčních produktů za pomoci mobilního zařízení.

Tato práce je tvořena ve spolupráci se společností Alef Nula a.s., kde toto zadání vzniklo. Zadání vychází z projektu, který má tři části:

1. Část
    - Analýza možnosti vzdáleného ovládání
    - Vytvoření knihoven pro vzdálené ovládání
    - Aplikace na platformě Android se základními funkcemi

2. Část
    - Plnohodnotná náhrada dotykového panelu na platformě Android
3. Část
    - Aplikace na platformě iOS

Zadání této práce bylo koncipováno tak, aby co nejvíce pokrylo první část tohoto projektu.

Kapitola 1

Cíle práce

1.1 Vymezení cílů

Cílem práce je navrhnout a implementovat aplikaci, pomocí které bude možné ovládat videokonferenční zařízení od společnosti Cisco z mobilního zařízení. V našem případě bude využito zařízení na platformě Android. Cílem je, aby bylo možné aplikaci poskytovat formou služby s možností spravovat oprávnění k ovládání jednotlivých videokonferenčních zařízení přes webové rozhraní. Dalším cílem je, aby bylo možné ovládat videokonferenční zařízení, která jsou umístěna v privátních sítích (v sítích s NAT – Network Address Translation).

Důvodů pro vznik této práce je několik. Tím hlavním je především vytvoření finančně efektivnějšího řešení, než v současnosti dodává společnost Cisco. Dalším důvodem je pak možnost bezdrátového připojení k videokonferenčním zařízením, či úprava aplikace na přání zákazníka.

1.2 Vazba cílů na části práce

Tato práce se dělí na čtyři části. První část práce se zabývá analýzou možností vzdáleného ovládání videokonferenčních zařízení. Specifikuje seznam zařízení k ovládání, předkládá jejich stručný popis a současné možnosti ovládání nejen od společnosti Cisco, ale i od jiných společností. Práce také hodnotí přítomnost aplikačních rozhraní a porovnává jejich možnosti ve vztahu k ovládání videokonferenčních zařízení s ohledem na použití v mobilních zařízeních.

Další část práce se věnuje návrhu aplikace pro ovládání videokonferenčních zařízení. Je zde vytvořen návrh architektury a zdůvodněny všechny jeho aspekty, také je tu návrh uživatelského prostředí nebo odůvodnění volby technologií.

Ve třetí části je pak popsána samotná realizace implementace. Jednotlivé kapitoly rozebírají detaily jednotlivých komponent a zároveň komentují použití knihoven.

Čtvrtá část se věnuje testování vzniklé aplikace. Je zde jak testování systému jako celku, tak testy jednotlivých komponent.

Kapitola 2

Analýza a návrh

Na úvod této kapitoly bude zmíněn nejdříve seznam zařízení, pro která si tato práce klade za cíl vytvořit ovládání. Dále zde budou popsány jednotlivé možnosti ovládání a zhodnoceny již existující řešení pro ovládání z mobilních zařízení. Na konci kapitoly je pak analýza možností vzdáleného ovládání a seznam funkčních požadavků.

2.1 Popis zařízení k ovládání.

Než postoupíme k dalším částem práce, je potřeba specifikovat zařízení [4], pro která si tato práce klade za cíl vytvořit aplikaci pro jejich ovládání. Tato zařízení se dají rozdělit do čtyř skupin. Následující kapitola rozřazuje zařízení do těchto skupin a popisuje jak jednotlivá zařízení, tak i dodávané nástroje k jejich ovládání. Popis jednotlivých nástrojů k ovládání je v kapitole (2.2).

2.1.1 Collaboration Desk Endpoints

Do této skupiny spadají produkty Cisco DX80, DX70 a produkty Cisco EX60, EX90. Jsou to osobní videokonferenční jednotky, které jsou se svými kompaktními rozměry navrženy na pracovní stůl a mohou sloužit i jako externí monitor. Jednotky Cisco EX60, EX90 je nutné ovládat dotykovým panelem Cisco Touch 8. Jednotky Cisco DX80, DX70 mají dotykový display, kterým se ovládají.

2.1.2 Collaboration Room Endpoints

Do této skupiny spadají produkty Cisco Telepresence MX200, MX300, MX700 a MX800. Tyto produkty jsou určeny pro konferenční místnosti, kde se používají nejen pro video hovory, ale i pro prezentaci ze zařízení připojených přes HDMI. Tyto produkty se ovládají přes přímo připojený dotykový panel Cisco Touch 10 nebo volitelně pomocí dálkového ovládání.

2.1.3 Immersive TelePresence

Do této skupiny spadají produkty Cisco TelePresence IX5000 a TX9000 Series. Tyto videokonference jsou určené pro skupinovou spolupráci v zasedacích místnostech. Díky více zobrazovacím plochám je možné sledovat několik účastníků v plném rozlišení, nebo plochy využít k prezentování z zařízení připojených přes HDMI. Tyto produkty se ovládají přes přímo připojený dotykový panel Cisco Touch 10.

2.1.4 TelePresence Integration Solutions

Do této skupiny spadají produkty Cisco TelePresence SX10, SX20 a SX80. Jsou to modulární videokonferenční zařízení určená pro skupinovou spolupráci v zasedacích místnostech. Je možné je připojit k libovolnému zobrazovacímu zařízení s HDMI. K zařízením této řady se standardně dodává dálkové ovládání "Cisco TelePresence Remote Control", volitelně je pak možné dokoupit "Cisco TelePresence Touch 8"nebo "Cisco TelePresence Touch 10". Produkty této řady jsou cenově nejdostupnější a také nejprodávanější, a to i přes velkou obchodní nevýhodu, kterou je cena atraktivního dotykového panelu na ovládání. Ta činí 3000 USD v ceníku GPL společnosti Cisco.

2.2 Popis nástrojů k ovládání

Tato kapitola uvádí popis nástrojů k ovládání, které dodává společnost Cisco.

2.2.1 Cisco TelePresence Remote Control

V této práci bude používaný zkrácený název pro toto zařízení a to "Remote Control". Remote Control je dálkové ovládání, které s daným zařízením komunikuje přes infračervený port. Poskytuje základní funkcionalitu, jako je například vytočení hovoru, příjetí hovoru, ovládání hlasitosti, ovládání kamery, volba zdroje k prezentování. Jeho výhody a nevýhody jsou sepsány v následující tabulce 2.1. Vizuální podoba Remote Control je na obrázku 2.1

Tabulka 2.1: Výhody a nevýhody Remote Control

výhody	nevýhody
pořizovací cena	nízká uživatelská přívětivost
bezdrátové připojení	absence klávesnice (pouze číselník)

2.2.2 Cisco TelePresence Touch 10

V této práci bude pro toto zařízení používaný zkrácený název "Touch 10". Touch 10 je ovládací panel s 10ti palcovým dotykovým displayem, který je

Obrázek 2.1: dálkové ovládání - Cisco TelePresence Remote Control

připojen k zařízení pomocí kabelu. Je to intuitivní zařízení, které slouží jako kvalitní nástroj nejen pro uživatelské ovládání, ale také pro konfigurační nastavení dané jednotky. Navíc oproti Remote Control poskytuje například možnost volby layoutu při prezentování či konferenčních hovorech. Jeho výhody a nevýhody jsou sepsány v následující tabulce 2.2. Vizuální podoba Touch 10 je na obrázku obrázku 2.2

Tabulka 2.2: Výhody a nevýhody Touch 10

výhody	nevýhody
uživatelská přívětivost a intuitivnost	vysoká cena
dotyková klávesnice	nutnost drátového připojení
rychlá odezva	

Obrázek 2.2: Cisco TelePresence Touch 10

2.2.3 Cisco TelePresence Touch 8

V této práci bude používaný zkrácený název pro toto zařízení a to "Touch 8". Touch 8 je ovládací panel s 8 palcovým dotykovým displayem, který je připojen k zařízení pomocí kabelu. Poskytuje stejnou funkčnost jako Touch 10. Vizuální podoba Touch 8 je na obrázku obrázku 2.3.

Obrázek 2.3: Cisco TelePresence Touch 8

Tabulka 2.3: Kompletní seznam zařízení a jejich ovládání

Zařízení	Dodávané ovládání	možnost Touch 8/10
Cisco Telepresence DX70	žadné	ano
Cisco Telepresence DX80	žádné	ano
Cisco Telepresence EX60	Touch 8	ano
Cisco Telepresence EX90	Touch 8	ano
Cisco Telepresence MX200	Touch 10	ano
Cisco Telepresence MX300	Touch 10	ano
Cisco Telepresence MX700	Touch 10	ano
Cisco Telepresence MX800	Touch 10	ano
Cisco TelePresence IX5000	Touch 10	ano
Cisco TelePresence TX9000	Touch 10	ano
Cisco TelePresence SX10	Remote Control	ne
Cisco TelePresence SX20	Remote Control	ano
Cisco TelePresence SX80	Remote Control	ano

2.3 Srovnání existujících řešení

2.3.1 vControl

Produkt vControl je vyvíjen společností VYOPTA. Jedná se o mobilní aplikaci na platformě Apple iOS. Zásadním omezením je fakt, že aplikace je určena pouze pro tablet. Aplikace je schopna připojit se ke všem požadovaným zařízením, která jsou uvedena v tabulce 2.3. Na tabletu představuje funkční náhradu za ovládací panely Touch 8 a Touch 10 v uživatelské sekci, i částečnou náhradu za sekci konfigurační. Ukázka aplikace vControl je na obrázku 2.4. Nevýhodou této aplikace je, že není podporována na mobilních zařízeních. Na dotazy, zda společnost plánuje podporu mobilních telefonů, odpověděla VYOPTA, že není v plánu. Stejně tak odpověděla, že není v plánu aplikace na platformě Android.

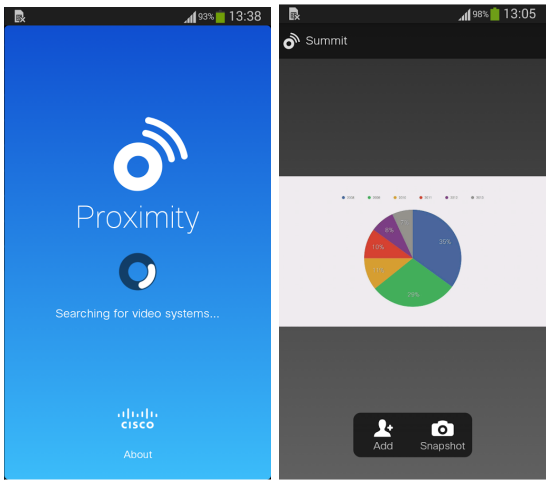
Obrázek 2.4: VYOPTA - vControl

2.3.2 Cisco Intelligent proximity

Jedná se o aplikaci [5] vyvinutou společností Cisco, která je dostupná jak na platformě Apple iOS, tak na platformě Android. Podporovanými zařízeními jsou mobilní telefon i tablet. Tato aplikace je na obou platformách zdarma. Cílem této aplikace není sloužit jako náhrada panelů Touch 8 a Touch 10. Aplikace sice umí vytočit a přijmout hovor, ale už nezvládne nastavení rozložení videa a prezentování, stejně jako ovládání kamery nebo nastavení hlasitosti. V aplikaci není možné přejít do konfiguračního módu a provádět jakákoliv nastavení zařízení, ke kterému je připojena. Účelem aplikace je sloužit jako doplněk pro prezentování a sdílení obsahu. Je možné zobrazovat prezentaci z mobilního zařízení, stejně jako zobrazit prezentaci probíhající na videokonferenci s možností prohlédnout si již odprezentované snímky. Aplikace také dává možnost pořídit snímek obrazovky v libovolný okamžik prezentace. Párování aplikace a videokonference probíhá pomocí ultrazvuku, za pomoci kterého si telefon a videokonference vymění informace potřebné pro síťové propojení. Podmínkou tohoto propojení je však, že mobilní zařízení a videokonferenční zařízení musí být připojena do stejné sítě. Detaily tohoto propojení bohužel nejsou zdokumentovány a neexistuje k němu žádné aplikační rozhraní. V současné době (20. 2. 2015) je aplikace podporována pro produkty Cisco Telepresence SX20, SX80, MX200 G2, MX300 G2, MX700 and MX800.

2.3.3 Závěr

Aplikace, které umožňují lidem ovládat videokonferenční zařízení pomocí mobilních zařízení, již existují. Nicméně ani jedna z nich neumožňuje kompletní ovládání veškeré funkčnosti přes všechny platformy.



Obrázek 2.5: Cisco Inteligent proximity

2.4 Analýza možností vzdáleného ovládání

Bylo zjištěno, že společnost Cisco vytvořila ke každému zařízení aplikační rozhraní, anglicky Application Programming Interface (API). Toto rozhraní bude dále nazýváno Cisco API.

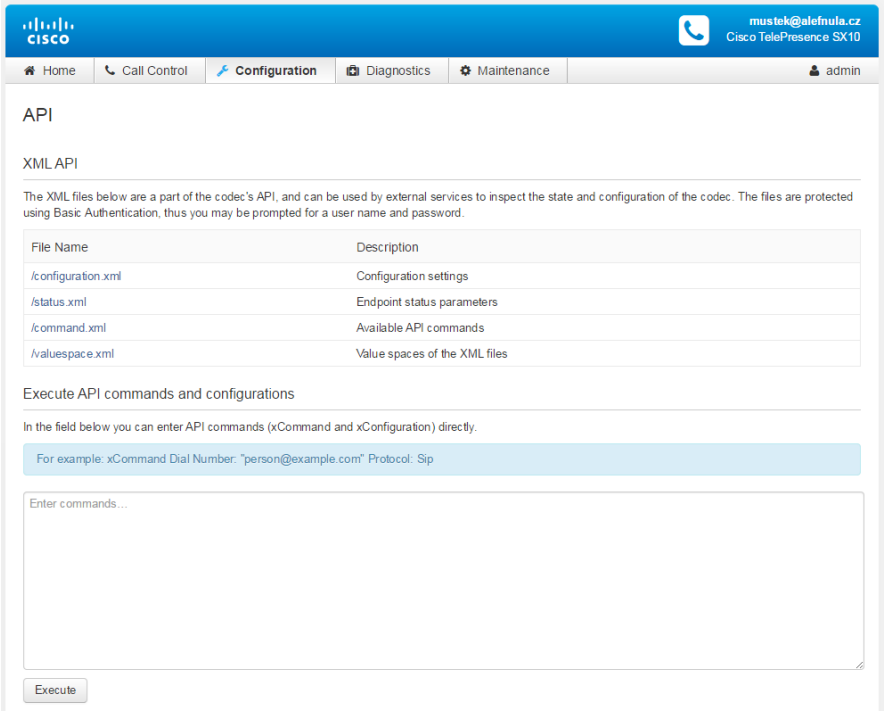
2.4.1 Cisco API

Toto rozhraní [6] je rozděleno do 4 částí: Commands, Configurations, Status a FeedbackRegistration, které dohromady poskytují komplexní možnost ovládat jednotlivá konferenční zařízení. Příkazy z části Commands dovolují provádět akce, jako je například vytočení čísla nebo zobrazení telefonního seznamu. Příkazy z části Configurations umožňují provádět konfigurační nastavení připojeného zřízení. Část Status poskytuje pohled na současný stav připojeného zařízení. Například je možné zjistit, zda na zařízení probíhá hovor. FeedbackRegistration je specifickou částí API, dovoluje uživateli registrovat se jako příjemce událostí vzniklých na připojeném zřízení (viz kapitolu 2.4.1.3).

Na administračním webu jednotlivých videokonferenčních zařízení je navíc jednoduché testovací prostředí pro toto rozhraní.

2.4.1.1 Komunikace

Cisco API poskytuje několik druhů připojení, z nichž 2 jsou určena pro vzdálenou komunikaci. Jedná se o Telnet/SSH a HTTP/HTTPS. Pro komunikaci pomocí kteréhokoliv z uvedených protokolů je potřeba znát IP adresu zařízení, uživatelské jméno a heslo. Pro účely naší aplikace nejvíce vyhovuje komunikace přes protokol HTTP/HTTPS, protože tento je snadno implementovatelný a použitelný v mobilních zařízeních.



Obrázek 2.6: Testovací prostředí pro zařízeni SX10

2.4.1.2 Formát vstupu a výstupu Cisco API

Cisco API podporuje dva formáty. První z nich je textový mód, kde je syntaxe tvořena jednořádkovým seznamem příkazů a jejich parametrů oddělených mezerami. Výstup je pak strukturován do řádkové syntaxe. Druhým formátem je XML, kde jsou jednotlivé příkazy skládány jako elementy a parametry příkazů jsou obsahem elementů. Výstup je stejně jako vstup strukturován do xml souboru.

Výhodou xml formátu je, že díky jeho strukturovanosti lze lépe programově zpracovat výstupy aplikačního rozhraní, stejně jako generovat příkazy pro vstup. Pro práci se xml soubory existují knihovny, jejichž použití bude komentováno v implementační části této práce.

