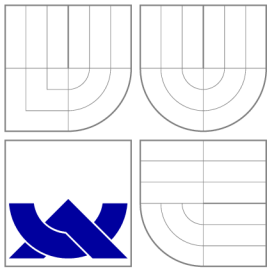
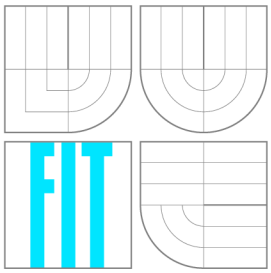


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VZDÁLENÝ PŘÍSTUP K MOBILNÍM ZAŘÍZENÍM

REMOTE ACCESS TO MOBILE DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR BOBULA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2016

Abstrakt

Táto práca sa zaoberá vzdialeným prístupom k mobilným zariadeniam so systémom Android a možnosťami ich vzdialeného ovládania. Cieľom bolo vytvoriť rozhranie pre vzdialené ovládanie mobilných zariadení pomocou vzdialenej plochy a pre vzdialenú zmenu nastavení mobilných zariadení. Zameral som sa na využitie vzdialeného ovládania pre asistenciu pri používaní mobilných zariadení. Riešením bolo vytvoriť aplikáciu s rozhraním pre ovládanie, ďalej aplikáciu, ktorá bude ovládaná a server, cez ktorý budú komunikovať. Server bol vytvorený s použitím technológie Node.js. Pre umožnenie komunikácie aplikácií v reálnom čase bola použitá knižnica Socket.io. Vzdialená plocha využíva posielanie snímok obrazovky a posielanie súradníc kliknutí na snímok. Kliknutia sú na vzdialenom zariadení vykonané prostredníctvom skrytého API. Vzdialená zmena nastavení využíva API z Android SDK. Podarilo sa vytvoriť systém, ktorý umožňuje vzdialene ovládať Android zariadenie a vzdialene meniť jeho nastavenie.

Abstract

This thesis deals with remote access to mobile devices with the Android system and their remote control possibilities. The aim was to create the interface for mobile devices remote control by remote desktop and also for changing mobile device's settings remotely. I focused on the utility of remote control for assistance in mobile devices usage. The solution was to create an application with the control interface, then application being controlled, and the server enabling communication between them. The server has been created with use of the Node.js technology. Socket.io library has been used for enabling communication between applications in real time. The remote desktop utilizes sending of screenshots and its clicks coordinates. These clicks on the remote device are executed by hidden API. Remote settings change uses API of Android SDK. It has been managed to develop the system that enable the control of the Android device and also to change its settings, both remotely.

Kľúčové slová

Mobilná aplikácia, vzdialený prístup, vzdialené ovládanie, vzdialená podpora, vzdialená plocha, Android

Keywords

Mobile application, remote access, remote control, remote support, remote desktop, Android

Citácia

BOBULA, Vladimír. *Vzdálený přístup k mobilním zařízením.* Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Burget Radek.

Vzdálený přístup k mobilním zařízením

Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne pod vedením pána Ing. Radka Burgeta, Ph.D. a uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Vladimír Bobula
24. mája 2016

Poďakovanie

Ďakujem vedúcemu práce Ing. Radkovi Burgetovi, Ph.D. za venovaný čas, konzultácie, odborné rady a vedenie. Ďalej by som rád poďakoval Ing. Tomášovi Poskerovi a spoločnosti Oscar Tech za príležitosť, rady a poskytnutie potrebných zdrojov.

Obsah

Kapitola 1

Úvod

Už pred príchodom smartfónov sa používatelia počítačov dostávali do situácií, pri ktorých potrebovali asistenciu pri spravovaní softvéru. Fyzický prístup k počítačom bol často časovo náročný a niekedy nemožný. Aj kvôli tomu bola vyvíjaná softvérová podpora vzdialeného prístupu k zariadeniam. Ďalším dôvodom bola možnosť využívať fyzicky nedostupné prostriedky na diaľku. Postupne sa tak vyvíjal softvér pre vzdialený prístup k zariadeniam. Dnes je bežné používať tzv. vzdialenú plochu. Jedná sa o pripojenie k vzdialenému zariadeniu, pri ktorom používateľ vidí aktuálnu plochu vzdialeného zariadenia a môže zariadenie ovládať (napr. pomocou klávesnice a myši). Táto funkcionalita je v súčasnosti často súčasťou operačného systému, no aj napriek tomu existuje množstvo softvérových riešení tretích strán.

Po príchode mobilných inteligentných zariadení (smartfóny, tablety, a iné zariadenia) začali tieto zariadenia postupne preberať čoraz viac funkcií počítačov. Je teda logické, že aj funkcionalita vzdialeného prístupu k týmto zariadeniam bola a je vyvíjaná.

Aktuálne dostupný softvér podporuje rôzne platformy (ako pre klientskú tak aj pre serverovú aplikáciu) a umožňuje pripojenie medzi rôznymi platformami. Poskytuje veľa možností pre vzdialené ovládanie.

Cieľom tejto práce nie je nahradiť alebo vyrovnať sa vyššie spomínaným riešeniam, ale vytvoriť v spolupráci s firmou Oscar Tech riešenie pre rodiny umožňujúce pomôcť členom rodiny, ktorí mobilné zariadenia používať nevedia. Dnes je bežné zdieľať pomocou moderných technológií multimediálny obsah. Vzdialený prístup a vzdialené ovládanie by mohlo dopomôcť k odstráneniu bariéry v zdieľaní obsahu s ľudmi, ktorým moderné technológie nie sú až tak blízke.

V práci je rozpísaná problematika vzdialenej podpory, jej možných funkcionalít a popis ich riešenia. Čitateľovi budú predstavené použité technológie a bude oboznámený s návrhom architektúry systému pre požadovanú funkcionalitu.

V závere je okrem zhrnutia naznačené budúce využitie výsledkov tejto práce.

Kapitola 2

Analýza problematiky vzdialenej podpory mobilných zariadení

Pri vzdialenej podpore je vhodné do softvéru zahrnúť aj možnosť komunikácie pre dohodnutie o vykonaní akcií a konzultácie stavu systému. Jednoduchý softvér pre podporu by mohol obsahovať možnosť odoslania snímky obrazovky, prípadne ďalších súborov kontaktu, ktorý by pomocou chatovania alebo volania navrhoval postup pre danú akciu. Problematickejšie však je umožniť pre podporu priamo ovládať zariadenie na diaľku. Hlavným problémom je, že niektoré nastavenia nie je možné použiť kvôli zabezpečeniam systému, na koľko by to mohlo byť zneužité.

2.1 Vzdialená zmena základných nastavení zariadenia

V rámci vzdialenej podpory môže byť potrebné meniť nastavenia zariadenia ako napríklad jas displeja, počet milisekúnd nečinnosti pre prechod do spánku, zmenu hlasitostí (pre volanie, média, notifikácie a iné). Pre tento účel je možné vytvoriť jednoduché rozhranie. Problémom je vykonanie zmien na vzdialenom zariadení, pretože systém nemusí zmenu nastavení umožňovať z bezpečnostných dôvodov.

Systém Android má zabezpečenie prístupu k rizikovým častiam rohrania riešené mechanizmom povolení. Aplikácii musí pre prístup k zmene systémových nastavení používateľ udeliť k tomu povolenie. Aplikačné rozhranie systémových knižníc poskytuje triedy pre správu rôznych nastavení, ktoré je možné využiť. Pre sprostrekovanie vzdialenej zmeny nastavení je potrebné dohodnúť komunikačné rozhranie medzi aplikáciou pre ovládanie a aplikáciu na ovládanom zariadení. Pomocou dohodnutých správ aplikácia odošle zmeny nasavení a na ovládanom zariadení budú správy spracované a vykonané príslušné akcie s využitím aplikačného rozhrania systémových knižníc. V používateľskom rozhraní pre zmenu nastavení vzdialeného zariadenia jez hľadiska spravovania vhodné, aby používateľovi boli zobrazené zvolené hodnoty týchto nastavení. Zobrazenie prináša potrebu synchronizácie, aby používateľ videl aktuálne hodnoty. Synchronizáciu je možné docieliť načítaním aktuálnych hodnôt pred zobrazením rozhrania a následným reagovaním na aktualizácie nastavenia. K aktualizácii môže dôjsť v prípade, že používateľ vzdialeného zariadenia zmení niektorú z hodnôt nastavenia práve v čase, keď je k zariadeniu vzdialene pripojený používateľ so zobrazeným rozhraním pre vzdialenú zmenu nastavení.

2.2 Vzdialená plocha

Pre sofistikovanejšie poskytnutie vzdialenej podpory sa používa vzdialená plocha. Ide o funkcionalitu pomocou ktorej je zobrazovaný obsah vzdialeného zariadenia zobrazený na zariadení fyzicky dostupnom. Umožňuje to vidieť aktuálny stav systému a ľahšie tak pochopiť problém. Okrem toho je tým dostupné ovládanie zariadenia akoby bolo fyzicky prítomné. Všetky klinutia (dotyky v prípade dotykových obrazoviek) na zobrazený obsah sa premietnu na vzdialené zariadenie (kliknutia sa vykonávajú akoby ich realizoval používateľ zariadenia). Pre realizovanie vzdialenej plochy popíšem dva rôzne prístupy.

2.2.1 Obsah realizovaný snímkami obrazovky

Hlavnou výhodou tohto prístupu je, že je zobrazený obsah tak, ako ho vidí používateľ, ktorý je fyzicky pri zariadení. Prináša to však viaceré nevýhody. Dátový tok bude väčší kvôli prenášaniu snímok obrazovky (veľkosť dátového toku bude priamo úmerná veľkosti snímok). Veľkosť snímok je ovplyvnená tým, v akom formáte je snímka odosielaná.

Formáty obrázkov:

- JPEG (Joint Photographic Experts Group) – Pre ukladanie obrázkov používa stratovú kompresiu. Využíva toho, že ľudské oko nevidí detaily medzi farebnými odtieňmi tak dobre ako detaily v svetlosti. Úrovňou kompresie sa dá zvoliť koľko detailov (dát) obrázka sa vynechá pre dosiahnutie menšej veľkosti súborov. Je vhodný najmä pre fotografie reálneho sveta, kde je v obrázku veľké množstvo rôznych farebných málo kontrastných prechodov.

- PNG (Portable Network Graphics) – Používa bezstratovú kompresiu. Pre kompresiu veľkosti súboru používa hľadanie vzorov v obrázku. Kompresia je reverzibilný, takže obrázok je obnovený presne. Použitie PNG je vhodné najmä pre obrázky obsahujúce oblasti s rovnakou farbou a ostrými hranami (napríklad logá) alebo tam, kde sa vyžaduje uloženie bez straty detailov (niekedy aj napriek väčšej veľkosti ako v JPEG – fotografie).

Okrem veľkosti snímok ovplyvní veľkosť dátového toku priamo úmerne aj počet snímok. Tento prístup však značne vyťažuje aj hardvér zariadenia (odosielanie, ale aj spracovanie snímok). To má okrem iných dopadov, dopad aj na batériu zariadenia (u mobilných zariadení je výdrž zariadenia kľúčová). Softvér pre prenášanie obrazu často používa pravidelné odosielanie určitého počtu obrázkov za sekundu. V prípade, že by nedošlo k žiadnej zmene obsahu ďalšie snímky zbytočne plytvajú dátovým tokom a výkonom zariadenia. Pokiaľ nechceme sledovať dynamický obsah, ale stačí nám obsah aktualizovať po vykonanej akcii (kliknutí), stačí snímku odosielať po vykonanej akcii. Výrazne sa tým zníži celkový počet odoslaných snímok.

V rámci kliknutia na snímok vzdialenej obrazovky sa musí získať relatívna poloha miesta kliknutia v rámci danej snímky. Poloha je reprezentovaná 2 súradnicami udávajúcimi polohu na x-ovej a y-ovej osi.

2.2.2 Emulácia obsahu

Riešenie vzdialenej plochy pomocou emulovania obsahu má hlavné obmedzenie v odlišnosti emulovanej plochy od reálnej. Namiesto snímok obrazovky sa prenáša popis systému (stav, nastavenie, informácie). Pokiaľ je cieľom použitia tohto prístupu zmenšenie dátového

toku emulované prostredie nebude identické s reálnym. Okrem zmenšenia dátového toku a šetrenia výkonu zariadenia sa môže zrýchliť aktualizovanie obsahu a aj vykonanie akcie v porovnaní prenosou obrazu pomocou snímok obrazovky (zrýchlenie záleží na konkrétnej implementácii). Čím viac sa emulátor má podobať reálnemu zobrazeniu (viac detailov), tým náročnejšia je realizácia. Komplikuje sa tým vývoj emulátora aj popisu systému. Rozhranie musí byť navrhnuté tak, aby bola zabezpečená spätná kompatibilita rôznych verzií emulátoru s rôznymi verziami aplikácií na vzdialených mobilných zariadeniach (nie je zaručené, že si používateľ nainštaluje aktualizáciu).

Pre dosiahnutie toho istého zobrazenia na oboch stranách, môže byť implementované na vzdialenom zariadení jednoduché prostredie s vzhľadom ako emulátor. Toto prostredie by potom realizovalo vrstvu medzi operačným systémom a používateľom.

2.3 Existujúce možnosti

Väčšina služieb poskytuje vzdialené ovládanie formou vzdialenej plochy iba pre ovládanie počítača z mobilného zariadenia. Dáva to zmysel vzhľadom k tomu, že mobilné zariadenia majú ľudia v prípade potreby pri sebe. Okrem menšej motivácie z hľadiska dopytu a využiteľnosti sú ďalšími prekážkami obmedzenia operačných systémov kvôli zabezpečeniu mobilných zariadení.

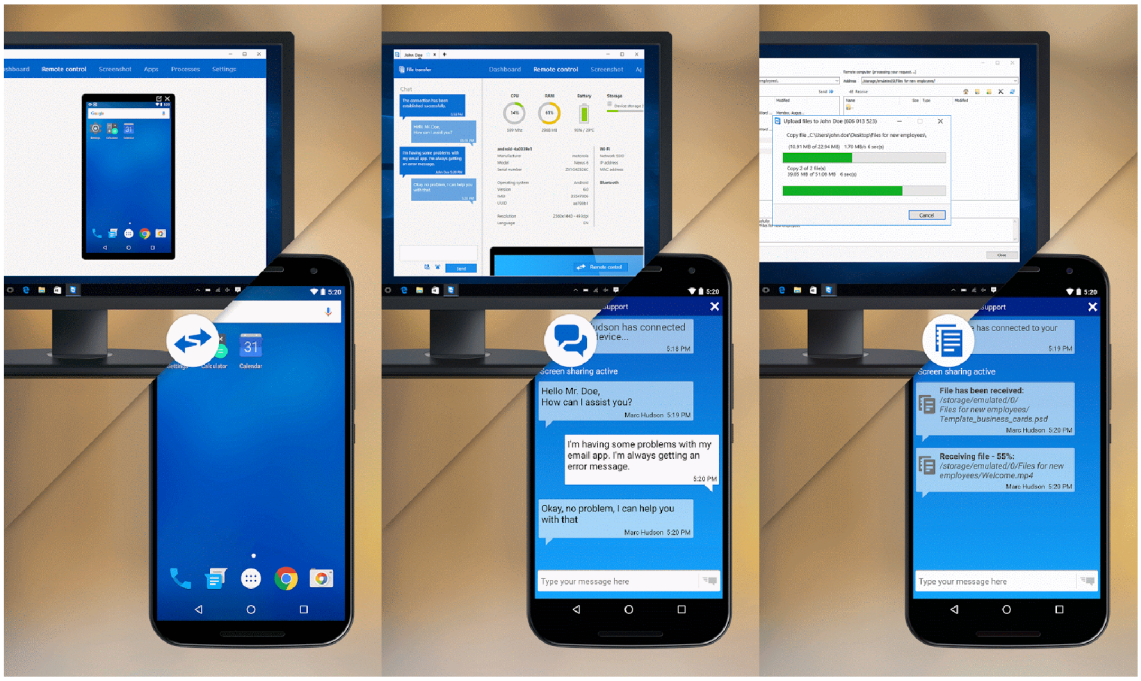
V tejto kapitole sa zamerám sa na riešenia troch najpoužívanejších operačných systémov – Android, iOS, Windows Phone.

Nepodarilo sa mi nájsť softvérové riešenie vzdialeného ovládania mobilných zariadení s operačným systémom Windows Phone. Aplikácia Remote Desktop od Microsoft podporuje vzdialené ovládanie počítača, nie opačným smerom. Podobnú funkčnosť poskytujú aj iné aplikácie.

