Mobilní aplikace na sledování a analýzu výdajů během zahraničních cest

Bc. Andrej Kapusta

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Andrej Kapusta**
Osobní číslo:	**A16758**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**prezenční**
Téma práce:	**Mobilní aplikace pro sledování a analýzu výdajů během zahraničních cest**
Téma anglicky:	**A Mobile Application for Tracking and Analysing Spending on Foreign Journeys**

Zásady pro vypracování:

1. **Proveďte rešerši existujících řešení.**
2. **Vypracujte stručný rozbor technologií, které budou použity k návrhu.**
3. **Proveďte rozbor a analýzu požadavků na zvolené řešení.**
4. **Realizujte navrženou aplikaci.**
5. **Navrženou aplikaci vhodným způsobem popište.**
6. **Věnujte pozornost zabezpečení.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MAKAN, Keith a Scott ALEXANDER-BROWN. Android Security Cookbook. 1. Birmingham: Packt Publishing, 2013. ISBN 978-1782167167.**
2. **ELENKOV, Nikolay. Android Security Internals: An In-Depth Guide to Android's Security Architecture. 1. San Francisco: No Starch Press, 2015. ISBN 978-1-59327-581-5.**
3. **WEI, Jason. Android database programming. 1. Birmingham: Pack Publishing, 2012. ISBN 978-1-84951-812-3.**
4. **MEDNIEKS, Zigurd, Laird DORNIN, G. Blake MEIKE a Masumi NAKAMURA. Programming Android. 2. Sebastopol: O'Reilly Media, 2012. ISBN 978-1-449-31664-8.**
5. **SANDOVAL, Jose. RESTful Java Web Services. 1. Birmingham: Pack Publishing, 2012. ISBN 978-1-847196-46-0.**
6. **MATTHEWS, Mark, Jim COLE a Joseph D. GRADECKI. MySQL and Java Developer's Guide. 1. Indianopolis: Wiley Publishing, 2003. ISBN 0-471-26923-9.**

Vedoucí diplomové práce: **Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **30. listopadu 2018**

Termín odevzdání diplomové práce: **17. května 2019**

Ve Zlíně dne 10. prosince 2018

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Mgr. Milan Adámek, Ph.D.
děkan

Ing. Miroslav Matýsek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 17.05.2019 Andrej Kapusta, v. r.

PREHLÁSENIE

Čestne prehlasujem, že som diplomovú prácu spracoval samostatne s využitím vlastných teoretických poznatkov i praktických skúseností a že som uviedol všetky použité pramene a literatúru, z ktorých som čerpal.

Čestne prehlasujem, že odovzdaná verzia diplomovej práce a elektronická verzia nahraná do IS/STAG sú totožné.

Ďalej sa chcem poďakovať Ing. Petrovi Šilhavému, Ph.D. za odborné rady, ochotu a pripomienky pri vypracovaní diplomovej práce.

ABSTRAKT

Kapusta, Andrej: Mobilní aplikace pro sledování a analýzu výdajů během zahraničních cest. Diplomová práca, Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, Ústav počítačových a komunikačních systémů. Vedúci diplomovej práce: Ing. Petr Šilhavý, Ph.D., stupeň odbornej kvalifikácie: Inžinier v študijnom programe Počítačové a komunikační systémy.
Zlín: FAI UTB v Zlíne, 2019, 76s.

Cílem diplomové práce je vytvoření mobilní aplikace, která umožní uživateli zaznamenávání výdajů během zahraničních cest. Práce zároveň analyzuje bezpečnost mobilních zařízení z hlediska bezpečnostních hrozeb, kterým jsou v současnosti vystaveny. Analýza pokrývá jak popis hrozeb, tak i prevenci vůči těmto útokům z aspektu užívatele. V další části je práce zaměřena na popis vybraných existujících aplikací, které pokrývají problematiku a jejich vzájemné porovnání.

Následuje část s návrhem mobilní aplikace spolu s požadavky na vyvíjený systém a způsobem realizace. Tato část pokrývá i provedení užívatelského rozhraní spolu s ukázkou realizované funkcionality.

Klíčová slova: mobilní aplikace, bezpečnost, hrozby, výdaje, cesta

ABSTRACT

Kapusta, Andrej: A Mobile Application for Tracking and Analysing Spending on Foreign Journeys, Diploma thesis, Tomas Bata University in Zlín, Faculty of Applied Informatics, The Department of Computer and Communication Systems. Supervisor: Ing. Petr Šilhavý, Ph.D., degree of qualification: Master in the field of study Computer and Communication Systems.

Zlín: FAI UTB in Zlín, 2019, 76p.

The aim of the thesis is to create a mobile application that allows the user to record expenses during foreign journeys. The work also analyzes the security of mobile devices in terms of the security threats they are currently facing. The analysis covers both the description of threats and the prevention of these attacks from the user's point of view. The next part is focused on the description of selected existing applications that cover the issue and their comparison.

The following is a section with the mobile application application design along with the requirements for the developed system and implementation method. This section also covers the execution of the user interface along with a demonstration of the implemented functionality.

Keywords: mobile application, security, threats, expenses, journey

OBSAH

ÚVOD

Mobilné zariadenia sú v súčasnej dobe využívané v každom aspekte života. Môžeme ich nájsť v rôznych odvetviach, od priemyslu cez monitoring, zdravotníctvo až po edukáčné účely. Najväčšie zastúpenie majú však u bežných používateľov. Existuje obrovské množstvo odlišných aplikácií, ktoré ponúkajú pestrú funkcionalitu. Esenciálnou výhodou je však spojenie možnosti mobility zariadenia a používateľskej aplikácie.

Vzhľadom k tomu, že využívanie smartfónov má stále relatívne krátku históriu, hlavnou skupinou používateľov je predovšetkým mladšia generácia. S dostupnosťou cestovania po celom svete sa využívanie aplikácií stáva globálnym, čím ich možnosť využitia narastá.

Pri nízkonákladovom cestovaní mladých ľudí predstavuje základný stavebný prvok rozumné využívanie zdrojov. Zdrojom sa predstavuje vyčlenený stav financií, ktorý je možné využiť na aktuálny výlet. Keďže je častokrát náročné udržať sa v stanovených číslach, využitie smartfónu predstavuje ideálnu možnosť zaznamenávania výdavkov a uchovávanie aktuálnej informácie o stave financií.

Okrem tejto základnej informácie sa môže využiť aj iná funkcionalita smartfónu v podobe lokalizácie zariadenia či internetového pripojenia. Vďaka tomu si môže používateľ celý výlet zaznamenávať, výsledkom čoho bude report v podobe štatistík využitia zdrojov a miest, kde svoje výdaje uskutočnil.

I. TEORETICKÁ ČÁST

1 ÚVOD DO PROBLEMATIKY

Mobilné zariadenia prešli za posledné dve desaťročia obrovským vývojom[1]. Od prvých zariadení, ktoré ponúkali základnú funkcionalitu (volanie, sms, neskôr prenos dát) až po každodenné využívanie všetkými vekovými kategóriami[2]. Zvyšovanie výkonu mobilných zariadení spolu s nárastom rýchlosti dátového prenosu prostredníctvom prichádzajúcej 5G mobilnej siete a zavádzaniu *IoT* predpovedá ich využívanie v každom aspekte nášeho života.

Tento fakt predstavuje aj nárast pripojených mobilných zariadení do dátovej prevádzky, prostredníctvom mobilných dát alebo Wi-Fi siete, kde sa musí dbať na dostatočné zabezpečenie zariadení. Útočníci využívajú rôzne spôsoby, ako sa dostať k používateľským dátam: využívaním zraniteľností operačného systému zariadenia či sťahovaných aplikácií, nedostatočného zabezpečenia zariadenia používateľom a využívanie jeho návykov či zachytávanie sieťovej prevádzky.

Všetky tieto hrozby so sebou prinášajú riziko straty alebo odcudzenia citlivých dát ako pri zariadeniach patriacich bežným používateľom, tak aj pri zariadeniach pripájajúcich sa do firemnej sféry. Súkromné spoločnosti dbajú viac na zabezpečenie svojich zariadení z dôvodu možného úniku citlivých firemných dát, a to prostredníctvom používateľských pravidiel či zamerania sa na autorizáciu používateľa.

V nasledujúcej kapitole sa však zameriame viac na hrozby, ktorým denne čelia bežní používatelia a na spôsoby zabezpečenia, ako riziká tejto hrozby eliminovať. Bližšie si ukážeme, ako konkrétne útoky využívajú zraniteľnosť zariadení a spomenieme si pravidlá, ako by sa mal používateľ správať, aby sa zbytočne nevystavoval hrozbe útoku a predchádzal zneužitiu citlivých dát.

[1] Zdroj: https://www.uswitch.com/mobiles/guides/history-of-mobile-phones/
[2] Zdroj: https://www.statista.com/chart/3760/mobile-etiquette/

2 EXISTUJÚCE RIEŠENIA POKRÝVAJÚCE PROBLEMATIKU

Vzhľadom k tomu, že aplikácia je vytvorená pre operačný systém Android, existujúce podobné riešenia na sledovanie výdavkov sú vybrané len pre tento konkrétny operačný systém. Nižšie spomenuté aplikácie sú voľne prístupné prostredníctvom aplikácie Google Play, kde boli ponúkané. Ich testovaním som zisťoval ponúkanú funkcionalitu spoločne s výhodami a nevýhodami, ktoré sa pri používaní objavili.

2.1 Travel Money

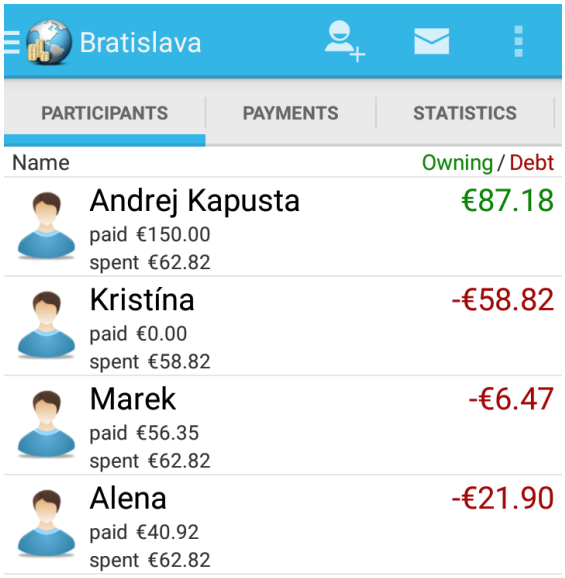
Aplikácia Travel Money[3] umožňuje jednoduchý spôsob zaznamenávania výdavkov prostredníctvom základného grafického rozhrania. Okrem základného spracovania je ponúkaná aj možnosť rozdelenia hodnoty pridaného výdavku medzi ďalších členov pri vytvorenej ceste. Táto funkcionalita zabezpečuje prehľadnosť množstva platieb a ich hodnôt vykonaných medzi pridanými členmi.

2.1.1 Funkcionalita

Podstatou aplikácie je zaznamenávanie výdavkov, ktorých suma sa prerozdeľuje medzi ostatných členov skupiny. Používateľ si môže do vytvorenej cesty zadať nelimitujúci počet členov, kde sa pri každej osobe zobrazuje aktuálny stav, t.j. suma, ktorú daná osoba minula, resp. suma, ktorú dlží voči ostatným členom skupiny.

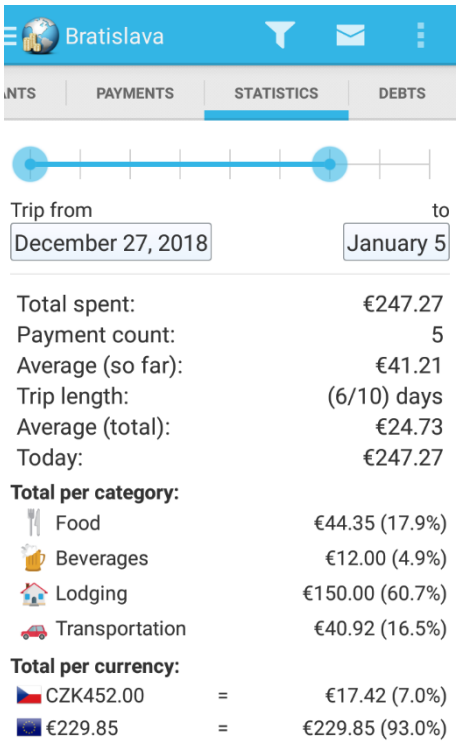
Každý nový výdavok obsahuje informáciu, ktorá osoba platbu vykonala. Možnosťou je aj platba viacerých členov súčastne. Používateľ si môže zadať kategóriu výdavku, menu, v ktorej bola platba vykonaná a výber osôb, ktoré participovali na výdavku. Každý výdavok uchováva adresu a čas vytvorenia výdavku. Na zaznamenávanie adresy je využívaná API Google Places.

[3] Viac na: http://travelmoney.marbotapps.com/



Obrázok 2: Zobrazenie používateľského rozhrania aplikácie Travel Money

Aplikácia ponúka zároveň jednoduchý štatistický prehľad o vykonaných výdavkoch. Informuje používateľa, aké množstvo prostriedkov bolo minutých na kategóriu alebo aká suma bola minutá v rámci meny. Keďže dominantou myšlienkou je zaznamenávanie výdavkov v skupine, jednou z funkcionalít je aj vypočítanie a zobrazenie sumy, akú si osoby v danej ceste navzájom dlžia.



Obrázok 1: Zobrazenie štatistiky výdavkov

Možnosťou aplikácie je aj vytvorenie reportu v podobe štatistiky platieb na konkrétny e-mail vo formáte PDF. Report obsahuje informácie o každej osobe a jej vytvorených výdavkoch.

2.1.2 Výhody

- možnosť pridávania viacerých členov do výletu,
- prepočítavanie výdavkov medzi členov výletu,
- zaznamenávanie lokalizácie výdavku,
- možnosť zaslania informácie o výdavkoch prostredníctvom e-mailu.

2.1.3 Nevýhody

- user Experience (UX) – neprepracovaný dizajn a nekonzistentnosť funkcionálnych prvkov aplikácie na rôznych fragmentoch obrazovky.

2.2 Travel Spend

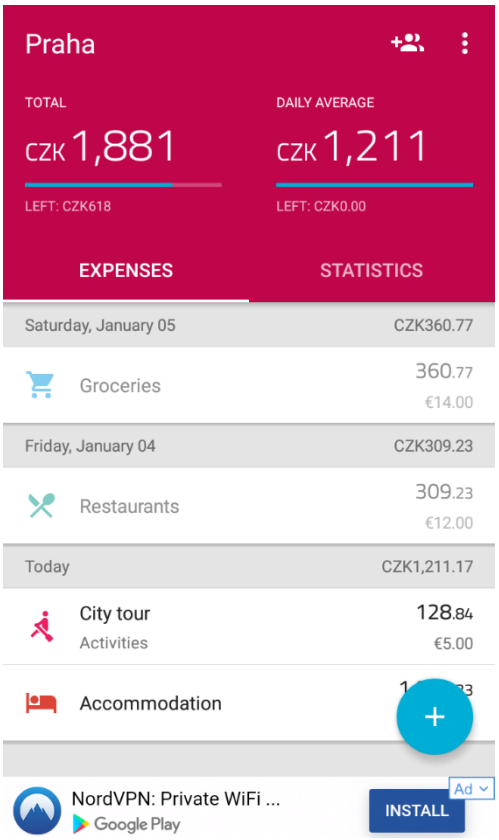
Aplikácia Travel Spend[4] predstavuje vizuálne prakticky spracovanú aplikáciu na zaznamenávanie výdavkov, čo pre používateľa znamená jednoduchú otrientáciu a rýchlu manipuláciu. Obmedzenie aplikácie predstavuje fakt, že časť funkcionality je spoplatnená. V prípade, že používateľ nechce za túto časť zaplatiť, môže spravovať maximálne jednu cestu, nemôže pridávať viacero osôb či exportovať výlet. Napriek týmto obmedzeniam je aplikácia použiteľná pre správu výdavkov.

2.2.1 Funkcionalita

Základná obrazovka aplikácie zobrazuje používateľovi základné informácie o sume, ktorú už minul, koľko mu zo zadaného rozpočtu ostáva či priemernú útratu za jeden deň. Zoznam výdavkov je spracovaný v liste podľa dátumu vytvorenia. Každý výdavok zobrazuje sumu v mene uloženej pri vytvorení cesty (napr. cesta do Českej republiky – mena Česká koruna) a sumu v mene, ktorá je zadaná ako domáca mena používateľa (napr. používateľ zo Slovenskej republiky – mena Euro).

[4] Vaic na: https://travelspend.launchaco.com/

Koláčový graf zobrazuje štatistiku výdavkov podľa kategórií v percentuálnom zobrazení, ako aj celkovú sumu, ktorá bola vynaložená na konkrétnu kategóriu.



Obrázok 3: Hlavné menu aplikácie Travel Spend

2.2.2 Výhody

- jednoduché používateľské prostredie,
- spôsob zobrazenia minutých a zostávajúcich prostriedkov.

2.2.3 Nevýhody

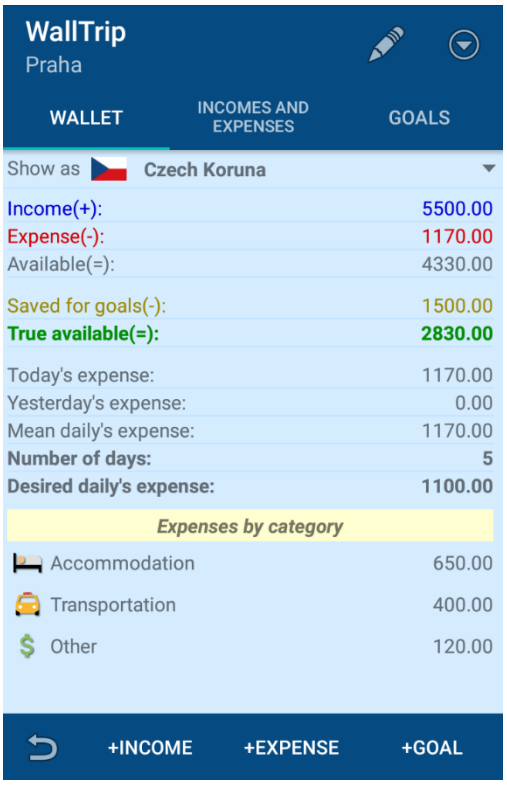
- limitovaná funkcionalita platenou verziou,
- žiadna lokalizácia výdavkov.

2.3 WallTrip

WallTrip[5] predstavuje jednoduchú aplikáciu na zaznamenávanie výdavkov pre jednotlivca. Je založená na jednoduchom dizajne pre používateľa so základnou funkcionalitou. Oproti predošlým mobilným aplikáciam umožňuje používateľovi okrem zadávania výdavkov aj zaznamenávanie príjmov alebo stanovovanie cieľov v rámci kategórií.

2.3.1 Funkcionalita

Základná obrazovka vytvoreného výletu zobrazuje aj aktuálnu štatistiku výdavkov. Používateľ má k dispozícii informácie o hodnote výdavkov a príjmoch, hodnote priemerných výdavkov či informáciu, akú priemernú sumu v rámci rozpočtu je možné denne minúť. Základná obrazovka taktiež zobrazuje informáciu, aká suma bola vynaložená na špecifickú kategóriu.

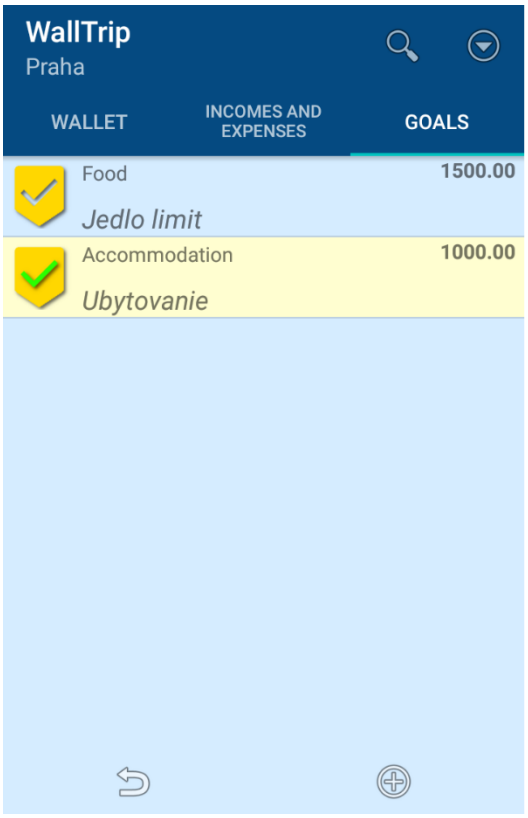


Obrázok 4: Zobrazenie používateľského prostredia aplikácie WallTrip

[5] Viac na: https://play.google.com/store/apps/details?id=com.evposli.mn.walltrip&hl=en_US

Vytváranie výdavku obsahuje len niekoľko základných možností. Používateľ si môže vybrať, či ide zadávať výdavok alebo príjem, hodnotu v mene, ktorú zadával pri vytvorení výletu, výber z kategórie a dátum vytvorenia. Aplikácia graficky znázorňuje farebný rozdiel medzi uloženým príjmom a výdavkom.

Navyše funkcionalita predstavuje možnosť zadania vlastného cieľa. Používateľ si zadá hodnotu vo vopred uloženej mene a špecifikuje kategóriu, ktorej sa hodnota dotýka. Takýmto spôsobom si dokáže zaznamenávať hranice výdavkov a je konkrétnejšia a vizuálne poskytuje lepšiu informovanosť o aktuálnej finančnej disponzibilite.



Obrázok 5: Prostredie funkcionality Goals

2.3.2 Výhody

- jednoduché spracovanie aplikácie,
- zadávanie cieľov,
- stanovenie rozpočtu,
- možnosť pridávania nie len výdajov, ale aj príjmu,
- offline aplikácia bez potreby využívania internetového pripojenia alebo GPS.

2.3.3 Nevýhody

- prepočet meny je stanovený len na základe hodnoty stanovenej používateľom,
- hlavná obrazovka zobrazuje príliš veľa informácií, čo môže byť pre používateľa neprehľadné,
- chýba zaznamenávanie polohy výdavku.

2.4 Zhodnotenie existujúcich riešení

Existujúce aplikácie ponúkané pre operačný systém Android, vrátane vyššie opospomenutých, ponúkajú rôzne možnosti funkcionality a zaznamenávania pre jednotlivca alebo skupinu používateľov s jedným administrátorom. Okrem základného zaznamenávania výdavkov na základe vybranej kategórie obsahujú niektoré aplikácie možnosť GPS lokalizácie výdavku či navyšovanie rozpočtu. Z celkového pohľadu je však zaznamenávanie informácií o výdavkoch jednoduché a väčšinou sa nekladie veľký dôraz na množstvo informácií, ktoré sa spolu s výdavkom môžu uchovať.

Veľké množstvo aplíkácií ponúka širokú funkcionalitu, ktoré sú však limitované spoplatnením za ich používanie. Pri aplikáciach, ktoré používateľovi ponúkajú plnú funkcionalitu zadarmo zase stagnuje grafické prevedenie, ktoré je buď príliš komplexné alebo celkový dojem detailov nie je prepracovaný. Ďalšie podobné riešenia, ktoré ponúkajú približne rovnakú funkcionalitu prezentujú aplikácie Trabee Pocket, Budget Travel, Travel Cutter či Fast Budget.

3 HROZBY A ÚTOKY

V danej kapitole sa bližšie pozrieme na hrozby, ktorým čelia používatelia a mobilné zariadenia pripojené do dátovej prevádzky. Môžeme ich rozdeliť do konkrétnejších kategórií podľa zamerania na hrozby mobilných aplikácií, webové, sieťové a fyzické hrozby. Ukážeme si ich konkrétne spôsoby ich využívania, ako aj metódy zabezpečenia proti týmto hrozbám. [1]

3.1 Hrozby mobilných aplikácií

Ako predstavuje názov, do tejto kategórie spadajú hrozby, ktoré sú priamo viazané na aplikácie dostupné prostredníctvom obchodov (Google Play, Apple Store), zraniteľné aplikácie či aplikácie nainštalované z neoverených zdrojov. Takéto aplikácie môžu obsahovať *malware*, t.j. škodlivý softvér, ktorý môže útočníkovi slúžiť k získaniu osobných alebo prístupových údajov či dokonca k prevzatiu kontroly nad zariadením – prístup ku kontaktom, emailom, fotkám, zasielanie SMS.

Okrem malware-u sa útočníci môžu dostať k citlivým údajom aj prostredníctvom existujúcich zraniteľností aplikácie. Zraniteľnosti môžu vychádzať z nedostatočnej úrovne zabezpečenia pri vývoji softvéru alebo chýbajúcich aktualizácií aplikácie pri jej prevádzke. Príkladom objavenia zraniteľností a slabín zobrazuje penetračný test aplikácie *Australian Goverment Healthcare*, ktorým sa dokázali dostať až k citlivým dátam používateľa, ako sú meno a dátum narodenia, číslo poistky, poskytovateľ zravotnej starostlivosti, či využívané služby zdravotnej starostliovosti.[2]

3.2 Webové hrozby

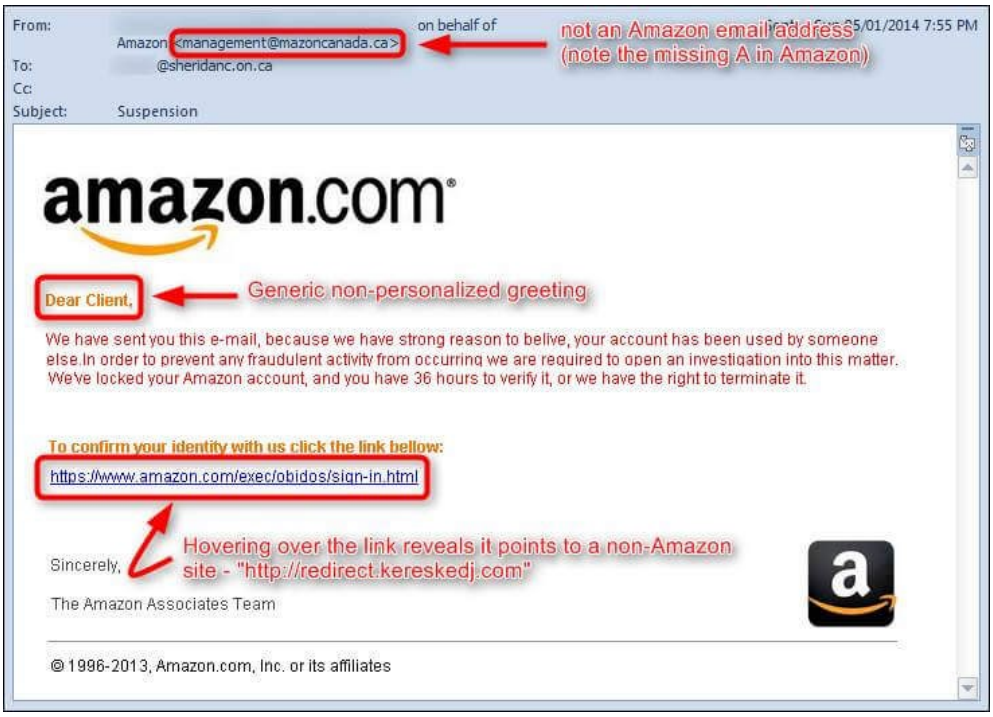
Webové hrozby sú všade tam, kde používateľ využíva aplikácie pripájajúce sa do internetu. Tieto hrozby v sebe skrývajú aj využívanie neznalosti používateľa, ktorý otvorí odkaz alebo stránku, ktorá sa maskuje za dôveryhodný zdroj, tzv. *phishing*. Takéto spôsoby útokov boli v minulosti využívané predovšetkým prostredníctvom emailu, v súčasnosti sa však využívajú aj v mobilných aplikáciach ako reklamy, či odkazy na rôzne zaujímavé články.

Okrem phising-u sú medzi webovými útokmi obľúbení aj *boti* (ang. *bots*). Tí predstavujú malware, ktorý infikuje zariadenie a kontaktuje centrálny server o úspešnom pokuse. Následne je bot schopný nepozorovane odosielať zhromaždené informácie späť na centrálny

server, prípadne čakať na spustenie DoS útoku[6] prostredníctvom infikovaného zariadenia. Hlavnou výhodou botov je vlastnosť, že sa vedia v systéme nepozorovane skrývať. Častokrát sa tvária ako súbory či procesy s podobným alebo takmer identickým názvom, ako bežné systémy. [3]

3.2.1 Phising

Phising predstavuje typ útoku, kedy sa útočník nesnaží preniknúť do zariadenia alebo aplikácie, ale využíva iba nepozornosť a neskúsenosť samotného používateľa. V minulosti sme sa mohli formou phishing-u najčastejšie stretnúť pri e-mailovej komunikácií, kedy útočníci využívali podvodné emailové správy vyžadujúce osobné údaje. S nástupom mobilných zariadení sa phishing stal sofistikovanejším nástrojom a viac sa zameral na mobilné aplikácie. Výskumy ukazujú, že až 48% takýchto útokov je mierených na mobilné zariadenia a používatelia sú 3x zraniteľnejší voči týmto útokom. [4]



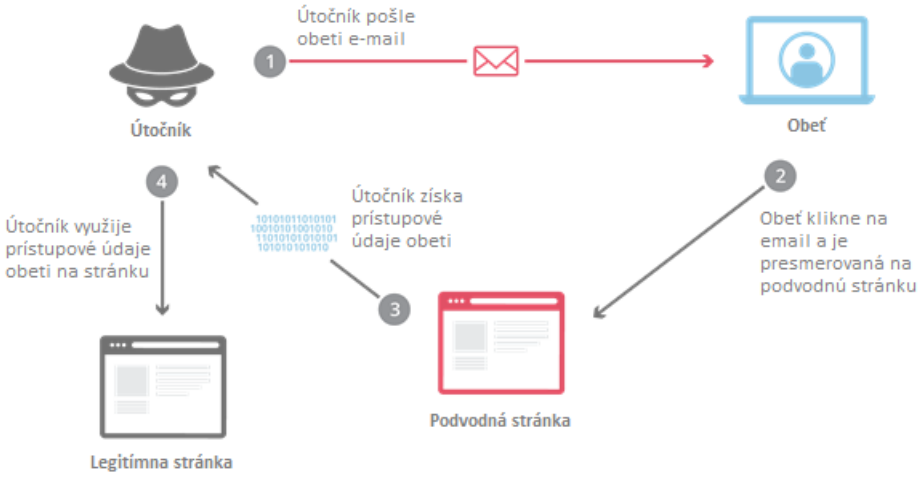
Obrázok 6: Príklad emailu pri phishingovom útoku. [15]

[6] Zdroj: https://us.norton.com/internetsecurity-emerging-threats-dos-attacks-explained.html

Najbežnejším príkladom phising-u je zasielanie podvodných emailov, ktoré majú prvky existujúcej inštitúcie, kde je používateľ zaregistrovaný a požadujú overenie prihlasovacích údajov, prípadne môžu odkazovať na podvodnú webovú stránku. Útočníci využívajú fakt, že používatelia dostávajú veľké množstvo emailov a dôkladne si neoverujú charakteristické prvky emailov.

Takýto útok nemusí prebiehať len prostredníctvom emailu, ale aj reklamou alebo zaujímavým článkom, ktorý používateľ nájde na internete. Okrem prihlasovacích údajov môže mať phishing povahu webového formulára, ktorý používateľa žiada vyplniť osobné údaje vrátane informácií o kreditnej karte. Takto sa vie útočník jednoducho dostať k veľkému rozsahu súkromných údajov len za pomoci samotného používateľa.

Aby však takéto stránky vzbudzovali dôveryhodnosť, využívajú *HTTPS*[7] zabezpečenie phishing-ovej lokality prostredníctvom získania *SSL* certifikátu, ktoré sú ponúkané rôznymi službami[8].



Obrázok 7: Spôsob realizácie phishingového útoku. [16]

Pri mobilných aplikáciach je v súčasnosti phising zameraný hlavne na mobilné aplikácie sociálnych sietí, kde phishing-ový útok tvorí až 34% všetkých útokov. Stále je však vo veľkej miere zameraný na klamlivé e-maily alebo odkazy pri mobilných hrách. [5]

[7] Zdroj: https://www.tutorialsteacher.com/https/what-is-https
[8] Zdroj: https://www.wired.com/story/phishing-schemes-use-encrypted-sites-to-seem-legit/

3.3 Sieťové hrozby

Sieťová hrozba predstavuje zneužitie komunikácie, ktorá prebieha medzi zariadeniami v sieti. Najväčšiu hrozbu predstavuje využívanie verejnej Wi-Fi siete. Takéto siete bez zabezpečenia, resp. so zastaraným zabezpečením WEP alebo WPA, predstavujú pre útočníkov obrovský potenciál. Používatelia, ktorí prevádzajú sieťovú komunikáciu v takejto sieti, vystavujú všetku svoju komunikáciu prakticky navonok. Útočník sa prostredníctvom útoku *Man-in-the-middle* dostane k používateľovým údajom úplne bez jeho vedomia.

3.3.1 Man-in-the-middle

Útok *Man-in-the-middle* (muž v strede) využíva komunikáciu medzi dvoma stranami. Pri mobilných zariadeniach to býva najčastejšie komunikácia medzi aplikáciou a databázovým serverom. Podľa odhadu, až 75% z 1000 najpoužívanejších aplikácií na Google Play nekontroluje platnosť certifikátov k pripájaným serverom a ignoruje SSL chyby pri komunikácií.[6] Takýto útok je zameraný hlavne na získanie citlivých dát pri prenose (obchodné tajomstvá, finančné údaje). Zraniteľnosť predstavuje fakt, že útočník je schopný sa dostať až k polohe zariadenia, čítaniu správ alebo dokonca k odpočúvaniu konverzácie.

Obrázok 8: Spôsob realizácie útoku Man-in-the-middle. [17]

Princípom tohoto útoku je zachytiť komunikáciu, ktorá prebieha medzi zariadeniami bez jej samotného narušenia. Útočník prakticky len počúva prenášané dáta a ukladá si potrebné informácie.

3.4 Fyzické hrozby

Tieto hrozby nemajú softvérový charakter, ale sú založené na fakte, že mobilné zariadenia nosíme v dnešnej dobe všade. Aj v súčasnosti existuje veľké množstvo zariadení, ktoré nemajú ani základné zabezpečenie zamknutia zariadenia v podobe PIN-u, vzoru alebo odtlačku prstu. Z takéhoto zariadenia sa dá za malý okamih zistiť veľké množstvo údajov o používateľovi, ktoré môžu byť následne zneužité.

Ďalšou hrozbou je strata súkromných údajov pri strate alebo krádeži zariadenia. V takomto prípade, i keď má zariadenie vstupný kód, je útočník schopný prelomiť toto zabezpečenie prostredníctvom špeciálnych aplikácií tretích strán. Preto je potrebné, aby používateľ chránil svoje dáta aj formou zablokovania zariadenia alebo vymazaním všetkých údajov na zariadení.

3.5 Hrozby úniku dát

Mobilné aplikácie uchovávajú len určité množstvo používateľských dát. Ďalšie dáta sú uchovávané na databázových serveroch, ktoré predstavujú pre útočníkov cenný zdroj informácií. Webové aplikácie a servery sú preto vystavené hrozbám, napríklad v podobe neprístupnosti serverov prostredníctvom DoS útoku. Pri takomto útoku sa server stane neprístupným, pretože je nútený v krátkom časovom okamihu vyriešiť obrovské množstvo prístupov na konkrétnu lokalitu. V tomto prípade sa útočník nesnaží získať dáta, ale znefunkčniť konkrétnu lokalitu, napríklad v prípade získania konkurenčnej výhody.

3.5.1 SQL Injection

SQL injekcia predstavuje bezpečnostnú chybu, kedy útočník využíva štruktúru SQL dopytu pri prístupe do databázy. [7] Typickým príkladom je zadávanie prihlasovacích údajov alebo registrácia používateľa. Základnou myšlienkou útoku je získanie existujúcich dát, ich modifikácia alebo zrušenie určitých tabuliek či dokonca samotnej databázy. K týmto útokom môže prichádzať vždy, keď sa aplikácia spolieha na vstupné parametre od používateľa pri vytvorení SQL dotazu. Tým, že tieto údaje vstupujú priamo do databázy, je potrebné ich validovať príslušným vzorom. [8]

Popísaný príklad SQL injekcie zobrazuje, ako môže útočník využiť zraniteľnosť aplikácie a autentifikovať sa ako administrátor:

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']

# SQL query vulnerable to SQLi
sql = "SELECT id FROM users WHERE username='" + uname + "' AND password='" + passwd + "'"

# Execute the SQL statement
database.execute(sql)
```

Obrázok 9: SQL pseudokód pre príklad SQL injekcie. [18]

Na obrázku (Obrázok 9) je zobrazený pseudokód vykonaný na web serveri prostredníctvom jednoduchej autentifikácie menom a heslom. Opisuje tabuľku *users* so stĺpcami *username* a *password.*

V prípade, že by útočník použil pre premennú *passwd* reťazec „*password' OR 1=1*", výsledný SQL dopyt by vyzeral:

SELECT id FROM users WHERE username='username' AND password='password' OR 1=1

Pri tomto dopyte je najpodstatnejšia časť OR 1=1, kedy WHERE klauzula vráti prvé ID z tabuľky používateľov, čo sú veľmi často údaje administrátora. Takto sa útočník dostane cez autentifikáciu do databázy bez ohľadu na používateľské meno a heslo.

Aby sme zabezpečili túto zraniteľnosť, je potrebné overovať údaje, ktoré vstupujú do SQL dopytu, napr. valídnu formu emailovej adresy alebo bezpečnejším SQL dopytom. [8]

4 ZABEZPEČENIE

Ako sme sa dozvedeli, mobilné zariadenia a používateľské či súkromné údaje sú pod neustálou hrozbou úniku alebo zneužitia. Aby sme túto možnosť eliminovali alebo aspoň znížili na najnižšiu možnú mieru, je potrebné dodržiavať základné bezpečnostné zásady a aplikácie, ktoré pomáhajú hrozbám predchádzať alebo zabezpečovať komunikáciu.

4.1 Pricípy zabezpečenia mobilného zariadenia

Aby sa používateľ chránil aspoň základným zabezpečením, je potrebné, aby dodržiaval určité bezpečnostné princípy:

- Uzamknutie zariadenia (PIN, vzor, odtlačok prsta),
- aktualizácie aplikácií,
- odstránenie starých a nepoužívaných aplikácií,
- odstraňovanie osobných informácií zo zariadenia,
- zálohovanie dát,
- používanie antivírusových programov a programov detekujúcich malware.

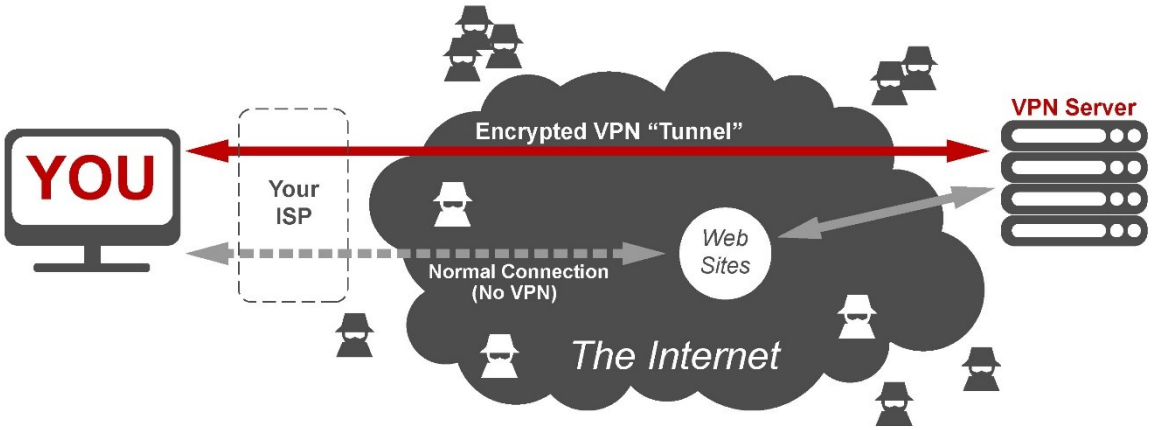
Okrem týchto základných princípov by používateľ mal dbať aj na ďalšie bezpečnostné odporúčania. Vzhľadom na obrovské množstvo existujúcich aplikácií, ktoré sú k dispozícií, by si mal okrem hodnotení danej aplikácie používateľ sledovať, aké povolenia pri inštalácií danej aplikácií povoľuje. Mal by sa zamyslieť nad tým, či skutočne aplikácia potrebuje prístup napr. k SMS, kontaktom či fotoaparátu. Sťahovaných aplikácií sa týka aj skutočnosť, že by ich mal používateľ sťahovať len z oficiálnych úložísk – Google Play, Apple's App Store, Amazon's Appstore a i.

4.2 Zabezpečená komunikácia

V kapitole 2.3 sme spomenuli odporúčanie, aby používateľ v žiadnom prípade nevstupoval do verejnej Wi-Fi siete [10]. Avšak navyše, aby svoju komunikáciu čo najviac zabezpečil v akejkoľvek sieti, je potrebné, aby ju zašifroval.

Pre tento účel je využívaná *VPN (ang. Virtual Private Network)*. Jedná sa o vytvorenie tunelu medzi komunikujúcimi stranami, pričom všetka prebiehajúca komunikácia je zašifrovaná. V prípade úspešného útoku, kedy by sa útočník dostal k prenášaným dátam, by získal iba zašifrovanú podobu, ktorá bude pre neho zbytočná. Okrem zabezpečenie prebiehajúcej komunikácie je používateľ čiastočne chránený pred identifikáciou

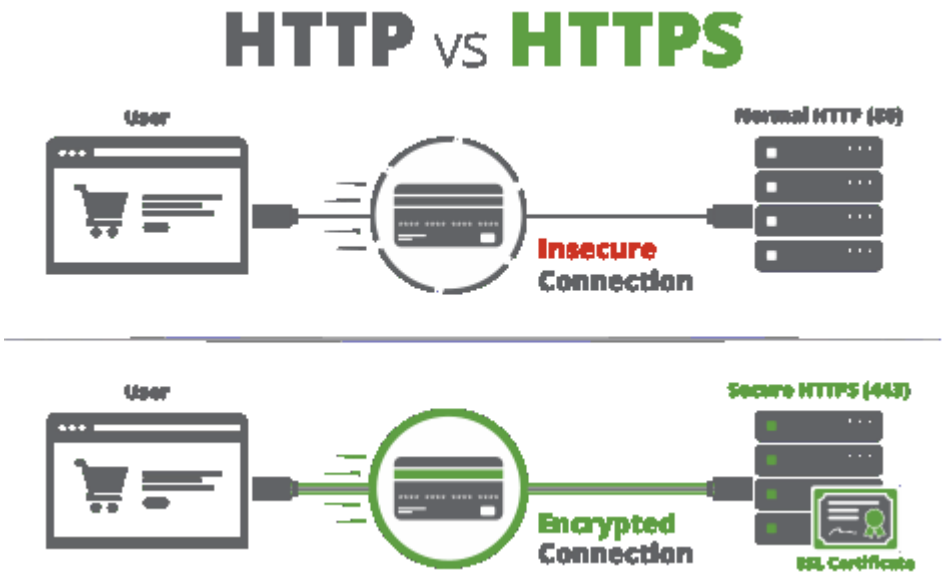
používaného zariadenia (IP adresa). To je zabezpečené VPN serverom, ktorý sa v komunikácií nachádza na začiatku tohoto procesu, takže svet v podobe internetu vidí len komunikujúci VPN server. Napriek tomu sa nejedá o dokonalé zabezpečenie komunikácie voči všetkým, z dôvodu, že každý štát si môže legislatívne udržiadať možnosť nahliadnuť do komunikácie, ktorá serverom prebiehala. Napriek tomuto faktu sa jedná o dôkladné zabezpečenie komunikácie. [11]



Obrázok 10: Zobrazenie princípu fungovania VPN. [19]

4.3 Zabezpečená autentizácia

Do bezpečnej komunikácie je potrebné rátať aj so zabezpečenou autentizáciou používateľa do príslušnej webovej aplikácie prostredníctvom protokolu HTTPS. Ten funguje na princípe šifrovanej komunikácie medzi klientom a serverom prostredníctvom *SSL* alebo *TLS* protokolu. Tento protokol nám zabezpečuje, že údaje, ktoré sú odosielané na server, najčastejšie používateľské meno a heslo, sú chránené šifrovaním. V opačnom prípade by mohla byť komunikácia zachytená a útočníkovi by boli poskytnuté tieto údaje v podobe textu.



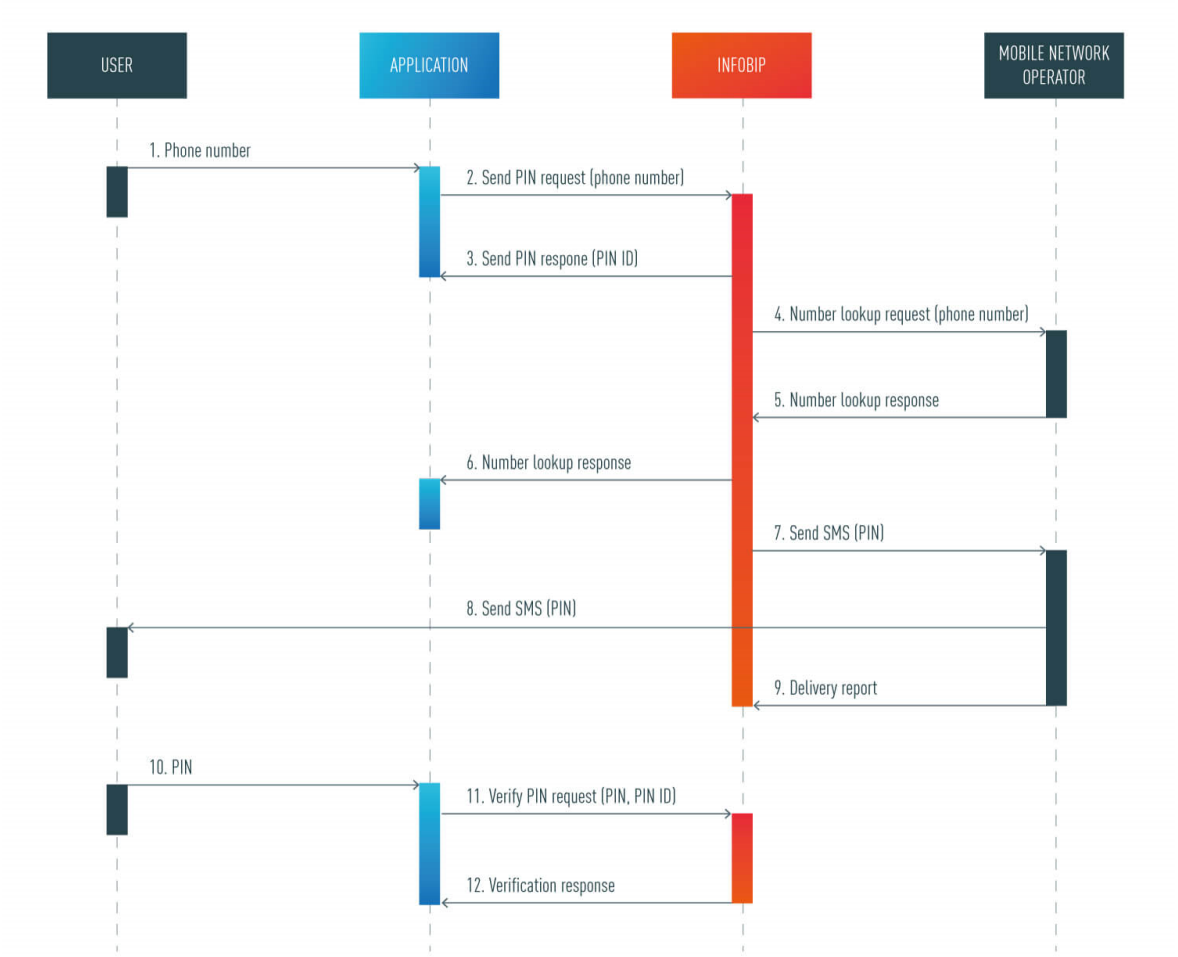
Obrázok 11: Porovnanie nezabezpečenej a zabezpečenej komunikácie pri HTTPS. [20]

Spomenutý HTTPS protokol zabezpečuje autentizáciu používateľa pri komunikácií so serverom, ale v prípade získania prihlasovacích údajov je prístup k webovej aplikácií vystavený hrozbe zneužitia. Pre väčšiu ochranu autentizácie sa využíva tzv. *2FA (ang. Two-Factor Authentication)*. Zabezpečuje, že okrem autentifikácie používateľa prostredníctvom prihlasovacích údajov a prostredníctvom HTTPS zdvojíme ochranu využitím tzv. tokenu. Token predstavuje hardvérový alebo softvérový nástroj, prostredníctvom ktorého je možné overiť používateľa prostredníctvom jednorázového generovaného kódu.

Hardvérová forma môže predstavovať zariadenie, ktoré vygeneruje náhodnú sériu číslic, ktoré sú následne porovnávané s aplikáciou tretej strany, ktorá tieto hodnoty vygenerovala. Ďalšie možnosti sú prostredníctvom napr. USB tokenov.

Pri softvérovej forme boli častokrát využívané tokeny vo forme SMS, kedy používateľovi bola zaslaná SMS, ktorá obsahovala sériu číslic pre identifikáciu. V súčasnosti sa ale táto forma autentifikácie neodporúča, pretože útočníci vedia získať obsah SMS správ prostredníctvom SS7[9] alebo skenovaním zariadenia.[12]

[9] Zdroj: https://www.kaspersky.com/blog/ss7-attack-intercepts-sms/16877/



Obrázok 12: Ukážka spôsobu komunikácie pri 2FA. [21]

V dnešnej dobe existuje niekoľko aplikácií tretích strán, ktoré slúžia na 2TF a sú využívané v rôznych sektoroch[10]. Tieto aplikácie ponúkajú rôzne dodatočné funkcionality a chránia proti zneužitiu prihlasovacích údajov. [13]

4.4 Zabezpečenie dát

Každá mobilná aplikácia využíva určité úložisko pre uchovávanie nahrávaných dát používateľom. Môže to byť využívaním internej databázy zariadenia alebo pripájaním sa na vzdialený databázový server. Keďže samotný používateľ nemá možnosť ovplyvňovať ako

[10] Zdroj: https://twofactorauth.org/

a kde sú dáta aplikácií ukladané, v tejto časti sa pozrieme na zabezpečenie úložísk z pohľadu vývoja aplikácie.

Vývojári mobilných aplikácií musia okrem vizuálnej podoby aplikácie a jej funkcionality zamerať aj na zabezpečenie zbieraných dát. Miera zabezpečenia sa môže odvýjať aj od toho, o aký typ ukladaných dát sa jedná (prihlasovacie údaje, dáta aplikácie) či v závislosti na type aplikácie, napr. aplikácia banky na prístup k účtu. Pri takýchto aplikáciach je potreba dbať na autentifikáciu používateľa (viac vrstvové overovanie) či na komunikáciu so vzdialeným serverom.

Dáta spravovanej aplikácie môžu byť uchovávané prostredníctvom interného úložiska zariadenia, externej SD karty, cache pamäte, internej databázy vytvorenej pre danú aplikáciu či externej databázy, na ktorú sa zariadenie pripája prostredníctvom internetu. Zameriame sa na najbežnejšie ukladanie dát a to prostredníctvom internej databázy zariadenia a externého úložiska.

4.4.1 Interná databáza zariadenia

Operačný systém Android ponúka pri vývoji aplikácií ukladanie dát využitím databázy **SQLite.** Táto databáza nepredstavuje komunikáciu klient/server, takže nie je potrebná komunikácia cez internet, ale uloženie prebieha na súbor, ktorý je umiestnený v hierarchickej štruktúre zariadenia. Pre každú aplikáciu je vždy vytvorená samostatná inštancia databázy, čo v praxi znamená, že jedna aplikácia nemá prístup do databázy druhej aplikácie. V prípade, že chceme databázu zdieľať a povoliť prístup iných aplikácií k jej obsahu, je potrebné implementovať nástroj *Content provider*[11].

Samotná databáza v jej základe neponúka žiadne zabezpečenie dát, pretože údaje sú ukladané do súboru vo formáte *plain text*. To znamená, že dáta nie sú šifrované a pri získaní prístupu do zariadenia sú vystavené priamemu zneužitiu.

[11] Zdroj: https://developer.android.com/guide/topics/providers/content-providers

Je teda na vývojárovi aplikácie, aby na základe štruktúry dát zabezpečil ich dostatočnú ochranu prostredníctvom:

A. Zablokovanie zálohovania databázy

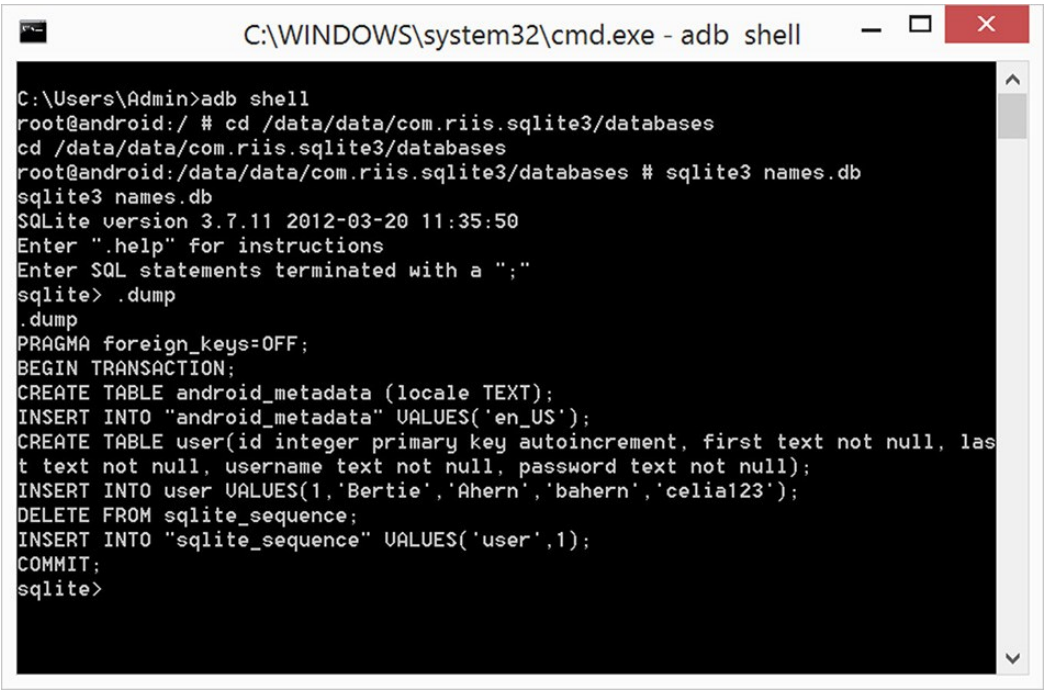
Základnú ochranu dát v zariadení predstavuje blokovanie zariadenia (heslo, pin, vzor). Pri takto blokovanom zariadení má útočník obmedzené možnosti prístupu k dátam. V prípade, že sa útočníkovi podarí prelomiť túto ochranu a dostane sa k súborovej štruktúre zariadenia, je potrebné údaje samostatne zabezpečiť.

SQLite databázu je možné jednoduchým spôsobom zálohovať, čo znamená, že útočník si môže veľmi rýchlo vytvoriť kópiu uložených dát. Android aplikácie majú štandardne nastavenie umožňujúce zálohovanie databázy. Zablokovaním tejto možnosti zabezpečíme, že aj pri zálohovaní celého systému bude databáza vyňatá z tejto procedúry.

Zakázanie zálohovania sa nastavuje priamo v zdrojovom kóde aplikácie v súbore *AndroidManifest.xml* prostredníctvom príkazu **android:allowBackup = "false"**[12].

Tento spôsob zálohovania však nepredstavuje úplnú ochranu dát. V prípade, že sa útočníkovi podarí „rootovať" zariadenie, t.j. dostať sa k najvyšším administrátorským právam, je schopný sa dostať prostredníctvom shell príkazov do súborovej štruktúry zariadenia, priamo k databázovému súboru a vypísať databázové tabuľky pre zobrazenie dát. [8]

[12] Zdroj: https://developer.android.com/guide/topics/manifest/application-element#allowbackup



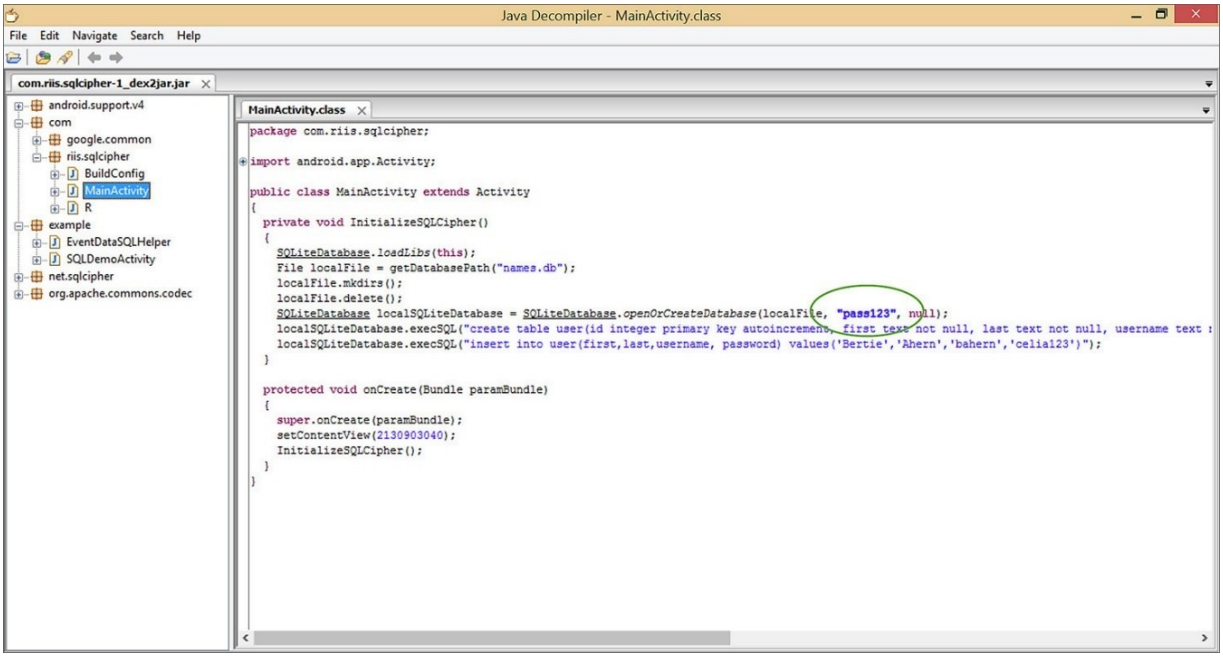
Obrázok 13: Spôsob získania obsahu databázy prostredníctvom shell príkazov. [22]

B. Šifrovanie ukladaných dát

Okrem ochrany dát pred zálohovaním, kedy je možné sa k nim dostať prostredníctvom „root“ módu predstavuej lepšie zabezpečenie ich samotné šifrovanie. Pri Android aplikáciach je možné využiť šifrovanie dát prostredníctvom nástoja **SQLCipher**[13]. Tento nástroj využíva *256-bit AES* šifrovanie, čo znamená, že na prelomenie tejto šifry je potrebné zistiť 2^{256} rôznach kombinácií, čo predstavuje dostatočnú ochranu aplikácií komerčného využitia.

Hoci údaje v databáze poskytujú dostatočnú ochranu, je potrebné uvažovať nad bezpečným uložením šifrovacieho kľúča. Na obrázku (Obrázok 9) je znázornenie uloženia kľúča pri inicializovaní šifrovania. Takto uložený kľúč je nezabezpečený a jeho získaním (napr. dekompiláciou kódu) sa útočník opäť jednoducho dostane k obsahu databázy.

[13] Zdroj: https://www.zetetic.net/sqlcipher/



Obrázok 14: Zaznamenanie hesla pri šifrovaní databázy. [23]

Preto je dôležité nezabúdať aj na uloženie, skrytie tohoto kľúča. Prvou možnosťou je uloženie nastavenia kľúča používateľom. Používateľ by pri vstupe do aplikácie zadal heslo, aby odblokoval možnosť práce s databázou. Avšak, tento postup obmedzuje používateľa a zvyšuje (nefunkcionálnu) interakciu s aplikáciou.

Ďalšou možnosťou je uloženie kľúča v internej pamäti zariadenia, v *Shared preferences*. Takto uložený kľúč môže byť nastavený pri prvom použití aplikácie a pri ďalších spusteniach sa bude načítavať s daného úložiska. Problémom je, že napriek zabezpečeniu databázy existuje možnosť získania údajov zo *Shared preferences* prostredníctvom nástrojov tretích strán, čo predstavuje riziko získania šifrovacieho kľúča.

Existujú aj ďalšie možnosti skrytia šifrovacieho kľúča v podobe uloženia premennej kľúča v kóde aplikácie či uložením do *Native Development Kit* (NDK). Pravdepodobne najbezpečnejšou variantou uloženia kľúča je v podobe externej databázy na princípe klient/server. V takomto prípade je kľúč uložený mimo zariadenia a používateľ ho získava prostredníctvom internetového pripojenia cez *web servis*. [8]

4.4.2 Externá databáza

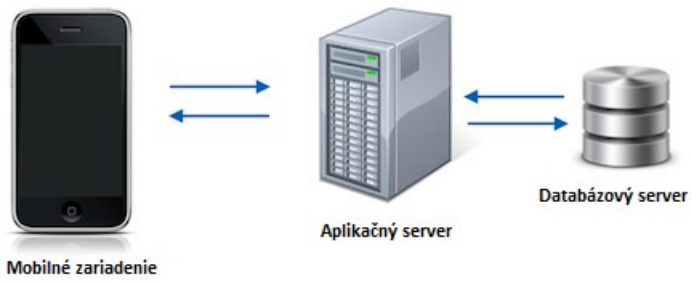
Pri rozsiahlejších aplikáciách s veľkým množstvom používateľov alebo ukladaných dát predstavuje externé úložisko v podobe databázového serveru alebo cloudu jednoznačnú výhodu. Zároveň však treba mať napamäti, že toto rozhodnutie sprevádza niekoľko faktorov.

Online stav:

Aplikácie môžu využívať súčasne lokálne úložisko (v offline stave) a pri nadobudnutí internetového pripojenia nastane synchronizácia všetkých dát medzi mobilným zariadením a serverom (štandardne nastavené pri WiFi sieti). V takomto prípade musí používateľ počkať, pokiaľ prebehne celková synchronizácia, resp. výkon zariadenia môže byť v dôsledku toho obmedzený. Keďže sa zariadenie synchronizuje hlavne na WiFi sieti, pri automatickom pripojení vo verejnom hotspot-e môže byť zariadenie vystavené útokom (kapitola 2.3).

Zabezpečenie databázy:

Preto by sme nemali zabúdať na dostatočné zabezpečenie komunikácie medzi klientom a serverom (kapitola 3.2). Okrem samotného zabezpečenia komunikácie je možné systém rozčleniť, aby rôzne entity mali rôzne prístupy medzi rozdielne časti systému. Medzi mobilné zariadenia a databázový server umiestnime aplikačný server, ktorý bude obhospodarovať databázu a súčasne autentifikovať používateľa, ktorý sa na databázu dopytuje. Takýmto spôsobom môžeme obmedziť prístup k databázy a súčasti definovať rôzne role používateľov (administrátor, používateľ). Takto definovaný prístup tvorí súčasne dvojitú ochranu databázy, kedy sa musí dopytujúce zariadenie autentifikovať na aplikačnom serveri a zároveň nemá priamy prístup na databázový server.



Obrázok 15: Zobrazenie komunikácie mobilného zariadenia so serverovou časťou.

II. PRAKTICKÁ ČÁST

5 ŠTRUKTURÁLNY OPIS APLIKÁCIE

V nasledujúcich kapitolách sa bližšie pozrieme na štruktúru aplikácie t. j. z akých prvkov je zložená a opíšeme si jej funkcionalitu spolu s používateľským prostredím.

5.1 Cieľ aplikácie

Účelom aplikácie je používateľovi zabezpečiť možnosť zaznamenávať výdavky podľa stanovenej cesty a prehľadne ho informovať o súbore dát, ktoré vložil do systému.

5.2 Funkcionalita aplikácie

5.2.1 Vytvorenie cesty

Pre možnosť zadávania výdavkov je potrebné, aby si používateľ vytvoril cestu, pre ktorú budú vytvorené výdavky existovať. Cesta bude uchovávať základné údaje, ako sú počet dní, vstupný rozpočet cesty, aktuálny rozpočet a menu na základe krajiny, do ktorej používateľ aktuálne cestuje.

Aplikácia uchováva dve rôzne meny. Jednou je predvolená mena aplikácie, ktorú si používateľ nastavuje prostredníctvom nastavení a druhou mena konkrétneho výletu. Výhodou tejto možnosti je jednoduchšia orientácia o aktuálnom stave financií používateľa, vzhľadom k tomu, že disponuje informáciou o sume v mene, ktorú každodenne využíva, ako aj v menej danej krajiny.

5.2.2 Vytvorenie výdavku

Po otvorení konkrétnej cesty má používateľ možnosť vytvoriť výdavok na základe kategórií, ktoré aplikácia ponúka. Určité kategórie bližšie specifikujú konkrétny výdavok pre širšie uchovávanie informácií, ktoré používateľovi poskytnú lepšiu predstavu využitia výdavku pri výslednom reporte cesty.

Pri zaznamenávaní výdavku je taktiež vyhľadávaná poloha používateľa prostredníctvom internetu, aby sa zaznamenala adresa výdavku pre lepšiu informovanosť používateľa. Pri vytvorení výdavku je zároveň volaný vzialený server, ktorý hodnotu výdavku zamieňa do predvolenej meny na základe aktuálnej hodnoty kurzu.

5.2.3 Informovanie o aktuálnom stave

Používateľ bude informovaný o aktuálnom stave rozpočtu a výdavkov prostredníctvom obrzovky, kde sa budú nachádzať údaje o aktuálnom stave rozpočtu v stanovených menách, priemernej sume, ktorú môže denne minúť či porovnaní platieb, aké množstvo financií platil v hotovosti alebo platobnou kartou (túto možnosť si môže zadať pri tvorbe výdavku).

5.2.4 Správa výdavkov

Používateľ je informovaný o vytvorených výdavkoch v podobe zoznamu, ktorý zobrazuje základné informácie o daných výdavkoch – názov, suma, kategória, spôsob platby. Taktiež má možnosť štatistiky, ktorá zobrazuje, aký objem výdavkov bol použitý na konkrétnu kategóriu. Poslednou možnosťou pri správe výdavkov je možnosť zobrazenia výdavkov na základe uchovanej GPS polohy prostredníctvom máp Google Maps. Používateľ tak má ďalší vizuálny prostriedok pre lepší prehľad kde sa pohyboval a na akých miestach svoj výdavok uskutočnil.

V priebehu cesty má používateľ aj možnosť exportovať svoju cestu spolu so všetkými výdavkami do PDF súboru, ktorý sa automaticky uloží do vytvoreného priečinka v súborovom systéme zariadenia. Vďaka tejto možnosti je používateľ schopný uchovať si informácie o vytvorenej ceste aj v inom formáte, resp. zdieľať tieto informácie aj s inými osobami.

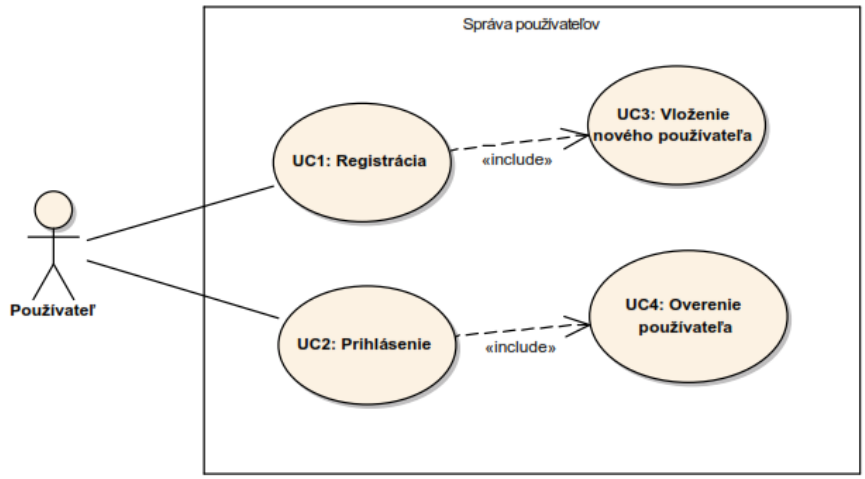
5.3 Schéma aplikácie

5.3.1 Diagram prípadov použitia

Diagram prípadov použitia opisuje interakciu používateľa so systémom. Scenáre prípadov použitia sú opísané v prílohe tejto práce (Príloha II).

A. User management

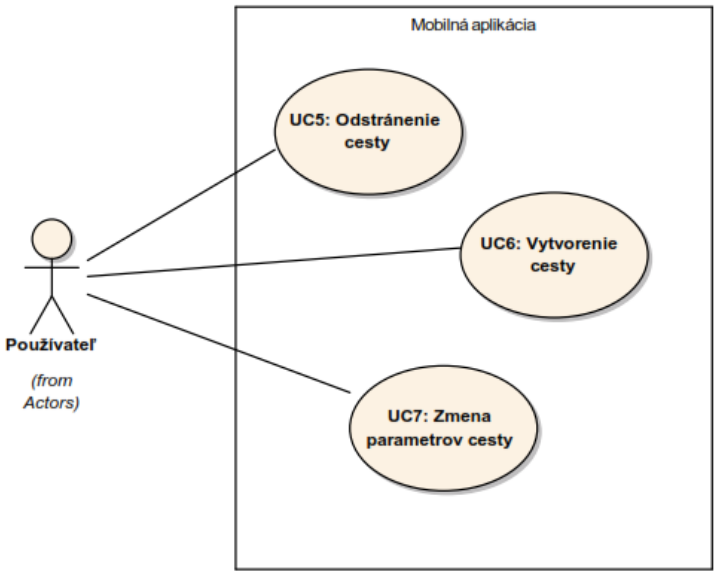
Používateľ má niekoľko základných možností pre správu aplikácie. Po nainštalovaní aplikácie je používateľ povinný zaregistrovať sa pred samotným prihlásením sa do nej. Pri procese registrácie musí byť používateľ pripojený na internet, pretože dáta sú odosielané na aplikačný server, ktorý spravuje komunikáciu s databázovým serverom. Tieto údaje bude mať používateľ stále k dispozícii prostredníctvom nastavení, kde si okrem prihlasovacích údajov môže zmeniť aj prednastavenú menu.



Obrázok 16: Základná funkcionalita aplikácie z pohľadu používateľa

B. **Trip management**

Základným prvkom aplikácie je vytváranie rôznych výletov. Každý výlet má niekoľko parametrov, ktoré si môže používateľ v ľubovolnom čase meniť.

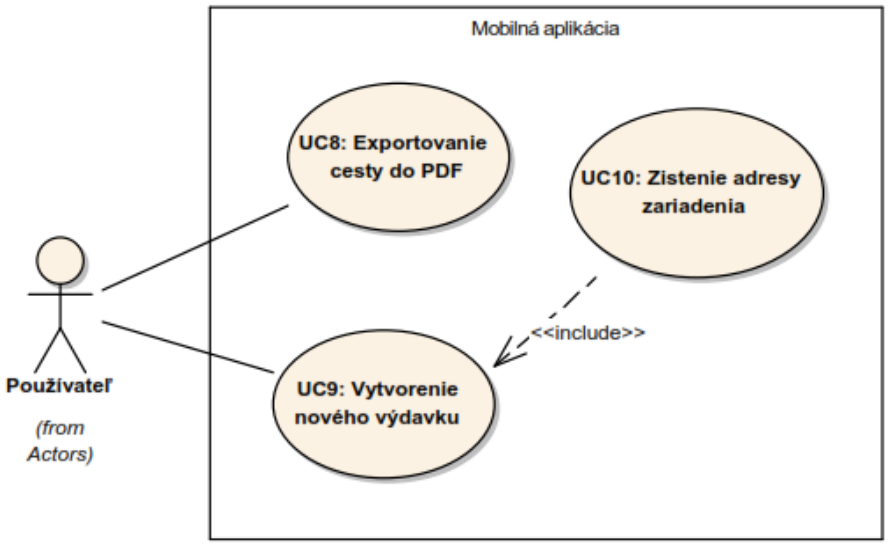


Obrázok 17: Prípady použitia pri správe výletu

- **Odstránenie výletu** – z databázy odstráni okrem výletu aj všetky existujúce výdavky spojené s výletom.
- **Zmena parametrov výletu** – používateľ môže upraviť niektoré parametre výletu (dátum, názov, rozpočet).
- **Správa výletu** – používateľ sa dostane do hlavného menu aplikácie, kde môže pracovať s výdavkami.
- **Vytvorenie výdavku** – používateľ môže vytvoriť nový výlet podľa predlohy.

C. **Selected trip**

Po tom, čo si používateľ vyberie jeden výlet zo zoznamu, automaticky sa mu otvorí výlet so všetkými existujúcimi parametrami a štatistikami. Základnou funkcionalitou je pridávanie nových výdavkov. Pri tejto operácií sa používateľovi zobrazí určitý počet kategórií, ktorý špecifikuje druh výdavku. Po vybratí kategórie sa následne zobrazí formulár na vyplnenie výdavku pre príslušnú kategóriu.



Obrázok 18: Prípady použitia pri vybranom výlete používateľom

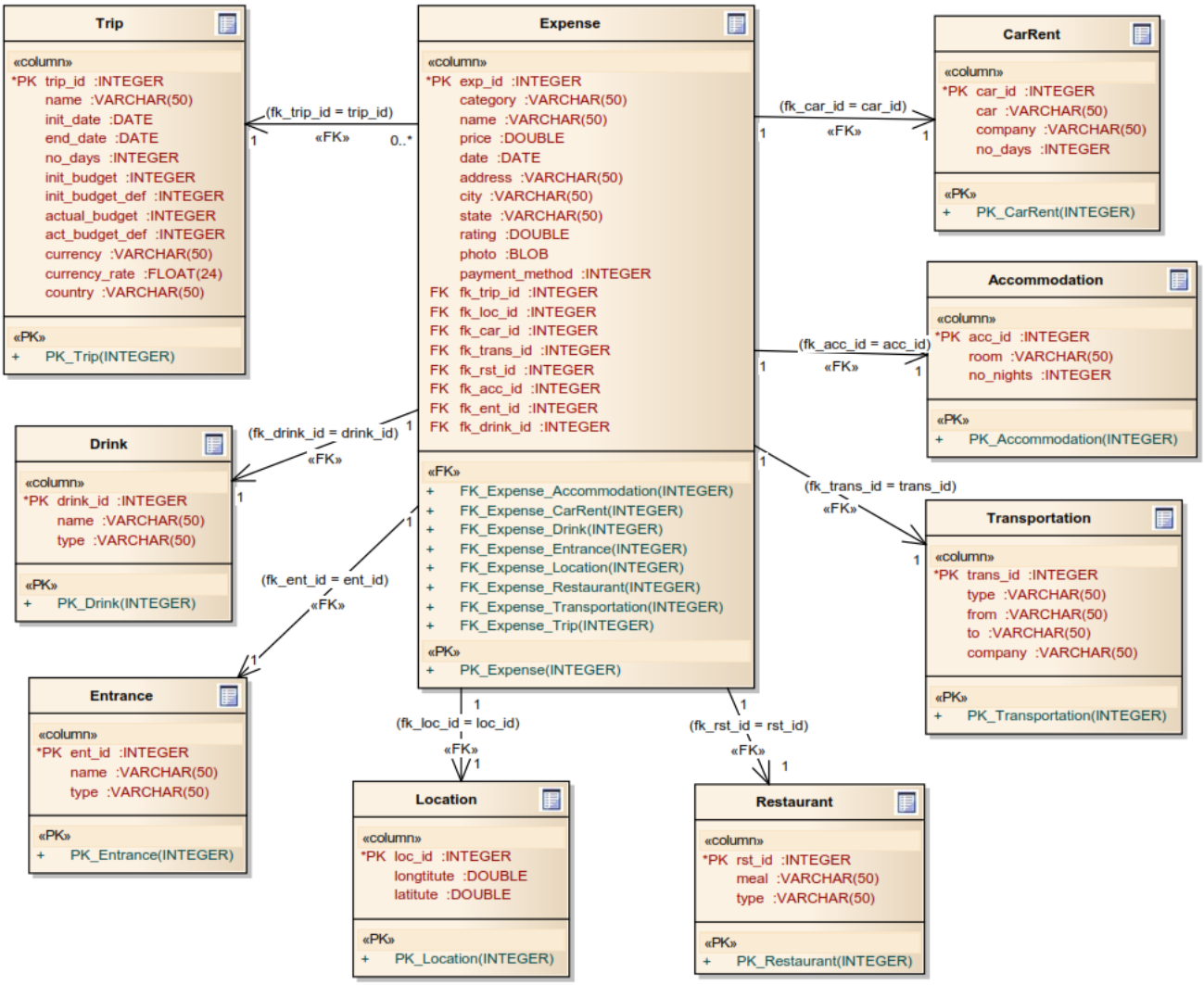
Súčasne s tým sa na pozadí vyhľadá poloha zariadenia a určí sa aktuálna adresa, kde sa používateľ so zariadením nachádza s aktuálnym dátumom. Po vyplnení všetkých potrebných náležitostí má používateľ možnosť uložiť daný výdavok. Táto operácia rovnako prebieha na pozadí aplikácie a po úspešnom uložení je používateľ informovaný.

5.3.2 Entitne – relačný diagram

Entitne-relačný model predstavuje model databázy, ktorá je vytvorená v mobilnom zariadení. V tejto databáze sú uložené všetky informácie o existujúcich výletoch, ako aj všetkých výdavkoch k nim prislúchajúcich. Predstavujú ich dve hlavné tabuľky databázy:

- Trip (cesta) – tabuľka uchováva základné informácie o vytvorených cestách ako trvanie cesty, krajina (automaticky nastavená mena krajiny), vstupný a aktuálny rozpočet.
- Expense (výdavok) – tabuľka zaznamenáva všetky informácie, ktoré sa týkajú výdavkov. Tieto údaje sa následne využívajú na zobrazenie aktuálneho stavu cesty (vzhľadom na financie) či spracovanie do štatistického súboru.
- Location (poloha) – tabuľka uchováva informácie o zemepisnej šírke a dĺžke výletu pre následné zobrazenie využitím nástroja Google Maps API.

Ostatné tabuľky *Drink, Entrance, Restaurant, Transportation, Accommodation, CarRent* uchovávajú dodatočné informácie spadajúce len pre danú kategóriu výdavku, napr. CarRent (požičanie auta) uchováva informácie o vozidle, spoločnosti a počte dní prenájmu; Restaurant (reštaurácia) uchováva jedlo a typ reštaurácie.



Obrázok 19: Entito-relačný model databázy

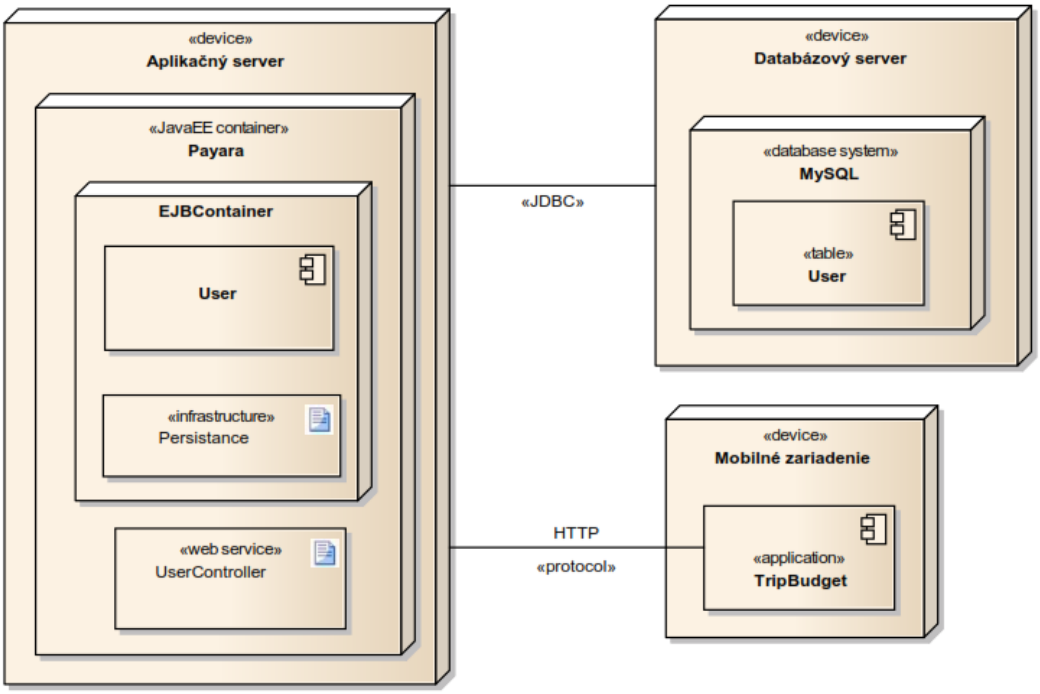
Keďže každý výdavok je špecifikovaný konkrétnou kategóriou, kardinalita databázového modelu predstavuje 1:1 medzi týmito tabuľkami. To znamená, že každý výdavok patrí len do jednej kategórie (resp. má len jeden záznam polohy) a zároveň každá špecifikácia výdavku v danej kategórií opisuje jeden špecifický výdavok.

Rozdielom je len vzťah *Trip – Expense*, kde každá cesta môže obsahovať ľubovoľné množstvo výdavkov, ale každý výdavok patrí práve pod jednu cestu, to znamená kardinalitu 1:N.

5.3.3 Diagram nasadenia

Diagram nasadenia opisuje rozloženie jednotlivých softvérových komponentov na hardvérových zdrojoch. V konkrétnom prípade sa systém delí na aplikačný server, databázový server a mobilné zariadenie, na ktorom operuje vyvýjaná android-ová aplikácia.

Mobilné zariadenie komunikuje prostredníctvom REST-ových volaní s aplikáciou na aplikačnom serveri Payara. Tá obhospodaruje databázový server prostredníctvom programovateľného interface-u (API) JDBC v jazyku Java. Databázový server pracuje na systéme MySQL, kde uchováva dáta v tabuľke používateľov.



Obrázok 20: Diagram nasadenia opisujúci štruktúru systému

Pri dopytovaní s databázou definuje aplikačný server model, ktorý kopíruje štruktúru databázovej tabuľky. Tento model je využívaný pri mapovaní dopytov do konkrétnej štruktúry využívaním objektovo-relačného mapovania. *UserController* opisuje webový servis, ktorý príjma a spracuváva požiadavky z mobilnej aplikácie, ako aj pripravuje odpovede na základe danej žiadosti.

6 REALIZÁCIA APLIKÁCIE

Táto kapitola opisuje návrh aplikácie z funkcionálneho aspektu, používateľského rozhrania a využitých technológií pri vývoji aplikácie.

6.1 Použité technológie

V nasledujúcej kapitole sa bližšie pozrieme na technológie, ktoré boli využité pri vývoji všetkých súčastí systému. Využitý aplikačný a databázový server boli inštalované na laptop pre testovacie účely.

6.1.1 Výber operačného systému

Mobilné zariadenia môžu pracovať na rôznych operačných systémoch. Medzi tie najrozšírenejšie patria Android a iOS, ktoré sú vyvíjané v jazyku Java, resp. Objective C. Vzhľadom k tomu, že podiel na trhu zariadení s operačným systémom Android je väčšinový[14], existuje širšia dostupnosť zariadení s týmto operačným systémom, preto rozhodnutie padlo na vývoj natívnej aplikácie pre tento systém.

6.1.2 Android API

Aby bola vybraná platforma operačného systému pre vývoj aplikácie čo najpresnejšia, je potrebné oboznámiť sa s úrovňou API (ang. Application Programming Interface) systému Android. API predstavuje sadu nástrojov (framework), ktorá sa využíva na komunikáciu aplikácie s ponúkanou funkcionalitou zariadenia. To znamená, že čím vyššiu verziu API operačný systém podporuje, tým väčšiu sadu nástrojov a funkcionality je možné pri vývoji aplikácie použiť. [14]

Na druhej strane, nie každé zariadenie vie s existujúcou sadou pracovať. Čím novší operačný systém, tým menšie množstvo zariadení danú sadu podporuje. Tento fakt sa odrazí na skutočnosti, že pri vývoji aplikácie vo verzii 27 (aktuálne najnovší operačný systém Oreo 8.1) by bolo schopných podporovať aplikáciu len 7,5% všetkých zariadení[15].

[14] Zdroj: http://gs.statcounter.com/os-market-share/mobile/worldwide
[15] Zdroj: https://developer.android.com/about/dashboards

Preto musí konečný výber API brať do úvahy dva faktory:

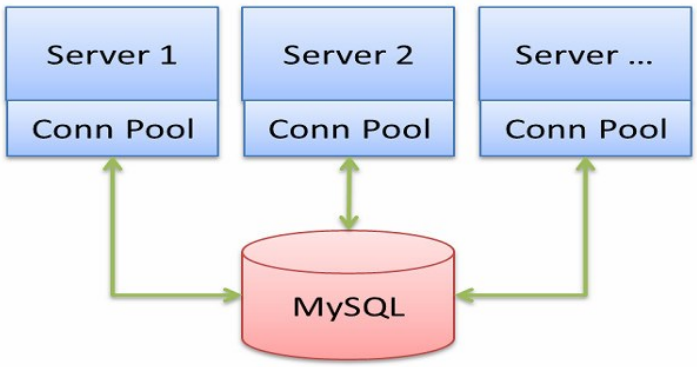
1) využívanú sadu nástrojov a teda komplexnosť riešenia,
2) rozsah zariadení, ktoré môžu aplikáciu využívať.

Ako bolo opísané v kapitole 4.4.1, pri vývoji aplikácie pre Android je dôležité vybrať správnu verziu API. Vzhľadom na štatistiky využívania verzií operačných systémov Android, až 88,9% zariadení využíva API úrovne 21 a vyššie. Práve pre možnosť uplatnenia aplikácie takéhoto rozsahu aplikácie bola zvolená daná úroveň API.

6.1.3 MySQL

MySQL predstavuje otvorený SQL relačný databázový server, ktorý je využívaný na správu dát. V kontexte aplikácie je využívaný na uchovávanie používateľských dát ako sú meno, heslo a email. Komunikuje s aplikačným serverom prostredníctvom dopytovacieho jazyka *SQL* na základe systému požiadavka - odpoveď. Aby mohla táto komunikácia prebiehať, využíva sa *JDBC konektor*.

Základom je využívanie rovnakého konektora so serverom, prostredníctvom ktorého je možné vytvoriť tzv. *connection pool*.



Obrázok 21: Zobrazenie princípu Connection pool. [24]

Ten predovšetkým slúži na zvyšovanie výkonu dopytov na databázu. Využíva sa hlavne pri webových aplikáciách, kde prebieha veľké množstvo dopytov v reálnom čase. Aby sa dopytujúci nemusel opätovne autorizovať, po prvom nadviazaní spojenia sú jeho údaje uložené do *pool*-u (bazéna). Následne, pri ďalších dopytoch, je už spojenie vytvorené, čo dopytujúcemu šetrí čas volania.

6.1.4 Payara

Aplikačný server slúži ako komunikačná brána medzi klientom a serverom (webový, databázový). Obhospodaruje prichádzajúcu komunikáciu a na základe existujúcej funkcionality požiadavky spracúva a preposiela do cieľovej stanice. Výhodou využívania aplikačného servera je možnosť spravovať aktuálnu komunikáciu a balancovať dopyty k serverom. K aplikačnému serveru sa spája aj pojem bezpečnosti. Keďže aplikačný server slúži ako brána, napr. pri databázovom servery spracúva dopyty od klienta a zostavuje konkrérne SQL dopyty na databázový server. Takýmto spôsobom je databázový server chránený voči útoku SQL injection. Aplikačný server teda predstavuje centrálnu jednotku, ktorá sa stará o bezpečnú komunikáciu s koncovým serverom a zároveň monitorovaniu dostupnosti a vyvažovaniu záťaže.

6.1.5 Zoznam vývojových prostredí a nástrojov

Vývojové prostredia:

- Android Studio 3.3.2,
- NetBeans IDE 8.2.

Aplikačný server:

- Payara 5.

Databázový server

- MySQL 8.0.15 (Community server – GPL),
- JDBC Connector J 8.0.15.

Java:

- JRE 1.8.0.

6.1.6 Zoznam využitých externých knižníc

HTTP klient pre Android a Java aplikácie:

- com.squareup.okhttp3.okhttp:3.11.0

Nástroj na vystvorenie PDF súborov:

- com.itextpdf:itexdtg:5.5.10

Grafické knižnice:

- de.hdodenhof:circleimageview:2.2.0
- com.github.lzyzsd:circleprogress:1.2.1
- com.github.hadiidbouk:ChartProgressBar-Android:2.0.6

Komunikácia s aplikačným serverom prostredníctvom HTTP:

- com.squareup.retrofit2:retrofit:2.5.0
- com.sqeareup.retrofit2:converter-gson

Zoznam krajín a príslušnej meny vo formáte JSON:

- https://github.com/samayo/country-json

6.2 Vlastnosti aplikácie

Program využíva vo veľkej miere internetové pripojenie. Z tohoto dôvodu je potrebné, aby pri využívaní aplikácie bolo zadiadenie konštatne pripojené k mobilným dátam alebo WiFi.

6.2.1 HTTP požiadavky

Aplikácia zaznamenáva pre používateľa hodnotu výdavkov v mene nastavenej aplikáciou a v mene cesty. Na konverziu aktuálneho kurzu sa využíva Currency Converter API[16], ktorý je dopytovaný prostredníctvom HTTP požiadavky. Formát požiadavky predstavuje dopytovanú HTTP lokalitu, meny použité pri konverzii a API kľúč:

```
https://free.currconv.com/api/v7/convert?q=EUR_CZK&compact&ultra&apiKey= api_kľúč
```

Odpoveďou je jednoduchá forma JSON formátu:

```
{
“EUR_CZK”: 25.741
}
```

6.2.2 Využitie JSON formátu

Pri vytváraní cesty má používateľ možnosť zadať krajinu, kde cestuje a na základe ktorej sa určí mena danej krajiny. Vzhľadom na to, že zoznam počtu krajín a následne priradenie

[16] Zdroj: https://free.currencyconverterapi.com

meny obsahuje väčší počet prvkov, najjednoduchším riešením je využitie JSON[17] formátu, ktorý uchováva hodnoty v štruktúre „meno“:“hodnota“.

```
{
    "list": [
        {
            "country": "Afghanistan",
            "currency_code": "AFN"
        },
        {
            "country": "Albania",
            "currency_code": "ALL"
        },
        {
            "country": "Algeria",
            "currency_code": "DZD"
        },
        {
            "country": "American Samoa",
            "currency_code": "USD"
        },
}
```

Obrázok 22: Zobrazenie zoznamu krajín v JSON formáte

Aby aplikácia mohla pracovať s prvkami uloženými v súbore, je potrebné spracovať jeho štruktúru, tzv. parsovať. Prvým krokom je načítanie príslušného súboru do JSONObject z uloženého súboru, ktorý je nasledovaný prehliadaním všetkých hodnôt súboru v cykle na základe jeho štruktúry. Získané hodnoty sa následne ukladajú do listu, ktorý obsahuje inštancie triedy Country.

[17] Viac na: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/

```
try {
            JSONObject obj = new JSONObject(loadJSONFromAsset
            ("countryByCurrencyCode.json", mContext));
            JSONArray m_jArry = obj.getJSONArray("list");

            for (int i = 0; i < m_jArry.length(); i++) {
                JSONObject jo_inside = m_jArry.getJSONObject(i);
                String country_name = jo_inside.getString("country");
                String currency_code = jo_inside.getString("currency_code");

                Country c = new Country(country_name, currency_code, null);
                countries.add(c);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
```

Obrázok 23: Ukážka zdrojového kódu spracovania súboru v JSON formáte

6.2.3 Lokalizácia zariadenia

Každý výdavok, ktorý používateľ vloží je lokalizovaný prostredníctvom internetu na základe WiFi pripojenia alebo celulárnej siete. Aby bola aplikácia schopná komunikovať s internetom, je potrebné povoliť toto využívanie v súbore Manifest.xml:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Obrázok 24: Zoznam povolení pre zariadenie

Prostredníctvom internetu zariadenie dostane údaje o zemepisnej šírke a dĺžke (ang. longtitude/latitute), na základe ktorých sa získava adresa za pomoci triedy Geocoder:

```
try {
                addresses = geocoder.getFromLocation(lati, longi, 1);
                String city;

                String address = addresses.get(0).getAddressLine(0);
                List<String> fullAddress = Arrays.asList(address.split(","));
                if (addresses.get(0).getLocality() == null)
                    city = addresses.get(0).getAdminArea();
                else
                    city = addresses.get(0).getLocality();
                String state = addresses.get(0).getAdminArea();
                String country = addresses.get(0).getCountryName();
                String postalCode = addresses.get(0).getPostalCode();

                addressLocation.setFullAddress(fullAddress.get(0));
                addressLocation.setCity(city);
                addressLocation.setCountry(country);
                addressLocation.setLongtitute(longi);
                addressLocation.setLatitute(lati);

