ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lesák** Jméno: **Jan** Osobní číslo: **466231**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Kontextově založený framework s ohledem na svoje okolí

Název bakalářské práce anglicky:

Pokyny pro vypracování:

Softwarové aplikace jsou navrženy s konkrétním účelem, který určují vlastníci firem na základě individuálních požadavků. Interakce uživatelského systému je specifikována ve fázi analýzy. Tato fáze určuje vstupy, výstupy, interakce a jiné. Existují techniky jak určit zda uživatel, který je v konkrétním kontextu, by měl být požádán o vyplnění textových polí, které nejsou potřebné pro splnění specifické obchodní úlohy [1,2,3]. Tyto techniky jsou navíc schopny určit, zda se mají nebo nemusí zobrazovat pole, stejně jako jak by mohl systém interagovat s uživatelem.
Cíle práce jsou následující:
1) Vytvořte analýzu problematiky aplikací, které umí reagovat na kontext okolního prostředí
2) Prozkoumejte možnosti frameworku [1,2,3] a rozšiřte ho o možnost reagovat na kontext okolí
3) Vytvořte pomocí něj testovací aplikaci, tu otestujte na předem definovaných scénářích (například poslání bankovního příkazu)
4) Vyhodnoťte testy

Seznam doporučené literatury:

1.TOMÁŠEK, M. and T. ČERNÝ. Automated User Interface Derivation for Remote Data in Standalone Apps. In: HUSNÍK, L., ed. Proceedings of the 19th International Scientific Student Conferenece POSTER 2015. 19th International Student Conference on Electrical Engineering, Praha, 2015-05-14. Praha: Czech Technical University in Prague, 2015. pp.1-6. ISBN 978-80-01-05499-4.
2. TOMÁŠEK, M. and T. ČERNÝ. On Web Services UI In User Interface Generation in Standalone Applications. In: ČERNÝ, T. and E.S. NADIMI, eds. Proceeding of the 2015 Research in Adaptive and Convergent Systems (RACS 2015). Research in Adaptive and Convergent Systems, Prague, 2015-10-09/2015-10-12. New York: ACM, 2015. pp. 363-368. ISBN 978-1-4503-3738-0. DOI 10.1145/2811411.2811537.
3.Tomasek, Martin, and Tomas Cerny. "Context-Aware User Interface Field Classification." IT Convergence and Security (ICITCS), 2016 6th International Conference on. IEEE, 2016.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **05.02.2019** Termín odevzdání bakalářské práce: [redacted]

Platnost zadání bakalářské práce: **20.09.2020**

[redacted]
Ing. Jiří Šebek
podpis vedoucí(ho) práce

[redacted]
podpis vedoucí(ho) ústavu/katedry

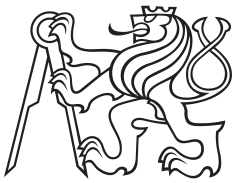
prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

20.5.2019
Datum převzetí zadání

[redacted]
Podpis studenta



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Kontextově založený framework s ohledem na svoje okolí

Jan Lesák

Vedoucí: Ing. Jiří Šebek
Obor: Softwarové inženýrství a technologie
Květen 2019

Poděkování

Děkuji vedoucímu práce panu Ing. Jiřímu Šebkovi za jeho čas, pomoc a rady, které mi pomohly dokončit tuto práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 20. května 2019

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací frameworku, který získá informace o okolním kontextu zařízení a tyto informace dále vyhodnotí. Některé frameworky již pracují s uživatelským kontextem a na základě těchto dat generují dynamické uživatelské rozhraní. Práce se věnuje rozšíření působnosti sběru takových dat nejen na uživatelovo zařízení, ale také na uživatelův okolní kontext, tedy zařízení v blízkém okolí. Získáním informací o okolním kontextu a porovnáním tohoto kontextu se známými bezpečnými kontexty framework vyhodnotí, zda se uživatel nachází ve známém prostředí, ve kterém se uživatel vyskytuje často. Na základě toho může aplikace využívající tento framework přizpůsobit svoje chování a především míru zabezpečení.

Klíčová slova: okolní kontext, kontextově orientovaný framework, WiFi, Bluetooth, bezpečnost prostředí, framework, Java, Android

Vedoucí: Ing. Jiří Šebek

Abstract

This bachelor thesis is focused on the design and implementation of a context recognition framework. This framework collects and evaluates information about the user's surrounding context. Some frameworks evaluate the device context and generate a dynamic user interface based on this context. The goal of this paper is to extend the scope of collecting context data not only on the device context, but also on the user's surrounding context and devices near the user. The framework collects information about the surrounding context and compares it with known safe contexts. Based on this comparison the framework evaluates whether the user is in his familiar environment where he occurs frequently or if he is in unknown surroundings. An application that uses this framework can then set its behavior and security level according to this information.

Keywords: surrounding context, context-aware framework, WiFi, Bluetooth, surroundings safety, framework, Java, Android

Title translation: Context-oriented framework based on the surroundings

Obsah

Obrázky

Tabulky

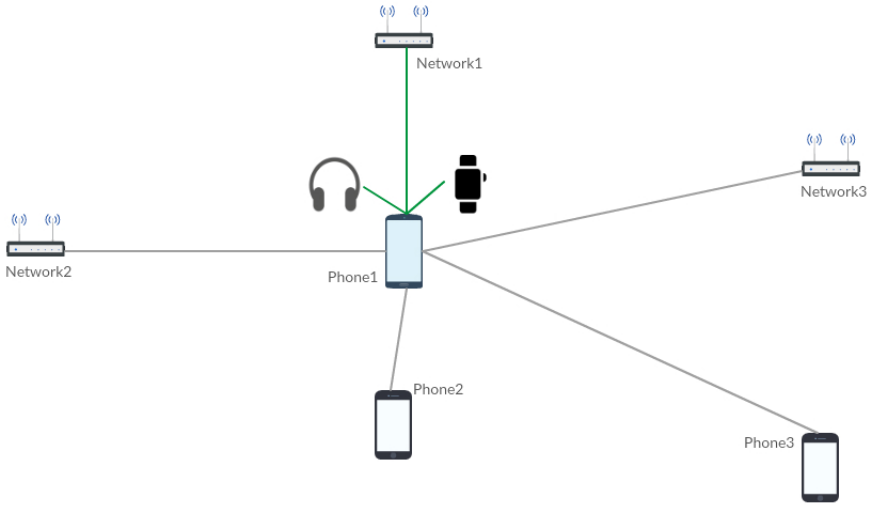
Kapitola 1

Úvod

Uživatelé velmi často provádějí rutinní úkony v pravidelných časech a na stejném místě. Například zadání platby z mobilního internetového bankovnictví uživatelé pravděpodobně neprovádějí na veřejnosti mezi lidmi v MHD, ale až doma v klidu, kde je nic neruší. Akce podobného bezpečnostního charakteru, jako je zadání platebního příkazu, většinou obsahují vícefázové ověření. To se výhodné, pokud je uživatel v neznámém prostředí nebo na veřejnosti, kde je vyšší riziko odcizení zařízení a pokusu o neoprávněné zadání platby. Pokud je však uživatel ve svém běžném bezpečném prostředí, například doma, může být vícefázové zabezpečení zbytečné a uživatele bude zdržovat od dalších úkonů. Pokud by si aplikace podle připojených zařízení a zařízení okolí uživatele zapamatovala, že daný kontext je běžný a bezpečný, při příštím podobném úkonu by nebylo vyžadováno vícefázové ověření. Takové rutinní akce by se tímto zjednodušily a zrychlily. V případě, že uživatel chce provést danou akci v jiném než známém kontextu, opět se zvýší bezpečnostní opatření a bude vyžadováno ověření. Tento princip nemusí být využit jen po stránce ověřování, ale i v dalších aspektech, které vývojáři aplikací uznají za vhodné. V neznámém prostředí může aplikace například skrýt některé citlivé osobní údaje, které by mohly být vypozorovány lidmi v okolí.

Na obrázku 1.1 je znázorněno jednoduché schéma možného okolního kontextu. Modrý telefon Phone1 představuje chytrý telefon uživatele připojený k WiFi síti Network1. K tomuto telefonu jsou dále přes Bluetooth připojeny bezdrátová sluchátka a chytré hodinky. Framework načte okolní kontext a zjistí, že se zde nachází také telefony Phone2 a Phone3 a WiFi sítě Network2 a Network3. Framework vypočítá i přibližnou vzdálenost zařízení a ví, že Phone2 představuje člověka, který se nachází v blízkosti uživatele. Pokud je tento telefon neznámý, je pravděpodobně neznámý i daný člověk. To může

znamenat určitou bezpečnostní hrozbu. Tuto informaci framework vyhodnotí a aplikace reaguje zvýšeným zabezpečením (např. skrytím citlivých informací na obrazovce).



Obrázek 1.1: Schéma okolního kontextu.

1.1 Využití

Využití můžeme najít především v aplikacích pracujících s osobními údaji nebo se zabezpečením heslem. Mobilní bankovnictví na svém telefonu využívá v současné době 61%[1] lidí vlastnících chytrý telefon. Aplikace by tak v neznámém, málo bezpečném prostředí, například nezobrazovala citlivá data. Naopak ve známém prostředí (např. doma) by nepožadovala vícefázové ověření při zadávání platby. Nemusí se ale jednat přímo o mobilní bankovnictví. Framework může najít své využití i v různých fin-tech aplikacích pro monitorování výdajů.

Vedlejší funkce frameworku, jako je získání seznamu okolních zařízení Bluetooth a WiFi, mohou být také využity aplikacemi, které s těmito daty pracují. Například pro různé aplikace věnujícím se sdílení mezi zařízeními nebo pro hry může být využitelná funkce zjištění Bluetooth zařízení v okolí. Aplikace by tak věděla, že se v blízkém okolí nachází nějaký další telefon nebo počítač a mohla by na tuto skutečnost reagovat.

Kapitola 2

Rešerše

Kontextově orientované aplikace se vyznačují schopností rozpoznat kontext, ve kterém se nachází a reagovat na něj[2]. To znamená, že aplikace se v různých situacích může zachovat jinak, právě kvůli rozdílnému kontextu. Nebo že aplikace sama provede nějakou činnost, právě proto, že se změnil kontext. Typický příklad je automatické otočení displeje. Aplikace, sleduje fyzickou orientaci telefonu. Pokud uživatel otočí telefon na šířku, tato změna kontextu je rozpoznána a i obrazovka se otočí do dané pozice. Další příklad, který je hodně používaný, je automatické nastavení jasu, kdy telefon pomocí senzoru sleduje úroveň světla v okolí a přizpůsobuje tomu jas displeje. Kontextově orientované frameworky a aplikace uživateli poskytují informace a funkce na základě toho, v jakém kontextu se právě nachází. Aplikace sleduje data jako jsou stav baterie, poloha nebo způsob připojení k internetu a následně je vyhodnocuje. Na základě toho je možné se například rozhodnout, zda budou detaily o produktu zobrazeny ve formě videa a obrázků, nebo v podobě prostého textu, pokud telefonu dochází baterie a není připojen k WiFi[4]. Praktické využití můžeme najít například v procesu nákupu letenky. Vezměme si příklad, kdy uživatel nakupuje na mobilním telefonu letenku na další den. Aplikace zpracuje uživatelův kontext a zjistí, že je na mobilním zařízení, k internetu je připojen přes mobilní data a zařízení je v pohybu. To, že uživatel nakupuje letenku na poslední chvíli a používá k tomu mobilní telefon značí, že pravděpodobně spěchá. Tyto informace aplikace vyhodnotí a uživateli se pak zobrazí pouze vstupy nezbytně nutné pro vyhledání letenky, tedy místo odletu a příletu a datum[3].

Tento přístup však bere ohled pouze na samotného uživatele a jeho kontext. Při rozšíření oblasti sběru dat i na okolní zařízení získáme mnohem větší představu o tom, v jakém kontextu se uživatel nachází. Po přidání okolního

kontextu můžeme vyhodnocovat bezpečnost a důvěryhodnost místa, kde se uživatel nachází. Na základě tohoto kontextu rozhodneme, jestli bude požadováno vícefázové ověření, nebo zda budou na displeji telefonu skryta citlivá data.

2.1 Existující řešení

Existují frameworky, které se zabývají převážně kontextem uživatele. Většinou zpracovávají data ze senzorů zařízení a jeho polohu. V této části je popsána funkcionalita jednotlivých řešení a důvod, proč nemohou být použita pro načtení okolního kontextu.

2.1.1 Google Awareness API

Google Awareness API[5] ve sjednocené formě poskytuje informace o sedmi různých kontextových datech:

- Čas
- Poloha (latitude a longitude)
- Místo (např. park, město)
- Aktivita uživatele (chůze, běh atd.)
- Vysílající Bluetooth zařízení (beacons)
- Informace o připojení sluchátek
- Počasí

Jeho velkou výhodou je také to, že se stará o vše, co je se sběrem těchto dat spojené, včetně spotřeby energie a systémových zdrojů. Vývojáři tedy stačí toto API použít a o optimalizaci pro co nejmenší spotřebu energie se nemusí starat. Google Awareness API se skládá ze dvou částí, kde každá poskytuje zcela odlišnou funkcionalitu. Aplikace předá Fence API určitá kritéria. Ve chvíli, kdy se nějaké kritérium splní, aplikace o této události obdrží zprávu a může na to reagovat. Kritéria se mohou skládat například z polohy, uživatelovy aktuální aktivity nebo zda je v blízkosti konkrétního

BLE vysílače. Naopak Snapshot API poskytuje detailní data o kontextu ve výše specifikovaných kategoriích. Pro zjištění okolního kontextu může být z části využito Snapshot API, zejména pro informace o poloze zařízení. Co se týče okolních zařízení, Google Awareness API dokáže snímat pouze takzvané BLE majáky, využívané především pro IoT. Klasická Bluetooth zařízení tedy tímto způsobem není možné načíst.

2.1.2 Radar

Radar[6] je platforma pro Android i iOS zpracovávající kontextové informace o poloze zařízení. Celá logika Radaru se skládá z Places, Geofences a Insights. Geofence je manuálně definované místo nebo oblast určená pro konkrétní aplikaci. V praxi to může být například domov, nebo práce. Places jsou místa z databáze Facebook Places, kterou využívá Facebook a Instagram a obsahuje více než 140 milionů[7] různých míst po celém světě. Pomocí Insights pak můžeme sledovat události vyvolané změnou polohy zařízení a vstoupení nebo opuštění definované Geofence či Place. Tento framework zajišťuje kvalitní sledování kontextu polohy zařízení, ale používá pouze informace o fyzické poloze a s okolními zařízeními nepracuje vůbec.

Kapitola 3

Analýza

Cílem frameworku je získat informace o zařízeních v okolí. Bude tedy nutné určit pomocí jakých technologií se budou dané informace zjišťovat, případně analyzovat možné rekurzivní hledání pomocí ostatních zařízení. Framework je zaměřen převážně na operační systém Android. OS Android je stále nejrozšířenější operační systém na mobilních zařízeních a na jaře roku 2019 se nacházel na 75%[8] všech mobilních zařízení. Mimo to je Android více otevřený co se týče oprávnění na systémové zdroje a práce s WiFi a Bluetooth, než druhý nejpoužívanější systém iOS.

3.1 Vývoj Android aplikací

Operační systém Android je open source systém založen na Linuxovém jádře a je používaný výhradně na mobilních zařízeních. Aplikace pro Android jsou vyvíjené především v programovacích jazycích Kotlin, Java a C++, přičemž tento framework bude psán v Javě. Ve výchozím stavu po instalaci mají všechny aplikace omezené oprávnění pro přístup k systémovým zdrojům. O konkrétní systémové zdroje a uživatelská data musí aplikace požádat a uživatel tuto žádost může a nemusí povolit. To platí například o přístupu k poloze zařízení, ovládání Bluetooth nebo práci se soubory uloženými v zařízení. Právě přístup k poloze a ovládání Bluetooth je vyžadován tímto frameworkem.

3.2 Použité technologie

V následujících odstavcích jsou stručně popsány hlavní technologie použité při vývoji tohoto frameworku.

3.2.1 WifiManager

WifiManager[9] umožňuje veškerou práci s WiFi na daném zařízení. Okolní sítě jsou načteny pomocí skenování přístupových bodů. Každá objevená síť obsahuje SSID a MAC adresu vysílače, které mohou být použity pro identifikaci a také sílu signálu. Pro zobrazení okolních WiFi sítí je potřeba aplikaci povolit přístup k poloze. To je nutné, protože existují databáze s přístupovými body a jejich polohou. Ze seznamu okolních sítí se tím pádem dá určit přibližná poloha zařízení. Tuto metodu určování polohy používá i samotný Android z důvodu nižší spotřeby baterie oproti GPS. Stejné informace dokáže WifiManager získat i o aktuálně připojené WiFi síti.

3.2.2 BluetoothManager a BluetoothAdapter

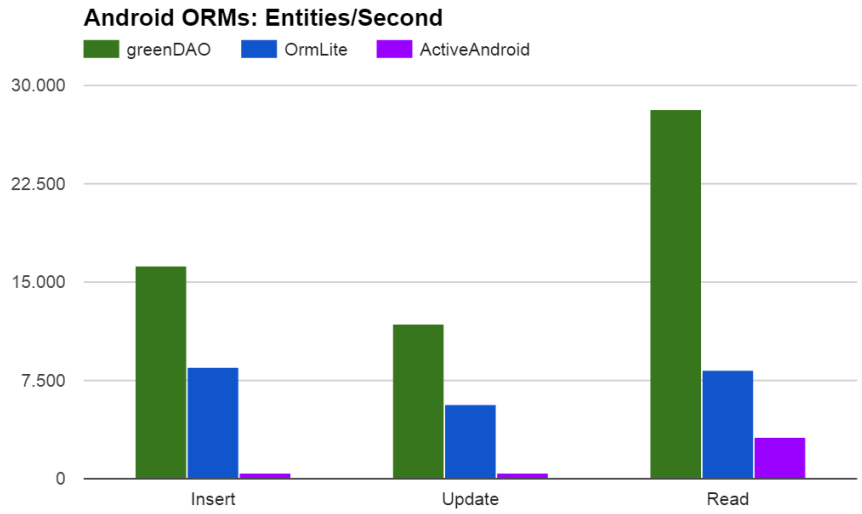
BluetoothManager[10] slouží k získání informací o připojených zařízení. Bluetooth zařízení může být připojeno více a můžeme zjistit jejich adresu, název a další informace. Pro získání okolních viditelných zařízení je potřeba využít BluetoothAdapter a samotné vyhledávání může trvat až 12 sekund [11]. Výsledek hledání nemá podobu listu, jako to je u připojených zařízeních a u WiFi, ale musí být registrovaný BroadcastReciever, pomocí kterého se zpracují informace o každém objeveném zařízení v průběhu asynchronního hledání.

3.2.3 SQLite databáze s greenDAO

Ukládání dat je ve frameworku řešeno pomocí SQLite databáze přímo v daném zařízení. Pro práci s databází je využito ORM greenDAO, které značně zjednodušuje vytvoření databáze, dotazování a následné mapování dat na Java objekty. Samotnou databázi a DAO vrstvu vytvoří greenDAO automaticky z nadefinovaného modelu. Vývojář tedy musí pouze napsat Java

entity s patřičnými anotacemi a greenDAO potřebné třídy vygeneruje. Pokud je později potřeba model změnit, stačí opět jen upravit entity a ostatní třídy nechat vygenerovat znovu.

GreenDAO bylo vybráno také z důvodu rychlosti a bezpečnosti. Zapisování a aktualizování dat probíhá zhruba 2x rychleji, než u konkurenčního OrmLite a čtení dat provádí přibližně 4x rychleji, jak je znázorněno na obrázku 3.1. Z dat ukládaných frameworkem se dá pomocí seznamu WiFi sítí zjistit přibližná poloha v čase skenování. Tato data jsou tedy pro uživatele velice citlivá a je kladen důraz na jejich bezpečnost. Z toho důvodu je pomocí GreenDAO využita nadstavba SQLCipher[12]. Tato nadstavba nad klasickým SQLite poskytuje 256-bitové AES šifrování databázových souborů. Pokud by se tedy případný útočník dostal až k databázovým souborům, citlivá uživatelská data budou chráněna šifrováním.



Obrázek 3.1: Porovnání výkonu greenDAO s dalšími ORM[13].

3.3 Rekurzivní získání kontextu

Další možnou funkcionalitou frameworku může být rekurzivní získávání okolního kontextu od ostatních zařízení. Framework by tak načetl nejen okolní kontext daného zařízení, ale zeptal by se ostatních telefonů v okolí na jejich okolní kontext. Tímto způsobem může být vytvořena komplexní představa o celém okolním kontextu. Systém Android ale nativně nepodporuje získání takových dat od okolních zařízení. Ostatní zařízení by musela sama vysílat a poskytovat tato data, například přes NearbyConnections[14]. Pokud by se framework nacházel i na okolních zařízeních, bylo by sdílení kontextů také možné.

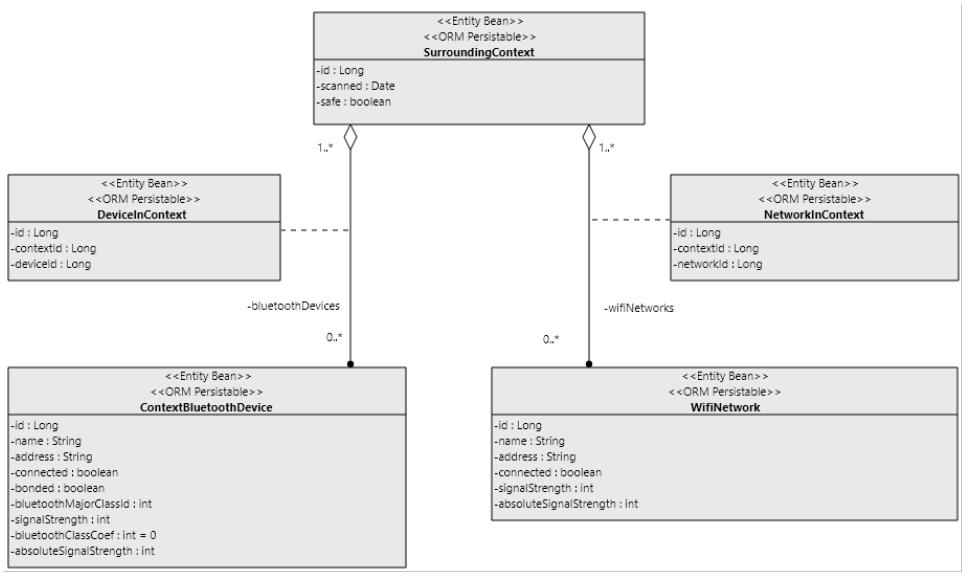
Kapitola 4

Návrh

Framework se skládá ze tří hlavních částí. Načtení okolního kontextu, výpočet úrovně bezpečnosti a označování nových bezpečných kontextů. Samotný kontext je poté složen z WiFi sítí a Bluetooth zařízení v okolí. Tyto dvě množiny zařízení se následně porovnají s uloženými bezpečnými kontexty. Návrh frameworku a rozdělení do jednotlivých částí je popsáno v následujících sekcích.

4.1 Datový model

Data, se kterými framework pracuje, se skládají z Bluetooth zařízení a WiFi sítí, které poté tvoří okolní kontext. Třída SurroundingContext představuje okolní kontext naskenovaný v určitém čase. Takový kontext má seznamy naskenovaných okolních sítí a zařízení a může být označen jako bezpečný. U obou skenovaných entit jsou evidovány informace o jejich MAC adrese, názvu, stavu připojení s síle signálu. Bluetooth zařízení navíc obsahuje také typ zařízení a zda je spárované. Konceptuální model těchto tříd je znázorněn na obrázku 4.1.



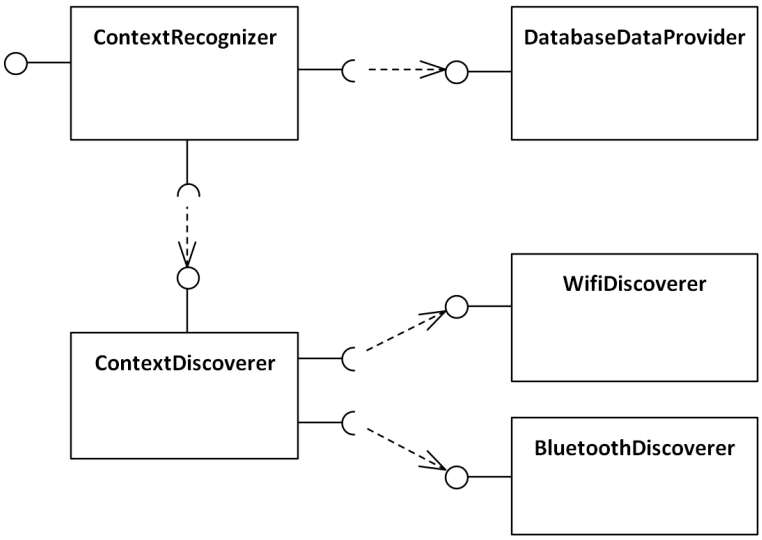
Obrázek 4.1: Konceptuální model tříd.

4.2 Komponenty

WifiDiscoverer a BluetoothDiscoverer jsou implementace rozhraní IWifiDiscoverer a IBluetoothDiscoverer. Tyto třídy se starají o samotné skenování WiFi sítí a Bluetooth zařízení z okolí a využívají k tomu služby poskytované systémem Android. ContextDiscoverer z načtených seznamů vytvoří objekt třídy SurroundingContext, reprezentující okolní prostředí. Třída ContextRecognizer aktuální kontext porovná s referenčními kontexty uloženými v databázi a určí jeho bezpečnost. ContextRecognizer se dále stará o označování nových bezpečných kontextů a udržuje je aktuální.

4.3 Návrhové vzory

Ve frameworku jsou využity návrhové vzory Observer a Singleton, které zaručují celkovou jednotnost frameworku a ulehčují zpětnou komunikaci mezi objekty.



Obrázek 4.2: Diagram komponent.

4.3.1 Singleton

Návrhový vzor Singleton zaručuje, že v celém běhu aplikace je pouze jedna instance dané třídy. Třída se sama stará o vytvoření jejího objektu a jeho následné poskytování. Singleton je zde využit, protože při určování bezpečnosti okolí cheme, aby se výpočet prováděl jednotně pro celou aplikaci. U tříd starajících se o skenování okolního kontextu zaručuje, aby se při vytvoření více instancí jednotlivých tříd neskenovalo stejné okolí vícekrát. Zařízení se vždy nachází pouze v jednom okolním kontextu a není tedy nutné skenovat ho ve více instancích. Zjednoduší se tak i využití frameworku v různých částech aplikace, kde jedna část může zahájit skenování okolí a druhá bude požadovat určení bezpečnostní úrovně tohoto okolí.

4.3.2 Observer

Návrhový vzor Observer použijeme, pokud chceme, aby nějaké objekty byly informovány o změně stavu jiného objektu. Zároveň ale potřebujeme za běhu určovat, které objekty budou informovány. Observer je využit ve třídě Context-Discoverer při skenování okolí. V případě, že se s bezpečností kontextu pracuje na více místech v aplikaci, je potřeba, aby všechny tyto komponenty pracovaly s aktuálním okolním kontextem. ContextDiscoverer v tomto případě v roli pozorovaného (observable) a jako pozorovatele přijímá implementace třídy DiscovererObserver. Jednotlivé části aplikace se zaregistrují jako pozorovatelé a v momentě, kdy je celý kontext naskenovaný, jsou o tom upozorněny.

4.4 Případy užití

Framework bude kromě hlavní funkce určování bezpečnosti okolí poskytovat i další vedlejší funkcionality. Aplikace si bude moct od frameworku vyžádat například jen seznam okolních zařízení, a to jak kompletní, tak pro jednotlivé typy (Bluetooth a WiFi). Aplikace bude dále moct pomocí frameworku sama definovat referenční bezpečné kontexty. V následující části jsou jednotlivé případy užití podrobněji popsány.

Obrázek 4.3: Případy užití.

4.4.1 Načíst okolní WiFi

Framework bude umožňovat načtení WiFi sítí v okolí a sítě, ke které je zařízení aktuálně připojené. Seznam těchto sítí framework vrátí v listu obsahujícím všechny informace o sítích, jako například SSID, MAC adresu přístupového bodu či sílu signálu.

4.4.2 Načíst okolní Bluetooth zařízení

Podobně jako WiFi sítě, bude framework poskytovat informace viditelných Bluetooth zařízeních v okolí a o aktuálně připojených zařízeních.

4.4.3 Načíst okolní kontext

Pro určení okolního kontextu bude framework poskytovat kompletní seznam všech okolních WiFi sítí a Bluetooth zařízení. Tato funkce spojí vyhledávání WiFi sítí a Bluetooth zařízení a poskytne je v podobě objektu, který představuje celkový okolní kontext.

4.4.4 Určit bezpečnost kontextu

S uloženými známými bezpečnými kontexty framework porovná aktuální získaný kontext. Na základě předem definovaných hranic a algoritmu se určí úroveň bezpečnosti okolního kontextu. Tato úroveň je vrácena aplikaci, která si o určení bezpečnosti požádala.

4.4.5 Určit nové bezpečné kontexty

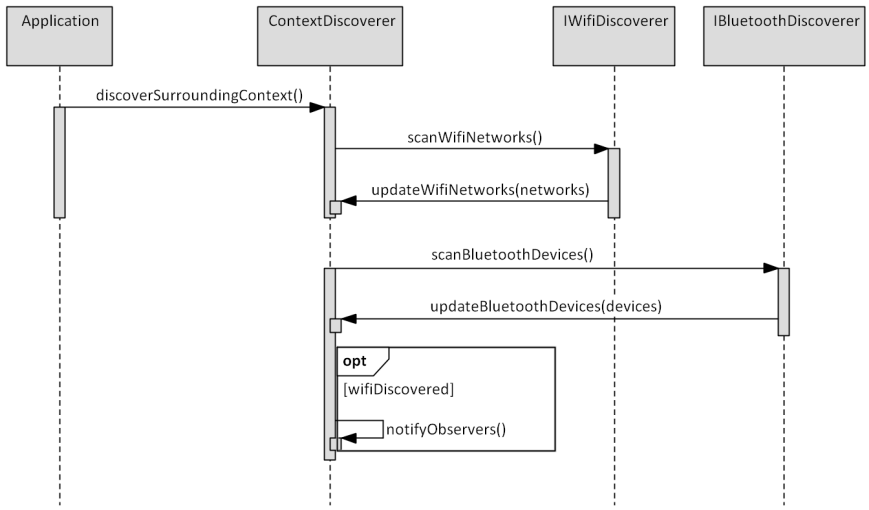
Framework bude umožňovat označit aktuální okolní kontext jako bezpečný. Tímto způsobem budou definované nové referenční kontexty, se kterými se budou další seznamy okolních zařízení porovnávat za účelem určení procentuální shody těchto kontextů. Po každém načtení okolního kontextu bude framework umožňovat porovnat jej s uloženými kontexty, které byly načteny v minulosti. Pokud bude mít aktuální kontext vysokou shodu s větším množstvím uložených kontextů, bude z těchto kontextů vytvořen případný nový bezpečný kontext. Aplikace využívající framework pak bude moct potvrdit označení nového bezpečného kontextu.

4.5 Sekvenční diagramy

Aplikace využívající tento framework pracuje převážně s jeho třemi hlavními funkcemi. Jejich průběh celým frameworkem je znázorněn a popsán v následujících diagramech.

4.5.1 Načítání kontextu

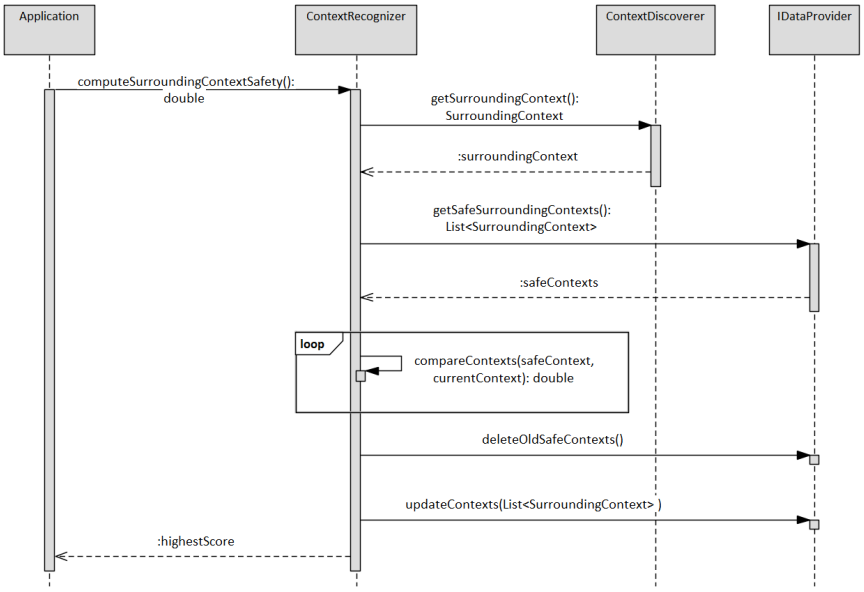
Aplikace zahájí pomocí třídy ContextDiscoverer skenování okolí. Přes WifiDiscoverer a BluetoothDiscoverer se začnou skenovat jednotlivá zařízení. Jakmile jsou oba typy okolí naskenovány, vytvoří se nový objekt typu SurroundingContext a případní posluchači jsou o tomto stavu informováni.



Obrázek 4.4: Sekvenční diagram načtení okolního kontextu.

4.5.2 Určení bezpečnosti okolí

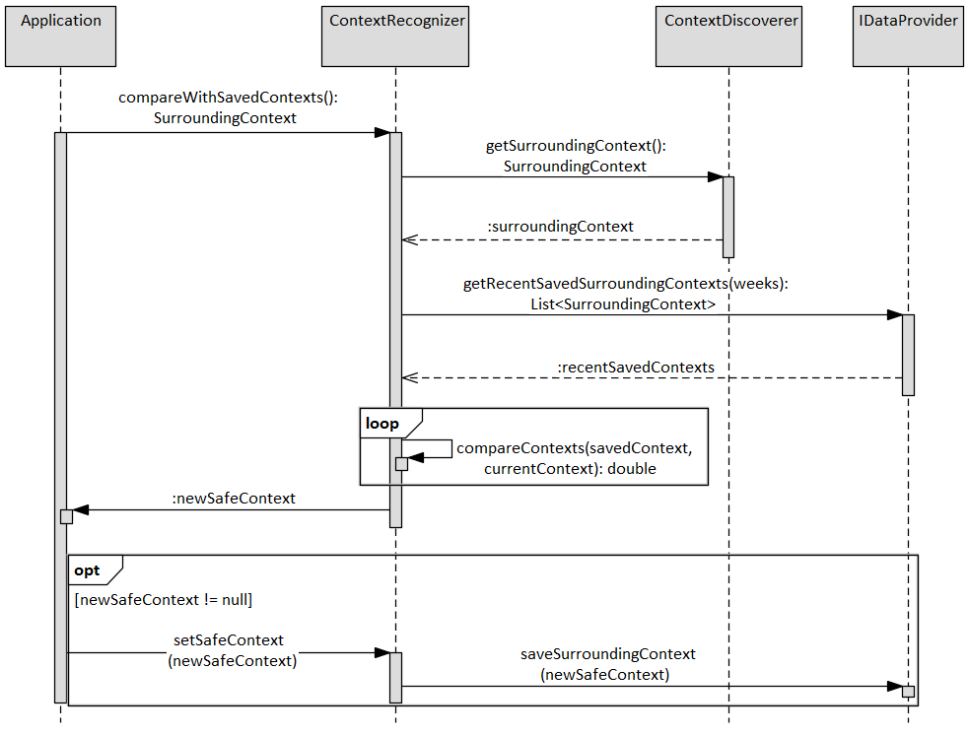
Po načtení okolí pošle aplikace třídě ContextRecognizer dotaz na bezpečnost okolí. ContextRecognizer aktuální kontext porovná se známými bezpečnými kontexty a pro každý určí jejich shodu, tedy bezpečnost aktuálního kontextu. Největší procentuální shoda je poté vrácena aplikaci, která si sama rozhodne, jak s touto informací naloží.



Obrázek 4.5: Sekvenční diagram určení bezpečnosti okolí.

4.5.3 Označení nového bezpečného kontextu

Po načtení okolí se daný kontext porovná kontexty naskenovanými v nedávné době. Pokud se více kontextů shoduje, z porovnání vyjde kandidát na nový bezpečný kontext. Pokud je takový kandidát aplikací potvrzen, uloží se do databáze.



Obrázek 4.6: Sekvenční diagram označení nového bezpečného kontextu.

Kapitola 5

Implementace

Pro implementaci byl byl použit programovací jazyk Java ve verzi 8. Framework je vytvořen převážně pro zařízení s operačním systémem Android.

5.1 Skenování okolí

Okolní kontext se skládá ze dvou druhů zařízení, která framework skenují. Jsou to WiFi sítě a Bluetooth zařízení v okolí. O samotné skenování se starají implementace rozhraní IWifiDiscoverer a IBluetoothDiscoverer, které využívají návrhový vzor Singleton.

5.1.1 Skenování WiFi

Třída WifiDiscoverer načítá informace o aktuálně připojené WiFi síti a ostatních sítích v okolí. K tomu využívá Android třídu WifiManager a pomocí metod getConnectionInfo() a getScanResults() zjistí potřebné informace o aktuálně připojené WiFi síti a výsledky z posledně provedeného skenování přístupových bodů. Ručně vyvolat skenování WiFi sítí bylo dříve možné zavolání metody startScan(), ale od API verze 28 je označena za zastaralé a v budoucnu nebude možné ručně spouštět skenování[15]. Získané informace jsou převedeny na list objektů WifiNetwork a třídě ContextDiscoverer je poslán aktuální seznam okolních sítí.

5.1.2 Skenování Bluetooth zařízení

Skenování Bluetooth zařízení je složitější než u WiFi sítí. Metoda getConnectedDevices(profile) třídy BluetoothManager zde použít nelze, protože ta podporuje pouze profily GATT a GATT_SERVER, přes které se ale nenačtou například bezdrátové reproduktory. Pro získání potřebných informací o připojených zařízeních je tedy využita třída BluetoohAdapter a metoda getProfileProxy(context, listener, profile). V posluchači, předávanému této metodě jako parametr, získáme všechna zařízení daného profilu, která jsou ve stavu STATE_CONNECTED. Na těchto zařízeních se poté naváže připojení ke GATT serveru daného Bluetooth zařízení a přes callback je získána síla signálu. Při testování se však občas stávalo, že se nepodařilo získat sílu signálu a u některých Bluetooth zařízení nedošlo k získání síly signálu nikdy. Převodník na absolutní sílu signálu byl tedy nastaven tak, že pokud nedošlo ke správnému získání útlumu v dB, byla síla signálu nastavena na 7 z celkové 10ti stupňové škály. K načtení okolních nepřipojených Bluetooth zařízení slouží metoda startDiscovery(). Tím začne vyhledávání okolních zařízení, které však může trvat zhruba 12 sekund[11]. Pro získání objevených zařízení je nutné registrovat BroadcastReceiver, který je navěšen na akce objevení nového zařízení a ukončení vyhledávání. U načítání nepřipojených zařízení je zjištění síly signálu spolehlivé. Po ukončení vyhledávání je třídě ContextDiscoverer poslán seznam všech objevených zařízení a BroadcastReceiver je odregistrován.

5.2 Určení bezpečnosti okolí

Bezpečnost aktuálního okolí se určuje pomocí procentuální shody s již známými bezpečnými kontexty. Princip algoritmu spočívá v tom, že se informace o seznamu n zařízení převedou na n-rozměrný vektor a následně je spočítán skalární součin obou vektorů. Může se stát, že více komponent aplikace využívající framework bude chtít určit bezpečí okolí. Jelikož okolí je načteno pouze jednou, došlo by tak k nechtěným duplicitním výpočtům. Po provedení výpočtu si výsledné skóre bezpečnosti framework zapamatuje a při dalších dotazech na bezpečnost stejného okolí nemusí provádět výpočet znovu.

5.2.1 Převody na číselnou hodnotu

Jelikož se shoda dvou kontextů počítá pomocí šíselných vektorů, je nutné jednotlivá zařízení převést na jejich číselné hodnoty. Listy tříd WifiNetwork a ContextBluetoothDevice jsou převedeny na listy třídy MatrixElement. Tato třída obsahuje MAC adresu zařízení, jeho hodnotu v referenčním kontextu a hodnotu v aktuálním kontextu. Tyto dvě hodnoty se následně využijí ve výpočtu. Kromě výpočtu samotné hodnoty zařízení, je postup převodu na list objektů MatrixElement u obou typu zařízení stejný. Využijeme zde tedy generiky a oba typy zařízení bude převádět jedna metoda. Pomocí generiky se dají vytvářet parametrizované metody. Takové metodě až při jejím volání předáme parametr označující třídu, se kterou má pracovat. WifiNetwork a ContextBluetoothDevice implementují rozhraní IContextDevice. Obě třídy tak mají metodu computeDeviceValue(), která spočítá hodnotu konkrétního zařízení. Metoda pro výpočet shody jednotlivých listů tedy přijímá jako parametr list generických objektů T implementujících rozhraní IContextDevice. Výpočet se u různých typů zařízení liší také přístupem k nově připojené síti, respektive nově objevenému Bluetooth zařízení. Proto se uvnitř metody kontroluje konkrétní třída generického parametru a na základě toho jsou provedeny příslušné kroky výpočtu.

Wifi

U WiFi sítí je hodnota zařízení z referenčního kontextu určena pouze násobkem síly signálu a koeficientu stavu připojení. U aktuálního kontextu je výpočet stejný. Pokud je však připojená síť jiná, než je v referenčním kontextu, je tato hodnota u připojené sítě vynásobena ještě koeficientem změny připojené WiFi.

```
//referencni kontext
    for sit in referencniSite
        hodnota = silaSignalu * stavPripojeni

//aktualni kontext
    for sit in aktualniSite
        hodnota = silaSignalu * stavPripojeni
        if sit je pripojena a v referencnim nebyla pripojena
            hodnota = ZMENA_SITE
```

Kód 5.1: Pseudokód výpočtu číselné hodnoty WiFi sítě.

Bluetooth

Bluetooth obsahuje navíc typ zařízení a informaci, zda je spárovaný. Číselná hodnota se tedy skládá ze síly signálu, stavu připojení a typu zařízení. Pokud zařízení není připojené, ale je spárované, je hodnota násobena navíc koeficientem spárovaného zařízení. Pokud je v aktuálním kontextu zařízení, které v referenčním nebylo, násobí se jeho hodnota ještě koeficientem nového zařízení v okolí. Jestliže totiž v okolí přibylo zařízení, může se jednat o dalšího člověka v blízkém okolí a tudíž to znamená větší bezpečností riziko, než když naopak zařízení oproti referenčnímu kontextu ubude.

```
//referencni kontext
    for zarizeni in referencniZarizeni
        hodnota = silaSignalu * stavPripojeni * typZarizeni
        if zarizeni neni pripojene a je sparovane
            hodnota *= SPAROVANE

//aktualni kontext
    for zarizeni in aktualniZarizeni
        hodnota = silaSignalu * stavPripojeni * typZarizeni
        if zarizeni neni pripojene a je sparovane
            hodnota *= SPAROVANE
        if zarizeni nebylo v referencnim kontextu
            hodnota *= NOVE_ZARIZENI
```

Kód 5.2: Pseudokód výpočtu číselné hodnoty Bluetooth zařízení.

Android rozlišuje celkem 11 hlavních typů Bluetooth zařízení[16]. Různé typy zařízení mohou znamenat různé bezpečnostní riziko a tedy i různě ovlivňují výslednou hodnotu zařízení a celkové skóre. Například PHONE nebo WEARABLE znamenají, že v okolí bude nejspíš další osoba. Naopak jestliže je zařízení typu TOY, velké riziko to nepředstavuje. V tabulce 5.1 jsou všechny hlavní třídy Bluetooth zařízení převedeny na číselnou hodnotu. Při skenování může být objeveno zařízení, u kterého bude typ neznámý. V takovém případě je číselná hodnota typu zařízení nastavena automaticky na úroveň 5 z celkových 10.

Typ zařízení	Hodnota
Phone	10
Wereable	10
Computer	8
Audio video	7
Health	5
Uncategorized	5
Toy	4
Imaging	3
Misc	2
Networking	2
Peripheral	2

Tabulka 5.1: Číselné hodnoty typů Bluetooth zařízení.

Úroveň signálu je při skenování označena jako RSSI a je reprezentována v jednotkách dBm. Pro výpočet hodnoty zařízení je tento útlum převeden na číslo v rozmezí od 1 do 10 včetně. Tedy nejvzdálenější zařízení má vysoký útlum signálu a jeho absolutní úroveň bude rovna 1, zatímco zařízení ve velmi těsné blízkosti s minimálním útlumem má úroveň 10.

5.2.2 Porovnání okolí

Dohromady číselné hodnoty zařízení v jednom okolí představují vektor. Vypočítáním skalárního součinu dvou vektorů získáme míru jejich shody. Pomocí následujícího vzorce, kde **u** je aktuální kontext a **v** je referenční, spočteme $\cos\alpha$. Úhel α se představuje úhel mezi vektory aktuálního a referenčního kontextu.

$$\cos\alpha = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}|\cdot|\vec{v}|} = \frac{u_1v_1 + u_2v_2 + \cdots + u_nv_n}{\sqrt{u_1^2+u_2^2+\cdots+u_n^2}\cdot\sqrt{v_1^2+v_2^2+\cdots+v_n^2}}$$

Obrázek 5.1: Vzorec pro výpočet úhlu mezi dvěma vektory.

Cosinus v tomto případě tedy znázorňuje shodu mezi těmito vektory. Pokud je úhel mezi vektory 90° a tedy $\cos\alpha$ je roven 0, množiny zařízení mají 0% shodu, což znamená, že nemají žádné společné zařízení. Vektory s úhlem 0° a $\cos\alpha$ roven 1 mají 100% shodu. Tedy obsahují stejná zařízení se stejnými vlastnostmi. Porovnání vektorů se u WiFi i Bluetooth provádí stejným způsobem.

5.2.3 Celková shoda okolí

Z porovnání množin WiFi i Bluetooth zařízení vyjdou číselné hodnoty od 0 do 1, představující shodu daných množin. Pro určení celkové shody, musíme tyto čísla sečíst. Předpokládáme, že neznámá Bluetooth zařízení, a tedy

možné další osoby v okolí, znamenají větší bezpečnostní riziko, než WiFi sítě v okolí. Například pokud bude ve stejné místnosti osoba s telefonem a chytrými hodinkami, tedy dvě nové Bluetooth zařízení, je to mnohem větší bezpečnostní riziko, než když se ve vedlejším bytě objeví nová WiFi síť. Z těchto důvodů je váhový poměr mezi skóre Bluetooth a WiFi roven 2:1. Tedy pokud shoda WiFi sítí bude 100% a Bluetooth zařízení 50%, výsledná shoda kontextů bude 66,67%.

5.2.4 Hodnoty koeficientů

U WiFi sítí má vyšší váhu síť, ke které je zařízení připojené. V běžných prostředích se obvykle vyskytuje větší množství WiFi sítí, takže okolní kontext je identifikován spíše připojenou sítí, než dalšími sítěmi v okolí. Pokud je aktuálně připojená WiFi jiná než byla v referenčním kontextu, znamená to významnou změnu kontextu. Na tuto změnu je tím pádem kladen větší důraz a hodnota nové připojené WiFi je vynásobena. Naopak u Bluetooth zařízení je kladen větší důraz na neznámá zařízení v okolí, která mohou znamenat další osoby a tedy bezpečnostní riziko. Z toho důvodu má nepřipojený Bluetooth vyšší koeficient. Pokud je však nepřipojený Bluetooth spárovaný, tedy je to známé a v minulosti připojené zařízení, koeficient výslednou hodnotu sníží. Jestliže se v aktuálním kontextu nachází zařízení, které v referenčním nebylo, může to znamenat nové osoby v blízkém okolí, a tím pádem má toto zařízení vyšší váhu.

Konstanta	Hodnota
Připojená WiFi	2
Nepřipojená WiFi	1
Připojené Bluetooth zařízení	1
Nepřipojené Bluetooth zařízení	1,5
Změna připojené WiFi	1,5
Nové Bluetooth zařízení v okolí	1,5
Spárované Bluetooth zařízení	0,5

Tabulka 5.2: Hodnoty konstant použitých při výpočtu.

5.3 Ukládání kontextů

Při každém vypočítání bezpečnosti okolního kontextu se aktuální okolí uloží do databáze. Ze všech uložených kontextů se později mohou vypočítat nové referenční kontexty.

5.3.1 Určení referenčního kontextu

Určování nových bezpečných kontextů může probíhat dvěma způsoby. Aplikace může ručně frameworku říct, že chce aktuální kontext označit jako bezpečný,

nebo se bezpečné kontexty vytváří automaticky samy. Aktuálně načtený kontext se porovná s kontexty v nedávné době, kromě těch, které jsou označeny jako bezpečné. Pro porovnávání těchto kontextů je použit stejný algoritmus, ale s tím rozdílem, že koeficienty pro změnu zařízení jsou nastaveny na 1. Tím pádem se porovnávají pouze přítomnosti a vlastnosti zařízení v kontextu, ale jejich změna připojení nemá vliv. Pokud je shoda aktuálního kontextu s uloženým kontextem alespoň 85%, uložený kontext je přidán do listu shodných kontextů. Pokud je takových shodných kontextů více, zkontroluje se, zda byly naskenovány ve více než 13 různých dnech za období 13 týdnů. To představuje buď průměrně jeden výskyt týdně, nebo velmi častý výskyt v okolí za krátké období. Jestliže se tedy člověk vyskytuje v jednom prostředí každý týden minimálně jednou a používá v něm aplikaci s tímto frameworkem, může to být považováno za známé a bezpečné prostředí.
Z taktových seznamů zařízení je poté vytvořen nový referenční bezpečný kontext. Před samotným označením nového kontextu jako bezpečný musí tuto akci potvrdit samotná aplikace, potažmo přímo uživatel.

5.3.2 Zánik bezpečného kontextu

Aby se seznam bezpečných kontextů udržoval aktuální, framework se stará také o to, že dlouho nenavštívené bezpečné kontexty maže. Při každém vytvoření nového bezpečného kontextu je mu nastavena výchozí aktuálnost. Tato číselná hodnota se poté při každém výpočtu bezpečnosti u všech bezpečných kontextu snižuje. U referenčního kontextu, u kterého došlo k nejvyšší shodě, je naopak tato hodnota zvýšena násobkem této číselné shody. Pokud tedy nejvyšší shoda byla 75%, je u tohoto bezpečného kontextu zvýšena jeho aktuálnost o 7,5. Po každém výpočtu bezpečnosti jsou z databáze odstraněny všechny bezpečné kontexty, kterým tato hodnota aktuálnosti klesla na nulu. Tímto se zaručí, že referenční kontexty, u kterých nedošlo delší dobou k nejvyšší shodě, budou z bezpečných kontextů odstraněny. Naopak jestliže má určitý bezpečný kontext při porovnávání často nejvyšší shodu, je jeho aktuálnost zvyšována a zůstává tak relevantní.

5.4 Instalace frameworku

Minimální požadavky frameworku:

- Android SDK vrze 24
- Vývojové prostředí podporující Gradle

Framework je poskytován jako Android knihovna v souboru s příponou .aar. Vývojář naimportuje tuto knihovnu do svého projektu, připojí další potřebné knihovny a může začít framework používat.
Postup instalace frameworku v Android Studiu.

1. V projektu kliknout pravým tlačítkem na aplikaci > New > Module.

2. Vybrat Import .JAR/.AAR Package.
3. Ve File name vybrat cestu ke knihovně a klinout na Finish.
4. Do build.gradle na úrovni aplikace přidat řádky
   - apply plugin: 'org.greenrobot.greendao'
   - implementation 'org.greenrobot:greendao:3.2.2'
   - implementation 'net.zetetic:android-database-sqlcipher:4.1.3'
   - implementation project(':surroundingcontextlibrary')
5. Do build.gradle na úrovni projektu přidat závislost *classpath 'org.greenrobot:greendao-gradle-plugin:3.2.2'*.

Při spuštění aplikace je nutné framework inicializovat zavoláním metody ContextRecognizerModule.init(Context context, String password).

Kapitola 6

Testovací aplikace

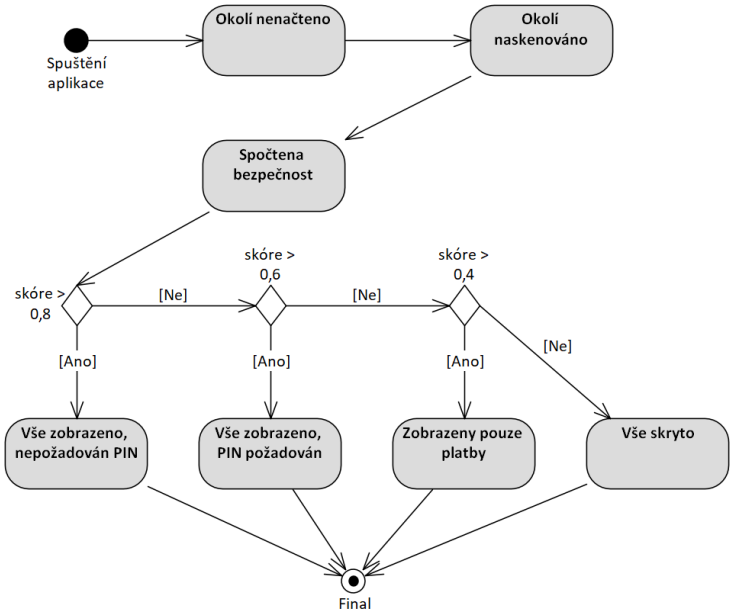
V rámci práce byla vytvořena také ukázková aplikace využívající tento framework. Aplikace simuluje některé funkce mobilního bankovnictví, tedy jednoho z možných případů využití frameworku. Data z aplikace nejsou nijak zpracována ani nikam odesílána, jedná se pouze o demonstraci frameworku. Při spuštění aplikace je, stejně jako u klasického mobilního bankovnictví, uživatel povinen zadat PIN. Po správném zadání PIN kódu je uživateli zobrazena obrazovka s přehledem o fiktivním účtu. Je zde vidět zůstatek na účtu a několik nedávných plateb. Kliknutím na tlačítko v levém dolním rohu se uživatel přesune na obrazovku pro zadání nové platby. Po vyplnění povinných údajů se zobrazí shrnutí platby. Potvrzením údajů o platbě a zadáním PINu se platba potvrdí. Nějak takto funguje běžné mobilní bankovnictví bez využití frameworku pro určení bezpečnosti okolí. Nyní si popíšeme, jak je zde framework využitý.

6.1 Skenování okolí

Vzhledem k tomu, že skenování Bluetooth zařízení trvá několik sekund, skenování je zahájeno ihned po spuštění aplikace. Při spuštění aktivity pro zadání PINu je na třídě ContextDiscoverer zavolána metoda discoverSurroundingContext(), čímž se na pozadí spustí skenování okolí. Při vytvoření obrazovky s přehledem o účtu je na ContextDiscoverer navěšen pozorovatel, který bude upozorněn, jakmile bude okolí naskenované.

6.2 Vyhodnocení bezpečnosti okolí

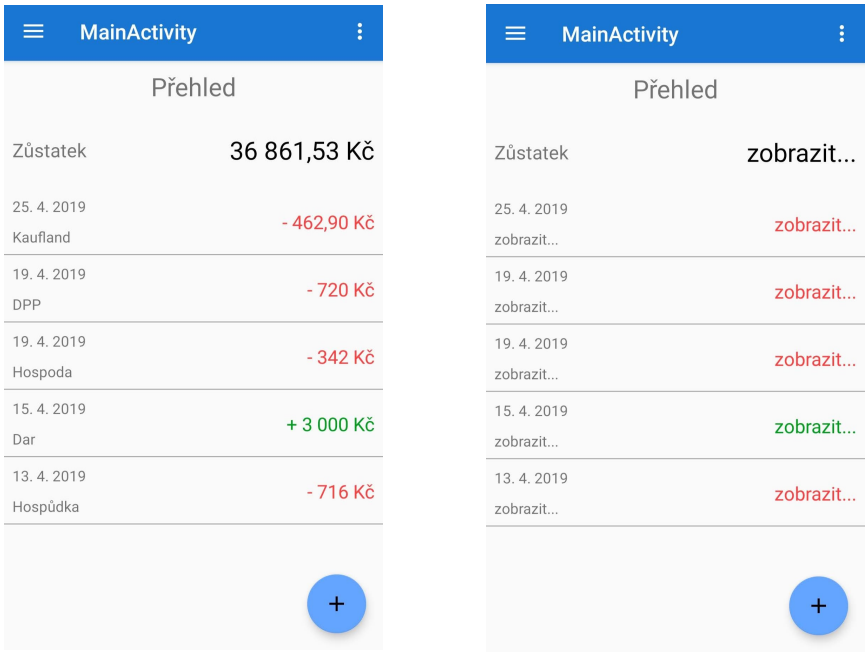
Jestliže při zobrazení přehledu o účtu není kontext ještě naskenovaný, framework vrátí hodnotu bezpečnosti -1. Nenačtený kontext je považován za nebezpečný a jsou tedy aplikována všechna bezpečnostní opatření. Jakmile je kontext načtený, je o tom informovaný pozorovatel, který pošle požadavek na určení bezpečnosti okolí. S určenou úrovní bezpečnosti kontextu dále aplikace pracuje reaguje na ní podle aktuální situace. Stavy aplikace v závislosti na určené bezpečnosti okolního kontextu jsou znázorněny na obrázku 6.1.



Obrázek 6.1: Stavový diagram ukázkové aplikace.

6.2.1 Přehled o účtu

Na obrazovce s přehledem o účtu jsou citlivá data jako zůstatek a detaily nedávných plateb. Pokud se uživatel nachází například v MHD nebo na jiném veřejném místě, lidé blízko něho mohou vidět informace zobrazené na displeji telefonu. Uživatel, který dbá na své soukromí nechce, aby ostatní lidé věděli, kolik má peněz na účtu a co si v poslední době koupil. Jestliže je tedy okolí vyhodnoceno jako nebezpečné, jsou tyto informace skryty a mohou být zobrazeny klepnutím. Detaily o nedávných platbách jsou skryty, pokud je bezpečnost okolí menší než 70% a zůstatek na účtu je skryt, pokud je hodnota menší než 40%. Rozdíl mezi známým a neznámým prostředím je zobrazen na obrázku 6.2. Pokud se uživatel přihlásí dřív než se stihne načíst kontext, jsou všechna citlivá data skryta a po načtení kontextu se případně automaticky zobrazí.



(a) : Bezpečné prostředí. **(b) :** Neznámé prostředí.

Obrázek 6.2: Obrazovka s přehledem o zůstatku a nedávných platbách.

6.2.2 Provedení platby

Po vyplnění potřebných údajů ve formuláři se uživatelovi zobrazí obrazovka se souhrnem nové platby a tlačítkem pro její potvrzení. Při potvrzení platby v mobilním bankovnictví je po uživateli běžně požadován opět PIN. Když ale aplikace ví, že se uživatel nachází ve známém a velmi bezpečném prostředí, nemusí po něm požadovat PIN znovu, když ho zadával při spuštění aplikace. V ukázkové aplikaci je po potvrzení platby zjištěna úroveň bezpečnosti aktuálního kontextu. Jestliže je úroveň větší než 80%, není po uživateli PIN požadován. V bezpečném prostředí může tedy uživatel provádět platby mnohem rychleji a pohodlněji. Zabezpečení aplikace je ale stále zachováno, jelikož při spuštění aplikace je PIN vyžadován vždy a k platbě se tedy dostane pouze uživatel, který PIN zná.

Kapitola 7

Testování

Navržený algoritmus byl ještě před samotnou implementací otestován, zda jeho výsledky odpovídají předpokladům. Pro ověření správnosti implementace algoritmu bylo vytvořeno několik unit testů s různými možnými scénáři. Dále byly provedeny uživatelské testy aby se ověřilo, zda mají uživatelé o tuto funkcionalitu zájem a zda jim snížení zabezpečení aplikace podle bezpečnosti okolního kontextu přijde bezpečné.

7.1 Testování algoritmu

Při návrhu algoritmu byly provedeny testy aby se ověřilo, že výsledné hodnoty odpovídají předpokladům. Testování bylo prováděno pomocí tabulek v Excelu a byly simulovány různé možné scénáře. Testováním se také vyzkoušel vliv hodnoty konstant na výsledné skóre u různých scénářů.
V tabulce 7.1 jsou nasimulované různé situace a demonstrován vliv změny okolí na výslednou shodu s referenčním okolím, které je označeno žlutou barvou. Číselné hodnoty reprezentují sílu signálu pro každé zařízení v rozmezí 0 - 10, kde 0 znamená, že se zařízení v okolí nenachází a 10 značí výskyt v těsné blízkosti. Změny v kontextu oproti kontextu referenčnímu jsou zvýrazněny oranžovou výplní. V situaci č. 1 přibylo připojené audio zařízení a v okolí se objevil telefon s chytrými hodinkami, což znamená, že je v okolí minimálně jeden další člověk. Tato změna oproti referenčnímu kontextu má veliký vliv na výslednou shodu, která je v tomto případě 58%. V situaci č. 2 se uživatel nachází v MHD. Má navíc připojené Bluetooth sluchátka a v okolí se nachází neznámé telefony a sluchátka. Takový kontext je považovaný za neznámý, což odpovídá jeho 15% shodě s referenčním kontextem. Situace číslo 3 a 4 ukazují, že pokud se v okolí objeví nové zařízení, má to větší vliv na celkovou shodu, než když naopak zařízení ubude. Dále je v situacích 5 a 6 vidět rozdíl, mezi připojeným a nepřipojeným zařízením. Nové nepřipojené zařízení má větší vliv na celkovou shodu, než zařízení připojené.
Situace 11 a 12 představují změny WiFi sítí v okolí. V první situaci došlo ke změnám pouze u nepřipojených sítí v okolí. Změnily se síly signálů, přibyly dvě nové sítě a jedna ubyla. Na shodu s referenčním okolím to nemá příliš velký vliv. U situace č. 12 naopak v okolí nedošlo k žádným změnám, ale zařízení je připojeno k nové síti. Taková změna je celkem neobvyklá a má

na procentuální shodu zásadní vliv. V tabulce 7.2 jsou znázorněny číselné hodnoty jednotlivých zařízení z výše popsaných situací. Tyto hodnoty tvoří vektor, který se poté porovnává s vektorem referenčního kontextu. Postup výpočtu hodnot jednotlivých zařízení je popsán v sekci 5.2.

		Připojené Bluetooth		Nepřipojené Bluetooth						
Spárováno				Ne	Ne	Ne	Ano	Ne	Ne	
Typ		Hodinky	Audio	Telefon	Telefon	Počítač	Audio	Hodinky	Audio	
Zařízení		**BT1**	**BT2**	**BT21**	**BT22**	**BT23**	**BT24**	**BT25**	**BT26**	**Shoda**
situace	0	10	0	0	8	9	5	0	0	**100,00%**
	1	10	8	8	6	8	7	7	0	**58,24%**
	2	10	8	8	0	0	0	8	9	**15,05%**
	3	10	0	0	0	9	5	0	0	**81,70%**
	4	10	0	8	8	9	5	0	0	**75,64%**
	5	10	10	0	8	9	5	0	0	**89,28%**
	6	10	0	0	8	9	5	0	10	**79,74%**

		Připojené WiFi		Nepřipojené WiFi						
WiFi síť		**AP1**	**AP2**	**AP21**	**AP22**	**AP23**	**AP24**	**AP25**	**AP26**	**Shoda**
situace	0	8	0	8	6	5	3	0	0	**100,00%**
	11	8	0	7	4	5	0	6	7	**88,05%**
	12	4	10	8	6	5	3	0	0	**40,04%**

Obrázek 7.1: Simulované situace různých okolních prostředí.

Číselné hodnoty zařízení								
BT1	**BT2**	**BT21**	**BT22**	**BT23**	**BT24**	**BT25**	**BT26**	**Vektor**
100	0	0	120	135	26,25	0	0	208,1
100	84	180	90	120	36,75	157,5	0	312,2
100	84	180	0	0	0	180	141,75	319,3
100	0	0	0	135	26,25	0	0	170,0
100	0	180	120	135	26,25	0	0	275,2
100	105	0	120	135	26,25	0	0	233,1
100	0	0	120	135	26,25	0	157,5	261,0

Číselné hodnoty zařízení								
AP1	**AP2**	**AP21**	**AP22**	**AP23**	**AP24**	**AP25**	**AP26**	**Vektor**
16	0	8	6	5	3	0	0	19,7
16	0	7	4	5	0	6	7	20,8
8	30	8	6	5	3	0	0	33,1

Obrázek 7.2: Číselné hodnoty zařízení v simulovaných situacích.

7.2 Unit testy

V této práci byly unit testy použity především k ověření správnosti implementace porovnávacího algoritmu. K vytvoření samotných testů byl použit framework JUnit [17]. V unit testech obecně ověřujeme funkcionalitu pouze konkrétní testované metody a snažíme se eliminovat vliv ostatních částí systému na výsledek testu. V případě testování výpočtu shody kontextů třída ContextRecognizer používá ContextDiscoverer a DatabaseDataProvider. Jejich funkcionalitu ale v tomto případě testovat nechceme, a tak pomocí tzv. mockování vytvoříme pouze jejich kopii, která ale nemá žádnou funkcionalitu.

Samotné mockování je zde provedeno frameworkem Mockito[18]. Někdy se může stát, že testovaná metoda je závislá na datech, které vrací metoda v této namockované třídě. Takové metodě můžeme pomocí Mockita nastavit návratovou hodnotu ručně. V průběhu testu se nespustí kód implementovaný touto metodou, ale rovnou se vrátí předaná hodnota. U mockovaných metod s návratovým typem void řekneme, že metoda nemá dělat nic.
Protože třída ContextRecognizer implementuje návrhový vzor Singleton, má v celé aplikaci pouze jednu instanci. To platí také o testech. Aby se zajistila nezávislost jednotlivých testů, musí být instance vytvořena v každém testu znovu. Singleton umožňuje vytvoření instance, ne ale její zničení. Proto musíme v testech přistoupit na vlastnost reprezentující instanci třídy a nastavit ji na null. V rámci práce byly vytvořeny tyto jednotkové testy:

- Porovnání Bluetooth listů
- Porovnání WiFi listů
- Porovnání celých kontextů
- Výpočet celkového skóre ze skóre Bluetooth a WiFi listů
- Určení bezpečnosti kontextu vůči referenčním kontextům
- Převody typů zařízení

7.3 Uživatelské testování

Cílem uživatelského testování je zjistit, jestli funkcionalitě frameworku běžní uživatelé věří a jestli by jim použití frameworku v mobilním bankovnictví přišlo výhodné. Někteří uživatelé kladou velký důraz na zabezpečení svých aplikací a tudíž by využití frameworku mohli považovat za snížení bezpečnosti celé aplikace. Testování probíhalo na ukázkové aplikaci, která simuluje mobilní bankovnictví.

7.3.1 Popis testování

Testování probíhalo v reálném prostředí ve třech místech a situacích, kde by uživatelé reálně mohli používat mobilní bankovnictví.

- Domácí prostředí bez ostatních lidí
- Domácí prostředí s dalšími lidmi v místnosti
- Veřejné prostranství (MHD, ulice)

V prvním testovacím prostředí se uživatel nacházel ve svém domově a v jeho blízkosti nebyli žádní další lidé, kromě moderátora testu. Toto prostředí se pro každého participanta testu liší. Participant žijící v hustě obydleném bytovém domě kolem sebe bude mít pravděpodobně veliké množství WiFi sítí

a Bluetooth zařízení ve větší vzdálenosti. Naopak člověk žijící v rodinném domě na okraji města bude kolem sebe mít jen velmi malé množství ostatních zařízení. Při druhém testu v blízkém okolí uživatele přibyli další dva lidé, kde každý měl nějaké Bluetooth zařízení (hodinky, sluchátka). Ve třetím testu byl uživatelův okolní kontext zcela neznámý. Před samotným testováním byli uživatelé seznámeni se základní funkcionalitou testovací aplikace a pro testování u všech účastníků byl použit jeden chytrý telefon, na kterém se předem označily referenční bezpečné kontexty.

7.3.2 Seznam úkolů

Účastníci testu měli v každém prostředí splnit tento seznam úkolů.

1. Přihlaste se do mobilního bankovnictví. PIN je 341569.
2. Zkontrolujte zůstatek na svém účtu.
3. Zobrazte si detaily o libovolných platbách.
4. Přejděte na obrazovku pro zadání nové platby.
5. Vyplňte platbu s parametry v tabulce 7.1.
6. Platbu odešlete.

Číslo účtu	8364287319/9167
Částka	3 650 Kč
Datum splatnosti	24.5.2019
Zpráva pro příjemce	Ubytování na dovolené

Tabulka 7.1: Detail platby.

7.3.3 Post-test

Post-test participanti vyplnili až po ukončení všech tří testovacích situací. Pomocí tohoto testu zjistíme, zda by byli uživatelé ochotni podobný framework v aplikacích využívat a jestli mu dostatečně důvěřují. Pro zaznamenání odpovědí byla použita Likertova škála. Tato metoda se používá pro určení míry souhlasu a nesouhlasu s daným tvrzením [19]. Odpovědi a jejich číselné ohodnocení jsou zobrazeny v tabulce 7.2.
Účastníci se měli vyjádřit k tvrzením vždy s ohledem na to, v jaké situaci test probíhal. Tvrzení v post-testu byla následující.

1. Úroveň zabezpečení v bezpečném prostředí mi přišla:
2. Úroveň zabezpečení v prostředí s dalšími lidmi v okolí mi přišla:
3. Úroveň zabezpečení na veřejném prostranství mi přišla:

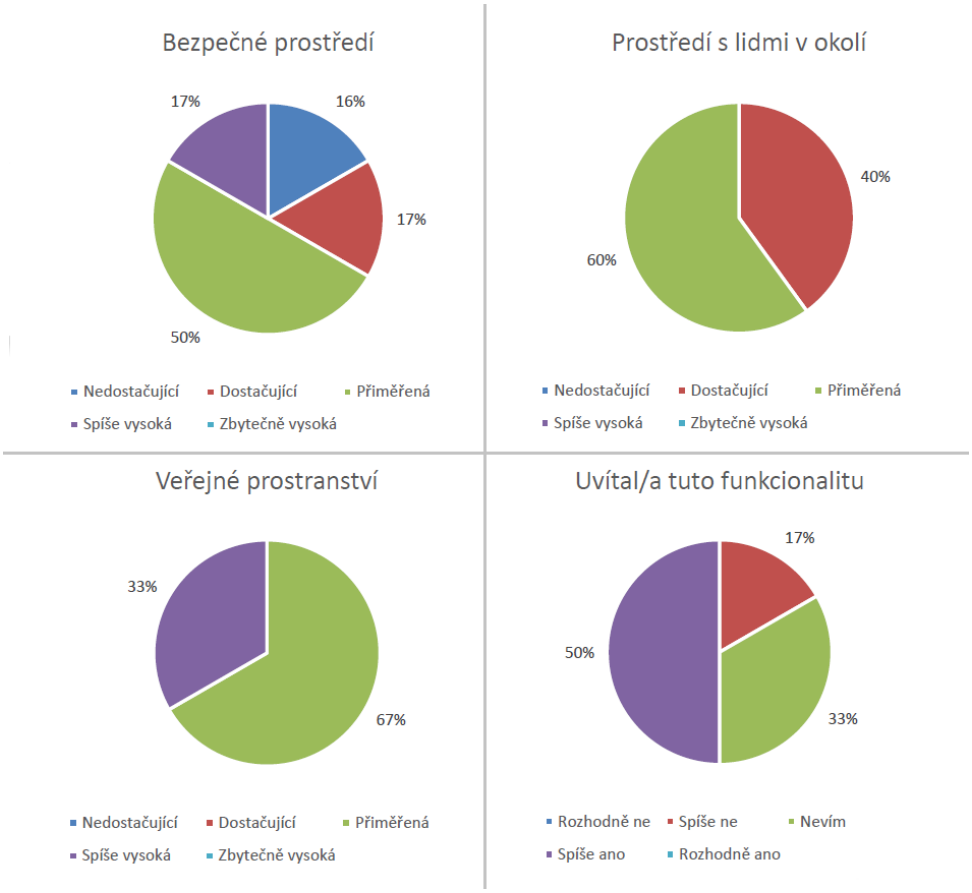
4. Uvítal/a bych využití takové to funkcionality v aplikacích.

Otázka	Odpovědi				
1 - 3 4	Nedostačující Rozhodně ne	Dostačující Spíše ne	Přiměřená Nevím	Spíše vysoká Spíše ano	Zbytečně vysoká Rozhodně ano
Hodnota	-2	-1	0	1	2

Tabulka 7.2: Hodnoty odpovědí post-testu.

7.3.4 Vyhodnocení uživatelského testování

Grafy na obrázku 7.3 představují poměry odpovědí v post-testu. V celkovém souhrnu považovali uživatelé úroveň zabezpečení v různých situacích převážně za odpovídající dané situaci. V některých případech bezpečného prostředí došlo k ne zcela přesnému vyhodnocení bezpečnosti okolí. Tyto nepřesnosti mohou být způsobeny okolím s vysokou hustotou obydlení, jako jsou například bytové domy. V těchto prostředích se průběžně mění velké množství zařízení a jejich síly signálu, což má vliv na celkové vyhodnocení shody s referenčním kontextem. Uživatelé, pro které bylo zabezpečení v některých prostředích příliš slabé uvedli, že ačkoliv bylo vyhodnocení okolního kontextu správné, použití v mobilním bankovnictví jim přišlo příliš nebezpečné. V ostatních typech aplikací, jako jsou například aplikace pro zaznamenávání peněžních výdajů, by tuto funkci naopak uvítali. Dále by uživatelé aplikaci důvěřovali víc, kdyby kromě okolního kontextu vyhodnocovala i další informace určující kontext, například polohu zařízení a jeho pohyb. Taková integrace je však mimo rozsah tohoto frameworku a je na samotných aplikacích, aby tuto funkcionalitu případně implementovali. Z provedených uživatelských testů se dá tedy říct, že funkcionalitu frameworku by uživatelé ocenili a věřili jí, ne však v tak vysoce zabezpečených aplikacích, jako je mobilní bankovnictví.

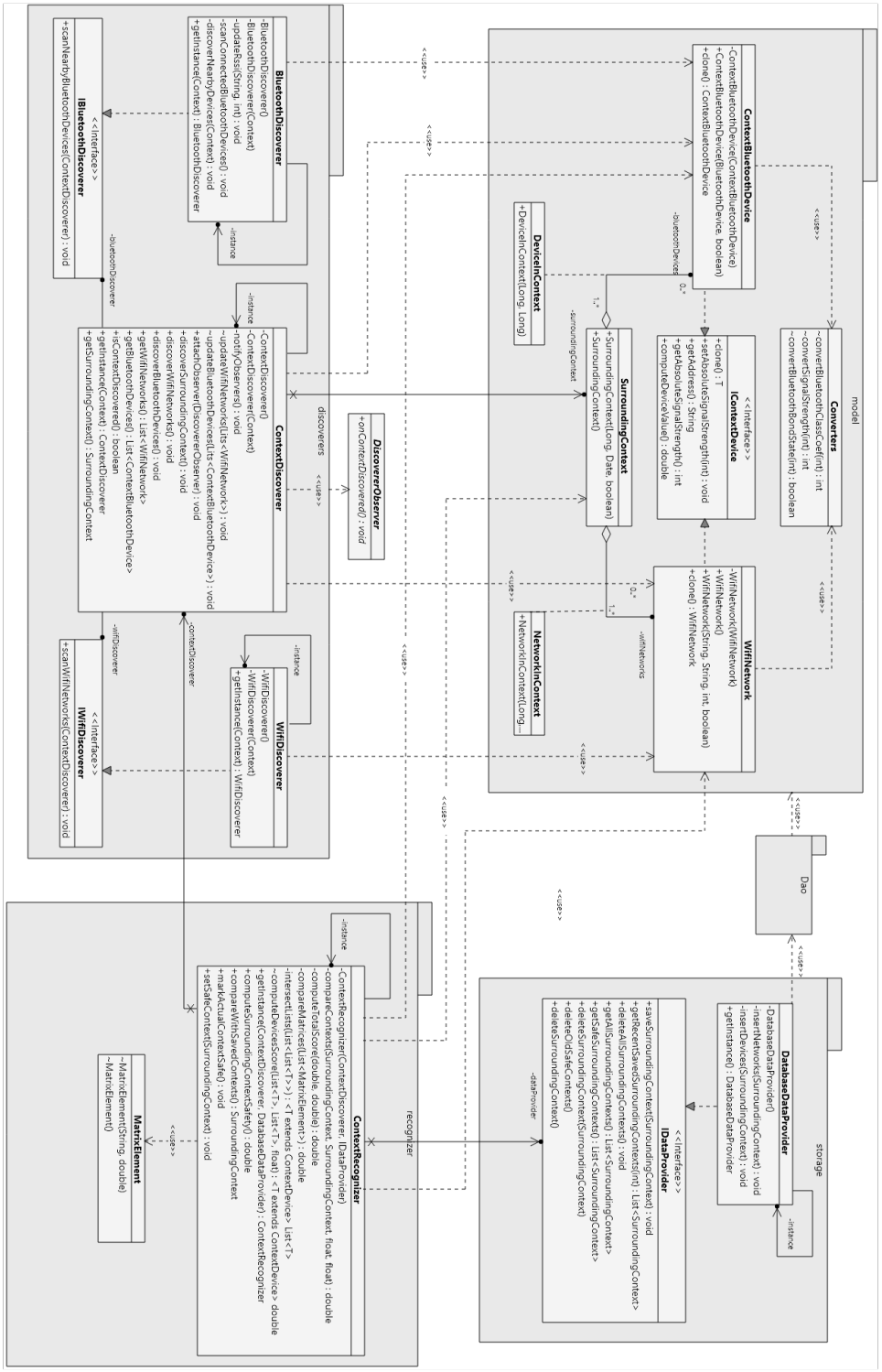


Obrázek 7.3: Poměry odpovědí post-testu uživatelského testování.

Kapitola 8

Závěr

Cílem této práce bylo navrhnout a vytvořit framework, který bude načítat a vyhodnocovat okolní kontext uživatele. Uživatelův okolní kontext se skládá z WiFi sítí a Bluetooth zařízení v okolí. Práce se zabývá třemi hlavními oblastmi. Je to načtení okolního kontextu, jeho porovnání s referenčními kontexty a také určení nových referenčních kontextů. Nejprve byla provedena analýza způsobu načtení zařízení v okolí pro systém Android. Pro získaná data byl navrhnut datový model, který sjednocuje množiny dvou typů zařízení do jednoho objektu, který je označen jako okolní kontext. U každého typu zařízení byly klasifikovány jejich vlastnosti a převedeny na číselnou hodnotu zařízení. Tato hodnota se následně používá při výpočtu shody vektorů. Za účelem porovnání dvou kontextů byl vymyšlen algoritmus, který tyto kontexty porovnává jako vektory. Na základě rozdílu těchto vektorů je určena procentuální shoda dvou okolních kontextů. Tento algoritmus byl v upravené formě použit i pro průběžné určování nových referenčních kontextů, které se vytváří z okolních kontextů, ve kterých se uživatel často vyskytuje. Následně bylo implementováno načítání zařízení v okolí a jejich převod na objekty, se kterými zbylé části frameworku dále pracují. V této části byla objevena nevýhoda načítání Bluetooth zařízení v okolí, a to jeho příliš dlouhá doba trvání. Následně byl implementován zmiňovaný algoritmus pro porovnání dvou okolí a určení nových referenčních kontextů. Správnost implementace algoritmů byla ověřena testy. Testování odhalilo, že pro různé scénáře procentuální shoda dvou kontextů odpovídá předpokladům. V budoucnu by se framework mohl rozšířit i na operační systém iOS. To by umožnilo využití frameworku v aplikacích na obou hlavních platformách mobilních zařízení. V současné podobě framework dokáže načíst okolní kontext pouze pro zařízení, na kterém je spuštěný. Pokud by byl framework i na okolních mobilních zařízeních a byla mezi nimi navázána komunikace, framework by mohl získal informace o mnohem širším okolí.



Obrázek 1: Diagram tříd frameworku.

Příloha A

Seznam ukázek kódu

Příloha B

Použité zkratky

BLE	Bluetooth Low Energy
DAO	Data access object
GATT	Generic Attributes
GPS	Global Positioning System
IoT	Internet of Things
MAC	Media Access Control
ORM	Object-Relational Mapping
OS	Operation system
PIN	personal identification number
RSSI	Internet of Things
SSID	Service Set Identifier

Příloha C

Literatura

[1] *Mobile Banking 2018* [online]. [cit. 2019-05-08]. Dostupné z: https://www.ezonomics.com/ing_international_surveys/mobile-banking-2018

[2] SCHILIT, B., N. ADAMS a R. WANT. Context-Aware Computing Applications: treaties and international agreements registered or filed and recorded with the Secretariat of the United Nations. *1994 First Workshop on Mobile Computing Systems and Applications* [online]. IEEE, 1994, 1994, , 85-90 [cit. 2019-05-18]. DOI: 10.1109/WMCSA.1994.16. ISBN 978-0-7695-3451-0. Dostupné z: http://ieeexplore.ieee.org/document/4624429/

[3] TOMASEK, Matrtin a Tomas CERNY. Automated User Interface Generation Involving Field Classification. *Software Networking* [online]. 2017, **2017**(1), 53-78 [cit. 2019-05-20]. DOI: 10.13052/jsn2445-9739.2017.004. ISSN 2445-9739. Dostupné z: http://www.riverpublishers.com/journal_read_html_article.php?j=JSN/2017/1/004

[4] ZHENG, Mao, Sihan CHENG a Qian XU. Context-Based Mobile User Interface. *Journal of Computer and Communications* [online]. 2016, **04**(09), 1-9 [cit. 2019-05-07]. DOI: 10.4236/jcc.2016.49001. ISSN 2327-5219. Dostupné z: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jcc.2016.49001

[5] *Google Awareness API* [online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/awareness/

[6] *Radar* [online]. [cit. 2019-05-07]. Dostupné z: https://radar.io/

[7] STERLING, Greg. *Facebook-s Places Graph makes 140 million locations available to developers for free* [online]. April 18, 2017 [cit. 2019-05-07]. Dostupné z: https://martechtoday.com/facebook-makes-140-million-places-available-developers-free-197567

[8] Mobile Operating System Market Share Worldwide. *Statcounter* [online]. [cit. 2019-05-07]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide

[9] *WifiManager* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.android.com/reference/android/net/wifi/WifiManager

[10] *BluetoothManager* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.android.com/reference/android/bluetooth/BluetoothManager

[11] *BluetoothAdapter startDiscovery* [online]. In: . [cit. 2019-05-08]. Dostupné z: https://developer.android.com/reference/android/bluetooth/BluetoothAdapter#startDiscovery()

[12] *SQLCipher* [online]. [cit. 2019-05-08]. Dostupné z: https://www.zetetic.net/sqlcipher/about/

[13] *GreenDAO* [online]. [cit. 2019-05-08]. Dostupné z: http://greenrobot.org/greendao/features/

[14] *Nearby Connections* [online]. [cit. 2019-05-08]. Dostupné z: https://developers.google.com/nearby/connections/overview

[15] *WifiManager startScan()* [online]. In: . [cit. 2019-05-08]. Dostupné z: https://developer.android.com/reference/android/net/wifi/WifiManager.html#startScan()

[16] *BluetoothClass.Device.Major* [online]. [cit. 2019-05-08]. Dostupné z: https://developer.android.com/reference/android/bluetooth/BluetoothClass.Device.Major

[17] *JUnit 5* [online]. [cit. 2019-05-08]. Dostupné z: https://junit.org/junit5/

[18] *Mockito* [online]. [cit. 2019-05-08]. Dostupné z: https://site.mockito.org/

[19] ROD, Aleš. Likertovo škálování. *E-LOGOS* [online]. 2012, (13) [cit. 2019-05-10]. ISSN 1211 -0442. Dostupné z: https://nb.vse.cz/kfil/elogos/science/rod12.pdf

Příloha D

Obsah CD

- SurroundigContext - Android projekt obsahující framework v modulu surroundingcontextlibrary a ukázkovou aplikaci v modulu app
- surroundingcontextlibrary.aar - framework v podobě knihovny
- diagramy - adresář s Enterprise Architect projektem obsahujícím UML diagramy a Visual Paradigm diagramem tříd
- LaTeX - zdrojové soubory textové části bakalářksé práce, použité obrázky v plném rozlišení
- LesakJanBP.pdf - elektronická verze bakalářské práce