**ZÁPADOČESKÁ UNIVERZITA V PLZNI**
FAKULTA EKONOMICKÁ

Bakalářská práce

# Vývoj aplikací pro mobilní telefony

# Applications development for mobile devices

Tomáš Fiala

Plzeň 2016

**Místo pro zadání**

**Čestné prohlášení**

Prohlašuji, že jsem bakalářskou práci na téma

*„Vývoj aplikací pro mobilní telefony“*

vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, dne ..... .....

podpis autora

**Poděkování**

Rád bych poděkoval vedoucímu práce doc. RNDr. Mikuláši GANGUROVI, Ph.D. za důvěru při vypracování práce a stránkám developers.android.com a stackoverflow.com bez kterých by tato práce nebyla možná dokončit.

# Obsah

# Úvod

S rostoucí popularitou mobilních zařízení roste i počet uživatelů, kteří používají telefon, jako jejich diář a kalendář. Současné aplikace nedokážou uživateli poskytnout informaci o správném načasování odchodu, aby dorazil na událost v čas, nebo nelze přesně nastavit způsob dopravy či upozornění.

Cílem této práce je návrh a vytvoření aplikace, která by uživateli umožnila správu událostí a zároveň ho v čas upozornila, že má vyrazit s ohledem na vybraný způsob dopravy a místo konání události.

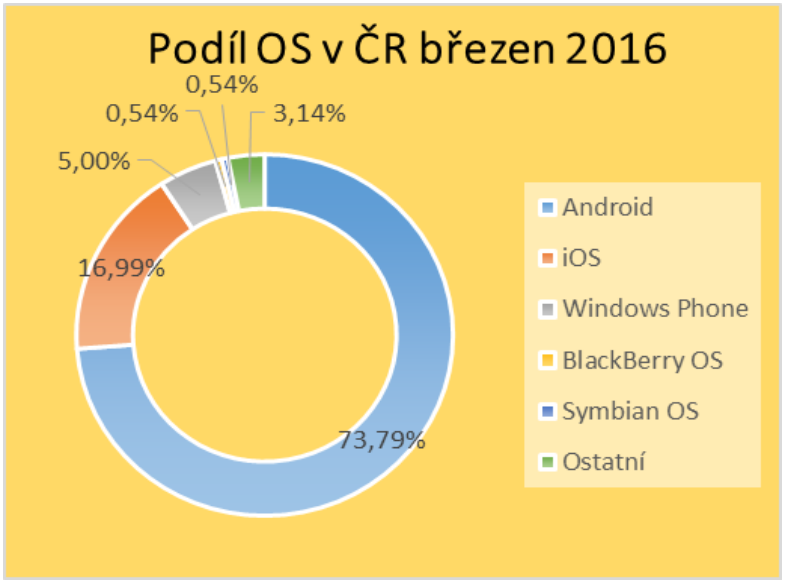
Tato práce se v teoretické části zaměřuje na výběr vhodné platformy a porovnání vývojových prostředí s ohledem na cílové uživatele a moderní trendy. Dále popisuje výběr vhodného nástroje pro vývoj a ladění, a možnosti použití služeb pro zjištění cesty na místo události. Nakonec jsou zde rozebrány použité komponenty.

Praktická část se zaobírá nastudováním vybrané platformy a vybráním vhodné metody pro zajištění požadované funkcionality aplikace. Zároveň bylo nutné seznámení se s uživatelským prostředím pro vývoj a samotným vytvořením funkční aplikace. Vytvořenou aplikaci bylo třeba podrobit uživatelskému testování toto testování vyhodnotit a zvážit implementaci získaných podnětů. Výstupem práce bude mobilní aplikace pro správu událostí.

# 1 Teoretická část

## 1.1 Porovnání platforem

Tato kapitola se zaměřuje na porovnání mobilních operačních systémů. Pro srovnání jsem vybral platformu Android a iOS jelikož jsou v ČR nejrozšířenější, viz obrázek 1. Cílem je nastínit stručnou historii, vývoj platforem a shrnout jejich výhody a nevýhody.




**Obrázek 1 - Podíl platforem v ČR [1]**

### 1.1.1 Android

Android je velice rychle se rozvíjející otevřený systém, který vyvíjí společnost Google Inc. Tento systém se používá jak v chytrých telefonech, tak v tabletech, chytrých hodinkách nebo noteboocích. V současné době se jedná o nejrozšířenější systém s největším podílem na trhu.[25]

Android vznikl v říjnu 2003, kdy byla založena společnost Android Inc. O 2 roky později byla odkoupena jako začínající a téměř neznámá firma její dceřinou společností Google Inc. A od té doby začal Google vyvíjet novou platformu s cílem vstoupení na trh mobilních telefonů. O další 3 roky později v roce 2008 byla představena první verze Android 1.0, zároveň bylo představeno open source SDK pro vývojáře, takže ho mohl každý využívat, což vedlo k velkému nárůstu počtu aplikací a oblíbenosti této platformy.[25]

Mezi výhody patří velice snadné nahrání aplikace na oficiální obchod. Samotný proces je velice jednoduchý a nepodléhá zdlouhavým schvalovacím procesem. Android je systém, který je navržený pro velké množství zařízení a nabízí velké množství nástrojů pro jeho přizpůsobení což je důsledek otevřenosti systému. Android je navíc skvěle provázaný s Google Službami jako Google Maps, Google Apps či sociální síť Google+.Nevýhodou je množství škodlivého softwaru, který může nakazit zařízení a způsobit veliké škody.

### 1.1.2 iOS

Operační systém iOS je systém vyvinutý firmou Apple. Inc. určený pouze pro telefony od téže značky. Tento operační systém lze nalézt v mimo telefonů navíc v tabletech iPad nebo chytrých hodinkých Apple Watch. Navzdory vysoké kvalitě produktů firmy Apple zaostává za svým největším konkurentem, systémem Android. [26]

Společnost Apple vznikla v roce 1976 ve městě Cupertin v Kalifornii. Tato oblast je dnes známá jako SilliconValley. V roce 2007 přichází s novinkou, chytrým mobilním telefonem s operačním systémem iOS. V roce 2008 vychází nový model a zároveň je uvolněno SDK. [26]

I když platforma od firmy Apple neokupuje první místo v rozšířenosti, stále má mnoho příznivců a mnoho odpůrců. Obecně jsou produkty od firmy Apple považovány za designovou záležitost. Systém iOS je uzavřený a odladěný na konkrétní zařízení. Nevýhodou je vyšší cena a možnost vývoje aplikací pouze na operačním systému do firmy Apple.

Vhodné zvolení platformy je stěžejní pro úspěšnost aplikace a záleží na více faktorech, které je potřeba zvážit. Nejprve je nutné určit, kdo jsou cíloví uživatelé a jaká zařízení využívají. Dále si je nutné uvědomit úskalí jednotlivých platforem. Apple může odmítnout vystavení aplikace v jeho obchodě a tím v podstatě znemožní jakoukoliv distribuci. Android na druhou stranu má nevýhodu v tom, že je nutné podporovat mnoho verzí operačního systému a typů zařízení. Další platformy jako Windows Phone, BlackBerry nebo Symbian mají malý podíl na trhu, viz obrázek 1.

## 1.2 Technologie pro vývoj

S rostoucí popularitou mobilních zařízení roste i počet frameworků pro vývoj. Framework slouží jako opora při vývoji aplikace, může obsahovat knihovny, podpůrné programy a doporučené postupy. Jeho cílem je usnadnit práci vývojářům tím, že řeší určité problémy za ně. Frameworky lze rozdělit do tří kategorií a to hybridní frameworky, cross-platform frameworky a nativní vývoj.

### 1.2.1 Hybridní frameworky

První kategorií jsou frameworky, které jsou založeny na webových technologiích jako HTML5, CSS a Javascriptu. Aplikace vyvinuté pomocí takového frameworku poté běží v takzvaném webkitu neboli renderovacím jádru prohlížeče a fungují v podstatě jako webová stránka a fungují na více platformách.

Příkladem takovýchto frameworků je například Ionic framework, který využívá knihovnu Angular.JS. Hlavní výhodou je, že je možné aplikaci vyvíjet pro více platforem zároveň. To přináší výhodu v rychlém vývoji, bohužel je těžké aplikaci dokonale odladit pro všechny zařízení z důvodu rozdílných prostředí, kde aplikace běží. Na podobné překážky narážejí weboví vývojáři, když musejí optimalizovat webovou stránku na více prohlížečů. Nativní funkce lze volat pomocí zásuvných modulů. Takové moduly se pak chovají jako nativní komponenty, ala mají i svá omezení a to, že nenabízejí výkon a funkce nativních API. Takovéto pluginy poskytuje například framework Cordova společnosti Apache. [2]

### 1.2.2 Cross-platform frameworky

Druhou kategorií jsou frameworky, které převádí kód z různých programovacích jazyků do nativního jazyka cílových zařízení. Aplikace je tedy napsána například v jazyce C# a následně převedena do nativní aplikace pro jednotlivé platformy.

Příkladem takovýchto frameworků je například Xamarin, který využívá jazyk C# a aplikace poté může být použita na systém Android, iOS, Windows Phone a BlackBerry zároveň. Výhodou je rychlejší vývoj nativně vypadajících aplikací. Takto vytvořené aplikace jsou robustní a mají nativní vzhled. Vývojáře však může odradit nutnost zakoupení licence pro komerční vývoj nebo řešení nativních funkcí pomocí pluginů. [23]

### 1.2.3 Nativní vývoj

Třetí kategorií je vývoj v nativním jazyce, což je pro Android jazyk Java a pro iOS jazyk Objective-C a Swift. Nativní aplikace se vyvíjejí jednotlivě pro každou platformu zvlášť pomocí vývojářských prostředí (tzv. IDE) a jazyka dané platformy. Vývojové prostředí nabízejí řadu užitečných nástrojů pro překlad zdrojového kódu, ladění aplikace, verzování a další. Nástroje jsou potřeba hlavně z důvodu složitosti nativního vývoje. Z uživatelského hlediska má nativní aplikace mnohem rychlejší odezvu a konzistentní vzhled oproti hybridním frameworkům. Využíváním nativních funkcí lze zajistit konzistentní vzhled a robustní architekturu. Nevýhodou je nutnost vytvářet aplikaci vícekrát, pokud chce vývojář poskytnout aplikaci na více platforem, což znamená vyšší náklady. [3]

### 1.2.4 Srovnání technologií

Mobilní vývoj je neustále se měnící oblast. Jelikož každý má jiné požadavky na technologii, kterou použije pro vývoj, nelze jednoznačně určit, která je lepší. Na základě předchozích kapitol je třeba vybírat dle určitých kritérií. V tabulce číslo 1 jsou srovnány nejdůležitější z nich. Zeleně jsou vyznačeny pozitivní vlastnosti a červeně negativní.[3], [4]

**Tabulka 1 - Srovnání technologií pro vývoj - zeleně pozitivní, červeně negativní**

| | **Nativní** | **Cross-platform** | **Hybridní** |
|---|---|---|---|
| Vykreslování | Nativní API | Nativní API | HTML,CSS, SVG |
| Výkonnost | Rychlé | Rychlé | Pomalé |
| Nativní vzhled | Nativní | Nativní | Simulované |
| Distribuce | Appstore | Appstore | Appstore |
| Kamera | Ano | Ano | Ano |
| Notifikace | Ano | Ano | Ano |
| Kontakty a kalendář | Ano | Ano | Ano |
| Geolokace | Ano | Ano | Ano |
| Nutnost připojení k internetu | Ne | Ne | Ne |
| Potřebné znalosti pro vývoj | Java, ObjectiveC | C#, Ruby | HTML,CSS, SVG |
| Rychlost vývoje | Pomalý | Rychlý | Rychlý |
| Podpora více platforem | Ne | Ano | Ano |
| Jiné požadavky | Nic | Licence | Nic |

*Zdroj: Vlastní zpracování podle [3], 2016*

## 1.3 Vývojářské nástroje

Pro nativní vývoj lze zvolit z řady vývojových prostředí. Pro vývoj nativních aplikací pro Android je nejrozšířenější oficiální prostředí Android Studio, které je zdarma a dostupné na všechny současně používané operační systémy jako je Windows, Linux a iOS. Pro vývoj aplikací pro zařízení společnosti Apple je oficiální nástroj aplikace xCode. Bez té není možné zkompilovat zdrojový kód a tudíž ani distribuovat aplikaci k uživateli.

### 1.3.1 Android Studio

Toto vývojové prostředí je k dispozici zdarma a instalace je velmi jednoduchá. Stačí stáhnout instalační balíček, který je dostupný na adrese http://developer.android.com/sdk/index.html. Součástí instalace je mimo samotného vývojového prostředí i emulátor pro Android, sada nástrojů Android SDK a kompilátor. SDK je balíček nástrojů, který vývojáři umožňuje využívat v aplikaci funkce dané platformy. Zároveň je pro instalaci vyžadována Java, jelikož je Android studio napsáno právě v Javě. Hlavní funkce, které usnadňují vývoj je nástroj pro přeložení aplikace Gradle, našeptávač funkcí a ve verzi 2+ je možné ladit aplikaci bez nutnosti opětovného překládání. [27]

### 1.3.2 xCode

Vývojové prostředí od společnosti Apple, které je dostupné zdarma, ale pouze na operační systém iOS na adrese https://developer.apple.com/xcode/download/ nebo přes Apple Appstore. Primárně slouží pro vývoj aplikací na operační systém iOS. Nabízí výborně optimalizovaný překladač a ladící nástroje pro rychlý vývoj.

### 1.3.3 Textové editory

Je možné vyvíjet aplikaci pomocí textového editoru jako Sublime Text nebo Atom. Nevýhodou je, že je nutné doinstalovat všechny potřebné komponenty manuálně a proto je vhodnější použít předpřipravené vývojové prostředí.

### 1.3.4 Zařízení pro testování

Pro testování aplikací je možné použít jak reálné zařízení, tak virtuální přístroj neboli emulátor.

Reálné zařízení je například mobilní telefon nebo tablet. Výhodou reálného zařízení při vývoji je, že nevyužívá systémové zdroje počítače, což může být výhodou u méně výkonných počítačů. Nevýhodou reálného zařízení je, že lze aplikaci testovat pouze na jedné verzi systému s konkrétními parametry. Pro testování aplikace na více zařízení je tedy nutné mít každý druh zařízení k dispozici.

Emulátor je program umožňující běh virtuálního zařízení na jiném. Konkrétně lze vytvořit virtuální mobilní zařízení na počítači, na kterém probíhá vývoj. Hlavní výhodou je, že je možné mít pro vývoj více druhů zařízení a lze tak snáze odladit aplikaci. Nevýhodou je, že emulátor využívá prostředky počítače a na méně výkonných počítačích je pomalý. Pro operační systém Android je k dispozici velké množství emulátorů od emulátoru vestavěném v prostředí Android Studio až po komerční emulátory jako Genymotion. [27]

#### 1.3.4.1 Android Studio Emulator

Tento emulátor, který je součástí Android SDK, podporuje telefony, tablety, hodinky a televize, které používají systém Android. Zároveň také umožňuje vytvořit zařízení dle konkrétních parametrů. Na emulátoru je tak možné spustit aplikaci, která je ve vývoji nebo testovat jak spolu spolupracuje více aplikací.

Mezi podporované funkce patří simulace polohy, připojení přes Wi-Fi a 3G, správa SD karty, webkamery nebo pořízení snímku obrazovky. Lze také upravit velikost obrazovky, přiblížit či oddálit obraz, přistupovat k SMS nebo telefonním hovorům. [5]

#### 1.3.4.2 Genymotion

Emulátor Genymotion, který je pro osobní použití zdarma, nabízí možnost nastavení polohy zařízení a mnoho verzí operačního systému Android od 2.3 až po 6.0. Emulátor Genymotion dokáže použít webkameru počítače jako kameru telefonu, je kompatibilní s nástrojem Android Studio. Zajímavou funkcí je možnost testování aplikace při slabé baterii. [31]

## 1.4 Třídy a služby

V této kapitole jsou popsány nejdůležitější komponenty používané při vývoji aplikací pro systém Android. Jelikož Android SDK nabízí veliké množství funkcí, je možné jeden problém řešit různými způsoby a většinou nelze jednoznačně říci, který je vhodnější.

### 1.4.1 Activity

Třída Activity se stará o vytvoření okna pro uživatele a je to důležitá část životního cyklu aplikace. Většina Activit je uživateli zobrazena jako full-screen okno (okno přes celou obrazovku). Jelikož jsou zařízení se systémem android většinou mobilní telefony, má systém poměrně omezené zdroje zejména paměť RAM. Z toho důvodu systém omezuje zdroje pro Activity na pozadí a zaměřuje se na Activity na popředí a to až do takové míry, že systém Activitu na pozadí úplně zastaví.

Každá Activita má předdefinované stavy a posloupnost těchto stavů se nazývá životní cyklus. Tyto stavy mají přesně dané pořadí a situaci, při které se spustí. Při změně stavu se volají předdefinované funkce. V tabulce 2 je uveden seznam metod, které jsou defaultně definovány ve třídě Activity. [7]

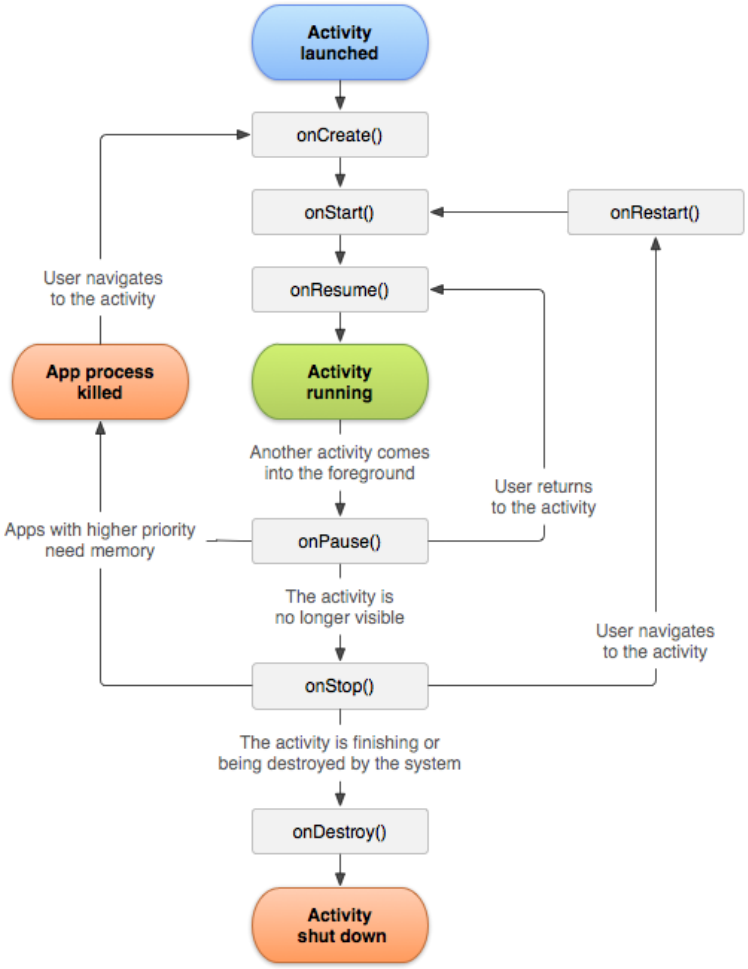
**Tabulka 2 - Funkce Activity**

| | |
|---|---|
| `onCreate(Bundle)` | Tato metoda je zavolána při vytvoření Activity. Zde se vytváří pohledy a načítá data. |
| `onStart()` | Voláno při zobrazení Activity uživateli. |
| `onRestart()` | Voláno poté co byla Activita zastavena a znovu nastartována. |
| `onResume()` | Voláno při interakci Activity s uživatelem. V této chvíli je Activita na vrcholu zásobníku. |
| `onPause()` | Voláno při pokusu systému o obnovení předchozí Activity. Zde je vhodné uložit neuložené data, zastavit animace a další funkce, které by mohli spotřebovávat zdroje zařízení. |
| `onStop()` | Voláno, pokud Activita již není pro uživatele viditelná z důvodu, že jiná Activita byla obnovena nebo byla zahájena Activita nová |
| `onDestroy()` | Poslední funkce volána před zničením Activity. Toto může nastat, pokud je Activita dokončena nebo systém zničí Activitu z důvodu nedostatku zdrojů. |

*Zdroj: Vlastní zpracování podle [7], 2016*

Všechny tyto metody lze přetížit pomocí anotace `@override` a vložit do nich vlastní funkcionalitu. Anotace `@override` slouží k přetížení rodičovských funkcí dané třídy.

Při inicializaci Activity je volána funkce `onCreate()`. V této funkci je nastaven zdroj layoutu pomocí funkce `setContentView(int)`. Lze zde také přiřadit data prvkům v layoutu, přiřadit funkcionalitu tlačítkům nebo nastavit obsah navigační lišty. Layout je rozložení prvků uživatelského prostředí ve formátu XML.




**Obrázek 2 - Životní cyklus Activity [7]**

Z diagramu na obrázku 2 je patrné, že se životní cyklus skládá ze tří stavů. Activita může mít stav Running (Běží), Paused (Pozastaveno) a Stopped (Zastaveno).

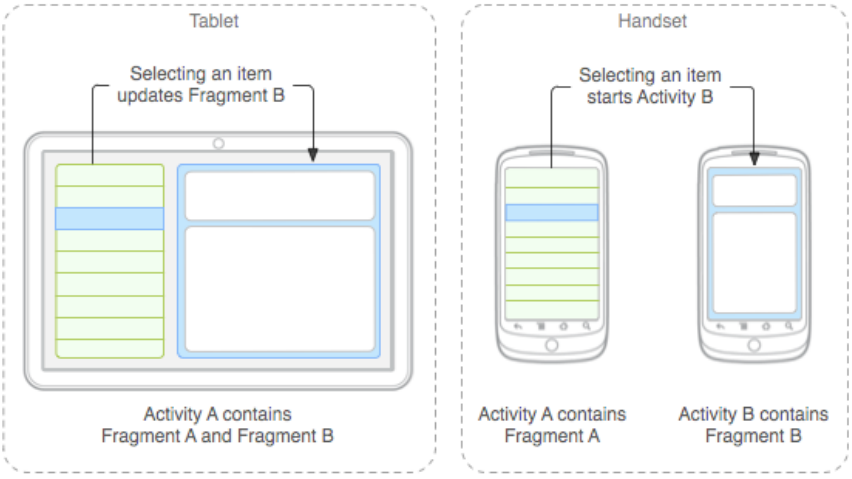
Stav Running nastává v případě, že je Activita v popředí a umožňuje interakci s uživatelem. Pokud je překryta jinou Activitou nebo je nad ní jiná průhledná Activita nachází se ve stavu Paused. To znamená, že Activita může být vidět, ale uživatel s ní nemůže pracovat, například při upozornění o nízkém stavu baterie nebo při vyzvání k přijmutí hovoru. Pokud Activita není viditelná, ale stále ji systém nezničil, pak se nachází ve stavu Stopped.

### 1.4.2 Intent

Pro přechod mezi jednotlivými Activitami se používá třída Intent. Tuto třídu si lze představit jako obálku, obsahuje třídu Activity, ze které je odeslána, třídu Activity, na kterou má přejít a může obsahovat dodatečné informace, které předá cílové Activitě. [7]

### 1.4.3 Fragment

Fragment je třída, která je implementována do Activity a slouží ke zlepšení uživatelského prostředí pro zařízení různé velikosti. Tato třída umožňuje, aby Activita na velkém displeji zobrazila 2 fragmenty vedle sebe, ale na malém pouze jeden a přecházela mezi nimi (viz obrázek 3). [20]




**Obrázek 3- Využití fragmentu [20]**

Fragment má životní cyklus velice podobný životnímu cyklu Activity a je přímo životním cyklem Activity ovlivněn. Fragment sjednocuje prvky layoutu jako tlačítka a zároveň kontejnery jako LinearLayout. [10]

Díky vzniku fragmentů v roce 2011 se pro každý kus uživatelského rozhraní, který může být použit vícekrát, začali vytvářet samostatné fragmenty. Dále nabízejí možnosti přidávání více fragmentů na obrazovku pomocí uživatelské interakce, animace, automatické zpětné tlačítko nebo přidání záložek do navigační lišty. [10]

Podobně jako Activita má i Fragment definované funkce pro řízení jeho životního cyklu. Při vytvoření se zavolá funkce `onCreateView()`, která vrátí objekt View, který reprezentuje daný fragment. Většinou fragment definuje XML soubor.

Pro přidání fragmentu do Activity je nutné vložit do XML souboru Activity fragment, kde atribut `android:name` určuje název třídy fragmentu. Při vytvoření Activity se díky tomu poté zavolá metoda `onCreateView()` dotyčného fragmentu. XML soubor Activity se dvěma fragmenty `ArticleListFragment` a `ArticleReaderFragment` pak vypadá následovně. [20]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

### 1.4.4 Databázový Helper

SQLiteDbHelper je třída, jejímž účelem je vytvoření a správa verzí databáze. Zde je definován SQL skript pro vytvoření a smazání databáze a jednotlivých tabulek. Je zde také vhodné vytvořit funkce pro vytvoření, upravení či smazání záznamů z tabulek. Hlavním důvodem je abstrahovat veškerý SQL kód z jiných částí aplikace.

Konstruktor této třídy vytvoří databázi a uloží její verzi. Při vytvoření databáze se zavolá funkce `onCreate(SQLiteDatabase)`, ve které se vykonají SQL skripty pro vytvoření tabulek. Pokud se zvýší verze databáze, zavolá se funkce

`onUpgrade(SQLiteDatabase,int,int)` a při snížení verze funkce `onDowngrade(SQLiteDatabase db, int, int)`, kde jsou parametry v pořadí objekt vytvořené databáze, číslo předchozí verze a číslo nové verze. [9]

### 1.4.5 Cursor

Tato třída poskytuje přístup k výsledkům při dotazu do databáze. Zavoláním metody `query(String)` je výsledkem právě třída Cursor. Pomocí cursoru lze získat informaci kolik výsledků pomocí metody `getCount()`, iterovat přes výsledné záznamy pomocí funkcí `moveToFirst()`, `moveToNext()` a `isAfterLast()`. Hlavní funkcionalitou je získání dat. Následující kód získá číslo sloupce pomocí funkce `getColumnIndexOrThrow(String)` a následně konkrétní hodnotu pomocí funkce `getString(int)`.

```
cursor.getString(cursor.getColumnIndexOrThrow("název_sloupce"))
```

Cursor také umožňuje uzavřít databázové spojení funkcí `close()` a znovu vykonat dotaz funkcí `requery()`.

Curzor lze obalit pomocí třídy SimpleCursorAdapter a poskytnout výsledný adaptér do ListView. Je důležité vědět, že při použití třídy CursorAdapter nebo jeho podtřídy musí výsledek dotazu obsahovat sloupec `_id` který je využit například při metodě `onListItemClick()`, čímž je určeno na kterou položku v seznamu uživatel klikl. [10]

### 1.4.6 Cursor Adapter

Adaptér je spojka mezi daty a zobrazením dat. Poskytuje přístup k jednotlivým položkám seznamu dat a stará se o zobrazení dat uživateli. Existují jak předdefinované adaptéry jako SimpleCursorAdapter nebo je možné vytvořit vlastní. Jednoduchý způsob jak zobrazit data při použití vlastního layoutu je rozšířit třídu CursorAdapter, jenž poskytuje funkci `bindView()`, která postupně přistupuje ke každému záznamu v cursoru. Nastavením hodnot jednotlivých polí v layoutu vznikne ListView. [10]

### 1.4.7 Service

Service, neboli služba je součást aplikace, která umožňuje běh dlouhých asynchronních operací na pozadí aplikace. Využívá se především pro úkoly jako načtení dat z webové služby, jelikož není závislá na Activitě a běží i po ukončení Activity. Stejně

jako Activita má služba svůj životní cyklus (tabulka 3), který zdědí od rodičovské třídy. Liší se tím, že nemá žádné uživatelské rozhraní. [10]

**Tabulka 3- Životní cyklus služby**

| | |
|---|---|
| `onCreate()` | Stejně jako u Activit je tato funkce volána při vytvoření služby |
| `onStartCommand()` | Voláno pokaždé, kdy je na službu zavolána funkce startService() |
| `onBind()` | Voláno pokaždé, kdy se klient spojí se službou pomocí funkce bindService() |
| `onDestroy()` | Voláno při ukončení služby |

*Zdroj: Vlastní zpracování podle [10][11], 2016*

Službu je nutné definovat v AndroidManifest.xml, což je soubor v domovském adresáři aplikace, kde jsou uvedeny základní informace o aplikaci. Mimo základních informací jsou zde také spravována práva a je zde uveden seznam služeb, receiverů, Activit a mnoho dalších.

Služby nejčastěji využívají Activity, i když to není pravidlem. Zavolání služby je možné dvěma způsoby, buďto odesláním požadavku zavoláním funkce `startService(Intent)`, čímž Activita ztratí se službou spojení nebo navázáním spojení pomocí funkce `bindService(ServiceConnction).`

Nejsnazší je zavolat službu pomocí funkce `startService(Intent)` podobně jako například `startActivity(Intent)` pro zahájení Activity. Intent zde má stejnou roly a to určit službu, se kterou chci komunikovat a odeslat parametry. Při zahájení služby systém udržuje službu spuštěnou na pozadí do té doby, než je ukončena například pomocí funkce `stopService(Intent)`. V tomto okamžiku se služba zastaví a následně je ukončena. Další možností je zavolání funkce `stopSelf(Intent)` v rámci samotné služby.

Druhou možností je zavolat službu pomocí funkce `bindService(Intent, ServiceConnection, int)`, která umožňuje zpřístupnění metod dané služby Activitám či jiným službám. Pro odpojení od služby je nutné zavolat `unbindService(Intent)` a poté již nelze bezpečně toto spojení použít. [10] [11]

### 1.4.8 Broadcast Receiver

Reciever je třída, díky které může aplikace přijmout informace od systému, aniž by byla samotná aplikace spuštěná. Pokud nastane událost, pro kterou byl receiver vytvořen, je systémem zavolána funkce `onReceive(Context, Intent)`. Poté co funkce skončí, systém může receiver použít znovu. Životní cyklus se tedy skládá pouze z této funkce. Pro ladění aplikace lze zaslat příkaz z příkazové řádky pomocí programu adb. Receiver je potřeba definovat v manifestu aplikace. [13]

### 1.4.9 Google Maps Web API

Služba od společnosti Google, díky které lze zjistit vzdálenost a trvání cesty mezi dvěma polohami. Tato služba funguje jako webové API a pro přístup je potřeba získat API klíč, který je dostupný zdarma.

Hlavní funkcí je tedy vyhledávání cesty pro různé druhy přepravy, jako je veřejná doprava, osobní automobil, chůze či jízdní kolo. Pro zjištění konkrétních informací je dále nutné specifikovat výchozí a cílovou polohu. Je možné také stanovit mezicíle, přes které má být cesta spočítána. Polohu lze definovat pomocí textového řetězce jako například "Plzeň" nebo souřadnicemi zeměpisné délky a šířky.

Základní požadavek má následující tvar:

```
https://maps.googleapis.com/maps/api/directions/<výstup>?<parametry>
```

Kde výstup může mít 2 hodnoty. První možnost je json, kdy bude výstup ve formátu JSON (JavaScript Object Notation) nebo xml což vrátí výstup ve formátu XML (Extensible Markup Language). K přístupu lze využít protokol HTTP nebo HTTPS pro zabezpečené připojení. Parametry zobrazené v tabulce 4 se dělí na povinné a nepovinné. [14]

**Tabulka 4 - Parametry Google Maps Directions API**

| **Povinné parametry** | |
|---|---|
| `origin` | adresa, souřadnice nebo ID místa, odkud se má spočítat cesta |
| `destination` | adresa, souřadnice nebo ID místa, na jaké se má spočítat cesta |
| `key` | přístupový API klíč aplikace, který identifikuje aplikaci |
| **Nepovinné parametry** | |
| `mode` | určuje způsob přepravy (driving, walking,bicycling, transit) |
| `alternatives` | pokud hodnota true, služba vrátí více alternativ pro cestu |
| `avoid` | určuje, čemu se má cesta vyhnout (tolls, highways,ferries,indoor) |
| `units` | určuje jednotky |
| `arrival_time` | čas doražení do cíle v milisekundách |
| `traffic_model` | určuje jak má služba předpovídat (best_quess, pessimistic,optimistic) |
| `transit_mode` | určuje preferovaný způsob veřejné dopravy |

*Zdroj: Vlastní zpracování podle [14], 2016*

### 1.4.10 Notifikace

Notifikace je zpráva, kterou lze zobrazit uživateli, i když právě nepoužívá danou aplikaci. Lze tak například zobrazit nový email. Navíc notifikace umožňuje přechod do specifické Activity aplikace a tak uživatel může snáze reagovat na událost, o které byl informován. Notifikace se zobrazuje v notifikační liště, kterou ovládá systém Android. Uživatel má možnost zobrazit detaily notifikace, odstranit ji nebo na ni kliknout.

Notifikace jsou důležitou součástí pro komunikaci s uživatelem a mají své vlastní pravidla, jak mají vypadat. Každá notifikace se skládá z ikony, titulku a obsahu notifikace. Dále je nutné definovat, kam se uživatel dostane po kliknutí na notifikaci.

Pro vytvoření notifikace slouží třída `NotificationManager`, která funguje jako systémová služba. Pro její použití je nutné získat objekt služby zavoláním funkce `getSystemService(NOTIFICATION_SERVICE)`. Tento získaný objekt

NotificationManager má k dispozici 3 metody `notify()`, která odešle notifikaci do notifikační lišty zařízení, `cancel()` a `cancelAll()` které naopak notifikace zruší.

Notifikace je možné sestavit pomocí třídy NotificationCompat.Builder. Ta přiřadí notifikaci ikonu, titulek a obsah notifikace. Dále přiřadí notifikaci Intent, který je zavolán při kliknutí na detail notifikace. Samotná notifikace může uživatele upozornit několika způsoby. Notifikace umí rozsvítit notifikační diodu, upozornit uživatele notifikačním tónem a použít vibrace. Je možné použít buďto jednotlivé upozornění nebo všechny najednou. [10],[19]

### 1.4.11 AndroidManifest

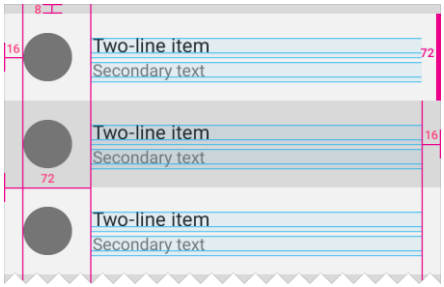
Každá aplikace musí mít soubor AndroidManifest.xml v domácím adresáři. Tento soubor obsahuje informace, bez kterých není možné aplikaci spustit. AndroidManifest obsahuje následující údaje:

- Obsahuje přístupová práva aplikace k systémovým funkcím
- Deklaruje práva, která musejí mít ostatní aplikace pro komunikaci s aplikací
- Účel manifestu je schraňovat informace o názvech balíků aplikace
- Obsahuje seznam všech tříd Activity, Service, BroadcastReceiver a ContentProvider [29]

### 1.4.12 Layout

Android má velice silné designové zásady neboli design guidelines. To znamená návod jak vytvářet aplikace tak, aby zapadli do vizuálního stylu platformy. Zajímavý je také přístup udělat aplikaci tak, aby se dala ovládat nosem. To znamená, aby ovládací prvky byli dostatečně veliké a měli kolem sebe dostatek prostoru.

Layout lze rozdělit na prvky jako tlačítka, popisky a uživatelské inputy a kontejnery jako LinearLayout. Android je velkým zastáncem Material designu, což je design inspirovaný vrstvami papíru a tuže. Má přesně stanovené jak mají prvky vypadat a jaké mají mít prvky mezi sebou mezery. Krásným příkladem je seznam, kde má každý prvek předem danou polohu. Na obrázku 4 je uveden jako příklad seznam se dvěma řádky a ikonou vlevo.




**Obrázek 4 - Seznam se dvouřádkovými položkami [30]**

List na obrázku 4 má definované následující hodnoty

- Font: RobotoRegular 16 sp
- Height: 48dp
- Icon padding, left: 16dp
- Top padding: 80dp

Takto jsou definované všechny komponenty materiálového designu, což má za následek konzistentní a profesionální vzhled aplikací. Jednotka sp (Scaleindependent pixel), která se přizpůsobuje velikosti fontu zařízení. Jednotka dp (Density-independent Pixel) je jednotka založena na hustotě pixelů. [30]

# 2 Praktická část

Tato část se zaměřuje na popis jednotlivých kroků vývoje aplikace pro správu událostí. Nejprve bylo nutné zvolit platformu a technologii pro vývoj.

## 2.1 Výběr platformy

Pro vývoj přichází v úvahu pouze vývoj pro systém Android nebo iOS, jelikož ostatní platformy jsou málo rozšířené a tím pádem ztrácí vývoj pro tyto platformy smysl.

Cíloví uživatelé mé aplikace jsou především studenti a aplikace má za cíl ulehčit uživatelům správu událostí a tím pádem jejich dochvilnost. Nejvíce uživatelů používá zařízení s operačním systémem Android [1].

Z těchto důvodů jsem se rozhodl vyvíjet aplikaci primárně pro systém Android, jelikož je základna uživatelů největší a neustále se rozšiřuje a zároveň je systém Android přístupnější pro vývoj. Pro vývoj aplikace pro iOS je nutné mít k dispozici počítač

s operačním systémem od firmy Apple. Oproti tomu aplikaci pro Android lze vyvíjet na jakémkoliv počítači s téměř každým operačním systémem. [22]

## 2.2 Výběr technologie pro aplikaci

Pro mou aplikaci jsem zvolil nativní vývoj v jazyce Java. Hlavním důvodem byla snadná integrace funkcí pro zjištění polohy a možnost zaměření ladění aplikace pouze pro jednu platformu. Přínosem také byla základní znalost jazyka Java, nemusel jsem se tak učit základy v tomto jazyce. Výhodou je také velice kvalitní a přehledná dokumentace v angličtině dostupná na adrese http://developer.android.com/, kde lze nalézt všechny informace.

Pro výběr jsem stanovil z mého pohledu nejdůležitější kritéria a následně jsem porovnal zvažované frameworky. Nejlépe vyšel nativní vývoj pro platformu Android. Výstupem srovnání je tabulka 5. Zeleně jsou vyznačeny splněná a červeně nesplněná. Pro srovnání jsem zvolil následující kritéria se stejnou vahou:

1. Podpora nativních funkcí
2. Možnost vývoje na operačním systému Linux
3. Kvalitní dokumentace
4. Nativní vzhled
5. Vývoj zdarma

**Tabulka 5 - Srovnání frameworků - zelená splněné kritérium, červeně nesplněné**

| Framework | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Ionic Framework | | | | | |
| Xamarin | | | | | |
| Native Android | | | | | |

*Zdroj: Vlastní zpracování, 2016*

## 2.3 Výběr podporovaných zařízení

Při zakládání projektu je nutné vybrat nejnižší podporovanou verzi SDK, neboli software development kit, což je sada nástrojů a funkcí, které má vývojář k dispozici. Zde je nutné se rozhodnout, zdali má aplikace za cíl pokrýt větší množství zařízení nebo zdali bude aplikace využívat funkce, které starší verze nepodporují. Android Studio usnadňuje výběr tím, že zobrazuje pokrytí zařízení při vybrání dané verze.

API level je celočíselná hodnota, která určuje, jakou verzi API nabízí daná platforma. Toto API vývojáři umožňuje komunikaci s interními funkcemi systému Android.

Nejvyšší pokrytí zařízení má Android 2.3 (API level 10) a to 100%. Android 4.0 má pouze o 2,7% nižší pokrytí a nabízí mnohem více funkcí pro vývojáře jako Calendar API nebo více funkcí uživatelského rozhraní aplikace. Díky těmto informacím jsem zvolil minimální verzi systému Android 4.0.

## 2.4 Organizace kódu

Ještě před samotným psaním kódu je vhodné si vytvořit přehlednou strukturu kódu. Celá aplikace se skládá z tříd, a zdrojů (resources).

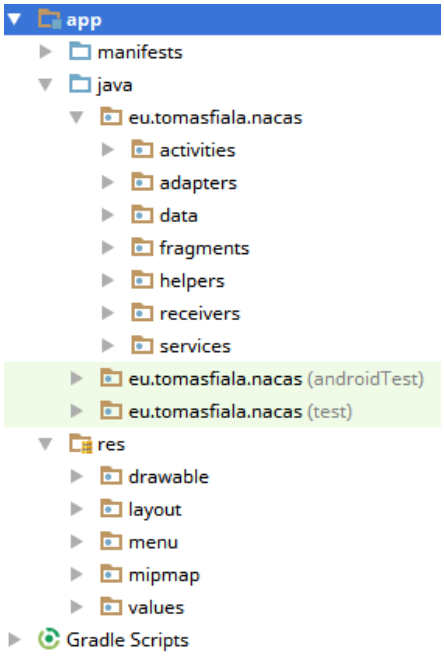
Třídy bylo nejvhodnější rozdělit do různých složek dle jejich typu. Tím vznikly následující složky.

- activities
  - obsahuje všechny Activity, čímž je snadné se orientovat v Activitách a není, je třeba dlouze hledat.
- adapters
  - obsahuje všechny adaptery
- data
  - obsahuje všechny třídy sloužící ke správě dat jako SQLiteHelper nebo Contract
- fragments
  - obsahuje všechny fragmenty
- helpers
  - obsahuje třídy, ve kterých je kód, který je využíván na více než jednom místě.
- receivers
  - obsahuje všechny receivery
- services
  - obsahuje všechny služby

Složka zdrojů `res` je již rozdělena na podsložky následovně.

- drawable
  - obsahuje ikony a obrázky
- layout
  - obsahuje layout Activit a fragmentů
- menu
  - obsahuje layout menu
- values
  - obsahuje hodnoty barev, styly a řetězce

Jelikož složka `layout` není nijak strukturovaná, je vhodné používat stejné předpony pro Activity, Fragmenty a Adaptery. Výsledná adresářová struktura aplikace při zobrazení pomocí nástroje Android Studio je zobrazena na obrázku 5.




**Obrázek 5 - Souborová struktura**

Poslední věcí je dodržování stylu psaní proměnných a identifikátorů. Pro identifikátory layoutu jsem zvolil takzvaný snake_case, kdy jsou slova spojena podtržítkem a pro proměnné v třídách camelCase, kdy následující slovo začíná velkým písmenem. [15]

## 2.5 Podobné aplikace

Nejzajímavější aplikací, která je v současné době k dispozici pro operační systém Android, je oficiální aplikace Google Now. Ostatní aplikace jsou k dispozici jen v angličtině a často nefungují správně.

Aplikace Google Now nabízí informace formou karet s informacemi. Informace se liší podle toho, co uživatel nastaví, například lze získávat informace o počasí, sportovních událostech nebo filmech v nejbližších kinech. Aplikace také předvídá Vaši cestu do práce a z práce. Lze přidat vlastní Připomenutí a aplikace uživatele upozorní, aby vyrazil na dané místo. [6]

## 2.6 Definování funkcí aplikace

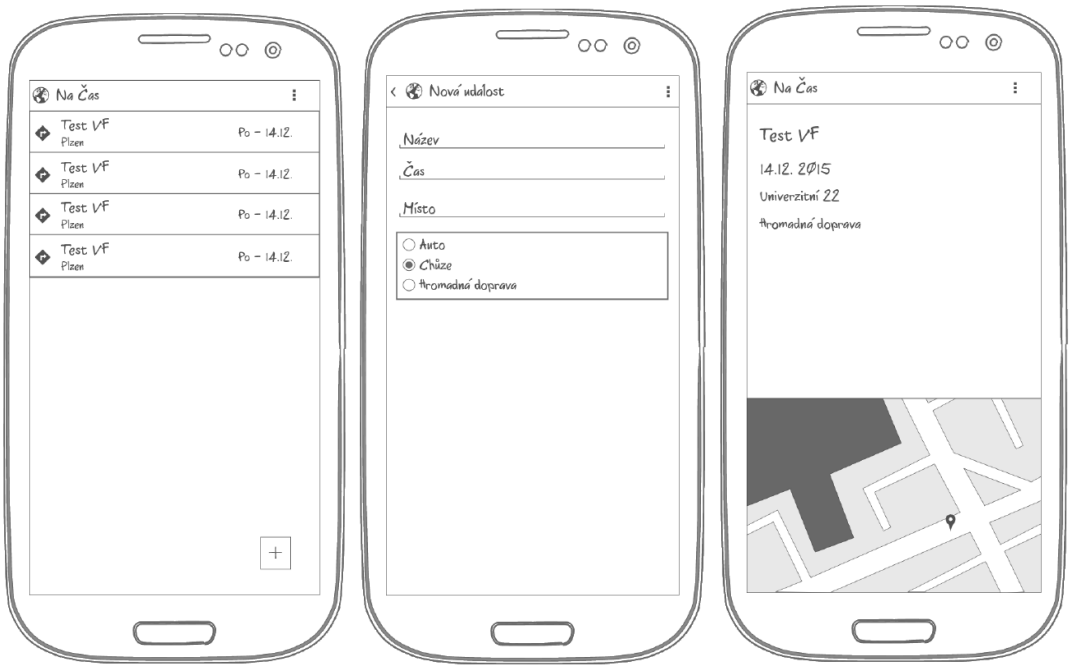
Tato kapitola popisuje funkce, které aplikace nabízí uživateli. Všechny tyto funkce jsou následně implementovány a spolu tvoří funkční aplikaci.

- Uživatel může vytvořit událost a u ní specifikovat kde a kdy se koná a preferovaný způsob dopravy. Tyto informace budou sloužit pro předpověď času, kdy má uživatel nejpozději vyrazit, aby na událost nedorazil pozdě.
- Aplikace upozorní uživatele, že je čas vyrazit.
- Uživatel může událost smazat.
- Aplikace může být použita jako zápisník, nevyžaduje vyplnění všech polí. Jediné povinné pole je poznámka.
- Na detailu události je možné zjistit bližší informace o cestě na událost. Zároveň je zobrazena mapa s vyznačeným místem, kde se událost koná.
- Pokud jsou nastaveny všechny potřebné atributy, aplikace uživatele upozorní, že má vyrazit na místo události.

## 2.7 Vytvoření layoutu

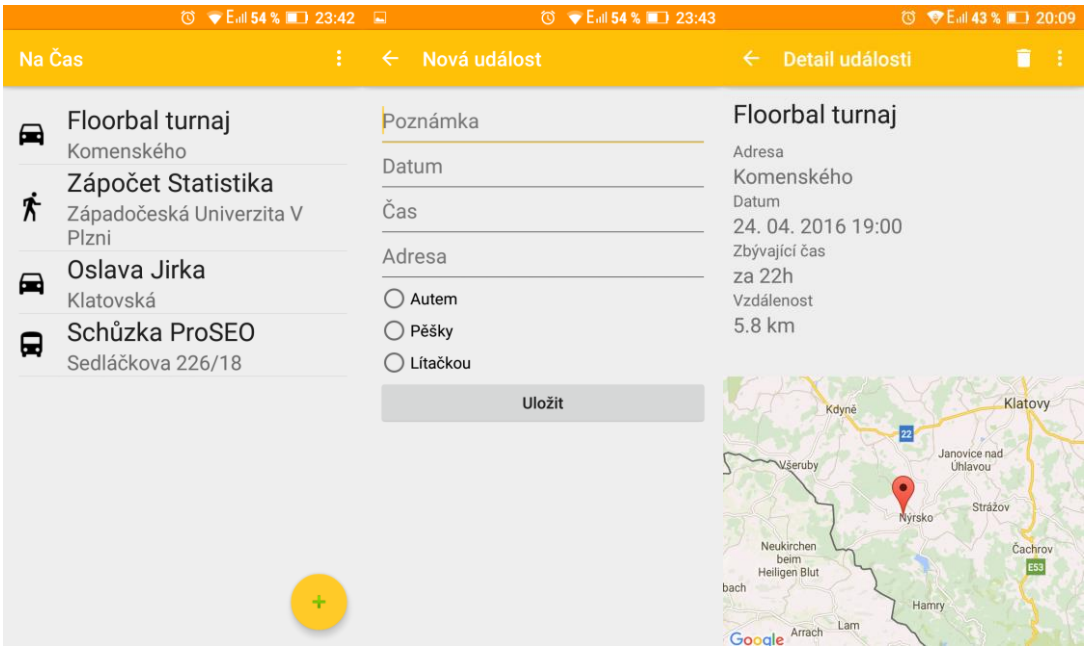
Layout určuje vzhled uživatelského rozhraní aplikace a je důležitý pro uživatelskou přívětivost aplikace, jelikož komplikované nebo nelogické uspořádání může uživatele odradit od používání aplikace.

Nejprve jsem vytvořil návrh aplikace pomocí nástroje NinjaMock dostupného zdarma pro nekomerční použití viz obrázek 6.




**Obrázek 6 - Návrh layoutu aplikace**

Aplikace se skládá z 3 Activit. Hlavní Activita slouží k zobrazení seznamu událostí, kliknutím na tlačítko „+“ uživatel přejde na Activitu, která slouží k vytvoření události a kliknutím na událost v seznamu uživatel přejde na detail události. Activita pro vytvoření nové události obsahuje pole pro určení místa a času události, napsání poznámky a vybrání způsobu přepravy. Activita detailu události zobrazuje detailní informace a mapu, kde se událost koná. Následně bylo potřeba vytvořit layout pomocí XML. Již vytvořený layout je zobrazen na obrázku 7.




**Obrázek 7 - Zobrazení layoutu aplikace (z leva seznam událostí, detail a formulář)**

## 2.8 Vytvoření Activit aplikace

V následující části je popsáno, jak byly vytvořeny jednotlivé Activity a jejich komponenty. Z důvodu velkého množství zdrojového kódu jsou zde vloženy pouze úryvky, které tvoří základní funkcionalitu.

### 2.8.1 Seznam událostí

Seznam událostí, který je na obrázku 7 první zleva, je hlavní Activita, která zobrazuje seznam zadaných událostí, seřazených od nejaktuálnější. Seřazení zajišťuje SQL dotaz s `ORDER BY` clausulí. Zároveň je zde informace o místě události a způsob dopravy na událost.

Kliknutím na tlačítko v pravém dolním rohu uživatel přejde na formulář pro vytvoření nové události. Ve třídě `MainActivity` je nastaven `setOnClickListener()`. Ten má za úkol odchytávat kliknutí na tlačítko `FloatingActionButton`. Po kliknutí na něj se zavolí metoda `onClick(Intent)` a zahájí se nová Activita. Ukázka přechodu na novou Activitu pomocí tlačítka:

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(MainActivity.this, NewEventActivity.class);
        startActivity(intent);
    }
});
```

Pro přechod na detail události je použit `setOnItemClickLisistener()` na jednotlivé položky `ListView`, kde je navíc předán parametr `ITEM_ID`, aby v Activitě `DetailActivity` bylo možné vyhledat konkrétní událost. Ukázka přechodu z hlavní Activity na detail při kliknutí na položku seznamu:

```
lvItems.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
        eventsCursorAdapter.getItem(position);
        Intent intent = new Intent(getActivity(), DetailActivity.class)
        .putExtra("ITEM_ID", (int)id );
        startActivity(intent);
    }
});
```

Kliknutím na událost v seznamu uživatel přejde na detail konkrétní události. Pokud je v seznamu více událostí, než se vejde na obrazovku, uživatel může scrollovat. To je dosaženo použitím `ListView`, které naplňuje `CursorAdapter`. Ten vkládá položky do seznamu, ale pouze tolik, kolik jich je zrovna viditelných a neustále ListView obnovuje. To znamená, že i při stovce položek seznamu je na obrazovce pouze 8 položek a `CursorAdapter` tedy vykresluje pouze 10 položek (o 2 navíc z důvodu načítání první skryté položky při posunutí). Jakmile uživatel použije scroll, položky se začnou překreslovat.

Layout je vytvořen celkově ze čtyř souborů a to `activity_main.xml`, `fragment_main.xml` a `list_item_events.xml` a `menu_main.xml`. V souboru `activity_main.xml` je definována horní navigační lišta a tlačítko pro přidání nové události. V souboru fragment_main.xml je definováno pouze ListView a v souboru list_item_events.xml je definován layout jedné položky seznamu. Nejprve je definován `LinearLayout` s horizontální orientací, což znamená, že v něm jednotlivé prvky leží vedle sebe horizontálně. Následně je definováno `ImageView` které slouží pro zobrazení

typu dopravy a poté další `LinearLayout`, tentokrát s vertikální orientací, což znamená, že budou vnitřní prvky řazeny pod sebou vertikálně. Celý soubor `list_item_events.xml` je zobrazen níže.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal" >
    <ImageView
        android:layout_height="32dp"
        android:layout_width="wrap_content"
        android:contentDescription="@string/transport_type"
        android:id="@+id/list_item_transport_type"
        android:src="@drawable/car"
        android:scaleType="centerCrop"
        android:gravity="center"
        android:paddingLeft="0dp"
        android:paddingRight="24dp"
        android:layout_gravity="center_vertical"
        />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/list_item_events_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge" />
        <TextView
            android:id="@+id/list_item_events_address"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium" />
    </LinearLayout>
</LinearLayout>
```

### 2.8.2 Detail události

Detail události (Obrázek 7 uprostřed) obsahuje detailní informace a mapu se zobrazeným místem události. Úryvek kódu níže popisuje, jak aktivita přistupuje k datům konkrétní události. Po obdržení Intent a získání primárního klíče události nalezneme pomocí třídy `EventsDbHelper` požadovanou událost a následně vypíšeme detaily do layoutu.

```
int eventId = intent.getIntExtra("ITEM_ID", 0);
EventsDbHelper dbHelper = new EventsDbHelper(getContext());
cursor = dbHelper.getEvent(eventId);
```

Navigační lišta obsahuje v pravém rohu tlačítko pro smazání události. Při jeho

stisknutí se zavolá metoda `deleteEvent(int)` pro smazání události a následně aplikace přejde na hlavní Activitu.

Mapa je zde řešena pomocí třídy `SupportMapFragment`, která je součástí Android SDK. Díky této třídě je možná načíst asynchronně mapu a vytvořit bod na mapě.

Jelikož je čas události a čas potřebný k cestě uložen v milisekundách, je třeba čas převést na formát pro uživatele čitelný čehož je docíleno funkcí `timeToHumanReadable(long)`, která převede počet milisekund do čitelného stringu a tak například místo `5000` získáme řetězec `za 5s`.

### 2.8.3 Formulář pro vytvoření nové události

Formulář, zobrazen na obrázku 7 vpravo, umožňuje přidat novou událost. Zároveň jsou zde využity fragmenty pro výběr data a času, které fungují jako dialogy. Po kliknutí na pole adresa se v samostatném okně zobrazí našeptávač adresy. Uživatel může vložit pouze samotnou poznámku, ostatní pole nejsou povinná.

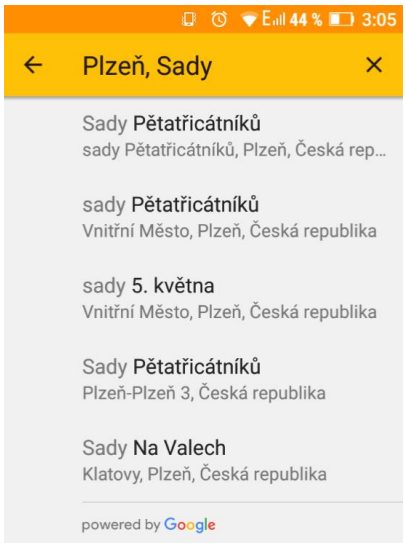
Našeptávač adresy je spuštěn při kliknutí do pole pro vyplnění adresy. Následně je zahájena Activita `PlaceAutocomplete`, která našeptává pomocí Google Places API místa dle uživatelského vstupu a obnovuje nabídku spolu s tím, co uživatel zadává.

Pro vytvoření našeptávače existuje více postupů. Je možné použít předpřipravený `PlaceAutocompleteFragment`, který není třeba nastavovat, stačí ho pouze přidat do XML souboru layoutu Activity a přidat listener pro navrácení hodnot. Druhá možnost je použít Intent a spustit novou Activitu. Tento postup umožňuje upravit vzled prvku, který bude Activitu spouštět, v mém případě textové pole adresa.

```
Intent intent = new
PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_FULLSCREEN)
      .build(getActivity());
startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);
```

Předchozí úryvek kódu vytvoří Intent pro přechod na našeptávač. Při vytvoření objektu Intent je možná nastavit mód zobrazení a to buď na `MODE_FULLSCREEN`, kdy je Activita zobrazena přes celou obrazovku nebo `MODE_OVERLAY`, kdy se zobrazí Activita formou poloprůhledného dialogu.

Používání našeptávače je zdarma, ale je podmíněno viditelností loga společnosti Google. [16] Aktivní našeptávač je zobrazen na obrázku 8.



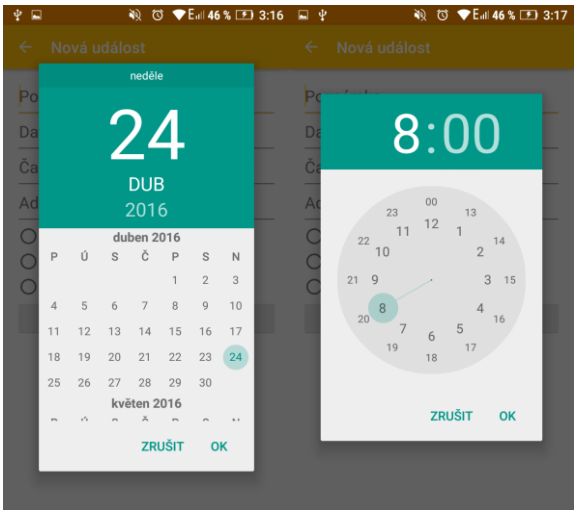

**Obrázek 8 - Našeptávač místa**

Po zadání místa se uživatel vrátí z Activity `PlaceAutocomplete` a v předchozí Activitě je zavolána funkce `onActivityResult(int,int,Intent)`, která obdrží zpět zadaný kód požadavku `PLACE_AUTOCOMPLETE_REQUEST_CODE` a podle něj rozhodne, jak má naložit s objektem `Intent`. Ten nyní obsahuje informace o místě, které uživatel zadal. Je tak možné zjistit souřadnice, krátké jméno místa, celou adresu a `place_id` což je unikátní identifikátor místa. [16]

Pro fungování našeptávače je potřeba připojení k internetu, s tím souvisí i nutnost přidání práv pro přístup aplikace k internetu. Pro povolení přístupu k internetu je nutné zažádat o práva pomocí user-permission tagu v AndroidManifest.xml následujícím způsobem:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Android poskytuje metodu pro zadání data a času ve formě dialogů. Dialog umožňuje výběr konkrétního data (dne, měsíce, roku) nebo času (hodina, minuta). Díky tomu je zajištěno, že uživatel zadá validní datum a čas. [17]




**Obrázek 9 - DatePicker a TimePicker**

Dialog pro zadání data je definován ve třídě `DatePickerFragment`, která rozšiřuje `DialogFragment` a implementuje `DatePickerDialog.OnDateSetListener`. Po nastavení data je zavolána funkce `onDateSet(DatePicker,int,int,int)`, kde jsou parametry zadaný rok, měsíc a den. Následně je možné tyto hodnoty uložit.

TimePicker je definován ve třídě `TimePickerFragment`, která rozšiřuje `DialogFragment` a implementuje `TimePickerDialog.OnTimeSetListener`. Po nastavení času je zavolána funkce `onTimeSet(TimePicker, int, int)` kde jsou parametry zadaná hodina a minuta. Příklad implementace TimePickeru: [17]

```
public class TimePickerFragment extends DialogFragment
        implements TimePickerDialog.OnTimeSetListener {

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Použití současného času jako vývhozí hodnoty
        final Calendar c = Calendar.getInstance();
        int hour = c.get(Calendar.HOUR_OF_DAY);
        int minute = c.get(Calendar.MINUTE);

        // Vytvoření nové instance dialogu pro TimePicker
        return new TimePickerDialog(getActivity(), this, hour, minute,
                DateFormat.is24HourFormat(getActivity()));
    }

    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        // Zpracování vstupu uživatele
    }
}
```

Po vrácení dané hodnoty je nutné datum formátovat pomocí `android.text.format.DateFormat`, aby byl zajištěn správný formát pro zařízení s jinými preferovanými formáty data. Zařízení v anglickém jazyce zobrazí datum ve formátu `"rok/měsíc/den"` a zařízení v českém jazyce `"den. měsíc. rok"`. Tyto hodnoty jsou poté v aplikaci zkompletovány a uloženy v milisekundách do databáze k příslušné události.

## 2.9 Funkce aplikace

V následující kapitole jsou popsány hlavní funkce aplikace a jejich implementace.

### 2.9.1 Vypočtení času upozornění

Pro zjištění času, kdy má uživatel vyrazit bylo potřeba vytvořit metodu, která zjistí čas potřebný na cestu. Z důvodu zajištění dokončení požadavku i při ukončení Activity bylo nutné vytvořit službu DirectionSerice, která pomocí webového API zjistí požadované informace. Společnost Google poskytuje veřejně dostupné webové API pro zjištění trasy, času a vzdálenosti mezi dvěma body. Jelikož uživatel zadává pouze cílové místo, je třeba zjistit, aktuální polohu uživatele pro výpočet. Pro zjištění polohy jsem vytvořil třídu LastLocationFinder, která zjistí polohu zařízení a vrátí objekt Location. Aktuální polohu lze zjistit pomocí třídy LocationManager za podmínky, že má aplikace povolení pro získání polohy. Tyto povolení se dělí na 2 druhy a to `ACCESS_COARSE_LOCATION`, která vrací přibližnou polohu v rámci například čtvrti a `ACCESS_FINE_LOCATION`, která vrací přesnou polohu s přesností na několik metrů. Pro povolení je nutné vložit do souboru `AndroidManifest.xml` následující kód:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Po zjištění aktuální polohy je možné zjistit požadované informace o trase. Odpověď serveru je ve formátu JSON a tak je nutné ji vhodně zpracovat a zjistit požadované informace jako čas potřebný na cestu a vzdálenost od cíle.

Upozornění uživatele je zajištěno formou notifikace, která se objeví v čase, kdy má uživatel vyrazit. Notifikaci zajišťuje třída AlarmReceiver, která rozšiřuje třídu BroadcastReceiver. Pro nastavení Intent pro AlarmReceiver byla vytvořena funkce `setAlarm(Long,int,int,String,String)`. Tato funkce nastaví notifikaci na zadaný čas s dalšími parametry. Nejprve je vytvořen Intent s parametry a poté nastavena notifikace.

```
public void setAlarm(Long alertTime, int eventId, String title, String address){
    Intent alertIntent = new Intent(this, AlarmReceiver.class)
            .putExtra("event_id", eventId)
            .putExtra("title", title)
            .putExtra("address", address);
    AlarmManager alarmManager = (AlarmManager)
            getSystemService(Context.ALARM_SERVICE);
    alarmManager.set(AlarmManager.RTC_WAKEUP, alertTime,
            PendingIntent.getBroadcast(this, eventId, alertIntent,
                    PendingIntent.FLAG_UPDATE_CURRENT));

}
```

Notifikace je nastavena pomocí tří parametrů a to typu notifikace, času notifikace a operace, která se má provést po ukončení notifikace. Typ notifikace je možné nastavit na 4 hodnoty popsané v tabulce 6.

**Tabulka 6 - Typy notifikací**

| | |
|---|---|
| ELAPSED_REALTIME | Notifikace neprobudí zařízení, zobrazí se, až v momentě kdy je zařízení aktivní. Notifikace se měří od spuštění zařízení |
| ELAPSED_REALTIME_WAKEUP | Probudí zařízení v čase notifikace, čas notifikace se měří od spuštění zařízení |
| RTC | Neprobudí zařízení, notifikace se měří v reálném čase |
| RTC_WAKEUP | Probudí zařízení v čase notifikace |

*Zdroj: Vlastní zpracování, 2016*

Pro notifikaci jsem zvolil typ `RTC_WAKEUP`, jelikož umožňuje notifikovat uživatele při uspaném telefonu a čas je zadán v reálném čase.

Samotné vytvoření notifikace provádí třída `AlarmReceiver`. Ve funkci `createNotification()` je získán objekt `mNotificationManager` pomocí funkce `getSystemService()` a na něj následně zavolána `funkce notify()`.

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        int eventId = intent.getIntExtra("event_id", 0);
        String title = intent.getStringExtra("title");
        String address = intent.getStringExtra("address");
        String timeToGo = context.getResources().getString(R.string.time_to_go);
        createNotification(context, title, address, timeToGo, eventId);
    }
    public void createNotification(Context context, String msg, String msgText,
String msgAlert , int id){
        PendingIntent notificationIntent = PendingIntent.getActivity(context, id,
                new Intent(context, MainActivity.class),0);
        NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(context)
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle(msg)
                .setTicker(msgAlert)
                .setContentText(msgText);

        mBuilder.setContentIntent(notificationIntent);
        mBuilder.setDefaults(NotificationCompat.DEFAULT_SOUND);
        mBuilder.setAutoCancel(true);
        NotificationManager mNotificationManager =
                (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
        mNotificationManager.notify(id, mBuilder.build());
    }
}
```

### 2.9.2 Ukládání dat

Pro ukládání dat jsem zvolil SQLite databázi. Tato databáze je velmi populární díky její nenáročnosti na paměť při poskytování uspokojivé rychlosti. Navíc je zabudována do systému Android a tak si ji může vytvořit každá aplikace. Pro vytvoření databáze je potřeba vytvořit podtřídu `SQLiteOpenHelper`, která vytvoří definované tabulky a stará se o verzi databáze. Já jsem tuto třídu nazval `EventsDbHelper`. Následně bylo nutné vytvořit SQL skripty pro vytvoření tabulek. Příklad může být vytvoření tabulky pro událost.

```
private static final String SQL_CREATE_EVENT =
    "CREATE TABLE " + EventEntry.TABLE_NAME + " (" +
        EventEntry._ID + " INTEGER PRIMARY KEY," +
        EventEntry.COLUMN_NAME_TITLE + TEXT_TYPE + NOT_NULL + COMMA_SEP +
        EventEntry.COLUMN_NAME_ORIGIN_ID + INTEGER_TYPE + COMMA_SEP +
        EventEntry.COLUMN_NAME_DESTINATION_ID + INTEGER_TYPE + COMMA_SEP +
        EventEntry.COLUMN_NAME_DATETIME + TEXT_TYPE + COMMA_SEP +
        EventEntry.COLUMN_NAME_DISTANCE + INTEGER_TYPE + COMMA_SEP +
        EventEntry.COLUMN_NAME_DURATION + INTEGER_TYPE + COMMA_SEP +
        EventEntry.COLUMN_NAME_TRANSPORT_TYPE + TEXT_TYPE +
        " )";
```

V kódu výše je způsob vytvoření SQL skriptu pro vytvoření tabulky. Třída `EventEntry` je umístěna ve třídě `EventsContract` a určuje, jak se dané sloupce v tabulce jmenují. Je také nutné ručně specifikovat verzi databáze a při změnění tabulek verzi inkrementovat. Při prvním vytvoření databáze je zavolána funkce `onCreate()`, která vykoná definované scripty `SQL_CREATE_EVENT` a `SQL_CREATE_LOCATION`.

```
public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_CREATE_EVENT);
    db.execSQL(SQL_CREATE_LOCATION);
}
```

Pro práci s daty jsem vytvořil funkce pro vložení/smazání/upravení/získání jedné nebo více událostí či poloh. Funkce pro získání dat vrací jako výsledek object Cursor.

## 2.10 Uživatelské testování aplikace

Uživatelské testování jsem prováděl převážně osobně a všímal jsem si, co uživatel dělá, nad čím přemýšlí a s čím si případně neví rady. Většina uživatelů nejdříve nevěděla, k čemu aplikace přesně slouží. Po té co jsem je s aplikací nechal chvíly pracovat, pochopili, že je možné přidat událost a na detailu události je možné získat více informací o cestě. Z tohoto testování bylo možné mnohem lépe vyvodit, jak uživatelé prochází aplikací, ale nebylo možné pozorovat dlouhodobé užívání aplikace. Z tohoto důvodu jsem vytvořil dotazník, který uživatelé následně vyplnili.

Z uživatelského testování vyplynulo, že uživatelé mají zájem o větší množství funkcí. Mezi návrhy bylo například zobrazení trasy v detailu události, možnost upravení události či zadání cílové polohy dle aktuální pozice zařízení.

Z výsledků dotazníků vyplynulo, že je aplikace jednoduchá na pochopení a dělá to, co uživatelé očekávají. Nejzáporněji dopadla otázka, zdali si uživatel myslí, že má aplikace dostatečně výrazná tlačítka.

## 2.11 Uživatelský manuál

Zde je popsán manuál pro používání aplikace NaČas, která má za cíl zlepšit dochvilnost uživatelů pomocí upozornění na událost v čase, kdy má nejpozději vyrazit. Pro instalaci je nutné mít telefon s operačním systémem Android verze 4.0 a vyšší.

## 2.12 Instalace

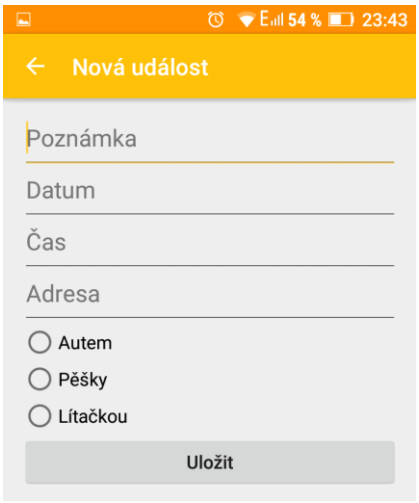
Aplikaci je možné nainstalovat pomocí instalačního balíčku pro Android. Je nutné povolit instalaci aplikací třetích stran. Po stažení balíčku lze následně aplikaci nainstalovat na zařízení.

Po prvním spuštění se Vám zobrazí prázdný seznam událostí. Událost můžete přidat pomocí tlačítka v pravém dolním rohu.

## 2.13 Přidání události

Při příchodu na Activitu pro novou událost je aplikace požádá o GPS pro zjištění aktuální polohy. Pokud GPS nepovolíte, aplikace nezjistí polohu a není tím pádem možné zjistit informace o cestě a nastavit upozornění. To však nemá vliv na uložení události. Při pozdějším povolením GPS aplikace chybějící informace zjistí zpětně.

V této Activitě máte možnost zadat poznámku k události, vybrat datum, čas a pomocí našeptávače konkrétní místo. Pro použití našeptávače je nutné připojení k internetu. Nakonec lze vybrat, jaký typ dopravy preferujete. Kliknutím na tlačítko Uložit se událost uloží a aplikace se vrátí na seznam událostí.

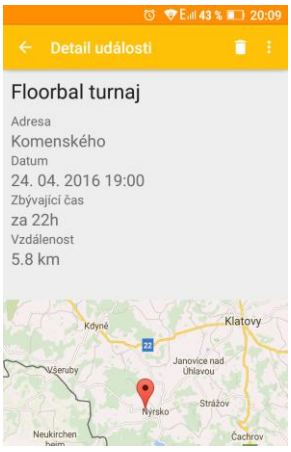



**Obrázek 10 - Activita pro vytvoření události**

Pokud se rozmyslíte a rozhodnete se událost neuložit, stačí kliknout na tlačítko zpět v levém horním rohu nebo ekvivalentní na hardwarové tlačítko.

## 2.14 Detail události

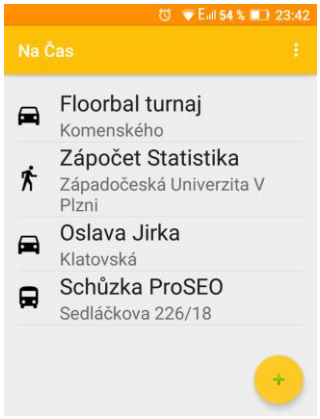
Pro zobrazení detailu události stačí kliknout na událost v seznamu událostí. Pokud byla vyplněna všechna pole a bylo dostupné připojení k internetu a GPS zobrazí se všechny dostupné informace o události. V navigační liště je tlačítko pro smazání události a ve spodní části mapa, kde je vyznačeno místo kde se událost koná. Je zde také zobrazeno, za jak dlouho máte vyrazit, abyste nepřišli pozdě. Aplikace zobrazí notifikaci, i když zařízení právě nepoužíváte.




**Obrázek 11 - Detail události**

## 2.15 Seznam událostí

V seznamu událostí můžete vidět všechny Vaše uložené události s informací, kde se událost koná, poznámkou a typem zvolené přepravy. Novou událost můžete přidat tlačítkem v pravém dolním rohu.




**Obrázek 12 - Seznam událostí**

# Závěr

Během zpracování bakalářské práce jsem se seznámil s možnostmi vývoje mobilních aplikací a pro realizaci mobilní aplikace jsem vybral nativní vývoj pro platformu Android. Díky získaným znalostem o vývoji nativních aplikací jsem byl schopen navrhnout a vyvinout funkční aplikaci pro správu událostí. Během vývoje jsem se musel učit nové postupy a získávat informace z internetových zdrojů a naučit se pracovat s nástroji jako Android Studio, které mi pomohlo s laděním aplikace.

Navrženou aplikaci jsem průběžně testoval a konzultoval s potencionálními uživateli a následně provedl pilotní průzkum. Navrhované změny jsem zaznamenal a implementoval, pokud jsem uvážil jejich opodstatněnost.

# Zkratky

**IDE -** vývojové prostředí neboli *Integrated Development Environment*

**API** – *Application Interface* je seznam metod, které umožňují komunikaci s jinou aplikací či službou a naopak

**XML** – strukturovaný datový formát *Extensible Markup Language*

**JSON** – strukturovaný datový formát *JavaScript Object Notation*

**SQL** – dotazovací jazyk používaný pro práci s daty v databázi

**CSS** – kaskádové styly používané pro grafickou úpravu webových stránek

**HTTP** – *Hypertext Transfer Protokol* umožňuje komunikaci pomocí webových služeb

**HTTPS** – zabezpečená verze HTTP

**SDK** – *Software development kit* je sada vývojových nástrojů

**RAM** – *Random Acess memory* je paměť určená pro ukládání mezipaměti

**GPS** – systém pro zjištění polohy

# Zdroje


[1] **StatCounter**. *GlobalStats* [online] [cit. 2016-04-13]. Dostupné z: http://gs.statcounter.com/#mobile_os-CZ-monthly-201504-201603-bar

[2] **Drifty Co**. Ionic Framework. *OverView* [online] [cit. 2016-04-14]. Dostupné z: http://ionicframework.com/docs/overview/

[3] **Korf, Mario**. Salesforce.com. *Native, HTML5, or Hybrid: Understanding you Mobile Application Development Options*. [online] 2015-04-15 [cit. 2016-04-14]. Dostupné z: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[4] **Mehra, Himashu**.theninehertz.com. *Native vs Hybrid vs Web Application: Which one should you choose?* . [online] 2015-09-13 [cit. 2016-04-15]. Dostupné z: http://theninehertz.com/native-hybrid-web-applications

[5] **Google Inc**. *Running Apps in Android Emulator*. [online] [cit. 2016-04-17]. Dostupné z: http://developer.android.com/tools/devices/emulator.html

[6] **Google Inc**.*Chytré karty Google*. [online] [cit. 2016-04-17]. Dostupné z: https://www.google.com/landing/now/

[7] **Google Inc**. *Activity*. [online] [cit. 2016-04-17]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html

[8] **Konečný, Matěj.** Zdroják.cz. *Vyvíjíme pro Android: Dialogy a activity* . [online] 2012-08-17 [cit. 2016-04-18]. Dostupné z: https://www.zdrojak.cz/clanky/vyvijime-pro-android-dialogy-a-activity/

[9] **Google Inc**. *SQLiteOpenHelper*. [online] [cit. 2016-04-17]. Dostupné z: http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html

[10] **Allen, Grant**. Android 4: průvodce programováním mobilních aplikací. 1. vyd. Brno: Computer Press, 2013. 656 s. ISBN 978-80-251-3782-6.

[11] **Felker, Donn**. Android Application Development for Dummies. For Dummies, 2010. 384 s. ISBN: 978-0-470-77018-4.

[12] **Google Inc**. *IntentService*. [online] [cit. 2016-04-17]. Dostupné z: http://developer.android.com/reference/android/app/IntentService.html

[13] **Vogel, Lars**. Vogella.com. *Android BroadcastReceiver - Tutorial*. [online] 2016-03-03 [cit. 2016-04-18]. Dostupné z: http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html

[14] **Google Inc**. *The Google Maps Directions API.* [online] 2016-04-19

[cit. 2016-04-19]. Dostupné z:
https://developers.google.com/maps/documentation/directions/intro#Introduction

[15] **Oestrich, Eric**. Smartlogic.io. *Organizing Your Android Development Code Structure* [online] 2013-07-09 [cit. 2016-04-19]. Dostupné z:
http://blog.smartlogic.io/2013/07/09/organizing-your-android-development-code-structure

[16] **Google Inc**. Google Places API for Android. *Place Autocomplete.* [online] [cit. 2016-04-20]. Dostupné z: https://developers.google.com/places/android-api/autocomplete

[17] **Google Inc**. *Pickers.* [online] [cit. 2016-04-21]. Dostupné z:
http://developer.android.com/guide/topics/ui/controls/pickers.html

[18] **Google Inc**. *AlarmManager.* [online] 2016-04-21 [cit. 2016-04-21]. Dostupné z:
http://developer.android.com/reference/android/app/AlarmManager.html

[19] **Google Inc**. *Notifications.* [online] [cit. 2016-04-21]. Dostupné z:
http://developer.android.com/guide/topics/ui/notifiers/notifications.html

[20] **Google Inc**. *Fragments.* [online] [cit. 2016-04-22]. Dostupné z:
http://developer.android.com/guide/components/fragments.html

[21] **Google Inc**. *Saving Data.* [online] [cit. 2016-04-22]. Dostupné z:
http://developer.android.com/training/basics/data-storage/index.html

[22] **LeCompte, Celeste**. Gigaom.com. *The App Developer's Guide to Choosing a Mobile Platform*. [online] 2010-02-15 [cit. 2016-04-22]. Dostupné z:
https://gigaom.com/2010/02/15/the-app-developers-guide-to-choosing-a-mobile-platform/

[23] **Xamarin Inc**. *Welcome*. [online] [cit. 2016-04-22]. Dostupné z:
https://developer.xamarin.com/

[24] **Wikipedia**. *Android (operační systém)*. [Online] 2016-03-12 [Cit. 2016-04-23] Dostupný z:
http://cs.wikipedia.org/wiki/Android_(opera%C4%8Dn%C3%AD_syst%C3%A9m)

[25] **Lombardo, Crystal**. NLCATP.com. *Top 8 Pos and Cons of Android*. [Online] 2015-05-10 [Cit. 2016-04-23] Dostupný z: http://nlcatp.org/top-8-pros-and-cons-of-android/

[26] **Wikipedia**. iOS. [Online] 2016-4-14 [Cit. 2016-04-23] Dostupný z:
https://en.wikipedia.org/wiki/IOS

[27] **Wikipedia**. Emulátor. [Online] 2016-4-13 [Cit. 2016-04-23] Dostupný z:
https://cs.wikipedia.org/wiki/Emul%C3%A1tor

[28] **Google Inc.** *Android Studio.* [Online] [Cit. 2016-04-23] Dostupný z: http://developer.android.com/sdk/index.html

[29] **Google Inc.** *App Manifest.* [Online] [Cit. 2016-04-23] Dostupný z: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[30] **Google Inc.** *Lists.* [Online] [Cit. 2016-04-23] Dostupný z: https://www.google.com/design/spec/components/lists.html

[31] **Genymobile**. Features. [Online] [Cit. 2016-04-23] Dostupný z: https://www.genymotion.com/features/

# Obrázky



# Tabulky

# Přílohy

Příloha A: CD se zdrojovým kódem aplikace a instalačním balíčkem

## Abstrakt

Fiala, Tomáš. *Vývoj aplikací pro mobilní telefony*. Bakalářské práce. Plzeň: Fakulta ekonomická ZČU v Plzni, 48s, 2016

Klíčová slova: vývoj, mobilní aplikace, android

Předložená bakalářská práce se zabývá vývojem mobilní aplikace pro správu událostí. Práce se skládá ze dvou základních částí, teoretické a praktické. První část se věnuje porovnáním vybraných platforem, zaměřuje se na jejich rozšířenost a rozebírá současné technologie pro vývoj mobilních aplikací na konkrétních příkladech. Závěr této části je tvořen rozebráním nejpoužívanějších tříd a služeb používaných při vývoji aplikací pro platformu Android s ukázkami zdrojového kódu. V praktické části je nejprve popsán výběr vhodné platformy, technologie, podporovaných zařízení a jejich zdůvodnění. Následně je popsán postup vytvoření layoutu aplikace. Vytvoření samotné aplikace je rozděleno do dvou částí a to na vytvoření aktivit a vytvoření funkcí. V závěru práce je shrnuto testování a uživatelský manuál.

## Abstract

Fiala, Tomáš. *Applications development for mobile devices*. Bachelor thesis. Pilsen: Faculty of Economics, University of West Bohemia, 48p, 2016

Key words: development, mobile applications, android

The presented bachelor thesis describes development of application for event management. The thesis is divided into two basic parts, theoretical and practical. The first part describes comparation of chosen platforms and focuses on its usage amongs users and analyzes current technologies for mobile applications development on specific cases. The end of this part consists dismantling the most used classes and services in application development for the Android platform with samples of source code. The practical part describes the selection of suitable platform, technology, supported devices and its justification. Subsequeltyl is described method for creating application layout. The development of the aplication is devided into two parts. The first part contains activity development and the second creating its functions. The end of this thesis describes testing and user guide.