2.4.1.3 Odběr událostí

Při jakékoliv změně stavu videokonferenčního zařízení vzniká takzvaná událost. Například při změně stavu z vyzvánění na spojený hovor vzniká událost CallSuccessful. Abychom se nemuseli stále dotazovat, zda se nezměnil například stav hovoru nebo nastavení hlasitosti, poskytuje Cisco API možnost odběru událostí. K tomuto odběru se lze přihlásit prostřednictvím registračního příkazu. V registračním příkazu je povinné uvést adresu HTTP serveru,

na který se budou události odesílat. Z toho ale samozřejmě vyplývá, že pro dané videokonferenční zařízení musí být povolen prostup k zařízení, jež bude události zpracovávat. Při registraci lze pomocí jazyka xpath definovat, které události chceme od zařízení odebírat. Každé videokonferenční zařízení umožňuje odesílat informace o událostech až na 4 adresy.

2.4.1.4 Zpozorované vlastnosti

I přes snahu společnosti Cisco mít API pro jednotlivá zařízení co nejpodobnější, jsou zde vidět malé odlišnosti. Tyto odlišnosti jsou dány převážně rozdílností funkcí samotných videokonferenčních zařízení. S těmito odlišnostmi budete nutné při použití API počítat. Dále bylo zjištěno, že jednotlivé možnosti API se liší i s různými verzemi firmware daných zařízení. S novějšími verzemi některé funkce přibývají, ale také ubývají. I s tímto bude třeba počítat.

2.5 Funkční požadavky

V této kapitole jsou shrnuty veškeré nároky a funkce, které jsou požadovány po aplikaci a systému jako celku.

Základním funkčním požadavkem je, že mobilní aplikaci musí být možné poskytnout formou služby (SaaS). Zároveň musí být možné poskytnout určité skupině uživatelů přístup k ovládání určité skupiny videokonferenčních zařízení. Je také nutné zajistit, aby bylo možné flexibilně přidávat a odebírat jak ovládaná zařízení, tak uživatele či jejich oprávnění. Jak vyplývá ze zadání, systém se dělí na dvě části - mobilní aplikaci pro ovládání zařízení a webové administrační rozhraní. Funkční požadavky jsou proto rozděleny do těch to dvou kategorií. V tuto chvíli je nutné definovat stěžejní pojmy.

Definice pojmů: Místnost, skupina, uživatel

Uživatel - Je definován uživatelským jménem a heslem, které ho opravňuje ke vstupu do mobilní aplikace.

Místnost - Je objekt, ve kterém se nachází právě jedno videokonferenční zařízení.

Skupina - Je logický kontejner, který může obsahovat uživatele a místnosti. V praxi si ho lze představit jako firmu, která má místnosti a uživatele.

2.5.1 Webové administrativní rozhraní

Toto rozhraní slouží k definici skupin, místností a uživatelů. Každá místnost obsahuje jedno videokonferenční zařízení, u kterého je nutné zadat všechny

potřebné údaje pro jeho vzdálenou obsluhu. Každému uživateli je možno přiřadit oprávnění k ovládaní jednotlivých místností (zařízení). Každá skupina obsahuje seznam místností a seznam uživatelů.

- Přihlášení do webového administračního rozhraní
- Odhlášení z webového administračního rozhraní
- Přidání a odebrání oprávnění uživateli
- Vytvoření, úprava a smazání místnosti
- Vytvoření, úprava a smazání skupiny
- Vytvoření, úprava a smazání uživatele
- Zobrazení seznamu místností
- Zobrazení seznamu skupin
- Zobrazení seznamu uživatelů
- Zobrazení seznamu uživatelů a místností ve skupinách

2.5.2 Mobilní aplikace

Aplikace slouží ke vzdálenému ovládání videokonferenčních zařízení. Po přihlášení k danému zařízení dovoluje uživateli zobrazovat a spravovat položky adresáře a obsluhovat hovory zařízení. Aplikace je schopna ovládat periferie daného zařízení, jako je například kamera nebo mikrofon.

2.5.2.1 Správa adresáře

- Zobrazení oblíbených kontaktů
  - Zobrazí všechny kontakty, které si uživatel přidal do oblíbených kontaktů.
- Vyhledávání v oblíbených kontaktů
  - Vyhledá v adresáři oblíbené kontakty a nabídne jejich vytočení.
- Vytočení hovoru z oblíbených kontaktů
- Přidání, upravení a smazání záznamu v oblíbených kontaktech.
  - Uživatel má možnost přidat si vlastní kontakt, případně kontakt upravit nebo smazat.

- Zobrazení adresářů
  - Jedná se o adresáře, jež má daná jednotka k dispozici od serveru, ke kterému je připojena. Tyto adresáře jsou definovány na straně serveru a z pohledu uživatele je nelze upravovat.
- Vyhledávání v adresářích.
  - Vyhledá kontakty ve všech dostupných adresářích a nabídne jejich vytočení.
- Vytočení hovoru z adresáře.
- Zobrazení historie hovorů
- Smazání historie hovorů
- Smazání jednoho záznamu z historie hovorů
- Vytočení hovoru z historie

2.5.2.2 Ovládání periferií

- Ovládání hlasitosti zvuku
- Ovládání citlivosti mikrofonu
- Ztlumení mikrofonu
- Ovládání kamery
  - Směrové ovládání kamery včetně možnosti přiblížení a oddálení.
- Volba zdroje obrazu
  - Uživatel si volí, který zdroj obrazu bude zobrazen na zobrazovacím zařízení. Zdrojem může být kamera nebo externí vstup (HDMI, DVI, případně jiný vstup dle modelu videokonferenčního zařízení).

2.5.2.3 Obsluha hovoru

- Vytočení hovoru
- Přepojení hovoru
- Přizvání účastníka do hovoru
- Přijetí hovoru
- Ukončení hovoru

- Sdílení obrazovky
- Volba layoutu
  - Možnost zvolit 1 z 5 možných rozložení videa a prezentace. (Single, Equal, Prominent, Overlay, Auto)

2.5.2.4 Základní funkčnost

- Nastavení vzdáleného serveru
  - V aplikaci je možné zvolit, s jakým serverem bude aplikace komunikovat.
- Přihlášení
  - Do aplikace se uživatel přihlásí jménem a heslem definovaným ve webovém administračním rozhraní.
- Výběr místnosti k ovládání
  - Uživatel si vybere z místností zobrazených aplikací.

2.6 Nefunkční požadavky

- Aplikace musí být schopna ovládat zařízení z libovolné počítačové sítě s přístupem na Internet.
- Minimální podporovaná verze Android systému je 4.0, což odpovídá API level 14.
- Uživatelské rozhraní bude optimalizované pro rozlišení od 640x480 do 1920x1080.
- Mobilní aplikace i webové administrační rozhraní je vyžadováno v anglickém jazyce.
- Mobilní zařízení nesmí znát udaje pro připojení k jednotkám.
- Přístup k ovládání musí být autentizován a autorizován.

2.7 Definice rolí

Pro účely navrhovaného systému definujeme pro webové administrační rozhraní a mobilní aplikaci 3 druhy uživatelských rolí.

2.7.1 Admin

Administrátor systému je role s nejvyšší úrovní oprávnění v systému, má přístup do webového administračního rozhraní a má právo:

- vytvářet, upravovat a mazat uživatele všech rolí,
- vytvářet, upravovat a mazat skupiny,
- vytvářet, upravovat a mazat místnosti,
- přiřazovat uživatele a místnosti do všech skupin.

Z bezpečnostních důvodů nemá tento uživatel možnost využívat mobilní aplikaci. Toto omezení má dva důvody. Při případném nešetrném zadávání nebo ztrátě telefonu by mohlo dojít k odcizení přihlašovacích údajů ze zařízení. Druhým důvodem je fakt, že by mohlo dojít k odchycení komunikace a získání tohoto uživatelského účtu.

2.7.2 Správce skupiny

Správce skupiny má přístup do webového administračního rozhraní, kde má právo:

- vytvářet, upravovat a mazat uživatele spadající do skupin, ke kterým má oprávnění,
- vytvářet, upravovat a mazat místnosti spadající do skupin, ke kterým má oprávnění,
- přiřazovat uživatele a místnosti do všech skupin, ke kterým má oprávnění.

Z bezpečnostních důvodů nemá tento uživatel možnost využívat mobilní aplikaci. Bezpečnostní důvody jsou zde stejné, jako v kapitole 2.7.1.

2.7.3 User

Má přístup do mobilní aplikace, kde může ovládat videokonferenční zařízení, ke kterým mu bylo přiděleno oprávnění. Nemá přístup do webového administračního rozhraní.

Kapitola 3

Návrh

Tato kapitola popisuje návrh řešení na základě funkčních a nefunkčních požadavků. Výběr vhodných komunikačních protokolů, programovacích jazyků a dalších technologií. Při návrhu zadání se vycházelo z poznatků a zkušeností získaných během analýzy.

3.1 Návrh architektury

Z funkčních a nefunkčních požadavků, ale i z jiných kapitol analýzy plynou různá omezení, která nelze přehlížet. Naopak je potřeba je důkladně rozebrat a při návrhu architektury zapracovat.

Funkční požadavky jsou rozděleny do dvou částí. V následující kapitole bude proveden nejdříve rozbor obecných požadavků, poté požadavků na ovládání videokonferenčních zařízení z mobilní aplikace, které jsou pro návrh architektury klíčové. Rozbor požadavků na webové rozhraní bude proveden až v další kapitole, kde bude komentováno jejich začlenění do celkové architektury.

3.1.1 Rozbor požadavků

Hlavním požadavkem je nutnost mít možnost řešení provozovat formou služby. Musí tedy existovat server, který bude schopen mobilní aplikaci poskytnout práva a informace k ovládání, a zároveň bude schopen tato práva i odebrat.

V analýze bylo zjištěno, že příkazy aplikačního rozhraní nejsou pro všechna zařízení stejná. Některé příkazy u některých zařízení nejsou vůbec k dispozici. To je dáno převážně tím, že zařízení se svými funkcemi liší. Tím pádem je potřeba k ovládání znát nejen jméno, heslo, adresu, ale i typ ovládaného zařízení.

Do budoucna je také důležité počítat se změnami v API, nejčastěji s přidáním nových funkcí. Proto není vhodné, aby knihovnu k ovládání zařízení

obsahovala přímo mobilní aplikace. Následkem by mohlo být časté aktualizování aplikace.

Jak bylo popsáno v kapitole o bezpečnosti, je nežádoucí, aby mobilní aplikace znala hesla a adresy videokonferenčního zařízení. Není tedy možné, aby videokonferenční zařízení ovládala mobilní aplikace přímo.

Dalším požadavkem je možnost ovládání videokonferenčních zařízení z libovolné počítačové sítě s přístupem na Internet. Je nutné počítat s tím, že mobilní aplikace na telefonu s přístupem na Internet nebude mít veřejně přístupnou IP adresu - například pokud je v síti s překladem adres (NAT). Zde je potřeba říci, že existuje více typů překladu adres, pro naši situaci je nejhorší Port Address Translation (PAT). V takovém případě by požadavky na ovládání byly schopné doputovat až k videokonferenčnímu zařízení, které by nebylo schopné odesílat informace o vzniklých událostech směrem k mobilní aplikaci.

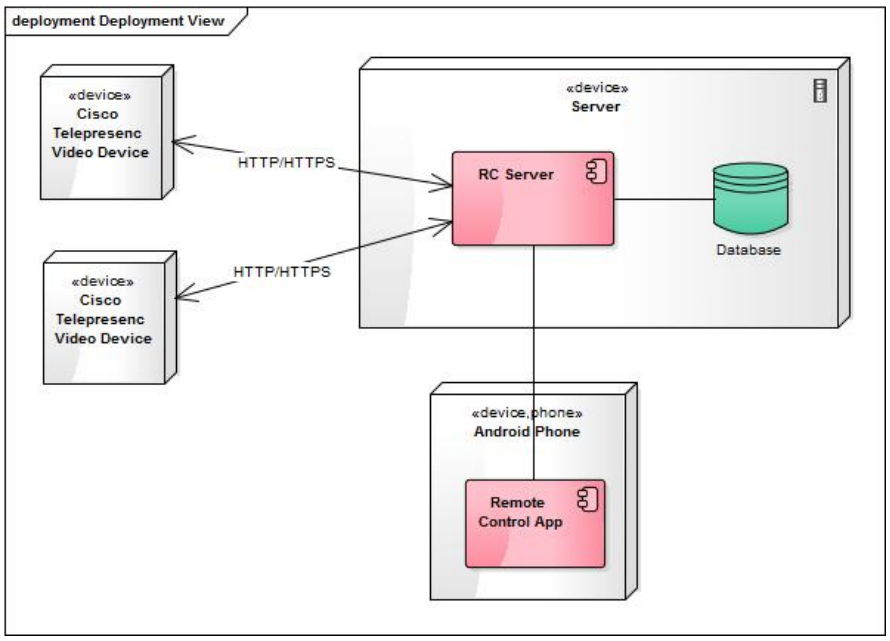
3.1.2 Závěr (výstup) z rozboru požadavků

Z rozboru požadavků vyplývá, že musí existovat server. Tento server byl pojmenován Remote Control server (RC server). RC server musí splňovat následující požadavky:

1. Server bude na veřejné IP adrese tak, aby bylo možné se k němu připojit mobilní aplikací odkudkoliv z Internetu.
2. Server bude mít přístup k potřebným informacím o videokonferenčních zařízeních k ovládání, uživatelských účtech a jejich oprávnění.
3. Server bude obsahovat knihovny k ovládání všech podporovaných typů zařízení.
4. Videokonferenční zařízení bude muset komunikovat se serverem protokolem HTTP.
5. Server bude muset komunikovat s videokonferenčním zařízením protokolem HTTP.

Řešení respektující tyto požadavky popisuje obrázek 3.1. Jak je vidět z obrázku, komunikační kanál mezi RC serverem a mobilní aplikací ještě není specifikován. Díky tomu lze říci, že požadavky 1 - 3 jsou splnitelné, ale u požadavků 4 a 5 je situace horší. Problém představuje především již definovaný protokol HTTP, který používají videokonferenční zařízení. Požadavek číslo 4 je splnitelný s podmínkou, že videokonferenční zařízení má přístup k Internetu, tedy je schopno komunikovat s RC serverem (zařízení nemusí mít vždy přístup k Internetu, například může sloužit jen pro hovor v lokální síti, nebo muže být připojeno v rámci lokální sítě k interní infrastruktuře a až tato infrastruktura může mít přístup k Internetu). Požadavek číslo 5 je splnitelný jen

v případě, že videokonferenční zařízení má veřejnou adresu, přesněji řečeno, je dostupné z veřejné adresy, tedy RC server je schopný s ním komunikovat.

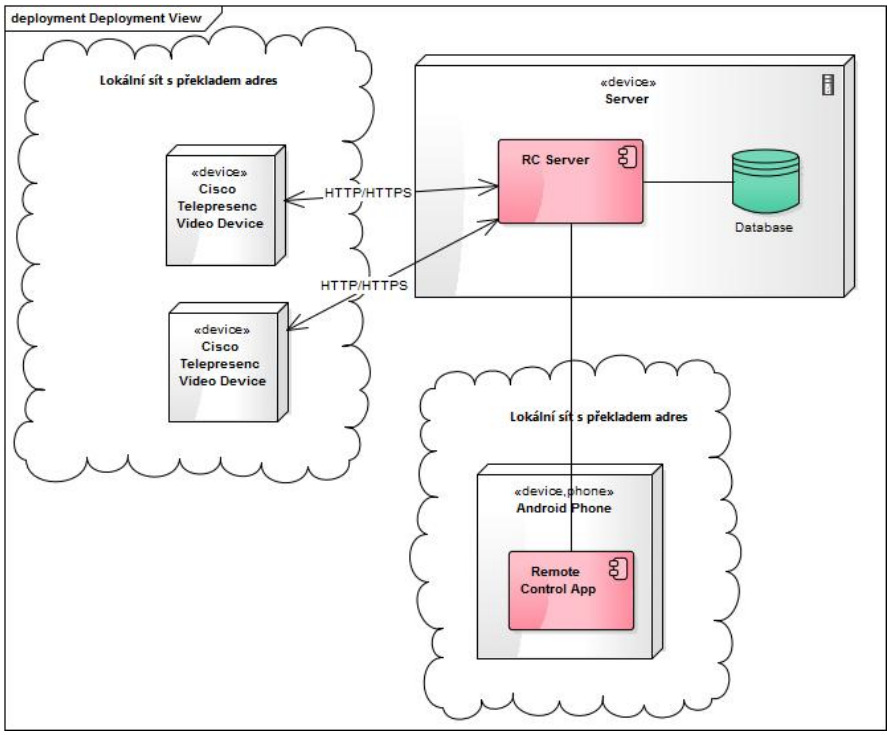


Obrázek 3.1: Model architektury - základní návrh

Splnění požadavku č. 4, tedy přístup zařízení k Internetu, je u všech podporovaných zařízení technicky možné. Ne vždy je to ale možné z bezpečnostních politik firem, které tato zařízení musí splňovat. Nicméně je to pořád reálné. U požadavku číslo 5 je situace složitější, a to hned ze dvou důvodů: tím prvním je opět bezpečnostní politika, tím druhým je nedostatek veřejných adres společnosti. IP adres je na světě omezené množství a velký počet zákazníků si nemůže dovolit přiřadit každému videokonferenčnímu zařízení veřejnou adresu. Nehledě na to, že to ani dělat nechtějí z bezpečnostních hledisek.