Momentálne nie je na Apple App Store dostupná oficiálna aplikácia umožňujúca vzdialené ovládanie mobilných zariadení s operačným systémom iOS. V súčasnosti je táto funkcionalita v rozpore s pravidlami a podmienkami Apple pre vývoj aplikácií. Verejné Apple API neumožňuje prístup k potrebným povoleniam a aplikácie snažiace sa o vyhnutie týmto povoleniam, nie sú povolené v App Store. Je teda možné využit súkromné (private) API pre vytvorenie aplikácie so žiadanou funkcinonalitou, ale distribuovanie aplikácie bude musieť viesť mimo App Store. Používatelia tejto aplikácie by museli využit napríklad aplikáciu Cydia pre inštaláciu neoficiálnych aplikácií. Tá je však dostupná pri úprave operačného systému nazývanej jailbreak. Ide o získanie práv správcu operačného systému zariadenia k nadobudnutiu povolení k akciám, ktoré sú inak pre používateľa zakázané. Touto modifikáciou sa však používateľom zruší záruka na ich mobilné zariadenie, na koľko ide o nepovolený zásah do operačného systému, ktorým môžu používatelia poškodiť softvérovú alebo hardvérovú funkčnosť zariadenia. Aj zabezpečenia mobilných zariadení s operačným systémom Android sťažujú vývoj softvérových riešení ich vzdialeného ovládania. Na tejto platforme sú však dostupné aplikácie poskytujúce požadovanú funkcionalitu na Google Play. Niektoré však vyžadujú nadštandardné práva pre prístup k zabezpečeným nastaveniam a funkciám. Podobne ako pre iOS aj pre Android zariadenia platí, že získanie nadštandardných práv v systéme vedie k nepovelenej úprave systému, čo môže znamenať zrušenie záruky zariadenia.

2.3.1 TeamViewer

Spoločnosť s obrovskými skúsenosťami s vývojom a distribúciou softvérových riešení pre online spoluprácu a komunikáciu (okrem iného aj vzdialené ovládanie) [12]. Svedčí o tom aj viac ako 200 miliónov inštalácií vo viac ako 200 krajinách po celom svete. Softvér je dostupný vo viac ako tridsiatich jazykoch. Vzhľadom k tomu, že pri vzdialenom ovládaní sa jedná v podstate o real-time komunikáciu, kvalita prenosu závisí od prepojenia. Na svojich webových stránkach spoločnosť uvádza, že prepojenie je smerované vo vysoko výkonnostnej celosvetovej sieti serverov na základe geolokalizácie.



Obr. 2.1: Na obrázkoch je vidieť rôzne funkcie aplikácie QuickSupport. Vľavo je vidieť vzdialenú plochu, vstrede možnosť chatovania, vpravo prenos súborov.[2]

Pre sprostredkovanie vzdialenej podpory mobilných zariadení vyvíjajú aplikácie TeamViewer QuickSupport. Aplikácie sú dostupné na App Store a Google Play. Poskytujú možnosti ako chatovanie, obojsmerný prenos súborov, zobrazenie informácií o zariadení, poslanie Wi-Fi nastavení. Verzia pre iOS neposkytuje vzdialené ovládanie. Pripojenie je zabezpečené s 256-bitovým šifrovaním relácie AES. Verzia pre Android túto funkcionalitu zaručuje pre vybrané značky zariadení (Samsun, Sony, Asus (pre biznis zákazníkov), Lenovo, HTC, LG, ZTE, Huawei, Alcatel One Touch / TCL). Pre plnohodnotnú funkčnosť aplikácie je potrebné, aby používateľ doinštaloval aplikáciu QS Add-On. QS Add-On je dostupná na Google Play v rôznych verziách pre rôzne typy zariadení a je potrebné nainštalovať správnu verziu. Automatizovanie inštalácie je poskytnuté v aplikácii QuickSupport.

Funkcie QuickSupport pre Android:

- Konverzácia (Chat)
- Zobrazenie informácií o zariadení
- Vzdialené ovládanie

- Prenos súborov
- Zoznam aplikácií (odinštalovanie aplikácií)
- Zoznam procesov (zastavenie procesov)
- Nastavenie siete Wi-Fi
- Zobrazenie informácií diagnostiky systému
- Snímka obrazovky zariadenia v reálnom čase
- Kopírovanie informácií do pamäťovej schránky zariadenia

Pre operačný systém Android TeamViewer je okrem QuickSupport dostupná aj aplikácia TeamViewer Host. Poskutuje rovnakú funkcionalitu ako aplikácia QuickSupport. Rozdiel je v inicializácii spojenia. Aplikácia poskytuje trvalý prístup a vzdialené ovládanie k mobilným zariadeniam s operačným systémom Android v bezobslužnom režime. V zariadení sa nevyžaduje žiadne potvrdenie. Pripojenie je možné vytvoriť aj keď zariadenie nie je používané. Okrem smartfónov a tabletov aplikácia cieli aj na predajné miesta (POS), set-top boxy, informačné tabule, predajné automaty a podobné Android zariadenia.

Spojiť sa s mobilnými zariadeniami je možné aplikáciou TeamViewer pre počítače s operačnými systémami Windows alebo Mac.

2.3.2 ScreenConnect

Softvérové riešenia od ScreenConnect Software sú zamerané na vzdialenú podporu. Spoločnosť poskytuje softvér najmä pre vzdialenú podporu počítačov (medzi podporované operačné systémy patrí Windows, Mac, Linux), no majú vyvinuté aplikácie aj pre Android a iOS. Pre iOS podporujú kvoli bezpečnostným obmedzeniam operačného systému iba komunikáciu. Aplikácie pre vzdialenú podporu mobilných zariadení s operačným systémom Android poskytujú spravovanie nastavení, procesov, účtov, vzdialené ovládanie (formou vzdialenech plochy). Plná funkcionalita vzdialenej podpory Android zariadení je zaručená pre mobilné zariadenia od výrobcu Samsung. Pre ostatné Android zariadenia je vzdialená podpora realizovaná formou chatovania a odosielania snímok obrazovky.

2.3.3 Mobizen

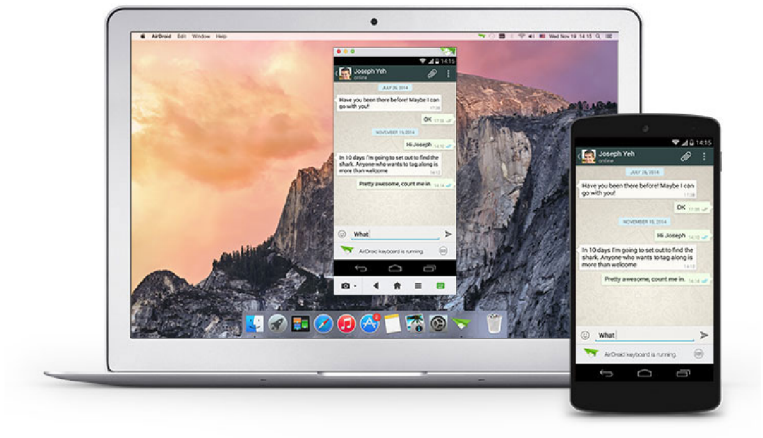
Aplikácia pre mobilné zariadenia s operačným systémom Android zameraná na nahrávanie obrazovky do videa a jeho zdieľanie na sociálne siete. Okrem toho však umožňuje vzdialené ovládania z aplikácie pre počítače s operačným systémami Windows, Mac alebo z webových stránok Mobizen. Na miesto vzdialenej podpory je kladený dôraz na poskytnutie rozhrania pre používanie mobilného zariadenia z počítača. Pri ovládaní je užitočná možnosť zadávania textových vstupov pomocou klávesnice.

2.3.4 AirDroid

Android aplikácia umožňujúca vzdialený prístup, ovládanie zariadenia, prenos súborov prostredníctvom počítača s operačným systémom Windows, Mac alebo prostredníctvom webového rozhrania. Aplikácia pre funkčnosť vzdialeného ovládania vyžaduje, aby mal používateľ operačného systému Android práva správcu. Zaujímavé je vypracovanie zobrazenia notifikácií zariadenia na počítači.

Obr. 2.2: Na obrázku je vidieť vzdialenú plochu v okne aplikácie ScreenConnect na operačnom systéme Windows.[4]

Obr. 2.3: Na obrázku je vidieť ukážku použitia služby Mobizen, vzdialenú plochu vo webovom prehliadači aj v okne aplikácie na operačnom systéme Windows.[6]



Obr. 2.4: Na obrázku je ukážka použitia služby AirDroid, vzdialenú plochu v okne aplikácie na operačnom systéme OS X.[8]

2.3.5 Droid VNC server

Zaujímavým riešením pre vzdialené ovládanie mobilných zariadení s operačným systémom Android je použitie systému VNC. Aplikácia je dostupná na Google Play, ale k jej funkčnosti potrebuje práva správcu systému. Pre ovládanie z počítača stačí spustiť klientskú aplikáciu pre vzdialený prístup podporujúcu VNC.

2.3.6 Iné riešenia

Ďalšími dostupnými aplikáciami zaoberajúcimi sa vzdialeným ovládaním Android zariadení sú Bomgar, RemoDroid, Remote Phone Access, ISL Light Remote Desktop.

Kapitola 3

Problematika vývoja mobilných aplikácií

Každý rok sa predajú stovky miliónov mobilných zariadení. Vďaka obrovskému množstvu zariadení je vývoj aplikácií pre mobilné platformy veľmi atraktívny. Samotný vývoj však komplikuje rozličnosť zariadení. Zariadenia sa totiž môžu odlišovať operačným systémom (ako typom tak aj jeho verziou), displejom (veľkosť, rozlíšenie, farby, dotykovou vrstvou), procesorom, veľkosťou operačnej pamäte, veľkosťou pamäte pre úložný priestor (interná, prípadne pamäťové karty), dostupnosťou kamery (prednej, zadnej), mikrofónu, klávesnice, tlačidiel, prostriedkov pre komunikáciu (mobilná sieť, Wi-Fi, Bluetooth, GPS, NFC, ...), senzorov (snímač odtlačkov prstov, akcelerometer, gyroskop, senzor priblíženia, kompas, barometer, snímač srdcového tepu, ...) a inými vlastnosťami. Vyvinúť a udržovať natívne aplikácie, ktoré by podporovali väčšinu zariadení nie je jednoduché.

3.1 Webová aplikácia

V dôsledku šetrenia zdrojov a skrátenia času pokým sa produkt dostane na trh sa často namiesto natívnych aplikácií vyvinie webová aplikácia[3]. Výhodné je implementovať najskôr mobilnú webovú aplikáciu pre rýchle zverejnenie do produkcie a následne vytvoriť natívne mobilné aplikácie, ktorých odkaz na stiahnutie sa umiestni na web. Webová aplikácia má určité limity a nie je možné, aby pre všetky účely nahradila natívne mobilné aplikácie. Pre jednoduchú prezentáciu firmy, poskytnutie základných informácií ako kontaktné údaje je to však postačujúce. Vďaka dodržiavaniu webových štandardov je webové aplikácie možné spustiť na rôznych systémoch a zariadeniach cez webový prehliadač.

Obrovskou výhodou je jednoduchšia a rýchlejšia distribúcia aplikácie. Stačí aby mal používateľ prístup k internetu, webový prehliadač a prístupnú webovú adresu aplikácie (verejná adresa alebo spojené pripojenie cez virtuálnu súkromnú sieť – VPN).

Vývoj webovej aplikácie je náročnejší pokiaľ má byť aplikácia optimalizovaná aj pre mobilné platformy. Aj napriek tomu je to jednoduchšie ako vývoj natívnych aplikácií pre rozličné mobilné platformy.

Mobilné zariadenia môžu mať menší displej, na ktorom musí byť zabezpečené správne zobrazenie obsahu. Ďalším dôležitým prvkom je navigácia medzi stránkami, ktorá by mala byť ľahko dostupná a dobre ovládateľná. Okrem menšej plochy musí byť pri vývoji zohľadnené, že na mobilnom zariadení používateľ pravdepodobne nebude mať pripojenú myš, ktorou by stránky ovládal. Na dotykovom displeji by bez myši napríklad bolo obtiažne

ovládať vysúvacie menu (obzvlášť ak by išlo o viac úrovňové menu). Ďalším nežiadaným prípadom by mohla byť požadovavka po používateľovi pre stisknutie tlačidla na klávesnici. V súčasnosti mobilné zariadenia väčšinou hardvérovú klávesnicu neobsahujú. Preto ak by aplikácia vyžadovala jej použitie, musí sa postarať o vyvolanie zobrazenia softvérovej klávesnice (pre HTML elementy, pri ktorých je očakávaný vstup z klávesnice sa o jej vyvolanie automaticky postará webový prehliadač). Dizajn musí počítať s tým, že mobilné zariadenia môžu byť výpočtovo výrazne slabšie, preto je vhodné nevyťažovať výkon vykresľovaním zbytočných zložitých efektov a animácií.

Mobilné zariadenie často pristupuje k webovému obsahu prostredníctvom mobilného internetu. Mobilný internet je v niektorých lokalitách pomalší (alebo zariadenie nemusí podporovať dostupné rýchlejšie rozhranie). Navyše toto pripojenie k internetu pravdepodobne bude viazané na dátový tarif obmedzeným na určitú veľkosť stiahnutých dát. Preto by mobilná verzia webovej aplikácie mala mať zredukovanú veľkosť potrebných dát pre stiahnutie (menšie rozlíšenia obrázkov, menej HTTP požiadaviek pomocou CSS sprite, menšia veľkosť pomocou vektorových formátov – SVG, písmo obsahujúce ikony, správne nastavená cache – nesťahovať znova získané dáta z predošlých prístupov).

Mnohé webové aplikácie riešia optimalizáciu zobrazenia pre mobilné platformy inou verziou aplikácie. Mobilná verzia aplikácie má inú webovú adresu (URL) - typicky má adresa mobilnej verzie webovej aplikácie subdoménou m. Pre zobrazenie správnej verzie webových stránok je nutné na serveri detekovať typ zariadenia a podľa daného typu zobraziť adekvátnu verziu stránok. Okrem toho je tento prístup nevýhodný v tom, že sa musia udržiavať dve separátne verzie webovej aplikácie.

3.1.1 Responzívny dizajn

Vďaka responzívnemu dizajnu[2] sa webový obsah dokáže prispôsobiť rôznym veľkostiam obrazovky za použitia jedného zdrojového kódu. Responzívny dizajn webu kladie dôraz na dva hlavné komponenty:

- Flexibilita – HTML elementom je namiesto absolútnych rozmerov prideľovaná relatívna veľkosť vďaka čomu sa rozloženie dokáže lepšie prispôsobiť veľkosti obrazovky
- Podmienené zobrazenie (CSS media queries) – možnosť pridaná v CSS3 pre priradenie CSS vlastností HTML elementom na základe parametrov zobrazenia (napríklad veľkosť obrazovky)

3.1.2 Mobile first

Prístup k responzívnemu dizajnu, pri ktorom sa najskôr navrhuje webová aplikácia pre mobilné zariadenia a až potom pre zariadenia ostatné (notebooky, stolné počítače, ...).

3.2 Natívna aplikácia

Motiváciou pre vytvorenie natívnych aplikácií pre rôzne platformy je často konkurenčná výhoda alebo vytvorenie komunikačného kanálu s používateľmi. V niektorých prípadoch sa požadovaná funkčnosť aplikácie nedá dosiahnuť prostredníctvom webovej aplikácie a je nutné vyvinúť natívne mobilné aplikácie. Medzi iné dôvody pre ich vytvorenie patrí rýchlosť aplikácie (ovládanie v aplikácii, zobrazenie dát, ...), vzhľad optimalizovaný na konkrétnu

platformu použitím jej špecifických grafických prvkov pre pohodlnejšie používanie aplikácie (ak aplikácia dodržuje odporučenia k dizajnu aplikácie, zvyklosti pre danú platformu, používateľom sa aplikácia bude dobre ovládať na základe podobného ovládania známeho z iných aplikácií).

Nevýhodou natívnych aplikácií je, že musia byť do zariadení nainštalované. Musí byť teda zabezpečená distribúcia aplikácie. Distribúcia aplikácií na väčšine mobilných zariadení je vyriešená prostredníctvom obchodu s aplikáciami. Ide v podstate o predinštalovanú aplikáciu, ktorá je súčasťou systému. Obchod je špecifický pre konkrétnu platformu. Obchody nezdieľajú aplikácie naprieč platformami, keďže v každom obchode sú dostupné aplikácie pre danú platformu.

Dôvody pre vytvorenie natívnej aplikácie:

- Využitie možností obchodov s aplikáciami rôznych platforiem (distribúcia, platby, marketing, štatistiky, ...)
- Vytvorenie hry alebo inej graficky náročnej aplikácie
- Potreba väčšieho výkonu
- Prístup k účtom na zariadení
- Prístup k telefónnemu zoznamu
- Prístup ku galérii (multimédiá)
- Prístup k informáciám o zariadení
- Možnosť meniť nastavenia zariadenia
- Použitie hardvérových funkcií (mikrofón, kamera, senzory, ...)
- Použitie telefónu, SMS
- Aplikácia má bežať aj na pozadí
- Aplikácia sa má spustiť pri štarte zariadenia
- Počas behu aplikácie sa telefón nemôže uspať
- Zobrazenie push notifikácií (príjem dát zo servera bez toho, aby klient – zariadenie poslalo požiadavku)

Nie je vylúčené, že v špecifických prípadoch bude aplikácia cielená iba na jednu platformu. Omnoho typickejšie je však sprístupniť aplikácie na stiahnutie na viacerých platformách. Cieľom je získať čo možno najväčšiu skupinu potenciálnych používateľov.

