

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

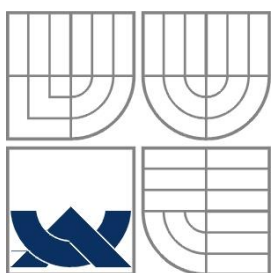
MOBILNÍ SYSTÉM PRO PODPORU CESTOVÁNÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

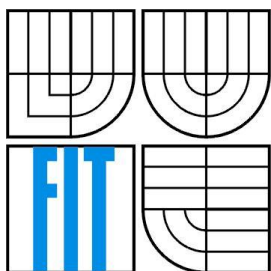
AUTOR PRÁCE
AUTHOR

Bc. Petr Blatný

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V
BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

MOBILNÍ SYSTÉM PRO PODPORU CESTOVÁNÍ

MOBILE SYSTEM FOR TRAVELLING SUPPORT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Petr Blatný

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Pavel Očenášek, Ph.D.

BRNO 2014

Abstrakt

Tato diplomová práce se zabývá popisem návrhu a implementace mobilního systému pro podporu cestování. Aplikace, pojmenovaná TravelHelper, je implementována pro platformu Android, a je tedy napsána v programovacím jazyce Java. Text proto popisuje práci s použitou platformou a její základní části. Hlavní funkcí aplikace je cestovní deník, umožňující uživateli zaznamenávat významné body cest. Aplikace umožňuje tyto cesty sdílet na sociální síť a zálohovat do souboru. Dalšími funkcemi aplikace jsou vyhledávání míst v okolí, správa cestovních výdajů, překladáč, převodník měn a seznam pro balení.

Abstract

This master's thesis describes process of design and development of a mobile system for travelling support. Application named TravelHelper is developed for Android operating system therefore is implemented in Java programming language. In thesis's text is described used platform and its primary construction blocks. Main function of application is a journeys' diary, which allows storing activities of a trips. Application also allows backing up and sharing journeys on a social networks. Other functions of the application are searching for interesting places nearby, manager of travel expenses, translator, currency converter, and a packing list.

Klíčová slova

Android, cestování, mobilní aplikace, překladáč, převodník měn, Mapy Google, Google+, Facebook, Java,

Keywords

Android, travelling, mobile application, translator, currency converter, Google Maps, Google+, Facebook, Java,

Citace

Blatný Petr: Mobilní systém pro podporu cestování, diplomová práce, Brno, FIT VUT v Brně, 2014

Mobilní systém pro podporu cestování

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Blatný
28. května 2014

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Pavlu Očenáškov Ph.D. za poskytování užitečných nápadů a rad a za jeho trpělivost. Také bych rád poděkoval všem, kteří mi poskytli jejich zařízení pro otestování aplikace.

© Petr Blatný, 2014

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah.....	1
1 Úvod	3
2 Použité technologie.....	4
2.1 Platforma Android.....	4
2.1.1 Architektura.....	4
2.1.2 Součásti aplikace	7
2.2 Použité knihovny	14
2.2.1 Google Services.....	14
2.2.2 Facebook SDK	17
3 Analýza požadavků.....	18
3.1 Srovnání konkurenčních řešení	18
3.1.1 Tripomatic	18
3.1.2 Tripit.....	19
3.1.3 Kayak.....	20
3.1.4 Money Tab	21
4 Specifikace a návrh aplikace	22
4.1 Funkce aplikace.....	22
4.1.1 Cestovní deník.....	22
4.1.2 Místa v okolí.....	22
4.1.3 Převodník měn.....	23
4.1.4 Překladač	23
4.1.5 Kontrola výdajů	23
4.1.6 Seznam pro balení	23
4.2 Návrh uživatelského rozhraní.....	23
4.3 Návrh databáze	28
5 Implementace.....	30
5.1 Cestovní deník.....	30
5.1.1 Seznam cest	30
5.1.2 Detail cesty	31
5.1.3 Sdílení.....	33
5.1.4 Zálohování.....	35
5.2 Nová položka.....	37
5.2.1 Cesta	37
5.2.2 Aktivita.....	38

5.2.3	Výdaj	39
5.3	Místa v okolí.....	39
5.4	Správa výdajů.....	41
5.5	Překladač	42
5.6	Převodník měn.....	43
5.7	Balící seznam	44
5.8	Databáze	45
5.8.1	Serverová databáze	47
5.9	Uživatelské rozhraní.....	47
6	Testování	51
7	Závěr	55
Seznam příloh.....		57
Příloha A.	Struktura balíčků zdrojových kódů.....	57
Příloha B.	Obsah CD/DVD	59

1 Úvod

Cestování je přirozenou součástí lidské osobnosti. Mnoho lidí cestuje za prací nebo školou každý den, pracovníci jezdí na služební cesty. Většina si však pod cestováním představí poznávání neznámých míst. V dnešní době je cestování díky mnoha druhům dopravy a jejich dostupnosti velmi jednoduché.

V dřívějších dobách využívali cestovatelé při návštěvách cizích míst tištěné mapy, slovníky a další pomůcky. V dnešní době díky rozvoji techniky může jediné zařízení obsáhnout všechny tyto funkce. Takovým zařízením může být dnes například chytrý telefon, který má člověk u sebe téměř na každém kroku. A právě díky tomu se jedná o ideální cestovní pomůcku.

Cílem této práce je navrhnout a implementovat mobilní aplikaci pro potřeby cestovatelů. Aplikace by měla umožnit vyhledat zajímavá místa a služby a tyto zobrazit přehledně na mapě. Umožnit uživateli překlad cizojazyčných textů do rodného jazyka a převod částky v cizí měně do domácí. Dále aplikace umožní sledovat a zaznamenávat polohu uživatele a tvořit tak cestovní deník. Tyto data aplikace umožní sdílet na sociální síti, kde je mohou zobrazit ostatní uživatelé.

V následující kapitole je popsán operační systém Android. Kapitola se zabývá popisem součástí aplikace a jejich životních cyklů. Dále kapitola popisuje knihovny použité v aplikaci. Jsou jimi Google Services a Facebook SDK. Je popsáno jejich obecné využití a nejdůležitější funkce. V další kapitole jsou popsány konkurenční řešení. Jsou popsány jejich funkce a uživatelské rozhraní. Dále jsou diskutovány jejich výhody a nevýhody oproti ostatním.

V kapitolách 4 a 5 jsou popsány technický návrh aplikace a její implementační detaily. Kapitoly se zaměřují na popis jednotlivých funkcí aplikace, uživatelského rozhraní a ukládání trvalých dat do databáze. V posledních kapitolách je popsáno testování aplikace a je nastíněn možný budoucí vývoj projektu.

2 Použité technologie

2.1 Platforma Android

Android je open source operační systém určený pro různá přenosná zařízení, jako jsou telefony, tablety, navigace a další. Je možné jej definovat jako balík software, který obsahuje upravené Linuxové jádro systému, middleware¹, uživatelské rozhraní a aplikace.

Společnost Android Inc. vyvíjela systém od roku 2003. V roce 2005 byla společnost odkoupena firmou Google Inc. Roku 2007 byla založena Open Handset Alliance, jejímiž členy jsou přední technologické společnosti. Ve stejném roce byl operační systém Android představen spolu s Android SDK, které poskytuje kompletní balík technologií umožňující návrh aplikací.

Od uvedení po dnešní dobu vzniklo 9 verzí tohoto systému a několik dalších pod-verzí. Tyto verze jsou pojmenovány podle cukrovinek. Jejich přehled zobrazuje [Tabulka 1](#). V každé nové verzi byly vydávány opravy a nová funkcionality. Například verze 3 byla určena pouze pro zařízení typu tablet, verze 4 přinesla přepracované a napříč aplikacemi sjednocené uživatelské rozhraní a hlasové ovládání. Zastoupení jednotlivých verzí operačního systému na trhu je patrné z grafu ([Obrázek 1](#)).

Tabulka 1: Verze operačního systému Android.

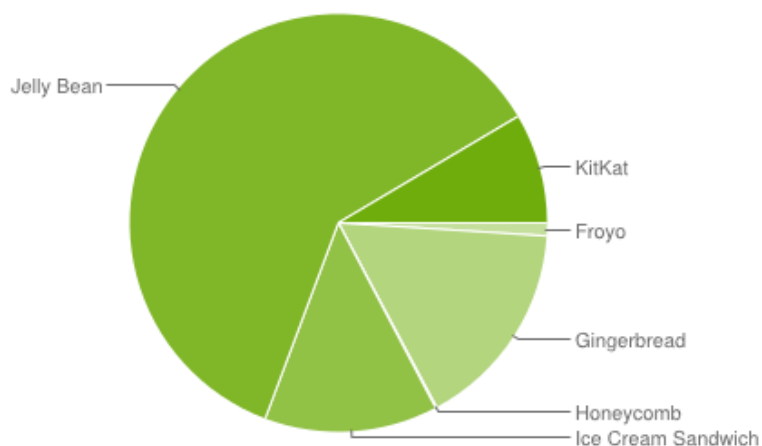
Číslo verze	Název	Verze API
1.5	Cupcake	3
1.6	Donut	4
2.1	Eclair	7
2.2	FroYo	8
2.3.0 – 2.3.7	Gingerbread	10
3.0 – 3.2	Honeycomb	13
4.0.0 – 4.0.3	Ice Cream Sandwich	15
4.1 – 4.3	Jelly Bean	16 – 18
4.4	KitKat	19

2.1.1 Architektura

Architekturu systému [\[3\]](#)[\[4\]](#) tvoří 5 vrstev. Jsou jimi:

- Linuxové jádro,
- Knihovny,
- Android runtime,
- Aplikační Framework,
- Aplikace.

¹ Middleware - software přemostující spodní vrstvy systému a vyšší aplikační vrstvu



Obrázek 1: Zastoupení verzí OS Android. Data grafu odpovídají měření v květnu 2014. [9]

Linuxové jádro ve verzi 2.6 tvoří vrstvu mezi hardware a vyššími vrstvami. Využívá systémové prostředky jako ovladač displeje, kamery, audia, klávesnice a radiových technologií (GSM, WIFI), správu napájení, zabezpečení, správu paměti a správu procesů.

Knihovny jsou založené na C/C++ knihovnách a jsou dostupné přes vrstvu Aplikační Framework. Patří mezi ně:

- Libc – systémové knihovny jazyka C/C++,
- SQLite – pro práci s databázemi,
- Surface Manager – práce s dotykovým ovládáním,
- WebKit – knihovna pro zobrazení a práci s webovými stránkami,
- Media Framework – knihovny pro přehrávání multimédií,
- OpenGL ES – knihovna pro zobrazování 2D a 3D grafiky,
- a další.

Android runtime je tvořen upraveným běhovým prostředím pro aplikace. Tím je Dalvik Virtual Machine, upravený virtuální stroj pro spouštění Java aplikací. Vrstvu proto tvoří knihovny programovacího jazyka Java, ve kterém jsou aplikace vytvářeny.

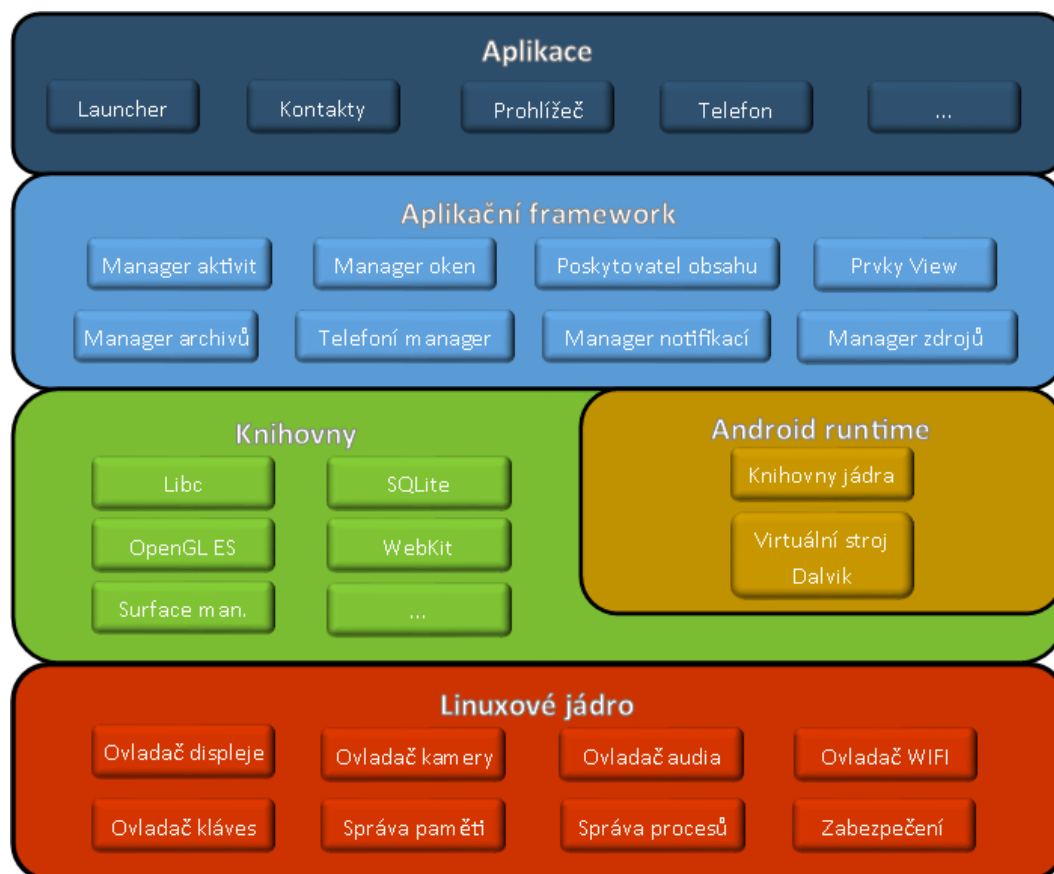
Aplikační Framework poskytuje vývojáři základní bloky pro výstavbu aplikace. Všechny aplikace, včetně systémových používají shodné API. Díky tomu je možné vytvořit vlastní aplikaci, která může nahradit funkci systémové aplikace (například aplikace pro SMS, kalendář, kontakty a další). Vrstvu tvoří tyto základní bloky:

- Activity manager – poskytuje správu životního cyklu aplikace.
- View systém – prvky třídy View, která je nadtrídou všech prvků uživatelského rozhraní.
- Notification manager – prvek pro správu upozornění.
- Location manager – poskytuje přístup k poloze zařízení.
- Resource manager – umožňuje práci s externími zdroji aplikace, jako jsou obrázky, zvuky a další.

- a další.

Aplikace v systému Android jsou například Telefon, Kontakty, Launcher, jenž tvoří výchozí obrazovku a poskytuje přístup k dalším aplikacím, Prohlížeč webových stránek a mnohé další.

Celá výše popsaná architektura je znázorněna na následujícím obrázku.



Obrázek 2: Architektura operačního systému Android. Zdroj [4], vlastní úprava.

2.1.2 Součásti aplikace

Aplikace pro android sestávají z několika význačným prvků. Jsou jimi aplikační kód, soubor *AndroidManifest.xml* a dále uživatelské rozhraní a zdroje. Tyto prvky tvoří archiv *AndroidPackage*, což je soubor s příponou „*apk*“, jenž obsahuje následující adresářovou strukturu:

```
MyProject/  
  src/  
    MainActivity.java  
  res/  
    drawable/  
      icon.png  
    layout/  
      main.xml  
      info.xml  
    values/  
      strings.xml  
  AndroidManifest.xml
```

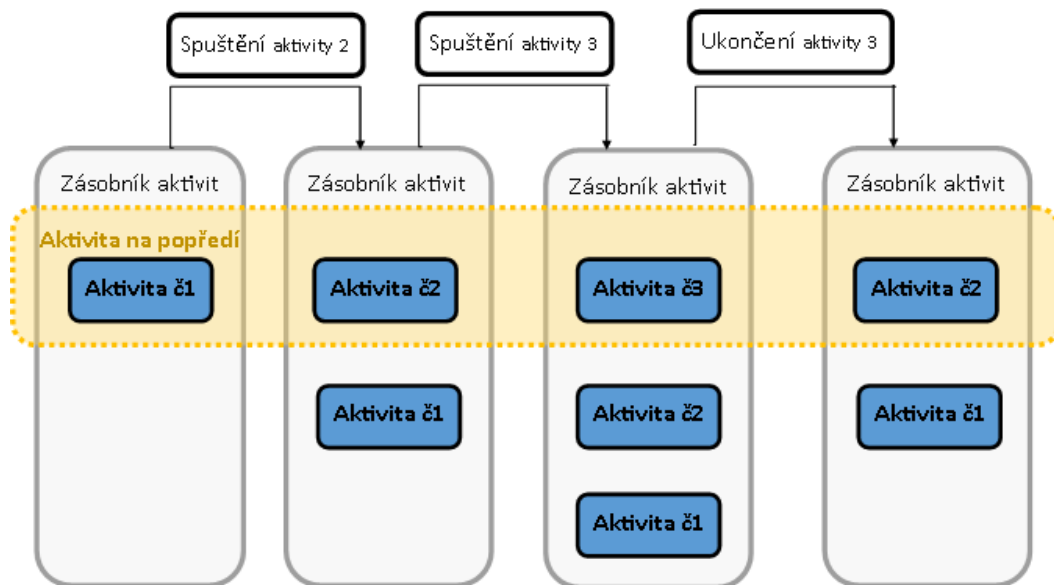
Každá aplikace běží v systému ve vlastním procesu, jenž je úplně oddělen od ostatních procesů. Proces aplikace je spouštěn, jakmile to některá z jejích součástí vyžaduje a ukončen je, jakmile již aplikace není využívána nebo když systém potřebuje uvolnit paměť pro jiné procesy. Každý takovýto proces potom běží ve vlastní instanci virtuálního stroje Dalvik. Procesy mohou přistupovat k dalším zdrojům dat, ale pouze za předpokladu, že jim byly uděleny práva.

Aplikační kód využívá základních komponent systému. Jsou jimi Aktivitty, Služby, Poskytovatelé obsahu, Přijímači Broadcastu. Komponenty mezi sebou komunikují prostřednictvím asynchronních zpráv zvaných Intent. Tyto komponenty mají v aplikaci různé využití a odlišný způsob komunikace s uživatelem.

Aktivita

Aktivita [10] reprezentuje vždy jednu obrazovku uživatelského rozhraní a tím umožňuje uživateli vykonávat různé akce. Aplikace obvykle sestává z více aktivit, které jsou mezi sebou různě provázány. Jedna z aktivit je vyznačena jako výchozí a aplikace po spuštění zobrazí právě tuto aktivitu. Z aktivitty je možné spouštět jiné (například v seznamu kontaktů při vybrání, je zobrazen detail). Start aktivitty znamená zastavení předchozí, avšak systém ji uloží do zásobníku aktivit. To umožňuje rychlý návrat na předchozí aktivitu při stisku tlačítka zpět. **Obrázek 3** znázorňuje funkci zásobníku aktivit.