Pro lepší nastínění situace je zde obrázek 3.2, který ukazuje nejhorší možnou situaci, se kterou lze počítat. Z obrázku je vidět, že pokud budou videokonferenční zařízení umístěná v síti s překladem adres typu PAT, RC server nebude mít možnost s nimi komunikovat. Zároveň z obrázku plyne, že komunikace mezi RC serverem a mobilní aplikací nemůže být pomocí HTTP protokolu, protože by vznikal stejný problém. Blíže tento problém vysvětluje kapitola 3.2 Volba komunikačních kanálů.

Další část této kapitoly se tedy bude věnovat tomu, jak se vyhnout těmto dvěma problémům co nejefektivnější cestou. Řešení, které se nabízí jako první, je vytvoření programu, který bude umístěn mezi serverem a videokonferenčním zařízením a bude mít přístup jak do interní sítě, tak do Internetu a bude



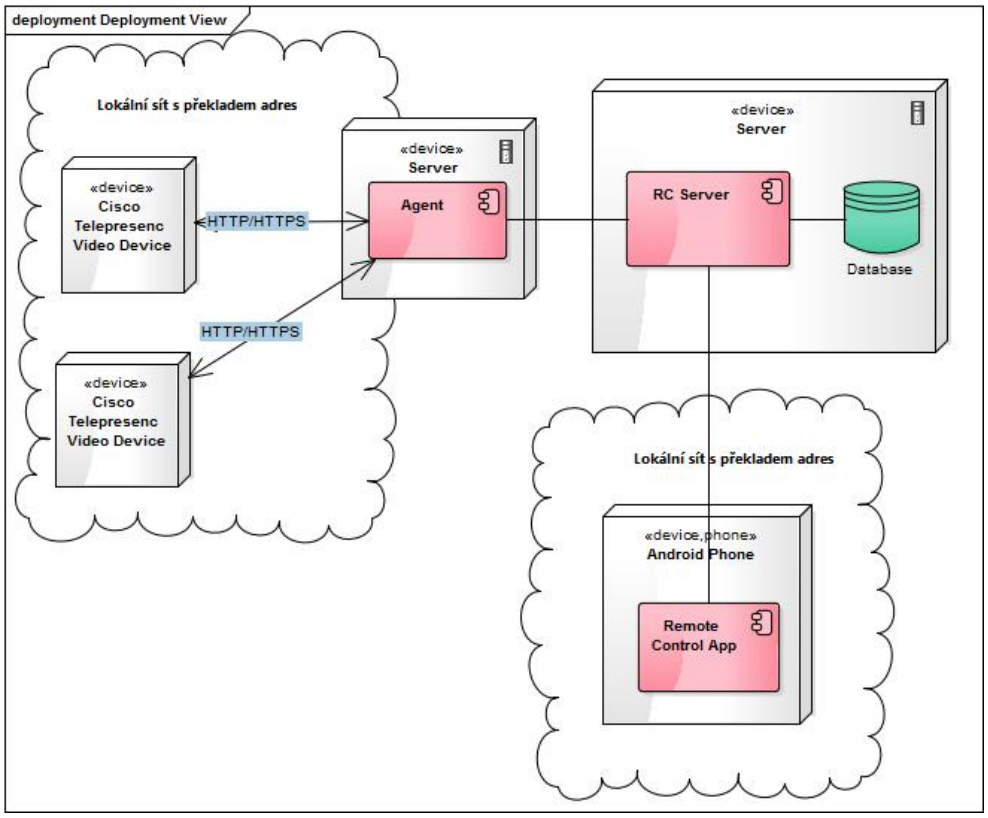
Obrázek 3.2: Model architektury - zařízení v sítích s NAT

na veřejné IP adrese. Tento program byl pojmenován Agent, jak znázorňuje obrázek 3.3. Tento návrh řeší problém komunikace s videokonferenčními zařízeními s neveřejnou adresou a bez přístupu k Internetu. Nicméně vzniká tak bezpečnostní riziko[1], které je pro většinu společností neakceptovatelný. Navíc toto řešení opět spotřebovává jednu veřejnou adresu.

Dalším způsobem, jak tyto problémy vyřešit, je umístit program Agent do sítě s překladem adres, jak znázorňuje obrázek 3.4. Tento program opět bude mít přístup jak do interní sítě, kde jsou připojená videokonferenční zařízení, tak do Internetu. Rozdílem je, že nebude umístěn na veřejné adrese, tudíž nebude možné se k němu připojit z venku. Aby mohl RC server ovládat zařízení, bude ale muset existovat spojení mezi Agentem a RC serverem. Jelikož Agent nemá veřejnou adresu, nemůže být inicializátorem spojení RC server, ale musí to být program Agent. Blíže tento problém vysvětluje kapitola 3.2 Volba komunikačních kanálů.

Tento způsob nevyžaduje žádnou veřejnou adresu a nevytváří prostup z Internetu do soukromé sítě. Je tedy akceptovatelný většinou společností.

[1] například IP spoofing



Obrázek 3.3: Model architektury - s programem Agent

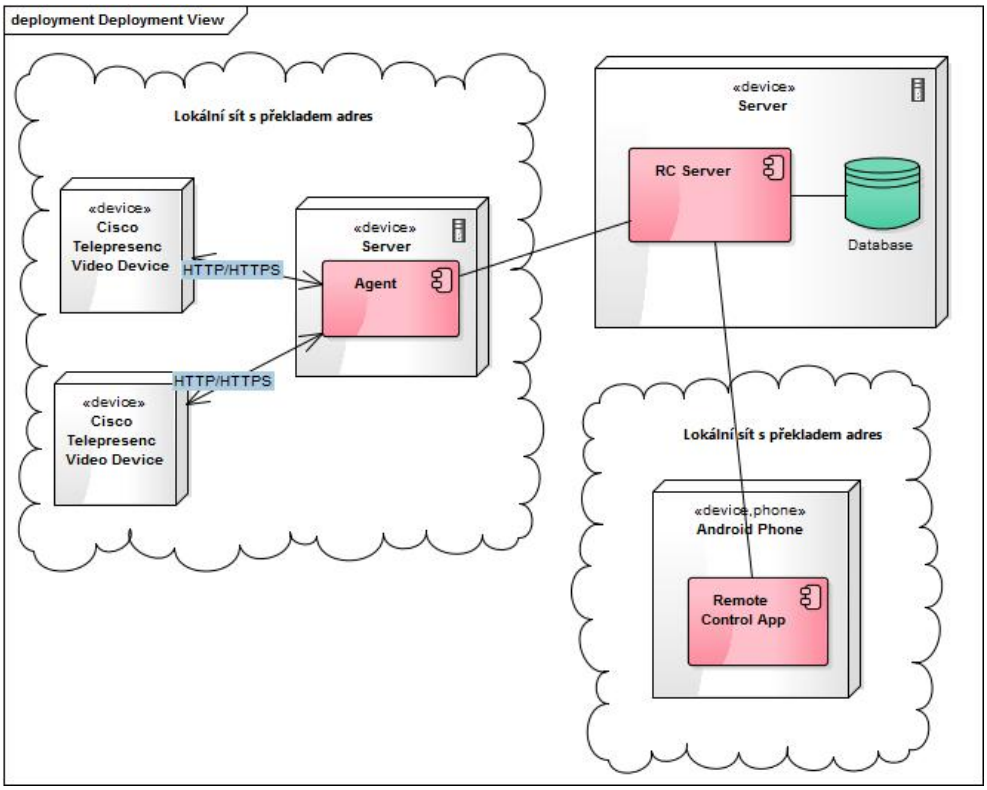
3.1.3 Rozbor požadavků na webové administrační rozhraní

Účelem webového administračního rozhraní je mít nástroj pro snadnou správu videokonferenčních zařízení, uživatelů a jejich oprávnění k ovládání. Webové administrační rozhraní musí být dostupné z Internetu, aby byla zajištěna možnost administrace odkudkoliv. Data bude webové administrační rozhraní ukládat do databáze. Je zde ale nutné, aby databáze byla přístupná i pro RC server, který má tuto potřebu ve svých požadavcích. Výsledný návrh architektury je tedy znázorněn na obrázku 3.5.

3.2 Volba komunikačních kanálů

Z předchozích kapitol již víme, které všechny komponenty bude systém obsahovat, a jaké jsou jejich klíčové vlastnosti a funkce. Co ale specifikováno nebylo, jsou jejich komunikační kanály (protokoly). Komunikace mezi videokonferenčními zařízeními a programem Agent je již dána. Zbývá specifikovat:

1. komunikaci mezi mobilní aplikací a RC serverem,



Obrázek 3.4: Model architektury - program Agent v sítích s NAT

2. komunikaci mezi programem Agent a RC serverem.

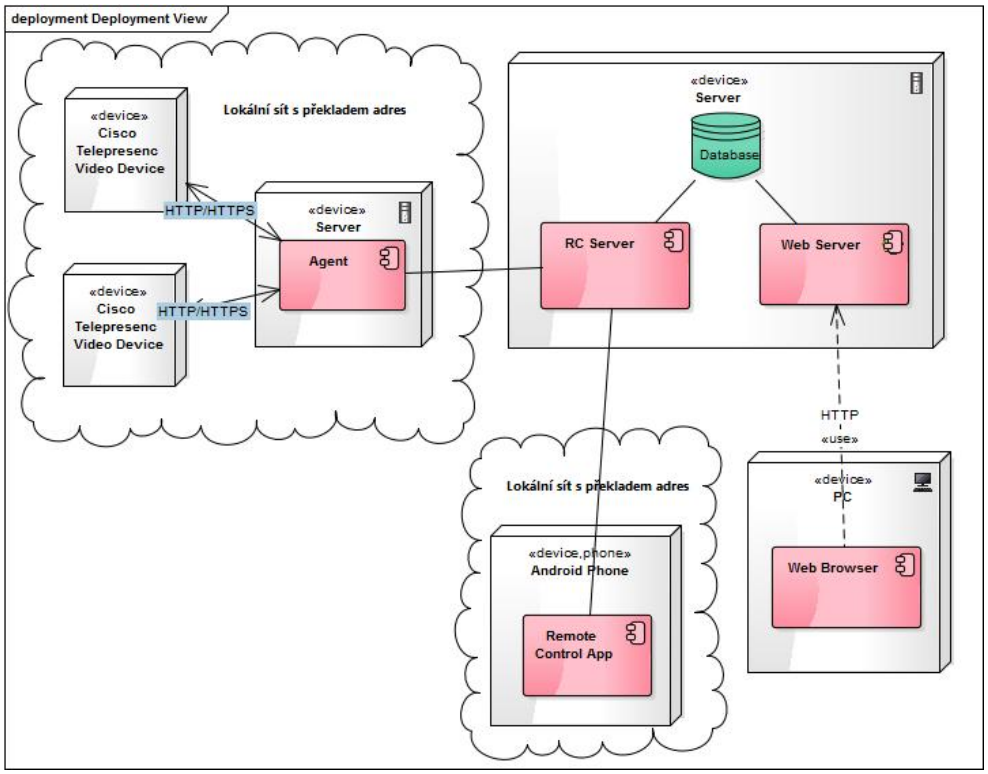
Výhody a nevýhody jednotlivých komunikačních kanálů budou posuzovány vzhledem k přínosům pro tento systém. Jak už z analýzy víme, oba klientské programy (mobilní aplikace a program Agent) se mohou nacházet v sítích s překladem adres typu PAT, nebo mohou být umístěny za firewallem. Tedy tyto zařízení nemusí být vždy možno kontaktovat z Internetu. Proto byly uvažovány pouze způsoby komunikace, které jsou schopné navázat spojení od klientské aplikace k RC serveru.

3.2.1 Komunikace mezi mobilní aplikací a RC serverem

Technologie RC serveru zatím nebyla určena, zatímco technologii mobilní aplikace známe. Jedná se o aplikaci pro operační systém Android, která je tvořena v programovacím jazyku Java.

Nalezené možnosti navazovaných spojení:

1. TCP socket
2. WebSocket



Obrázek 3.5: Model architektury včetně webového administračního rozhraní

3. Message queueing system (např. RabbitMQ)
4. Google Cloud Messaging for Android

3.2.1.1 TCP socket

Použitím TCP[7] mezi sebou mohou aplikace na počítačích připojených do počítačové sítě vytvořit spojení, přes které mohou obousměrně přenášet data. Protokol garantuje spolehlivé doručování paketů ve správném pořadí. Pro navázání spojení musí navazující aplikace znát cílovou adresu a port. Po navázání spojení mohou obě strany odesílat a přijímat data.

- výhody
  - V případě navázaného spojení je rychlost komunikace oběma směry vysoká (vyšší než u frontovacích systémů).
- nevýhody
  - TCP socket neposkytuje žádné způsoby autentizace a autorizace.
  - TCP socket neposkytuje nativní možnost šifrováni.

  - Není možno ovládat uspané mobilní zařízení.

3.2.1.2 WebSocket

Tento protokol[8] kombinuje výhody TCP spojení s výhodami HTTP. Poskytuje full-duplex komunikační kanál přes jedno TCP spojení. K navázání tohoto spojení je využit protokol HTTP.

- výhody
  - V případě navázaného spojení je rychlost komunikace oběma směry vysoká.
  - Je možné využití způsobů autentizace a autorizace protokolu HTTP/HTTPS.
  - Komunikaci lze zabezpečit pomocí SSL.
- nevýhody
  - Není možno ovládat uspané mobilní zařízení.

3.2.1.3 Message queueing system

Message queueing system[9] neboli v překladu frontovací systém umožňuje komunikaci aplikacím běžícím na různých zařízení v různý čas. Existuje jich celá řada, například: IronMQ, RabbitMQ, Microsoft Message Queuing, Amazon Simple Queue Service. Každá aplikace připojená k frontovacímu systému má možnost jak vložit zprávu do fronty, tak se dotázat, zda fronta neobsahuje zprávy pro připojenou aplikaci.

- výhody
  - Není nutno navazovat spojení mezi aplikací a serverem.
- nevýhody
  - Není možno ovládat uspané mobilní zařízení.
  - Dochází k častému dotazování na obsah, což způsobuje přenesení většího objemu dat.
  - Dochází ke zpoždění závislému na frekvenci dotazování.

3.2.1.4 Google Cloud Messaging for Android

Google Cloud Messaging (GCM) [10] pro Android je služba, která umožňuje posílat data z vašeho serveru do vašeho zařízení se systémem Android, a také přijímat zprávy ze zařízení. GCM služba se zabývá všemi aspekty frontování zpráv a jejich doručováním cílovým aplikacím běžícím na systému Android.

- výhody
  - Lze využít autentizaci a autorizaci pomocí tokenu.
  - Je možné zasílat zprávy uspanému mobilnímu zařízení.
- nevýhody
  - Je nutný 3-party server.
  - Není zde podpora pro zařízení s iOS. (Jak bylo řečeno v úvodu, v budoucnu se počítá i s aplikací pro iOS.)

3.2.1.5 Zvolená komunikace mobilní aplikace a RC serveru

Z výše uvedených metod komunikace a jejich hodnocení vyplývá, že pro komunikaci mezi mobilní aplikací a RC Serverem se nejvíce hodí buď WebSocket nebo GCM. Oba protokoly lze snadno zabezpečit a oba poskytují dostatečnou rychlost komunikace a odezvy. Výhodou GCM je, že dokáže zasílat zprávy i uspanému zřízení. To ale pro naši aplikaci není nutné. Výhodou WebSocketu je, že tento protokol se stal v roce 2011 standardem podle W3C [8] a i díky tomu existuje velké množství knihoven napříč různými programovacími jazyky, mezi nimiž je i knihovna pro iOS operační systém. A jelikož zadání této práce vychází z projektu, který má v plánu i aplikaci pro iOS, bude využit protokol WebSocket.

3.2.2 Komunikace mezi programem Agent a RC serverem

Jelikož se jedná o stejný druh spojený se stejnými požadavky jako v předchozím případě, je možné použít stejné komunikační kanály. Oproti předchozí kapitole je zde změna v tom, že ani jedna z komponent nemá ještě určenou technologii. To nám dává ještě větší volnost při výběru komunikačního kanálu.

Další možnosti komunikačních kanálů:

3.2.2.1 WCF s net.tcp bindings

Windows Communication Foundation (WCF)[11] od společnosti Microsoft je jednotný programovací model pro vytváření aplikací orientovaných na služby. Ten umožňuje vývojářům vytvářet bezpečné, spolehlivé a transakční řešení, které lze integrovat na různých platformách. Jedná se tedy o webovou službu, kde pro přenos dat muže být využito více protokolů, pro naše účely je nejlepší spojení nazývané jako net.tcp, kde je celý přenos zapouzdřen do protokolu TCP.