Vývoj natívnych aplikácií pre jednotlivé platformy zvlášť je náročnejší na zdroje. Ide totiž o vytvorenie viacerých aplikácií. Po dokončení aplikácií bude väčšia réžia pri ich údržbe, nakoľko každá zmena bude musieť byť vykonaná na všetkých aplikáciách. Navyše aplikácie môžu byť prepojené a komunikovať cez určitý komunikačný protokol. V tom prípade je potrebné vynaložiť väčšie úsilie na vytvorenie robustnejšieho návrhu, aby počítal s prípadnými rozdielmi rôznych verzií na rozličných platformách (napríklad v dôsledku pridania, odobratia, či úpravy nejakej funkcie. Tieto problémy minimalizuje použitie riešení pre multiplatformný vývoj.

3.2.1 Multiplatformný vývoj mobilných aplikácií

Z dôvodov ušetrenia zdrojov pri vývoji, udržovaní aplikácie a jej rýchlejšie uvedenie na trh vznikli nástroje, ktoré si kladú za cieľ umožniť programátorovi z jedného zdrojového kódu, generovať aplikácie pre rôzne platformy. Dnes pre daný účel existuje viacero frameworkov tretích strán. Podľa spôsobu ako je realizovaný prevod inštrukcií kódu, aby boli vykonané na rozličných platformách sa delia na interpretované, prekladané a webové. Frameworky, ktoré používajú prekladanie, umožňujú vytvoriť aplikáciu v jednom jazyku. Pri preklade sa jazyk nahradzuje programovacím jazykom cieľovej platformy spolu s využitím softvérového balíku SDK (software development kit) danej platformy. Ďalším typom sú frameworky využívajúce web. Vytvorená webová aplikácia je vložená do natívnej aplikácie, v ktorej je spracovávaná ako webový obsah. Tieto frameworky používajú pluginy pre sprostredkovanie funkcinality zariadenia a využitie natívnych grafických prvkov pre vytvorenie lepšieho používateľského prostredia. Posledný typ framoworkov používa pre vykonávanie inštrukcií kódu na rôznych platformách interpret. Zdrojový kód je spolu s interpretom pre danú platformu preložený do natívnej aplikácie.

NativeScript

NativeScript je nový open source vývojový systém pre tvorbu multiplatformných mobilných aplikácií v jazyku JavaScript. Pre dizajn vzhľadu sa používajú kaskádové štýly (CSS) a jazyk XML V čase tvorenia tejto práce sú plne podporované platformy Android a iOS. V súčasnosti sa pripravuje podpora systému Windows 10 Mobile. NativeScript používa pre kompiláciu a vykonávanie kódu v JavaScripte V8 JavaScript Engine na Androide a JavaScriptCore na iOS.

Hlavným rozdielom NativeScriptu od konkurenčných riešení je sprostredkovanie priameho prístupu k natívnym prvkom cieľovej platformy.

Apache Cordova

Cordova je open source framework umožňujúci vytvorenie natívnych aplikácií z kódu v jazykoch HTML, JavaScript a kaskádových štýlov (CSS). Medzi odporované platformy patria Android, iOS, Windows Mobile, Windows Phone, BlackBerry 10, Ubuntu, Firefox OS, Fire OS. Funkčnosť zdrojového kódu naprieč platformami je realizovaná prostredníctvom komponenty pre zobrazenie webových stránok (web view). Natívna aplikácia zobrazí túto komponentu, ktorá načíta zdrojové kódy a zobrazí stránku. Takto vytvorené aplikácie sa nazývajú hybridné.

Cordova poskytuje prístup k viacerým funkcionalitám zariadenia ako napríklad kamera, mikrofón, kontakty, dialógy, vibrácia, či geolokalizácia. Pre prístup k funkcionalitám zariadenia, ktoré nie sú priamo dostupné v Cordove, je možné vytvoriť plugin. Plugin obsahuje natívny kód spárovaný s knižnicou v jazyku JavaScipt. Pre Cordovu komunita vyvinula stovky pluginov, takže často používané prvky už sú pripravené pre integráciu.

Distribúcie

Adobe PhoneGap PhoneGap je pôvodnou distribúciou frameworku Cordova, ktorú kúpila firma Adobe. Zostáva naďalej zadarmo a open source. PhoneGap ponúka množstvo nástrojov pre vývoj. PhoneGap Build je komerčná služba umožňujúca vygenerovanie natívnych aplikácií bez nutnosti inštalácie knižníc a nástrojov pre príslušné platformy. Pre

používanie je potrebné vytvorenie účtu. Službu je možné používať s obmedzenými možnosťami zadarmo (iba jedna súkromná aplikácia, obmedzenie veľkosti aplikácie, ...). Zdrojové kódy sa nahrajú na server v zip archíve alebo je možné prepojiť službu s Git účtom a prekladať aplikáciu priamo z repozitára.

Ionic Ionic je open source framework pre vývoj hybridných aplikácií. Poskytuje HTML, CSS a JavaScript komponenty optimalizované na mobilné zariadenia, prácu s gestami a nástroje pre vytváranie interaktívnych aplikácií. Ionic používa JavaScriptový framework AngularJS. Je výkonovo optimalizovaný, k čomu využíva minimálnu prácu s DOM a hardvérovú akceleráciu. Ionic vyvíja cloud platformu, v ktorej poskytuje správu aplikácií a integráciu služieb (ako napríklad push notifikácie, sociálne siete, ...).

Xamarin

Xamarin[4] je platforma umožňujúca programovať multiplatformné iOS, Android, Windows Phone a Windows aplikácie v jazyku C#. Xamarin je postavený na open source projekte Mono. Ide o projekt umožňujúci fungovanie frameworku .NET na rôznych platformách. Vo verzii Xamarinu pre Android je Mono pre Android a iOS funguje na MonoTouch. Pomocou nich C# pristupuje k natívnemu Android a iOS aplikačnému programovému rozhraniu (API). Vývojár teda môže používať natívne používateľské rozhranie, animácie, grafiku, notifikácie, hardvérové komponenty (fotoaparát, mikrofón, senzory, ...).

Zmeny API nových verzií platforiem nie sú dostupné hneď, keďže najskôr musia byť dostupné prostredníctvom Xamarinu. Vývojári teda vždy musia počkať na vydanie novej verzie.

Xamarin Forms umožňuje vytvoriť multiplatformné používateľské rozhranie. Je to výhodné pri menších aplikáciách, pokiaľ nie sú špecifické požiadavky na rôzny vzhľad na rozličných systémoch. Xamarin Forms obsahuje dosť chýb a v niektorých prípadoch je prostredie pomalšie ako natívne. Pokiaľ sa jedná o komplexnejšiu aplikáciu je vhodné programovať používateľské rozhranie zvlášť pre každú platformu.

Do 31. marca 2016 bola najnižšia licencia pre vývoj aplikácií 25 dolárov za mesiac. Aj kvôli tomu mal Xamarin malú komunitu v porovnaní s inými open source systémami. To malo za následok menej dostupných knižníc tretích strán a menšiu podporu v komunite. V spomínaný deň sa zmenil platobný model a pre študentov a jednotlivcov je teraz Xamarin dostupný zadarmo. To by mohlo mať za následok väčšiu popularitu Xamarinu.

3.2.2 Vývoj pre konkrétnu platformu

Vývoj mobilných aplikácií pre každú platformu zvlášť je nákladnejší na vývoj, údržbu pokiaľ majú byť podporované viaceré platformy. Je to spôsobené tým, že aplikácie sú vytvárané separátne. Implementácia je o to zložitejšia ak aplikácie medzi sebou komunikujú. Je dôležité eliminovať problémy spôsobené rozdielmi platforiem, dodržovať dohodnuté rozhranie. Pridanie nového rozhrania alebo zmena v tom stávajúcom musí ošetriť to, že nové verzie so zmeneným rozhraním nie je možné sprístupniť na rôznych platformách naraz.

Windows

V dobe tvorby tejto práce je podľa mňa pre vývoj natívnych aplikácií pre platformu Windows vhodné zvoliť Windows 10 Universal Windows Platform (UWP). UWP aplikácie je

možné používať na všetkých zariadeniach so systémom Windows 10. UWP aplikácie zľahčujú vývoj použitím jedného API, jednej aplikácie a jej distrubúciu v jednom obchode (Windows Store).

Aplikácie môžu byť zhotovené prostredníctvom jazyka JavaScript, jazykov .NET frameworku (C#, Visual Basic) alebo jazykov C/C++ vďaka rozšíreniu C++ nazvaného C++/Cx. JavaScript používa pre tvorbu používateľského rozhrania jazyky HTML, CSS, zatiaľ čo ostatné jazyky používajú Extensible Application Markup Language (XAML).

Pre kompilovanie UWP aplikácií je nutné mať operačný systém Windows a nainštalovať si integrované vývojové prostredie (IDE) Visual Studio. Pred verziou Visual Studio 2015 bola dostupná iba spoplatnená verzia. Microsoft s verziou 2015 sprístupnil verziu Visual Studio Community Edition 2015 zadarmo pre jednotlivcov, študentov, akademický výskum, výučbu a malé profesionálne tímy. Pre vývoj je potrebný balík Windows 10 SDK (Visual Studio ho obsahuje).

Distribúcia hotovej UWP aplikácie je realizovaná prostredníctvom Windows Store. Je nutné vytvoriť si účet pre vývojára a zaplatiť jednorázový poplatok (pre jednotlivca 19$, pre firmu 99$). Pre uverejnenie v obchode je potrebné nahrať preloženú aplikáciu a vyplniť príslušné údaje.

iOS

K vyvíjaniu aplikácií pre iOS[5] je nevyhnutný operačný systém OS X s nainštalovaným integrovaným vývojovým prostredím (IDE) X Code. Xcode je dostupný na stiahnutie v obchode aplikácií Mac App Store. K vývoju pre najnovšie verzie iOS je nutné pravidelne aktualizovať Xcode a v dôsledku toho aj aktualizovať OS X. Súčasná verzia Xcode 7.3.1 vyžaduje minimálne OS X 10.11 (El Capitan). OS X je určený pre počítače MacBook, iMac od firmy Apple, ktoré sú potrebné k jeho behu. Existujú spôsoby inštalácie na iných počítačoch, ale žiadna nie je oficiálna a nezaručujú chod systému bez problémov. Ďalšou neoficiálnou možnosťou je využiť virtualizačný softvér (napríklad VMware Workstation, VirtualBox) a nainštalovať OS X na virtuálny stroj.

Distribúcia aplikácií na iOS je realizovaná cez obchod aplikácií App Store. K nahrávaniu aplikácií do App Store je požadovaný vývojársky účet s ročným členským poplatkom (štandardný účet 99$, podnikový účet 299$). Okrem toho člen získa prístup k beta verziám operačných systémov, rozširujúce knižnice, distrubúciu aplikácie testerom, podporu. Pred verziou Xcode 7 bez členstva nešlo inštalovať vytvárané aplikácie na reálne zariadenia (iba ak by bolo zariadenie upravené zrušením obmedzením a získania vyšších (root) práv – iba so stratou záruky). Aplikácie je možné testovať na iOS simulátore (dostupný v Xcode). Má však isté obmedzenia (napr. nie je dostupná kamera) a preto nie je možné všetky aplikácie testovať na simulátore. Pred zverejnením aplikácie v obchode je doporučené otestovať ju na podporovaných reálnych zariadeniach.

Až do predstavenia jazyka Swift v júni 2014 sa pre vývoj iOS aplikácií používal programovací jazyk Objective-C. Jazyk Swift bol vytvorený firmou Apple pre svoje produkty a od verzie 2.2 (3. 12. 2015) je open source. Aplikácie sú od Xcode verzie 6 a vyššie prekladané kompilátorom LLVM používajúc Objective-C behové prostredie (runtime), ktoré umožňuje aby bol kód v jazykoch C, C++, Objective-C, Swift a bežal ako jeden program.

Do verzie iOS 9 mohla byť v jeden čas zobrazená na displeji iba jedna aktívna aplikácia. S touto verziou bola pre modely iPad Pro, iPad Air 2 a iPad mini 4 predstavená funkcia Split View. Ide o rozdelenie zobrazovanej plochy na dve polovice, pričom na každej je zobrazená iná aplikácia. V aplikácii však musí byť naprogramovaná podpora Split View. Od verzie iOS

4 môžu po stlačení tlačidla domov (Home) aplikácie bežať na pozadí. Beh na pozadí je v iOS striktne obmedzený na niektoré situácie, pri ktorých je to nutné a aj pri nich sú obmedzenia na to, čo môže aplikácia vykonávať. Je veľmi pravdepodobné, že sieťové pripojenie aplikácie bude uzavreté. Každá aplikácia má iba jedno okno, prostredníctvom ktorého komunikuje s používateľom. Aplikácie môže používateľ prepínať dvojitým stlačením tlačidla domov.

Na rozdiel od programov spúšťaných na operačných systémoch stolových počítačov sú na iOS presne vymedzené zdroje, ku ktorým aplikácia môže pristupovať. Každá aplikácia má vyhradený svoj vlastný priestor (sandbox), v ktorom má povolené čítanie a zapisovanie. Aplikácie v tomto priestore ukladajú predvoľby, dokumenty a iné dáta potrebujúce perzistentné uloženie. Nie je povolené pristupovať k nastaveniam a funkciám vyžadujúcim administrátorské práva (root).

iOS aplikácie sú limitované veľkosťou obrazovky ako aj iné mobilné zariadenia. Dizajnér s tým musí počítať a navrhovať podľa toho používateľské rozhranie. Aj napriek veľkému rozlíšeniu moderných displejov (iPhone 6 Plus 1080 x 1920) sú pixely na fyzicky pomerne malej ploche (iPhone 6 Plus 5,5 palca). Je nutné brať do úvahy, že zariadenie je ovládané dotykom prsta. Dotyk má určitú plochu, ktorá musí byť zohľadnená v rozhraní, aby ovládacie prvky neboli príliš blízko seba a dali sa jednoducho ovládať. Oproti iným platformám pre mobilné zariadenia má iOS obrovskú výhodu v tom, že variant zariadení a teda aj displejov nie je veľa, a preto sa dajú pri návrhu a vývoji zohľadniť ich parametre.

Systém iOS bol vytvorený tak, aby kládol veľký dôraz na používateľa a jeho komfort pri používaní systému. Samotný optimalizovaný systém nestačí, keďže používateľ je veľmi často v prostredí aplikácie a preto systém má nemalé nároky na vývojárov aplikácií a ich používateľské rozhranie. Aplikácia musí zobraziť úvodnú obrazovku a mať načítaný obsah čo najskôr (v niekoľkých sekundách). O veľa dlhšie aplikácii nemôže trvať ani jej ukončenie. V prípade, že používateľ stlačí tlačidlo domov, aplikácia sa minimalizuje a musí urýchlene ukončiť prácu a uložiť dosiaľ neuložené dáta. Ak by jej to totiž trvalo dlhšie ako 5 sekúnd, systém aplikáciu ukončí a všetky neuložené dáta sa stratia. Existuje API, ktoré umožňuje získať dodatočný čas pre prácu na pozadí, ale znova platí, že je to vymedzené pre špecifické situácie.

Spoločnosť Apple kontroluje splnenie svojich predpisov a podmienok pre vývoj iOS aplikácií pri nahrávaní do obchodu App Store. Každá aplikácia a aj každá jej aktualizácia je pred povolením k zverejneniu v obchode odoslaná k preskúmaniu. Aplikácie čakajú v rade, kým sa nedostanú na rad aby ich za to zodpovedná osoba odkúšala, skontrolovala a rozhodla o schválení. Ide o zdĺhavý proces, kvôli ktorému nemusí byť aktualizovaná verzia odstraňujúca kritickú chybu používateľom dostupná aj týždeň a viac od nahrania do App Store.

Android

Android je operačný systém určený primárne pre mobilné zariadenia. V roku 2005 spoločnosť Google kúpila spoločnosť, ktorá ho vyvíjala. Odvtedy sa stará o vývoj Google. Obrovskou výhodou je, že po vydaní novej verzie sú sprístupnené zdrojové kódy. Android má veľkú open source komunitu vývojárov, ktorí používajú Android Open Source Project (AOSP) pre vytvorenie vlastných distribúcií Androidu. Prináša to však so sebou aj nevýhody. Výrobcovia hardvéru si upravujú systém na mieru, čo môže viesť v určitých krajných situáciách k rozličnému chovaniu a prípadným chybám. Veľkou nevýhodou je pomalá alebo chýbajúca distribúcia aktualizácií systému. Po uvedení novej verzie systému na trh musia výrobcovia modifikovaných distribúcií zabezpečiť kompatibilitu a pripraviť novú verziu

ich distribúcie. Situáciu ešte viac zhoršuje, že jeden výrobca má rôzne verzie distribúcie pre rôzne modely zariadení. Modifikované verzie sú teda pripravené so značným oneskorením. V horšom prípade sa k niektorým modelom aktualizácie nepripravujú vôbec. Pri aktualizáciách systému nejde iba o vylepšenie používateľského rozhrania, pridanie nových funkcionalít, optimalizáciu knižníc, ale často aj o zabezpečenie systému. Obrovské množstvo Android zariadení teda má bezpečnostné diery, ktoré už sú v novšej verzii opravené, ale bežný používateľ sa k oprave nedostane.

3.3 Vývoj Android aplikácií

V tejto časti sú popísané prostriedky pre vývoj Android aplikácie, štruktúra a možnosti aplikácie. Pri základných komponentoch je stručne uvedené k čomu slúžia a ako sa používajú.