Životní cyklus aktivitty jednoznačně definuje její chování a reakce na uživatelské a systémové akce. Aktivitu je možné spustit dvěma různými způsoby pomocí příkazů `startActivity` a `startActivityForResult`. První zajistí běžné spuštění aktivitty, naopak druhý definuje způsob, jak ze spuštěné aktivitty při návratu získat nějaká data. Tato data, stejně



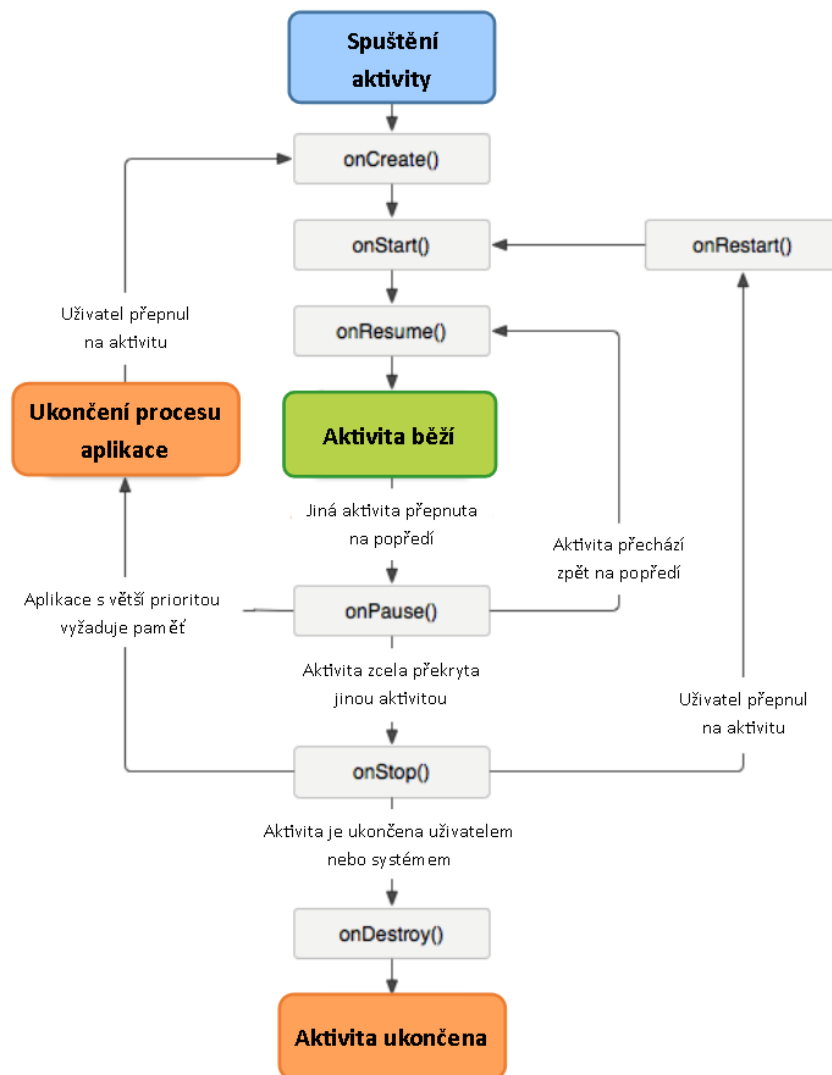
Obrázek 3: Znáznornění funkce zásobníku aktivit. Čerpá z [10], vlastní úprava.

jako data, která předáváme spouštěné aktivitě, jsou zabalena ve zprávě Intent. Ukončení spuštěné aktivity je možné příkazem `finish` nebo `finishActivity` při ukončování jiné. Životní cyklus aktivity sestává ze tří základních stavů:

- Resumed – Aktivita je v popředí a má uživatelský kontext.
- Paused – Jiná aktivita má kontext, ale aktivita je stále viditelná (překryta jen z části).
- Stopped – Aktivita je zcela překryta jinou.

Průběh životního cyklu aktivity zobrazuje Obrázek 4. Správu životního cyklu je možné provádět pomocí metod aktivity a lze díky nim rozdělit životní cyklus na 3 vnořené smyčky.

- Celý životní cyklus – je ohraničen metodami `onCreate` a `onDestroy`. Při vytváření se obvykle inicializuje uživatelské rozhraní, naopak při rušení se ukončují procesy na pozadí.
- Viditelný životní cyklus – ohraničují metody `onStart` a `onStop`. Tyto metody mohou být za celý životní cyklus aktivity spuštěny několikrát.
- Životní cyklus na popředí – ohraničují metody `onResume` a `onPause`. Tyto aktivity jsou v průběhu volány nejčastěji, například při uspání zařízení.



Obrázek 4: Životní cyklus aktivity. Čerpá z [10], vlastní úprava.

Služby

Služba (Service) [7] je komponenta, která neposkytuje uživatelské rozhraní. Využívá se k běhu déle trvajících operací na pozadí. Může to být například služba přehrávání hudby nebo komunikace se serverem. Služba může běžet nezávisle na aplikaci, která ji spustila. Tedy například přehrávání hudby pokračuje i po opuštění aplikace přehrávače. Je možné rozlišit dva druhy služeb, spouštěné a připojované. Spouštěné služby vytváří a spouští aplikace, ve které je služba obsažena, příkazem `startService`. Tato služba vykonává určitou operaci a po jejím dokončení se ukončí. Připojované služby mohou využívat i jiné aplikace. Takové služby běží stále na pozadí. Potřebuje-li aplikace data, jež služba poskytuje, připojí se ke službě příkazem `bindService`. Ke službě může být takto připojeno i více aplikací zároveň. Připojovaná služba se ukončí po odpojení posledního klienta. Možný je i případ kombinování obou typů, tedy ke spouštěné službě se mohou připojit další klienti.

Životní cyklus služby se liší od aktivity, kdy u služby jsou rozlišeny pouze dva stavy:

- Celý životní cyklus – ohraničují metody `onCreate` a `onDestroy`.
- Aktivní životní cyklus – liší se podle typu služby. Pro spouštěnou službu je ohraničen metodou `onStartCommand`. Koncové ohraničení nemá. Připojovaná služba je ohraničena metodami `onBind` a `onUnbind`.

Poskytovatelé obsahu

Poskytovatelé obsahu (ContentProviders) [11] zpřístupňují data aplikace a definují způsob jejich zabezpečení. Poskytují standartní rozhraní pro získání dat z jednoho procesu do jiného. K získání přístupu k datům slouží objekt `ContentResolver`, který komunikuje s instancí objektu `ContentProvider`. Ke zpřístupnění standardních systémových dat jako kontakty nebo kalendář jsou v systému obsaženy základní poskytovatelé, k nimž je možno se libovolně připojit.

Přijímači broadcastu

Přijímači broadcastu slouží pro příjem systémových a jiných oznámení. Například zpráva o zhasnutí obrazovky, slabé baterii a dalších. Aplikace může zprávy typu broadcast vytvářet, například při dokončení stahování dat, které mohou vyžadovat další aplikace. Tato komponenta neposkytuje uživatelské rozhraní, může však vyvolat například upozornění v systémové liště. Obvykle však poskytuje prostředek pro reakci aplikací na nějakou událost.

Intent

Jak bylo popsáno výše, komponenty mezi sebou komunikují prostřednictvím zpráv zvaných *Intent* [12]. Tyto zprávy mohou být zasílány komponentám v rámci dané aplikace, ale i mezi různými aplikacemi. Objekt `Intent` obsahuje strukturovaná pasivní data, která popisují, jaká akce má být provedena při jejich přijetí, nebo naopak nesou informaci o akci, která se udála. Systém obsahuje oddělené mechanismy pro doručování Intentů jednotlivým komponentám.

- Při spouštění aktivity je zasílán `Intent` metodami `startActivity` a `startActivityForResult`. Tento `Intent` nese informaci, které aktivitě má být předán a může nést další data. Stejně tak při vracení obsahu ukončované aktivity metodou `setResult` je zasílán `Intent` s informací o výsledku aktivity. Takovéto `Intenty` jsou doručovány pouze té aktivitě, které jsou určeny.
- `Intent` je zasílán stejným způsobem i při spouštění služby metodou `startService` nebo při doručení nových instrukcí spouštěné službě. Obdobně pro připojované služby vytváří `Intent` požadované spojení mezi službou a volající komponentou. Tímto způsobem může dojít i ke spuštění služby, pokud neběží.

- Intenty jsou využívány jako prostředek při rozesílání broadcastových zpráv metodou `sendBroadcast`. Tyto Intenty jsou doručeny všem posluchačům broadcastu.

Objekt `Intent` obsahuje informace podstatné pro komponentu, jenž `Intent` přijímá (např. akci kterou má provést), a dále informace přijímané systémem (např. informace, jak spustit danou aktivitu). Detailně sestává z následujících komponent:

- Jméno komponenty – značí komponentu, jenž přijímá `Intent`. Jméno je specifikováno úplným jménem třídy, která komponentu obsahuje. Jedná se o nepovinný atribut, který pokud chybí, systém určí adresáta z ostatních atributů.
- Akce – je název akce, jenž má být vykonána. Třída `Intent` obsahuje konstanty reprezentující akce. Tento atribut dále ovlivňuje podobu dalších argumentů, stejně jako metoda definuje své atributy a návratovou hodnotu.
- Data – jsou dvojice URI identifikátoru a MIME² typ dat. Tyto hodnoty závisí na atributu akce (např. pro akci `ACTION-CALL` budou data obsahovat URI: `tel:` specifikující volané telefonní číslo).
- Kategorie – je textový zápis druhu komponent, které `Intent` přijímají. Kategorií v atributu může být i více než jedna. Třída `Intent` obsahuje konstanty definující některé kategorie.
- Extra – je dvojice klíč-hodnota, jenž jsou další informace doručované přijímající komponentě. Namísto atributu data může akce specifikovat atribut extra.
- Příznaky (Flags) – obsahují informace o tom, jak má systém spouštět aktivity a dále jak s nimi pracovat. Tyto příznaky jsou uvedeny ve třídě `Intent`.

Intenty mohou být děleny do dvou skupin:

- Explicitní – jenž specifikují komponentu, které mají být doručeny. Jsou používány převážně v rámci jedné aplikace, kde jsou známy jména komponent.
- Implicitní – nenesou jméno cíle. Jsou proto využívány pro komunikaci mezi aplikacemi.

K datům obsažených v explicitním `Intentu` má přístup pouze cílová komponenta. Naopak u implicitních `Intentů`, kde není specifikovaný příjemce, musí systém určit nejvhodnější komponentu (nebo komponenty) jenž `Intent` přijmou. Systém porovnává obsah `Intent` zprávy a `Intent`-filtru. Porovnávají jsou atributy Akce, Data a Kategorie.

Intent filtry slouží k informování systému, které Intenty komponenty aplikace přijímají. Každá komponenta může mít specifikováno i více filtrů. Filtrování probíhá pomocí atributů `Intentu` Akce, Data a Kategorie. Ve filtru je tedy možné pozorovat tyto 3 druhy definicí. Test na akci a na kategorii jsou dotazovány na název akce respektive kategorie. Test na data je dotazován

² MIME – zkratka pro Multipurpose Internet Mail Extensions („Víceúčelová rozšíření internetové pošty“). Internetový standard umožňující posílat e-maily s přílohou různých formátů.

na MIME typ dat a na samotná data. Pokud Intent nesplní některý z testů, není komponentě doručen. Příklad Intent filtru je zapsán v následujícím kódu [12]:

```
<intent-filter>
  <action android:name="android.intent.action.GET_CONTENT" />
  <category android:name="android.intent.category.DEFAULT" />
  <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
</intent-filter>
```

Soubor AndroidManifest

Každá aplikace musí obsahovat soubor *AndroidManifest.xml* ve svém kořenovém adresáři. Tento soubor předkládá základní informace o aplikaci systému. Systém tyto informace musí mít k dispozici ještě před prvním spuštěním aplikace. Manifest obsahuje následující informace:

- Obsahuje jméno Java balíčku, jenž je jednoznačným identifikátorem aplikace.
- Vyjmenovává jednotlivé komponenty aplikace: aktivity, služby, přijímače broadcastu a poskytovatele obsahu, z nichž aplikace sestává. U každé komponenty definuje její název a možnost, jak může být spuštěna. Tímto prostředkem jsou výše zmíněné Intent filtry.
- Popisuje oprávnění, jenž aplikace k běhu vyžaduje. Oprávnění slouží k omezení přístupu k částem kódu a datům v zařízení a mají za úkol zabránit jejich zneužití. Oprávnění se zapisují jednoznačným pojmenováním. Aplikace může specifikovat vlastní oprávnění pro ochranu svých komponent.
- Definuje minimální podporovanou verzi API, pro kterou je aplikace implementována. Aplikace poté nemůže být nainstalována na systém s nižší verzí API.
- Vyjmenovává použité knihovny (nadstandardní oproti systémovým), které aplikace využívá pro svůj běh.

Uživatelské rozhraní a zdroje

Pro oddělení externích zdrojů od aplikačního kódu slouží složka *res/* umístěná ve složce aplikace. Toto oddělení může sloužit pro využití alternativ na různých zařízeních s různou konfigurací. Tyto různé konfigurace je možné specifikovat vytvořením více podsložek daného typu zdroje. Vždy je však třeba zachovat výchozí složku *default*. Tato funkce slouží například pro lokalizování aplikace do více jazyků. Potom pro každý požadovaný jazyk je vytvořena složka obsahující lokalizační soubory. Je podporováno několik typů externích zdrojů:

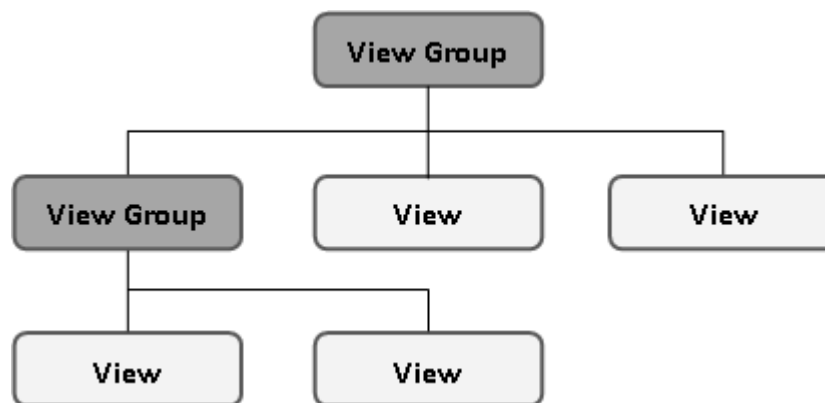
- Animace – ve složce *anim/*
- Barvy stavů – ve složce *color/*, slouží pro definování stavů a jejich barev. Např. různé barvy tlačítek při stisknutí, s fokusem, základní.
- Obrazové zdroje (drawable) – slouží především pro uložení grafiky aplikace. Tyto se nachází ve složce *drawable/*.

- Rozvržení (layout) – ve složce *layout/*, slouží pro definici uživatelského rozhraní.
- Menu – slouží pro definici položek rozbalovacího seznamu. Těchto může být v aplikaci větší množství. Je pro ně vyhrazena složka *menu/*.
- Textové řetězce – slouží převážně pro lokalizování aplikace do různých jazyků a pro formátování zobrazovaného textu. Nachází se ve složce *value/*.
- Styl – ve složce *value/*, slouží pro formátování a definování vzhledu uživatelského rozhraní.
- Další – opět uložené ve složce *value/*, mohou definovat různá XML data.

Zdroje v XML souborech musí být definovány způsobem: “@typ_zdroje/název_zdroje“ (např. @string/hello). Tím je zajištěno automatické vygenerování proměnné “R.typ_zdroje.název_zdroje“ (např. R.string.hello pro předchozí příklad), přes kterou je možné zdroje používat v aplikačním kódu.

Uživatelské rozhraní sestává z prvků tříd *View* a *ViewGroup* [6]. Prvky *View* vykreslují na obrazovku objekty, které může uživatel používat. Prvky *ViewGroup* v sobě slučují prvky *View* nebo jiné *ViewGroup* a tím vytváří rozvržení uživatelského rozhraní. Systém Android poskytuje sadu prvků odvozených od *View* a *ViewGroup*, jenž nabízí základní komponenty uživatelského rozhraní (tlačítka, vstupní pole, seznamy a další) a definování jejich rozložení (*LinearLayout*, *RelativeLayout*, a další).

Uživatelské rozhraní aktivit (a dalších komponent) aplikace sestává z hierarchie prvků *View* a *ViewGroup*, jehož příklad ilustruje **Obrázek 5**. Každá *ViewGroup* je neviditelný kontejner obsahující další prvky. Tím je možné vytvářet libovolnou hierarchii prvků, ale složitější návrh může být náročnější na výkon. Rozhraní je možné vytvářet sestavováním komponent v návrháři, který je součástí vývojářského software nebo je ručně zapisovat do XML souboru (strukturou podobné HTML). Obě možnosti je možné kombinovat. Každý prvek rozhraní vytvoří svůj XML element, jehož nastavení se provádí pomocí atributů. Prvky rozhraní je možné přes generované proměnné využívat v aplikačním kódu a měnit jejich parametry nebo jim přiřazovat akce (např. stisk tlačítka, výběr prvku seznamu, a další). Uživatelské rozhraní je



Obrázek 5: Příklad hierarchie prvků *ViewGroup* a *View*. Čerpá z [6], vlastní úprava.