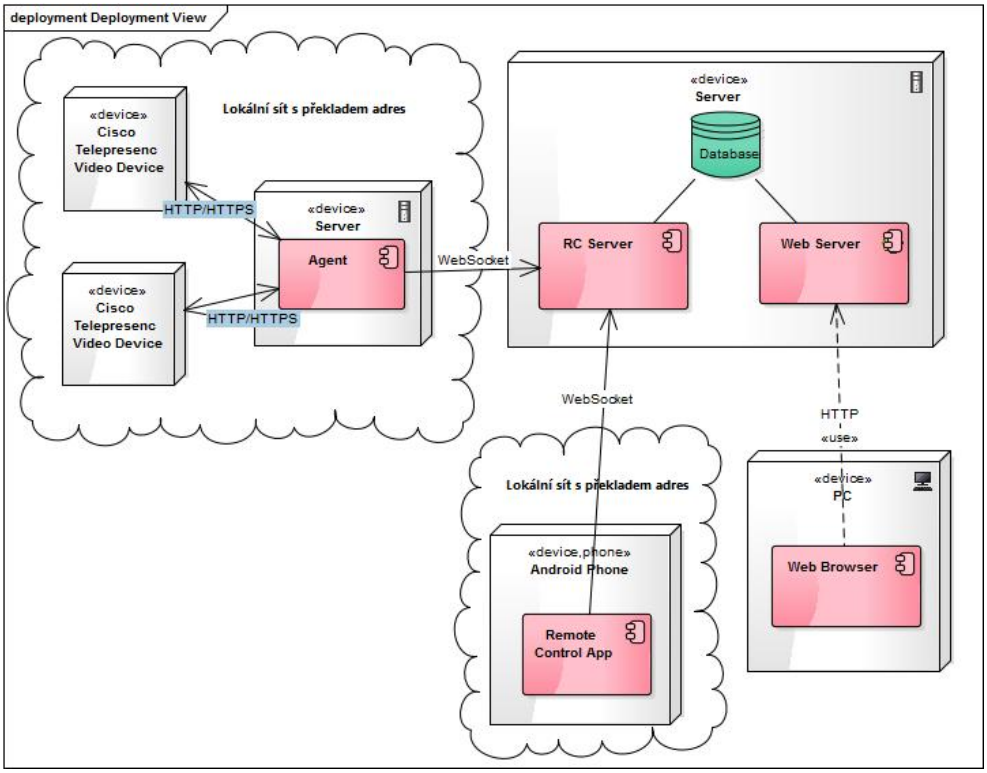
- výhody
  - Komunikace je rychlá oběma směry.

  - Je možné využití způsobů autentizace a autorizace, které poskytuje model WCF.
  - Zabezpečení komunikace metodami, které poskytuje model WCF například TLS.

- nevýhody
  - Není možno ovládat uspané mobilní zařízení.
  - Aplikace na obou stranách musí použít dotNet (.Net).

3.2.2.2 Zvolená komunikace pro program Agent a RC server

WCF s net.tcp bindings je vhodné a použitelné pro komunikaci mezi programem Agent a RC serverem. Nicméně je potřeba říci, že oproti protokolu WebSocket, který byl zvolen pro komunikaci mezi mobilní aplikací a RC serverem, nepřináší WCF s net.tcp žádné výrazné výhody. A proto byl kvůli jednotnosti zvolen protokol WebSocket. Stejné spojení u obou kanálu také ušetří čas při implementaci a sníží komplikovanost RC serveru. Návrh výsledné architektury je tedy na obrázku 3.6.



Obrázek 3.6: Výsledný model architektury

3.2.3 Autentizace

Díky použití protokolu WebSocket lze použít snadné šifrování samotné komunikace, to ale neřeší problém autentizace. Autentizační proces byl navržen podle článku [1] , který celý proces znázorňuje na obrázku 3.7. Je zde znázorněna komunikace mezi klientskou a serverovou aplikací. Autentizační proces začíná zadáním uživatelského jména a hesla v klientské aplikaci, tyto údaje jsou následně zaslány na server HTTP protokolem, metodou POST. Server zkontroluje údaje a v případě, že jsou údaje špatné, zašle odpověď o chybné autorizaci s kódem 401. V případě správných přihlašovacích údajů vygeneruje server token, který vloží do odpovědi na autentizační dotaz. Token bude umístěn v HTTP hlavičce pod klíčem token. Token má na straně serveru nastavenou expirační dobu.

Poté, co klientská aplikace obdrží token, zašle na server požadavek o upgrade na websocket protokol, který je posílán opět HTTP protokolem, metodou POST a do hlavičky vloží získaný token. Server přijímá tento požadavek, a pokud obsahuje validní token, který nevypršel, dává klientovi odpověď s kódem 101 (Switching Protocols). Po obdržení tohoto požadavku klientem je WebSocket spojení úspěšně navázáno.

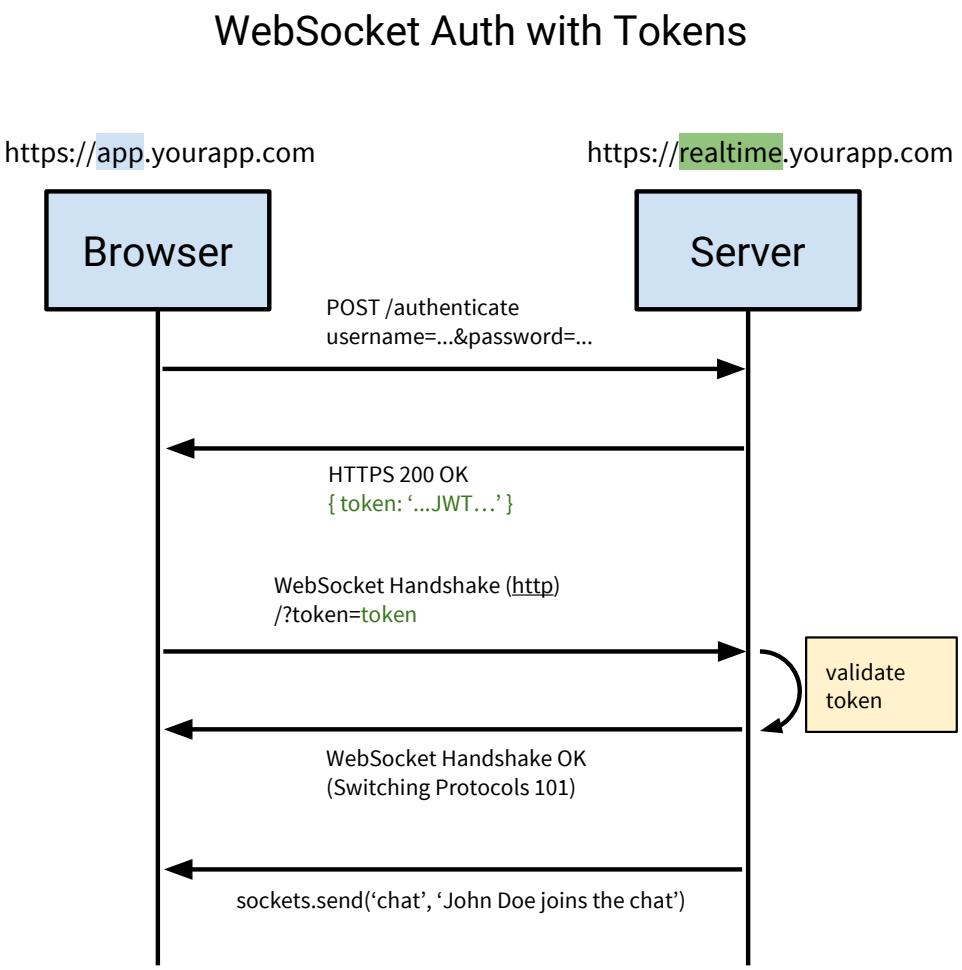
3.3 Datová vrstva

Už dříve některé kapitoly narazily na téma uložení dat. V této kapitole bude toto téma podrobněji rozebráno, bude diskutován typ datového úložiště a struktura uložených dat.

Než bude vybrán typ datového úložiště, je potřeba si říci, jaká data je potřeba ukládat, a které komponenty k nim potřebují přístup. Z návrhu architektury už víme, že datové úložiště budou využívat dvě komponenty, webové administrační rozhraní a RC server. Pro RC server bude datové úložiště pouze zdrojem dat, nebude tedy potřeba zápisu. Veškeré data budou zapisována pomocí webového administračního rozhraní.

Data, která je potřeba perzistentně uložit, lze ve zjednodušené formě shrnout jako:

- Informace o zařízeních neboli místnostech, které jsou nutné pro jejich ovládání.
  - Název
  - Popis
  - IP adresa
  - Jméno
  - Heslo



Obrázek 3.7: Sekvenční diagram komunikace pro autentizaci [1].

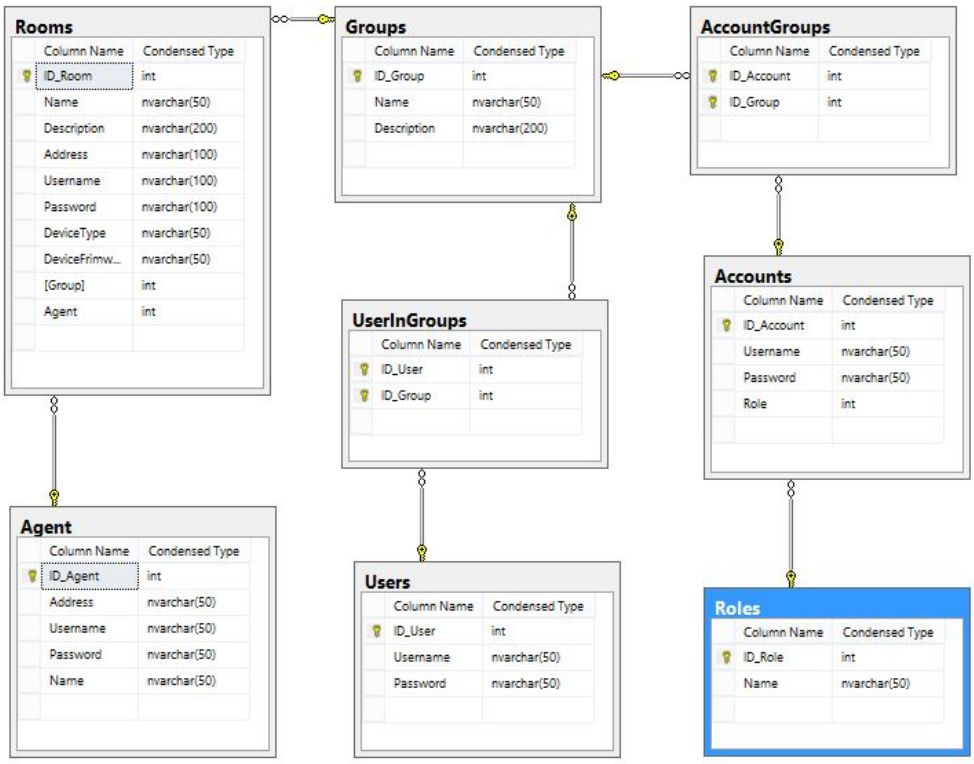
  - Typ zařízení
  - Device Firmware
- Informace o uživatelských účtech
  - Jméno
  - Heslo
  - Uživatelská role
- Informace o Skupinách a jejich obsahu
  - Jméno
  - Popis
  - Seznam uživatelů
  - Seznam místností
- Informace o Agentech
  - Jméno
  - Heslo
  - IP Adresa

Pro uložení těchto dat je vhodná relační databáze, do které lze data jednoduše serializovat, a která nám díky jazyku SQL umožní jednoduše se nad daty dotazovat. Databázových systémů existuje celá řada, například Oracle, SQL Server, MySQL, PostgreSQL a další. [12] K jejich hlavním funkcím mimo jiné dnes patří například zálohování nebo možnost optimalizace dotazování pomocí indexů. Relační databáze byla také preferována společností zadavatele.

3.3.1 Návrh struktury dat

Návrh struktury dat je znázorněn obrázkem 3.8, který zároveň ukazuje i vazby jednotlivých entit.

Návrh odráží strukturu dat, která je potřeba uložit včetně jejich propojení s ohledem na funkční požadavky. Obsahuje tedy tabulku `Rooms` se všemi potřebnými informacemi o místnostech včetně vazby na tabulku `Agents`, která nese data o tom, který agentský program má danou místnost obsluhovat. Tabulka `Groups`, neboli skupiny, slouží jako logický kontejner pro uživatelské účty a místnosti. Zajímavostí je existence tabulek `Users` a `Accounts`, kde v tabulce `Users` jsou uloženy účty pro přístup do mobilní aplikace a v tabulce `Accounts` jsou účty pro přístup do webového administračního rozhraní. Důvodem tohoto rozdělení je především fakt, že u účtů pro přístup do mobilní aplikace bude uloženo šifrované heslo, zatímco u účtů pro přístup do webového administračního rozhraní bude uložen hash daného hesla.



Obrázek 3.8: Návrh struktury dat.

3.4 Volba technologií

Z návrhu architektury víme, že projekt se bude skládat ze čtyř komponent - RC Server, Agent, Webové administrační rozhraní a Mobilní aplikace. Mobilní aplikace je pro operační systém Android, pro který jsou aplikace tvořeny v programovacím jazyku Java.

Jako programovací jazyk pro ostatní komponenty byl vybrán jazyk C# s .Net Frameworkem 4.5. Následující kapitoly popisují jeho výhody a odlišnosti, které vedly k jeho zvolení.

3.4.1 Jazyk LINQ

Jedna z hlavních výhod jazyka C# je Language Integrated Query (LINQ) [13, str. 31]. Rozsah tohoto jazyka je velice rozsáhlý, proto jen velice zkráceně. LINQ je součástí jazyka C# od jeho verze 3.0 a jeho účelem je dotazování nad daty:

- LINQ to objects - umožňuje dotazování nad kolekcemi nebo přesněji nad

objekty implementujícími rozhraní lEnumerable.

- LINQ to XML - poskytuje objektový pohled na XML dokumenty.
- LINQ to ... - dalších implementací je celá řada, ale v této práci nejsou použity. Patří mezi ně LINQ to SQL, LINQ to Active Directory, nebo LINQ to Amazon.

Ukázka použití technologie LINQ:

```
List<int> userIdlist = (from userData in
    ClientsUserData where userData.Value.RoomId == 1
    select userData.Value.Id).ToList();
```

Vyhledávání v kolekcích typu slovníku je, pokud se hledá podle klíče, jednoduché, jsou ale případy, kdy je třeba hledat podle hodnoty. V ukázce výše lze vidět dotaz, který projde všechny záznamy v kolekci a u objektu, kde se hodnota proměnné RoomId rovná 1, vrátí hodnotu proměnné Id daného objektu. Výsledek dotazu je potom převeden do kolekce List<int>.

Většina příkazů LINQ používá zpožděné vyhodnocování (deforred execution), narazí-li tedy program na strukturu LINQ, uloží si do paměti zástupný objekt a výraz vyhodnotí až při prvním přístupu k němu nebo při použití tzv. nondeferred operátoru, tedy příkazu bez zpožděného vyhodnocení (např. ToList()). Vyhodnocení je tak vždy aktuální a lze jej opakovat vícekrát:

```
List<string> names = new List<string> {"Adam",
    "Edward", "Bob", "Andrew"};
var query = (from name in names where
    name.StartsWith("A") select name);

// Vysledek: Adam, Andrew
List<string> aNames = query.ToList();
names.Add("Amy");

// Vysledek: Adam, Andrew, Amy
aNames = query.ToList();
```

3.4.2 ASP.NET WebForms

Tato technologie slouží pro tvorbu webových stránek a je součástí .Net Frameworku od verze 1.0. Tato technologie umožňuje programátorovi využívat serverové ovládací prvky, jako je například: Button, Label nebo Gridview. S těmito prvky lze pak snadno pracovat na pozadí každé stránky. Lze jim přiřazovat určité vlastnosti nebo na nich zachytávat události. Programátor se tak nemusí věnovat programování těchto funkcionalit.

3.4.3 Další výhody

Mezi další pro práci ale méně významné výhody [13, str. 31] patří například Properties s automatickými get a set metodami. Dále přetěžování operátorů (lze porovnávat objekty typu string operátorem ==) nebo partial classes.

3.4.4 Volba databáze

Jako databázový stroj byl zvolen MS SQL Server 2008, který byl požadavkem společnosti ALEF NULA a.s.

3.5 Návrh uživatelského prostředí

Dobrá aplikace musí být nejen dobře naprogramována, ať už z hlediska efektivity fungování nebo vnitřní architektury, ale musí být také dobře použitelná. Dobré uživatelské rozhraní umožňuje uživateli pracovat s aplikací účelně a pohodlně. Rozhraní, které navíc dodržuje předepsaná pravidla, pomáhá uživateli vytvářet správné stereotypy práce se systémem. Zvyšuje tak efektivnost uživatelů, kteří mohou uplatnit dobré návyky z ostatních aplikací. Proto se práce věnuje i tomuto tématu.

3.5.1 Task Graf

Na základě funkčních požadavků a případů užití byl vytvořen task graf, kde jsou jednotlivé případy užití rozčleněny do kategorií a naznačeny jednotlivé přechody mezi nimi. Tento task graf lze najít na obrázku 3.9.