3.3.1 Príprava prostiedkov pre vývoj aplikácie

Obrovskou konkurenčnou výhodou Androidu voči ostatným platformám je dostupnosť vývoja na najpoužívanejších operačných systémoch – Windows, Linux, Mac. Android poskytuje pre uľahčenie vývoja aplikácií integrované vývojové prostredie (IDE) Android Studio. Výhodou Android Studia je, že sa nejedná o IDE pre vývoj rôznych projektov, ale o IDE presne pre vývoj Android aplikácií. Vývojár môže využiť rôzne nástroje pre ladenie a optimalizáciu aplikácií. V minulosti Android poskytoval oficiálnu podporu pre IDE Eclipse vo forme zásuvného modulu Android Development Tools (ADT). Dnes už Android oficiálnu podporu k iným IDE neposkytuje, takže vývojári používajúci NetBeans, Eclipse sú odkázaní na moduly vyvíjané komunitou.

Väčšina aplikácií pre Android je programovaných v jazyku Java. Systém Android spravuje aplikácie cez balíky Application package file (APK). V balíkoch je bajtkód Dalvik. Bajtkód vznikne prekladom zdrojových kódov v jazyku Java. Pre vývoj Java aplikácií je potrebné nainštalovať vývojovú sadu jazyka Java – Java development kit (JDK) – dostupú na stránkach firmy Oracle[1].

Knižnice s rozhraním pre programovanie aplikácií a vývojárske nástroje potrebné na zostavenie, testovanie a ladenie aplikácií je možné získať stiahnutím a nainštalovaním cez nástroj Android SDK Manager.

Užitočným vývojovým nástrojom je emulátor Android zariadení. Prostredníctvom emulátoru je možné vyvíjať aplikácie bez dostupného fyzického zariadenia a testovať aplikácie na rôznych virtuálnych zariadeniach (Android Virtual Device – AVD). Nástroj AVD Manager umožňuje vytvárať, spúšťať a spravovať AVD. Vytváranie a upravovanie AVD umožňuje meniť nastavenia zariadenia, ako napríklad verzia systému Android, hardvérové nastavenia.

3.3.2 Základy programovania pre Android

Aplikácie sú tvorené rôznymi komponentami. Komponenty sú spúšťané z iných komponentov použitím tzv. záujmu (intent). Základom operačného systému Android je Linuxové jadro [7]. Ide o viacužívateľský systém, pričom jednotlivé aplikácie sú brané ako používatelia systému. V základnom nastavení má aplikácia spustený vlastný proces, ktorému je priradené unikátne používateľské číslo. Proces je spúšťaný systémom pri potrebe vykonávať nejakú časť aplikácie. K ukončeniu procesu môže dôjsť, ak už nie je ďalej užitočný alebo pri

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html

nedostatku pamäte v rámci uvoľňovania pamäte systémom. Aplikácie bežia z hľadiska kódu izolovane, pretože každému procesu je pridelený vlastný virtuálny stroj (virtual machine).

Manifest aplikácie

Aplikácia musí obsahovať v koreňovom priečinku projektu súbor `AndroidManifest.xml`[1]. Manifest obsahuje základné informácie o aplikácii. Komponent musí byť deklarovaný v manifeste aby systémom mohol byť spustený.

Niektoré z položiek, ktoré manifest môže obsahovať:

- Názov Java balíku aplikácie.
- Deklarácie komponentov použitých v aplikácii. Na základe informácií o komponentoch systém vyhodnotí podmienky nutné pre spustenie komponentu.
- Deklarácie povolení, ktoré aplikácia potrebuje k prístupu k chráneným častiam aplikačného rozhrania a povolení ostatných aplikácií, potrebné k prístupu k danej aplikácii.
- Deklarácie použitých vlastností systému (napríklad bluetooth, kamera, ..)
- Deklaráciu minimálnej verzie systému požadavú aplikáciou.

Komponenty Android aplikácie

Aplikácie v Androide sú zložené zo základných komponentov, ktoré pomáhajú aplikáciu definovať. Cez komponent môže systém pristupovať k aplikácii. Každý komponent je unikátna časť. Niektoré komponenty na sebe môžu byť závislé. Sú podľa účelu delené na štyri rôzne typy – aktivity, služby, poskytovatelia obsahu, zámery. Životný cyklus rôznych typov je odlišný.

Aktivity

Základná stavebná časť aplikácie reprezentujúca obsah jednej obrazovky[7]. Aplikácia môže byť zložená z viacerých aktivít pre rôzny účel. Aktivity sú na sebe nezávislé. V systéme Android sa zvyčajne aktivity pomerne často otvárajú a zatvárajú vzhľadom k tomu, že typicky jedna aktivita slúži na jeden účel. V aktivite je možné spustiť aktivitu inej aplikácie.

Služby

Služby sú komponenty umožňujúce vykonávanie dlhotrvajúcich operácií na pozadí[2]. Neposkytujú používateľské rozhranie. Môžu byť využité na komunikáciu medzi procesmi. Služba môže pokračovať v behu aj potom ako používateľ spustí inú aplikáciu. Príkladmi použitia môžu byť vstupno-výstupné operácie alebo prehrávanie hudby.

[2] http://developer.android.com/guide/components/services.html

Správca obsahu

Správca obsahu (trieda `ContentProvider`[3]) poskytuje prístup k štrukturovanej sade dát. Okrem zapúzdrenia dát poskytujú mechanizmus pre definovanie bezpečnosti dát. Vytvárajú štandardné rozhranie pre poskytnutie dát procesu iným procesom. K prístupu dátam cez poskytovateľa obsahu príde poskytovateľovi požiadavka o dáta. Poskytovateľ vykoná požadovanú akciu a vráti výsledok.

Aplikácia pristupuje k dátam správcu obsahu cez klientský objekt triedy ContentResolver. Ten poskytuje metódy pre prácu s dátami. Metódy sú nazvané rovnako ako v inštancii správcu obsahu (objekt konkrétnej podtriedy `ContentProvider`).

Zámery

Zámer je objekt, ktorým môže aplikácia poslať cez systém požiadavku k vyžiadaniu akcie inej aplikácie[4]. Nezobrazujú používateľské rozhranie. Môžu v prípade výskytu špecifickej udalosti upozorniť používateľa zobrazením notifikácie. Zámery môžu vytvárať systém (napr. nízky stav batérie), ale aj aplikácie. Aktivity môžu pomocou nich spúšťať iné aktivity, služby alebo posielať správy typu `broadcast`.

Spúšťanie aktivít

Aplikácie sa skladajú z aktivít (delené zvyčajne podľa účelu). Aktivity sa spúšťajú pomocou zámerov[4]. Zámer môže byť vytvorený ako explicitný alebo implicitný. V explicitnom je určený konkrétny komponent, ktorý systém okamžite spúšťa. Implicitný zámer deklaruje iba akciu. Vyhodnotenie, ktorý komponent sa vyberie pre obslúženie zámeru, je ponechané na systém. Systém vyhľadá aktivity porovnávaním obsahu zámeru s filtrami zámerov. Filtre sú umiestnené v manifestoch iných aplikácií. V prípade nájdenia komponentu pre daný zámer, systém spúšťa komponent a predá mu zámer. V prípade nájdení viacerých komponent, systém zobrazí voľbu s možnosťami. Používateľ si vyberie, ktorým komponentom (aplikáciou) chce vykonať danú akciu. Do zámeru sú vložené dáta. Do explicitného zámeru je vkladaný názov komponentu. Do implicitného sa vkladá akcia. Akcia je identifikovaná reťazcom určujúcim akciu. Pre určenie akcie sú dostupné vstavané akcie, ale je možné aj vytvorenie vlastných akcií. V zámere je typ dát popísaný pomocou objekt typu URI alebo typom dát MIME alebo ich kombináciou. Typ MIME je tiež možné vytvoriť vlastný alebo použiť vstavaný. Pre vykonanie spustenia aktivity sa používa metóda `startActivity`, ktorej je argumentom predaný zámer. Niekedy je potrebné vedieť výsledok vykonávania spúšťanej aktivity, čo je možné pomocou spustenia metódou `StartActivityForResult`. Výsledok je možné získať preťažením metódy `onActivityResult`.

Používateľské rozhranie

Používateľské rozhranie aplikácie je všetko, čo používateľ vidí a s čím môže ovládať aplikáciu. Android poskytuje veľa vstavaných komponentov, z ktorých je možné vytvoriť rozhranie. Pre definovanie rozvrhnuia týchto komponentov slúži layout. Základnou triedou používateľského rozhrania je trieda `View`[5]. Od nej sú odvodené všetky elementy používateľského rozhrania aplikácie. Triedy odvodené od `View` možu byť zoskupené prostredníctvom

[3] http://developer.android.com/guide/topics/providers/content-provider-creating.html
[4] http://developer.android.com/guide/components/intents-filters.html
[5] http://developer.android.com/guide/topics/ui/overview.html

triedy `ViewGroup`, ktorá je odvodená od triedy `View`. Zoskupovaním objektov týchto tried vzniká stromová hierarchia.

Layout

Definuje štruktúru objektov používateľského rozhrania[6]. Layout môže vzniknúť vytvorením deklarácie elementov rozhrania v súbore formátu XML. Druhou možnosťou je vytvárať elementy ako inštancie tried `View`. Typicky sa pre statickú (predom známu) štruktúru používa rozloženie deklarované v súbore formátu XML. V zdrojovom kóde je možné počas behu aplikácie vyhľadať objekty reprezentujúce prvky rozhrania a meniť ich vlastnosti. V špecifických prípadoch sa v kóde vytvárajú nové objekty, ktoré sa do rozloženia pridajú.

Deklarovaním layoutu v súbore formátu XML sa oddelí definícia vzhľadu aplikácie od definície chovania v zdrojovom kóde. Vďaka tomu je možné zmeniť vzhľad bez nutnosti meniť kód (pokiaľ sa nemení typ objektu, ktorý sa používa v kóde) [6]. Pre rôzne konfigurácie (napr. orientácie obrazoviek, rôzne rozlíšenia, a odlišné jazyky) môžu byť definované špecifické layouty. Priradenie špecifického layoutu pre danú konfiguráciu je riešené pomocou tzv. kvalifikátorov. Priradenie rozloženie prvkov v súbore formátu XML je priradené konkrétnemu prvku v kóde aplikácie prostredníctvom príslušných metód (napr. `setContentView` pre aktivitu).

Pre deklarovanie prvkov používateľského rozhrania v súboroch typu XML platí, že súbor začína uvedením XML hlavičky. Súbor s layoutom musí obsahovať koreňový element. Môže ním byť objekt triedy `View` alebo `ViewGroup`, v ktorom môžu byť vložené ďalšie objekty. Názvy a vlastnosti elementov sú pomenované podľa štruktúry a názvov tried a metód. Pre XML formát trieda objektu predstavuje názov elementu a atribúty objektu zodpovedajú názvom atribútov elementu. K jednoznačnej identifikácii objektov `View` slúži identifikátor – atribút ID. Layouty sú ukladané v súbore s príponou .xml v priečinku projektu `res/layout` (prípadne v priečinku s príslušným kvalifikátorom – napríklad `res/layout-port` pri rozložení pre orientáciu obrazovky na výšku).

Prvky sú vkladané do kontajnerov, ktoré ich organizujú do rozličných štruktúr. Kontajnery sú odvodené od triedy `ViewGroup`. Najpoužívanejšie typy kontajnerov:

- Relatívny kontajner – Pre usporiadanie prvkov používa relatívne pozície. Prvky môžu byť umiestnené relatívne k iným prvkom v rámci toho istého kontajneru alebo ku kontajneru.
- Lineárny kontajner – Lineárne organizovanie prvkov so zvolením orientácie. V prípade vertikálnej orientácie do stĺpca, v prípade horizontálnej do riadku. V tomto druhu kontajnera môžu byť prvky rozložené na základe priradenej váhy – atribút `weight`. Miesto pre jednotlivé prvky bude v prípade použitej váhy pridelené proporčne podľa nej.
- Tabuľkový kontajner – Prvky sú usporiadané do tabuľky pomocou riadkov a stĺpcov. V jednom riadku môže byť nula a viac buniek. Bunka môže obsahovať jeden objekt triedy `View`.

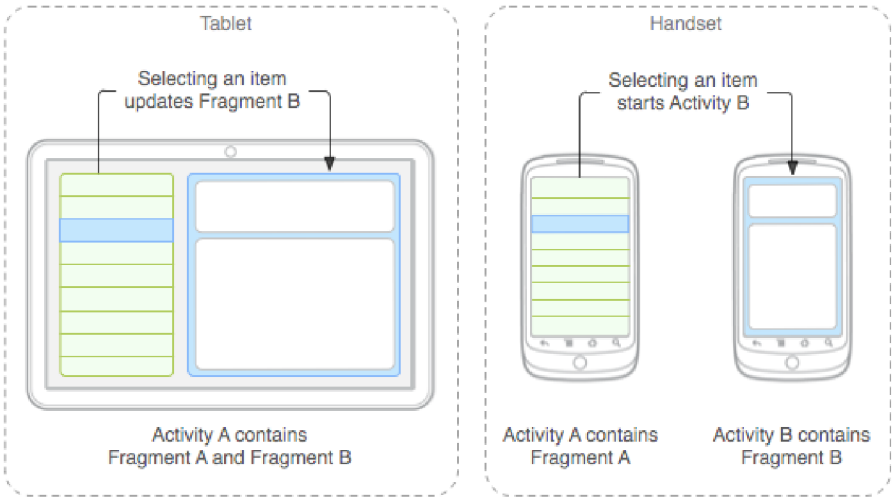
Fragmenty

V aplikačnom rozhraní pre Android verzie 11 bol pridaný prvok fragment (trieda `Fragment`). Pôvodne boli vytvorené pre tablety pre ľahšie vytvorenie rozličného rozloženia zobrazenia

[6] http://developer.android.com/guide/topics/ui/declaring-layout.html

toho istého obsahu na rôznych veľkostiach obrazoviek. Možnosti fragmentov je výhodné využiť aj pre mobilné telefóny. Fragment je viazaný na aktivitu ako medzivrstva medzi používateľským rozhraním a aktivitou.

Fragment môže obsahovať chovanie, používateľské rozhranie a vytvoriť znovupoužiteľné moduly. Musí byť vždy priradený k aktivite[7]. V aktivite môže byť viac fragmentov súčasne. Fragment môže byť znovu použitý vo viacerých aktivitách. Jeho životný cyklus je priamo ovplyvnený životným cyklom priradenej aktivity.



Obr. 3.1: Príklad použitia dvoch fragmentov v jednej aktivite pre sprostredkovanie rôzneho rozloženia podľa displeja. Na mobilnom telefóne je zobrazený iba jeden fragment, zatiaľ čo na tablete sú zobrazené obidva fragmenty.[7]

Fragment môže byť deklarovaný v layoute aktivity. V takom prípade je priradený k príslušnému `ViewGroup` v hierarchii rozloženia aktivity. Pri deklarácii do súboru layoutu sa použije XML element `fragment`. Pre vloženie fragmentu v kóde aktivity sa využíva trieda `FragmentTransaction`. Fragmenty majú používateľské rozhranie vo vlastnom layoute. Fragment je možné použiť aj na vykonávanie činnosti na pozadí bez vlastného používateľského rozhrania.

Prostriedky

Prostriedkami aplikácie sú nazývané statické súbory, ktoré obsahujú zdroje používané v aplikácii (rozloženie grafických prvkov, obrázky, reťazce, hudobné súbory, ponuky, farby, štýly, animácie, ...). Prostriedky sa vkladajú do priečinka `/res` umiestneného v koreňovom priečinku aplikácie[8]. Oddelením prostriedkov od zdrojového kódu, je možné meniť ich na jednom mieste bez zmien v kóde. Ďalšou výhodou je možnosť priradenia rôznych prostriedkov rôznym konfiguráciám zariadení (napríklad podľa rozlíšenia a orientácie displeja, jazykovej mutácie). To je možné, ak sa okrem východiskových prostriedkov vložia aj alternatívne. Východiskové sú použité v prípade, že neexistujú alternatívne, ktoré by zodpovedali použitému zariadeniu. Výber alternatívnych prostriedkov riešia kvalifikátory. Kvalifikátorom je krátky reťazec pridaný na koniec názvu priečinka (napr. v priečinku `/res/values-sk` budú

[7] http://developer.android.com/guide/components/fragments.html
[8] http://developer.android.com/guide/topics/resources/overview.html

hodnoty použité v systéme v slovenskom jazyku). Kvalifikátor popisuje vlastnosti zariadenia určeného pre použitie prostriedkov v danom priečinku. Android poskytuje sadu rôznych kvalifikátorov pre.

Možnosti ukladania

Android poskytuje rôzne možnosti pre ukladanie trvalých dát aplikácií. Dôležitými aspektmi pre výber spôsobu ukladania dát je množstvo priestoru, ktoré budú potrebovať a prístup k nim[9].

- Zdieľané nastavenia (Shared Preferences) – Pre ukladanie dát (nie iba nastavení) v pároch kľúč – hodnota. Hodnota musí byť primitívneho dátového typu. Trieda `SharedPreferences` poskytuje rozhranie pre správu takto uložených dát.
- Vnútorné úložisko – Pre ukladanie súkromných dát v pamäti zariadenia. K takto uloženým dátam vo východiskovom nastavení iné aplikácie nemajú prístup. Dáta sú odstránené po odinštalovaní aplikácie.
- Vonkajšie úložisko – Verejných dáta je možné uložiť v zdieľanej vonkajšej pamäti (napríklad v SD karte).
- Databáza typu SQLite – Pre uloženie štruktúrovaných dát v databázy. K databázy nemajú prístup iné aplikácie.
- Sieťový server – Pre uloženie dát je možné použiť vlastný sieťový server.