možné vytvořit kompletně v aplikačním kódu, avšak takové řešení je nepřehledné. Příklad uživatelského rozhraní zapsaného pomocí XML souboru je na následujícím kódu [6]:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am a TextView" />
  <Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am a Button" />
</LinearLayout>
```

2.2 Použité knihovny

2.2.1 Google Services

Google nabízí soubor služeb [19], které je možné používat po vytváření funkčních aplikací. Služby umožňují spravovat distribuci aplikace, nahlížet na statistiky o užívání aplikace uživateli a poskytují funkce jako přihlašovací systém, mapy, cloud a další. Ačkoli nejsou přímou součástí systému Android, využívá jich většina uživatelů. Je možné je využívat na zařízeních, které obsahují oficiální aplikaci Obchodu Play³. Dostupné jsou na všechny zařízení využívající systém ve verzi 2.0 nebo vyšší a některé funkce je možno využít i na dalších zařízeních.

Mezi hlavní nabízené služby patří:

- Hry – Google Play Games Services umožňují do herních aplikací přidávat sociální prvky. Těmi mohou být achievements, žebříček skóre, real-time multiplayer a další funkce. Umožňuje přihlášení k účtu Google a sdílení postupu ve hře s ostatními.
- Mapy – Google Maps Android API umožňují přidat do aplikace mapy, zobrazit pozici na mapě, zanést do mapy vlastní data a další prvky. Více se problematice věnuje následující text.
- Lokalizace – Location API umožňuje jednoduše využívat znalosti o poloze zařízení bez vytváření vlastních služeb využívajících lokalizačních technologií. Díky optimalizaci mohou nabídnout nízkou spotřebu energie a výkonu zařízení.

³ Google Play: <https://play.google.com/store>

- Google+⁴ - Umožňuje aplikaci propojit se stejnojmennou sociální sítí. Poskytuje vývojářům funkce zabezpečeného přihlášení, přístup k uživatelskému profilu, publikování příspěvků na síti a doporučování obsahu pomocí tlačítka *+1*. Další informace jsou popsány v následujícím textu.
- Disk – Google Drive Android API přináší uživateli možnost přístupu k dokumentům uloženým na službě Disk. Hlavními přednostmi jsou automatická synchronizace dokumentů mezi vzdáleným úložištěm a zařízením uživatele. Dále poskytuje standardní uživatelské rozhraní pro procházení vzdálenou adresářovou strukturou a pro přístup k dokumentům. Poslední hlavní předností API je jednoduché vyhledávání vzdálených souborů díky filtrům kombinující různá metadata.
- Peněženka – Google Wallet umožňuje provádět platby přímo z aplikace stisknutím jednoho tlačítka. Služba je dostupná pouze na území US.
- Reklamní systém – Google Mobile Ads umožňuje vytvořit autorovi aplikace příjem na základě reklamy zobrazené v aplikaci. Reklamní systém Google je v současnosti největším poskytovatelem reklamy na internetu a v mobilních aplikacích.

Google Maps

Služba umožňuje v aplikaci zobrazit Mapy od společnosti Google Inc. Tyto jsou stejné jako ve výchozí aplikaci *Mapy Google* a API [14] poskytuje téměř stejnou funkcionalitu. Hlavními rozdíly jsou nemožnost zobrazení přizpůsobeného obsahu a nefunkčnost některých ikon zobrazených v mapě (jako zastávky apod.). Naopak API umožňuje propojení s aplikací přímo pomocí metod systému Android.

Třída poskytující mapy nese název `GoogleMap`, která poskytuje objekt map pro použití v aplikaci. Třída automaticky zajišťuje spojení s mapovou službou, stahování dlaždic mapy, jejich zobrazování na obrazovku, zobrazování pomocných ovládacích prvků a obsluhu gest pro práci s mapou, jako přiblížení a oddálení nebo posuv a další. API umožňuje tyto automatické operace a další chování řídit poskytnutím příslušných metod. Například umožňuje přepsat metodu při kliknutí na oblast mapy nebo zobrazit uživatelské ikony nad mapou.

Pro použití mapy jsou připraveny dva různé prvky uživatelského rozhraní: objekty `MapView` a `MapFragment`. Oba poskytují kontejner pro zobrazení mapových podkladů a funkcionalitu objektu Google Map. Aktivita implementující tento objekt musí obsáhnout ve svých metodách inicializaci, nastavení a odstranění příslušného objektu.

⁴ Google+: <https://plus.google.com>

Typy map

Pro zobrazení mapy je možné využít několik různých typů mapy. Jsou jimi:

- Normální – zobrazuje schéma silnic a cest, jejich názvy a další prvky, jako řeky a další.
- Satelitní – zobrazuje satelitní snímky.
- Hybridní – využívá podkladů satelitních snímků a navíc přidává zvýraznění cest a silnic.
- Terénní – zobrazuje typografická data, zvýrazňuje rozdílnou nadmořskou výšku a některé cesty.

Změnu typu mapy je možné provést pomocí metody `setMapType` s příslušným parametrem, určujícím vybraný typ. K tomuto účelu nese objekt globální konstanty, definující jednotlivé typy mapy.

Marker

Nad mapou je možné zobrazit další informace, které vyžaduje aplikace. Hlavní možností je *Marker* [13], což je značka pro pozici na mapě. Ten v základu využívá standardní ikonu, která je použita i v aplikaci Google Map. Ikonu je možné upravit libovolnou barvou nebo ji zcela nahradit jiným obrázkem. Ke značce je možné zobrazit *InfoWindow*, které může zobrazovat další informace o značce (například název, popis a další).

Značku lze do mapy přidat metodou `addMarker`, jejímž atributem je instance objektu `Marker`. Tento objekt má jeden povinný parametr, jímž je pozice v souřadném systému mapy. Další atributy slouží k přizpůsobení vzhledu značky a nastavení dalších parametrů.

Mapové API umožňuje definovat reakce na uživatelskou interakci se značkou. Je možné rozlišit dvě akce provedené se značkou:

- Kliknutí – je možné obsloužit metodou `onMarkerClickListener`, ve které se definuje obslužná akce. Bez definování je zobrazeno informační okno značky.
- Posunutí – obsluhuje metoda `onMarkerDragListener`. Značku je možné přesouvat, pokud je to v jejím nastavení povoleno.

Google+

Jak už bylo popsáno výše, Google+ Android API [19] umožňuje vývojáři přístup k funkcím sociální sítě Google+. Objekt zpřístupňující tyto funkce je instancí třídy `PlusClient`. Hlavními přístupnými funkcemi jsou:

- Zabezpečené přihlášení [18] – umožňuje využívat Google účet pro přihlášení uživatelů a tím umožnit jejich rozlišení. API nabízí přihlašovací tlačítko Google+ `SingIn` jako prvek uživatelského rozhraní ve třídě `SingInButton`. Jeho metody zajišťují přihlášení k účtu uživatele zařízení.

- Přístup k profilu [17] – z objektu `PlusClient` lze získat informace o přihlášeném uživateli. Tyto informace poskytuje třída `Person`. Přes tuto třídu je možné získat i seznam přátel uživatele, který je uložen v objektu `PersonBuffer`. Ten je kolekcí jednotlivých objektů `Person`.
- Sdílení obsahu [15] – ke sdílení obsahu je připraven `ShareIntent`, který spouští aktivitu zobrazující uživateli standardní obrazovku pro sdílení obsahu. Tu je možné předvyplnit pomocí objektu `PlusShare`, kterým se vytváří `Intent`. Je možné nastavit typ obsahu (text, video, atd.), text zprávy a odkaz URL na obsah.
- Doporučení obsahu [16] – umožňuje uživateli pomocí tlačítka `+1` doporučit obsah ostatním uživatelům. Doporučený obsah se zobrazí na síti Google+ i v příslušném obsahu. API poskytuje nativní tlačítko ve třídě `PlusOneButton`. Metody této třídy automaticky zajistí komunikaci se službou. Vývojář musí zadat URL odkaz, jehož obsah je doporučován.

2.2.2 Facebook SDK

Společnost Facebook, Inc. umožňuje vývojářům, stejně jako Google, používat vlastní API pro propojení aplikací se stejnojmennou sociální sítí. Mezi hlavní funkce patří [20]:

- Přihlášení pomocí Facebook účtu – Přihlášeného uživatele je možné identifikovat pomocí objektu `GraphUser`. Tlačítkem `loginbutton` lze získat potřebné informace o uživateli a vytvořit tak instanci objektu.
- Sdílení obsahu a stavu – Sdílení stavu je možné pomocí objektu `FacebookDialog`. Tento dialog je možné upravit pomocí speciálních parametrů, které předvyplní sdílené informace. Uživateli se zobrazí okno s možností vyplnit zprávu a změnu stavu potvrdit.
- Správa profilu – Pomocí objektu `GraphUser`, získaného přihlášením uživatele, je možné zobrazit informace o uživateli a provádět jejich změnu.
- Propojení se seznamem přátel – Přátele je možno načíst a vybírat pomocí `FriendPickerDialog`, který zobrazí seznam přátel přihlášeného uživatele. Po výběru vrací seznam uživatelů v objektech `GraphUser`.
- Zobrazení význačných míst v okolí – Místa v okolí je možné zobrazit pomocí objektu `PlacePickerFragment`. Tomu je třeba nastavit atributy `Location`, určující střed hledaného okolí, a zobrazovaný nadpis `LabelText`. Po dokončení výběru je vrácen objekt typu `GraphPlace`, ze kterého je možné získat další informace.

3 Analýza požadavků

3.1 Srovnání konkurenčních řešení

Kapitola porovnává jednotlivá konkurenční řešení. Komentuje jejich funkčnost a uživatelské rozhraní. Porovná jejich výhody a nevýhody oproti ostatním. Dále navrhne, které funkce bude užitečné použít ve vytvářené aplikaci.

3.1.1 Tripomatic

Tripomatic je webová služba⁵ nabízející turistického průvodce a záznam cesty. Službu je možné využít přes stejnojmenné aplikace pro mobilní platformy Android a iOS⁶. Aplikace poskytuje přehledné a příjemné uživatelské rozhraní, ale některé funkce chybí. **Obrázek 6** zobrazuje vzhled aplikace.



Obrázek 6: Znáznornění aplikace Tripomatic.

Služba slouží pro upořádání denního plánu cesty. Základem plánování je cesta (Trip), u níž je třeba zvolit cílovou destinaci. Poté je uživateli zobrazena nabídka s připravenými cestami. Uživatel může zvolit podle délky cesty v počtu dnech nebo podle zaměření (kultura, sport a další). Další možností je vybrat prázdnou nabídku a cestu si naplánovat podle potřeby. V předpřipravených možnostech je již naplánovaný itinerář pro jednotlivé dny. Je možné přidat

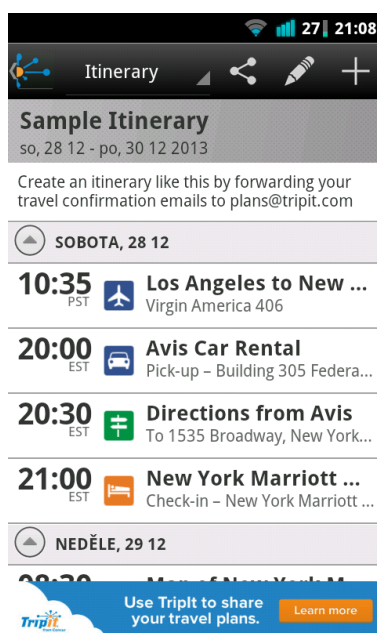
⁵ Dostupná na adrese <http://www.tripomatic.com>

⁶ iOS – Mobilní platforma společnosti Apple.

nebo odebrat některé aktivity. Přidávat je možné aktivity (kulturní památky, významná místa, atd.), ubytování a cestování. Výběr se provádí pomocí zvýrazněných bodů na mapě nebo vytvořením bodu vlastního. V aplikaci chybí možnost vyhledání místa dle názvu, možné je pouze ručně hledat v mapě. Dále neumožňuje vytvořit cestu bez známé (v databázi obsažené) destinace.

3.1.2 Tripit

Webová služba Tripit⁷ umožňuje plánovat cesty uživatele a spravovat jejich kompletní itinerář. Uživatelské rozhraní je na první pohled zmatené a je třeba delší orientace na jednotlivé funkce. Jsou dostupné mobilní aplikace pro platformy Android, BlackBerry, iOS a Windows Phone. Aplikace má jednoduché uživatelské rozhraní a všechny funkce jsou přímo dostupné. Vzhled aplikace zobrazuje **Obrázek 7**. Aplikace i webová služba jsou dostupné zdarma, ale ke zpřístupnění pokročilých funkcí (jako hlídání času odletu letadel a další) slouží placený účet.



Obrázek 7: Znáznornění aplikace Tripit.

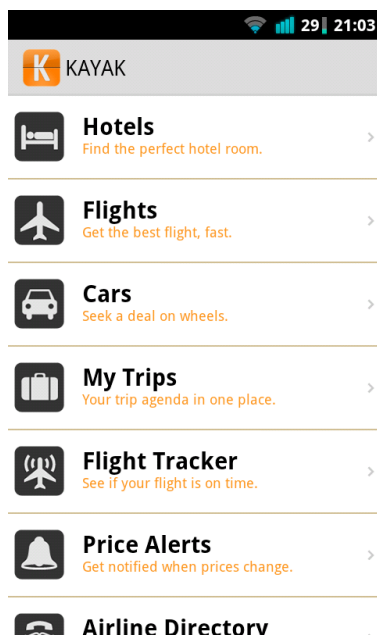
Aplikace zobrazuje správu cest (Trip), které je možné vytvářet i upravovat. Každá cesta sestává z plánu, který je tvořen jednotlivými záznamy. Je možné přidávat záznamy z kategorií: přeprava, aktivity a ostatní. U kategorie cestování jsou v nabídce cestování letadla, kde je možné zadat informace o letu, letišti, času přiletu a odletu a další. Dalšími možnostmi jsou půjčování vozu, lodní nebo železniční doprava a další. U každé je možné vyplnit informace o příjezdu, odjezdu, poskytovateli, ceně a dalších užitečných informacích. V kategorii aktivity jsou k dispozici ubytování, setkání, restaurace a jiné aktivity. U každé možnosti je k dispozici zadání informací o časech, místu a dalších. V kategorii ostatní jsou poznámka, adresa a projetá

⁷ Tripit – služba dostupná na adrese <https://www.tripit.com>

vzdálenost (z bodu do bodu). Veškeré informace je třeba zadat ručně a je možné přidávat spolucestovatele a fotky.

3.1.3 Kayak

Služba Kayak slouží především pro vyhledávání letů, hotelů, půjčoven vozidel, ale umožňuje vytvářet celé cesty a jejich itinerář. Služba má webovou⁸ a mobilní aplikaci pro platformy Android, iOS a Windows Phone. Obě varianty nabízejí přehledné uživatelské rozhraní, které poskytuje přímý přístup ke všem hlavním funkcím. Aplikaci zobrazuje **Obrázek 8**.



Obrázek 8: Znáznornění aplikace Kayak.

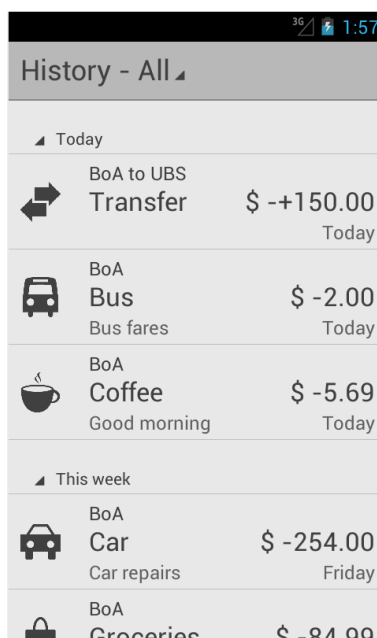
Vyhledávání letů umožňuje zadat výchozí a cílové letiště, u kterých při zadávání nabízí seznam možností, a datum odletu a příletu. Po vyhledání je zobrazena nabídka jednotlivých letů seřazených dle ceny. Zobrazeny jsou parametry jako doba letu, čas odletu a příletu, počet přestupů a další. Vyhledávání hotelů funguje na stejném principu. Po zadání města a intervalu ubytování je zobrazen seznam dostupných hotelů. Ve webové službě je zobrazena stránka s výsledky ze služby Booking.com, která umožňuje přímé zarezervování vybraného hotelu. V aplikaci jsou výsledky zobrazeny přímo a informují o ceně a hodnocení hotelu. Hledání půjčoven vozidel opět funguje shodně. Po zadání města je zobrazen seznam vozů od různých společností. Seznam je řazen dle ceny a zobrazí všechny potřebné parametry. Sestavení cestovního plánu je u této služby druhořadá funkce. Cestu je možné tvořit pomocí předchozích funkcí aplikace, ale je možné přidávat i vlastní. Mobilní aplikace dále nabízí funkce jako zobrazení detailu o konkrétním letu, rychlé zobrazení kontaktních informací o letišti, převodník měn, seznam věcí k zabalení a další.

⁸ Webová služba Kayak je dostupná na adrese <https://www.kayak.com>

3.1.4 Money Tab

*Money Tab*⁹ je mobilní aplikace sloužící uživateli pro správu financí. Umožňuje přidávat příjmy a výdaje. Záznamy je možné třídit do kategorií, které uživatel vytvoří. Je možné spravovat více různých účtů a tedy i převody mezi nimi. Aplikace umožňuje vytvoření pravidelných transakcí (plat, nájem, atd.) a rychlých zkratk, pro nejčastější výdaje. Správa historie transakcí umožňuje zobrazení podle týdnů. Statistiku je možné zobrazit dle dne, týdne, měsíce, čtvrtletí a roku. Je zobrazen počáteční stav, celkové příjmy a výdaje a konečný stav účtu. Dále je možné zobrazit graf zobrazující stav účtu v průběhu měsíce a seznam transakcí s grafickým znázorněním částky.

Aplikace nabízí přehledné uživatelské rozhraní (Obrázek 9), které má strukturu podobnou originálním aplikacím systému Android. Všechny funkce jsou jednoduše dostupné a aplikace nabízí široké možnosti nastavení. Umožněno je exportování transakcí do souboru csv a možnost synchronizace s Google účtem. Nevýhodou aplikace je pouze placená verze a absence češtiny.



Obrázek 9: Obrazovka aplikace Money Tab.

⁹ Money Tab, odkaz ke stažení:

<https://play.google.com/store/apps/details?id=ch.talionis.mobile.moneytab>

4 Specifikace a návrh aplikace

4.1 Funkce aplikace

V kapitole budou uvedeny všechny zvolené funkce. Mezi něž patří cestovní deník, zobrazení mapy, překladač, převodník měn. Pro každou funkci bude popsán způsob jejich fungování na obecné úrovni. V další části kapitoly je popsáno uživatelské rozhraní jednotlivých funkcí a návrh databáze pro uložení statických dat.

4.1.1 Cestovní deník

Aplikace poskytuje uživateli možnost vytvářet deník, který zaznamenává jeho cesty. Uživatel má možnost přidávat a upravovat cesty. Cestu tvoří aktivity v časovém sledu. Uživatel může zadat jméno aktivity a vybrat její kategorii. Kategorie aktivit jsou:

- Přprava,
- Ubytování,
- Kultura,
- Jídlo,
- Sport,
- Schůzka,
- Další.

Dále uživatel zadává umístění aktivity několika způsoby. První možností je výběr aktivity na mapě, kdy je umístění ukládáno jako zeměpisné souřadnice. Další možností je vybrat umístění aktivity z databáze uložených míst. Obě tyto možnosti vytvoří v databázi nový záznam.

Jednotlivé cesty deníku umožňuje aplikace sdílet na sociální síti Facebook a Google+, odkud je mohou prohlížet přátelé uživatele. Aplikace dovoluje uživateli zálohovat cesty na externí paměť zařízení nebo na online úložiště Disk Google.

4.1.2 Místa v okolí

Tato funkce aplikace umožňuje uživateli rychle najít zajímavé aktivity v jeho okolí. Tyto aktivity jsou zobrazeny na mapě včetně pozice uživatele. Aplikace umožňuje filtrování aktivit, je možné vybrat, které kategorie budou zobrazeny. Aplikace umožňuje uživateli zjistit detaily o vybraném místě. Místa budou načítána z internetové databáze a ukládána do aplikace. Tato možnost uživateli zajistí informace o navštívených místech i bez připojení k internetu. Místa může uživatel ukládat do oblíbených a přidávat hodnocení. Dále aplikace dovoluje vyhledat zajímavá místa dle názvu.

4.1.3 Převodník měn

Převodník měn slouží uživateli k převodu cizí měny na jeho domácí dle aktuálního kurzu. Aplikace tedy umožňuje zvolit zdrojovou a cílovou měnu. Cílová měna je automaticky nastavována podle lokalizace zařízení. Následně aplikace pomocí dotazu na internetovou službu převede požadovanou částku do zvolené měny.

4.1.4 Překladač

Funkce překladače je jasně daná. Vstupní zadaný text převede na přeložený dle nastavených parametrů. Aplikace umožní zadání vstupního i výstupního jazyku. Výstupní jazyk je, stejně jako v případě převodníku měn, automaticky nastaven dle preferencí zařízení. Po stisku tlačítka pro přeložení dojde k překladu. Dotaz na přeložení se posílá na webovou službu, jejíž výsledek je zobrazen uživateli.

4.1.5 Kontrola výdajů

Aplikace umožňuje uživateli ukládat výdaje, které během cesty zaznamenal a dovoluje je uživateli spravovat a mít tak detailní přehled o utracených financích. U každého záznamu je možné zadat název a jeho typ. Typy výdajů odpovídají kategoriím aktivit cesty. Hlavním parametrem je částka a měna výdaje a přiřazení k některé z vytvořených cest. Aplikace umožňuje výdaje filtrovat podle různých parametrů (např.: minimální částka, typ výdaje, datum, atd.). Stejně tak aplikace umožňuje zobrazit součet výdajů cesty, který automaticky převádí do lokální měny.

4.1.6 Seznam pro balení

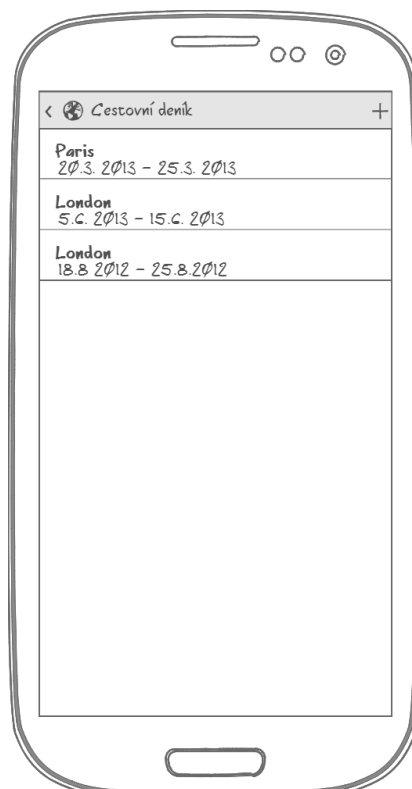
Balící seznam může cestovateli ušetřit spoustu starostí. Aplikace v základu obsahuje podstatné položky, které jsou v dnešní době pro cestování nejdůležitější. Uživatel může přidávat i položky vlastní. Každá položka je po vybrání označena, aby uživatel snadno rozlišil, které položky má nebo nemá sbaleny. Aplikace uživateli nabízí uložit stav balení pro pokračování později a vymazání tohoto stavu.

4.2 Návrh uživatelského rozhraní

Hlavním vstupním bodem aplikace je uživatelské rozhraní. To musí být navrženo s cílem na dotykové ovládání a mělo by poskytnout více jazykových verzí. Uživatelské prostředí aplikace je popsáno pro každou obrazovku aplikace. Znázorněno je schematicky jednoduchým obrázkem. U každé obrazovky je popsáno, jakých komponent využívá. V návaznosti na předchozí



Obrázek 11: Výchozí obrazovka aplikace.



Obrázek 10: Obrazovka cestovního deníku.

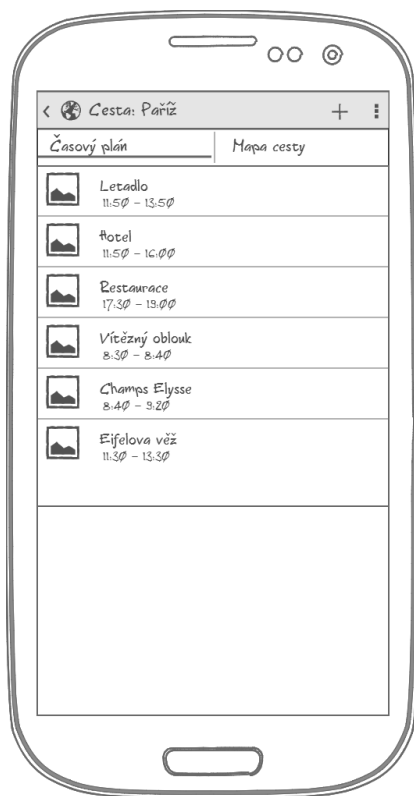
podkapitoly o funkcích aplikace, jsou tyto zmíněny z pohledu vytvoření uživatelského rozhraní. Kapitola dále zmiňuje, jakým způsobem je možné mezi jednotlivými obrazovkami přecházet a znázorňuje totéž pomocí diagramu.

Aplikaci tvoří několik obrazovek uživatelského rozhraní. Každá z předchozích funkcí je zastoupena minimálně jednou. Výchozí obrazovku (viz [Obrázek 11](#)) tvoří tlačítka umožňující uživateli přístup k jednotlivým funkcím aplikace. Možnost přechodů mezi obrazovkami je naznačena na diagramu ([Obrázek 18](#)).

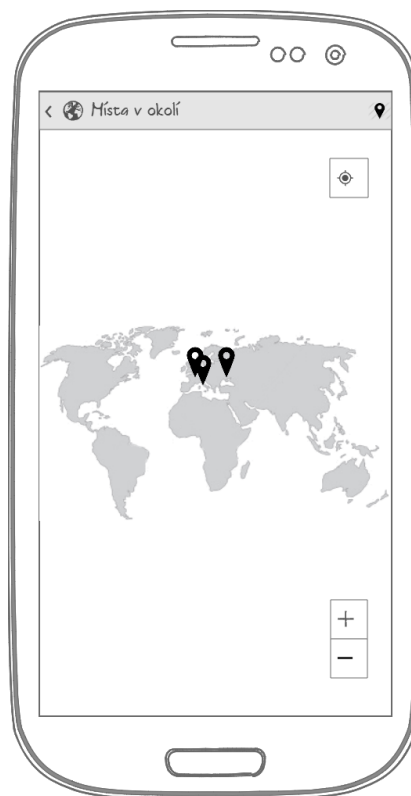
Cestovní deník

Cestovní deník je rozdělen na několik obrazovek. První obrazovka ([Obrázek 10](#)) v pořadí po zvolení této funkce aplikace je seznam uložených cest. Cesty jsou uspořádány dle přidání a každý záznam tvoří jméno cesty a její časový interval, tedy počáteční a koncové datum. Dále obrazovka obsahuje tlačítko pro přidání cesty. Po stisknutí tohoto tlačítka je uživateli zobrazena obrazovka sloužící pro zadání výchozích údajů, tedy názvu počátečního a koncového data a popisu. Navíc obsahuje tlačítko pro uložení. Dále je umožněno editovat a mazat cesty přes kontextové menu seznamu. Dalšími funkcemi v tomto seznamu jsou sdílení a zálohování cesty.

Další obrazovkou ([Obrázek 12](#)) je zobrazení časového plánu aktivit cesty nebo jejich zobrazení na mapě. Na tuto obrazovku je uživatel přesměrován při dokončení nové cesty a při vybrání cesty ze seznamu. Mezi oběma typy zobrazení aktivit je možné přepínat pomocí záložek. První typ zobrazení obsahuje dle času uspořádané aktivity, u nichž jsou zobrazeny název,



Obrázek 12: Obrazovka časového plánu cesty.



Obrázek 13: Obrazovka pro místa v okolí.

počáteční a koncový čas a obrázkem je vyznačen typ aktivity. Druhý typ zobrazení obsahuje mapu, na které jsou vyznačené jednotlivé aktivity. Po stisku jejich značky jsou zobrazeny detaily o aktivitě. V záhlaví obrazovka obsahuje název zvolené cesty, tlačítko pro přidání nové aktivity a tlačítko pro sdílení cesty na sociálních sítích.

Přidávání nové aktivity umožňuje obrazovka, jejímž obsahem jsou pole pro zadání názvu, výběr typu aktivity, nastavení data a obou časů a tlačítka pro zadání umístění. To je možné přes pole adresa nebo výběr z mapy. Výběr aktivity z mapy je umožněn přes obrazovku, která je shodná s obrazovkou pro funkci Místa v okolí (popis dále). Na závěr obrazovka obsahuje tlačítko pro uložení.

Místa v okolí

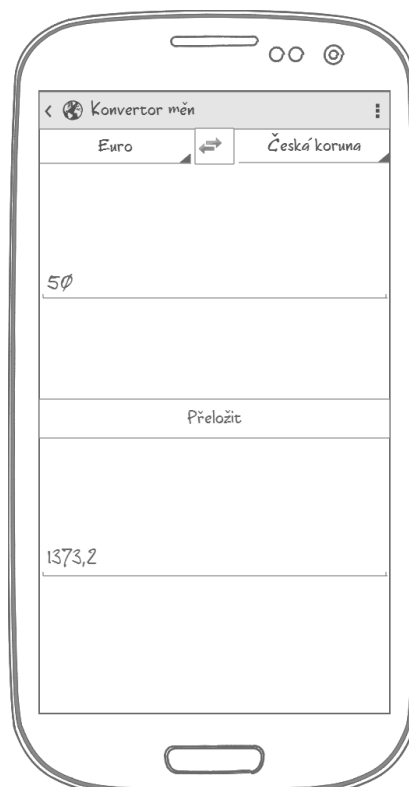
Funkce pro zobrazení okolních aktivit je tvořena jedinou obrazovkou (Obrázek 13). Tuto z většiny vyplňuje mapa, na které jsou zobrazeny jednotlivé aktivity a pozice uživatele. Dále obsahuje tlačítka pro interakci s mapou, jako je přibližování, oddalování a vycentrování na pozici uživatele. Posledními funkcemi jsou tlačítko pro vyhledávání míst v okolí dle kategorie a změna typu zobrazené mapy.

Překladač, Převodník měn a Balící seznam

Obrazovka překladače (viz Obrázek 15) obsahuje vstupní pole pro zadání překládaného textu, pole pro přeložený text a dvě tlačítka pro zvolení vstupního a výstupního jazyka. Po jejich



Obrázek 15: Obrazovka překladače.



Obrázek 14: Obrazovka převodníku měn.

stisknutí se zobrazí seznam dostupných jazyků, ze kterých může uživatel vybírat. Vstupní a výstupní jazyk je možné rychle přepnout pomocí dalšího tlačítka. Konečně tlačítkem pro přeložení dojde k očekávané akci.

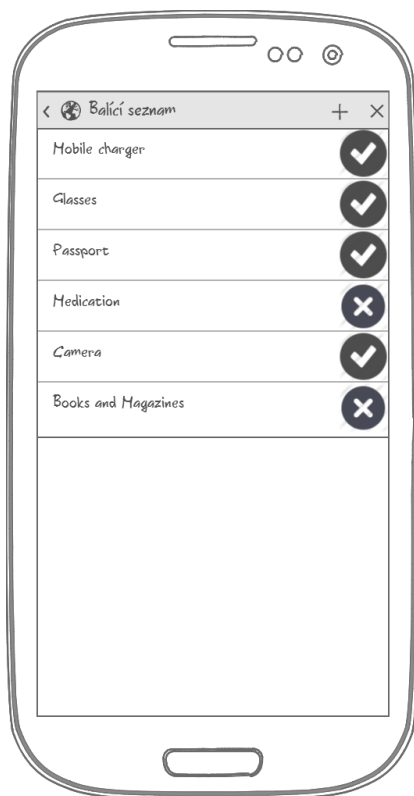
Podobně funguje převodník měn. Obrazovka (Obrázek 14) opět obsahuje 2 textová pole, jedno pro zadání vstupní částky a druhé pro vypočítanou částku. Dále dvě tlačítka pro zvolení výchozí a cílové měny a tlačítko pro jejich záměnu. Opět po stisku tlačítek pro změnu měny je zobrazen seznam. Poslední tlačítko slouží pro vykonání převodu.

Balící seznam tvoří jednotlivé položky. Ty jsou reprezentovány svým názvem a políčkem pro zaškrtnutí. Dále obrazovka obsahuje tlačítka pro přidání položky seznamu, uložení výběru a vymazání výběru. Návrh obrazovky ilustruje Obrázek 17.

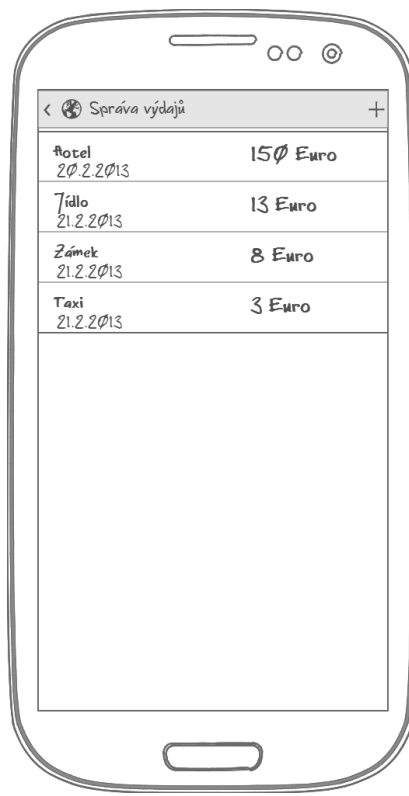
Kontrola výdajů

Kontrolu výdajů tvoří dvě obrazovky. První z nich zobrazuje seznam výdajů pro zvolenou cestu. Název cesty je zobrazen v záhlaví obrazovky, ve kterém jsou dále zobrazena tlačítka pro přidání nového výdaje, změnu cesty, filtraci výdajů podle různých parametrů a zobrazení součtu výdajů. Dále obrazovka obsahuje seznam výdajů cesty. Seznam zobrazuje název a datum výdaje a jeho částku. Typ výdaje je zobrazen zástupným obrázkem. Obrazovku znázorňuje Obrázek 16.

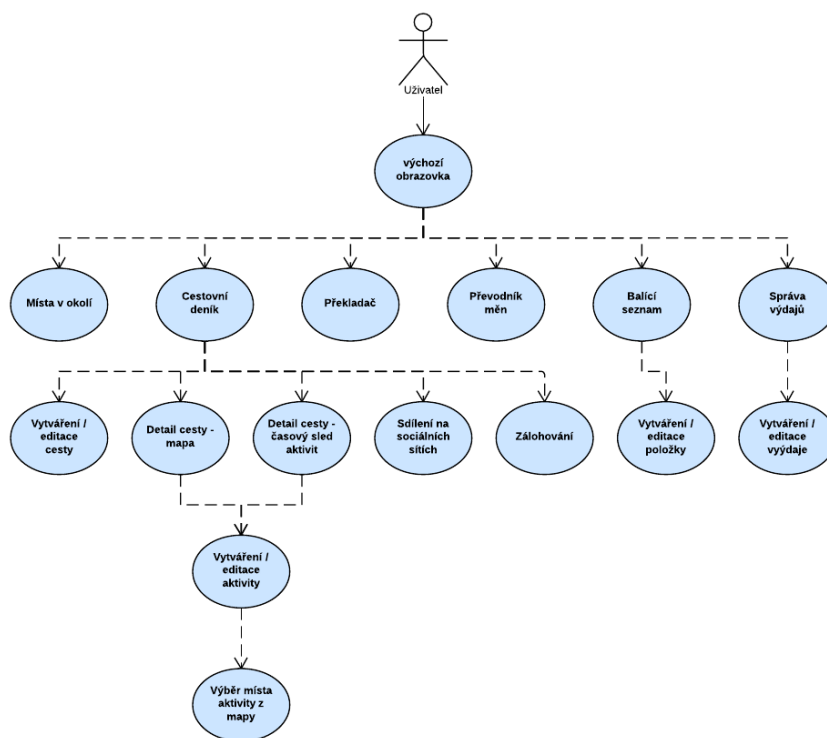
Druhou obrazovkou je detail výdaje, kde je možné údaje editovat. Zobrazeny jsou jednotlivé položky pro zadání hodnot. U položek typ výdaje a měna jsou zobrazeny rozbalovací seznamy pro výběr připravené hodnoty. V záhlaví obrazovky se nachází tlačítko pro uložení.



Obrázek 17: Obrazovka balíčního seznamu.



Obrázek 16: Obrazovka seznamu výdajů cesty.



Obrázek 18: Diagram přechodů mezi obrazovkami aplikace.

4.3 Návrh databáze

Pro uložení dat bude aplikace využívat databázi. Jak je popsáno výše, OS Android využívá databázový systém SQLite. V kapitole jsou navrženy jednotlivé tabulky databáze a popsána data, která jsou v tabulce uložena. Celá databáze je znázorněna ER diagramem (Obrázek 19).

Ukládaná data tvoří záznamy o cestách uživatele. Cestu tvoří název, datum začátku a konce a volitelný popis. Tyto data obsahuje tabulka *Journey*.

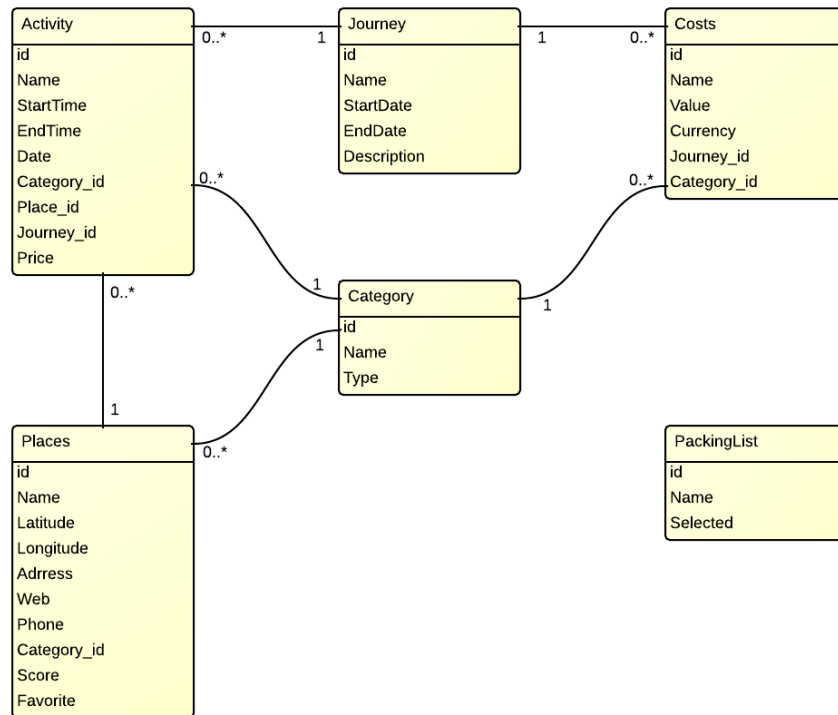
Cesta je tvořena záznamy o aktivitách cestovatele. Každá aktivita má svůj název, typ aktivity (odkaz do tabulky *Category*), čas začátku aktivity a její dobu trvání a datum aktivity. Aktivity jsou zadávány pomocí adresy nebo z pozice na mapě nebo z uložených míst v tabulce *Places*. Pokud tabulka záznam o vybraném místě neobsahuje, záznam se přidá a do tabulky se vloží odkaz na vytvořený záznam. Aktivita musí mít zaznamenáno, ke které cestě náleží, tedy odkaz na příslušný záznam do předchozí tabulky. Data o aktivitách jsou uloženy v tabulce *Activity*.

Tabulka *Places* ukládá místa zájmu. Ukládané informace jsou identifikátor, název, GPS pozice zadaná souřadnicemi zeměpisné šířky a délky, adresa, webová adresa místa, kontaktní telefon a kategorie místa. Dále se k místu ukládá uživatelské hodnocení a příznak, zda se jedná o oblíbené místo.

Aby bylo možné rozlišit různé typy aktivit a míst (doprava, kultura, ubytování, sport, ...), obsahuje databáze tabulku *Category*. Záznam tabulky tvoří identifikátor, název kategorie a její typ. Typ určuje oblast použití kategorie dle aktivity, místa nebo výdaje.

Stejně jako aktivity cesty jsou výdaje ukládány do databáze, konkrétně do tabulky *Costs*. Její záznamy tvoří identifikátor, název položky, její cena a měna a časový údaj. Tyto údaje doplňuje odkaz na tabulku cest, pro přiřazení výdaje k cestě a odkaz na kategorii výdaje do příslušné tabulky.

Dále jsou v databázi uloženy položky seznamu pro balení. Záznam v databázi tvoří název položky a informace o stavu zabalení. Údaje jsou v tabulce *PackingList*.



Obrázek 19: ER diagram databáze.

5 Implementace

Tato kapitola popisuje jednotlivé funkce aplikace z pohledu implementace. U každé funkční části jsou popsány jednotlivé třídy, které je implementují. Pro tyto funkční bloky je pro lepší pochopení znázorněn diagram tříd. Pro větší přehlednost a skrytí některých implementačních detailů nebudou diagramy tříd obsahovat vždy úplně všechny atributy a metody. Na závěr kapitoly bude popsáno uživatelské rozhraní aplikace.

Implementovaná aplikace byla nazvána TravelHelper, kdy tento název jasně říká, k čemu aplikace slouží. Aplikace byla implementována pro Android API od verze 10, tedy systému verze 2.3 nebo vyšší (viz [Tabulka 1](#)). Pro využití některých funkcí využívá aplikace systémových oprávnění. Jsou jimi:

- Internet – oprávnění využívat internetové připojení.
- Access network state – oprávnění zjišťovat stav sítě.
- Write external storage – oprávnění zapisovat na externí úložiště zařízení.
- Access coarse location – oprávnění získávat přibližnou polohu zařízení.
- Access fine location – oprávnění získávat přesnou polohu (GPS) zařízení.
- Read gservices – oprávnění zjišťování přítomnosti knihovny Google Play Service.

V příloze je k dispozici struktura balíčků zdrojových kódů projektu, jehož název je stejný jako aplikace, tedy TravelHelper.

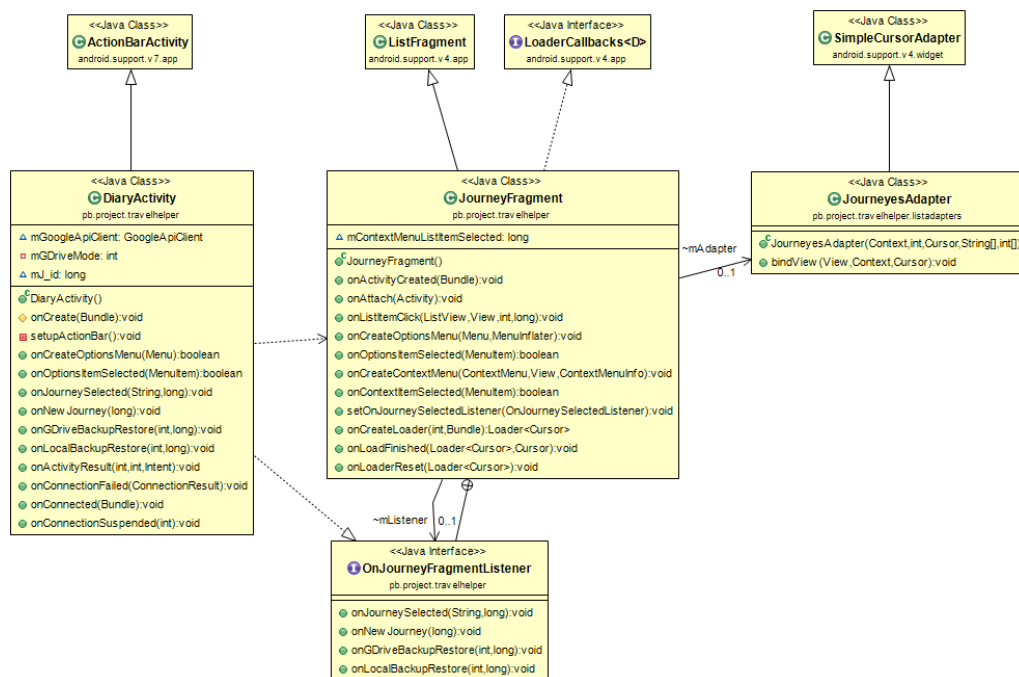
5.1 Cestovní deník

Funkce cestovního deníku je v aplikaci rozdělena dvěma obrazovkami. Stejné rozdělení lze využít i při popisu tříd implementujících tuto funkci. Dále budou jako samostatné části popsány sdílení cest a jejich zálohování.

5.1.1 Seznam cest

Hlavní třídou seznamu cest je `DiaryActivity`, která implementuje třídu `ActionBarActivity`, jenž je speciální implementací základního stavebního kamene Android aplikací aktivity (viz část kapitoly [2.1.2](#)). Diagram tříd zobrazuje [Obrázek 20](#).

Hlavní část této aktivity tvoří seznam cest. Práci s tímto seznamem zajišťuje vlastní třída `JourneyFragment`, která implementuje metody třídy `ListFragment`. Tato třída automaticky zajišťuje zobrazení seznamu a nabízí metody pro jeho obsluhu. Mezi nejdůležitější z těchto metod patří metoda `onListItemClick()`, která umožňuje reagovat na výběr položky seznamu. `Fragment` nad položkami seznamu poskytuje kontextové menu, díky kterému je umožněno uživateli upravovat, mazat a zálohovat cesty. Kontextové menu je vytvářeno metodou



Obrázek 20: Diagram tříd cestovního deníku.

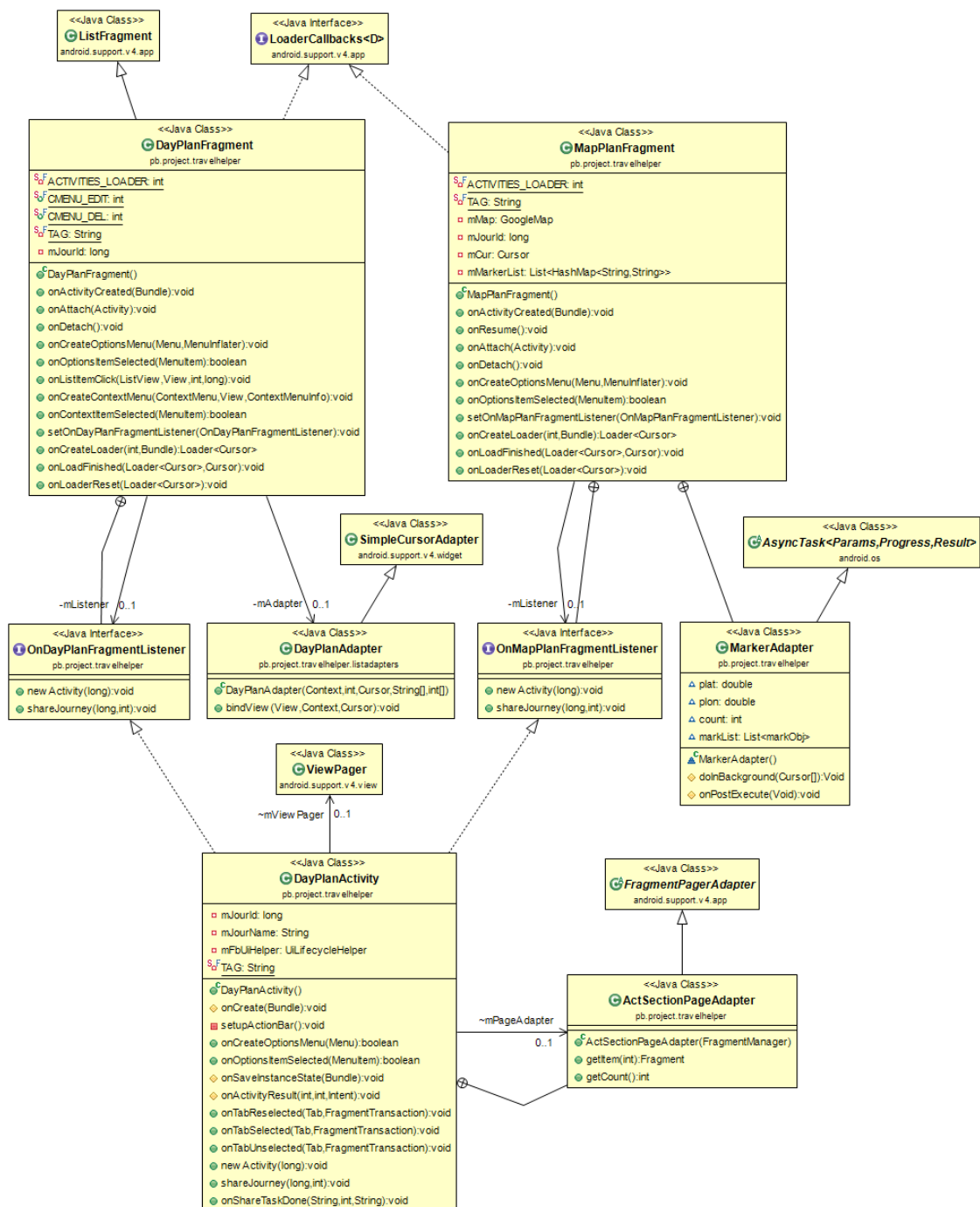
`onCreateContextMenu()`, ve které je menu vytvořeno z XML souboru definovaného v části zdrojů. Obsluhu akcí menu zajišťuje metoda `onContextItemSelected()`.

Pro vyplnění seznamu daty jsou využívány metody rozhraní `LoaderCallbacks`, které fragment implementuje. Toto rozhraní přijímá data z databáze poskytované třídou `DBContentProvider` (kapitola 5.8). Data musí být vyžádána na základě odkazu `Uri`, který specifikuje, ze které tabulky databáze jsou data čtena. Inicializaci rozhraní poskytuje objekt `LoaderManager` metodou `initLoader()`. To po získání dat v podobě objektu `Cursor` aktualizuje data seznamu. Jejich přímé zobrazení obsluhuje třída `JourneysAdapter`, jenž je implementací třídy `SimpleCursorAdapter`. Tato třída metodou `bindView()` nastaví grafickou podobu řádku seznamu a jeho položky vyplní daty.

Pro komunikaci fragmentu s jeho nadřazenou aktivitou je poskytováno rozhraní `onJourneyFragmentListener`. Implementací jeho metod musí aktivita reagovat na události z fragmentu. První z těchto metod `onJourneySelected()`, zajišťuje přechod do další části aplikace. Zde se jedná o zobrazení detailů cesty při jejím vybrání ze seznamu. Metodou `onNewJourney()`, reaguje aktivita na požadavek vytvoření nové nebo úpravu stávající cesty. Zde je opět spouštěna další část aplikace, již se věnuje kapitola 5.2. Taktéž obsluhuje zbývajících metod rozhraní se věnují další kapitoly.

5.1.2 Detail cesty

Podobně jako předchozí část, i implementace třídy detailu cesty sestává z nosné aktivity `DayPlanActivity` (viz Obrázek 21). Tato třída implementuje rozhraní `TabListener` a



Obrázek 21: Diagram tříd plánu cesty.

využívá třídu ViewPager. Kombinace jejich využití dává uživatelskému rozhraní aktivity podobu záložek. O obsluhu jejich přepínání se stará metoda zmíněného rozhraní `onTabSelected()`.

Vyplnění obsahu záložek zajišťuje třída `ActSectionPagerAdapter`, která je implementací třídy `FragmentPagerAdapter`. Z této třídy je využívána metoda `getView()`, která dle zvolené záložky přiřadí jejímu obsahu fragment. Tyto fragmenty jsou `DayPlanFragment` a `MapPlanFragment`. Oba zobrazují detail cesty různým způsobem.

Zobrazení časového plánu

Plán cesty uspořádaný podle času do podoby seznamu zobrazuje první z fragmentů. Stejně jako u seznamu cest se i zde jedná o implementaci třídy `ListFragment` a rozhraní `LoaderCallback`. K vyplnění seznamu daty je definována třída `DayPlanAdapter`, zpracovávající data z objektu `Cursor`.

`Fragment` poskytuje pro komunikaci s aktivitou rozhraní `OnDayPlanFragmentListener`, poskytující metodu `newActivity()`. Ta slouží aktivitě pro reakci na požadavek vytváření nové nebo editaci stávající aktivity.

Zobrazení na mapě

`Fragment` zobrazující jednotlivé aktivity na mapě implementuje třídu `SupportMapFragment`, od které získává podporu pro zobrazení. Detaily pro obsluhu a interakci s mapou popisuje kapitola 5.3. `Fragment` dále implementuje rozhraní `LoaderCallback` pro získávání dat z databáze. Pro zobrazení dat na mapě slouží třída `MarkerAdapter` implementující třídu `AsyncTask`. Tato třída umožňuje běh kódu metody `doInBackground()` na pozadí. Metodou `onPostExecute()` jsou výsledky zobrazeny.

Stejně jako předchozí `fragment` i zde je aktivitě poskytováno rozhraní `OnMapFragmentListener`. Rozhraní poskytuje shodné metody a v aktivitě tak může být kód společný pro obě implementace.

5.1.3 Sdílení

Sdílení cesty je implementováno třídou `ShareTask`, dědící metody třídy `AsyncTask` (viz [Obrázek 22](#)). Třída přijímá dva parametry identifikátor cesty a typ sdílejší služby. Hlavním funkčním bodem třídy je metoda `doInBackground()`. Metoda pomocí objektu `ContentResolver` získá databázová data dané cesty a jejích aktivit. Těmito daty zaplňuje instanci objektu `JSONObject`.

Takto vytvořená data odesílá pomocí HTTP dotazu metodou `POST`. K tomuto slouží objekt `HttpPost`, kterému se metodami nastaví formát hlavičky a vstupních dat. Hlavním nastavovaným parametrem je URL adresa vzdáleného serveru. Dotaz odesílá objekt `HttpClient` a odpověď ukládá do `HttpResponse`. Data odpovědi jsou zpracovávána metodou třídy `convertStreamToString()`. Z této odpovědi je generován výstupní parametr, kterým je odkaz určený pro sdílení.

Třída jako prostředek pro oznámení o dokončení zpracovávání a poskytnutí výsledku poskytuje rozhraní `ShareTaskListener`. Třídy využívající sdílení musí implementovat metodu `onShareTaskDone()` tohoto rozhraní. V implementaci této metody je rozhodnuto, která služba bude pro sdílení využita.

Google+

Jak již bylo popsáno výše je pro sdílení na sociální síť Google+ využíván objekt `PlusShare`. Jeho metodami `setType()`, `setText()` a `setContentUrl()` je možné vyplnit údaje o sdílené cestě. Metodou `getIntent()` je získán objekt `Intent`, jenž je použit jako parametr metody `startActivityForResult()`. Tato standardní systémová metoda spustí novou aktivitu pro sdílení obsahu.

Facebook

Sdílení na sociální síť Facebook využívá výše představenou oficiální knihovnu Facebook SDK. Pro publikování obsahu na této sociální síti je využit objekt `ShareDialog`. Metodami `setName()`, `setDescription()` a `setLink()` jsou nastaveny stejné parametry sdílení jako v předchozím případě. Samotné sdílení je provedeno metodou `present()`, která jako návratovou hodnotu používá objekt `PendingCall`. Z těchto dat je možné určit objektem `UILifecycleHelper`, zda sdílení bylo úspěšné či nikoliv.

Webová aplikace

Pro zobrazení sdílených cest je vytvořena webová aplikace¹⁰. Úkolem této stránky je přehledně zobrazit plán cesty včetně vizualizace aktivit na mapě. Stránku tedy tvoří záhlaví, ve kterém jsou zobrazeny informace o cestě, dále panel s mapou a seznam aktivit cesty.

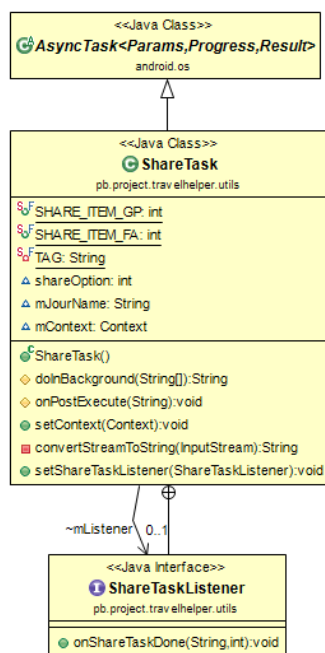
Data cesty jsou stránce předána ve formátu JSON (viz kapitola 5.8.1) a jsou zpracovávána kódem v jazyce Javascript ve funkci `load()`. Tato funkce postupně vytvoří jednotlivé aktivity cesty a vyplní jejich obsahem seznam. Položky seznamu navíc slouží jako odkaz pro zobrazení podrobností o místě nad mapou.

Jako mapové podklady jsou využívány, stejně jako v aplikaci, Mapy Google, které jsou přístupné přes javascript API¹¹. Díky tomu je možné na stránku umístit rámec s mapou a knihovna se automaticky postará o zobrazení mapových podkladů, interakci s mapou a její ovládání. Samotný objekt mapy je vytvářen pomocí konstruktoru `google.maps.Map`, ve kterém je možné nastavit parametry mapy. Těmi mohou být střed pohledu, úroveň přiblížení a typ mapy, včetně nastavení stylu jeho přepínání.

O přidávání značek reprezentujících aktivity cesty se stará funkce `createMarker()`, která jako parametry přijímá načtené hodnoty o aktivitě. Pomocí těchto parametrů jsou poté vyplněny informace o značce. Značka je vytvářena objektem `google.maps.Marker`, s parametrem nesoucím pozici bodu. Při vybrání bodu nebo při vybrání položky seznamu je zobrazeno pole s detaily o místě. Toto pole je vytvářeno objektem `google.maps.InfoWindow`,

¹⁰ Adresa webu: <http://eva.fit.vutbr.cz/~xblatn03/DP/dp.html?jid=1>

¹¹ Google Maps Javascript API v3: <https://developers.google.com/maps/documentation/javascript/>



Obrázek 22: Digram tříd sdílení cesty.

a jako parametr je mu předáván *html* kód popisující obsah pole. Následně je okno zobrazeno metodou `open()`, jejímiž parametry jsou objekt mapy a značky.

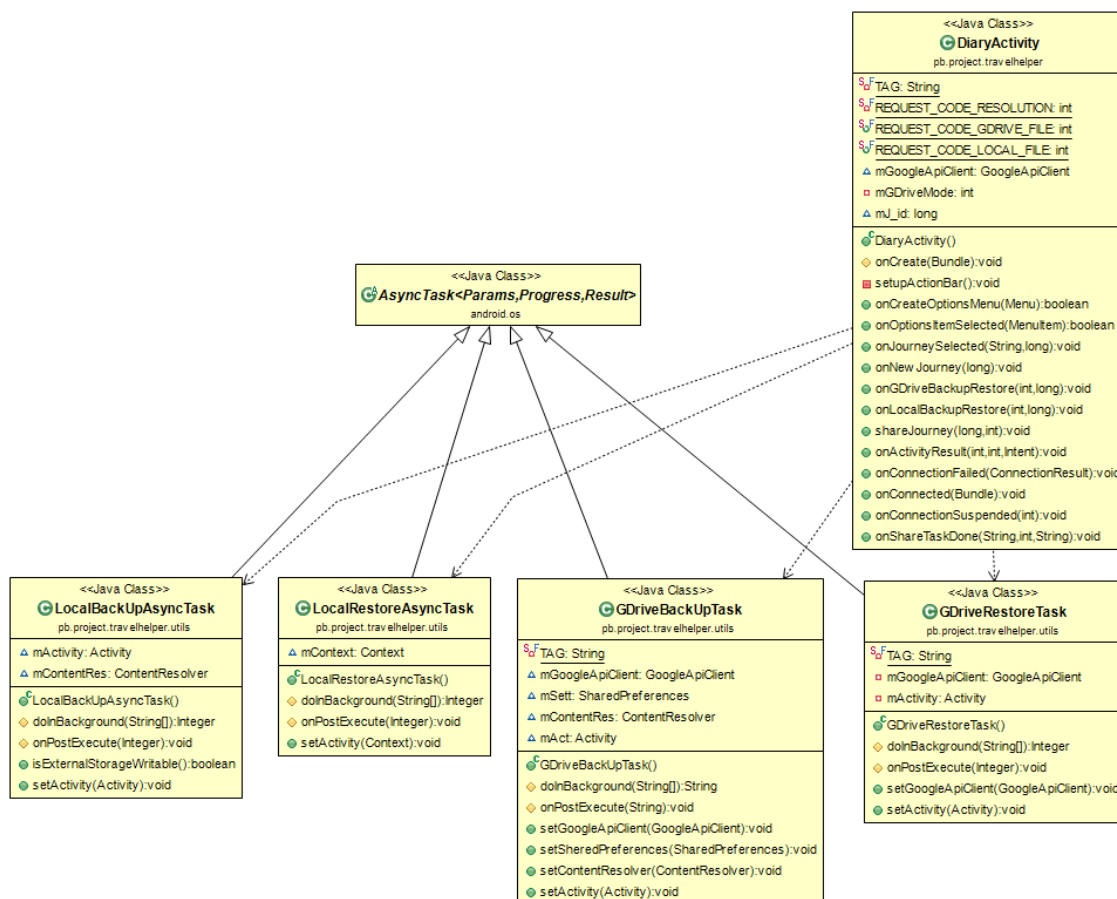
5.1.4 Zálohování

Mezi další funkci cestovního deníku patří zálohování cest, které je rozlišeno dvěma způsoby uložení. Prvním způsobem je uložení cesty do souboru na SD kartu telefonu a druhým způsobem je vytvoření souboru na internetovém úložišti Google Drive. Zálohování je spouštěno ze třídy `DiaryActivity`, která je obrazovkou aplikace se seznamem cest (viz výše). Stejně tak je v aplikaci umožněno obnovení cesty ze zálohy. Diagram tříd implementujících zálohování zobrazuje [Obrázek 23](#).

Lokální (SD Karta)

Lokální zálohování je prováděno na SD kartu do složky s názvem aplikace (`TravelHelper`). Pokud tato složka neexistuje, aplikace ji automaticky vytváří metodou `mkdir()` třídy `File` s parametry cestou k nadřazené složce a názvem vytvářené složky. Zálohování je implementováno ve třídě `LocalBackUpAsyncTask`, obnovování potom ve třídě `LocalRestoreAsyncTask`. Obě tyto třídy jsou implementací třídy `AsyncTask` a tedy pro vykonávané operace využívají její metody `doInBackground()` a pro zobrazení výsledků metody `onPostExecute()`.

Proceduru zálohování lze rozdělit na tři pod-části: načtení dat, příprava dat pro zápis a uložení souboru. Načtení dat z databáze je prováděno objektem `ContentResolver` metodou `query()` a data jsou vrácena v podobě objektu `Cursor`. Takto získaná data jsou zapisována do



Obrázek 23: Diagram tříd zálohování cesty.

objektu `OutputStream` ve formátu JSON. Soubor je vytvářen konstruktorem objektu `File`, jehož parametry jsou cesta k souboru a název souboru. Tomuto souboru se předávají data v objektu `OutputStream`, jehož metodou `flush()` se provede zápis.

Procesem opačným je obnovení zálohy. Uživateli je nabídnuto vybrání požadovaného souboru. K tomuto účelu je metodou `startActivityForResult()` spouštěn správce souborů (je předpokládána přítomnost jeho aplikace v zařízení). Výsledkem dotazu je cesta k vybranému souboru, která je předána objektu `File`. Pro načtení obsahu souboru slouží objekt `InputStream`, který je zpracován objektem `BufferedReader`. Metodou `readLine()` jsou postupně získána data o cestě. Tato data jsou nyní uložena do databáze aplikace metodou `insert()` objektu `ContentResolver`.

Disk Google

Druhou možností zálohování je využití vzdáleného úložiště Google Disk¹². Přístup k funkčnímu rozhraní je na platformě Android přístupný skrze základní knihovnu Google Services (viz kapitola 2.2.1). Knihovna pro přístup k úložišti poskytuje objekt `GoogleApiClient`. Obsluhující aktivita musí implementovat metody (`onConnected()` a

¹² Google Disk: <https://drive.google.com/>

`onConnectedSuspended()`) rozhraní `ConnectionCallbacks` pro připojení a odpojení klienta. Po připojení klienta je možné dotazovat vzdálené soubory pomocí objektu třídy `Drive.DriveApi`, který umožňuje vyhledávat, vytvářet, prohlížet a upravovat soubory a složky a jejich obsah. Každý soubor i složka na Google Disku sestávají z metadat a samotného obsahu. Stejným způsobem se k souborům přistupuje. Nejdříve je třeba vyplnit metadata, jako typ souboru a jeho název, a následně je možné vyplnit obsah daty.

Samotný proces zálohování je podobný lokálnímu zálohování. Stejným způsobem jsou načítána a připravována data na odeslání. Soubory s cestami jsou ukládány opět do složky se jménem aplikace umístěné v kořenovém adresáři uživatelského prostoru. Při vytváření složky jsou metadata zapisována do objektu `MetadataChangeSet` a samotná složka je vytvořena metodou `createFolder()`. Vytváření souboru s daty cesty je řešeno stejným přístupem, ale metoda pro vytvoření má název `createFile()` a navíc přijímá v parametru objekt s daty obsahu souboru v podobě objektu `ContentResult`. Oběma metodám je nutné předat instanci klienta, kterým je realizováno připojení na vzdálené servery. Po dokončení zálohování je do aplikační databáze uložen identifikátor právě vytvořeného souboru.

Obnovování cesty funguje opět na opačném principu k zálohování. Nejdříve je uživateli zobrazen dialog s přístupem do jeho osobní složky na úložišti. Po vybrání souboru je zahájeno zpracování souboru předaným identifikátorem. Obsah souboru `DriveFile` je načítán objektem `ContentsResult` metodou `openContents()`. Data jsou stejně jako v případě lokálního obnovení načítána objektem `BufferedReader`. Po dokončení načítání dat je do databáze uložena cesta i všechny její aktivity.

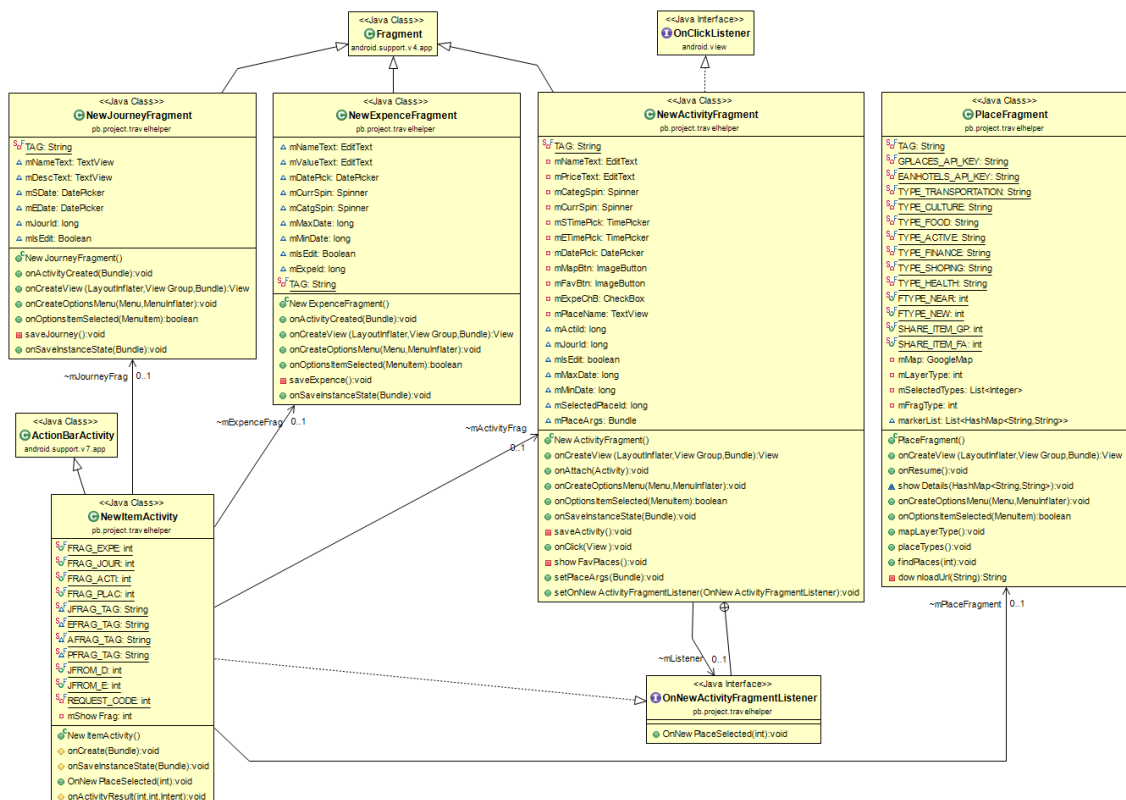
5.2 Nová položka

Vytváření nových dat, ať u se jedná o cesty, její aktivity nebo výdaje, je nedílnou součástí aplikace. Funkčně je tato vlastnost oddělena na vlastní obrazovku aplikace a je tedy popsána vlastní třídou `NewItemActivity` (Obrázek 24). Tato aktivita je při vytváření vyplněna jedním z fragmentů, které rozlišují typ přidávané položky a zajišťují potřebné metody pro obsluhu.

5.2.1 Cesta

Vytváření cesty zajišťuje třída `NewJourneyFragment`, jejímiž argumenty jsou prvky uživatelského rozhraní, přes které uživatel zadává údaje o vytvářené cestě. Těmi jsou název cesty, počáteční a koncové datum a volitelný popis.

Uložení je provedeno metodou `saveJourney()`, která ověří platnost zadaných dat a následně provede zápis do databáze. Zápis je prováděn přes objekt `ContentResolver` a jeho



Obrázek 24: Diagram tříd vytváření nových záznamů.

metody `insert()` nebo `update()`, pokud se jedná o vytváření nové respektive aktualizování stávající cesty.

5.2.2 Aktivita

Aktivitu vytváří třída `NewActivityFragment`. Stejně jako předchozí i tato třída definuje v atributech prvku uživatelského rozhraní pro zadání dat. Ukládání obsluhuje metoda `saveActivity()`, která ověřuje zadané údaje a následně je ukládá do databáze stejným způsobem jako předchozí třída.

S vytvářením nové aktivity souvisí vybrání místa, ve kterém aktivita probíhá. Místo je možné vybrat dvěma způsoby: z mapy nebo z oblíbených. Oblíbená místa jsou načítána z databáze. Jejich seznam je zobrazen uživateli v dialogu, kde může jedno místo zvolit. Zobrazení tohoto dialogu zprostředkovává metoda `showFavPlaces()`.

Pro výběr místa z mapy implementuje třída rozhraní, jehož metodu `onNewPlaceSelected()` musí implementovat obsluhující aktivita (`NewItemActivity`). Touto metodou je spuštěn fragment `PlaceFragment`, který umožňuje vybrat požadované místo. Třída tohoto fragmentu je využívána i pro zobrazení míst v okolí a bude popsána v kapitole 5.3. Výběr místa je umožněn pomocí dialogu s detailem místa a následně jsou metodou `setPlaceArgs()` nastaveny potřebné argumenty aktivity.

5.2.3 Výdaj

Správa výdajů, popsaná v kapitole 5.4, vyžaduje vytváření nových výdajů cestovatele. Vytváření výdajů zajišťuje třída `NewExpenseFragment`, fungující stejně jako předchozí třídy. Atributy třídy opět tvoří komponenty uživatelského rozhraní. Ukládané hodnoty jsou zadávány pomocí textových polí a rozbalovacích seznamů.

Ukládání výdaje je rozlišeno na nový nebo stávající a dle toho jsou ukládány do databáze. Tuto funkci zajišťuje metoda `saveExpense()`, která nejdříve zkontroluje formát zadaných dat a případně uživateli zobrazí varování o jejich nekorektnosti.

5.3 Místa v okolí

Funkci aplikace místa v okolí zajišťuje třída `NearPlacesActivity`, implementující třídu `ActionBarActivity` (Obrázek 25). Tato třída pouze zajišťuje zobrazení obrazovky aplikace a její vyplnění zajišťuje třída `PlaceFragment`, která dědí od třídy `SupportMapFragment`. Tento fragment tak zobrazuje mapu a umožňuje interakci s ní. To je umožněno přes objekt `GoogleMap` a jeho metody. Aplikace nastavuje parametry mapy, jako způsob ovládání a zobrazování aktivních prvků. Změnu typu mapy zajišťuje metoda `setMapType()` pomocí některé z předdefinovaných konstant.

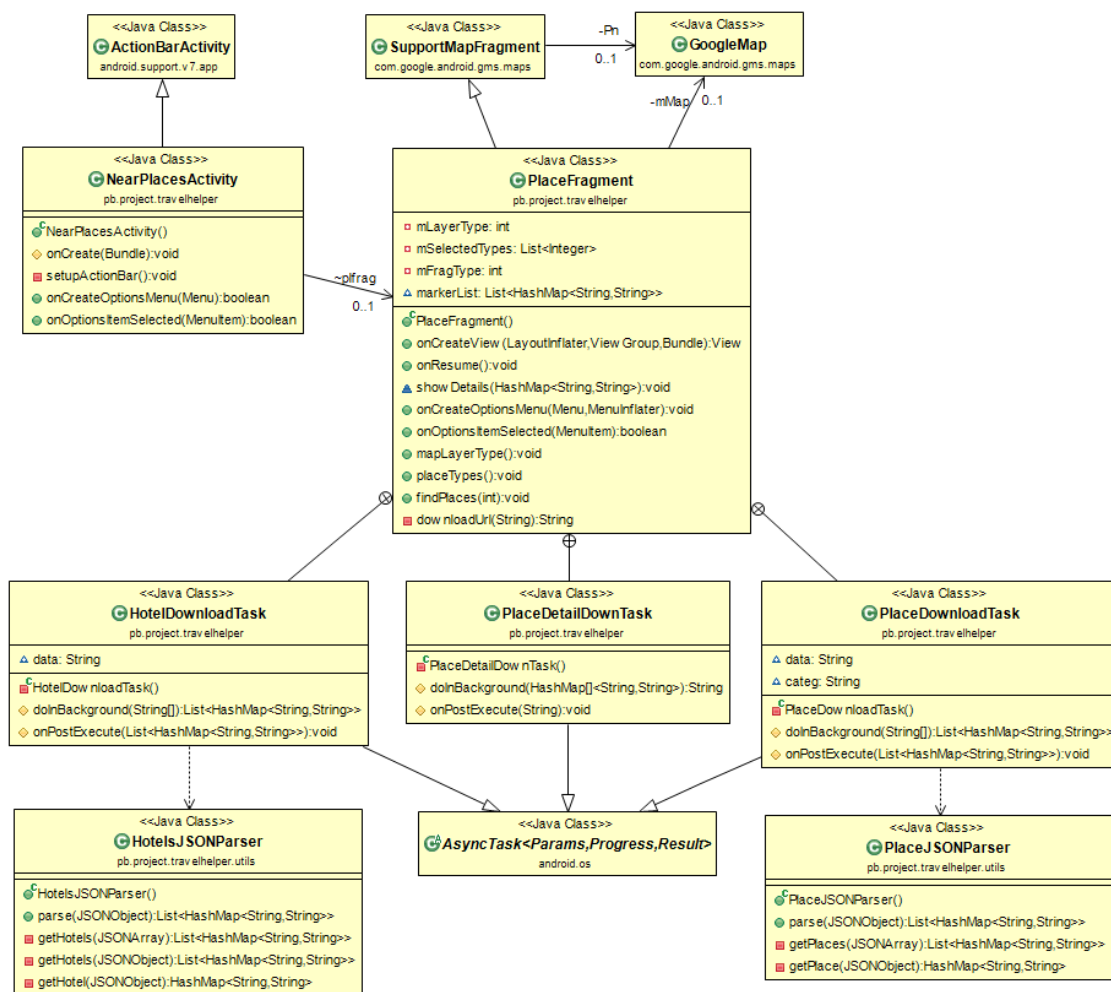
Fragment pro poskytnutí míst v okolí potřebuje získat pozici uživatele. Zjištění pozice umožňuje objekt `LocationManager`, který metodou `getBestProvider()` získá objekt poskytovatele pozice a následně metodou `getLastKnownPosition()` získá pozici uživatele. Pozici uživatele z objektu `Location` je následně třeba promítnout na mapu. Toto zajišťuje metoda `moveCamera()`.

Aplikace umožňuje zobrazit místa v okolí, které člení do kategorií: ubytování, doprava, kultura, jídlo a pití, aktivní, finance, nakupování a zdraví. Kategorii zobrazených míst umožňuje uživateli vybrat dialog, který je vytvářen metodou `placeTypes()`. Po dokončení výběru je metodou `findPlaces()` zaslán dotaz pro získání míst. Místa jsou získávána ze dvou různých zdrojů. Kategorii ubytování zajišťuje databáze hotelů *Expedia Affiliate Network*¹³, která pro jejich získávání poskytuje potřebné API. Ostatní kategorie míst jsou získávány ze služby *Google Places*¹⁴, která rovněž poskytuje API pro získání dat.

Stahování dat ze vzdálených databází služeb zajišťují třídy `HotelsDownloadTask` a `PlaceDownloadTask`. Obě třídy dědí od třídy `AsyncTask` a dědí její metodu

¹³ Databáze hotelů Expedia Affiliate Network <http://www.expediaaffiliate.com/index.php>

¹⁴ Google Places <https://developers.google.com/places/>



Obrázek 25: Diagram tříd implementujících zobrazení míst v okolí.

doInBackground(), pro vykonávání úlohy na pozadí. Třídy jako parametr přijímají předpřipravenou URL adresu, kterou je zasílán dotaz. Podoba adres pro získání dat o hotelech i místech je v následujícím kódu.

```

StringBuilder strb = new StringBuilder("http://dev.api.ean.com/ean-
    services/rs/hotel/v3/list?minorRev=[26]");
strb.append("&cid=55505");
strb.append("&apiKey="+EANHOTELS_API_KEY);
strb.append("&latitude="+lat);
strb.append("&longitude="+lon);
strb.append("&searchRadius=5");
strb.append("&searchRadiusUnit=KM");
strb.append("&locale="+Locale.getDefault().toString());
  
```

```

StringBuilder sb = new StringBuilder("https://maps.googleapis.com/
    maps/api/place/nearbysearch/json?");
sb.append("location="+lat+","+lon);
sb.append("&radius=5000");
sb.append("&sensor=true");
sb.append("&language="+Locale.getDefault().getDisplayLanguage());
sb.append("&key="+GPLACES_API_KEY);
sb.append("&types="+TYPE_CULTURE)
  
```

O stažení dat ze serverů u obou tříd zajišťuje metoda `downloadUrl()`, která zajišťuje HTTP spojení objektem `URLConnection`. Metodou `connect()` je otevřeno spojení a metodou `getInputStream()` jsou do objektu `StringBuffer` načtena data. Získaná data mají v obou případech formát JSON a jejich zpracování provádějí třídy `HotelJSONParser` a `PlacesJSONParser` prostřednictvím metody `parse()`. Postupně je procházena struktura dat a jsou získávána jen požadované záznamy. Výsledek je uložen do objektu `List`.

Protože základní data o místech z Google Places neobsahují všechny požadované informace, jsou pro každé získané místo třídou `PlaceDetailDownTask` tyto informace získány.

Po dokončení získávání dat jsou zobrazeny výsledky na mapě. Obě předchozí třídy provádějí přidávání značek do mapy prostřednictvím metody `onPostExecute()`. Značky na mapu se přidávají pomocí objektu `MarkerOptions`, jemuž je pomocí metod nastaven popis (`title()`), pozice (`position()`) a ikona dle kategorie (`icon()`). Samotné přidání značky je provedeno metodou `addMarker()` objektu `GoogleMap`.

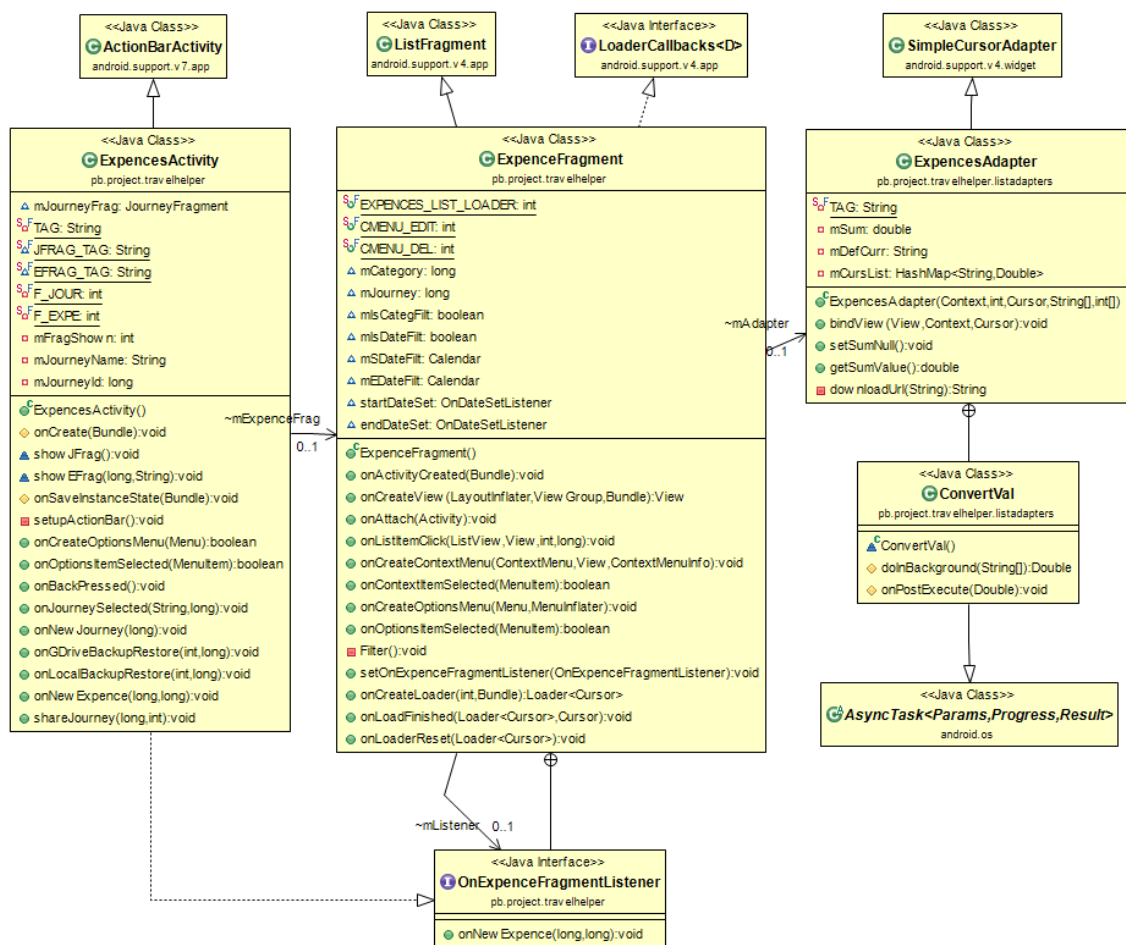
Aby mohl uživatel zjistit podrobnosti o zobrazených místech, je potřeba mu tyto informace nějakým způsobem zobrazit. K tomuto slouží komponenta *InfoWindow*, která pro každou značku volá metodu `showDetails()`. Tato metoda zobrazí dialog, který nese požadované informace. Těmi jsou název a adresa místa a uživatelské hodnocení místa. Dále se v dialogu nacházejí tři tlačítka. První pro zobrazení webové stránky místa (pokud je dostupná), druhé pro přidání místa do oblíbených a poslední pro sdílení. Sdílení právě navštíveného místa je možné, stejně jako v případě sdílení celé cesty, možné na sociální síť Google+ a Facebook.

5.4 Správa výdajů

Správa výdajů je obdobně jako cestovní deník rozdělena na dvě obrazovky. První z nich tvoří seznam cest, který je zastoupen třídou `JoureyFragment` (viz 5.1.1). Po výběru jedné z cest je zobrazen seznam jejích výdajů. Tento seznam implementuje třída `ExpenccFragment`, která dědí od třídy `ListFragment`. Dále třída implementuje rozhraní `LoaderCallbacks`, pomocí kterého jsou načítána data o výdajích z databáze. **Obrázek 26** zobrazuje diagram tříd.

Pro vyplnění jednotlivých řádků s výdaji slouží třída `ExpenccAdapter`, která je implementací třídy `SimpleCursorAdapter`. Pro každý výdaj je zobrazen jeho název, datum a především částka a měna. Výdaje jsou též rozlišeny dle piktogramu kategorie. Třída navíc počítá součet právě zobrazených výdajů. Jelikož mohou být výdaje v různých měnách, je třeba je při sčítání převést do jedné společné. Měna pro součet je vždy určena dle nastavení zařízení.

O převod částek v neodpovídající měně se stará podtřída `ConvertVal`, která je implementací `AsyncTask` pro vykonávání na pozadí. Třída využívá stejného systému pro



Obrázek 26: Diagram tříd správy výdajů.

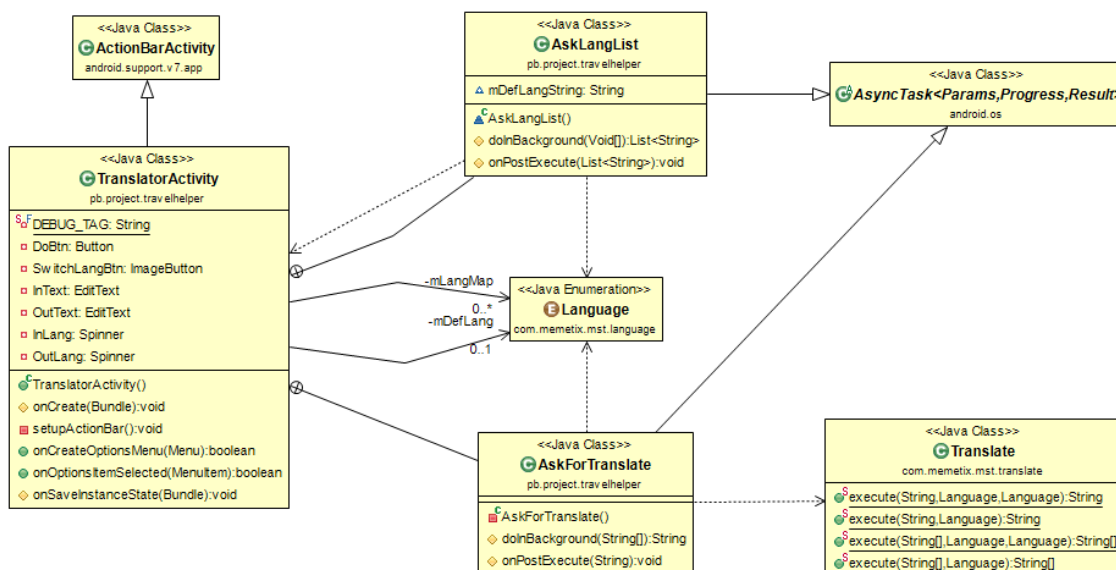
převod, jako převodník měn (viz 5.6). Po získání částky ve výchozí měně, je částka přičtena k součtu.

Výdaje je možné filtrovat pomocí dvou parametrů: kategorie a data. Pro výběr filtru je zobrazen dialog, ve kterém uživatel může nastavit parametry filtrování. Filtr dle kategorie je zajištěn rozbalovacím seznamem. Filtr dle data zajišťují dialogy pro výběr počátečního a koncového data. Oba filtry je možné kombinovat.

5.5 Překladač

Třída překladače `TranslatorActivity`, jejíž diagram zobrazuje Obrázek 27, implementuje třídu `ActionBarActivity` z balíku `Android.Support.v7.app`, který pro starší verze systému přidává podporu nových technologií. Atributy této třídy jsou jednotlivé komponenty uživatelského rozhraní. Pomocí těchto atributů a jejich odvozených metod je možné ovládat aplikaci. Nejvýznamnějším prvkem je tlačítko `DoBtn`, které spouští vykonání překladače.

Třída obsahuje dvě podtřídy implementující třídu `AsyncTask`, která umožňuje vykonávání kódu na pozadí. Třídy od ní musí dědit metodu `doInBackground()`, jejíž



Obrázek 27: Diagram třídy překladače.

implementace vykonává požadovaný kód. Mohou také dědit metodu `onPostExecute()`, která je automaticky spuštěna po dokončení činnosti na pozadí. Konkrétně se jedná o třídy `AskForTranslate` a `AskLangList`. První z nich zajišťuje překlad požadovaného textu. K překladu je využívána knihovna *Microsoft Translator Java API*¹⁵, obsahující dvě hlavní třídy `Translate` a `Language`. Překlad je uskutečněn voláním metody `execute(text, from, to)`, jejíž atributy mají očekávaný význam. Atribut `text` obsahuje překládaný text a atributy `from` a `to` značí jazyk z a do kterého je překlad prováděn.

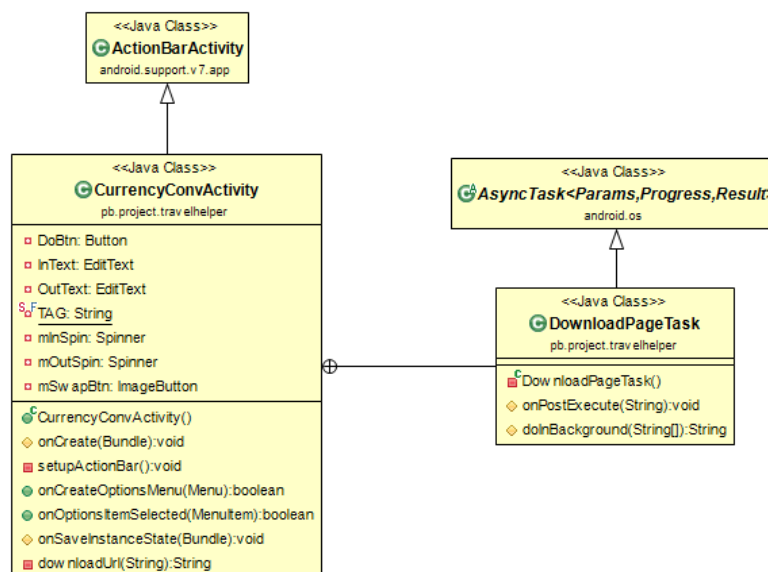
Druhá podtřída vykonává zjištění dostupných jazyků, které knihovna podporuje. Tyto jazyky třída lokalizuje do jazyka používaného v zařízení. Takto upravený seznam jazyků je využit k vyplnění nabídek výběru výchozího a cílového jazyka.

5.6 Převodník měn

Obrázek 28 zobrazuje diagram tříd převodníku měn, který obsahuje třídu `CurrencyConvActivity` sloužící pro převod měn. Třída dědí od `ActionBarActivity` je implementací aktivity systému Android. Třída v attributech definuje prvky uživatelského rozhraní. Například prvky typu `Spinner`, slouží jako rozbalující nabídka s dostupnými měnami pro převod. Atributem definujícím tlačítko `DoBtn`, je spouštěno vykonání převodu.

Třída obsahuje podtřidu `DownloadPageTask`, která implementuje třídu `AsyncTask`. V metodě `doInBackground()` je spuštěna metoda `downloadUrl()`, která zajišťuje zaslání HTTP dotazu a příjem dat. K tomuto je využíván objekt `HttpURLConnection`, jehož metodou

¹⁵ Knihovna dostupná z <https://code.google.com/p/microsoft-translator-java-api/>



Obrázek 28: Diagram tříd převodníku měn.

`connect()` je zaslán dotaz na nastavenou URL adresu. V metodě `onPostExecute()` je po přijetí dat zpracován a zobrazen výsledek dotazu.

Dotazy na převod částky jsou směřovány službě Yahoo Finance¹⁶, která poskytuje jednoduché API. To je přístupné pomocí URL dotazů, kdy parametry pro převod jsou definovány pomocí HTTP GET atributů. Těmito parametry jsou výchozí a cílová měna, částka pro převod a formát výstupních dat. Měny jsou zadávány pomocí mezinárodních kódů.

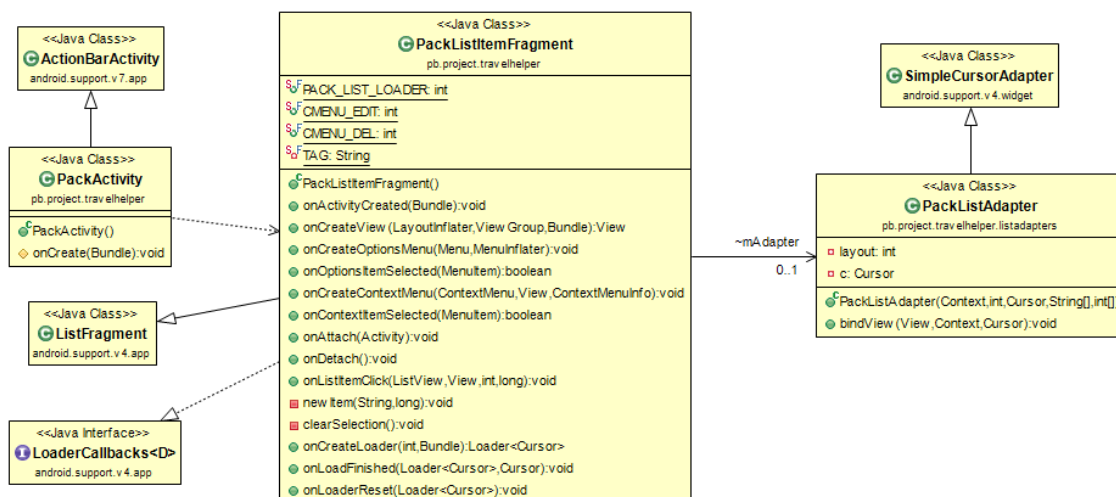
5.7 Balící seznam

Balící seznam je tvořen třemi třídami (viz Obrázek 29). První třída, specifikující Android aktivitu, je `PackActivity`. Tato třída je kompletně definována pomocí XML souboru uživatelského rozhraní. Hlavním funkčním prvkem této třídy je fragment specifikovaný vlastní třídou `PackListFragment`. Tato třída dědí metody z třídy `ListFragment` a implementuje rozhraní `LoaderCallback`.

Třída `ListFragment` zajišťuje obsluhu a zobrazení seznamu prvků. Ten je tvořen objektem `ListView`. Pro vyplnění seznamu daty je využíváno třídy `PackListAdapter`, která rozšiřuje základní systémovou třídu `SimpleCursorAdapter`. V této třídě je možné definovat formát a obsah dat seznamu. K tomuto účelu slouží metoda `bindView()`, která specifikuje obsah jednoho řádku seznamu. Zde konkrétně je vyplněn název položky a zatrhávací políčko.

Seznam prvků je vyplňován daty uloženými v databázi. Přístup k těmto datům je zajištěn pomocí metod rozhraní `LoaderCallback`, který je inicializován metodou `initLoader()` objektu `LoaderManager`. Metoda třídy balícího seznamu `onCreateLoader()` vytváří objekt

¹⁶ Yahoo Finance <https://finance.yahoo.com/currency-converter/>



Obrázek 29: Diagram tříd balíčku seznamu.

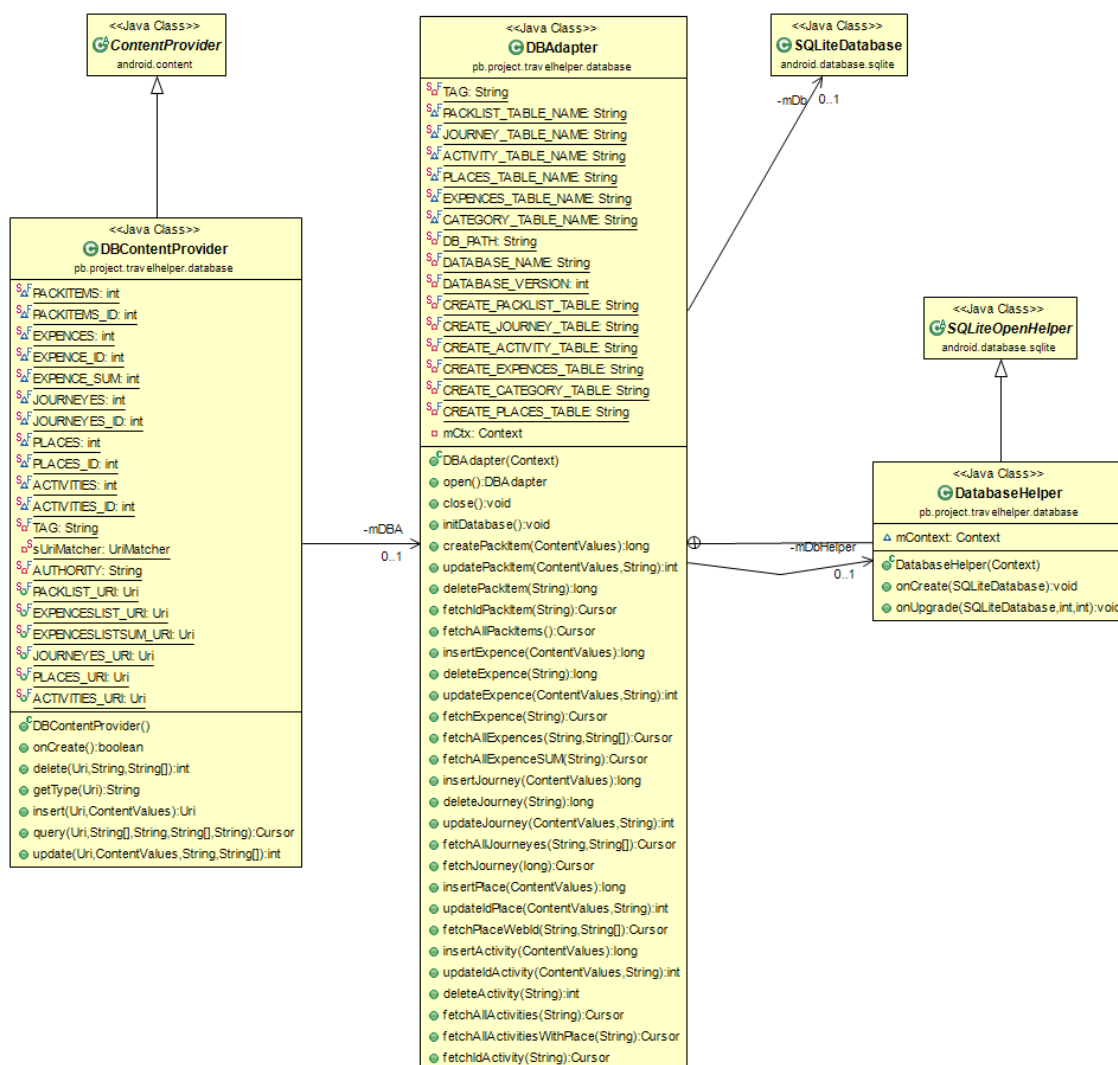
CursorLoader, který svým konstruktorem nastavuje potřebné parametry pro ContentProvider, jenž poskytuje data (viz kapitola 5.8). Data jsou přebírána z metodě onLoadFinished() v podobě objektu Cursor, který je předáván adaptéru.

Pro vytváření nové položky seznamu slouží metoda newItem(), jejímiž parametry jsou název položky name a identifikátor pl_id. Tyto parametry jsou nastavovány pouze v případě, kdy je metoda volána pro upravení stávající položky. Metoda vyvolá dialog pro zadání nového (nebo upravení stávajícího) jména položky. Po stisku potvrzovacího tlačítka jsou data uložena do databáze. K tomuto jsou využívány metody insert() a update() objektu ContentResolver. Pomocí parametrů metod dotazuje ContentResolver úpravu databáze voláním stejnojmenných metod třídy ContentProvider.

Metodou clearSelection() je zajištěno zrušení výběru prvků seznamu. Jedná se o úpravu databáze, proto je využíván objekt ContentResolver a metoda update. Jejímž parametrem je objekt ContentValues, který nastavuje hodnotu výběru na 0. Touto hodnotou jsou následně přepsána data ve všech řádcích databáze v příslušném sloupci.

5.8 Databáze

Aplikace pro uchování dat uživatele používá databázový systém. Na platformě Android je tímto systémem SQLite, jenž je přístupný přes třídu SQLiteOpenHelper. Implementaci této třídy v projektu obsahuje třída DBAdapter (Obrázek 30). Třída poskytuje metody pro otevření a zavření databáze. V obou případech je využit objekt podtřídy DatabaseHelper. V této třídě jsou definovány metody pro vytváření databáze (onCreate()) a její povýšení na novou verzi (onUpdate()). Obě metody mají jako parametr objekt třídy SQLiteDatabase, pomocí kterého metodou execSQL() a předem připravených konstant vytvářejí nebo povyšují databázi.



Obrázek 30: Diagram tříd obsluhy databáze.

Třída dále obsahuje metody obsluhy jednotlivých tabulek. Pro každou tabulku jsou tak definovány metody Insert, pro vytvoření nového záznamu, Delete a Update, pro smazání a aktualizaci záznamu. Další metody slouží pro získání záznamů z tabulek. Jedná se o metody `FetchId()`, pro získání jediného záznamu určeného jeho identifikátorem, a `FetchAll()`, pro získání všech záznamů z tabulky. Pro některé tabulky jsou implementovány další metody zpřístupňující speciální výběr dat potřebný pro vykonání některých tříd. Příkladem takové funkce je metoda `fetchActivitiesWithPlace()`, která k záznamům z tabulky Activities přidává i záznamy z tabulky Places. Těchto dat je využíváno při zobrazení bodů cesty na mapě a při sdílení a zálohování.

Databázi ostatním třídám aplikace zpřístupňuje třída `DBContentProvider`. Tato třída dědí od třídy `ContentProvider`, čímž získává metody `Insert()`, `Update()`, `Delete()` a `Query()`. Tyto metody zajišťují zpřístupnění dat voláním metod z třídy `DBAdapter` popsané výše. `ContentProvider` je v aplikaci využíván k rychlejšímu přístupu k databázi, jelikož pracuje na pozadí a stará se o veškerý přístup k datům.

5.8.1 Serverová databáze

Pro sdílení cest uživatele je využíván databázový systém *MySQL* umístěný na serverové straně. Struktura dat v tabulkách je shodná s daty v databázi aplikace. Webová část databáze však obsahuje pouze tabulky *Journeys* a *Activities*. Druhá ze jmenovaných kombinuje data z tabulek *Activities* a *Places* v aplikaci. Tyto tabulky postačují k určení a vizualizaci cesty na webové stránce.

Pro práci s webovou databází slouží dva skripty PHP umístěné na serverové straně. Skript *saveToDB.php* slouží pro uložení cesty a jejích dat do databáze. Skript využívá standardních funkcí jazyka PHP pro práci s *MySQL* databází. Pomocí metody `mysql_connect()` se připojí do zadané databáze. Data jsou skriptu předávána ve formě HTTP POST dotazu. Konkrétní podoba dat je stylizována do formátu JSON, která jsou dále metodou `json_decode()` zpracována do pole. Z tohoto pole jsou jednotlivá data pro cestu i její aktivity metodou `mysql_query()`, která jako parametr přijímá text dotazu, uložena do databáze. Skript vrátí jedinou hodnotu, kterou je identifikátor nově vytvořené cesty v databázi. Tento index je získán metodou `mysql_insert_id()`.

Druhým skriptem pro práci s databází je *loadFromDB.php*. Tento skript využívá výše popsaná webová služba. Skript se nejdříve stejně jako předchozí připojí k databázi. Dále však pracuje opačným způsobem než předchozí. Získává z databáze data pomocí indexu, získaného z HTTP parametru GET. Standardními SQL dotazy *Select* získá nejdříve data o cestě a následně i o všech jejích aktivitách. Tyto data ukládá do pole, ze kterého metodou `json_encode()` vytvoří data ve formátu JSON. Tato data jsou předávána webové aplikaci pro zpracování.

5.9 Uživatelské rozhraní

Uživatelské rozhraní bylo navrženo pro co nejsnadnější ovládání a přehlednost. Využívá standardních prvků třídy *View*, ze kterých vytváří jednotlivé obrazovky aplikace. Jak bylo popsáno v kapitole 2.1.2, uživatelské rozhraní je od aplikačního kódu odděleno do složky *layouts* ve zdrojích projektové složky aplikace. Zde se jednotlivé části rozhraní definují v samostatných XML souborech. Uživatelské rozhraní, tak jak bylo popsáno v kapitole 4.2, zachovává většinu struktury návrhu. Nejčastěji doplňuje další prvky spojené s dostupnými funkcemi.

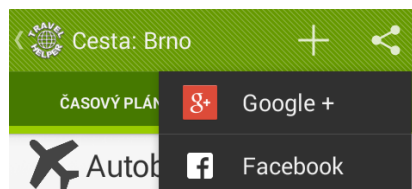
ActionBar

S ohledem na současný vývoj systému Android je napříč aplikací využíván prvek *ActionBar*. Tento prvek nahrazuje nabídku menu ze starších verzí systému. V současnosti plní *ActionBar* funkci hlavního ovládacího prvku aplikace, který je zobrazen jako lišta v záhlaví obrazovky.

Aktivity popisující obrazovky pak dědí od třídy `ActionBarActivity`, která tuto funkce zpřístupňuje i na starší verze systému Android.

Ovládací prvky lišty je možné definovat stále jako položky menu, pomocí XML souborů ve složce `menu`. Definují se pomocí elementu `Item`, kterému jsou nastaveny atributy definující identifikátor prvku, popisný text, ikonu a další. Na následujícím kódu je příklad definice akčních prvků lišty. V příkladu lze vidět definování akce pro přidávání položky cesty. Další prvek umožňuje sdílení cesty pomocí prvků v submenu. Takto definované menu je zobrazuje [Obrázek 31](#).