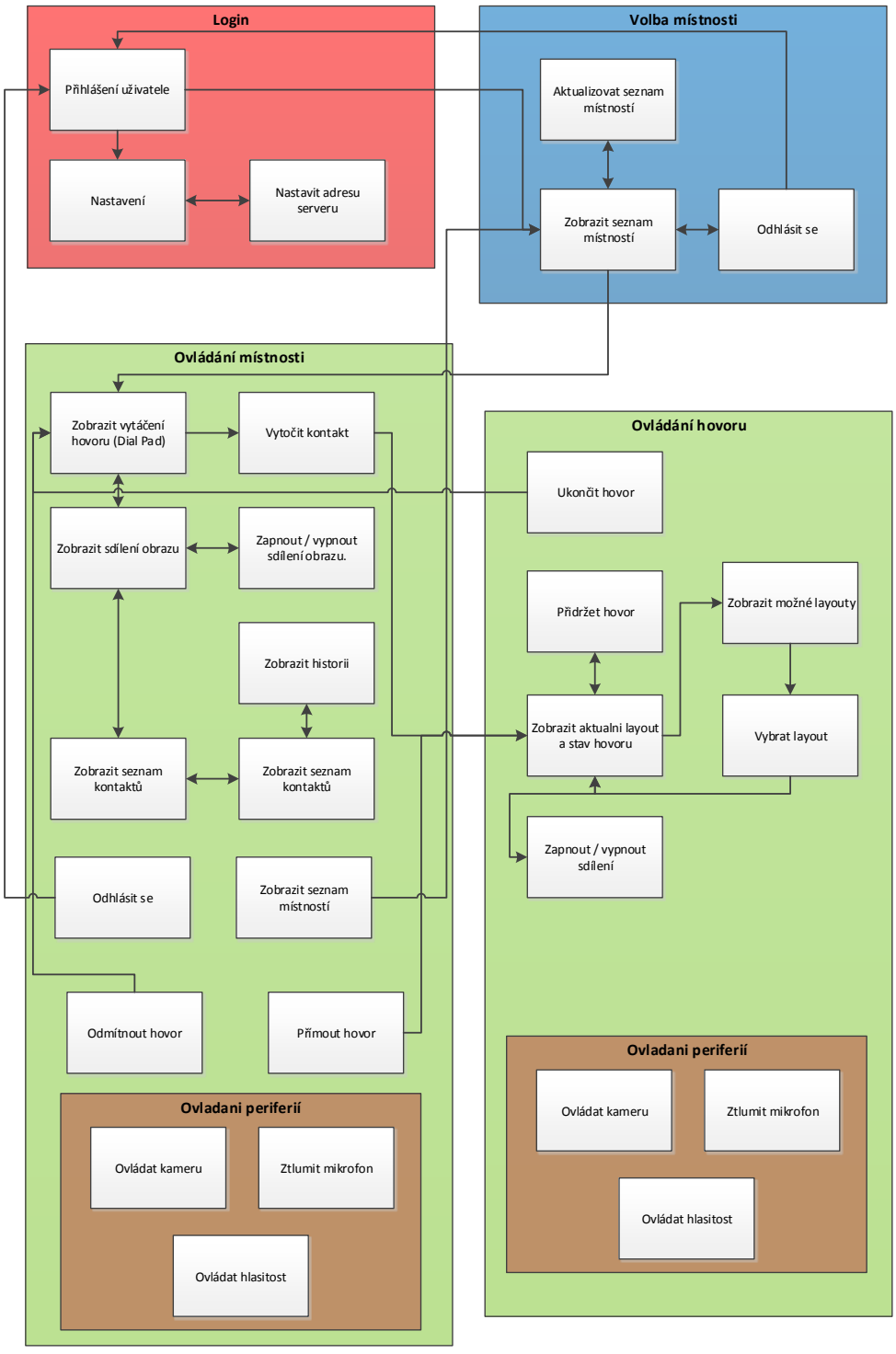
3.5.2 Low-fi prototype

Vytvořený task graf otvírá cestu k další části návrhu uživatelského rozhraní a tou je Low-fi prototyp [14] . Nejčastějším provedením tohoto prototypu je vytvoření wireframes, ať už v papírové podobě, nebo za pomoci nějakého software. Pro tuto práci byl použit software Justinmind Prototyper 6.4. Výhodou tohoto software je, že dovoluje používat prvky se shodnou grafikou jako systém android a díky tomu wireframes vypadají realističtěji a lze si aplikaci lépe představit.

Další výhodou je, že je zde možnost přiřazovat jednotlivým elementům akce, díky nimž lze realivat přechody mezi jednotlivými wireframes a aplikaci tak lépe testovat. Jelikož low-fi prototyp obsahuje 20 jednotlivých wireframes, je zde ukázána jen část a kompletní návrh je přiložen na CD.

3.5.3 Testování Low-fi prototype

Testování prováděli 3 testeři, každému testerovi byly postupně sdělovány instrukce, co má dělat. Například přihlas se do aplikace, vyber místnost, vytoč



Obrázek 3.9: Task graf pro mobilní aplikaci.

Obrázek 3.10: Wireframes ukázka

telefonní číslo, vyhledej kontakt Vojtěch Hlavačka a tento kontakt vytoč, sniž hlasitost a další. U každé akce bylo sledováno uživatelovo chování. Na konci každý tester sepsal své poznatky z testování.

3.5.3.1 Výsledek testování Low-fi prototype

- Tlačítko connect u výběru místností se ukázalo jako matoucí. Proto bylo úplně odstraněno a výběr místnosti je možné provést kliknutím přímo na položku.
- Testeři připomínkovali tlačítko nastavení, které není často používané. Proto bylo tlačítko přesunuto do kontextového menu.
- Ikona sdílení se ukázala jako nedostatečně vystihující, proto byla nahrazena.
- U sdíleni obrazu testeři připomínali, že není poznat, který zdroj obrazu je zvolen.

Ve výsledku se tedy koncept předvedl jako realizovatelný.

3.5.4 Testování Hi-fi prototyp

Na základě wireframes z předchozí kapitoly byla vytvořena prototypová aplikace pro operační systém android. Aplikaci opět testovali 3 testeři, kteří dostávali stejné úkoly jako při testování Low-fi prototypu.

3.5.4.1 Výsledek testování Hi-fi prototype

- Tlačítko s obrázkem klávesnice na obrazovce pro vytáčení se ukázalo jako nadbytečné, uživatelé při pokynu klikali přímo do textového pole, při němž se klávesnice rovnou zobrazí.
- Testeři připomínali, že dvojí ovládání hlasitosti je matoucí.
- Při spojeném hovoru se stávalo, že tester klikl na horní šipku zpět, která ho vyhodila pryč z hovoru. Proto byla šipka odstraněna.

3.5.5 Heuristická analýza

Heuristická analýza spočívá v detekci a identifikaci míst, kde by se uživatel mohl potýkat s problémy. Postupným procházením aplikace tvoří odborník dokument obsahující seznam chyb a problémů s použitelností. Odhalování chyb a slabých míst v rozhraní probíhá za pomocí porovnávání jejího současného stavu s pravidly (heuristikami), která jsou předem daná.

Nejznámějšími heuristickými pravidly je deset bodů použitelnosti podle dánského profesora informatiky Jakoba Nielsena [15] . Následně bude uvedeno všech 10 bodů a ke každému bodu bude komentováno hodnocení aplikace.

1. Viditelnost stavu systému (Visibility of system status) – Systém nesmí zůstat zamrzlý a nereagovat na uživatelské vstupy.
   - V pořádku.
2. Zachování konvencí (a metafor) reálného světa. Ikony (a metafory) se musí chovat jako to, co zobrazují / na co odkazují.
   - Ikony odpovídají funkcím, které dělají.
3. Minimální zodpovědnost (a stres). Vždy je možnost vrátit se zpět do předchozího stavu. Zrušení dlouho trvajících operací (rollback). Varování před provedením nevratné akce.
   - Díky tlačítku zpět platformy android je umožněno vždy se vrátit zpět.
   - Při přihlašování uživatele a při načítání seznamu místností nelze operace zrušit.
   - Při probíhajícím hovoru tlačítko zpět vrátí uživatele do ovládání místnosti bez upozornění. Pro opětovné zobrazení hovoru je potřeba znovu zvolit odpovídající místnost.
4. Shoda s použitou platformou a obecnými standardy. Pokud to jde, použít standardní systémové komponenty, systémové barvy a fonty.

- Jsou použity typické komponenty pro systém Android.

5. Prevence chyb. Uživatel by neměl mít možnost zadat špatnou hodnotu. Zvýraznit povinné položky formulářů.

   - Při vyplňování telefonního čísla nebo emailové adresy nejsou kontrolovány nevalidní hodnoty.

6. Kouknu a vidím. Nezatěžovat uživatelovu paměť. Akce, které uživatel může momentálně provést, by měly být viditelné a snadno dosažitelné. Nepotřebné kontrolky a informace nepotřebujeme.

   - Při snížení hlasitosti na nulovou hodnotu se zobrazené tlačítko nezmění a uživatel nemá informaci o ztlumené hlasitosti.

7. Flexibilita a efektivita. Klávesové zkratky/funkční klávesy. Jsou opravdu všechny akce/nastavení potřeba?

   - V pořádku.

8. Minimalita (Klapky na očích). Zobrazovat pouze informaci, která aktuálně k něčemu opravdu je. Čím méně možností uživatel má, tím rychleji koná

   - Všechny zobrazené volby umožňují nějakou akci.

9. Smysluplné chybové hlášky. Chybové hlášení v běžném jazyce – žádné kódy. Chybové hlášení by mělo popsat, co se stalo špatně, jak se to stalo a jak tomu příště předejít, připadně možná řešení doporučit.

   - Splňuje ve všech bodech.

10. Help a dokumentace. Systém by měl být použitelný bez jakékoliv nápovědy, nicméně nápověda musí být k dispozici.

    - Vzhledem k tomu že se jedná o prototyp aplikace, nebyla nápověda společností zadavatele vyžadována. Jako kvalitní technickou dokumentaci lze považovat tuto práci.

3.5.5.1 Závěr Heuristické analýzy

Největší nedostatky zjištěné heuristickou analýzou byly u 5. a 6. pravidla. Tyto nedostatky se povedlo v hi-fi prototypu opravit.

3.5.6 Závěr testování uživatelského rozhraní

Při testování byly zjištěny pouze dílčí nedostatky, které netvoří neřešitelnou překážku pro postup k vytvoření plně funkční aplikace.

Kapitola 4

Realizace

V této kapitole bude přiblížena implementační část práce. Jelikož je implementace celého systému rozsáhlá, budou zde komentovány pouze ty nejdůležitější a nejzajímavější části. Z návrhu už čtenář ví, že řešení se dělí na více komponent. Jednotlivé komponenty zde budou rozebrány a budou komentovány jejich funkce a případné použití knihoven.

4.1 TelePresencLibrary

Tato komponenta sice v návrhu nebyla nikde zmíněna, ale její vznik má svoji logiku. Jedná se o knihovnu pro ovládání videokonferenčních zařízení z řady Cisco TelePresence. Tato knihovna poskytuje dvě základní funkcionality. Tou první je sada metod, které generují XML příkazy pro ovládání. Druhou je realizace spojení s koncovým zařízením a sada metod, které dokáží zařízení ovládat a vrací informaci o výsledku ovládání.

Samotná knihovna má dvě třídy `TelePresence` a `TelePresenceXml`. Třída `TelePresenceXml` disponuje sadou statických metod pro generování XML příkazů. Třída `TelePresence` dědí od této třídy. Samotná třída `TelePresence` používá pro realizaci spojení s zařízeními knihovnu `WebClient`, která zapouzdřuje HTTP komunikaci. Na zpracování výstupu od zařízení byla použita knihovna `XmlDocument`, která kontroluje existenci elementu reprezentující úspěšnou operaci. Díky těmto knihovnám a třídě `TelePresenceXml` je pak schopná realizovat ovládání s koncovým zařízením.

4.2 Data Repository

Přístup k datům uloženým v databázi potřebují dvě komponenty, konkrétně RC server a webové administrační rozhraní. Obě tyto komponenty jsou implementovány v jazyce C#, proto by bylo vhodné, aby obě komponenty měly pro

přístup jednu společnou knihovnu, která bude poskytovat komplexní rozhraní pro práci s databází.

Pro přístup k datům byl použitý návrhový vzor repository [16] s objektově relačním mapováním. Jeho úkolem je mapovat relačně uložená data z databáze do objektů, jejichž struktura je uložená v namespacu `Data.Engine.Model`. Samotný návrhový vzor repository pak definuje pro každou sekci dat Interface, který definuje možné operace s daty. V našem případě se jedná o následujících 5 interfaců:

```
interface IAccountRepository
interface IUserRepository
interface IRoomRepository
interface IGroupRepository
interface IAgentRepository
```

Pro každý interface pak mohou být vytvořeny separátní implementace nad různými datovými úložišti. V našem případě byla vytvořena konkrétní implementace pro databázi běžící na SQL server 2008. Každá z těchto implementací dědí od třídy `BaseRepository`, která poskytuje možnost připojení k databázi. Toto připojení je pak schopná poskytovat jednotlivým metodám daných repositářů.

Přístupy k těmto implementacím vytváří třída `RepositoryHandler`, která sama využívá návrhový vzor singleton [17].

4.3 Remote Control Server

Jedná se o komponentu umístěnou v síti Internetu s přístupem do databáze, která je klíčová pro připojení mobilní aplikace a programu agent. Dá se tedy říci, že je to hlavní část celého systému.

Jak bylo řečeno, jeho úkolem je komunikace s programem agent a mobilní aplikací. Z návrhu víme, že obě komunikace jsou navazované spojení ze strany klientských aplikací směrem k RC serveru. Server tedy poslouchá na dvou adresách - jedné pro agenty a jedné pro mobilní aplikace. Při příchozím požadavku zkontroluje, zda je požadavek vybaven potřebnými atributy pro autorizaci a pokud ano, dovolí navázat spojení s klientskou aplikací. Podrobně byl proces autentizace popsán v kapitole 3.2.3. Po navázání spojení si server v paměti drží potřebná data o klientech. Jaká data si server drží a jak probíhá další komunikace, bude rozepsáno v kapitolách pro jednotlivé komunikace.

4.3.1 Komunikace s Agentem

Bezprostředně po navázání spojení s programem Agent zjistí server, zda jsou v databázi uložena nějaká zařízení, která by daný agent mohl ovládat a od kterých by mohl odebírat události. Pokud nějaká taková zařízení existují, seri-

alizuje data o zařízeních do XML a obratem je zašle agentovi. Dále pak server vyčkává na další zprávy od Agenta.

4.3.1.1 Druhy zpráv mezi Agentem a RC serverem

RC Server je schopen zpracovat 4 druhy zpráv od agenta:

Hello Touto zprávou dává agent na vědomí, že je zapnut. A jako odpověď očekává opět zprávu Hello. Důvodem této komunikace je, aby Navázaný TCP kanál nebyl přerušen z důvodu delší nečinnosti v tomto kanálu. Interval posílání této zprávy je nastavitelný v Agentovi. Základní nastavení je 1 minuta.

Rooms Touto zprávou Agent žádá o zaslání informací o místnostech neboli zařízeních, které může ovládat a od kterých může odebírat události.

RoomConnected Touto zprávou informuje agent, že úspěšně navázal spojení s koncovým zařízením a je toto zařízení schopen ovládat.

Event Touto zprávou Agent zasílá informaci o vzniklé události na koncovém zařízení.

Listing 4.1: Ukázka zprávy Event

```
<?xml version="1.0" encoding="utf-8" ?>
<Event>
  <RoomId>9</RoomId>
  <IncomingCallIndication item="1">
    <RemoteURI item="1">
       sip:vojtech.hlavacka@4video.cz </RemoteURI>
    <DisplayNameValue item="1"> Vojtech Hlavacka
       </DisplayNameValue>
    <CallId item="1">8</CallId>
  </IncomingCallIndication>
</Event>
```

Všechny zprávy RC server očekává ve formátu XML. Strukturu těchto zpráv definují XSD soubory, které jsou přiloženy s implementací na CD.

Při příchozí zprávě server nejdříve zjistí, o kterou zprávu se jedná a zda je obsah zprávy validní. Pokud ne, odpovídá zprávou `Error`, pokud ano, zachová se následně dle typu zprávy. Buď odpoví zpět agentovi, jako je tomu například u zpráv `Hello` nebo `Rooms`, nebo v případě zprávy `Events` zjistí z jejího obsahu, ze kterého zařízení událost pochází. Následně vyhledá, zda má u sebe registrované nějaké klienty mobilní aplikace, kteří jsou připojeni k této místnosti a tuto zprávu jim zašle.

V případě zprávy `RoomConnected` si server pouze uloží do paměti, že zařízení je funkční a připojeno.

4.3.1.2 Konkrétní implementace

O zpracování zpráv od agenta se stará třída `AgentMessageParser`. Ta pro práci s XML zprávami používá knihovnu LINQ to XML dostupnou z System.Xml.Linq, která umožňuje snadné dotazování nad daty.

Pro komunikaci s programy Agent má RC server třídu `WSHandlerAgent`, která využívá knihovnu Microsoft.WebSocket. Tato třída zajišťuje autorizaci a navázání spojení s klientem, v případě úspěšného navázání spojení předává získaný kontext klienta třídě `AgentWebSocket`, ta si kontext společně s informacemi o připojeném agentovi ukládá do kolekce navázaných spojení. V případě, že bude připojeno více klientů najednou, hrozí přístup z více vláken zároveň. Proto byla zvolena kolekce ConcurrentDictionary, do které lze přistupovat z více vláken souběžně.