Používanie databázy

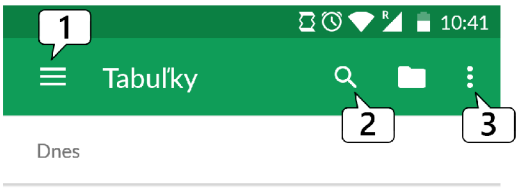
Databáza typu SQLite je plne podporovaná systémom Android. Databáza je prístupná zadaním názvu iba triedam aplikácie. Názov databázy musí byť unikátny v rámci aplikácie. Databázu typu SQlite je možné vytvoriť novou triedou odvedenou od triedy `SQLiteOpenHelper`, ktorá bude mať preťaženú metódu `onCreate`. Táto metóda može príkazmi SQLite vytvoriť požadované tabuľky.

Prístup k databáze sprostredkuje jej inštancia, ktorá ju reprezentuje – objekt triedy `SQLiteDatabase`. Táto trieda má metódy pre vykonanie SQLite operácií. Pre získanie inštancie databáze pre čítanie slúži metóda `getReadableDatabase`, pre získanie inštancie databáze s možnosťou do nej zapisovať sa používa metóda `getWritableDatabase`.

ActionBar

Prvok `ActionBar` je grafický prvok – lišta v hornej časti obrazovky[10]. Umožňuje zobrazenie názvu a ikony aplikácie. Využíva sa k nápovede používateľovej aktuálnej pozície v aplikácii a zobrazenia ponuky možností. Použitím prvku `ActionBar` je vytvorené podobné rozhranie ako v iných aplikáciách priblížením sa ku zvyčajnému rozhraniu aplikácí na systéme Android [1]. Trieda `ActionBar` je v systéme Android dostupná až od verzii 3.0. Pre umožnenie používania triedy `ActionBar` v starších verziách je dostupné využitím knižnice Support Library pre systémy od verzie 7 – Android 2.1. Ide o knižnicu `appcompat`. Musí byť pre zabezpečenie kompatibility vložená do projektu aplikácie. Používaním knižnice sa rozhranie triedy `ActionBar` mierne odlišuje.

[9] http://developer.android.com/guide/topics/data/data-storage.html
[10] http://developer.android.com/guide/topics/ui/actionbar.html



Obr. 3.2: Zobrazenie prvku `ActionBar` (obrázok z aplikácie Google Sheets). 1 - ikona vysúvacieho menu, 2 - položky pre akcie, 3 - položka pre zobrazenie ďalších akcií.

Vysúvacie menu

Vysúvacie menu je panel zobrazujúci časť používateľského rozhrania (typicky navigáciu alebo možnosti), ktorý je väčšin času schovaný za ľavou/pravou hranou displeja. Vysunutie menu je realizované ťahom prsta od ľavej/pravej hrany displeja smerom do stredu alebo po stlačení tlačidla (ikony) v prvku `ActionBar`[11]. Knižnica Support Library umožňuje toto menu používať v systéme od verzie 4. Aplikačné rozhranie poskytuje prístup k menu prostredníctvom triedy `DrawerLayout`. Pri použití tohto menu je v súbore s rozvrhnutím používateľského rozhrania deklarovaný ako koreňový element trieda `DrawerLayout`, v ktorej je umiestnený skupina prvkov pre zobrazenie obsahu aplikácie počas skrytého menu a skupina prvkov zobrazujúci obsah menu.

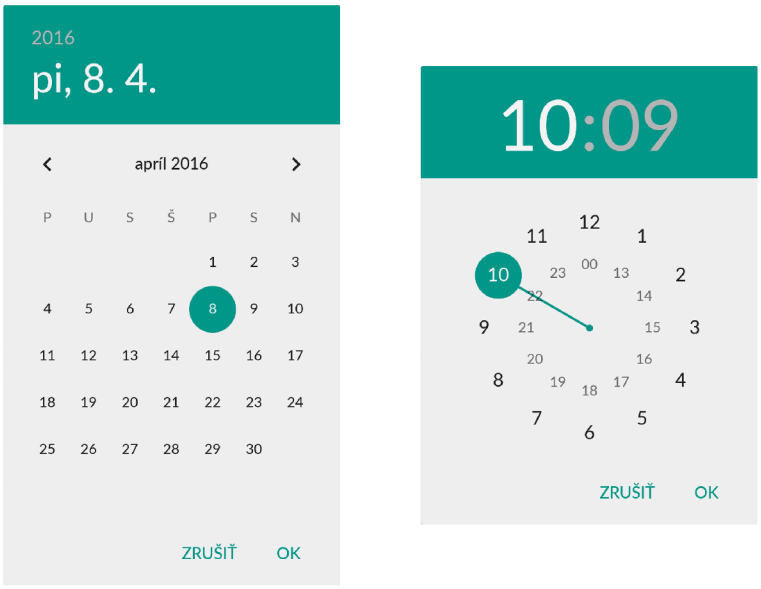
Výber dátumu a času

Výber dátumu a času môže byť riešený prostredníctvom tried aplikačného rozhrania `TimePickerDialog` a `DatePickerDialog`. Ide o dialógy, ktoré umožňujú používateľovi vložiť hodnoty pre dátum a čas[12]. Triedy zaisťujú aj kontrolu validity zadaného vstupu. Dialógy je možné prednastaviť konštruktorom. Metódami `updateDate`, `updateTime` je možné nastaviť dátum a čas. Rozhraním `OnDateSetListener` je možné registrovať načúvací objekt, ktorého metóda `onDateSet` umužňuje reagovať na zadanie hodnôt používateľom. V dokumentácii je doporučené využiť pre sprostredkovanie dialógov triedu `DialogFragment`. Trieda poskytuje rozhranie pre zobrazenie a správu dialógu.

3.3.3 Podpora verzií platformy Android

Rozvíjaním platformy Android dochádza k deleniu na verzie. Rozdiely vo verziách prinášajú komplikácie. V novej verzii môže byť nové aplikačné rozhranie, ktoré v predošlých verziách nie je. Podporovať čo možno najviac aktívnych zariadení je kvôli dostupnosti veľmi žiadané. Sada knižníc Android Support Library bola vytvorená pre umožnenie využitia nového rozhrania a zároveň zachovanie spätnej kompatibility so staršími verziami platformy. Je vhodné určiť, ktoré verzie platformy majú byť aplikáciou podporované ešte pred samotnou implementáciou a následne tomu implementáciu prispôsobiť.

[11] http://developer.android.com/design/patterns/navigation-drawer.html
[12] http://developer.android.com/guide/topics/ui/controls/pickers.html



Obr. 3.3: Zobrazenie dialógov pre výber dátumu a času.

Kapitola 4

Technológie pre vytvorenie servera

Pre vývoj serveru na daný účel je kľúčové správne vybrať a použiť najvhodnejšie z dostupných technológií. V tejto kapitole sú popísané najdôležitejšie technológie.

4.1 Node.js

Node.js je open-source multiplatformná platforma pre vývoj serverových webových aplikácií [14]. Vytvoril ho Ryan Dahl spolu s ďalšími vývojármi firmy Joyent v roku 2009. Aplikácie sú písané v jazyku JavaScript. Jedná sa o architektúru riadenú udalosťami. Poskytuje API pre neblokujúcu prácu so vstupom a výstupom. Pre vykonávanie kódu používa V8 JavaScript engine používaný v prehliadači Google Chrome[15].

Pre Node.js je dostupné obrovské množstvo balíkov, ktoré značne uľahčujú a zrýchľujú vývoj aplikácií. Pre prácu s balíkmi sa používa správca balíkov npm.

4.2 CoffeeScript

CoffeeScript bol vytvorený pre zjednodušenie kódu jazyka JavaScrip. CoffeeScript sa kompiluje do jazyka JavaScript a počas behu už k prekladu nedochádza.

4.3 Knižnice pre komunikáciu so serverom

V tejto časti sú popísané knižnice, ktoré okrem iného umožňujú komunikáciu so serverom prostredníctvom HTTP metód a WebSocket protokolu. Stručne je predstavený aj protokol WebSocket.

4.3.1 Zappajs

Je to modul, ktorý poskytuje minimalistické rozhranie v jazyku CoffeeScript pre používanie modulov Express, Socket.io a iných [10].

4.3.2 Express

Express je minimalistický flexibilný framework pre vytváranie Node.js webových aplikácií [16]. Umožňuje nastavenie rôznych modulov ako midlvér. Medzi významné a často používané moduly patria moduly body-parser a cookie-parser. Body-parser zabezpečuje spracovanie

dát z požiadavku formátu JSON, RAW, text, URL-enkódovacom formulári. Cookie-parser spracúva hlavičky cookies a ukladá ich ako páry kľúč-hodnota kde kľúčom je názov cookie. Express vykonáva smerovanie podľa definovaných akcií vybraných podľa HTTP metódy (napr. GET, POST) a URL. Pre express je dostupných viac ako štrnásť šablónovacích systémov (template engines). Navyše je možné dynamicky vykreslovať HTML stránky pomocou predávania argumentov šablónam.

4.3.3 WebSocket

Protokol WebSocket poskytuje full-duplex komunikačný kanál cez jediné TCP spojenie. Je navrhnutý pre implementáciu vo webových prehliadačoch a webových serveroch, ale môže byť použitý akoukoľvek klientskou alebo serverovou aplikáciou. Handshake protokolu je HTTP serverami interpretovaný ako Upgrade požiadavok. WebSocket musí podporovať serverová aj klientská aplikácia. V súčasnosti je podporovaný najpoužívanejšími webovými prehliadačmi – Google Chrome, Internet Explorer, Mozzila Firefox, Safari, Opera.

4.3.4 Socket.io

Socket.io je knižnica pre komunikáciu webových aplikácií v reálnom čase napísaná v jazyku JavaScript [9]. Umožňuje obojsmernú komunikáciu medzi klientami a serverom. Podľa toho je aj knižnica rozdelená na dve časti – klientská časť pre beh vo webovom prehliadači, serverová časť pre beh na serveri node.js. V súčasnosti sú už dostupné klientské časti knižnice Socket.io pre Android, iOS a dostupná je aj verzia v C++. Socket.io má architektúru riadenú udalosťami. Socket.io pre komunikáciu primárne používa protokol WebSocket. Pokiaľ by nebolo možné použitie protokolu WebSocket, použije sa polling. Toto je vyriešené interne a knižnica sa používa jednotným rozhraním. Jednou z mnohých výhod knižnice Socket.io je podpora horizontálneho škálovania (viac inštancií servera).

4.4 Databázy

Pre perzistentné ukladanie dát používateľov bude potrebné použiť databázu. Databáza bude použitá aj pri udržovaní dát aktuálnych pripojení (sessions). V tejto časti sú popísané databázy, ktoré je možné na daný účel použiť.

4.4.1 NoSQL

NoSQL sú vhodné pre ukladanie štrukturovaných dát, ktoré sú modelované inak ako tabuľkovými reláciami (používaných v SQL) [8]. Medzi hlavné výhody NoSQL databáz patrí jednoduchosť návrhu a horizontálna škálovateľnosť (distribuovanie záťaže na viac serverov).

4.4.2 MongoDB

MongoDB je NoSQL databáza. MongoDB je open-source multiplatformná dokumentovo orientovaná databáza [13]. Pre ukladanie dát používa formát BSON – formát podobný JSON. Používa dynamické schémy – je možné vytvárať kolekcie bez definovania štruktúry (názvov kľúčov a typov hodnôt položiek). Je povolené meniť štruktúru dokumentov pridávaním alebo odstraňovaním položiek. Všetky dokumenty v jednej kolekcii nemusia byť identické.

4.4.3 Redis

Výhodou NoSQL databáze Redis je veľká výkonnosť vďaka práci s dátami v pamäti na rozdiel od databáz, ktoré zapisujú každú zmenu na disk [11].

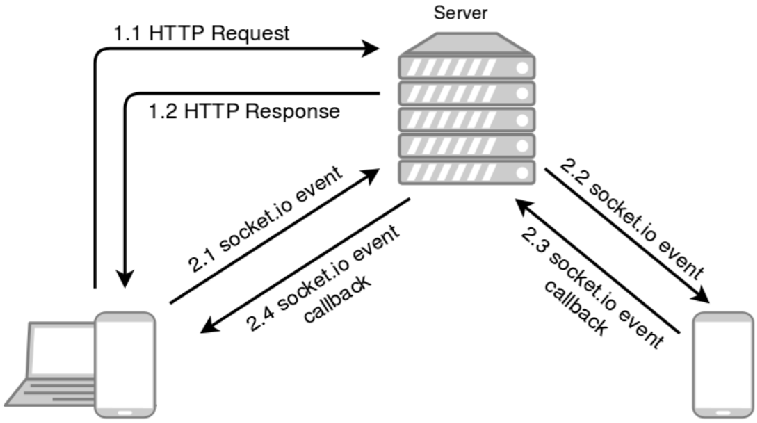
Kapitola 5

Návrh systému

V tejto kapitole je popísaný návrh architektúry systému a návrhy používateľských rozhraní aplikácií.

5.1 Návrh architektúry systému

Archytektúra systému musí byť navrhnutá tak, aby poskytovala zabezpečenú komunikáciu medzi aplikáciami a serverom. V tejto kapitole je popis návrhu.

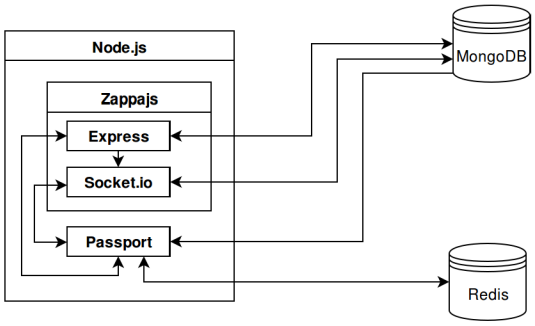


Obr. 5.1: Na obrázku je ukážka komunikácie aplikácie so serverom prostredníctvom HTTP a ukážka komunikácie medzi aplikáciami cez Socket.io.

Pre prepojenie aplikácie som zvolil model klient-server. Pre implementáciu servera som sa rozhodol použiť platformu Node.js spolu s programovacím jazykom CoffeeScript.

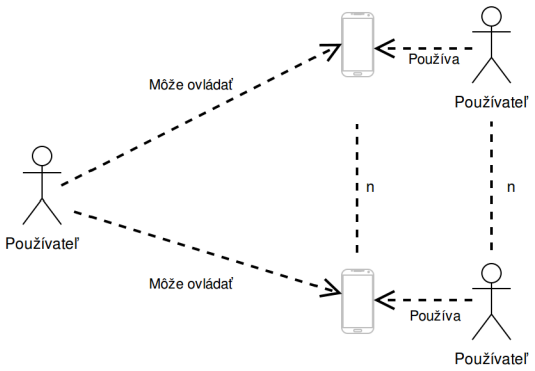
Pre vytvorenie serverovej aplikácie som použil Zappajs. Pre prácu so sessions sa používa open-source NoSQL databáza Redis. Aplikácie budú komunikovať medzi sebou a so serverom prostredníctvom knižníc Socket.io. Webová aplikácia využije klientskú časť knižnice naprogramovanú v Javascripte a Android aplikácie využijú verziu naprogramovanú v Jave[1]. Aplikácie okrem toho používajú aj HTTPS požiadavky pre komunikáciu so serverom. Pomocou nich prebieha autentizácia. Využíva sa k tomu knižnica passport. Skontroluje prihlasovacie údaje s uloženými. Po úspešnom prihlásení sa používateľovi v HTTP odpovedi vráti aj identifikátor session. Ten nastaví pre posielanie správ cez Socket.IO knižnicu. Daný

[1] https://github.com/socketio/socket.io-client-java



Obr. 5.2: Na obrázku sú zobrazené prepojenia technológií použitých pre serverovú aplikáciu a použitých databáz.

identifikátor sa overuje na strane serveru, aby bola komunikácia zabezpečená. Pre perzistentné ukladanie dát je použitá NoSQL databáza MongoDB. V nej budú uložené dáta používateľov ako e-mailová adresa, heslo, nastavenia zariadenia a väzby medzi používateľmi.



Obr. 5.3: Obrázok znázorňuje možnosť vzdialeného ovládania viacerých mobilných zariadení. Prihlásený je na vzdialenom zariadení práve jeden používateľ.

Väzby budú určovať, ktorý používateľ môže vzdialene ovládať koho mobilné zariadenie. Prístup k zariadeniu zabezpečuje nainštalovaná aplikácia. K aplikácii je možné pripojiť sa cez Android alebo webovú aplikáciu. Na oboch stranách spojenia bude musieť byť pre prepojenie prihlásený používateľ a bude musieť mať povolenie k ovládaniu.

Aplikácia na vzdialenom zariadení bude spracovávať prichádzajúce správy, vyberať a vykonávať podľa nich príslušné akcie. Pre implementáciu vzdialenej plochy sa budú v prípade, že je funcia aktívna, na zariadení vytvárať snímky obrazovky po každej akcii meniacej obsah snímky, ktoré sa budú odosielať pre zobrazenie obsahu. Druhá strana kliknutím (stlačením) na zobrazenú snímku vypočíta polohu kliknutia relatívne k snímke a odošle ju vzdialenému zariadeniu, ktoré polohu prepočíta pre dané rozlíšenie a kliknutie vykoná.

V službe sú dva typy používateľov – junior a senior. Senior je používateľ, ktorý potrebuje zjednodušiť rozhranie systému a prípadne aj pomoc s nastavovaním a ovládaním zariadenia. Junior je používateľ, ktorý môže byť administrátorom seniorovho zariadenia alebo môže byť prepojený so seniorom kvôli komunikácii.

Sú dostupné tri klientské aplikácie. Junior má dostupnú webovú aplikáciu a Android aplikáciu, zatiaľ čo senior iba Android aplikáciu. Pre juniora je výhodné poskytnúť webovú aplikáciu, keďže vzdialenú asistenciu pri ovládaní seniorovho zariadenia a komunikáciu so

seniorom by mal mať dostupnú aj pokiaľ nevlastní Android zariadenie. Naopak u seniora sa kvôli predpokladaným zručnostiam webová aplikácia nevytvorila. Webová aplikácia má výrazne obmedzené možnosti pre vytvorenie zjednodušujúcej medzivrstvy medzi hostiteľským systémom a používateľom. V prípade, že by používateľ zatvoril webový prehliadač, nebolo by možné spojiť sa s jeho zariadením a vzdialene ho ovládať.

V prípade Android aplikácie bolo možné vytvoriť jednu aplikáciu s rôznym obsahom podľa typu používateľa. K vytvoreniu dvoch separátnych aplikácií bolo niekoľko. Jedným faktorom bolo zmenšenie veľkosti aplikácie tým, že aplikácia neobsahuje časti, ktoré daný typ používateľa nepotrebuje. Dôležitejším dôvodom bol rozdiel vo funkčnosti. Seniorova aplikácia musí bežať nonstop na popredí a nemôže byť omylom vypnutá alebo minimalizovaná. Pre tieto účely musia byť vykonané úpravy a je potrebné mať príslušné povolenia, ktoré v prípade juniora sú zbytočné. Pri aplikácii seniora sa predpokladá používanie zariadenia s pripojeným adaptérom. Aplikácia sa snaží udržať sieťové pripojenie neustále spojené. To ma za následok väčšiu spotrebu elektrickej energie, keďže sa stále využívajú zdroje zariadenia (procesor, sieťové prvky, ...). Pri juniorovej aplikácii naopak nemôže dôjsť k nadmernému vyťažovaniu batérie.

5.2 Návrh aplikácie pre juniora

Hlavným účelom aplikácie je umožnenie poskytovania vzdialenej podpory. Ďalším dôležitým prvkom sú funkcie pre komunikáciu s ostatnými používateľmi (hovor, videohovor, textové správy) a zdieľanie obrázkov, fotografií. Aplikácia preto poskytuje na toto zamerané používateľské rozhranie.

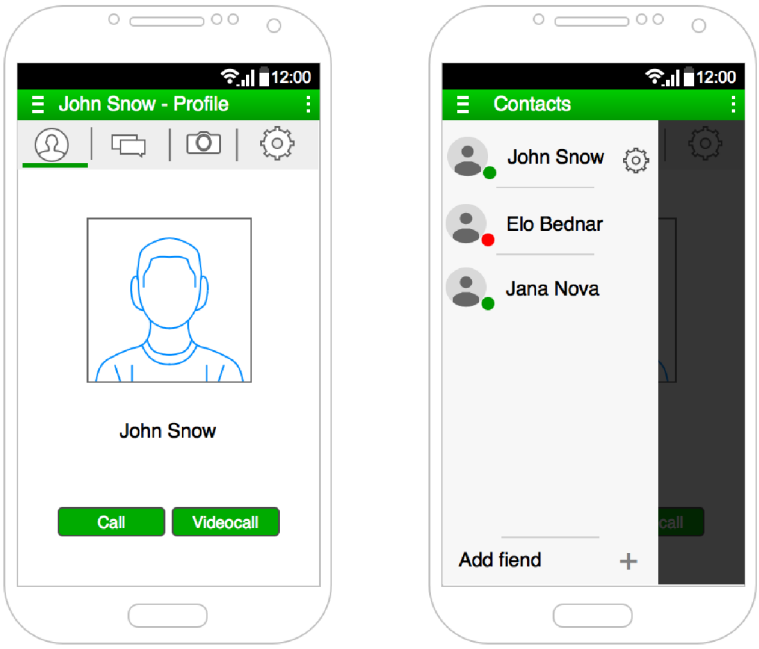
Používateľské rozhranie je rozdelené na dva logické celky. Prvým celkom je zoznam kontaktov používateľa. Používateľ musí vidieť svoje kontakty a musí byť jasne označená dostupnosť kontaktu (online/offline). Kontakty, ktorým môže používateľ poskytovať vzdialenú podporu musia byť jasne odlíšené. Podľa typu zvoleného kontaktu totiž bude zobrazené iné rozhranie. V druhom logickom celku sú informácie o vybranom používateľovi a komunikačné rozhranie. Pre komunikáciu sú dostupné textové správy, hovor, videohovor a je možné odosielať a prijímať obrázky. V prípade kontaktu, ktorý môže používateľ vzdialene ovládať je dostupné príslušné rozhranie.

Kontakty budú zobrazené vo vysúvacom menu. Vďaka tomu bude mať používateľ kontakty kedykoľvek dostupné a navyše budú vizuálne oddelené od vybraného kontaktu. Prostredníctvom hornej lišty bude používateľ vidieť, aký kontakt je aktuálne vybraný a v akej časti aplikácie sa nachádza. Tlačidlo vľavo indikuje vysúvacie menu a pomocou tlačidla vpravo sa zobrazí menu používateľa.

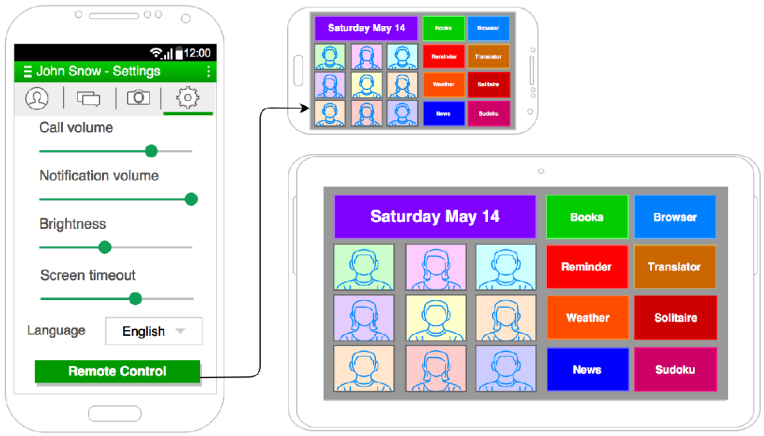
Pre vybraný kontakt bude zobrazený príslušný obsah rozriedený do troch kategórií, prípadne štyroch, ak sa jedná o kontakt, ktorý môže používateľ vzdialene spravovať. Kategórie budú zobrazené v kartách. Pre navigáciu medzi kartami a zvýraznenie aktuálne vybranej kategórie bude slúžiť lišta s ikonami znázorňujúce kategórie.

V kategórii profil bude zobrazená profilová fotografia kontaktu a možnosti pre spustenie hovoru, videohovoru. Ďalšími kategóriami sú správy a obrázky, v ktorých ako názov napovedá pôjde o odosielanie a prijímanie obrázkov, správ a zobrazenie predošlej komunikácie. Posledná kategória poskytuje rozhranie pre vzdialenú podporu.

Kategória so vzdialeným nastavením umožňuje prostredníctvom posuvníkov a rozbaľovacej ponuky zobraziť aktuálne nastavené hodnoty preferencií a prípadne ich zmeniť. Používateľ má prístupné tlačidlo pre vstup do vzdialeného ovládania. Po stlačení tohto tlačidla sa v prípade, že je kontakt pripojený, zobrazí obsah vykreslený na zariadení kontaktu.



Obr. 5.4: Na obrázkoch je zobrazené používateľské rozhranie aplikácie pre juniora. Vľavo je zobrazená karta profilu kontaktu s tlačidlami pre uskutočnenie hovoru. V hornej lište je popis práve zobrazenej časti aplikácie. Pod lištou je zobrazený výber karty s ikonami značiacimi karty. Na obrázku vpravo je zobrazené vysunuté menu s kontaktami používateľa.

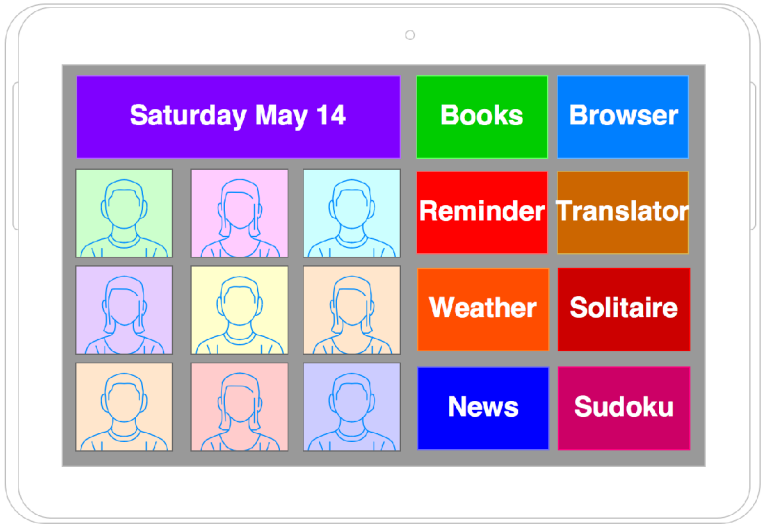


Obr. 5.5: Na obrázku je zobrazené rozhranie pre vzdialenú zmenu nastavenia a tlačidlo pre vstup do vzdialeného ovládania. Vpravo je zobrazené vzdialené ovládanie pomocou vzdialenej plochy – na mobilnom telefóne sa zobrazí obsah obrazovky vzdialeného zariadenia.

Pôvodný návrh používateľského rozhrania sa s postupom práce na aplikácii menil. Hlavnými faktormi pre zmeny v návrhu bolo zistenie nových detailov, súvislostí a po zverejnení aplikácie aj spätná väzba používateľov. Jedným z príkladov je nová kategória so záznamami udalostí na zariadení, ktorá vznikla požiadavkami používateľov pre poskytnutie prehľadu o stave zariadenia a vykonaných akciách (ako napríklad kedy sa zariadenie odpojilo/pripojilo).

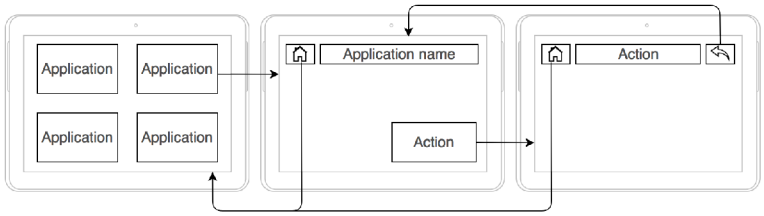
5.3 Návrh aplikácie pre seniora

Kľúčové v aplikácii určenej pre seniora je zameranie na čo najjednoduchšie používateľské rozhranie. Predpokladá sa, že aplikáciu bude ovládať človek, pre ktorého je ovládanie mobilných zariadení príliš zložité. Môže byť spôsobené málo skúsenosťami s modernými zariadeniami, ale aj zdravotným obmedzením, či inými dôvodmi. Kvôli zväčšenému zobrazeniu obsahu je aplikácia určená pre tablety.



Obr. 5.6: Na obrázku vpravo je zobrazené zjednodušené používateľské rozhranie aplikácie pre seniora. Dlaždice a písmo sú zväčšené pre ľahšie používanie seniorom.

V návrhu je jednoduchosť docielená veľkými ovládacími prvkami vo forme dlaždíc. Kvôli možnému zhoršenému zraku je zvolená väčšia veľkosť textu a prvky sú odlíšené dostatočným kontrastom farieb. Navigácia v aplikácii je taktiež prispôsobená pre ľahké ovládanie. Používateľ sa z úvodnej ponuky môže dostať maximálne do druhej úrovne zanorenia. Pokiaľ sa nachádza mimo úvodnej ponuky vždy má dostupnú hornú lištu s tlačidlom pre návrat na úvodnú ponuku a popis kde sa v aplikácii nachádza. V prípade, že je v druhej úrovne zanorenia, je navyše zobrazené tlačidlo pre návrat o úroveň vyššie (o krok späť).



Obr. 5.7: Na obrázku je zobrazená zjednodušená navigácia v aplikácii pre seniora. Používateľ sa môže dostať maximálne do druhej úrovne zanorenia. Pokiaľ používateľ nie je na úvodnej obrazovke, je zobrazená horná lišta, ktorá vždy umožňuje návrat na úvodnú obrazovku alebo späť.

Kapitola 6

Implementácia aplikácií

Táto kapitola sa zaoberá implementáciou aplikácií. U každej z aplikácií je uvedené z akých častí sa skladá, aké komponenty boli využité, ako medzi aplikáciami prebieha komunikácia.

6.1 Implementácia aplikácie pre juniora

Hlavná časť aplikácie je v aktivite `MainActivity`. Tá je spúšťaná pri štarte aplikácie. Trieda `MainActivity` je vytvorená rozšírením triedy `AppCompatActivity` z knižníc Support Library pre spätnú kompatibilitu, ktorá umožňuje použitie prvkov pridaných v novších verziách Androidu na zariadeniach so staršou verziou. Vďaka tomu bolo možné použiť pre implementáciu hornej lišty triedu `ActionBar`.

Pre zobrazenie kategórií je použitá trieda `ViewPager`. Inštancii tejto triedy je priradený adaptér obsahujúci fragmenty, ktoré zapuzdrujú jednotlivé časti aplikácie. Adaptér bol vytvorený rozšírením triedy `FragmentStatePagerAdapter`, ktorá narozdiel od `FragmentPagerAdapter` povoľuje zničenie fragmentu a uloženie iba jeho stavu v prípade, že fragment nie je viditeľný. Vďaka tomu `ViewPager` nemusí držať obsah všetkých fragmentov v pamäti. V tomto prípade je to výhodné vzhľadom k tomu, že jedným z fragmentov je `GalleryFragment`, ktorý zobrazuje obrázky používateľov. Bez tejto alebo inej formy optimalizácie by sa pri používaní aplikácie na zariadeniach s menšou operačnou pamäťou (RAM) mohol objaviť problém s nedostatkom pamäte – `OutOfMemoryError`.

Kontakty sú zobrazené vo vysúvacom menu (Navigation Drawer), ktoré je implementované triedou `DrawerLayout`. Jej použitie je deklarované ako koreňový element v XML súbore `acitivity_main.xml` definujúcim rozloženie grafických prvkov. `MainActivity` tento súbor použije pre vykreslenie rozhrania. V `MainActivity` je kód, ktorý nastavuje chovanie `DrawerLayout`. O zobrazenie kontaktov sa stará adaptér implementovaný triedou `ContactsCursorAdapter` odvedenou od triedy `CursorAdapter`. Adaptéru sa nastaví databázový kurzor ukazujúci na dáta kontaktov.

Rozhranie pre vzdialenú správu zariadenia je vo fragmente `TabletAllSettingsFragment`. V tomto fragmente je zoznam rôznych nastavení. Zoznam je zobrazený pomocou triedy `SettingsAdapter`, ktorá je odvodená od triedy `BaseAdapter`. Ide o adaptér, ktorý zo zdrojov načíta pole textov – názvov možností v nastaveniach. Takto je umožnené aby boli texty lokalizované do príslušného jazyka. Po zvolení niektorej položky zo zoznamu sa zobrazí odpovedajúci fragment.

Vzdialené nastavenie je vo fragmente `TabletSettingsFragment`. Pre zvolenie hodnôt z určitého rozsahu je použitý posuvník, ktorý je implementovaný pomocou triedy `SeekBar`.

Na inštanciu triedy `SeekBar` je zaregistrovaný načúvací objekt `OnSeekBarChangeListener`, ktorého metóda `onProgressChanged` je zavolaná v prípade, že došlo k zmene hodnoty. V tejto metóde sa prostredníctvom inštancie `LocalBroadcastManager` odosiela správa službe `SocketConnectionService`, ktorá odosiela cez socket správu so zmenou hodnoty na server. Server zabezpečuje delegovanie zariadeniu. Podobne to funguje aj pri zmene jazyka. Voľba jazykov je zobrazená pomocou vysúvacieho menu implementovaného triedou `Spinner`. Inštancii triedy `Spinner` je nastavený načúvací objekt `onItemSelectedListener` s metódou `onItemSelected` volanou po vybraní položky. Objektu `Spinner` je nastavený adaptér implementovaný triedou `ArrayAdapter`, ktorý zobrazuje textové položky predané hodnotou v zozname objektov typu `String`.

Vo fragmente `RemoteViewFragment` je implementované rozhranie pre vzdialené ovládanie zariadenia pomocou vzdialenej plochy. Zabezpečuje dve hlavné úlohy – zobrazenie plochy vzdialeného zariadenia a odoslanie príkazov (kliknutí). Fragment zobrazí na celú obrazovku prvok `ImageView`, do ktorého sa pri vzdialenom ovládaní načítavajú obrázky. Ide o snímky obrazovky vzdialeného zariadenia, ktoré prídu zo zariadenia cez server socket spojením a po spracovaní službou `SocketConnectionService` sú cez `LocalBroadcastManager` poslané do fragmentu.