```
\<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_daypl_new"
        android:icon="@drawable/ic_action_new"
        android:title="@string/action_daypl_new"
        android:showAsAction="always"/>
    <item android:id="@+id/menu_daypl_share"
        android:icon="@drawable/ic_action_share"
        android:title="@string/action_daypl_share" >
        <menu>
            <item android:id="@+id/menu_daypl_gpshare"
                android:icon="@drawable/signin_btn_icon_dark"
                android:title="Google+"/>
            <item android:id="@+id/menu_daypl_fashare"
                android:icon="@drawable/com_facebook_inverse_icon"
                android:title="Facebook " >
        </menu>
    </item>
</menu>
```



Obrázek 31: Příklad lišty ActionBar se zobrazenou podnabídkou.

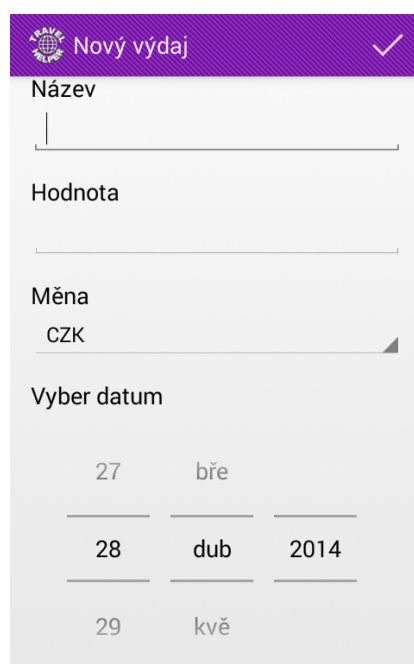
Obrazovky

Obrazovka může být tvořena vlastním uživatelským rozhraním nebo může být její obsah vyplněn fragmentem. V obou případech je uživatelské rozhraní popsáno XML souborem. Obvyklým kořenovým elementem je `LinearLayout`, ve kterém jsou jednotlivé elementy skládány postupně za sebe v horizontálním nebo vertikálním směru. Rozhraní však může být popsáno i relativním uspořádáním, kde elementem je v takovém případě `RelativeLayout`. Pokud je třeba uživatelské rozhraní protáhnout mimo velikost obrazovky je použit prvek `ScrollView`, který umožní posun obrazu i na nezobrazené prvky rozhraní.

V následujícím kódu je příklad definice obrazovky pro přidání nového výdaje, který popisuje soubor `fragment_new_expence.xml`. Zde je vidět definování posuvného elementu a

lineárního uspořádání. Dále jsou definovány prvky pro vyplnění údajů o výdaji (pro přehlednost zkráceno). Podobu této obrazovky znázorňuje [Obrázek 32](#).

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
        <TextView android:id="@+id/name_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/expe_new_name_label" />
        <EditText android:id="@+id/nexpe_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="text" />
        .
        .
        .
    </LinearLayout>
</ScrollView>
```



Obrázek 32: Obrazovka přidávání nového výdaje.

Položka seznamu

Položku seznamu nemusí tvořit pouze jediné textové pole, ale může být vyplněna libovolně strukturovaným rozráním. To je možné definovat stejně jako v případě obrazovky v XML souboru a může být tvořeno libovolnými prvky. Jako příklad z aplikace je možné uvést podobu

Obrázek 33: Příklad řádku seznamu výdajů.

řádku ze seznamu výdajů, kterou zobrazuje **Obrázek 33**. Řádek je tvořen obrázkem rozlišujícím kategorii výdaje, názvem, datem a konečně částkou a měnou výdaje. Toto uspořádání rozhraní je možné definovat následujícím kódem.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="100"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/expe_categ_image"
        android:layout_width="46dp"
        android:layout_height="46dp"
        android:layout_weight="15" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="45"
        android:gravity="center_vertical"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/expe_item_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <TextView
            android:id="@+id/expe_item_date"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <TextView
        android:id="@+id/expe_item_value"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="32"
        android:gravity="right|center_vertical"
        android:textAlignment="gravity"
        android:textSize="30sp" />
    <TextView
        android:id="@+id/expe_item_curr"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_marginLeft="5dp"
        android:layout_weight="8"
        android:gravity="bottom" />
</LinearLayout>
```

6 Testování

Testování je velmi důležitou součástí při vývoji jakékoliv aplikace. Jeho úkolem je odhalit chyby a ověřit správné chování aplikace. Kapitola popisuje zařízení, na kterých byla aplikace testována a dále se věnuje jednotlivým testům funkcí aplikace.

Testování probíhalo na několika reálných zařízeních a na emulátoru. Výhodou emulátoru je možnost nastavení libovolných parametrů emulovaného zařízení, jako například velikost a rozlišení obrazovky. Nevýhodou je však nedostupnost knihoven pro zobrazení map a nemožnost zaměření aktuální pozice. Z toho důvodu na emulátoru probíhají ve větší míře pouze testy vzhledu uživatelského rozhraní a testy na reálných datech převážně na fyzických zařízeních. Použité přístroje využívané při testování shrnuje následující tabulka.

Značka a název zařízení	Verze systému Android	Velikost displeje [“]	Rozlišení displeje
HTC Desire	2.3.7	3.7	480 x 800
HTC Desire	4.4.2	3.7	480 x 800
HTC Sensation XE	4.0.4	4.3	480 x 800
Huawei Y300	4.2.2	4	480 x 800
LG Optimus L9	4.0	4.7	540 x 960
Samsung Galaxy Note	4.3	10.1	2560 x 1600

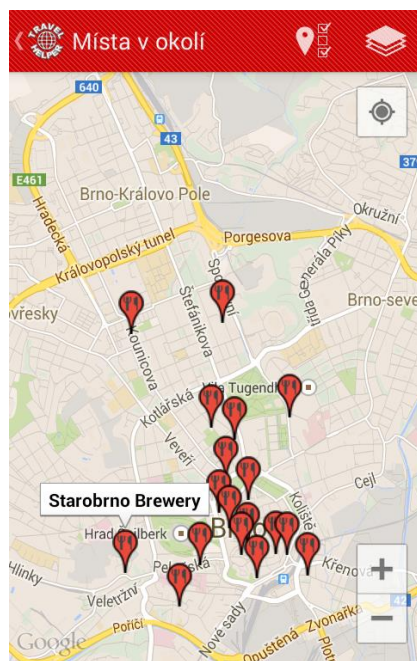
Pro účely testování poskytuje Android SDK nástroj Logcat, kterým je možné sledovat výstup systému a aplikací. Tento nástroj je velmi užitečný při ladění chyb programu, protože vypisuje chybová hlášení a příčinu vygenerování výjimky při ukončení aplikace

Testování vyhledávání míst

Pro vyhledání dostupných zájmových míst v okolí je třeba zaměřit pozici zařízení. Pozice je získána ve formě souřadnic: zeměpisná šířka a délka. Dále je pro dotaz nutné specifikovat požadovanou kategorii míst. Místa jsou vyhledávána v okruhu 5 km od zadaných souřadnic. Jako příklad je zobrazen dotaz na restaurace se středem v centru Brna. Takový dotaz má formu:

```
https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=49.19543125388614,16.607240103185177&radius=5000&sensor=true&language=čeština&key=API_KEY&types=bar|cafe|food|meal_takeway|restaurant
```

Data poskytnutá službou jsou zpracována a zobrazena na mapě v aplikaci. **Obrázek 34** vizualizuje takto nalezená data.



Obrázek 34: Zobrazení vyhledaných míst.

Testování sdílení cesty

Vytvořenou cestu je možné sdílet na sociální síti Google+ a Facebook. Data z databáze je nejdříve potřeba nahrát na webovou aplikaci, která je zpracuje a uloží k pozdějšímu zobrazení. Podobu odesílaných dat znázorňuje následující kód (zkráceno). Aplikace následně vygeneruje odkaz na sdílenou cestu. Tento odkaz je možné odeslat na jednu z vybraných sítí. Obrázek 1 **Obrázek 35** a **Obrázek 36** zobrazují takto sdílený odkaz. Podobu stránky, vizualizující sdílenou cestu zobrazuje **Obrázek 37**.

```
{
  "journey_id": "38",
  "name": "Brno ",
  "sdate": "1399327200857", "edate": "1399931999857",
  "activities": [
    {
      "act_id": "83",
      "name": "Hotel",
      "stime": "1399361438993", "etime": "1399368638993",
      "pl_name": "Hotel Slovan",
      "address": "Lidická 23",
      "lat": "49.2024", "lon": "16.6072",
      "journey_id": "38", "category": "0"
    },
    ...
    {
      "act id": "86",
      "name": "večeře ",
      "stime": "1399399250790", "etime": "1399406450790",
      "pl_name": "Starobrna Brewery",
      "address": "Hlinky 160/12, Brno, Česká republika",
      "lat": "49.1911", "lon": "16.5917",
      "journey id": "38", "category": "3"
    },
  ]
}
```




Obrázek 35: Sdílená cesta na síti Google+

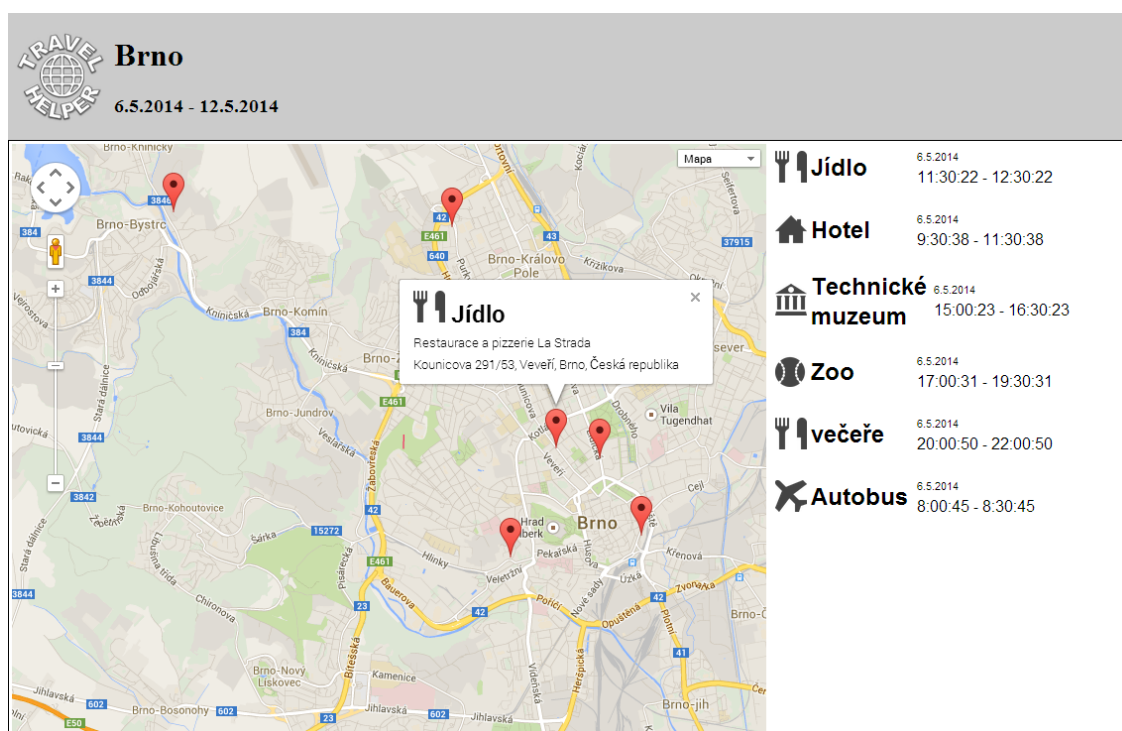


Obrázek 36: Sdílená cesta na síti Facebook.

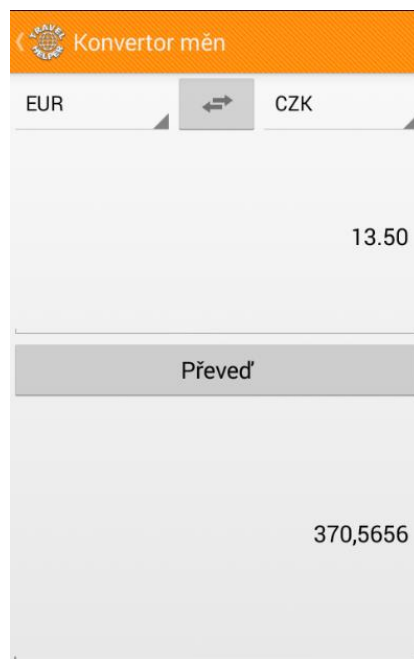
Testování konvertoru měn

Konvertor měn pro svoji práci využívá internetové připojení, jelikož pro získání výsledku využívá webovou službu. Uvažujme případ, kdy chce uživatel například převést částku 13,50 € na České koruny. Zadá požadovanou částku do vstupního pole a dále musí nastavit příslušné měny z rozbalovacích seznamů. Pro Euro vybere zkratku EUR a pro Českou korunu CZK, která je uživateli předvyplněna. Samotný převod je zahájen stiskem tlačítka pro převod. Je vytvořen dotaz, který se posílá internetové službě, jehož podoba je následující:

<http://quote.yahoo.com/d/quotes.csv?s=EURCZK=X&f=l1&e=.csv>



Obrázek 37: Podoba webové stránky vizualizující sdílenou cestu.



Obrázek 38: Test převodníku měn.

Výsledkem tohoto dotazu je aktuální kurz mezi zvolenými měnami, kterým je vynásobena požadovaná částka. Výsledná hodnota je zobrazena ve výstupním poli. Příslušné nastavení i výsledek zobrazuje [Obrázek 38](#).

7 Závěr

Cílem této práce bylo navrhnout a vytvořit aplikaci podporující činnosti cestovatelů. Aplikace byla vytvořena pro platformu Android, které se věnuje úvodní kapitola práce. Popsány byly nejdůležitější komponenty, které následně aplikace využívá. Jako prostředek pro stanovení funkcí aplikace sloužili popsané konkurenční aplikace. Vytvořený systém proto kombinuje některé jejich funkce a přidává některé vlastní.

Každá z funkcí výsledné aplikace má svůj přínos pro cestování. Cestovní deník umožňuje uživateli zaznamenávat aktivity a zážitky z cesty. V této formě je může dále prezentovat pomocí webové aplikace sdílené přes sociální sítě. Vyhledávání míst v okolí je pro cestovatele nepostradatelná funkce, převážně v neznámých místech. Umožňuje vyhledávat zajímavá místa a zobrazovat jejich detaily, jako adresa, uživatelské hodnocení nebo webová stránka. Překladač a konvertor měn jsou při cestách do zahraničí velmi využívané funkce. Ne každý ovládá světové jazyky a rozumí cizím řečem. Stejně tak určit správnou hodnotu zboží v cizí měně a porovnat ji tak se známou cenou může být výhodné. Podobnou funkci zastává i správa cestovních výdajů, pomocí které uživatel zaznamenává náklady na cestu. Stejně nepostradatelnou funkcí může pro někoho být seznam pro balení. Jeho položky a jejich indikace může usnadnit balení cestovních zavazadel a díky tomu na nic nezapomenout.

Při dalším vývoji projektu v budoucnosti je možné jej umístit na distribuční obchod Google Play. Zde by byla aplikace veřejně přístupná a mohla by díky tomu získávat ohlasy na vylepšení a doplnění dalších funkcí přímo od jejích uživatelů. Menším nedostatkem aplikace je lokalizace pouze do dvou jazyků (angličtina a čeština) a proto by bylo vhodné doplnit další světové jazyky. Stejnou pozornost při úpravách by bylo vhodné směřovat k uživatelskému rozhraní. Zde by byla vhodná optimalizace rozhraní pro zařízení s větší úhlopříčkou displeje (tablety). Jako příklad rozšíření je zde možnost doplnit funkci balícího seznamu o možnost definovat více těchto seznamů pro různé cestovní příležitosti.

Literatura

- [1] Allen, Grant. *Beginning Android 4*. New York: APress, 2012. ISBN 978-143-0239-857.
- [2] Steele, James a To, Nelson. *The Android developer's cookbook*. New York: Upper Saddle River, 2011. ISBN 978-0-321-74123-3.
- [3] Smith, Dave a Friesen, Jeff. *Android recepies: a problem-solution approach*. New York: Springer Science Business Media, 2001. ISBN 978-1-4302-3414-2.
- [4] GARGENTA, Marko. *Learning Android*. Sebastopol, Calif.: O'Reilly, c2011. ISBN 14-493-9050-1.
- [5] DARWIN, Ian F. *Android cookbook*. Beijing: O'Reilly, c2012. ISBN 14-493-8841-8.
- [6] Google Inc. *UI Overview*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <http://developer.android.com/guide/topics/ui/overview.html>.
- [7] Google Inc. *Services*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <http://developer.android.com/guide/components/services.html>.
- [8] Google Inc. *Application Fndamentals*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <http://developer.android.com/guide/components/fundamentals.html>.
- [9] Google Inc. *Dashboards*. Android Developers. [Online] 2014. [Citace: 20. 05. 2014]. <http://developer.android.com/about/dashboards/index.html>.
- [10] Google Inc. *Activities*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <http://developer.android.com/guide/components/activities.html>.
- [11] Google Inc. *Content providers*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <http://developer.android.com/guide/topics/providers/content-providers.html>.
- [12] Google Inc. *Intents and Intent filters*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <http://developer.android.com/guide/components/intents-filters.html>.
- [13] Google Inc. *Markers*. Google Developers - Google Maps Android API v2. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.google.com/maps/documentation/android/marker>.
- [14] Google Inc. *Map Objects*. Google Developers - Google Maps Android API v2. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.google.com/maps/documentation/android/map>.
- [15] Google Inc. *Sharing to Google+ from your Android app*. Google Developers - Google+ Platform for Android. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.google.com/+mobile/android/share/>.
- [16] Google Inc. *Recommend content with the +I button*. Google Developers - Google+ Platform for Android. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.google.com/+mobile/android/recommend>.
- [17] Google Inc. *Getting people and profile information*. Google Developers - Google+ Platform for Android. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.google.com/+mobile/android/people>.
- [18] Google Inc. *Google+ Sing-in for Android*. Google Developers - Google+ Platform for Android. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.google.com/+mobile/android/sign-in>.
- [19] Google Inc. *Google Services Overview*. Android Developers. [Online] 2013. [Citace: 20. 05. 2014]. <https://developer.android.com/google/index.html>.
- [20] Facebook. *Facebook SDK for Android*. Facebook Developers. [Online] 2013. [Citace: 20. 05. 2014]. <https://developers.facebook.com/docs/android/>.

Seznam příloh

Příloha A Struktura balíčků zdrojových kódů.

Příloha B Obsah CD/DVD.

Příloha A. Struktura balíčků zdrojových kódů

Aplikační projekt v prostředí Eclipse má tuto strukturu obsahující zdrojové kódy:

```
/TravelHelper
src/
    pb.project.travelhelper/
        CurrencyConvActivity.java
        DayPlanActivity.java
        DayPlanFragment.java
        DiaryActivity.java
        ExpenccFragment.java
        ExpencesActivity.java
        JourneyFragment.java
        MainActivity.java
        MapPlanFragment.java
        NearPlacesActivity.java
        NewActivityFragment.java
        NewExpenccFragment.java
        NewItemActivity.java
        NewJourneyFragment.java
        PackActivity.java
        PackListItemFragment.java
        PlaceFragment.java
        TranslatorActivity.java
    pb.project.travelhelper.database
        DBAdapter.java
        DBContentProvider.java
    pb.project.travelhelper.listadapters
        DayPlanAdapter.java
        ExpencesAdapter.java
        JourneyesAdapter.java
        PackListAdapter.java
        ShareSpinnerAdapter.java
    pb.project.travelhelper.utils
        GDriveBackUpTask.java
        GDriveRestoreTask.java
        HotelsJSONParser.java
        LocalBackUpAsyncTask.java
        LocalRestoreAsyncTask.java
        PlaceJSONParser.java
        ShareTask.java
res/
    layout/
        activities_item.xml
```

```

activity_currency_conv.xml
activity_day_plan.xml
activity_diary.xml
activity_expences.xml
activity_main.xml
activity_map.xml
activity_new_item.xml
activity_pack.xml
activity_translator.xml
dailog_expe_filter.xml
dialog_acti_place_detail.xml
dilog_place_detail.xml
divider.xml
expe_sum_item.xml
expece_item.xml
fragment_map_plan.xml
fragment_new_activity.xml
fragment_new_expece.xml
fragment_new_journey.xml
fragment_place.xml
journey_item.xml
mainfunc_item.xml
packlist_item.xml
spinner_share_item.xml
menu/
    day_plan.xml
    diary.xml
    expences.xml
    jour_ctx.xml
    jour.xml
    main.xml
    map.xml
    new_item.xml
    pack.xml
values/
    arrays.xml
    strings.xml
values-cs/
    strings.xml

```

Příloha B. Obsah CD/DVD

K práci přiložený disk obsahuje následující strukturu:

/

Text práce/

pdf

Tato práce ve formátu PDF

src/

Zdrojový text této práce ve formátu Microsoft Office docx

Aplikace/

apk

Instalační balíček aplikace

TravelHelper/

Zdrojové kódy aplikačního projektu prostředí Eclipse

Webová služba/

src/

Zdrojové soubory prostředí webové aplikace