Třída `AgentWebSocket` obsluhuje metody `OnOpen`, `OnMessage`, `OnClose` a `OnError`, tyto metody slouží k obsluze událostí navázaného spojení.

Dále tato třída poskytuje statickou metodu `SendToClients`, která slouží pro komunikaci s klientem, neboli s programem Agent.

4.3.2 Komunikace s mobilní aplikací

Po připojení klienta mobilní aplikace k RC serveru si server uloží data o uživateli do své paměti. Do proměnné, která určuje, ke které místnosti je klient připojen, přiřadí server prázdnou hodnotu. Dále pak server vyčkává na další zprávy od uživatele.

4.3.2.1 Druhy zpráv mezi mobilní aplikací a RC serverem

RC Server je schopen od uživatele zpracovat 5 druhů zpráv:

Hello Touto zprávou dává agent na vědomí, že je zapnut. A jako odpověď očekává opět zprávu Hello. Důvod této komunikace je stejný jako u Hello zpráv mezi RC serverem a Agentem.

Rooms Touto zprávou uživatel žádá o zaslání seznamu místností, které bude moci ovládat.

RoomSelect Touto zprávou informuje mobilní aplikace, že uživatel vybral danou místnost k ovládání, tedy že chce odebírat události ze zařízení umístěného v místnosti.

Command Touto zprávou posílá mobilní aplikace příkaz k provedení určité ovládací akce. Například vytočení telefonního čísla nebo zvýšení hlasitosti.

Configuration Tento druh zpráv je zde připraven pro další verzi, ve které se počítá i s možností úpravy konfigurace videokonferenčních zařízení pomocí mobilní aplikace.

Listing 4.2: Ukázka zprávy Command

```
<?xml version="1.0" encoding="utf-8" ?>
<Command>
  <Dial>
    <Number>vojtech.hlavacka@4video.cz</Number>
  </Dial>
</Command>
```

Všechny zprávy RC server opět očekává ve formátu XML. Strukturu těchto zpráv definují XSD soubory, které jsou přiloženy s implementací na CD.

Při příchozí zprávě stejně jako u zprávy od agenta server zjistí, o kterou zprávu se jedná a zda je obsah správy validní. Pokud ne, opět odpovídá zprávou `Error`. V případě validní zprávy:

- Na zprávu `Hello` odpovídá zprávou `Hello`.
- Při zprávě `Rooms` zjistí z databáze, jaké místnosti může daný uživatel ovládat. Místnosti serializuje do XML souboru a zašle je zpět mobilní aplikaci.
- Při zprávě `RoomsSelected` si server pouze uloží do paměti, že daná aplikace ovládá zvolené zařízení a chce tedy odebírat informace o událostech.
- Na zprávu `Configuration` odpovídá zprávou `NotImplemented`.
- V případě zprávy `Command` server získá z databáze informace o zařízení, kterému je příkaz na ovládání směrován. Na jejich základě pomocí knihovny `TelePresenceLibrary` vygeneruje XML příkaz k ovládání a tento příkaz zašle příslušnému Agentskému programu, který má zařízení na starosti.

4.3.2.2 Konkrétní implementace

O zpracování zpráv od agenta se stará třída `AgentMessageParser`. Pro komunikaci s mobilními aplikacemi má RC server třídu `WSHandlerAndroid`, která využívá knihovnu Microsoft.WebSocket.

Tato třída zajišťuje autorizaci a navázání spojení s klientem. V případě úspěchu předává získaný kontext klienta třídě `MobileWebSocket`, ta si kontext společně s informacemi o připojeném uživateli ukládá do kolekce navázaných spojení.

Třída `MobileWebSocket` obsluhuje metody `OnOpen`, `OnMessage`, `OnClose` a `OnError`, tyto metody slouží k obsluze událostí navázaného spojení.

Dále tato třída poskytuje statickou metodu `SendToClients`, která slouží pro komunikaci s klientem, tedy mobilní aplikací.

4.4 Remote Control Agent

Tato komponenta je prostředníkem mezi jednotlivými videokonferenčními zařízeními a RC serverem. Má na starosti realizaci koncové komunikace se zařízeními. Zjednodušeně lze říci, že pouze přeposílá informace, které získá.

Jejím úkolem je ihned po startu navázat spojení na RC server, od RC serveru si vyžádat seznam zařízení, ke kterým se může zaregistrovat pro odběr událostí a která také může ovládat. V případě úspěšné registrace uvědomí RC server o tomto stavu. Dále program vyčkává na události od zařízení, nebo na zprávy od RC serveru.

4.4.1 Typ Programu

Samotný program je realizován jako Microsoft Windows service [18], dříve známý jako služby NT. Tento koncept umožňuje vytvořit programy určené pro neustálý běh na pozadí operačního systému Windows. Takto vytvořenou službu je možné pak nainstalovat na operační systém Windows, kde se zaregistruje jako služba. Díky tomu je pak možné sledovat její stav ve správci služeb, nastavit jí automatický start při startu systému, případně službu restartovat. Službu je pak také možno spustit v bezpečnostním kontextu specifického uživatele, který je odlišný od současně přihlášeného uživatele v systému.

4.4.2 Komunikace s RC serverem

Komunikaci s RC serverem zajišťuje třída `RCServerCommunicator`, která pro realizaci spojení protokolem WebSocket používá nativní knihovnu System.Net.WebSocket, která je součástí .Net Frameworku 4.5. Třída `RCServerCommunicator` poskytuje metodu `Connect`, jenž zapouzdřuje navázání spojení včetně procesu autentizace. Dále poskytuje property `IsConnected`, která indikuje, zda je spojení s RC serverem navázáno. Metoda `Send` dovoluje odeslat data RC serveru. Dále je zde blokující metoda `ReceiveStart`, která přijímá data od RC serveru a poté je předává třídě `MessageParser` na zpracování.

Listing 4.3: Realizace property IsConnected

```
public bool IsConnected { get { return _socket !=
   null && _socket.State == WebSocketState.Open; } }
```

4.4.3 Kontrola konektivity

Jelikož je spojení na RC server klíčové pro jakoukoliv další činnost programu, je tedy nutné, aby v případě, že se spojení s RC serverem rozpadne, byl proveden opětovný pokus o navázání spojení. Tento pokus musí být opakován dokud není spojení navázáno. Aby ale nedocházelo ke zbytečnému vytížení hardwaru, na kterém program běží, je tento pokus realizován jednou za zvolenou periodu, základní nastavení je 30 vteřin.

O tuto funkcionalitu se stará metoda `ConnectionCheck` z třídy ConnectivityManager, která využívá property `IsConnected`. Je spuštěna v samostatném vlákně a každou zvolenou periodu kontroluje, zda je spojení s RC serverem stále aktivní. Pokud ano, čeká na další periodu, pokud ne, pokusí navázat spojení.

Listing 4.4: Realizace metody ConnectionCheck

```
public void ConnectionCheck()
{
    while (true)
    {
        if (!rcServer.IsConnected)
        {
            Connect();
        }
        Thread.Sleep(period * 1000);
    }
}
```

4.4.4 Zpracování zpráv od RC serveru

Toto zpracování má na starosti třída `MessageParser`. Ta obdobně jako u RC serveru používá ke zpracování knihovny LINQ to XML. Třída umí zpracovávat následující 4 druhy zpráv:

Hello Odpověď, že RC server dostal `Hello` zprávu od Agenta.

Rooms Jedná se o odpověď na požadavek o seznam místností k ovládání. Zpráva obsahuje seznam místností se všemi potřebnými informacemi.

Error Jedná se o odpověď serveru na zprávu poslanou Agentem o neúspěšném zpracování této zprávy.

Control Požadavek RC serveru na ovládání určité místnosti.

Všechny zprávy RC server opět očekává ve formátu XML. Strukturu těchto zpráv definují XSD soubory, které jsou přiloženy s implementací na CD.

Při příchozí zprávě Agent zjistí, o kterou zprávu se jedná a zda je obsah správy validní. Pokud ne, zapíše chybu do chybového logu. V případě validní zprávy:

- U zprávy `Hello` zaznamená čas příchodu zprávy a čeká na periodu, kdy bude posílat zprávu `Hello` on.
- Při zprávě `Error` zapíše obsah zprávy do chybového logu.
- Při zprávě `Rooms` deserializuje obsah zprávy a tento objekt předá třídě `EndpointCommunicator`, jejíž funkce bude ještě popsána.
- V případě zprávy `Control` deserializuje obsah zprávy a volá metodu SendToRoom z třídy `EndpointCommunicator`, která zajistí odeslání ke koncovému zařízení.

4.4.5 Komunikace s koncovými zařízeními

Tuto komunikaci má na starosti třída `EndpointCommunicator`. Tato třída zapouzdřuje komunikace s videokonferenčními zařízeními. Obsahuje 2 veřejné metody. Metodu `ConnectAllRooms` a přetíženou metodu `SendToEndpoint`, dále obsahuje statickou kolekci RoomsDictionary, která je typu ConcurrentDictionary.

V případě volání metody `ConnectAllRooms` program iterativně projde kolekci místností a pro každou místnost provede následující kroky:

1. V registračním příkaze pro odběr událostí nahradí adresu serveru za adresu, na které je program schopen vytvořit Http Listenter.
2. Zašle zařízení v místnosti požadavek k zaregistrování o příjem událostí.
3. Vyčká na odpověď od zařízení.
   a) V případe úspěchu pokračuje bodem 4.
   b) V případe neúspěchu zašle RC serveru zprávu `Error` s informací o neúspěchu.
4. Vytvoří Http Listener na adrese zaslané v registračním příkaze danému zařízení. A tento listener spustí v novém vlákně.
5. Uloží informace o místnosti a identifikátoru Listeneru do kolekce RoomsDictionary.
6. Informuje RC server s úspěšné registraci zařízení zprávou `RoomConnected`.

Pro HTTP Litener je použita knihovna System.Net.HttpListener, která je součástí .Net Frameworku. Pro zasílání zpráv jednotlivým zařízením je použita knihovna System.Web.WebClient, která je také součástí .Net Frameworku.

4.5 Mobilní aplikace - Remote Control App

Jedná se o mobilní aplikaci pro operační systém Android, která je určena koncovým uživatelům. Jejím účelem je zprostředkování ovládání místností, ke kterým má uživatel přístup.

Hlavním stavebním blokem aplikace pro systém android je takzvaná aktivita. [19] Aktivita je komponenta aplikace, která poskytuje jednotlivé obrazovky, se kterými uživatel může interagovat. Každá aktivita poskytuje okno, ve kterém je prezentováno uživatelské rozhraní. Okno typicky vyplní celou obrazovku, ale může být menší než obrazovka a může se vznášet nad ostatními okny.

V pozadí je aktivita tvořena minimálně dvěma soubory. Definicí uživatelských komponent, včetně jejich rozložení, která je provedena pomocí XML souboru uloženého ve složce `\res\layout` a dále Java třídou, která dědí od `android.app.Activity` a obsahuje zdrojové kódy pro ovládání uživatelských prvků. Tato třída je uložena ve složce `\src\java`.

4.6 Části mobilní aplikace

Aplikace je rozdělena na více částí, které jsou v jazyku Java reprezentovány jako jednotlivé `package` neboli balíky.

package RemoteControl Tento balík obsahuje veškeré prvky související s uživatelským rozhraním. Jsou zde tedy všechny třídy, které nějakým způsobem obsluhují komponenty uživatelských prvků.

package ServerCommunication Tento balík obsahuje třídu pro komunikaci s RC serverem.

package Engine Obsahuje třídy provádějící logiku samotné aplikce. Jsou to tedy třídy, které neinteragují s prvky uživatelského rozhraní.

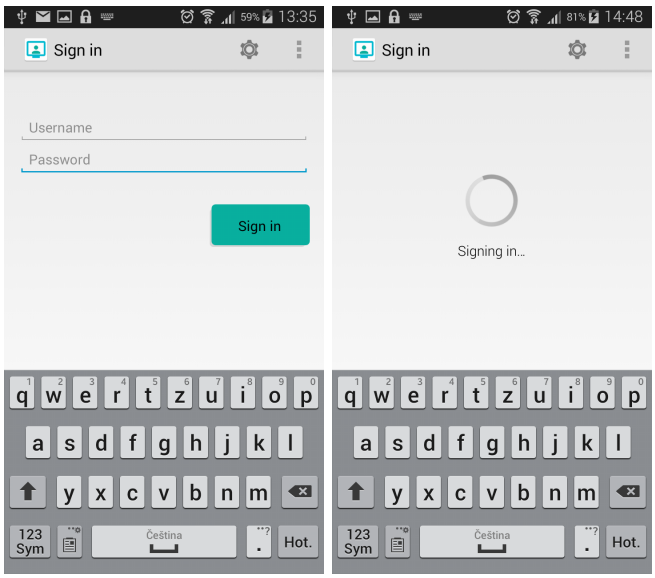
4.6.1 Část uživatelská - RemoteControl package

Aplikace je tvořena několika hlavními aktivitami, které zde budou ukázány a popsány. Mezi ty hlavní, kterým se zde budeme věnovat patří LoginActivity, RoomAktivity, RoomConnectedActivity a CallActivity.

4.6.1.1 LoginActivity

Jedná se o aktivitu s příznakem MainActivity, který značí, že se jedná o aktivitu, která se při startu aplikace spustí jako první. Poskytuje uživateli možnost

zadat uživatelské jméno a heslo, a tím se přihlásit do aplikace, tedy k RC serveru. Po úspěšném přihlášení aplikace přechází na RoomActivity.



Obrázek 4.1: Ukázka grafické podoby LoginActivity

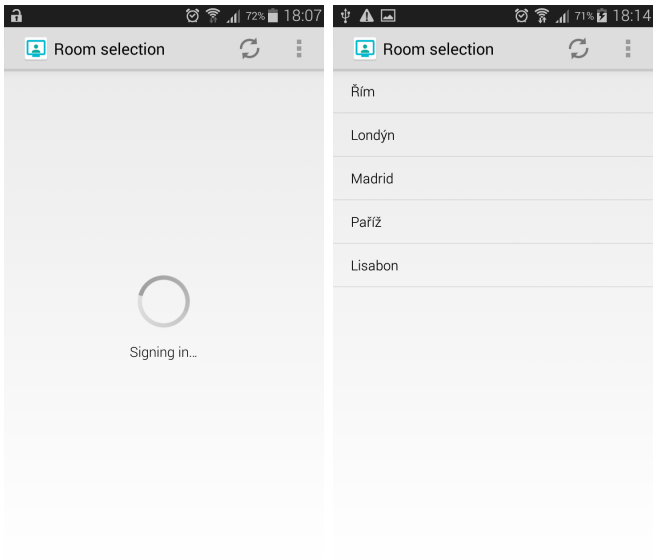
Po zadání uživatelského jména a hesla aplikace tato data uloží do SharePreferences, v případě dalšího spuštění aplikace se data vyčtou a uživatel je automaticky přihlášen. Při přihlašování je uživateli zobrazen progress bar s informací o probíhajícím přihlašování.

4.6.1.2 RoomActivity

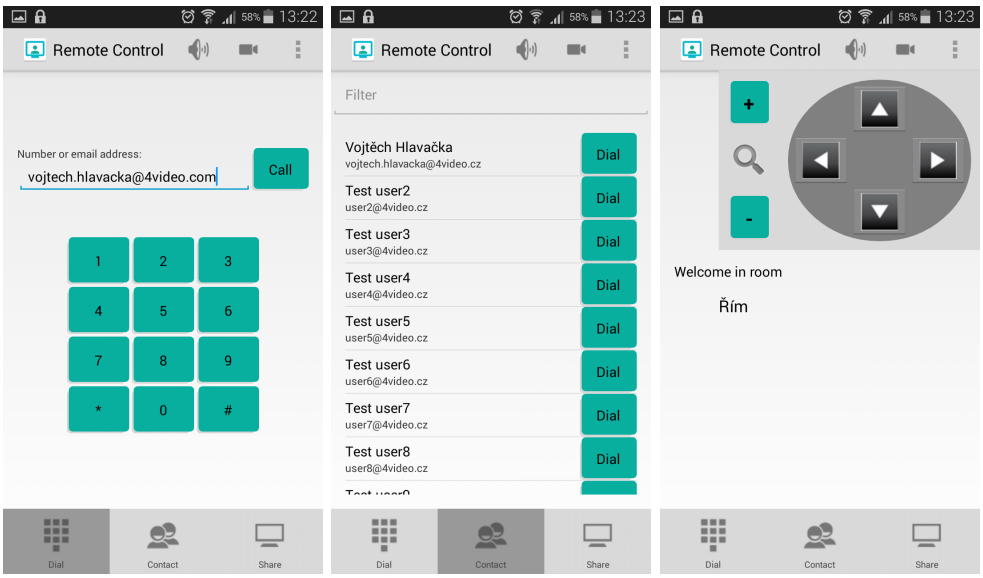
Tato aktivita uživateli zobrazí list místností, které může ovládat. Uživatel má možnost zvolit jednu z nich a přejít tak do aktivity RoomConnected. Při této akci se odesílá RC serveru zpráva `RoomSelected`. Pomocí kontextového menu se pak uživatel může odhlásit z aplikace a přejít tak na LoginActivity. Další možností je pak obnovení seznamu místností.