            } catch (IOException e1) {
                e1.printStackTrace();
            }
```

Obrázok 25: Ukážka zdrojového kódu pre vyhľadanie polohy zariadenia

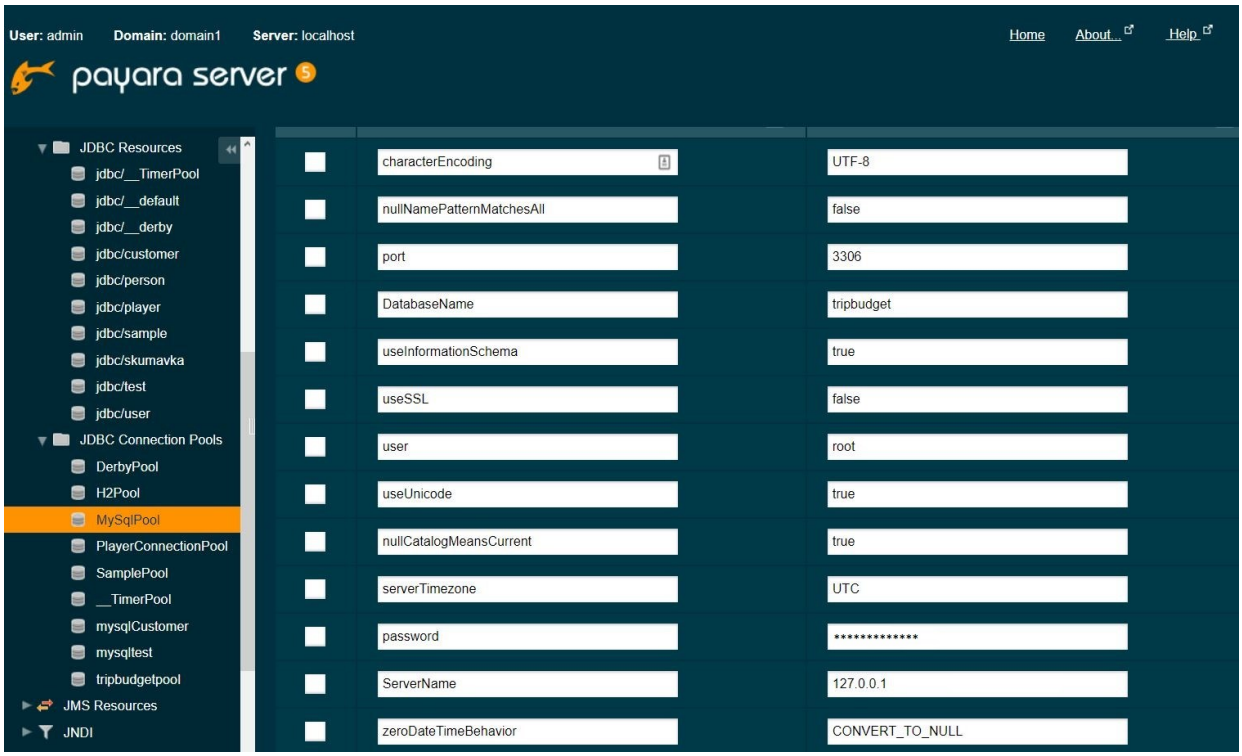
Získané údaje sa ukladajú do inštancie triedy AddressLocation, ktorá sa následne využíva prostredníctvom listener-u pri zobrazovaní adresy používateľovi, resp. ukladaní informácie do databázy.

6.2.4 Komunikácia s aplikačným a databázovým serverom

Pre správu používateľov mobilnej aplikácie je využívaný databázový server, ktorý uchováva *username, password* a *email* používateľa. Vzhľadom k tomu, že priama komunikácia medzi mobilnou aplikáciou a databázovým serverom môže byť vystavená hrozbám útoku (napr. SQL injekcia), využíva sa aplikačný server ako prostredník medzi dopytom a odpoveďou.

Aby aplikačný server mohol prevádzkovať potrebnú službu, musí byť prepojený s databázovým serverom. Na to slúži *JDBC Connection Pools* a *JDBC Resources*:

1) *JDBC Connection Pools* uchováva všetky potrebné nastavenia databázy pre vytvorenie komunikácie. Najdôležitejšími parametrami sú databázové meno, používateľ, používateľské heslo, meno servera (IP adresa) a port, na ktorom komunikuje. Na základe týchto údajov je možné potvrdiť spojenie prostredníctvom *ping*-u.



Obrázok 26: Zobrazenie nastavení aplikačného servera pre komunikáciu s databázou

2) *JDBC Resource* predstavuje prepojenie s nastaveným *connection pool.* Zároveň sa využíva pri vývoji webového servisu pre aplikačný server, kde sa do nastavení ukladá názov *JDBC Resource* pre identifikáciu *connection pool.*

Po nastavení komunikácie databázového a aplikačného servera prostredníctvom rozhrania je potrebné vytvoriť serverovú časť, ktorá bude komunikovať s klientom. Komunikácia môže prebiehať napríklad prostredníctvom *SOAP* alebo *REST* volaní. Pre realizáciu tohoto projektu bola využitá REST-ová komunikácia medzi klientom a serverom.

REST predstavuje architektúru, ktorá umožňuje pristupovať k dátam prostredníctvom metód HTTP. Ponúka 4 základné metódy známe ako CRUD– *create* (vytvoriť), *retrieve* (čítať), *update* (zmena), *delete* (zmazanie). Pre prístup k zdroju sa využívajú v podobe *GET*, *POST*, *PUT*, *DELETE*:

1) **GET** slúži sa získanie údajov. Môže byť kombinovaný so vstupnými parametrami, napr. získanie používateľa s ID = 1: GET /user/1.
2) **POST** volá službu pre uloženie nového prvku. Nevolá pri tom konkrétny element ako môže volať GET alebo PUT metóda.

3) **PUT** operácia predstavuje zmenu konkrétneho prvku zdroja. Narozdiel od POST volá konkrétny prvok, pričom v sebe uchováva novú hodnotu, ktorá sa zmení na základe parametra.
4) **DELETE** slúži na vymazanie daného prvku podľa vstupného parametra.

Pre REST komunikáciu môže byť využívaný formát správ v podobe XML alebo JSON:

1) **XML** predstavuje značkovací jazyk, ktorý využíva značky (tagy) pre vytvorenie štruktúry:

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

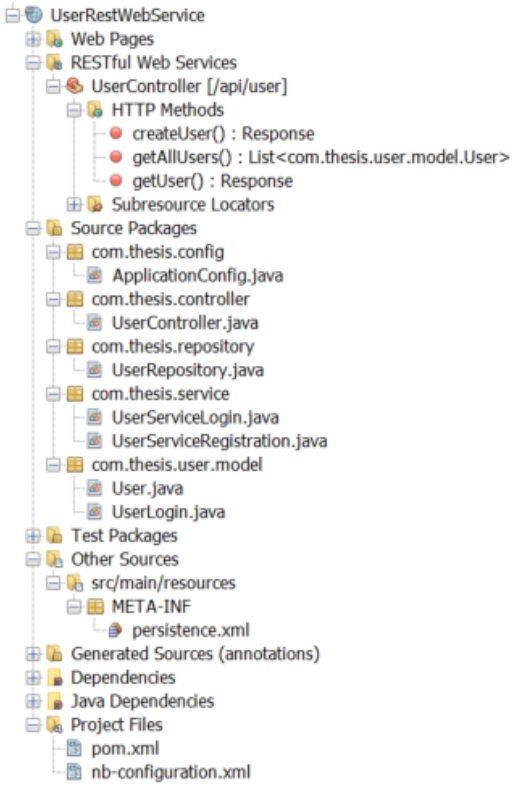
Obrázok 27: Zobrazenie údajov vo formáte XML

2) **JSON** formát je spomenutý spolu s ukážkou v kapitole 7.2.3.

Pre aplikáciu bola zvolená komunikácia správ vo forme JSON, ktorá je pre spracovanie jednoduchšia a komplexnejšie podporovaná.

6.2.5 Web servis

Web servise je nasadený na aplikačnom servery a slúži na komunikáciu s klientom. Opisuje REST metódy, ktoré vie prijať od klienta v požadovanom formáte, následne ich spracúva a na základe daného dopytu komunikuje s databázovým serverom. Je vyvíjaný prostredníctvom jazyka Java a nasadený na server vo formáte *.war*.



Obrázok 28: Súborová štruktúra aplikácie nasadenej na aplikačný server

Aplikácia web servisu sa skladá z niekoľkých častí. Hlavnými sú systémové súbory ***pom.xml*** a ***persistance.xml***, kde sa nastavujú parametre, nastavenia a závislosti, ktoré sú potrebné pre správne fungovanie aplikácie. Definuje sa tu napríklad aj **jdbc/user**, čo predstavuje JDBC Resource pre komunikáciu s databázou.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  <persistence-unit name="MYSQL_PU" transaction-type="JTA">
    <jta-data-source>jdbc/user</jta-data-source>
    <class>com.thesis.user.model.User</class>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.schema-generation.database.action" value="create"/>
    </properties>
  </persistence-unit>
</persistence>
```

Obrázok 29: Zdrojový kód pre aplikačný server pre komunikáciu s databázou

Podľa súborového systému môžeme vidieť, že je aplikácia rozdelená logicky na niekoľko častí. Balík *com.thesis.user.model* obsahuje triedu, ktorá uchováva model a ten korešponduje s databázovou tabuľkou na serveri. Tento model poskytuje formu, ktorú napĺňajú dáta dopytované z databázy. *UserController.java* opisuje formát dopytov a metód, ktoré môže klient využívať pre komunikáciu s databázovým serverom. Tu sa definujú metódy *GET* a *POST* spolu s formátom cesty, prípadne parametrami pre úspešný dopyt.

Na obrázku môžeme vidieť metódu *POST*, ktorá okrem informácie o type metódy obsahuje anotácie ***@Path*** a ***@Produces***. Prvá menovaná špecifikuje cestu metódy, URI adresu, ktorá musí byť zadaná pre REST dopyt. Druhá anotácia hovorí o formáte, v ktorom vráti odpoveď klientovi. Ako bolo spomínané vyššie, je možné vybrať medzi XML a JSON formátom, v konkrétnom prípade bol vybratý JSON formát. Telo metódy zabezpečuje prijatie prvku, ktorý je následne spracovaný a podľa poskytnutej odpovede z databázy a následnom spracovaní je odoslaná konkrétna odpoveď späť klientovi.

```
@POST
@Path("/login")
@Produces("application/json")
public Response getUser(User user){
    List<User> userResponse = userRepository.findLogin(user.getUsername());

    Object o = userResponse.get(0);
    Object[] obj = (Object[]) o;
    String password = (String) obj[1];

    if(userResponse == null){
        return Response.status(Response.Status.NOT_FOUND).build();
    } else {
        boolean found = serviceLogin.chceckPassword(password, user.getPassword());

        if (found)
            return Response.ok(user).build();
        else
            return Response.status(Response.Status.NOT_FOUND).build();

    }
}
```

Obrázok 30: Zdrojový kód pre aplikačný server spracúvajúci REST volanie

Ďalšou časťou aplikácie je *UserRepositorry.class*, ktorá vytvára dopyty s databázovým serverom. Okrem toho, že táto trieda využíva SQL query, zároveň využíva Java Persistance API (JPA), čo predstavuje mapovanie Java objektov na databázu ORM (Object-relation mapping).

```
@Transactional(REQUIRED)
public void create(User user){
    em.persist(user);
}
```

Obrázok 31: Zdrojový kód implementácie pre ukladanie elementu do databázy

Namiesto skladania SQL dopytov sa využívajú vstavané implementácie (napr. Hibernate, EclipseLink), ktoré zabezpečujú komunikáciu na základe objektov pri ich zasielaní ale aj prijatí – využívanie objektu triedy *User.java*.

```
public List<User> findLogin(String username){

    String stringQuery = ("select u FROM User u WHERE u.username= :username");
    TypedQuery<User> query = em.createQuery(stringQuery, User.class);
    query.setParameter("username", username);
    List<User> resultList = query.getResultList();
    return resultList;
}
```

Obrázok 32: Zdrojový kód pre vyhľadanie používateľa pomocou SQL dopytu

Na obrázku môžeme vidieť vloženie používateľa do tabuľky prostredníctvom využitia JPA, bez využívania SQL dopytu. Vstupný parameter je objekt formátu User. Anotácia ***@Transactional*** predstavuje atomickú operáciu, čo znamená, že vykonávaná akcia musí byť úspešne ukončená zápisom do databázy (committed) alebo úplne vrátená (rolled back).

6.3 Opis používateľského prostredia

Používateľské prostredie aplikácie sa skladá z dvoch hlavných častí:

- správa cesty,
- správa výdavkov.

6.3.1 Správa cesty

Používateľ môže spravovať cesty ich vytváraním, úpravou parametrov alebo odstránením cesty. Pri spustení aplikácie sa používateľovi zobrazia už existujúce cesty so zadanými parametrami ako sú *dátum začiatku a konca cesty, zostávajúci rozpočet (Actual), vstupný rozpočet (Budget), krajina*, ktorú navštívi a *názov* cesty.

Aplikácia zároveň uchováva informácie o dvoch menách – mena aplikácie a krajiny, do ktorej je cesta absolvovaná. Oba tieto údaje sú používateľovi k dispozícii, pre jednoduchšiu informovanosť. Konverzný kurz medzi danými menami je získavaný prostredníctvom volania pri vytvorení cesty.

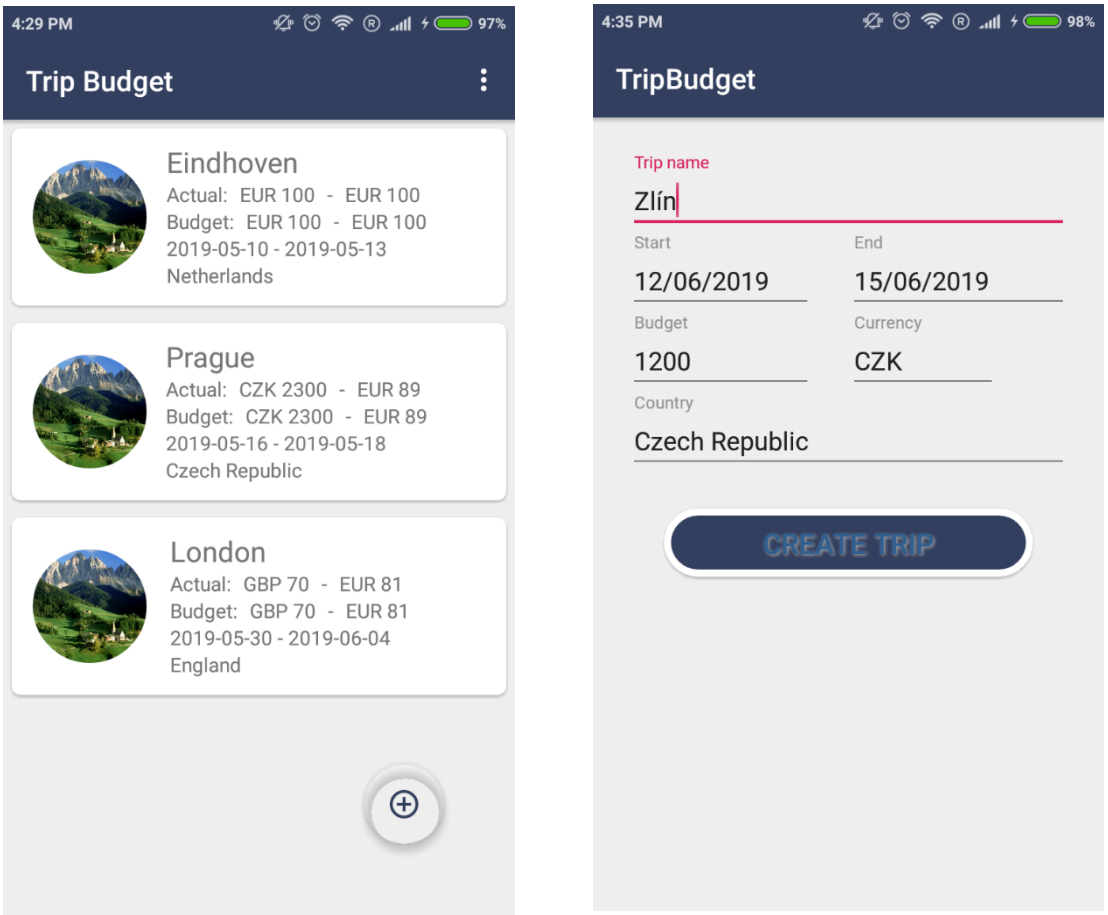
Pri úprave parametrov cesty má používateľ možnosť meniť názov, rozpočet a trvanie cesty, okrem krajiny, resp. meny prislúchajúcej krajiny. Pri zmene rozpočtu sa výsledok odrazí aj v používateľskom rozhraní. Takže v prípade navýšenia rozpočtu si používateľ jednoducho zmení jeho výšku.

6.3.2 Správa výdavkov

Väčšiu a hlavnú časť aplikácie tvorí správa výdavkov vybranej cesty. Menu sa skladá z troch častí – *Balance (aktuálny stav), Map (mapa), Statistics (štatistika)*, ktorých prístup je v podobe kariet. Tento prístup bol zvolený preto, aby používateľ vykonával čo najmenej kliknutí na displej a obrazovky aplikácie sa mu kontinuálne zobrazovali. Posúvaním obrazovky doľava, resp. doprava, sa používateľ prakticky premiestňuje v celej aplikácií a dosahuje celú jej funkcionalitu.

Hlavné menu ponúka všetky základné informácie o ceste. Obrazovka je štruktúrovaná na 2 časti:

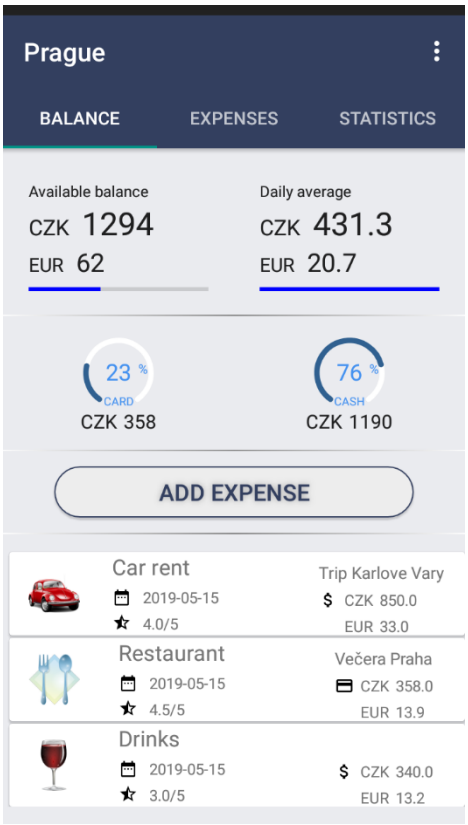
- číselné hodnoty o stave cesty,
- zoznam vytvorených výdavkov



Obrázok 33: Zobrazenie zoznamu ciest a vyplnenej predlohy pri vytvorení cesty

Číselné hodnoty sú rozdelené na dve časti. Horná polovica zobrazuje *available balance* (zostávajúci rozpočet) a *daily average*(priemerná suma, ktorú môže používateľ minúť za deň). Obe tieto hodnoty sú zobrazené ako v mene krajiny danej cesty, tak aj podľa meny aplikácie. Pod nimi sa nachádza *progress bar* (ukazovateľ postupu), t.j. grafické znázornenie, koľko používateľovi ostáva z celkového rozpočtu – modrá farba predstavuje dostupné zdroje.

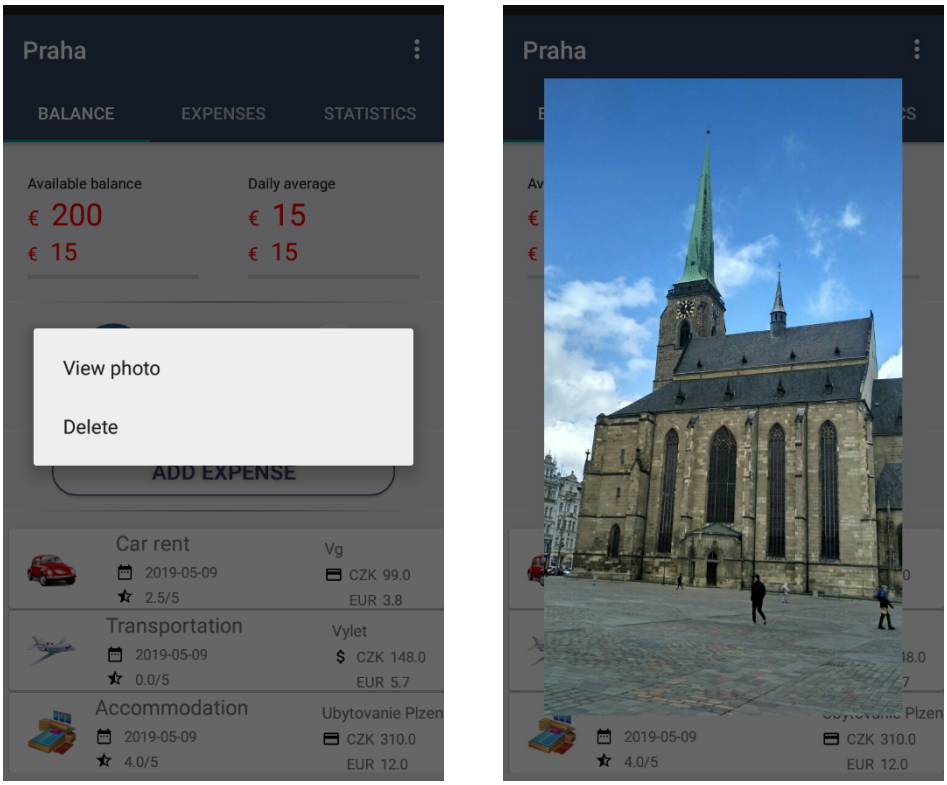
Dolná časť zobrazuje rovnako dva *progress bar-i*, ale v podobe kruhu. Používateľa informuje o percentuálnom podiele, koľko výdavkov zaplatil prostredníctvom platobnej karty, resp. hotovosťou. Zobrazuje taktiež sumu, ktorá bola na to vynaložená.



Obrázok 34: Zobrazenie hlavnej obrazovky aplikácie

Druhá časť obrazovky zobrazuje zoznam výdavkov, ktoré boli vytvorené. Výdavok poskytuje informáciu o *kategórií výdavku, dátume vytvorenia, hodnoteniu, názvu* a *ceny*. Používateľ má dve možnosti, ako pracovať s prvkom v zozname výdavkov. Dlhším kliknutím na výdavok sa mu zobrazia možnosti *View photo* (zobraziť fotku) a *Delete*

(Odstrániť). *View photo* predstavuje možnosť zobraziť fotku, ktorú používateľ uložil pri vytvorení výdavku (kapitola 7.3.5.) a *Delete* vymazanie daného výdavku z databázy.



Obrázok 35: Zobrazenie možnosti práce s výdavkom
a zobrazenie obrázku uloženého pre výdavok

6.3.3 Zobrazenie vytvorených výdavkov

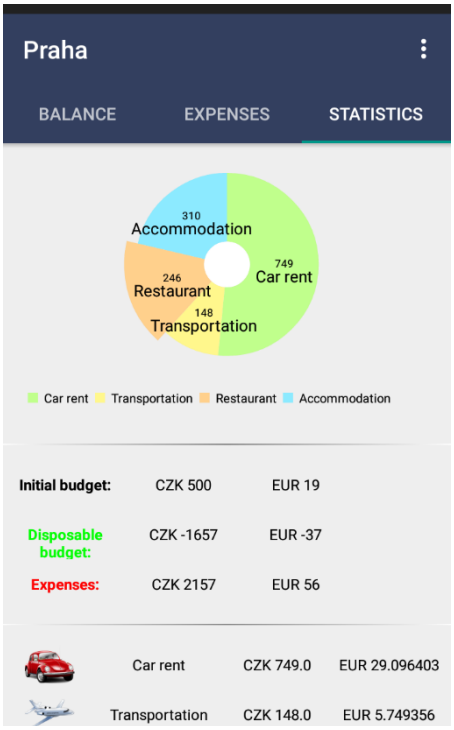
Posunom na kartu *Expenses Map* sa používateľovi zobra zí mapa vytvorených výdavkov podľa ich uloženej polohy. Kliknutím na *tag* (červený bod na obrazovke) sa používateľovi zobrazí názov výdavku a adresa miesta, kde bol vytvorený.

V pravom dolnom rohu sa nachádzajú ďalšie dve možnosti (funkcionalita Google Maps API), ktoré kliknutím na ne aplikáciu automaticky presmerujú na aplikáciu Google Maps nainštalovanú v zariadení a umožnia používateľovi navigovanie na miesto vytvorenia daného výdavku, resp. zobrazenie daného miesta v aplikácií.

Obrázok 36: Mapa výdavkov

6.3.4 Štatistika výdavkov

Posledná karta menu obsahuje štatistické informácie o vytvorených výdavkoch. Na obrazovke dominuje koláčový graf, ktorý zobrazuje rozdelenie vytvorených výdavkov na základe kategórií, spolu s celkovou sumou, ktorá bola na danú kategóriu minutá. Stredná

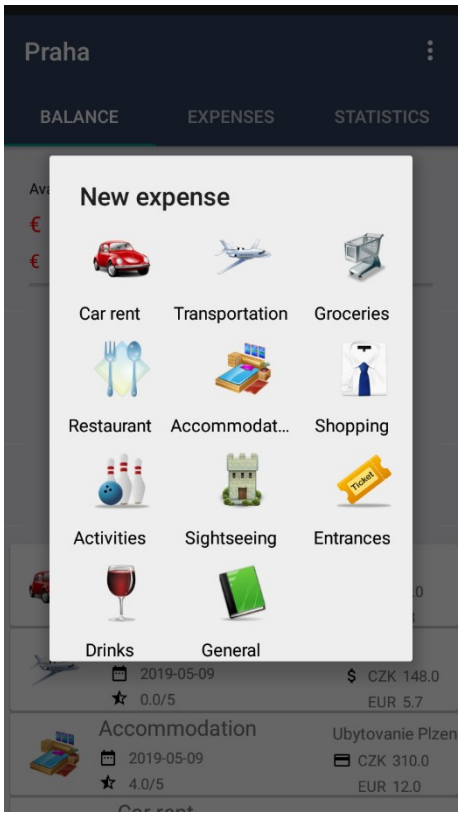


Obrázok 37: Obrazovka štatistiky

časť zobrazuje jednoduchý tabuľkový prehľad, ktorý zobrazuje informácie v oboch menách, a to konkrétne *initial budget* (vstupný rozpočet), *disposable budget* (dostupný rozpočet), *expenses* (výdavky). Posledná časť zobrazuje prehľad sumy výdavkov na základe kategórií.

6.3.5 Vytvorenie výdavku

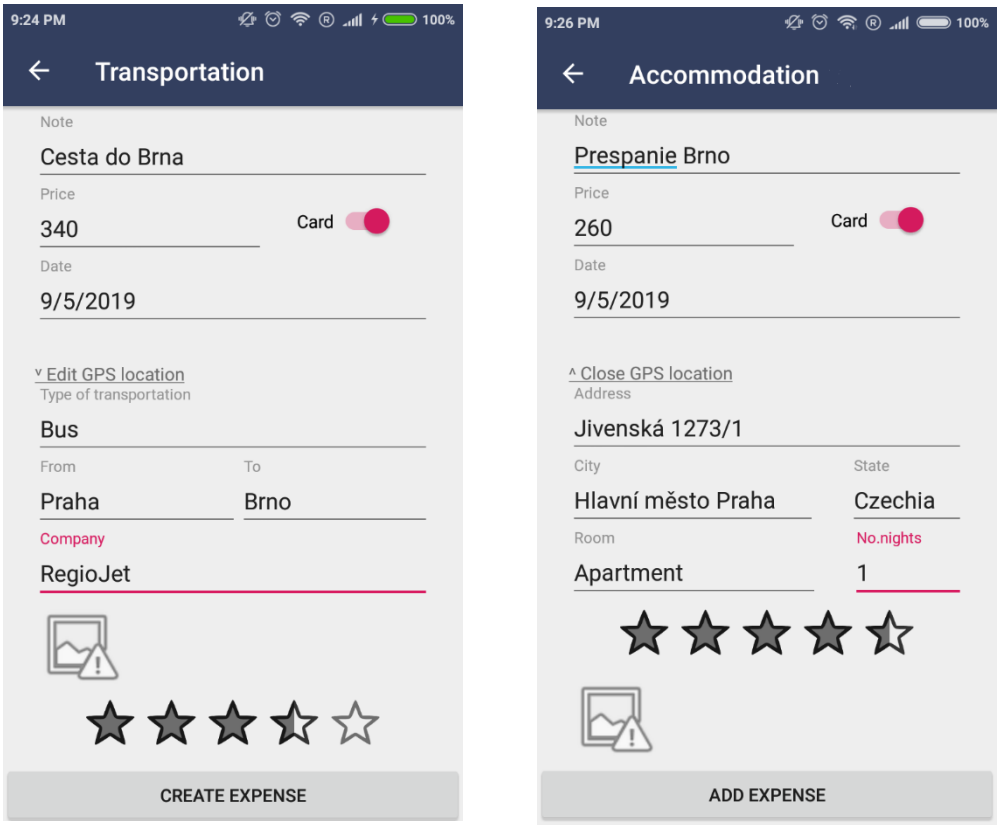
Používateľ vytvorí výdavok kliknutím na tlačidlo *Add expense* (pridať výdavok) na úvodnej karte obrazovky. Následne sa mu zobrazí dialógové okno s ponukou výdavku podľa kategórie. Po kliknutí na určitú kategóriu sa používateľovi zobrazí formulár výdavku. Ten sa líši podľa kategórie z dôvodu, že niekoré výdavky špecifikujú určité informácie pre lepšie zaznamenávanie výdavkov.



Obrázok 38: Zobrazenie ponuky výberu výdavkov

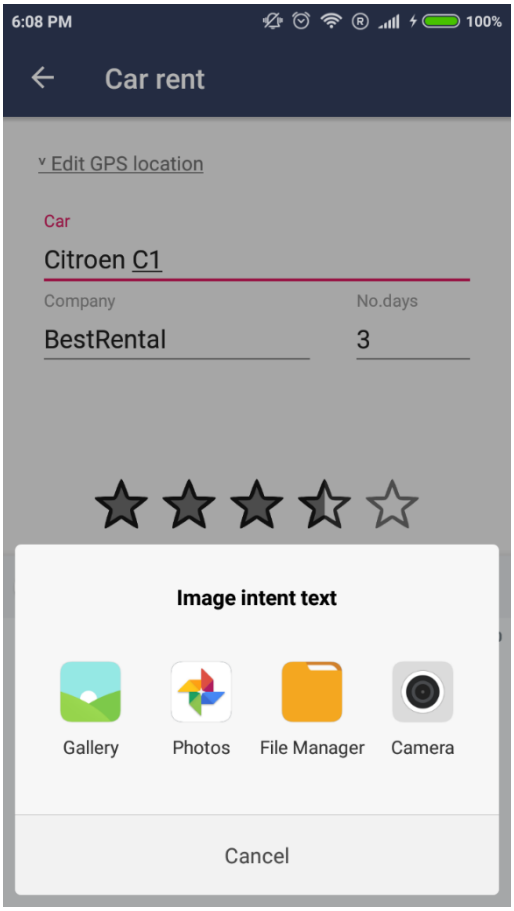
Každý výdavok ponúka možnosť zaznamenať základné informácie – *note* (poznámka), *price* (cena), *date* (dátum), *address* (adresa), *city* (mesto), *state* (krajina), *rating* (hodnotenie), *image* (obrázok). Niektoré procesy sú však v aplikácií automatizované. Dátum je automaticky zadaný na aktuálny deň, pričom ho používateľ môže zmeniť na ľubovoľný prostredníctvom nástroja *datePicker*. Ako bolo spomínané vyššie, aplikácia získava polohu

zariadenia na základe internetového spojenia, čo znamená, že adresa je automaticky vyplnená na základe GPS súradníc.



Obrázok 39: Zobrazenie vyplnenej predlohy výdavku podľa kategórie

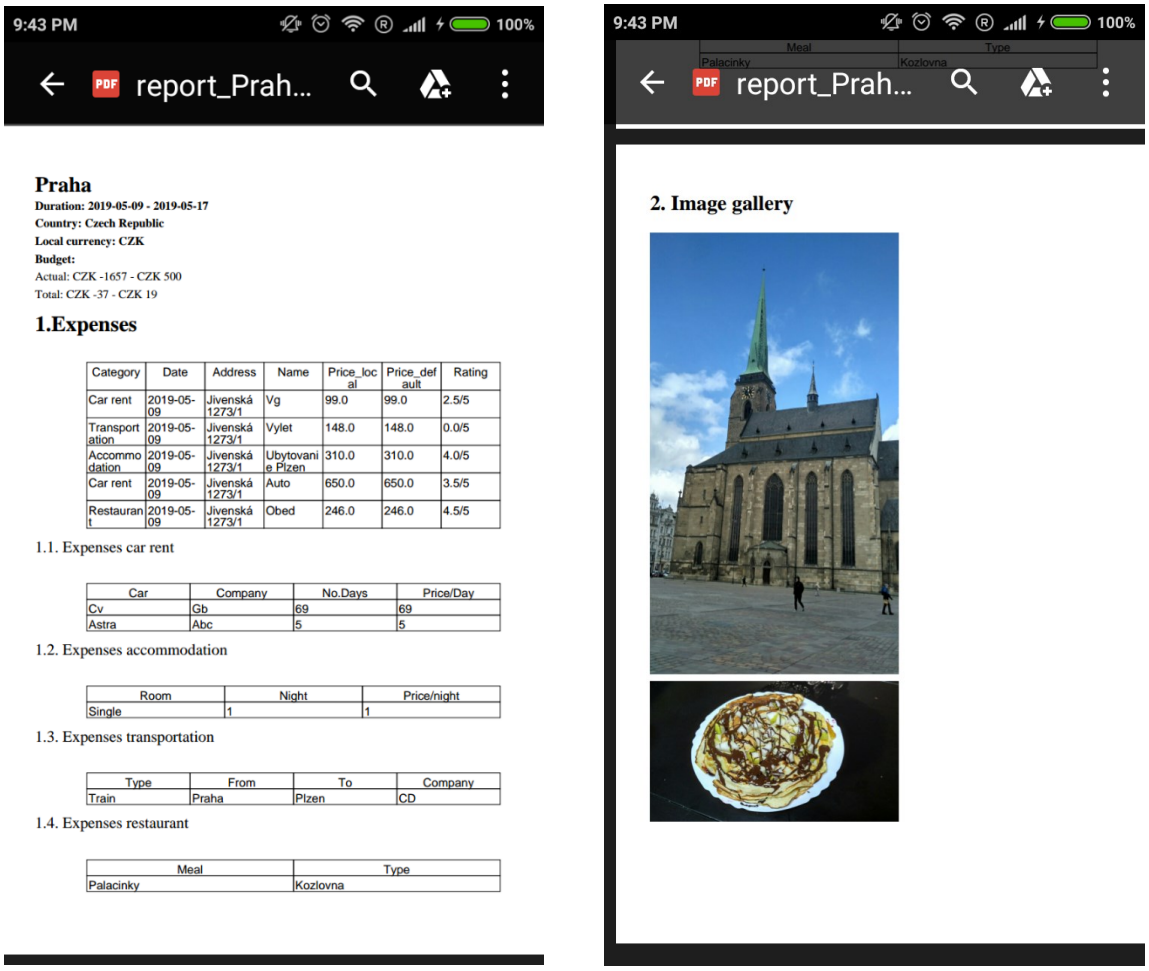
Používateľ má taktiež možnosť uloženia obrázku do aplikácie. K tomuto účelu slúži trieda *ImagePicker*, ktorá ponúka prístup ku galérií, správcovi súborov alebo ovládaniu fotoaparátu. Takto má používateľ možnosť vytvoriť obrázok v danom momente alebo si vybrať už existujúci z pamäte zariadenia.



Obrázok 40: Zobrazenie časti vyplneného výdavku spolu s možnosťou výberu obrázku

6.3.6 Exportovanie výletu

Jednou z funkcií aplikácie je aj export existujúcej cesty do PDF súboru, ktorý sa uloží do internej pamäte zariadenia, pričom si vytvorí vlastný adresár, kam vytvorený súbor ukladá. Vytvorený PDF súbor má programátorsky vytvorenú šablónu, podľa ktorej sa exportujú všetky potrebné informácie z aplikácie: *základné informácie o ceste*, *zoznam výdavkov cesty, detailný zoznam podľa kategórií, galéria obrázkov.*



Obrázok 41: Zobrazenie exportovaného PDF súboru cesty

6.4 Zabezpečenie aplikácie

Prihlásenie do aplikácie patrí medzi základné a najdôležitejšie bezpečnostné prvky. Používateľ sa do aplikácie prihlasuje prostredníctvom mena a hesla, ktoré uviedol pri registrácii. Aby heslo bolo dostatočne chránené v prípade neoprávneného získania, pri procese registrácie sa využíva šifrovanie hesla prostredníctvom *SHA*-256. Toto označenie predstavuje hashovaciu funkciu, ktorá vstupný reťazec zašifruje prostredníctvom 256 bitov.

Okrem zabezpečenia hesla je potrebné zabezpečiť aj prístup k databázovému serveru pre minimalizovanie úniku citlivých dát používateľov. Z tohoto dôvodu pri získavaní alebo overovaní prístupových údajov aplikácia nekomunikuje priamo s databázovým serverom, ale prostredníctvom webových servisov aplikačného serveru (bližšie opísané v kapitole 7.2.5).

ZÁVĚR

Táto záverečná práca bola zameraná na vývoj mobilnej aplikácie pre sledovanie a analýzu výdajov počas zahraničných ciest. Popri tomto vývoji sme sa v teoretickej časti práce bližšie oboznámili s témou bezpečnosti mobilných zariadení a opísali existujúce riešenia, ktoré pokrývajú danú problematiku.

Vybrali sme si tri mobilné aplikácie, ktoré sú voľne dostupné z online trhoviska pre operačný systém Android a pozreli sa na ponuku ich funkcionality spolu s grafickou stránkou, ako aj používateľským prostredím. Z porovnania týchto aplikácií vyšla najlepšie aplikácia TravelSpend, ktorá bodovala hlavne funkcionalitou a veľmi príjemným používateľským prostredím. Nevýhodou bola nutnosť zaplatenia poplatku pre sprístupnenie nadstavbových vlastností.

Po zhodnotení existujúcich riešení sme sa pozreli na hrozby a zabezpečenia mobilných zariadení. V súčasnosti je téma bezpečnosti zariadení pripájajúcich sa do Internetu stále dôležitejšia, preto sme si bližšie opísali typy hrozieb, ktorým zariadenia čelia. Oboznámili sme sa bližšie aj s princípom fungovania hrozieb ako sú phishing a man-in-the-middle, či hrozby úniku dát prostredníctvom SQL injekcie.

Okrem oboznámenia sa s hrozbami je dôležitou časťou aj spôsob zabezpečenia. Tie sme opísali v kapitole, ktorá obsahuje informácie o praktických radách, ako by mal používateľ pracovať so svojim zariadením a aké bezpečnostné pravidlá by mal dodržiavať (napr. neprihlasovať sa na nezabezpečenú WiFi). Opísali sme si aj spôsoby zabezpečenej komunikácie prostredníctvom využívania HTTPS protokolu alebo výhody využívania VPN. Pre bezpečnú autentizáciu sme si detailnejšie rozobrali prihlasovanie prostredníctvom dvojitej autentizácie 2FA. Okrem praktických nástrojov sme spomenuli aj zabezpečenie dát, konkrétne pre zariadenia Android pri vývoji aplikácie využitím šifrovania obsahu databázy.

Praktická časť sa delí na dve sekcie, tá prvá je analytická, kde sme si opísali návrh aplikácie z pohľadu diagramu prípadov použitia či entito-relačného diagramu, ktorý opisoval štruktúru databázy v mobilnom zariadení. Sekvenčným diagramom sme opísali prípad použitia, ktorý opisoval vytvorenie výdavku pre konkrétnu cestu, aby sme si ukázali komunikáciu medzi jednotlivými komponentami aplikácie. Výsledkom návrhu je diagram nasadenia, ktorý opisuje komunikáciu medzi aplikačným serverom, databázovým serverom a mobilným zariadením, na ktorom vyvíjaná aplikácia pracuje.

Druhá polovica praktickej časti opisovala realizáciu návrhu s pohľadom na konkrétne obrázky, ktoré sa týkali napr. nastavenia aplikačného servera pre komunikáciu s databázovým serverom, časti zdrojového kódu web servisu, ktorý bol nasadený na aplikačnom serveri a i. Okrem využitých serverov sme si opísali aj komunikáciu, ktorá prebiehala medzi nimi. Medzi mobilnou aplikáciou a aplikačným serverom prostredníctvom REST volaní, resp. s databázou prostredníctvom JDBC konektora.

Nasledoval opis samotnej mobilnej aplikácie, kde bola opísaná funkcionalita, spolu s používateľským rozhraním, ktoré opisovalo realizované návrhy. Opis sa skladal zo sekcií, ktoré bližšie opisovali funkcionalitu aplikácie a možnosti, ktoré používateľ má pri jej používaní. Aplikácia používa dátovú prevádzku (pripojenie do internetu) z dôvodu komunikácie s aplikačným serverom (prihlasovanie/registrácia), ako aj komunikáciu so serverom pre zisťovanie aktuálneho kurzu meny. Implementácia databázy bola realizovaná prostredníctvom Room databázy, ktorá pracuje na SQLite a tá je uložená v internom úložisku zariadenia.

Využitím dátového prenosu aplikácia používa aj externé API, konkrétne Google Maps, pre zobrazenie polohy výdavkov. Využíva aj niekoľko externých knižníc, ktoré pomáhajú k spracovaniu JSON formátu, vytvoreniu diagramov pre analýzu dát alebo vytvoreniu a exportovaniu PDF súborov.

Aplikácia sa snaží využívať potenciál mobilných zariadení využívaním hardvérových komponentov, ako je fotoaparát či GPS lokalizácia alebo využívaním internetového pripojenia na zisťovanie polohy zariadenia a komunikáciu so vzdialeným serverom.

SEZNAM POUŽITÉ LITERATURY

[1] Man Ho A., Choo, KKR. (2017). Mobile Security and Privacy. Cambridge. Dostupné na: https://learning.oreilly.com/library/view/mobile-security-and/9780128047460/Cover.xhtml

[2] D'Orazio, CH., Choo, KKR. (2015), A generic process to identify vulnerabilities and design weaknesses in iOS healthcare apps. In Proceedings of 48th Annual Hawaii International Conference on System Sciences (HICSS 2015), pp. 5175–5184, 5–8 January 2015, IEEE Computer Society Press. Dostupné na: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2545755

[3] Norton. What are bots? Dostupné na: https://us.norton.com/internetsecurity-malware-what-are-bots.html

[4] Boodaei M. (2011). Mobile Users 3 Times More Vulnerable to Phishing Attack. Dostupné na: https://securityintelligence.com/mobile-users-3-times-more-vulnerable-to-phishing-attacks/

[5] Porta La L. (2018). Phishing. Dostupné na: https://www.wandera.com/mobile-security/phishing/mobile-phishing-attacks/

[6] Rossi B. (2016). What are mobile main-in-the-middle attacks and how dangerous are they? Dostupné na: https://www.information-age.com/what-are-mobile-man-middle-attacks-and-how-dangerous-are-they-123461071/

[7] Tutlane. SqLite injection Attacks. Dostupné na: https://www.tutlane.com/tutorial/sqlite/sqlite-injection-attacks

[8] Nolan, A. (2014). Bulletproof Android: Practical Advice for Building Secure Apps. Addison-Wesley. Crawfordsville. Dostupné na: http://ptgmedia.pearsoncmg.com/images/9780133993325/samplepages/9780133993325.pdf

[9] Rose Ch. (2017). The Security Implications Of The Internet Of Things. Journal of Cybersecurity Research – 2017 (2,1).

[10] Dolly J. (2018). Why you should never, ever, connect to public WiFi. Dostupné na: https://www.csoonline.com/article/3246984/why-you-should-never-ever-connect-to-public-wifi.html

[11] Mason J. (2019). VPN Beginner's Guide. Dostupné na: https://thebestvpn.com/what-is-vpn-beginners-guide/

[12] Raphael JR. (2017). What is two-factor authentication (2FA)? How to enable it and why you should? Dostupné na: https://www.csoonline.com/article/3239144/what-is-two-factor-authentication-2fa-how-to-enable-it-and-why-you-should.html

[13] Drzhzhin A. (2018) SMS-based two-factor authentication is not safe – consider these alternative 2FA methods instead. Dostupné na: https://www.kaspersky.com/blog/2fa-practical-guide/24219/

[14] Android developers. Uses SDK. Dostupné na: https://developer.android.com/guide/topics/manifest/uses-sdk-element#ApiLevels

[15] In: Phishing Definition, Prevention, and Examples [online]. [cit. 2019-05-09]. Dostupné z: https://resources.infosecinstitute.com/category/enterprise/phishing/#gref

[16] In: What is a Phishing attack? [online]. [cit. 2019-04-20]. Dostupné z: https://www.cloudflare.com/learning/security/threats/phishing-attack/

[17] In: CS111 Winter 2013 Lecture 17: Introduction to Security [online]. [cit. 2019-04-20]. Dostupné z: http://web.cs.ucla.edu/classes/winter13/cs111/scribe/17b/

[18] NOLAN, A. Bulletproof Android: Practical Advice for Building Secure Apps [online]. In: . Crawfordsville: Addison-Wesley, 2014 [cit. 2019-04-15]. Dostupné z: http://ptgmedia.pearsoncmg.com/images/9780133993325/samplepages/9780133993325.pdf

[19] In: VPN Services [online]. [cit. 2019-04-22]. Dostupné z: https://cybersecurity.osu.edu/cybersecurity-you/use-right-tools/vpn-services

[20] In: How to Install an SSL Certificate [online]. 2018 [cit. 2019-04-23]. Dostupné z: https://sucuri.net/guides/how-to-install-ssl-certificate

[21] In: Two Factor Authentication Made Easy [online]. [cit. 2019-04-24]. Dostupné z: https://www.researchgate.net/figure/A-high-level-overview-of-a-Web-2FA-sequence_fig1_280027625

[22] NOLAN, A. Bulletproof Android: Practical Advice for Building Secure Apps [online]. In: . Crawfordsville: Addison-Wesley, 2014 [cit. 2019-04-15]. Dostupné z: http://ptgmedia.pearsoncmg.com/images/9780133993325/samplepages/9780133993325.pdf

[23] NOLAN, A. Bulletproof Android: Practical Advice for Building Secure Apps [online]. In: . Crawfordsville: Addison-Wesley, 2014 [cit. 2019-04-15]. Dostupné z: http://ptgmedia.pearsoncmg.com/images/9780133993325/samplepages/9780133993325.pdf

[24] In: OneProxy :: Connection Pool is Matter when Deploying Applications by Docker! [online]. [cit. 2019-04-30]. Dostupné z: http://www.onexsoft.com/en/oneproxy-php-connection-pool.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
JDBC	Java Database Connectivity.
SOAP	Simple Object Access Protocol.
SQL	Structured Query Language.
SSL	Secure Socket Layer.
TLS	Transport Layer Security.
WSDL	Web Services Description Language.
XML	eXtensible Markup Language.

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: PUBLIKOVANIE APLIKÁCIE NA GOOGLE PLAY

Ak by mal vývojár záujem publikovať mobilnú aplikáciu na operačný systém Android, je potrebné postupovať niekoľkými krokmi:

1) Vytvorenie Developer Account (účtu vývojára) – na začiatku procesu je potrebné založiť si účet, prostredníctvom ktorého bude aplikácia publikovaná. Na vytvorený účet môže byť použitý existujúci Google účet. Pri registrácií je zároveň uhradiť registračný poplatok vo výške 25 dolárov.
2) Prepojenie Merchant Account (obchodného účtu) – v prípade, že aplikácia nebude distribuovaná zadarmo, ale platená, je potrebné vytvorenie aj obchodného účtu, kde je možné spravovať a analyzovať predaj aplikácie.
3) Opis aplikácie – aplikácia sa v Google Play zobrazí spolu s jej opisom (názov, stručný opis, celkový opis aplikácie) a grafickými náhľadmi (video, obrázok), ktoré treba v danom kroku vyplniť. Spolu s tým je potrebné kategorizovať, do akej oblasti aplikácia spadá a vyplniť kontakné údaje, ktoré budú slúžiť na podporu zákazníkovi.
4) Nasadenie aplikácie – v ďalšom kroku prebieha pridanie aplikácie do účtu v podobe .apk (Android Package Kit) súboru. Tento formát predstavuje inštalačný súbor aplikácie (podobne ako .exe súbor pri OS Windows).
5) Pri nasadení aplikácie je potrebné nastaviť, či bude používateľmi povolené hodnotenie aplikácie alebo nie. Treba brať ale ohľad na to, že aplikácie, ktoré nemajú hodnotenie môže viesť k suspendácií alebo vymazaniu.
6) Posledným krokom je nastavenie aplikácie, či bude používateľom sprístupnená zadarmo alebo bude platená. Google play tu má avšak podmienky, na ktoré treba brať ohľad. V prípade, že aplikácia bude nastavená ako spoplatnená, neskôr je možné ju v ktoromkoľvek čase nastaviť pre používateľov zadarmo. Avšak, tento systém nefunguje opačne, pretože ak bude aplikácia zadarmo, nie je ju možné neskôr spoplatniť.
7) Po všetkých týchto krokoch je aplikácia úspešne nasadená, avšak tým to nekončí. Aby bola aplikácia úspešná, je potrebné ju propagovať a zabezpečiť jej dostatočný marketing.

PŘÍLOHA P II: SCENÁRE PRÍPADOV POUŽITIA

1) **UC1: Registrácia**

Aktér: Používateľ

Vstupná podmienka: Spustená mobilná aplikácia s obrazovkou "Registration".

Hlavný scenár:

1. Používateľ vyplní zobrazený formulár.

2. Používateľ stlačí tlačidlo potvrdenia registrácie.

3. Systém odošle požiadavku registrácie.

4. Vykonanie UC3: Pridanie používateľa.

5. Systém potvrdí úspešné zaregistrovanie používateľa.

Výstupná podmienka: Systém zobrazí obrazovku zoznamu ciest.

2) **UC3: Vloženie nového používateľa**

Aktér: Používateľ

Vstupná podmienka: Používateľ stračil tračidlo odoslania registračného formulára

Hlavný scenár:

1. Webový servis príjme požiadavku zaslanú používateľom.

2. Webový servis zašifruje prijaté heslo používateľa.

3. Webový servis vytvorí model pre databázovú aplikáciu.

4. Webový servis zašle požiadavku na databázovú aplikáciu.

5. Databázová aplikácia vloží hodnoty modelu do databázy.

6. Webový servis vráti odpoveď používateľovi.

Výstupná podmienka: Používateľ je úspešne zaregistrovaný.

3) **UC2: Prihlásenie**

Aktér: Používateľ

Vstupná podmienka: Spustená mobilná aplikácia s obrazovkou "Login".

Hlavný scenár:

1. Používateľ vyplní zobrazený formulár.

2. Používateľ stlačí tlačidlo potvrdenia prihlásenia.

3. Systém odošle požiadavku prihlásenia.

4. Vykonanie UC4: Overenie používateľa.

5. Systém potvrdí úspešné prihlásenie používateľa.

Výstupná podmienka: Systém zobrazí obrazovku zoznamu ciest.

4) **UC4: Overenie používateľa**

Aktér: Používateľ

Vstupná podmienka: Používateľ stračil tračidlo odoslania prihlasovacieho formulára.

Hlavný scenár:

1. Webový servis príjme požiadavku zaslanú používateľom.

2. Webový servis zašifruje prijaté heslo používateľa.

3. Webový servis zašle SQL dopyt pre databázovú aplikáciu s menom používateľa.

4. Databázová aplikácia vráti výsledok z databázy.

5. Webový servis potvrdí zhovu používateľského mena.

6. Webový servis potvrdí zhodu používateľského hesla.

Výstupná podmienka: Webový servis vracia používateľovi potvrdenie o úspešnom prihlásení.

Altenatívne scenáre:

5a. Webový servis zistí, že sa používateľské meno nezhoduje

5a1. Webový servis pošle používateľovi odpoveď, že používateľ nebol nájdený.

6a. Webový servis nenájde zhodu s používateľským heslom.

6a1. Webový servis pošle používateľovi odpoveď, že sa heslo nezhoduje.

5) **UC5: Odstránenie cesty**

Aktér: Používateľ

Vstupná podmienka: Obrazovka zobrazujúca všetky vytvorené cesty

Hlavný scenár:

1. Používateľ vykoná dlhé sktlačenie na konkrétnu cestu.

2. Systém používateľovi zobrazí možnosti správy cesty.

3. Používateľ klikne na možnosť "Delete".

4. Systém zobrazí otázku, či cestu skutočne vymazať.

5. Používateľ potvrdí vymazanie cesty stlačením tlačidla "Delete".

6. Systém odstráni zadanú cestu.

Výstupná podmienka: Systém obnový obrazovku všetkých existujúcich ciest bez zmazanej cesty.

Alternatívny scenár:

5a. Používateľ zruší vymazanie cesty tlačidlom "Cancel".

5a1. Systém obnoví obrazovku zobrazujúcu všetky existujúce cesty.

6) **UC6: Vytvorenie cesty**

Aktér: Používateľ

Vstupná podmienka: Obrazovka zobrazujúca všetky vytvorené cesty

Hlavný scenár:

1. Používateľ stlačí tlačidlo pridania cesty.

2. Systém používateľovi zobrazí formulár na vytvorenie cesty.

3. Používateľ vyplní všetky polia formulára.

4. Používateľ stlačí tlačidlo potvrdenia vytvorenia cesty.

5. Systém oznámi používateľovi úspešné vytvorenie cesty.

Výstupná podmienka: Systém zobrazí obrazovku všetkých ciest spolu s vytvorenou cestou.

7) **UC7: Zmena parametrov cesty**

Aktér: Používateľ

Vstupná podmienka: Obrazovka zobrazujúca všetky vytvorené cesty

Hlavný scenár:

1. Používateľ vykoná dlhé sktlačenie na konkrétnu cestu.

2. Systém používateľovi zobrazí možnosti správy cesty.

3. Používateľ klikne na možnosť "Edit".

4. Systém zobrazí používateľovi formulár cesty s existujúcimi parametrami.

5. Používateľ zmení požadované parametre.

6. Používateľ stlačí tlačidlo na potvrdenie uloženia.

Výstupná podmienka: Systém zobrazí obrazovku všetkých ciest.

8) **UC8: Exportovanie cesty do PDF**

Aktér: Používateľ

Vstupná podmienka: Hlavná obrazovka vybratej cesty

Hlavný scenár:

1. Používateľ otvorí menu v lište cesty.

2. Systém zobrazí možnosti menu.

3. Používateľ strači na možnosť "Export to PDF".

4. Systém vytvorí PDF súbor a uloží ho do súborového systému.

Výstupná podmienka: Systém potvrdí úspešné vytvorenie PDF súboru.

9) **UC9: Vytvorenie nového výdavku**

Aktér: Používateľ

Vstupná podmienka: Hlavná obrazovka vybratej cesty zobrazujúca aktuálny stav rozpočtu

Hlavný scenár:

1. Používateľ stlačí tlačito na vytvorenie výdavku.

2. Systém zobrazí možnosti kategórie výdavku

3. Používateľ vyberie kategóriu výdavku.

4. Systém zobrazí formulár výdavku.

5. Systém nastaví do formulára aktuálny dátum zariadenia.

6. Vykonanie UC10: Zistenie polohy zariadenia.

7. Používateľ vyplní formulár výdavku.

8. Používateľ stlačí tlačidlo pre potvrdenie výdavku.

9. Systém pridá výdavok do databázy.

Výstupná podmienka: Systém zobrazí hlavnú obrazovku cesty spolu s pridaným výdavkom.

10) **UC10: Zistenie polohy zariadenia**

Aktér: Systém

Vstupná podmienka: Otvorenie formulára nového výdavku

Hlavný scenár:

1. Systém získa zemepisnú výšku a šírku zariadenia na základe internetového pripojenia.

2. Na základe zem. výšky a šírky systém určí adresu zariadenia.

3. Systém vyplní požadované polia formulára vyhľadanou adresou.

Výstupná podmienka: Vyplnené polia formulára.