Fragment má pre prijímanie fotiek zaregistrovaný prijímač `BroadcastReceiver`. Keďže Socket.io verzie 0.9.x nepodporuje odosielanie binárnych dát, obrázok príde v správe formátu JSON zakódovaný kódovaním BASE64. Po príchode do fragmentu sa obrázok dekóduje a zobrazí v prvku `ImageView`. Pre dekódovanie je použitá trieda `Base64`.

Objekt `ImageView` má zaregistrovaný načúvací objekt `onTouchListener`, ktorého metóda `onTouch` je zavolaná po stlačení `ImageView`. V metóde je získaná poloha dotyku (daná súradnicami v dvojrozmernej karteziánskej sústave súradníc) v rámci obrázka. Súradnice sú prevedené na pomer k stranám, vzhľadom k tomu, že zariadenia môžu mať rozdielne rozlíšenia displejov a poloha dotyku by na inom rozlíšení odkazovala na inú časť zobrazenej plochy. Takto upravené súradnice sú následne odoslané cez službu `SocketConnectionService`, ktorá ich prepošle na ovládané zariadenie.

6.2 Implementácia aplikácie pre seniora

Aktivitou spúšťanou pri štarte aplikácie je `MainActivity`, ktorá spúšťa ďalšie potrebné komponenty. Táto aktivita zobrazí spodnú lištu s možnosťou nápovede a ukončenia aplikácie. Okrem toho sú v nej registrované prijímače `BroadcastReceiver`, pomocou ktorých aktivita prijíma prichádzajúce akcie zo služby `SocketConnectionService`. Služba spracúva správy chodiace cez socket spojenie a cez inštanciu `LocalBroadcastManager` odosiela príslušné akcie aktivite a fragmentom pre vykonanie potrebných operácií (často súvisiacich s aktualizovaním grafického používateľského rozhrania).

`MainActivity` zobrazuje oznámenia o prichádzajúcich správach, obrázkoch, žiadostiach o priateľstvo. Väčšina udalostí je vykreslená univerzálnym dialógom implementovaným triedou `NewEventsDialog`. Inštancii dialógu sa pridá dané oznámenie. Týmto spôsobom je možné agregovať viaceré oznámenia do jedného dialógu namiesto vykreslenia viacerých dialógov, čo je pre používateľa prehľadnejšie. V prípade, že je otvorený detail kontaktu, nezobrazí sa dialóg, ale aktualizuje sa zobrazenie (buď zmenou počtu nevidených správ, obrázkov alebo vykreslením správy, obrázku v zobrazenom zozname).

Výnimkou z udalostí, ktorá sa zobrazuje iným dialógom je dialóg pre prichádzajúci hovor. Je tomu hlavne z dôvodu, že sa jedná o dialóg s vyššou prioritou, nakoľko je nutné v reálnom čase vyžiadať od používateľa práve jednu akciu (prijatie/odmietnutie hovoru). Tento

dialóg je implementovaný pomocou triedy `IncomingCallDialog`, ktorá sa stará o zobrazenie a odoslanie používateľovho výberu. Okrem toho je potrebné zabezpečiť, aby zariadenie použilo aj zvukovú notifikáciu. To je implementované v triede `Ringer`. Okrem vyzváňania trieda rieši prebudenie obrazovky pri začatí vyzváňania. Je to dosiahnuté inštanciou triedy `PowerManager`, pomocou ktorej sa vytvorí objekt triedy `WakeLock`. Tento objekt je vytvorený s parametrami, ktoré určujú aké časti zariadenia nemôžu prejsť do spánku. V tomto prípade je použitý parameter pre prebudenie procesora aj displeja. Keďže tento zámok spánku má výrazný dopad na zvýšení spotreby zariadenia, zámok je uvoľnený hneď po skončení vyzváňania.

Pre vykreslenie hlavnej časti aplikácie je použitý fragment `HomeScreenFragment`. Pre rozloženie grafických prvkov je použitá trieda `CustomGridLayout`, ktorá je odvedená od triedy `GridLayout`. `CustomGridLayout` navyše poskytuje metódy pre vytvorenie dlaždicového vzhľadu ako výpočet výšky prvkov. Pre prekreslenie dlaždíc v prípade zmeny veľkosti objektu `CustomGridLayout` je použité preťaženie metódy `onSizeChanged`, v ktorej sa volá metóda fragmentu `HomeScreenFragment` `redrawTiles`. Dlaždica je realizovaná triedou `GridTile`, ktorá je rozšírením triedy `RelativeLayout`. Nastavuje veľkosť, odsadenie dlaždice a jej umiestnenie v mriežke `CustomGridLayout`. Od triedy dedia triedy `ButtonTile`, `TimeTile`, `ContactsGridTile`, `BatteryTile`, ktoré špecifikujú detaily vzhľadu určitého typu dlaždice.

Po príchode socket udalosti pre zmenu nastavení, služba `SocketConnectionService` vytvorí inštanciu triedy `SettingsManager`, ktorá spravuje nastavenia. Trieda obsahuje metódy pre zmenu konkrétnych nastavení, ktorým sa parametrom predáva nová hodnota. Pri zmene nastavení sa nové hodnoty ukladajú perzistentne do zariadenia pomocou rozhrania `SharedPreferences`. Po spustení aplikácie (aj v prípade reštartu zariadenia) sa načítajú predvolené hodnoty nastavení. Pre možnosť zmeniť aplikáciou nastavenia zariadenia musí používateľ súhlasiť s povolením (`android.permission.WRITE_SETTINGS`).

Zmena hlasitosti je možná prostredníctvom služby `AudioManager`. Metódou jej inštancie `setStreamVolume` sa predaním hodnoty a typu hlasitosti nastaví nová hodnota. Na Androide sa rozlišuje hlasitosť alarmov, hudby, notifikácií, zvonenia, systému, hovoru.

Editovanie nastavenia jasu obrazovky je realizovaná cez rozhranie `WindowManager`, konkrétne triedou `LayoutParams`. Inštanciu `LayoutParams` je možné získať cez atribúty okna aplikácie k čomu slúžia metódy `getWindow` a `getAttributes`. Parametru inštancie `screen Brightness` je priradená požadovaná hodnota a takto upravené atribúty je potrebné oknu nastaviť metódou `setAttributes`.

Zvolenie hodnoty pre interval vypnutia obrazovky prebieha pomocou triedy `System`, v ktorej sú uložené rôzne preferencie. Preferencie sú uložené v tabuľke vo forme dvojíc kľúč/hodnota. Nová hodnota je kľúču priradená pomocou triednej metódy `putInt`, ktorej sa okrem kľúču, hodnoty predáva argumentom aj inštancia triedy `ContentResolver`. Objekt `ContentResolver` umožňuje prístup k nastaveniam.

Použitie iného jazyka ako je nastavený v systéme je dostupné upravením konfugurácie. Najskôr sa získa aktuálna kofigurácia v podobe objektu triedy `Configuration` zavolaním metódy `getConfiguration` na objekt triedy `Resources`, ktorý reprezentuje zdroje aplikácie a je získaný metódou `getResources` z kontexktu aktivity. Parametru locale objektu `Configuration` je priradená nová hodnota lokalizácie. Hodnota je typu `Locale`. Nový objekt `Locale` reprezentujúci lokalizáciu je vytvorený konštruktorom, ktorému je predaný hodnotou reťazec s kódom jazyka. Kód pozostáva z dvoch písmen podľa definície `ISO 639-1` (napríklad `ën¨` pre angličtinu). Po zmene jazyka je potrebné vynútiť prekreslenie `View` objektov aby sa zobrazili znovu s inými textami. Je to vykonané zneplatnením koreňového

`View` objektu metódou `invalidate`. Objekty `View` sú po volaní tejto metódy prekreslené až potom ako nie sú viditeľné.

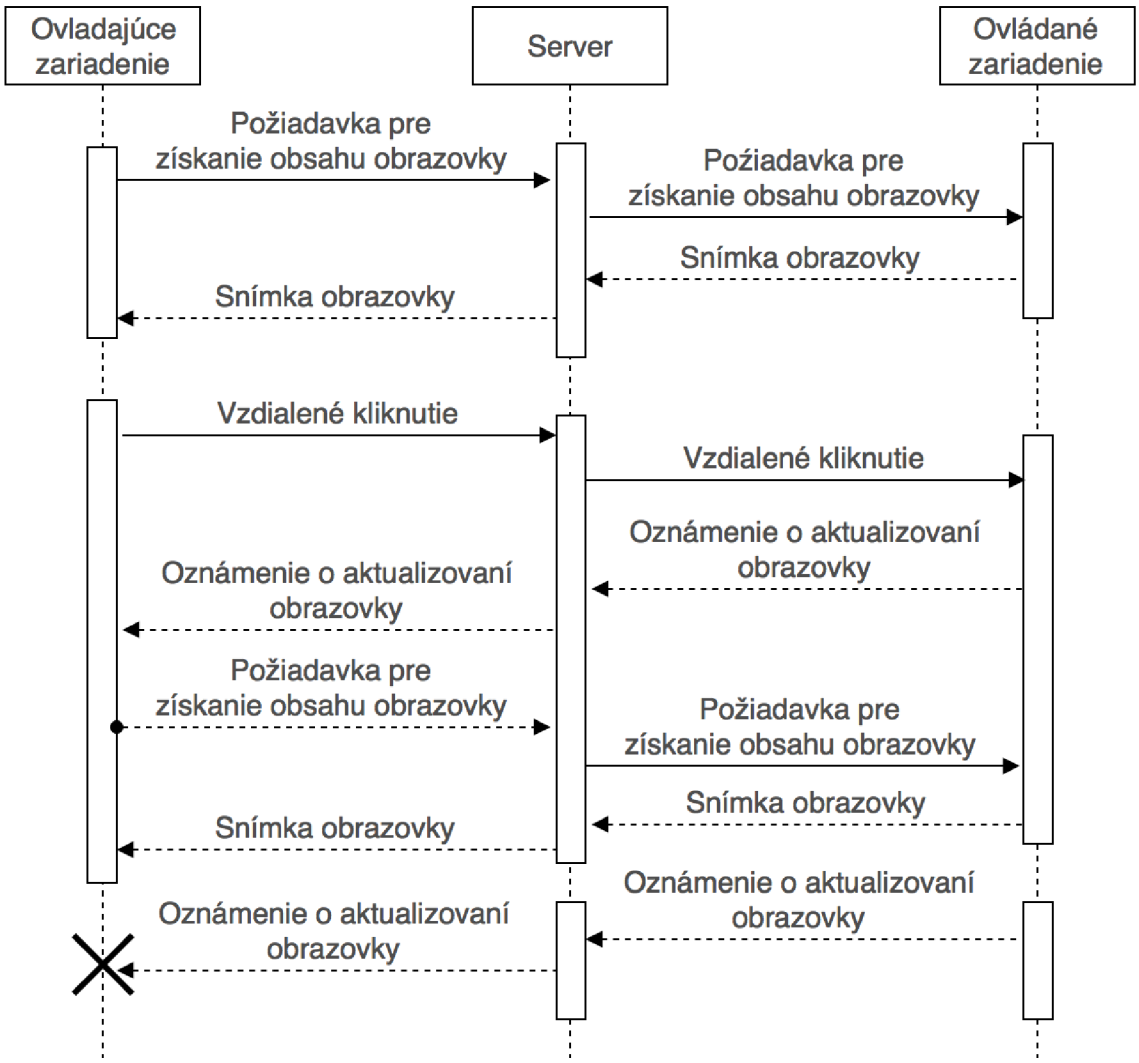
Vykonávanie stlačenia dotykovej obrazovky programovo je z bezpečnostných dôvodov možné iba v rámci aplikácie. Aj preto bola aplikácia vytvorená ako domovská aplikácia (launcher). Aplikácia je teda v prípade zvolenia používateľom spúšťaná hneď pri štarte systému. A je viditeľná pokiaľ používateľ nespustí inú aplikáciu. Ďalším dôvodom bolo zmenšenie rizika, že by sa používateľ dostal von z aplikácie a nevedel aplikáciu znova spustiť.

Pre zaregistrovanie možnosti domovskej aplikácie je nutné v manifeste deklarovať do kategórie `MAIN` filtrov záujmov kategórie `HOME` a `DEFAULT`. Aby používateľ nastavil aplikáciu ako domovskú aplikáciu pri vstupe do aplikácie, bolo nutné vyvolať pri spustení ponuku s výberom domovskej aplikácie. Ponuku zobrazuje metóda `showPrefferedActivities`. Princíp je v tom, že je v manifeste ako launcher deklarovaná prázdna aktivita, ktorá je zakázaná a po jej povolení sa pri stlačení tlačidla domov zobrazí ponuka. Metóda pomocou triedy `PackageManager` a metódy `setComponentEnabledSetting` povolí zakázanú aktivitu, pomocou spustenia zámeru (objekt triedy `Intent`) s kategóriou `HOME` metódou `startActivity` sa zobrazí ponuka a následne sa aktivita zase zakáže. Vďaka opätovnému zakázaniu aktivity sa používateľovi prázdna aktivita nezobrazí v ponuke.

Vzdialené ovládanie pomocou vzdialenej plochy sa začína príchodom socket udalosti pre získanie snímku obrazovky. Odoslanie snímku obrazovky vykonáva metóda `sendScreen`. Snímok je vytvorený z cache pamäte pre vykreslovanie. K tomu sa využívajú metódy koreňového objektu `View` `setDrawingCacheEnabled` (pre povolenie cache), `buildDrawingCache` (vynúti vytvorenie cache pre vykreslovanie pokiaľ už nie je platná cache vytvorená). Metóda `getDrawingCache` vráti objekt `Bitmap` reprezentujúci snímok obrazovky vo forme bitmapového obrázka. Z obrázka je metódou compress vytvorený stream bajtov ukladaný do objektu `ByteArrayOutputStream`. Metóde sa parametrom nastavuje požadovaný formát obrázka `JPEG` a požadovaná kvalita obrázku (0 – 100). Kvôli šetreniu dátového toku bola nastavená kvalita na tridsať percent, čo je dostatočné kvalita obrázku pre ovládanie. Zo streamu je metódou `toByteArray` vytvorené pole bajtov, ktoré je pomocou metódy `encodeToString` triedy `Base64` zakódované kódovaním BASE64 do reťazca. Reťazec je následne odoslaný cez socket službou `SocketConnectionService` na ovládajúce zariadenie.

Príchodom socket správy pre programové vykonanie dotyku na obrazovke (kliknutia) začne sekvencia nasledovných akcií. Zo správy sa vyextrahujú súradnice kliknutia a prevedú sa z relatívnych na absolútne hodnoty podľa rozlíšenia obrazovky daného zariadenia. Vytvorí sa inštancia triedy MotionEvent pomocou metódy obtain, ktorej sú parametrami predané hodnoty času stlačenia, času udalosti, akcie, súradnice kliknutia, stav špeciálnych kláves (napríklad zapnutý caps lock, v tomto prípade to nie je potrebné preto je použité číslo nula ako indikátor žiadneho špeciálneho stavu). Ako akcie sú použité `MotionEvent.ACTION_DOWN` a `MotionEvent.ACTION_UP`. Pre vloženie objektov reprezentujúcich kliknutie do systému sa používajú metódy `dispatchTouchEvent` (volaná na koreňový View objekt) a `injectPointerEvent`. Metóda `injectPointerEvent` vkladá do systému objekt `MotionEvent` a používa k tomu systémové metódy `injectPointerEvent` a `injectInputEvent`. Na verziách systému starších ako Android 4.1 sa využíva metóda `injectPointerEvent` triedy `WindowManager`, na ostatných verziách sa používa metóda `injectInputEvent` triedy `InputManager`. Tieto metódy boli pred verziou Android 4.4 skryté a pre spätnú kompatibilitu sa k nim pristupuje pomocou tzv. zrkadlenia, ktoré je uskutočnené metódami getMethod pre získanie skrytej metódy a invoke pre jej zavolanie.

Aktualizácia obsahu po vykonanom kliknutí je vyriešená preťažením metódy `dispatch TouchEvent`, v ktorej sa odošle socket správa ovládajúcemu zariadeniu ako oznámenie o no-



Obr. 6.1: Na obrázku je zobrazený zjednodušený sekvenčný diagram znázorňujúci komunikáciu aplikácií pri ovládaní pomocou vzdialenej plochy. Prvou akciou je načítanie obsahu obrazovky po spustení funkcie vzdialená plocha. Ďalšou akciou je vykonanie kliknutia a následné načítanie nového obsahu obrazovky. Treťou akciou je oznámenie o aktualizovaní obrazovky spôsobené tým, že používateľ vzdialeného zariadenia vykonal kliknutie na zariadení. Ovládajúce zariadenie už nemá vzdialenú plochu spustenú a na oznámenie neodpovie.

vom obsahu obrazovky. V prípade, že je v ovládajúcej aplikácii ešte otvorené ovládanie, príde ako odpoveď socket správa pre poslanie snímku obrazovky. Keďže používateľ môže vykonať viacnásobné kliknutie, je pridaná kontrola aby sa správa pre obnovenie snímky neodosielala častejšie ako jednu sekundu.

6.3 Služba pre komunikáciu cez socket

Aplikácie používajú po pripojení používateľa na komunikáciu so serverom framework Socket.io. Táto komunikácia je v aplikáciách realizovaná v službe `SocketConnectionService`. Dôvodom pre vytvorenie služby na tento účel bolo oddelenie, aby sa pripojenie nadviazalo a spravovalo na jednom mieste a prepojenie nebolo závislé na životných cykloch rôznych komponentov ako aktivity a fragmenty. Služba umožňuje beh aj po ukončení aplikácie a môže teda bežať na pozadí, riešiť prichádzajúce udalosti a prípadne spustiť aplikáciu (napríklad prichádzajúci hovor). Pre komunikáciu s ostatnými komponentami služba používa rozhranie `LocalBroadcastManager`.

Prácu so socketom sprístupňuje inštancia triedy `SocketIOClient` a je vykonávaná v samostatnom vlákne, aby vstupno-výstupné operácie nezdržovali vlákno pre prácu s používateľským rozhraním. Inštancii sú po pripojení socketu metódou `connect` zaregistrované načúvacie objekty pre udalosti ako odpojenie, obnovenie spojenia, chyba spojenia a príchod správ zo serveru. Každý tento objekt vykonáva príslušné akcie. Pre odosielanie správ na server bola vytvorená metóda safeEmit, ktorej sa parametrami predá názov správy a objekt triedy `JSONArray` obsahujúci obsah správy. Metóda skontroluje pripojenie socketu a v prípade, že je pripojený správu odošle metódou `emit`.

6.4 Hovor a videohovor

K sprostredkovaniu hovoru a videohovoru cez internet je použitá technológia WebRTC, nakoľko je vytvorená na tento účel a je široko podporovaná. Kvôli tomu, že je produkt určený na používanie celosvetovo, je použitá komerčná služba OpenTok. Keďže majú viac serverov rozmiestených po svete, zariadenie sa pripojí na server umiestnený bližšie a kvalita spojenia bude lepšia. Okrem kvality spojenia je službou pomocou STUN a TURN serverov zabezpečené pripojenie naprieč firewallom, NAT. Služba poskytuje knižnice pre integráciu do serverovej aj klientských aplikácií.

Na serveri je vytvorený komunikačný protokol pre signalizáciu hovoru. Klienti si vymieňajú signalizačné správy cez socket. Po prijatí hovoru server prostredníctvom knižnice opentok vytvorí funkciou `createSession` spojenie. K spojeniu sa klienti budú pripájať prostredníctvom príslušných knižníc s parametrami identifikátor spojenie a token, ktoré im server po vygenerovaní pošle. Generovanie tokenu je realizované funkciou `generateToken`.

V Android aplikáciách sú v službe `SocketConnectionService` implementované signalizačné správy pre hovor. Po prijatí hovoru sa aplikácia pripája k spojeniu. Vytvorí sa objekt `Session`, ktorého konštruktor vyžaduje kontext aplikácie, API kľúč a identifikátor spojenia. API kľúč nie je zakódovaný v aplikáciách, ale je načítaný zo serveru, aby nenastal problém s jeho zmenou v prípade potreby. Objektu `Session` sa zaregistruje načúvací objekt (v tomto prípade aktivita), ktorý implementuje metódy rozhrania `Session` volané pri rôznych nových stavoch spojenia. Následne sa spojenie pripojí zavolaním metódy connect s tokenom ako parametrom.

Tabuľka 6.1: Metódy rozhraní knižnice OpenTok, ktoré je nutné naimplementovať v načúvacom objekte (listener).

Rozhranie	**Metóda**	**Okolnosť**
SessionListener	onConnected	Pripojenie spojenia.
	onDisconnected	Odpojenie od spojenia.
	onStreamReceived	Prijatie prúdu dát (stream).
	onStreamDropped	Odpojenie prúdu dát (stream).
	onError	Chyba spojenia.
PublisherListener	onStreamCreated	Vytvorenie prúdu dát (stream).
	onStreamDestroyed	Zrušenie prúdu dát (stream).
	onError	Chyba publikovania.
VideoListener	onVideoDataReceived	Príchod dát s videom.
	onVideoDisabled	Zastavenie príjmu videa (napr. druhá strana zakázala odosielanie videa).
	onVideoEnabled	Druhá strana povolila odosielanie videa.
	onVideoDisableWarning	Server zistí zlú kvalitu videa, po zhoršení bude video zakázané.
	onVideoDisableWarningLifted	Server zistí zlepšienie kvality videa a nehrozí vypnutie videa.

Kapitola 7

Testovanie

Pri vývoji aplikácie bola snaha podporovať čo možno najviac zariadení. S postupom riešenia projektu prišli určité obmedzenia, ktoré vylúčili niektoré zariadenia. Aj napriek počiatočnej snahe podporovať zariadenia až do Android verzie 2.2 bola kvôli knižniciam OpenTok pre volanie nastavená minimálna verzia systému Android 4. Ďalšími podmienkami sú hardvérové prostriedky kamera a mikrofón kvôli videohovoru. Aplikácia pre seniorov bola z dôvodu špeciálneho používateľského rozhrania určená pre použitia iba na tabletoch na rozdiel od aplikácie pre juniorov, ktorá je dostupná aj pre mobilné telefóny.

Počas celej doby vývoja aplikácií boli aplikácie priebežne testované na rôznych reálnych zariadeniach a emulátoroch z Android SDK s rôznymi konfiguráciami. Dostupné zariadenia, ale predstavujú veľmi malé zastúpenie rôznych zariadení, ktoré môžu aplikáciu spustiť. Nástroj služby Google Play - Developer Console zobrazuje počet 10 691 podporovaných zariadení aplikácie určenej pre tých čo budú vzdialene ovládať a 2643 podporovaných zariadení aplikácie pre seniorov (menší počet spôsobilo najmä obmedzenie na tablety). K distribúcii aplikácií pre účely testovania bola použitá služba Google Play, konkrétne jej možnosti alfa a beta testovania.

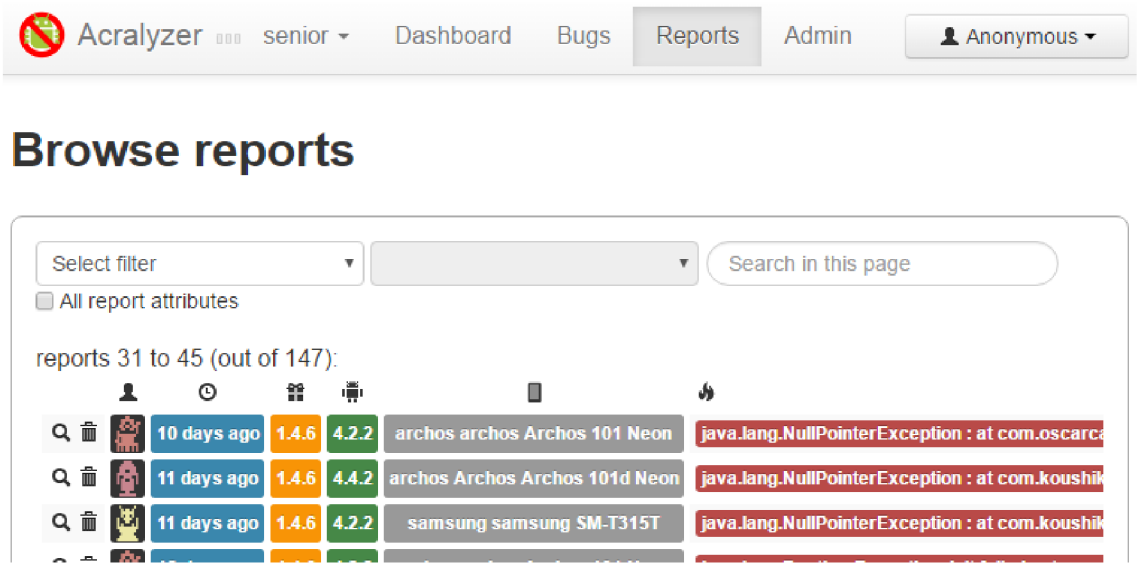
Pred zverejnením aplikácií boli nahrané do alfa testovania. V tejto fázy prebiehalo testovanie interne v rámci firmy. Do testovania boli zapojení zamestnanci a partneri firmy prostredníctvom ich Google používateľských účtov. Toto testovanie prinieslo cenné informácie z hľadiska problémov s funkčnosťou, ale aj z pohľadu vylepšení ovládania a návrhov nových funkcionalít.

Google Play umožňuje zbieranie hlásení o problémoch používania aplikácie (napríklad pády aplikácie, nereagovanie aplikácie), používateľ však musí povoliť odoslanie hlásenia. Pre získanie hlásení o pádoch aplikácie bez potvrdenia používateľom bola použitá open source knižnica ACRA[1] (Application Crash Reports for Android). Knižnica poskytuje veľké množstvo volieb ako výber spôsobu odosielania hlásení, definovanie interakcie s používateľom, výber odosielaných údajov. Umožňuje pridanie vlastných dát do hlásení a odoslanie hlásenia aj bez pádu aplikácie. Pre zber hlásení bol použitý server z open source projektu Acralyzer[2]. Acralyzer používa pre ukladanie CouchDB, ktorú využíva aj pre webovú aplikáciu (CouchApp). Webová aplikácia zobrazuje hlásenia s možnosťou filtrovania. Dáta sú z Android aplikácií pre bezpečnosť používateľových dát odosielané šifrovane cez HTTPS. Pre odosielanie je použitá metóda požiadavku PUT a dáta sú vo formáte JSON. Databáza je zabezpečená a požiadavka musí obsahovať správne prihlasovacie údaje pre autorizáciu.

[1] https://github.com/ACRA/acra
[2] https://github.com/ACRA/acralyzer

Tabuľka 7.1: Mobilné zariadenia, na ktorých bola aplikácia odskúšaná.

Názov zariadenia	Typ	Veľkosť	Rozlíšenie displeja [px]	RAM	Verzia systému
OnePlus One	mobil	5,5"	1080x1920	3 GB	6.0.1
LenovoYoga Tablet 10 HD+	tablet	10,1"	1920x1200	2 GB	4.4.2
Lenovo Yoga Tablet 10	tablet	10,1"	1280x800	1 GB	4.4.2
3Q RC1018C	tablet	10,1"	1280x800	1 GB	4.3
Sony Xperia T	mobil	4,55"	720x1280	1 GB	4.3
Alcatel Pixi 3 (8) 3G	tablet	8"	1280x800	1 GB	5.0.1
Acer Iconia Tab A210	tablet	10,1"	1280x800	1 GB	4.1.1
Samsung Galaxy S4	mobil	5"	1080x1920	2 GB	5.0.1
Xiaomi Note 2	mobil	5,5"	1080x1920	2 GB	5.0.1
Doogee F5	mobil	5,5"	1080x1920	3 GB	5.1
GoClever Tab TERRA 70	tablet	7"	800x400	0.5 GB	4.1
Samsung Galaxy Tab 3 Lite 7.0	tablet	7"	1024x600	1 GB	4.2.2



Obr. 7.1: Na obrázku je vidieť zobrazenie hlásení vytvorené knižnicou ACRA vo vizualizácii vytvorenej prostredníctvom projektu Acralyzer.

Pre získanie spätnej väzby od používateľov a zabezpečenie technickej podpory bola do aplikácie pre juniorov integrovaná komerčná služba Zendesk. Aplikácie boli zverejnené v službe Google Play. Ku dňu 16. 5. 2016 má aplikácia pre seniorov 1149 stiahnutí, aplikácia pre juniorov 662 stiahnutí a podľa databázy je zaregistrovaných 1813 používateľov. Od používateľov prichádzali dotazy, pripomienky, návrhy na vylepšenia a niektoré z nich ovplyvnili vzhľad a funkcionalitu ďalších verzií aplikácie. Vďaka nastavenému zbieraniu chýb a podrobným hláseniam, bolo možné vyskytnuté chyby dohľadať a opraviť.

Bola vytvorená Google skupina pre používateľov aplikácie. Táto skupina bola pridaná do beta testovania v nástroji Developer Console. Používatelia prihlásením do beta testovania získali prístup k beta verziám aplikácií a mali tak možnosť skôr vyskúšať vylepšenia. Pridanie beta testovania bolo dôležitým krokom ku kvalitnejším stabilným verziám aplikácií. Po vytvorení novej verzie prejde aplikácia najskôr interným alfa testovaním, potom je verzia sprístupnená beta testerom a až potom pokiaľ neboli zistené nejaké problémy sa aplikácia dostane do produkcie. Za cenu pomalšieho zverejnenia novej verzie je vývoj flexibilnejší, keďže po objavení nedostatkov pri testovaní sa daná verzia na trh nedostane a začne sa vyvíjať nová verzia, ktorá znova prejde procesom testovania. Takýmto postupom je možné opraviť chyby predtým, akoby sa dostali do zariadení väčšieho množstva používateľov. To je v prípade mobilných aplikácii veľmi dôležité vzhľadom k tomu, že aktualizáciu s opravami si používateľ nemusí nainštalovať.

Kapitola 8

Záver

Práca sa zaoberá možnosťami vzdialeného prístupu k mobilným zariadeniam. Cieľom bolo preštudovať existujúce možnosti vzdialeného prístupu a ovládania mobilných zariadení, navrhnúť a vytvoriť architektúru a rozhranie pre takýto systém.

Preštudoval som existujúce možnosti vzdialeného prístupu a ovládania mobilných zariadení. V rámci analýzy problematiky som popísal možnú funkcionalitu vzdialeného ovládania. Pre funkcionalitu vzdialenej plochy som vypracoval možnosti riešenia spolu s ich výhodami a nevýhodami. Navrhol som architektúru systému s ohľadom na potreby aplikácií ako napríklad komunikácia v reálnom čase. Vytvoril som návrh rozhrania pre vzdialený prístup, súčasťou ktorého sú dve aplikácie – jedna slúži na ovládanie a druhá je ovládaná.

Návrh sa mi podarilo implementovať. Výsledkom práce sú Android aplikácie s rozhraním pre vzdialenú zmenu nastavení a ovládanie pomocou vzdialenej plochy zverejnené v službe Google Play a server poskytujúci rozhranie pre komunikáciu aplikácií a ukladanie dát používateľov.

Veľkou výhodou bola integrácia výsledkov práce do komerčného produktu. Vďaka tomu mohlo moje riešenie vyskúšať viac ako 1800 používateľov zaregistrovaných do služby Oscar Senior. Ďalší vývoj produktu je ovplyvnený spätnou väzbou používateľov. Podľa dotazov používateľov bola najväčším problémom registrácia a prepojenie účtov a nepochopenie rozdelenia na dve aplikácie. Z týchto dôvodov sa firma Oscar Tech rozhodla vytvoriť novú aplikáciu, ktorá bude obsahovať obidve strany ovládania. Väčšinu z riešenia vzdialeného prístupu a ovládania je však možné s miernymi úpravami použiť v novej aplikácii.

Literatúra

[1] Allen, G.: *Android 4: průvodce programováním mobilních aplikací.* Computer Press, 2013, iSBN 978-80-251-3782-6.

[2] Clarissa, P.: *Learning responsive Web design: a beginner's guide.* O'Reilly, 2014, iSBN 978-1-4493-6294-2.

[3] Jeff, M.: *Professional mobile application development.* Wiley, 2012, iSBN 978-1-118-20390-3.

[4] Jonathan, P.: *Xamarin Cross-platform Application Development - Second Edition.* Packt Publishing, 2015, iSBN 978-1-78439-788-3.

[5] Matt, N.: *Programming iOS 9: Dive Deep into Views, View Controllers, and Frameworks.* O'Reilly, 2015, iSBN 978-1-491-93685-6.

[6] Murphy, M. L.: *Beginning Android 2. New York.* Apress, 2010, 24 s., iSBN 978-1-4302-2629-1.

[7] Ujbányai, M.: *Programujeme pro Android. Vyd. 1. Praha.* Grada, 2012, iSBN 978-80-247-3995-3.

[8] WWW stránky: NoSQL. `https://en.wikipedia.org/wiki/NoSQL`, [cit. 2015-12-14].

[9] WWW stránky: Socket.IO. `https://en.wikipedia.org/wiki/Socket.IO`, [cit. 2015-12-15].

[10] WWW stránky: Zappajs. `https://zappajs.github.io/zappajs`, [cit. 2015-12-18].

[11] WWW stránky: Introduction to Redis. `http://redis.io/topics/introduction`, [cit. 2015-12-19].

[12] WWW stránky: TeamViewer - Spoločnosť. `https://www.teamviewer.com/sk/company/company.aspx`, [cit. 2015-12-19].

[13] WWW stránky: MongoDB. `https://en.wikipedia.org/wiki/MongoDB`, [cit. 2015-12-20].

[14] WWW stránky: Node.js. `https://en.wikipedia.org/wiki/Node.js`, [cit. 2015-12-20].

[15] WWW stránky: Node.js. `https://nodejs.org/en`, [cit. 2015-12-20].

[16] WWW stránky: Express. `http://expressjs.com`, [cit. 2015-12-21].

Prílohy

Zoznam príloh

Príloha A

Obsah CD

Adresárová štruktúra CD:

- `src` - zdrojové súbory aplikácie
- `thesis` - zdrojový súbory písomnej správy