4.6.1.3 RoomConnectedActivity

Tato aktivita už umožňuje uživateli ovládat zařízení. Uživatel může ovládat hlasitost, kameru, hledat v kontaktech, přijímat a vytáčet hovory. Pro realizaci této aktivity byla použita komponenta `ViewGroup`, která dovoluje rozdělit obrazovku na více částí a není pak nutné pro každou záložku tvořit novou aktivitu.



Obrázek 4.2: Ukázka grafické podoby RoomAktivity



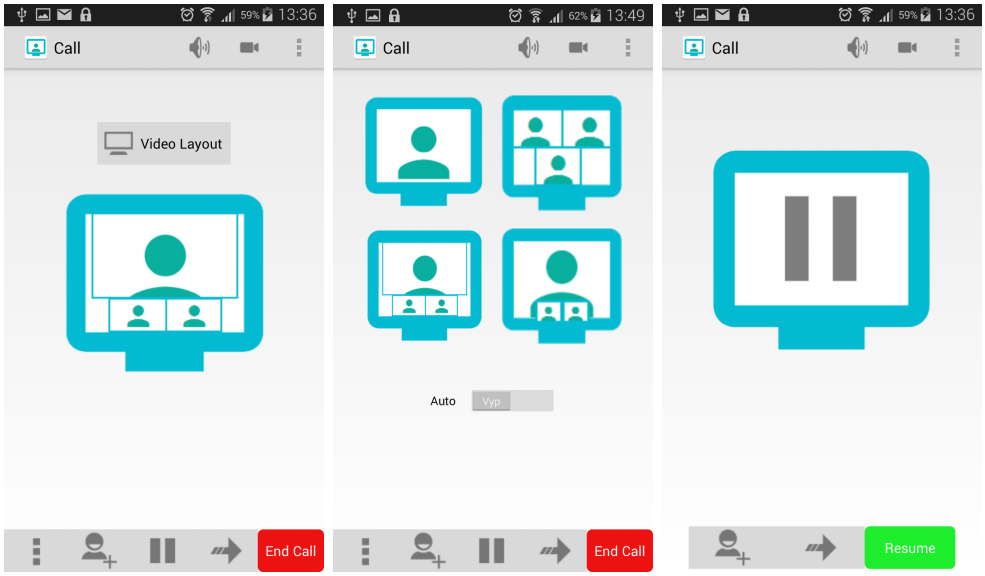
Obrázek 4.3: Ukázka grafické podoby RoomConnectedActivity

4.6.1.4 CallActivity

Do této aktivity lze přejít jen pokud je na videokonferenčním zařízení, které je ovládáno, úspěšně spojený hovor. To znamená, že tato aktivita není vyvolávána uživatelem z nějaké z předchozích aktivit. Ale je vyvolávána třídou z package Engine, jakožto reakce na zprávu RC serveru. Tato akce bude blíže rozebrána v následujících kapitolách.

V případě, že je tedy spojen hovor a aplikace přešla do této aktivity, má uživatel možnost volit layout rozložení videa a prezentace, zapínat a vypínat sdílení, přidržet nebo ukončit hovor. Dále má také možnost ovládat kameru nebo hlasitost stejně jako v předchozí aktivitě. Jelikož, jak se v zadaní píše, jedná se o prototypovou aplikaci, tak ostatní funkce, jako je přizvání dalších účastníků do hovoru nebo přepojení hovoru, zatím nejsou implementovány. Uživateli je zobrazena informace, že tyto funkce budou přístupné v další verzi.

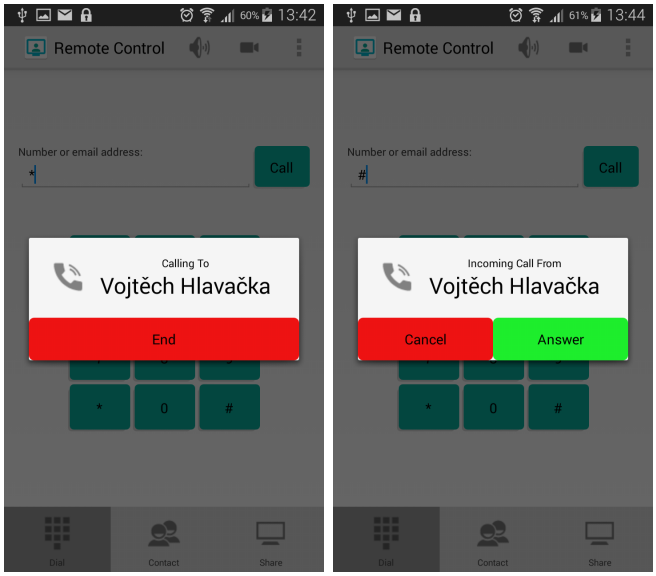
Pro realizaci této aktivity byla opět použita komponenta `ViewGroup`.



Obrázek 4.4: Ukázka grafické podoby CallActivity

4.6.1.5 Dialog Handler

Tato třída slouží pro zobrazování dialogů. Používá návrhový vzor singleton, díky kterému poskytuje možnost kdykoliv zobrazit dialog odchozího nebo příchozího hovoru, skrýt všechny dialogy nebo poskytnout informaci, který dialog je zobrazen.



Obrázek 4.5: Ukázka grafické podoby dialogů

4.6.2 Komunikace s RC serverem - RemoteControl package

Tuto komunikaci má na starosti třída `WebSocketCommunicator`, která využívá knihoven `Java-WebSocket:1.3.0`. Tyto knihovny poskytují třídu, která má metodu `connect` pro navázání spojení, metodu `send` pro odeslání a 4 abstraktní metody (`OnOpen`, `OnMessage`, `OnError`, `OnClose`) pro události o navázaném spojení.

Celá tato třída pak nabízí 3 statické veřejné metody. Metodu `connect`, která zapouzdřuje připojení k RC serveru včetně autorizace, metodu `disconect` pro rozvázání spojení s RC serverem a metodu `sendMessage`, která dovoluje odeslat správu serveru.

Dále má třída vlákno `helloThread`, které je spuštěno při navázání spojení. Úkolem tohoto vlákna je pravidelně odesílat serveru zprávu `Hello`, kterou dává klient najevo, že je probuzen, funkční a udržuje tak aktivitu v komunikačním kanálu.

4.6.3 Logika aplikace - RemoteControl Engine

V tomto balíku se nachází třída `MessageHandler`, Rozhraní `IControlClass` a jeho implementace `ControlClass`. Rozhraní `IControlClass` definuje seznam funkcí neboli zpráv, které lze odeslat RC serveru. `ControlClass` potom danou zprávu vygeneruje a za pomoci třídy `WebSocketCommunicator` odešle RC serveru.

4.6.3.1 Třída MessageHandler

Hlavní metodou této třídy je metoda `processMessage`. Tato metoda je volána metodou `OnMessage` z třídy `WebSocketCommunicator`. Jejím úkolem je zpracovat zprávy od RC serveru.

Metoda nejdříve zkontroluje, zda je zpráva v formátu XML a zda je XML správně strukturované. Pokud ano, kontroluje, zda obsahuje zprávu, kterou je možno rovnou zpracovat. Takovéto zprávy jsou čtyři:

- `IncomingCall`
- `CallConnected`
- `OutgoingCall`
- `CallDisconnect`

V případě `IncomingCall` a `OutgoingCall` zobrazuje příslušný dialog, v případě `CallConnected`, `CallDisconnect` přechází na příslušnou aktivitu.

V případě, že není možné zprávu rovnou zpracovat, ukládá zprávu do fronty zpráv čekajících na vyzvednutí.

4.7 Remote Control Server Web UI

Jedná se o webové stránky, které jsou určeny pro administrátory nebo správce skupin. Jejich úkolem je poskytnout jednoduchou správu místností, uživatelů a skupin.

Pro tvorbu webových stránek byla použita technologie ASP.NET WebForms, která byla popsána v kapitole 3.4.2.

K vytvoření webu byla použita šablona "ASP.NET Web Aplication", která je součástí Microsoft Visual studia 2012. Tato šablona programátorovi předpřipraví strukturu projektu a připojí sadu knihoven pro práci s webem.

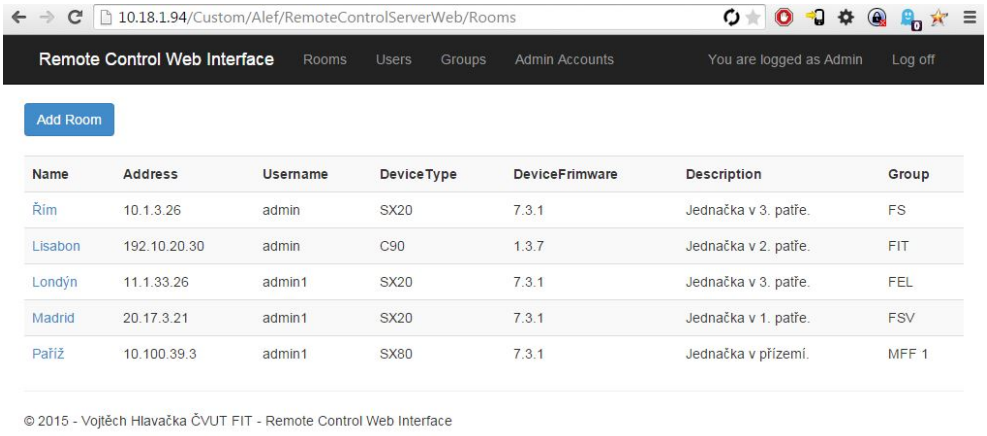
Pro přístup a práci s daty používá tento web knihovnu Data Repository, jejíž popis je v kapitole 4.2.

Pro vzhled a responsní uživatelské rozhraní byl použit framework bootstrap, který výrazně zpříjemňuje uživatelské rozhraní.

4.7.1 Jednotlivé části webového administračního rozhraní

Toto rozhraní se dělí na 4 části neboli stránky.

Login Tato stránka umožňuje se přihlásit. Nepřihlášený uživatel nemá v systému kromě přihlášení žádná práva.



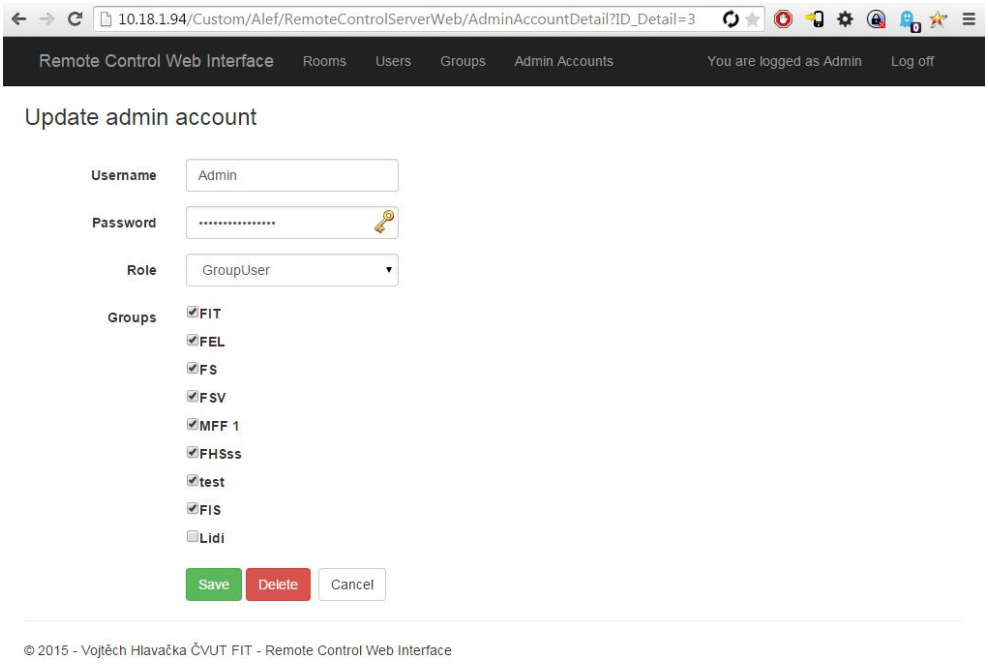
Obrázek 4.6: Webové administrační rozhraní - seznam skupin.

Rooms Tato sekce umožňuje zobrazit seznam místností. Pokud má uživatel práva jenom správce skupiny, je mu zobrazen pouze seznam místností, ke kterým má oprávnění. Administrátor vidí všechny místnosti. Dále je umožněno přidávat, upravovat a mazat tyto místnosti.

Users Tato sekce umožňuje zobrazit seznam uživatelů. Administrátor vidí všechny uživatele. V případe správce skupiny jsou zobrazeny pouze uživatelé spadající do jeho skupiny. Dále je umožněno přidávat, upravovat a mazat tyto uživatele.

Group Tato sekce umožňuje zobrazit seznam skupin. Administrátor vidí všechny skupiny. V případe správce skupiny jsou zobrazeny pouze skupiny, kterých je správcem. Dále je umožněno přidávat, upravovat a mazat tyto skupiny.

Admin accounts Tato sekce je přístupná pouze administrátorovi, který může vytvářet účty s právem administrátor nebo správce skupiny. Každému vytvořenému účtu může administrátor přiřadit skupiny ke správě.



Obrázek 4.7: Webové administrační rozhraní - upravení administrátorského účtu

Kapitola 5

Testování

V této kapitole bude popsáno testování jak jednotlivých částí systému, tak i testování systému jako celku. Zároveň bude proveden test zotavení systému po pádu jednotlivých komponent.

5.1 Test funkčnosti webového administračního rozhraní

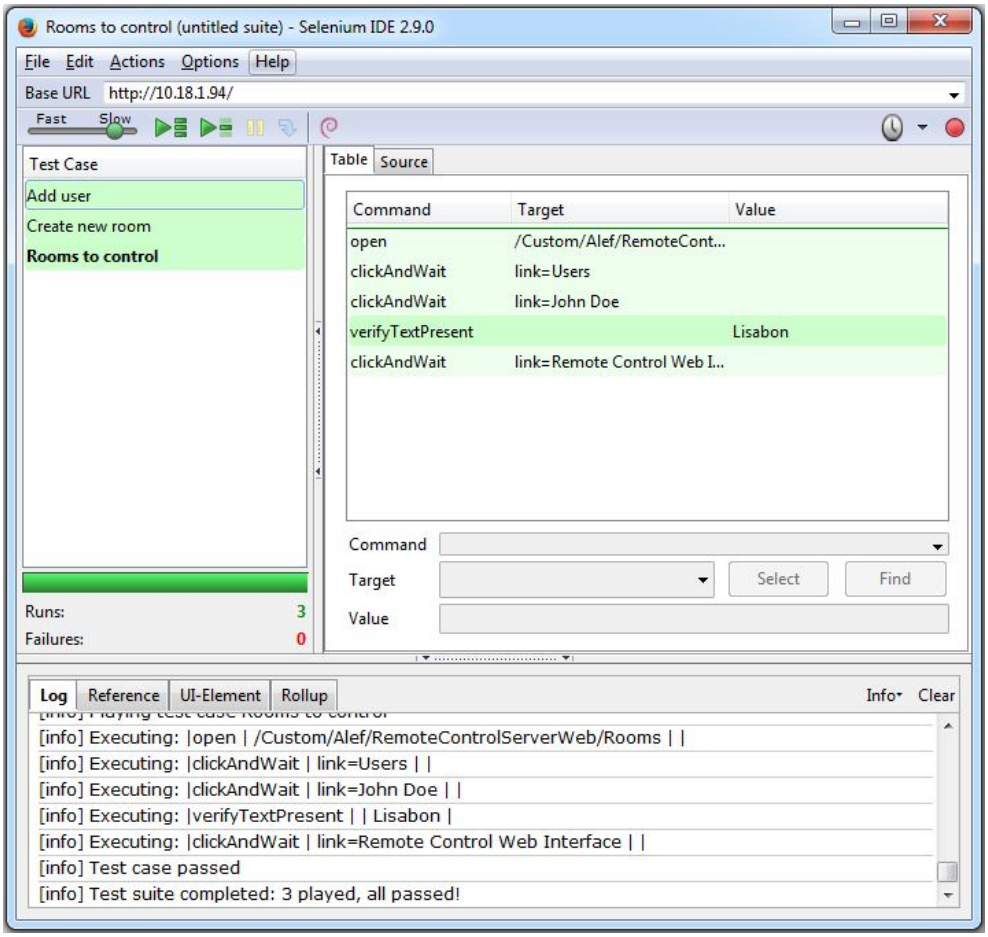
Pro webového rozhraní byly vytvořeny automatické testy. Pro jejich vytvoření byl použit software Selenium IDE.

Selenium IDE [20] je integrované vývojové prostředí pro Selenium skripty. Je implementováno jako rozšíření pro Firefox a umožňuje nahrávat, upravovat a ladit jednotlivé testy. Selenium IDE umožňuje nahrávat a přehrávat testy ve skutečném prostředí, ve kterém pak aplikace běží. Selenium umožňuje i ruční editaci nahraných skriptů.

Celkový test se skládá ze 4 částí, v první části je testováno vytvoření uživatele. Skript nejdříve zkouší, zda se při vytváření uživatele beze jména zobrazí všechny chybové hlášky. Následně zkusí vytvořit uživatele bez přiřazené skupiny a zase kontroluje přítomnost chybové hlášky. Poté zkouší vytvořit uživatele korektně. Pokud vytvoření proběhne, kontroluje existenci uživatele v seznamu uživatelů.

Druhá část testuje vytvoření nové místnosti. Nejdříve test zkouší vytvořit místnost bez vyplněných polí a následně kontroluje existenci chybových hlášek. Poté zadá místnost korektně a kontroluje, jestli se místnost vytvořila a jestli má všechny zadané parametry.

Část třetí testuje přiřazení místnosti do stejné skupiny, jako byla přiřazena uživateli v prvním kroku. V případě úspěšného přiřazení kontroluje, zda uživatel má místnost v seznamu jeho zařízení k ovládání.



Obrázek 5.1: Úspěšné spuštění testů v Selenium IDE

Pokud všechny 3 části skončí úspěšně, test zahájí část čtvrtou, ve které po sobě smaže veškerá data a tím otestuje i funkčnost mazání.

5.2 Test funkčnosti mobilní aplikace

Jelikož aplikace slouží pro ovládání externích zařízení, není možné použít podobné automatické testování, jako bylo použito u webového administračního rozhraní. Proto byl pro testování funkčnosti použit uživatelský test funkčnosti.

Aplikaci testovali celkem 3 testeři. Při testech bylo sledováno:

1. Schopnost jednotlivých funkcí ovládat koncové zařízení.

2. Schopnost aplikace reagovat na události vzniklé na zařízení.

3. Zpoždění ovládacích příkazů.

4. Zpoždění reakce na události vzniklé na zařízení.

5.3 Výsledky testů

5.3.1 Schopnost ovládat koncové zařízení.

Aplikace se ukázala jako schopná ovládat koncové zařízení, pro všechny své implementované funkce.

5.3.2 Schopnost aplikace reagovat na události zařízení.

Tato schopnost byla testována z 2 pohledů:

1. Událost byla vyvolána aplikací.

2. Událost byla vyvolána nezávisle na aplikaci (například dálkovým ovládáním).

Při testu aplikace dokázala správně reagovat na 4 druhy událostí. A to na vytáčení neboli spojování hovoru, na událost o korektně spojeném hovoru, na ukončení hovoru a na příchozí hovor. Aplikace prokázala schopnost reagovat na všechny události nezávisle na tom, kým byla událost vyvolána.

5.3.3 Zpoždění ovládacích příkazů

Už v průběhu vývoje byla zjištěna delší reakční doba na ovládací příkazy. Ta byla způsobena tím, že před každým ovládacím příkazem byl posílán příkaz s žádostí o deaktivaci pohotovostního režimu. Proto bylo pořadí těchto příkazů prohozeno. Výsledná doba odezvy se pak zmenšila zhruba na polovinu. Nicméně i po tomto opatření je zpoždění viditelné, v průměru tvoří zpoždění 0,9 vteřiny, což je ale stále akceptovatelná hodnota.

Dalším zjištěným faktem je, že velice zřídka a pravděpodobně s náhodným výskytem se stává, že ovládací příkaz trvá viditelně déle než 1 vteřinu. Jelikož je celá komunikace asynchronní, může docházet k tomu, že příkaz s žádostí o deaktivaci úsporného režimu předběhne ovládací příkaz. Tuto hypotézu se však zatím nepodařilo ověřit a to především kvůli jejímu náhodnému výskytu.

5.3.4 Zpoždění reakce na události vzniklé na zařízení

U většiny událostí je doba reakce mobilní aplikace na danou událost stejná jako u zpoždění ovládacích příkazu. V průměru je hodnota zpoždění mezi 0,9 - 1 vteřinou. Vyjímkou je však událost o příchozím hovoru, zde je odezva zhruba

3 vteřiny, což už je hodnota na hraně akceptovatelnosti.

Po podrobnějším zkoumání bylo zjištěno, že při příchozím hovoru vznikají dvě události, kterými zařízení dává vědět o příchozím hovoru. První z nich je zpráva `Conference` a druhou `IncomingCallIndication`. RC server však posílá zprávu o události příchozího hovoru směrem k mobilní aplikaci až na základě zprávy `IncomingCallIndication`. Toto zpoždění lze tedy optimalizovat.

5.4 Test pádu serveru

Jak už čtenář ví, tak RC server je klíčovou částí celého systému. Jsou k němu připojeny nejen mobilní aplikace, ale i všechny programy Agent. A jelikož lze předpokládat, že jak programy Agent, tak mobilní aplikace se mohou nacházet na zařízeních připojených v sítích s překladem adres, tak server po zotavení ze svého pádu, nemůže tato zařízení zkontaktovat. Proto byl každé z klientských aplikací implementován algoritmus, který pravidelně kontroluje připojení směrem k serveru a pokud spojení není navázáno, zkouší jej navázat. Funkce těchto algoritmů bude předmětem tohoto testování.

Pro tento test byla k RC serveru připojena jedna mobilní aplikace a jeden program Agent. Pád serveru byl simulován 2 způsoby:

1. Při softwarovém vypnutí IIS serveru, na kterém RC server běží.
2. Odpojení počítače z počítačové sítě, která poskytuje konektivitu k počítači.

V testu bylo sledováno:

1. Schopnost znovu připojení programu Agent.
2. Existence zápisu o rozpojení komunikace v chybovém logu programu Agent.
3. Schopnost znovu připojení mobilní aplikace.
4. Zobrazení informace o nečinnosti serveru směrem k uživateli.

5.4.1 Výsledek testu pádu serveru

1. Programu Agent prokázal schopnost znovu se připojit k RC serveru a to v obou případech simulovaného pádu serveru. Průměrná doba znovu obnovení spojení po nastartování RC serveru byla 20 vteřin. Tato hodnota byla očekávána, jelikož perioda, se kterou program Agent kontroluje dostupnost serveru byla nastavena na 30 vteřin.
2. Programu Agent korektně zapsal informace do chybového logu.

3. V obou případech simulovaného pádu serveru byl uživatel přesměrován na obrazovku pro přihlášení do aplikace, kde mu byla zobrazena hláška o chybě při spojení se serverem. Po nastartování RC serveru byl uživatel automaticky přihlášen.

4. Jak bylo popsáno v předchozím bodě uživateli byla zobrazena hláška o chybě spojení se serverem.

5.5 Test výkonu

Jelikož lze očekávat, že k RC serveru bude připojeno více mobilních zařízení najednou, byl proveden test výkonu RC serveru, přesněji řečeno test zatížení RC serveru. Pro tento test byl vytvořen program se jménem PerformanceTester (je přiložen na CD). Tento program simuluje připojení většího množství klientů. Úkolem každého takto vytvořeného klienta je autentizovat se, a dále v nekonečné smyčce s periodou přibližně 50 milisekund posílat požadavky na ovládání testovacího zařízení. Server pak tyto požadavky posílá agentovi, který byl upraven tak, aby tyto požadavky zahazoval.

Dále pak byla použita mobilní aplikace, u které byly sledovány 2 věci:

1. S jakým zpožděním přichází od RC serveru odpovědi na `hello` zprávy.

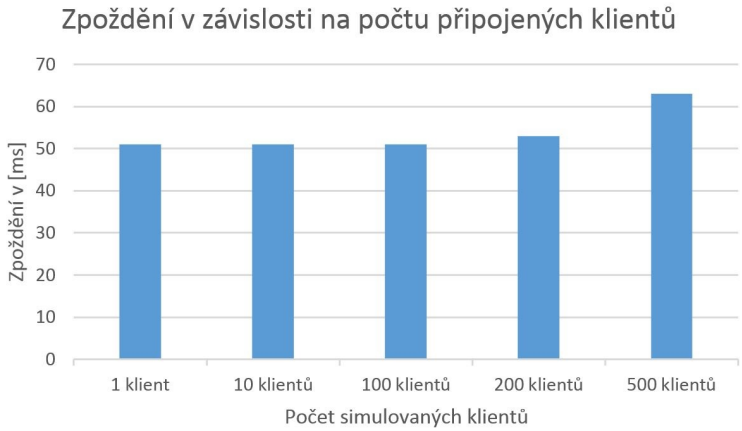
2. Zda se viditelně zpomalilo ovládání reálného videokonferenčního zařízení.

Testování probíhalo ve čtyřech kolech. V prvním kole bylo PerformanceTesterem simulováno 10 klientů, což při periodě 50 milisekund je 200 požadavků za vteřinu. V druhém kole bylo simulováno 100 klientů, což při periodě 50 milisekund je 2000 požadavků za vteřinu. V třetím kole 200 klientů a ve čtvrtém kole 500 klientů, což při stejné periodě je 10000 požadavků za vteřinu.

5.5.1 Výsledky testů výkonu

Zpoždění odpovědí pro testy s jednotlivými parametry je znázorněno v grafu na obrázku 5.2. Z grafu je vidět, že doba zpoždění se zvětšuje až pro 200 simulovaných uživatelů. Toto zvětšení je však minimální, větší zvětšení doby odezvy je vidět už u 500 simulovaných uživatelů, tedy 10000 požadavků za vteřinu. Zde už byla vidět i změna vytížení procesoru procesem RC serveru, vytížení stouplo přibližně na 20 %.

V testech bylo také sledováno zpoždění ovládání reálného videokonferenčního zařízení. Jelikož čas mobilního zařízení, videokonferenčního zařízení a RC serveru nelze nikdy přesně synchronizovat, není možné toto zpoždění přesně změřit. Bylo tedy zkoumáno jen viditelní zpoždění. Toto zpoždění nebylo zpozorováno.



Obrázek 5.2: Zpoždění hello zpráv v závislosti na počtu připojených klientů.

5.5.2 Závěr testů výkonu

I přes připojení 500 simulovaných klientů, kteří vytvářeli 10000 požadavků za vteřinu, se nepodařilo server vytížit tak, aby bylo zpoždění ovládání pro uživatele viditelné. Jelikož pro praxi je tento počet požadavků na server značně naddimenzován, lze tedy říci, že RC server v tomto testu obstál.

5.6 Testovací prostředí

Pro testování byla postavena laboratoř, ve které celé testování probíhalo.

5.6.1 Parametry laboratoře

K ovládání bylo vyčleněno zařízení: Cisco TelePresence System Codec C20 s firmware 7.1.

RC server byl spuštěn na virtuálním stroji s následujícími parametry:

Tabulka 5.1: Parametry virtuálního stroje pro RC server.

Název	Vmware player
Prosesor	Intel Core i7-2640M 2,8GHz (pouze 2 jádra)
Pamět RAM	3,3 GB
Operační systém	Windows 8.1.1, 64-bit

Program Agent byl spuštěn na virtuálním stroji s následujícími parametry:

Tabulka 5.2: Parametry virtuálního stroje pro program Agent.

Název	Vmware player
Prosesor	Intel Core i7-2640M 2,8GHz (pouze 1 jádro)
Pamět RAM	2 GB
Operační systém	Windows 7, 64-bit

K testování mobilní aplikace byl použit telefon Samsung Galaxy S5 mini s operačním systémem Android 4.4.2.

Závěr

Cílem práce bylo navrhnout a implementovat prototypovou aplikaci, pomocí které bude možné ovládat videokonferenční zařízení od společnosti Cisco z mobilního zařízení. Dále realizovat návrh a implementaci webového administračního rozhraní, které umožní definovat autorizaci přístupu uživatelů k videokonferenčním zařízením a díky tomu umožní celé řešení provozovat formou služby.

S ohledem na zadání diplomové práce byly všechny cíle úspěšně splněny. Výsledná aplikace je schopna realizovat ovládání základní funkčnosti videokonferenčních zařízení. Definici přístupů uživatelů do aplikace a k jednotlivým videokonferenčním zařízením je možné realizovat pomocí vytvořeného webového administračního prostředí a je tedy možné provozovat celé řešení jako službu.

Pozitivní je také vyjádření zadavatelské společnosti, která tuto práci hodnotí jako kvalitní základ pro realizaci finálního produktu, který bude plnohodnotným doplňkem videokonferenčních zařízení od společnosti Cisco.

Vývoj aplikace dnem odevzdání práce rozhodně nekončí. Kromě testování, které probíhalo v laboratoři, bude dále probíhat testování ovládání různých typů videokonferenčních zařízení a také testování ovládání zařízení umístěných v produkční síti. Vzhledem k povaze aplikace lze doiplementovat další funkce.

Literatura

[1] Romaniello, J. F.: Token-based Authentication with Socket.IO [online]. January 2014, [cit. 2015-02-26]. Dostupné z: https://auth0.com/blog/2014/01/15/auth-with-socket-io/

[2] Thiel, E.: *Mluvíme tělem - řeč těla prozradí víc než tisíc slov.* Knižní klub, první vydání, ISBN 80-7302-049-1, 136 s.

[3] Heggestuen, J.: One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet [online]. December 2013, [cit. 2015-02-04]. Dostupné z: http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10#ixzz3Tnjc5GD3

[4] Cisco Systems, Inc.: *Collaboration Endpoints [online].* [cit. 2015-02-19]. Dostupné z: http://www.cisco.com/c/en/us/products/collaboration-endpoints/index.html

[5] Cisco Systems, Inc.: *Cisco Intelligent Proximity [online].* March 2015, [cit. 2015-02-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.cisco.proximity.client

[6] Cisco Systems, Inc.: *Cisco TelePresence System - Application Programmer Interface (API) Reference Guide [online].* December 2014, [cit. 2015-02-09]. Dostupné z: http://www.cisco.com/c/dam/en/us/td/docs/telepresence/endpoint/codec-c-series/tc7/api-reference-guide/codec-c20-api-reference-guide-tc73.pdf

[7] Information Sciences Institute University of Southern California: *TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION [online].* September 1981, [cit. 2015-04-04]. Dostupné z: http://tools.ietf.org/html/rfc793

[8] WWW Consorcium: *The WebSocket API [online].* September 2012, [cit. 2015-04-11]. Dostupné z: http://www.w3.org/TR/websockets/

[9] Microsoft Corporation: *Message Queuing (MSMQ)[online].* [cit. 2015-04-15]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms711472.aspx

[10] Google, Inc.: *Google Cloud Messaging for Android[online].* [cit. 2015-04-16]. Dostupné z: https://developer.android.com/google/gcm/index.html

[11] Microsoft Corporation: *Windows Communication Foundation [online].* [cit. 2015-04-16]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms711472.aspx

[12] Hess, K.: Top 10 Enterprise Database Systems to Consider in 2015 [online]. May 2015, [cit. 2015-03-14]. Dostupné z: http://www.serverwatch.com/article.php/3883441/Top-10-Enterprise-Database-Systems-to-Consider.htm

[13] Minářů, V.: *Grafický generátor TCL scriptu pro Cisco Voice Gateway.* Diplomová práce, České vysoké učení technické, Fakulta Elektrotechnická, Praha, 2011.

[14] Egger, F.: Lo-Fi vs. Hi-Fi Prototyping: how real does the real thing have to be? [online]. May 2000, [cit. 2015-03-04]. Dostupné z: http://www.telono.com/articles/OZCHI2000.htm

[15] Nielsen, J.: 10 Usability Heuristics for User Interface Design [online]. January 2000, [cit. 2015-03-04]. Dostupné z: http://www.nngroup.com/articles/ten-usability-heuristics/

[16] Microsoft Corporation: *The Repository Pattern [online].* [cit. 2015-04-15]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff649690.aspx

[17] Microsoft Corporation: *Implementing Singleton in C# [online].* [cit. 2015-04-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff650316.aspx

[18] Microsoft Corporation: *Introduction to Windows Service Applications [online].* [cit. 2015-04-15]. Dostupné z: https://msdn.microsoft.com/en-us/library/d56de412(v=vs.110).aspx

[19] Google, Inc.: *Activities [online].* [cit. 2015-04-15]. Dostupné z: http://developer.android.com/guide/components/activities.html

[20] SeleniumHQ: *Selenium IDE [online].* [cit. 2015-04-30]. Dostupné z: http://www.seleniumhq.org/projects/ide/

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

IIS Internet Information Services

NAT Network address translation

GCM Google Cloud Messaging

GPL Global Price List

HDMI High-Definition Multimedia Interface

DVI Digital Visual Interface

API Application programming interface

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

SSH Secure Shell

RC server Remote Control server

TCP Transmission Control Protocol

W3C World Wide Web Consortium

WCF Windows Communication Foundation

LINQ Language Integrated Query

SQL Structured Query Language

MS Microsift

IDE Integrated Development Environment

SaaS Software as a Service

PAT Port Address Translation

TLS Transport Layer Security

.Net Microsoft .NET Framework

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `Android_RemoteControl` ..... Mobilní aplikace
    - `RemoteControlAgent` ..... program Agent
    - `RemoteControlServer`
      - `Data` ..... Data Repository
      - `RemoteControlServer` ..... Remote Control Server
        - `Schemes` ..... schémata pro validaci XML zpráv
      - `RemoteControlServerWebUI` .... Webové administrační rozhraní
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `wireframes` ..... všechny vytvořené wireframes
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF