UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2018 **Zdeněk Novotný**

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Aplikace pro docházkový systém

Zdeněk Novotný

Bakalářská práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Zdeněk Novotný**
Osobní číslo:	**I14153**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace pro docházkový systém**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je navrhnout a realizovat aplikaci pro identifikaci účastníků konferencí Klubu Českých Turistů za využití IRQ kódu. V teoretické části autor navrhne aplikaci, která umožní za využití chytrého telefonu načíst potřebné informace z členské karty Klubu Českých Turistů, data přenést do aplikace na PC (OS Widnows) a umožní jejich základní správu. V praktické části autor realizuje návrh aplikace za využití inteligentních telefonů s OS Android, přenos dat do PC a klientskou aplikaci na PC pro správu získaných dat.

Rozsah grafických prací:

Rozsah pracovní zprávy: **40 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

BURD, Barry A. Java programming for android developers for dummies. First edition. Indianapolis, IN: John Wiley and Sons, 2014. –For dummies. ISBN 978-1-118-61212-5. MEDNIEKS, Zigurd R. Programming Android. 2nd ed. Beijing: O'Reilly, c2012. ISBN 978-1-4493-1664-8.

Vedoucí bakalářské práce: **Mgr. Josef Horálek, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 09. 05. 2018

Zdeněk Novotný

PODĚKOVÁNÍ

Děkuji tímto Mgr. Josefu Horálkovi, Ph.D. za poskytnutí zajímavého tématu bakalářské práce a rovněž za jeho vedení a rady. Doc. Ing. Josefu Kotykovi, Csc. děkuji za podněty z praxe coby člena Klubu Českých Turistů a pánům Ing. Jiřímu Kyselovi, Ph.D. a Ing. Zdeňku Šilarovi, Ph.D. za cenné konzultace ohledně QR kódů a vývoje mobilních aplikací. Dále děkuji vší své rodině, obzvláště pak otci, za vytrvalou podporu ve studiu a konečně jmenovitě slečnám Adéle, Martině, DiS. a Martině, Mgr. za inspiraci i kritiku.

ANOTACE

Práce pojednává o aplikaci pro identifikaci účastníků akcí Klubu českých turistů. V první části je čtenář seznámen s problematikou QR kódů či návrhu aplikací pro OS Android a s návrhem výsledné aplikace. V druhé části je dokumentována implementace této aplikace v jazyce Java s využitím prostředí Android Studio.

KLÍČOVÁ SLOVA

Android, Android Studio, docházkový systém, chytrý telefon, Java, QR kód, quick response

TITLE

An application for attendance system

ANNOTATION

The research discusses an application, which should identify attendants of KČT conferences. The first part focuses on the issue of QR codes or OS Android application development and a suggestion of a concrete application. The second part documents the implementation of the application in Java, using Android Studio IDE.

KEYWORDS

Android, Android Studio, attendance system Java, QR code, quick response, smartphone

Obsah

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API	Application Programming Interface
ART	Android runtime
AVD	Android Virtual Device
IDE	Integrated Development Environment
OS	Operating System
QR	Quick Response
RAM	Random Access Memory
SDK	Software Development kit
USB	Universal Serial Bus

ÚVOD

Klub českých turistů je zájmové sdružení turistů vzniklé v roce 1888. Klub je dodnes aktivní a v pořádá spousty akcí po celé České republice i za jejími hranicemi. [6] Při těchto akcích je snaha účastníky evidovat. Doposud evidence docházky členů probíhala papírovou formou, každý návštěvník se zapsal na docházkový list.

Tato forma evidence je však pomalá a ve výsledných záznamech nelze ani rychle vyhledávat. Jejich dlouhodobá úschova je taktéž náročná, neboť papír trpí vlhkostí a zabírá prostor.

Pro další použití je tak mnohem výhodnější tyto informace digitalizovat, což by ale znamenalo další přepisování papírových záznamů do počítače.

Cílem práce je navrhnout a vytvořit aplikaci, která umožní evidenci účastníků na základě členské karty. Řešení by tedy mělo evidenci zrychlit, zjednodušit a výstupem je uvažován dále zpracovatelný počítačový soubor.

Nejprve bude představen QR kód, neboť právě prostřednictvím QR kódů bude probíhat identifikace. Každá členská karta Klubu Českých Turistů je vybavena QR kódem, který obsahuje jednoznačný identifikátor člena. Aplikace tedy musí umožnit přečtení kódu pomocí fotoaparátu zařízení a zpracování takto získaných dat na výstup.

Dále bude pozornost věnována operačnímu systému Android, protože na něj má být aplikace vytvořena. To má svůj důvod, Android je totiž nejrozšířenějším operačním systémem na poli chytrých telefonů. [2] V souvislosti s tím budou přiblíženy též možnosti vývoje mobilních aplikací a vývojové prostředí Android studio, které je oficiálně doporučovaným nástrojem na vývoj aplikací pro OS Android. [9]

Nakonec bude přistoupeno k samotné implementaci aplikace na základě tohoto návrhu. Výsledkem by měla být aplikace která bude rychlejší a pohodlnější než dosavadní systém. Měla by se snadno používat a být použitelná na většině moderních chytrých telefonů s OS Android, bez jakéhokoli dalšího hardwaru.

Rovněž bude navržena a implementována klientská aplikace pro PC s operačním systémem Microsoft Windows, která umožní základní správu získaných dat.

1 Quick Response kód

Kód byl vytvořen firmou Denso-Wave již v roce 1994 a specifikován roku 2000 standardem ISO 18004, tehdy byl používán především v pásové výrobě k rychlé identifikaci komponent. [5] Masového rozšíření se dočkal až s příchodem chytrých telefonů, které jsou vybaveny digitálním fotoaparátem a softwarem schopným tyto kódy číst i generovat, a tak si QR kód může přeložit v podstatě kdokoli, bez složitého počítání či speciální čtečky.[13]

V této části se zběžně seznámíme s QR kódy a s možnostmi jejich zpracování. QR kódy jsou obdobou čárových kódů, které všichni dobře známe například z obchodů. V porovnání s běžnými čárovými kódy však QR kód pojme mnohem více informací.

Kód je vždy čtvercový a délka jeho strany je od 21 do 177 bodů. V případě největší formy je možné do něj uložit až 4296 znaků. Jelikož při čtení hodně závisí na použitém hardwaru a může dojít ke zkreslení snímaných dat, obsahuje navíc autokorekci, která umožňuje obnovu až třiceti procent (při použití nejvyšší korekce chyb - úrovně H) snímaných informací.[5]

Do QR kódu můžeme uložit libovolný text. Využívá se například k předání internetové adresy, hesla k Wi-Fi, kontaktních údajů, souřadnic globálního polohového systému, k propagačním účelům a podobně.

Informace jsou uloženy binárně v matici černých a bílých čtverců. Tři vrcholy jsou vyhrazeny pro poziční značky. Spojnice mezi jejich vnitřními rohy jsou vždy pravidelně přerušované čáry, střídavě černý a bílý čtverec. v kódu se pak mohou vyskytovat v závislosti na velikosti a zvolené korekci chyb další, menší čtyřúhelníky, což jsou tzv. zarovnávací značky. U zarovnávacích značek jsou zakódovány informace o verzi a formátu kódu a ve zbytku obrázku jsou již samotné informace. [5]

Obrázek 1 QR kód, zdroj: vlastní

Informace lze samozřejmě binárně číst, překládat na znaky a sestavit tak výsledný text, opačně lze zase text zakódovat do vlastního QR kódu. To ovšem není nutné, neboť existuje spousta aplikací, které to udělají za nás. A to nejen na počítač, ale i pro chytré telefony.
Někteří výrobci takové nástroje vydávají coby standardní výbavu svých chytrých telefonů, ale i uživatel, který takovým programem vybaven nebyl, si jistě vybere z bezplatných aplikací v obchodě Google Play. Za zmínku rozhodně stojí aplikace Barcode Scanner či QR droid.

2 Android

Účelem této bakalářské práce je implementovat aplikaci pro OS Android. Proto se nyní zaměříme na tento operační systém.

2.1 Charakteristika

Android je operační systém určený pro mobilní zařízení. Základem je jádro linuxu a jde o open source software. Původně byl určen pro chytré telefony, v dnešní době je nasazován též na tabletech, chytrých televizích, hodinkách a dalších. Vyvíjen je firmou Google, pod hlavičkou konsorcia Open Handset Alliance. [2][4]

Android je nasazován na rozmanitý hardware, je třeba brát v úvahu nejen různé procesory a čipové sady, ale i různé rozlišení displejů a vybavení (geolokace, fotoaparát...). Také je vhodné aby byl systém co nejúspornější, neboť mobilní zařízení jsou zpravidla poháněna baterií.

Android nesestává pouze z operačního systému. Součástí je uživatelské prostředí, ovladače pro mobilní operátory, ovladače pro výrobce zařízení a též vývojové prostředí. [2]

V systému Android je dodatečný software, tedy aplikace, dostupný z obchodu Google Play.

2.2 Historie

V říjnu roku 2003 založili v Kalifornii Andy Rubin, Rich Miner, Nick Sears a Chris White společnost Android Inc. s cílem vytvořit nový, moderní operační systém pro mobilní telefony. O dva roky později společnost odkoupil Google a pod vedením Andyho Rubina byl zahájen vývoj platformy OS Android, operačního systému založeném na linuxovém jádře. [2]

V roce 2007 bylo založeno konsorcium Open Handset Alliance. Členy konsorcia jsou mnozí mobilní operátoři, výrobci mobilní techniky a komponent či producenti softwaru. [2][4]

Open Handset Alliance si dala za cíl především vývoj otevřeného standardu pro mobilní zařízení. Důraz je kladen na rozvoj mobilních aplikací a snížení nákladů spojených s vývojem a distribucí. [2][4]

Současně s ohlášením vzniku tohoto seskupení byla představena platforma Android a zamýšlený cíl, tedy masové použití této open-source platformy napříč mobilními zařízeními a jejich výrobci. Nedlouho poté následovalo i vydání Android SDK, taktéž pod open-source licencí, pro vývoj aplikací pro platformu Android. [2]

V roce 2008 bylo představeno první komerční zařízení s operačním systémem Android, kterým byl mobilní telefon T-Mobile G2, u nás prodáván později coby HTC Dream. V roce 2009 bylo již zařízení s Androidem představeno přes dvacet. V roce 2010 se stal Android nejčastějším

operačním systémem na prodaných zařízeních, v roce 2012 měl dokonce nadpoloviční podíl a v roce 2013 dosáhl drtivého osmdesátiprocentního podílu na trhu nových mobilních zařízení. V současnosti je Android nejčastějším operačním systémem vůbec. Netáhne ho jen velký trh chytrých telefonů, dominuje též tabletům, chytrým hodinkám, chytrým televizím a dalším mobilním zařízením. Aktivně používaných zařízení s operačním systémem Android je v současné době přes dvě miliardy. [2]

2.3 Architektura

Operační systém Android má pětivrstvou architekturu. [12] Představme si je nyní podrobněji.

2.3.1 Jádro operačního systému

Stará se o přímou komunikaci s hardwarem, abstrahuje hardware pro vyšší vrstvy. Jedná se o modifikované Linuxové jádro, očesané o grafické uživatelské rozhraní X a některé knihovny. Jelikož vychází z Linuxu, je i jádro Androidu monolitické. [2][4][12]

2.3.2 Knihovny

Předpřipravené sady funkcí a struktur, které tvoří logické celky, jsou shlukovány do knihoven. Tyto knihovny jsou pak použitelné systémem nebo ostatním softwarem. [2][12]

2.3.3 Android Runtime

Android Runtime se stará o instalaci a spouštění aplikací.

Do verze Android 4.3 obsahovala virtuální stroj Dalvik (DVM). Dalvik nahrazoval Java Virtual Machine, který není volně šiřitelný. Za jeho vývojem stál tým Dana Bornsteina z firmy Google. Oproti JVM byl u DVM kladen důraz na úsporu energie z důvodu použití na mobilních zařízeních. Tento virtuální stroj byl později nahrazen Dalvikem Turbo, vyvíjeným firmou Myriad Group. Dalvik Turbo je zpětně kompatibilní s původním Dalvikem, ale je mnohem rychlejší a úspornější. [2][12]

Tato vrstva dále obsahuje knihovny jazyka Java, chybí však knihovny grafického uživatelského rozhraní. Ty byly nahrazeny odpovídajícími knihovnami přímo pro Android.

Překlad kódu aplikace má tak několik stádií. Nejdříve je zkompilován do bytekódu Javy úplně stejně jako u standardního Java programu. Výsledný bytekód pak kompilátor Dalviku zkompiluje znovu do vlastního bytekódu. Ten je pak spuštěn na virtuálním stroji Dalvik. Každá spuštěná aplikace má vlastní instanci Dalvik virtual machine. [2][12]

Od verze 4.4 (včetně) se používá dopředná kompilace. Spouštění kompilovaného kódu je mnohem efektivnější, aplikace jsou tedy rychlejší a úspornější co se týče spotřeby energie.

Uváděn je ve výsledku až dvojnásobný výkon zařízení a přibližně o třetinu vyšší výdrž na baterii. Naopak instalace aplikace trvá déle, neboť navíc oproti použití DVM kompiluje do nativního kódu a při kompilaci optimalizuje do konečné podoby. [2][12]
Princip je takový, že se kód nejprve přeloží do Java bytekódu a ten je pak znovu kompilován Dalvikem, stejně jako při použití DVM. Výsledný Dalvik bytekód je však navíc ještě v rámci instalace zkompilován do nativního kódu zařízení. [2][12]

2.3.4 Aplikační framework

Tato vrstva je stěžejní pro vývojáře aplikací na operační systém Android. Obsahuje spoustu služeb, které lze volat přímo z aplikací. Umožňuje například komunikovat s aplikacemi běžícími na pozadí, posílat upozonění na stavový řádek, komunikovat s hardwarem (skrze nižší vrstvy), zpřístupňovat data z ostaních aplikací či reagovat na ovládací prvky. Obsahuje mimo jiné tyto služby:

- **Sada View** je použita pro sestavení grafického uživatelského rozhraní, obsahuje rozličné widgety, jako tlačítka, textová pole a podobně.
- **Content providers** umožňují přístup k obsahu jiných aplikací, například k fotkám pořízeným k tomu určenou aplikací nebo kontakty aplikace sociální sítě.
- **Resource manager** poskytuje přístup k nekódovým zdrojům, tedy kupříkladu textu či obrázkům.
- **Notification manager** je odpovědný za upozornění ve stavovém řádku systému.
- **Activity manager** má na starost celý životní cyklus aplikace od jejího vytvoření po uvolnění zdrojů. [2][12]

2.3.5 Aplikace

Samotná aplikace určená pro koncového zákazníka.

2.4 Instalace aplikací

Základním zdrojem aplikací v operačním systému Android je obchod Google Play. Obchod obsahuje vedle placených aplikací i spoustu aplikací zcela zdarma. Značná část z aplikací zadarmo však vydělává pomocí reklam.
Nabídka aplikací je obrovská a neustále roste, za což Android vděčí především obrovské uživatelské základně a vývoji v jazyce Java. Do obchodu se lze dostat i pomocí prohlížeče například na PC. Objednané aplikace se pak automaticky stáhnou do zařízení spolu s aktualizacemi. [2]

Existují i další repozitáře aplikací, namátkou třeba F-Droid. Dalším způsobem je nahrávání aplikací přímo z počítače pomocí SDK nebo instalace hotového balíku (Android projekt zkompilovaný do souboru s příponou APK). K tomu je zapotřebí povolit v nastavení instalaci z neověřených zdrojů. [2]

Lze také upravit samotný operační systém, nebo jej dokonce nahradit jiným (rooting). Tyto postupy ovšem samozřejmě obvykle vedou ke ztrátě záruky na zařízení a mohou zařízení nenávratně poškodit. K tomuto se váže hezký výraz „bricking“ (brick = cihla), který, jak už jméno napovídá, popisuje proces softwarových změn, po kterých je zařízení využitelné nanejvýš jako těžítko či projektil.

Podotknu, že na svých dvou chytrých telefonech s Androidem jsem změnu operačního systému absolvoval několikrát, a to vždy úspěšně a s uspokojivým výsledkem.

2.5 Licence

Android je prezentován jako otevřený systém a ze značné části skutečně otevřený je. Knihovny, moduly jádra (Linux), aplikační rozhraní a dokonce i mnohé aplikace jsou open-source. Některé součásti jsou však uzavřené a vlastněné firmou Google. Několik technologií, které Android může obsahovat, je navíc patentovaných a výrobce zařízení, které je využívá, si je tak musí licencovat. [2]

Přesto je otevřený natolik, aby si jej výrobci mohli upravovat pro svůj hardware a aplikace, což usnadňuje integraci. Android navíc může být využíván ke komerčním účelům. [2] Díky tomu je nasazovaný v podstatě na jakákoli mobilní zařízení a dočkal se obrovského úspěchu napříč vývojáři i koncovými uživateli.

3 Vývoj mobilních aplikací

V této části si povíme obecně o vývoji mobilních aplikací. Při rozvaze je zapotřebí vzít v potaz náročnost aplikace, cílovou skupinu, zdroje a mnohé další aspekty. Možnosti jsou v zásadě tři:

- nativní aplikace,
- hybridní aplikace
- a mobilní web. [1][17]

3.1 Nativní aplikace

Nativní aplikace je vyvíjena pro konkrétní platformu. V případě OS Android probíhá implementace v jazyce Java.

Hlavní výhodou tohoto přístupu je výkon, aplikace je překládána do nativního kódu a pracuje rychleji a úsporněji než hybridní. Také má přístup ke vší funkcionalitě API a využívá ovládací prvky systému, takže aplikace lépe „pasuje“ do systému.

Nevýhodou je pak nepřenositelnost – pro jiný systém je třeba napsat celou aplikaci znovu, neboť používá jiný programovací jazyk i API. Změna aplikace pak znamená změnu pro každý systém, což je neefektivní a v praxi drahé. [1][17]

3.2 Hybridní aplikace

Hybridní aplikace využívá tzv. WebView. Jde o instanci nainstalovaného internetového prohlížeče, ve kterém se spustí webová aplikace. To umožňuje spustit aplikaci na kterékoli platformě, nedosahuje však takové efektivity jako nativní. Moderní frameworky pro hybridní programování podporují vlastní WebView pro cílové platformy a dokonce odpovídající grafické sady, aby působily jako nativní aplikace. To je důležité pro uživatelský dojem.

Výhodou je zde hlavně přenositelnost, navíc lze aplikaci testovat na webovém prohlížeči. Na druhou stranu je výpočetně náročnější, není tedy vhodná pro slabý hardware či složité výpočty. [1][17]

3.3 Mobilní web

Mobilní web je vhodný zejména pokud již webovou aplikaci máme, anžto ji stačí přizpůsobit pro vstupní a výstupní rozhraní mobilního zařízení. Mnohdy tak postačí změny v kaskádových stylech a výsledkem je univerzální aplikace, kterou navíc není třeba distribuovat skrze obchody ani instalovat na cílovém zařízení.

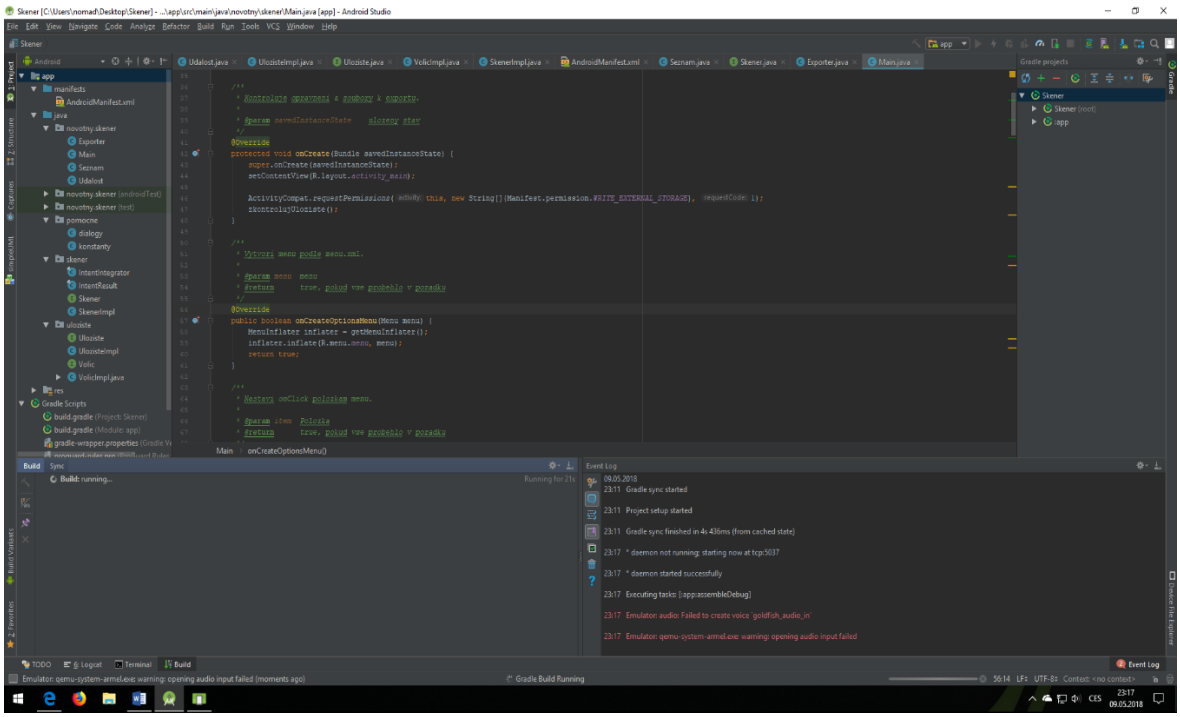
Nemůže však využívat všech možností API a je velmi obtížné přizpůsobit vzhled cílové platformě, nemluvě o odezvě či nutnosti být při používání připojený k internetu. [1][17]

4 Android Studio

V současné době je oficiálně doporučovaným nástrojem pro vývoj aplikací na operační systém Android vývojové prostředí Android Studio. Jeho vývoj začala společnost JetBrains a založila jej na komunitní verzi svého vývojového prostředí IntelliJ IDEA. Google tento projekt odkoupil a oficiálně představil v květnu roku 2013 coby nástupce Android developer Tools plug-inu ve vývojovém prostředí Eclipse. Android studio je dostupné pro operační systémy Windows, Mac OS X a Linux a je zcela zdarma. [3][9][10][14]

Vývojové nástroje jsou součástí Android Software Development Kitu.

- **SDK Tools** – Tvořeny nástroji pro debugging, testování, správu virtuálních strojů Android, emulátor Androidu, analýzu grafického rozvržení a dalšími potřebnými programy.
- **SDK Platform-tool** – Obsahuje další nástroje pro vývoj, které jsou ale závislé na verzi platformy. Jsou tedy aktualizovány s každou novou verzí.
- **Android SDK platforms** – Obsahuje zdroje platformy. Pro každou instalovanou platformu obsahuje sadu knihoven, systémový obraz, ukázkové kódy či skiny emulátoru. Kompilace Android projektu a běh virtuálního stroje Android vyžaduje alespoň jednu platformu. [2]



Obrázek 2 Android Studio, zdroj: vlastní

4.1 Instalace

Instalace probíhá velmi jednoduše. Stačí stáhnout balík a instalátor připraví kompletní vývojové prostředí včetně Android SDK. K běhu a kompilaci je zapotřebí také Java JDK. Součástí Android SDK je i emulátor, takže hned po instalaci můžeme v Android studiu vytvářet virtuální stroje s operačním systémem Android.

4.2 Založení projektu

Vytvořit lze knihovnu, nebo přímo klientem spouštěnou aplikaci. Kromě klasických nastavení jako je umístění nebo jméno projektu se také vybírá druh cílového zařízení, barevné schéma či cílová a minimální verze.

Na základě druhu cílového zařízení Android studio přizpůsobuje grafický návrhář a možnosti. Například chytré hodinky budou zřejmě pracovat s řádově nižším rozlišením než například chytrá televize.

Barevné schéma poskytuje přednastavené sady barev pro grafické komponenty. Ty lze pochopitelně následně ladit dle potřeby.

Minimální verze je pak velmi důležitá. Určuje nejstarší generaci operačního systému Android, od které bude možné aplikaci použít. Přestože je tu s námi již dlouho Android 8, jeho nejpoužívanější generací je Android 5 a běžně se setkáme i s Androidem verze 4 či dokonce 3. [1] Je tedy vhodné zachovat co největší zpětnou kompatibilitu.

Na druhou stranu se od zvolené verze Androidu odvíjí možnosti, které můžeme použít. Obecně lze říci, že čím nižší cílová verze, tím více potenciálních zákazníků bude po zveřejnění aplikace mít, ovšem tím více se musí vývojář omezovat. Například pro požadavky o přístupová práva za běhu programu je třeba minimálně Android 6. I výběr minimálního a cílového API lze pochopitelně změnit později.

4.3 Struktura projektu

Projekt se skládá ze tří základních složek. Struktura v souborovém systému je ve skutečnosti jiná, toto rozdělení slouží k snadné orientaci ve zdrojích, tedy k zefektivnění práce a lze jej upravit dle osobních preferencí.

4.3.1 manifests

V této složce je soubor AndroidManifest.xml a případné další manifesty. Ty definují zásadní vlastnosti aplikace, namátkou jméno aplikace, vyžadovaná oprávnění či použité aktivity. [9]

4.3.2 java

Složka java obsahuje zdrojové kódy Javy včetně případného JUnit testu, zde je veškerá aplikační logika. [9]

4.3.3 res

Ve složce res jsou další zdrojové soubory. Najdeme tu například definici grafického rozložení, řetězce používané v GUI, obrázky, ikonu aplikace a další. [9]

4.4 Design

Design aplikace se vytváří v grafickém návrháři, který funguje metodou drag-and-drop komponenty. Vytvářený styl si návrhář zapisuje do XML souboru. Alternativně lze zapisovat přímo do XML a Android studio naopak zapisované změny dynamicky zobrazuje v návrháři.

4.5 Základní součásti aplikace

Aplikace pro operační systém Android sestávají z aktivit, služeb, content providerů a broadcast receiverů. Každá tato komponenta je popsána v souboru AndroidManifest.xml, který je součástí projektu.

4.5.1 Aktivita

Aktivita je v podstatě obrazovka, tedy sada ovládacích či zobrazovacích prvků, které vidí uživatel, a akcí, které jsou na ně vázané. Aktivity lze přepínat – jedna aplikace má obykle hned několik aktivit. Aktivity si mezi sebou mohou předávat informace pomocí Intentů.
K zahájení aktivity je třeba založit proces, alokovat paměť a aktivitu vykreslit. Zahájení aktivity je tedy celkem náročný proces a nemělo by k němu docházet častěji, než je opravdu třeba.
Proto je Android vybaven správcem aktivit (Activity Manager), který řídí celý životní cyklus každé aktivity. Pro správu využívá zásobník aktivit, tedy pokud zahájí novou aktivitu, uloží ji na zásobník nad všechny ostatní a pokud tuto novou aktivitu ukončí, na vrcholu zásobníku je předchozí, přerušená. [2][7]

Životní cyklus aktivity:

- **start aktivity** – založení procesu, alokace paměti, inicializace, vykreslení grafického rozvržení aktivity;
- **běh aktivity** – aktivita je vykreslena na displeji a čeká na interakci uživatele, v tomto stavu může být vždy pouze jedna aktivita;

- **zastavení aktivity** – aktivita byla zahájena ale momentálně není zobrazována, v popředí je jiná aktivita – přerušená aktivita je tedy kdesi v zásobníku; případě bylo vyvoláno modální dialogové okno a to aktivitu překrývá, aktivita nereaguje;
- **přerušení aktivity** – aktivita byla zastavena (není v popředí) a zařízení nemá dostatek paměti; Activity Manager aktivitu zničí a v případě potřeby ji později zase obnoví;
- **ukončení aktivity** – aktivitu již nemá smysl udržovat v zásobníku, její zdroje jsou uvolněny a proces zabit. [2][7]

4.5.2 Služba

Služba je proces na pozadí, jeho aktivita se nepromítá do grafického rozhraní. Využití najde především při časově náročnějších úlohách, například při komunikaci s databází nebo zpracování většího oběmu dat. Pokud bychom takovou úlohu zadali aktivitě, mohl by operační systém na základě dlouhé odezvy dojít k závěru, že se aktivita zasekla. [4][16]

Službu lze spustit pomocí metody startService() nebo bindService(). V prvním případě se služba provede a ukončí, nebo může být ukončena jinou komponentou. V druhém případě je vázána na komponentu, která ji spustila. Pouze tato rodičovská komponenta pak může službu ukončit. Službu lze případně svázat s více komponentami a k jejímu ukončení pak dojde až po odpojení všech rodičů. [2][7]

Životní cyklus služby

- **zavolání služby** – založení procesu, alokace paměti, inicializace služby;
- **běh služby** – služba pracuje na pozadí;
- **ukončení služby** – služba byla ukončena. [2][7]

4.5.3 Content provider

Content provider zprostředkovává výměnu dat mezi jednotlivými aktivitami, nebo celými aplikacemi. Umožňuje přístup k datům aplikace jiným aplikacím, je-li to povoleno. Data mohou být v souborech, v databázi SQLite či na webu. [2][7]

S content providerem se pracuje podobně jako s databází metodami insert, query, update a delete. To umožňuje snadnou práci s daty jiné aplikace například při nahrazení pohodlnější alternativou. Nainstalujeme-li místo standardního správce textových zpráv jiný, může nová aplikace s odpovídajícím oprávněním spravovat zprávy svého předchůdce. [2][7]

4.5.4 Broadcast receiver

Ani tato komponenta nedisponuje grafickým rozhraním. Běží na pozadí a sleduje okolí aplikace. Naslouchá oznámením a případně na ně reaguje.V rámci reakce na oznámení může i sama spouštět nové komponenty. Aplikace může také vyvolávat oznámení, a to jak systémová tak svá vlastní. Mezi ta systémová patří třeba vybitá baterie či nová SMS. [2]

4.6 Ladění

Nedílnou součástí vývoje aplikací je jejich testování. Vyzkoušet aplikaci lze samozřejmě na skutečném zařízení, ale připravíme se tak o možnost sledovat co se děje pod vrstvou uživatelského rozhraní. Ladění pak může být extrémně náročné, neboť je často obtížné identifikovat původ problému.

Součástí Android SDK je proto možnost sledovat běh aplikace na skutečném zařízení připojeném k počítači. Stačí pomocí USB připojit zařízení, povolit na něm USB ladění a instalaci z neznámých zdrojů a na počítači v Android Studiu spustit aplikaci. Po sestavení se automaticky nainstaluje a spustí na připojeném zařízení. Nemonitoruje přitom jen výstup z ladění aplikace, ale také využití systémových prostředků zařízení.

Další možností je pak emulátor, kde se celé testování odehrává na počítači. Android studio obsahuje přednastavené virtuální stroje Nexus, umožňuje ale i vytvoření vlastních zařízení. Nastavuje se mimo jiné druh procesoru, velikost displeje, paměti RAM, úložiště (i externí), Android API či kamery. Ty lze emulovat zcela nebo propojit s kamerou počítače. Ne každou situaci lze ale emulátorem věrohodně napodobit (např. přenos souboru přes bluetooth). [2][9][10][14]

Jak samotné Android studio, tak emulátory, jsou však velmi paměťově náročné. Jako příklad dám laptop, na kterém jsem aplikaci vyvíjel, HP ProBook 4320s s procesorem Intel Core i3 a 4 GB operační paměti.

Operační systém Arch Linux zabíral po startu kolem 100 MB RAM, přesto stačilo spustit jen samotné Android studio a paměť byla využita téměř zcela. Se zapnutým virtuálním strojem Android pak sestavení mé aplikace trvalo kolem deseti minut, se zapnutým webovým prohlížečem či jinou další aplikací doba sestavení přesahovala čtvrt hodiny. Po rozšíření na 8 GB operační paměti se zkrátila na řádově sekundy až desítky sekund.

Byť 4 GB RAM vskutku není v dnešní době hodně, musím konstatovat, že mě využití paměti Android studia nemile překvapilo. Alternativou je zmiňované ladění přes USB, kde není zapotřebí AVD, na druhou stranu je potřeba skutečné zařízení s Androidem odpovídající verze.

A jelikož aplikaci obvykle nechceme funkční – a tedy testovanou - jen na jednom, ale na mnoha různých zařízeních s rozličnými rozlišeními displejů, výkony i verzemi Androidu, je použití emulátorů samozřejmě výrazně snazší a levnější než nákup všech relevantních cílových strojů.

4.7 Zpětná kompatibilita

Před uvedením Android studia bylo výchozím nástrojem pro tvorbu Android aplikací vývojové prostředí Eclipse s ADT pluginem. Android studio sice neumožňuje přímo otevřít projekt z Eclipse, lze jej však z Eclipse vyexportovat a následně v Android studiu naimportovat. [14]

4.8 Uživatelské rozhraní

Uživatelské rozhraní Android studia je velmi propracované. Vychází ostatně z IntelliJ Idea od firmy JetBrains, která má v tomto ohledu spoustu zkušeností. [3][9][10][14] Středu obrazovky vévodí okno kódu či návrháře, po stranách jsou pak různé nástroje, průzkumníky, náhledová okna a podobně. Rozložení je intuitivní a modulární, takže ho lze snadno přizpůsobit osobním potřebám.

Rozhraní nabízí bohatý výběr nástrojů, které mají usnadnit a zefektivnit práci programátora. I za ně Android Studio vděčí především svému původu. Kontext nástrojů se mění dle zaměření – jiný je pochopitelně v návrháři, ale může se měnit i v závislosti na druhu otevřeného dokumentu.

V nástrojích lze rychle vyhledávat jednoduše kliknutím na okno nástrojů a psaním hledaného výrazu. Lze samozřejmě vyhledávat i v kódu. Android Studio obsahuje také řežim „Distraction Free Mode“, který vypne všechny nástrojové panely, okna a upozornění, aby se programátor mohl soustředit pouze na kód otevřeného dokumentu.

4.9 Kompletace kódu

Kompletace kódu vychází taktéž z IntelliJ Idea, lze ji rozdělit na 3 skupiny:

- základní kompletace,
- chytrá kompletace,
- kompletace výrazů.

4.9.1 Základní kompletace

Našeptává proměnné, typy, metody, výrazy a podobně. Po druhém stisku zkratky nabízí i prvky které nejsou momentálně přístupné (například nejsou naimportovány, nejsou viditelné ze současného kontextu...). [9]

4.9.2 Chytrá kompletace

Nabízí možnosti podle aktuálního kontextu. Snaží se odhadnout záměr programátora a dokáže generovat celé bloky kódu. Po dvojitém stisku obsahuje i celé řetězce možností.

Lze sem též zařadit rychlou opravu, která slouží k úpravám a opravám bloků kódu které vykazují chyby nebo například nadbytečný kód. Tato fukce pochopitelně do značné míry s chytrou kompletací spolupracuje. [9]

4.9.3 Kompletace výrazů

Dokončuje výrazy, přidává chybějící závorky, formátuje kód a tak dále. [9]

4.10 Gradle

Jednou z hlavních úloh Android Studia je všechny vytvořené soubory nakonec sestavit do jediné aplikace. O sestavení se stará Gradle Build System. Projekt je sestavován dle nastavení v souborech build.gradle, kde jsou uvedeny všechny náležitosti projektu jako například cílová verze, jméno a podobně, včetně závislostí aplikace. Úpravou těchto souborů lze zasáhnout do sestavování projektu, přizpůsobit ho či přidat další možnosti.

Lze například na základě jednoho projektu a jeho modulů generovat několik APK balíčků, které budou disponovat různými možnostmi nebo podstrčit zdroje ladícímu režimu, které však v produkční verzi nemají co dělat.

Více výsledných aplikací má typické využití při souběžně volné a placené verzi aplikace, kdy volná aplikace má obvykle omezenou funkcionalitu, nebo obsahuje reklamy. Také lze tvořit různé aplikace například na základě rozlišení displeje cílového zařízení. Tyto projekty přitom dále sdílí stejné zdroje.

Vývojové profily pak umožňují kupříkladu definovat aplikaci zdroje, které pomohou při ladění, ale nebudou součástí produkční verze programu. Příkladem jsou přihlašovací údaje, které se nám nechce zadávat při každém ladění, výchozí data tabulek a podobně. Profilů si lze definovat libovolné množství.

Samotné sestavení pak Gradle provádí dle závislostí definovaných pro zvolený profil. Soubor build.gradle existuje pro projekt jako celek a další stejnojmenné soubory pak může mít každý modul zvlášť. [3][10][14]

4.11 Uveřejnění aplikace

Když je aplikace hotová a připravená ke vstupu na trh, je na čase ji nahrát do obchodu Google Play. V Android studiu použijeme podvolbu sestavení „Vytvořit podepsaný soubor APK“.

Tento balík obsahuje celou sestavenou aplikaci včetně všech zdrojů a může být nahrán do obchodu.

Při vytváření je třeba zadat nebo vytvořit klíč. Tento soubor slouží později k ověření, zda je nová verze aplikace zasílána odpovědným člověkem. Případnému utočníkovi tak k nahrání podvodné aktualizace nestačí jen přístup k účtu Google tvůrce, ale potřebuje též klíč. Tento soubor je třeba uchovávat v bezpečí, bez něj totiž není možné vydat novou verzi aplikace. [10]

5 Docházkový systém

Nyní přistoupíme k návrhu samotné aplikace, přičemž berme v potaz informace uvedené výše. Je dobré zvážit pro koho je aplikace určena a co vlastně řeší, stejně jako možnosti konkrétního jazyka a API.

5.1 Cílová skupina

Cílovou skupinou jsou pořadatelé konferencí z řad Klubu českých turistů. Je třeba mít na paměti, že s aplikací bude velmi pravděpodobně zacházet člověk, který nemá s podobnými systémy mnoho (ne-li žádné) zkušenosti.

Vzhled aplikace a ovládací prvky mají zásadní vliv na budoucnost celého projektu, neboť jsou exponovány uživateli a tak přímo ovlivňují jeho zkušenost s aplikací. Ruku v ruce s touto zkušeností pak jde ochota uživatele aplikaci používat. V současnosti je standardem, že i nezkušený uživatel by měl být schopen novou aplikaci snadno pochopit a ovládat. Vzhled a komfort jsou tak mnohdy důležitější než samotná funkcionalita.

5.2 Platforma

Ze zadání je požadována implementace pro operační systém Android, nikoli multiplatformní aplikace. Nemá tedy smysl připravit se o výhody nativního řešení. Další výhodou této volby je nativní grafické prostředí, což je pro uživatele příjemnější, jak už bylo zmíněno dříve.

5.3 Programovací jazyk

Jelikož bylo učiněno rozhodnutí vytvořit nativní aplikaci pro OS Android, bude implementace realizována v jazyce Java.

5.4 Vývojové prostředí

Již dříve jsme se seznámili s Android Studiem, které je oficiálně doporučovaným nástrojem pro tvorbu aplikací na OS Android. S tímto vývojovým prostředím se také velmi dobře pracuje, programátor se v něm snadno zorientuje a grafický návrhář je až překvapivě pohodlný na používání. Systém drag-and-drop je zde totiž velmi přesný, narozdíl od těch používaných ve většině ostatních vývojových prostředí. K implementaci aplikace jsem proto zvolil právě Android Studio.

5.5 Jazyk aplikace

Vzhledem k cílové skupině aplikace lze předpokládat, že bude obsluhována česky mluvícím uživatelem. Celá aplikace bude proto implementována v českém jazyce. Přesto, pro případ že

by byla přeci jen žádoucí lokalizace, budou všechny použité řetězce zařazeny mezi konstanty v jediné statické třídě. Tak bude možné jazyk aplikace rychle a snadno změnit.

5.6 Čtení QR kódu

Každý člen Klubu českých turistů disponuje členským průkazem, na kterém je QR kód. Tento QR kód musí aplikace přečíst. Cílové zařízení tedy musí mít fotoaparát a odpovídající software. Implementovat vlastní čtení z obrazových dat fotoaparátu by bylo nesmyslně složité, v Obchodě Google najdeme hned několik aplikací k tomu určených.

Z nich jsem vybral Barcode Scanner, který nabízí snadnou integraci pomocí tzv. „IntentIntegratoru“, poskytovaného pro tyto účely přímo samotnými tvůrci. [11]

Barcode Scanner je aplikací určenou ke čtení čárových a dalších obrazových kódů, QR nevyjímaje a pomocí zmiňovaného IntentIntegratoru (což je ve své podstatě třída jazyka Java, která aplikaci Barcode Scanner spustí a komunikuje s ní) ji lze do určité míry nastavit.

Pro účely aplikace bude nastavena pouze pro čtení QR kódů.

5.7 Zpracování dat

V QR kódu na členském průkazu je předepsaným způsobem zaneseno osobní číslo člena, jeho jméno, příjmení a datum narození. K získání jednotlivých údajů tak stačí přečtený řetězec rozdělit a jednotlivé položky uložit do vhodné struktury. K takovému úkolu poskytuje Java hned několik nástrojů.

Není přitom od věci si uvědomit, že člen může mít i prostřední jméno (jména) a při implementaci mít tuto možnost na zřeteli.

5.8 Uchování dat

Uchování dat lze řešit různými přístupy. V rámci této aplikace bylo zvoleno ukládání do souborů. Vzhledem k povaze údajů byl vybrán formát CSV, který umožní správu dat v libovolném tabulkovém procesoru. Takový program pak nabízí možnost filtrování, řazení, statistik návštěvnosti, aktivity členů a podobně.

Samozřejmě by bylo vhodné záznamy ukládat na paměťovou kartu, je-li k dispozici. Android API ale bohužel tuto variantu přímo nenabízí. [15] Nabízí vnější úložiště, které je výchozí složkou při procházení souborů připojeného zařízení z počítače, to ale může být emulované. Na druhou stranu emulované být nemusí a pak jeho zavolání bez externího úložiště zavolá chybu.

Nabízí se využití interního úložiště, jenže to skrývá v systémových složkách pod názvem android.[jméno aplikace], což není vhodné místo kde nechat běžného uživatele hledat jeho soubory.
Proto aplikace musí vhodným způsobem pracovat s oběma úložišti.

5.9 Klientská aplikace na PC

V zadání je požadována též aplikace pro správu získaných dat. K různým statistikám, filtrů, a podobným poslouží mnohem lépe kterýkoli tabulkový procesor. Proto bude počítačová aplikace sloužit především k finalizaci souborů. Musí umožnit načtení souborů, jejich slití, eliminaci duplicitních záznamů a uložení výsledného CSV souboru.
Slévání souborů umožní pořizovat více evidencí na jedné akci paralelně, v případě vysoké návštěvnosti tak lze účastníky evidovat hned několika chytrými telefony, což evidenci zřejmě násobně zrychlí.

6 Praktické řešení

V rámci řešení byly aplikace implementovány dle předešlého návrhu. K testování mobilní aplikace byly použity:

- emulovaný Nexus 5 s OS Android 7,
- Samsung J1 s OS CyanogenMod 11, Android 4,
- Huawei L21 s OS Android 5,
- Huawei P9 Lite s OS Android 6,
- Samsung S3 s OS Lineage 14, Android 7.

Nejdříve se zaměříme na mobilní aplikaci, která je hlavním předmětem práce. Vzhledově byla aplikace navržena jednoduše a přehledně, aby její používání bylo pokud možno intuitivní. Skládá se celkem ze čtyř aktivit a několika dalších tříd.

6.1 Kompatibilita

Po důkladném zvážení možností bylo za minimální zvoleno SDK 19 (Android 4.4) a za cílové SDK 25 (Android 7.1). Osobně považuji verze starší než Android 6 za zastaralé a nevhodné k běžnému používání, především pro hardware na kterém se tyto zpravidla vyskytují, ale Android 4.4 KitKat je stále hojně rozšířen a nelze jej ignorovat.

Obešel jsem se tedy například bez lambda kalkulu a zachoval značnou zpětnou kompatibilitu. Tyto informace se zadávají již při vytváření projektu, lze je ale kdykoli změnit. Jsou uloženy ve skriptu Gradlu.

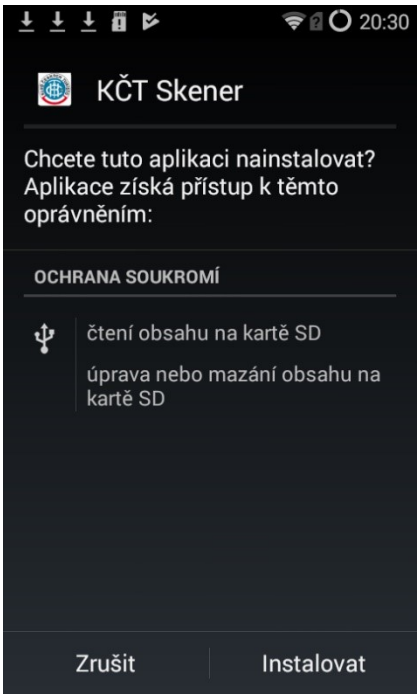
```
compileSdkVersion 25
defaultConfig {
    minSdkVersion 19
    targetSdkVersion 25
```

6.2 Oprávnění

Aplikace vyžaduje oprávnění k zápisu do vnějšího úložiště, aby mohla ukládat soubory. Jelikož je ale kompatibilita zachována až k Androidu 4.4 a v Androidu 6.0 došlo k výrazné změně v přidělování práv, je nutné oprávnění obstarat hned dvakrát.

Poprvé pro Android starších verzí (před Androidem 6), kdy si aplikace žádala o všechna práva hned při instalaci. Toho docílíme položkou v souboru AndroidManifest.xml.

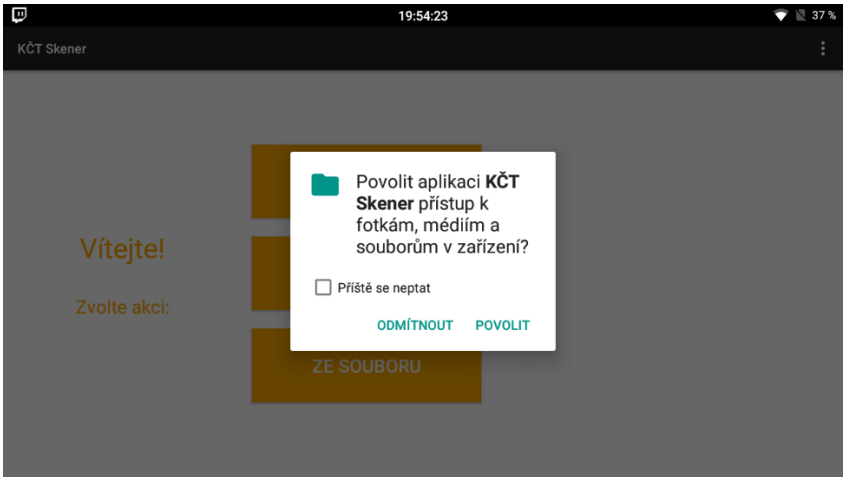
```
<uses-permission
  android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```



Obrázek 3 Aplikace - požadavek o oprávnění při instalaci, zdroj: vlastní

Zadruhé pak pro moderní OS Android, kde si aplikace může o práva žádat za běhu. Na systému Android 6.0 a vyšším je totiž výše uvedená informace ignorována.

```
ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
```



Obrázek 4 Aplikace - požadavek o oprávnění za běhu

6.3 Aktivity

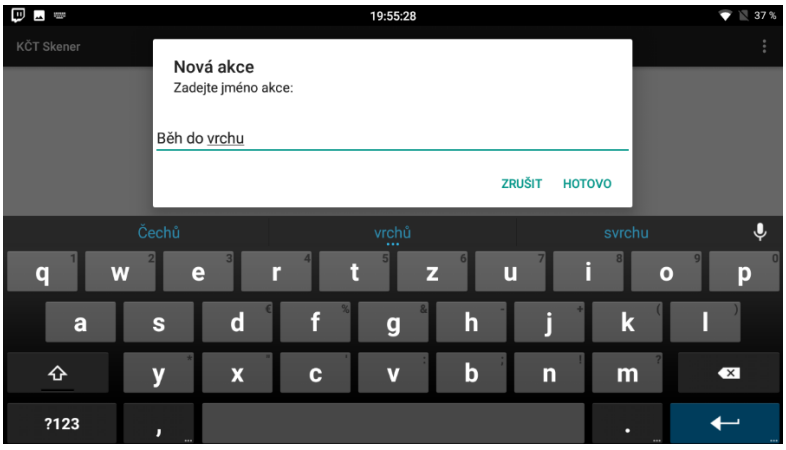
6.3.1 Rozcestník

Hlavní aktivita, která se zobrazí po spuštění, je v podstatě rozcestník. Obsahuje tlačítka pro novou akci, existující akci, nebo akci ze souboru. Při dostatečném vodorovném rozlišení (například i při otočení běžného chytrého telefonu na bok) se tlačítka přeskupí.

Obrázek 5 Aplikace - Rozcestník vertikálně, zdroj: vlastní

Obrázek 6 Aplikace - Rozcestník horizontálně, zdroj: vlastní

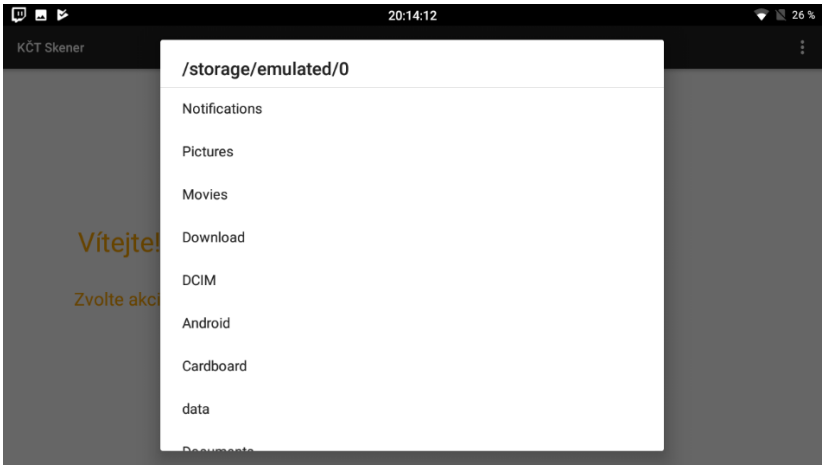
Nová akce se musí nejprve vytvořit, takže tato volba vyvolá dialog, kam uživatel zadá jméno akce a následně je spuštěna aktivita Událost.



Obrázek 7 Aplikace - Dialog nové akce, zdroj: vlastní

Existující akce nabídne již evidované akce z telefonu. Prohledá vnitřní a vnější úložiště (je-li k dispozici) a pomocí aktivity Seznam zobrazí všechny nalezené události. Po výběru konkrétní akce ji pak otevře taktéž v aktivitě Událost.

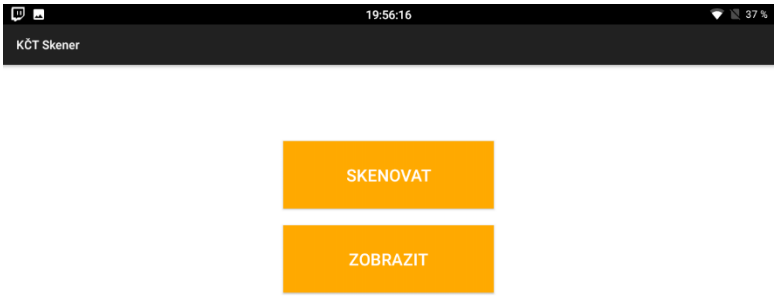
Výběrem ze souboru se otevře dialog, ve kterém uživatel prochází souborový systém. Tak lze otevřít např. stažený soubor. Tento soubor je však již jen pro čtení, jeho otevřením v aplikaci se zobrazí přímo seznam účastníků.



Obrázek 8 Aplikace - Procházení souborového systému, zdroj: vlastní

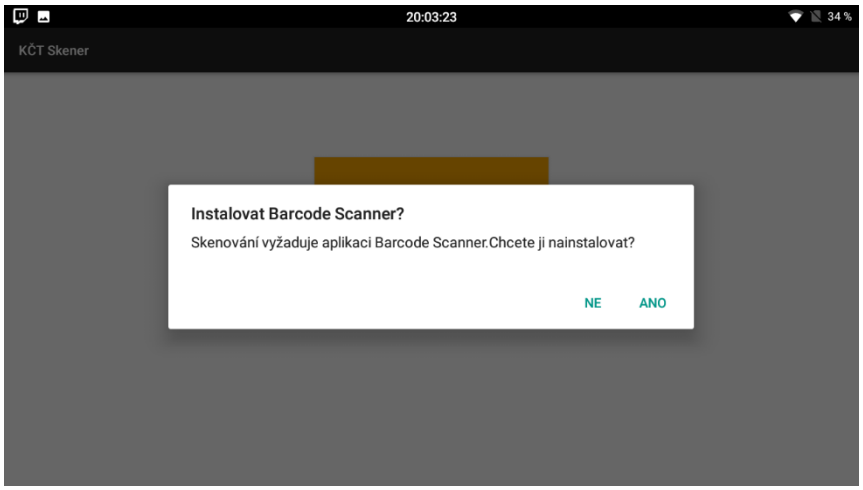
6.3.2 Událost

Událost nabízí možnost skenování a zobrazení. Tato aktivita má nastaven režim ‘landscape’, tedy je vždy ve vodorovném rozložení.

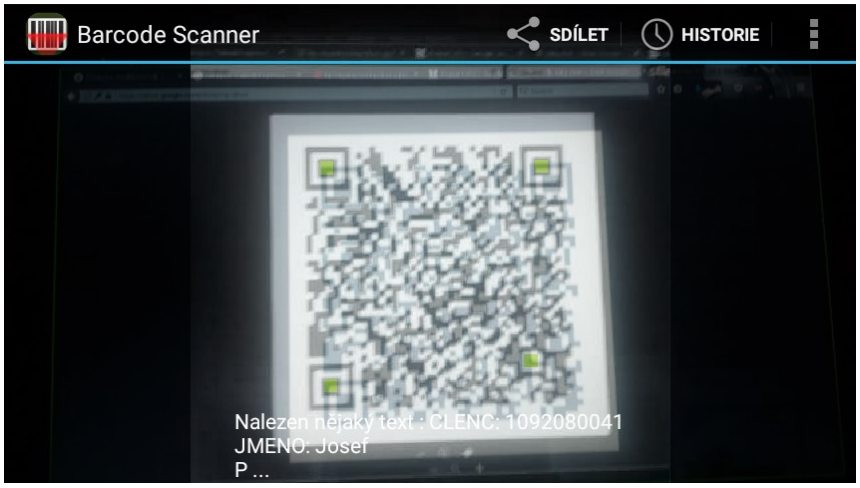


Obrázek 9 Aplikace - Událost, zdroj: vlastní

Důvodem je, že integrovaná aplikace Barcode Scanner pracuje taktéž pouze v horizontálním režimu, ale když je volána aktivitou ve vertikálním rozložení, spustí se svisle a okamžitě se otočí. Důsledkem toho se ztratí první načtený záznam. Při vývoji popisované aplikace se takový problém vyskytl také, ale o tom níže. Kliknutím na tlačítko Skenovat se otevře aplikace Barcode Scanner a je zahájeno skenování QR kódů. Ta je nastavena tak, aby skenovala neustále, dokud není ukončena stisknutím tlačítka ‘zpět’. Pokud aplikace Barcode Scanner není k dispozici, nabídne její stažení.



Obrázek 10 Aplikace - Dialog stažení čtečky, zdroj: vlastní

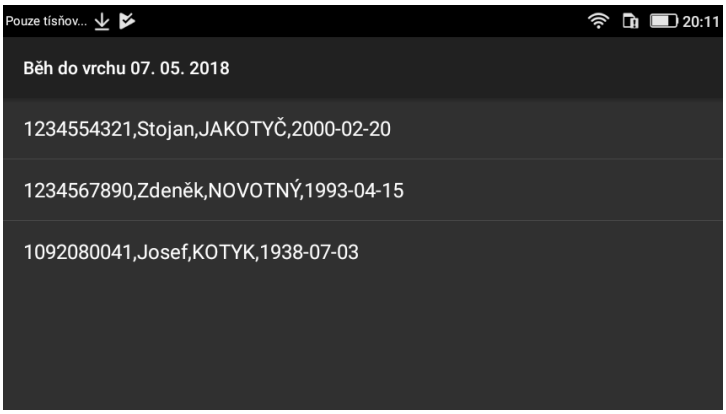


Obrázek 11 Aplikace - Barcode Scanner, zdroj: vlastní

Volbou Zobrazit je otevřena aktivita Seznam, která vypíše účastníky akce.

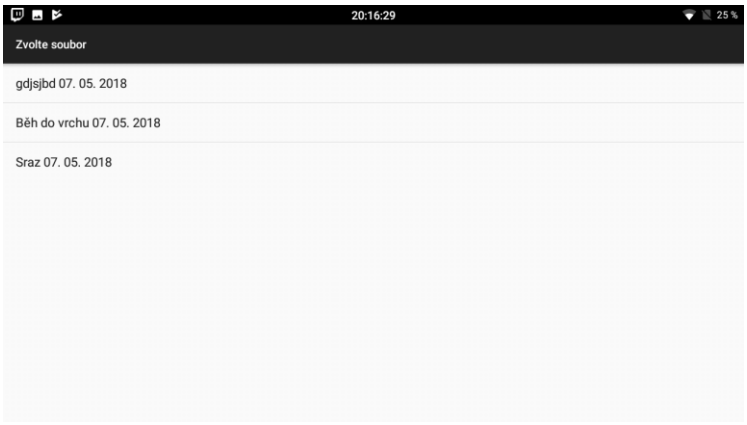
6.3.3 Seznam

Aktivita Seznam zobrazí seznam k prohlížení účastníků, nebo k vybrání Události. Android SDK k tomu nabízí rozložení ListView, které poskytnuté položky zobrazí jako seznam. Aktivita Seznam zmíněné rozložení používá a obaluje jej funkcionalitou, vhodnou pro naši aplikaci. Výsledný seznam má dva režimy. Zaprvé jde o prostý seznam, který je používán pro vypsání účastníků akce.



Obrázek 12 Aplikace - Seznam účastníků, zdroj: vlastní

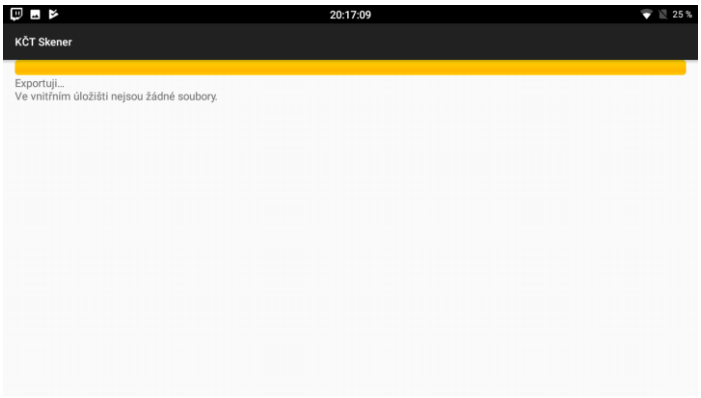
Zadruhé je k dispozici klikací seznam, který obsahuje obsluhu kliknutí pro každou položku v seznamu. Ten je uplatněn při otevírání existujících souborů. Každý soubor je reprezentován položkou svého jména a kliknutím na něj se otevře Událost pro daný soubor.



Obrázek 13 Aplikace - Seznam akcí (klikací), zdroj: vlastní

6.3.4 Exportér

Aktivita Exportér prohlédne vnitřní úložiště a pokud je k dispozici vnější úložiště a ve vnitřním jsou uloženy události, exportuje je na vnější úložiště. Proces navíc vizualizuje progress barem.



Obrázek 14 Aplikace – Exportér, zdroj: vlastní

Tato aktivita je spouštěna z menu, které nabízí také volbu ‘O aplikaci’, jež vypíše stručné informace o programu.

6.4 Obsluha

Viditelné součásti jsme si představili, ovšem z hlediska implementace je samozřejmě zajímavější to, co je běžnému uživateli skryto. Jednotlivé aktivity se musí volat a vzájemně mezi sebou komunikovat.

K čemu by například bylo celé skenování, pokud bychom naskenované údaje nemohli dále použít? A co když zařízení otočíme na bok? Pro správce aktivit to znamená aktivitu zničit a znovu spustit v odpovídajícím rozložení. Tak ale ztratíme všechna data.

To, stejně jako veškerou další funkcionalitu skrývající se za ovládacími prvky uživatelského rozhraní, ošetřují objekty jazyka Java. Proto si nyní představíme použité třídy a také některé metody z kódu.

6.4.1 Třídy

Kromě již popsaných aktivit obsahuje projekt tyto třídy:

- Dialogy – obsahuje metody pro vyvolání dialogu se zadaným textem,
- Konstanty – obsahuje konstanty pro extras intentů řetězce používané aplikací a podobně,
- IntentIntegrator – integruje Barcode Scanner, pro naše účely očesaná o nepotřebnou funkcionalitu,
- IntentResult – třída, prostřednictvím které IntentIntegrator předává naskenovaná data,
- SkenerImpl – spravuje IntentIntegrator, tedy čtečku, a zpracovává její výstup, realizuje rozhraní Skener,
- UlozisteImpl – stará se o úložiště a zápis do souborů, realizuje rozhraní Uloziste,
- VolicImpl – tvoří dialog k procházení souborového systému, realizuje rozhraní Volic,
- ListenerList – pomocná třída pro správu handlerů VolicImpl.

6.4.2 Metody

V této sekci si představíme několik metod. Jde pouze o výběr, který má ilustrovat především komunikaci mezi aktivitami.

Metoda vyber shromáždí soubory z obou úložišť a nabídne je uživateli formou seznamu. Je zde také využit Intent ke spuštění aktivity Seznam. V intentu lze spouštěné aktivitě předat v podstatě jakékoli informace pomocí takzvaných extras.

```
public void vyber(View v) throws IOException {
ArrayList<File> soubory = new ArrayList<>();
// Kontroluje zda má oprávnění.
if (ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
```

```
== PackageManager.PERMISSION_GRANTED) {
// Sesbírá soubory této aplikace z vnějšího úložiště.
soubory.addAll(getRelevantniSoubory(
new File(Environment.getExternalStoragePublicDirectory(
Environment.DIRECTORY_DOCUMENTS), konstanty.SLOZKA_APLIKACE_JMENO)));
}
       // Sesbírá soubory této aplikace z vnitřního úložiště.
soubory.addAll(getRelevantniSoubory(
new File(this.getFilesDir(), konstanty.SLOZKA_APLIKACE_JMENO)));
       ArrayList<String> jmena = new ArrayList<>();
       for (File soubor : soubory) jmena.add(zjistiNazev(soubor));
       Intent intent = new Intent(this, Seznam.class);
Intent.putExtra(konstanty.INTENT_EXTRA_NADPIS,
konstanty.NADPIS_VYCHOZI_SOUBOR);
intent.putExtra(konstanty.INTENT_EXTRA_DRUH,konstanty.ZAZNAM_SOUBOR);
intent.putExtra(konstanty.INTENT_EXTRA_SEZNAM, jmena);
intent.putExtra(konstanty.INTENT_EXTRA_SEZNAM_SOUBORU, soubory);
startActivityForResult(intent, konstanty.REQUEST_KOD_SEZNAM);
}
```

Naproti tomu překrytá metoda onActivityResult slouží ke zpracování dat, která dostaneme od spuštěné aktivity po jejím ukončení. Tato konkrétní převezme od seznamu výsledek, ve kterém je soubor k otevření a ten předá aktivitě Událost.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (requestCode == konstanty.REQUEST_KOD_SEZNAM) {
      if (resultCode == RESULT_OK) {
          final Intent intent = new Intent(this, Udalost.class);
          intent.putExtra(konstanty.INTENT_EXTRA_JE_NOVY,
konstanty.SOUBOR_STAVAJICI);
          intent.putExtra(konstanty.INTENT_EXTRA_SOUBOR,
                  (File)data.getExtras().get(konstanty.INTENT_EXTRA_SOUBOR));
          startActivity(intent);
      }
  }
}
```

Zde je překrytá metoda onCreate (volá se při tvorbě aktivity) aktivity Seznam. Tato aktivita slouží k zobrazení seznamu buď prostého, nebo klikacího.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_seznam);
       //promenna na komponentu z layoutu
ListView listView = (ListView) findViewById(R.id.lsSzn);
//aby nemizely zaznamy pri otoceni
if (savedInstanceState != null)
setIntent((Intent)savedInstanceState.getParcelable(
konstanty.BUNDLE_EXTRA_INTENT));
//titulek z intentu
setTitle(getIntent().getStringExtra(konstanty.INTENT_EXTRA_NADPIS));
//seznam u intentu
ArrayList<String> seznamZaznamu = getIntent().getExtras()
.getStringArrayList(konstanty.INTENT_EXTRA_SEZNAM);
//pokud jsou to soubory, bude seznam klikaci
```

```
if (getIntent().getIntExtra(konstanty.INTENT_EXTRA_DRUH, -1) ==
konstanty.ZAZNAM_SOUBOR) {
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int    position, long id)
{
//kliknuti pak vraci intent na onActivityResult nadrazene
                    //aktivity a v nem soubor k otevreni
                          Intent vysledek = new Intent();
                          vysledek.putExtra(konstanty.INTENT_EXTRA_SOUBOR,
                                       ((ArrayList<File>)getIntent().getExtras()
                                .get(konstanty.INTENT_EXTRA_SEZNAM_SOUBORU))
.get(position));
                           setResult(RESULT_OK, vysledek);
                                finish();
                          }
               });
        }
       //pokud jde jen o vypis, listener netřeba

       ArrayAdapter<String> adapter = new ArrayAdapter<>(this,     //adapter seznamu
             android.R.layout.simple_list_item_1, seznamZaznamu);
       listView.setAdapter(adapter);
}
```

Metoda zpracujVysledek rozebere výsledek podle přednastavených klíčových slov.

```
private String zpracujVysledek(String vysledek) {
       String zpracovanyVysledek = "";
       StringTokenizer st = new StringTokenizer(vysledek);
       String[] tokeny = konstanty.TOKENY_QR_KODU;
       String tok;
       while (st.nextToken().compareTo(tokeny[0]) != 0);
       for (int i = 1; i < 5; i++) {
             if (i > 3) {
             zpracovanyVysledek += st.nextToken();
             continue;
       }
       int j = 0;
       while ((tok = st.nextToken()).compareTo(tokeny[i]) != 0) {
             if (j++ > 0) {
                   zpracovanyVysledek += " ";
             }
             zpracovanyVysledek += tok;
       }
       zpracovanyVysledek += ",";
  }
  return zpracovanyVysledek;
}
```

Konstruktor třídy úložiště nejdříve zjistí, zda je k dispozici vnější úložiště, pak poskládá cestu k souboru z cesty k úložišti, předpony „sknr“, současného data, očesaného jména akce a přípony „.csv“. Ukazatel na soubor si uloží pro další operace.

Neexistuje-li cesta, vytvoří ji. Neexistuje-li soubor, vytvoří jej a vloží do něj jméno akce v původním znění a datum.

```
public UlozisteImpl(Activity aktivita, String jmenoAkce) {
	try {
if (Environment.MEDIA_MOUNTED.equals(Environment
.getExternalStorageState())) {
			soubor = new File(Environment.getExternalStoragePublicDirectory(
Environment.DIRECTORY_DOCUMENTS), konstanty.SLOZKA_APLIKACE_JMENO);
		} else {
			soubor = new File(aktivita.getApplicationContext()
.getFilesDir(), konstanty.SLOZKA_APLIKACE_JMENO);
		}
			soubor.mkdirs();
			soubor = new File(soubor.getPath()
.concat(vytvorJmenoSouboru(jmenoAkce)));
		if (soubor.createNewFile()) {
			zapis(jmenoAkce.replace("\n", " ").replace("\r", ""));
			zapis(getDatum());
		}
	} catch (Exception ex) {
		dialogy.vypisInfoDialog(aktivita, "Chyba",
			"Při vytváření souboru akce se vyskytla chyba.");
	}
}
```

6.5 Možná rozšíření

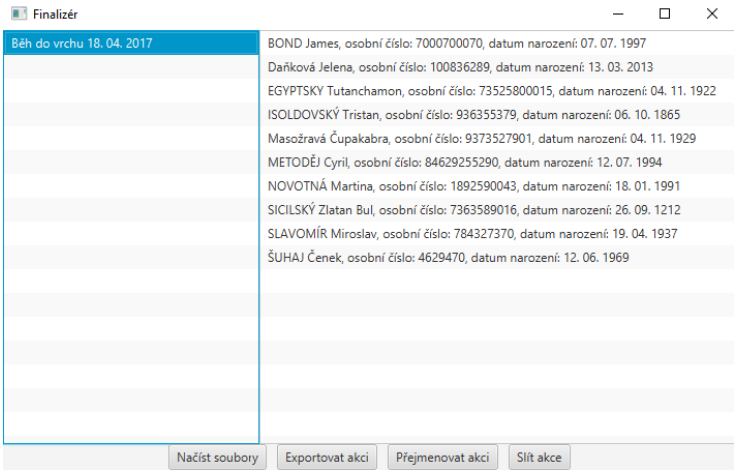
Mezi vhodnými rozšířeními aplikace rozhodně dominuje možnost komunikace s databází, tedy přímé ukládání záznamů na server. Odpadla by tak nutnost přenášet informace z mobilu na PC, slévat je a uchovávat v souborech.

Databáze by pochopitelně vyžadovala připojení k internetu, nebo alespoň k databázi v lokální síti. Na druhou stranu, kromě pohodlí přenosu po síti by také umožnila snadné spojování záznamů s dalšími informacemi v rámci databáze a jejich filtrování před zobrazením.

K tomu se váže další možné rozšíření, a sice robustnější klientská aplikace, spíše na PC, která by umožňovala například vyhledávání, filtrování dle věku účastníka nebo například statistiky návštěvnosti a aktivity členů. Možností je v tomto ohledu spousta. Nutno však podotknout, že vhodné programy k tomuto účelu již existují, a sice tabulkové procesory. Ty jsou důvodem, proč byly výstupem aplikace zvoleny soubory ve formátu CSV. Zmíněná aplikace by tak byla spíše usnadněním pro koncového uživatele, která by automatizovala relevantní filtrování a výpočty.

6.6 Aplikace pro PC

Jak už bylo předesláno ve fázi návrhu, aplikace pro PC slouží jako finalizér. Umožňuje načíst více akcí a ty pak slévat, přejmenovat a uložit jako hotový seznam. Výsledkem je opět CSV soubor. I tato aplikace je implementována v jazyce Java s využitím JavaFX.



Obrázek 15 Aplikace pro PC, zdroj: vlastní

7 ZÁVĚR

Cílem práce bylo navrhnout a implementovat aplikaci pro identifikaci účastníků konferencí Klubu českých turistů za využití členského průkazu s QR kódem a chytrého telefonu s OS Android.

Postupně byl představen QR kód, pomocí kterého jsou členové identifikováni, operační systém Android, pro který je aplikace určena a dále tři v současnosti nejpoužívanější přístupy tvorby mobilních aplikací.

Z nich byla v rámci návrhu vybrána cesta nativní aplikace, což v případě Androidu znamenalo vývojové prostředí Android Studio a programovací jazyk Java. Android studio se ukázalo být velmi kvalitním a intuitivním nástrojem.

Navržená funkcionalita byla úspěšně implementována a výsledkem je funkční aplikace pro OS Android, která umožňuje identifikaci a evidenci členů dle zadání.

Zkušební verze aplikace prošla i praktickým testem, ve kterém byla srovnávána rychlost zápisu osobního čísla ze vzorku průkazů ručně na klávesnici s rychlostí čtení aplikace. Spěšný zápis na klávesnici byl zhruba čtyřikrát pomalejší. Navíc při rychlejším psaní docházelo často k překlepům, zatímco aplikace čte přesně.

Nemluvě pak o komfortu o komfortu čtení údajů prostým ukázáním kartičky oproti dosavadní nutnosti zapisovat se ručně na papír.

Aplikace je funkční a skutečně umožňuje evidenci členů Klubu českých turistů. Díky druhé aplikaci na PC dokonce lze evidovat paralelně na více zařízeních a následně soubory slít.

Uživatel též může navázat skenováním již vytvořené akce a prohlížet si nejen lokální, ale i například stažené soubory.

Cílovou skupinou aplikace jsou pořadatelé akcí Klubu Českých Turistů, ale aplikace by s určitými modifikacemi posloužila i komukoli jinému, kdo chce evidovat návštěvníky svých akcí pomocí mobilního telefonu. Díky objektově orientovanému přístupu lze libovolnou komponentu relativně snadno nahradit.

Změnou komponenty Skener lze kupříkladu integrovat jiný čtecí engine. Aplikace by tak mohla s vhodným hardwarem číst například bezkontaktní karty.

Z této práce by mohla vycházet v podstatě jakákoli evidence mobilním zařízením.

8 POUŽITÁ LITERATURA

1 A Guide to Mobile App Development: Web vs. Native vs. Hybrid. *ClearBridge: mobile* [online]. Vaughan, 7.3.2017 [cit. 2018-05-10]. Dostupné z: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/

2 Android (operační systém). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 23. 5. 2018 [cit. 2018-05-10]. Dostupné z: https://cs.wikipedia.org/wiki/Android_(operační_systém)

3 Android Studio. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 17. 9. 2017 [cit. 2018-05-10]. Dostupné z: https://cs.wikipedia.org/wiki/Android_Studio

4 BURD, Barry A. Java programming for android developers for dummies. First edition. Indianapolis, IN: John Wiley and Sons, 2014. --For dummies. ISBN 978-1-118-61212-5.

5 QR code In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 7. 5. 2018 [cit. 2018-05-10]. Dostupné z: https://en.wikipedia.org/wiki/QR_code

6 Klub českých turistů: Vaše dobrá značka [online]. Praha, 2018 [cit. 2018-05-10]. Dostupné z: https://www.kct.cz/

7 KONEČNÝ, Matěj. Vyvíjíme pro Android: Dialogy a activity. *Zdroják: o tvorbě webových stránek a aplikací* [online]. Praha: Devel.cz Lab, 2012-, 17.8.2012 [cit. 2018-05-10]. Dostupné z: https://www.zdrojak.cz/clanky/vyvijime-pro-android-dialogy-a-activity/

8 MEDNIEKS, Zigurd R. Programming Android. 2nd ed. Beijing: O'Reilly, c2012. ISBN 978-1-4493-1664-8.

9 Meet Android Studio. *Android Developers* [online]. 2007- [cit. 2018-05-10]. Dostupné z: https://developer.android.com/studio/intro/

10 MULLIS, Alex. Android Studio tutorial for beginners. *Android Authority* [online]. 2018, 11.11.2017 [cit. 2018-05-10]. Dostupné z: https://www.androidauthority.com/android-studio-tutorial-beginners-637572/

11 OWEN, Sean. ZXing ("Zebra Crossing") barcode scanning library for Java, Android. *GitHub: The world's leading software development platform* [online]. San Francisco, 2007-, 24.4.2018 [cit. 2018-05-10]. Dostupné z: https://github.com/zxing/zxing

12 Platform Architecture. *Android Developers* [online]. 2007- [cit. 2018-05-10]. Dostupné z: https://developer.android.com/guide/platform/

13 POLÁŠEK, Roman. QR kódy - na co jsou, jak je vytvářet, číst a používat. *Stahuj.cz: Magazín* [online]. Praha: Economia, 2018, 19.1.11 [cit. 2018-05-10]. Dostupné z: http://magazin.stahuj.centrum.cz/qr-kody-na-co-jsou-jak-je-vytvaret-cist-a-pouzivat/

14 SEMECKÝ, Vojtěch. Android Studio – nové vývojové prostředí. *Zdroják: o tvorbě webových stránek a aplikací* [online]. Praha: Devel.cz Lab, 2012-, 4.11.2013 [cit. 2018-05-10]. Dostupné z: https://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/

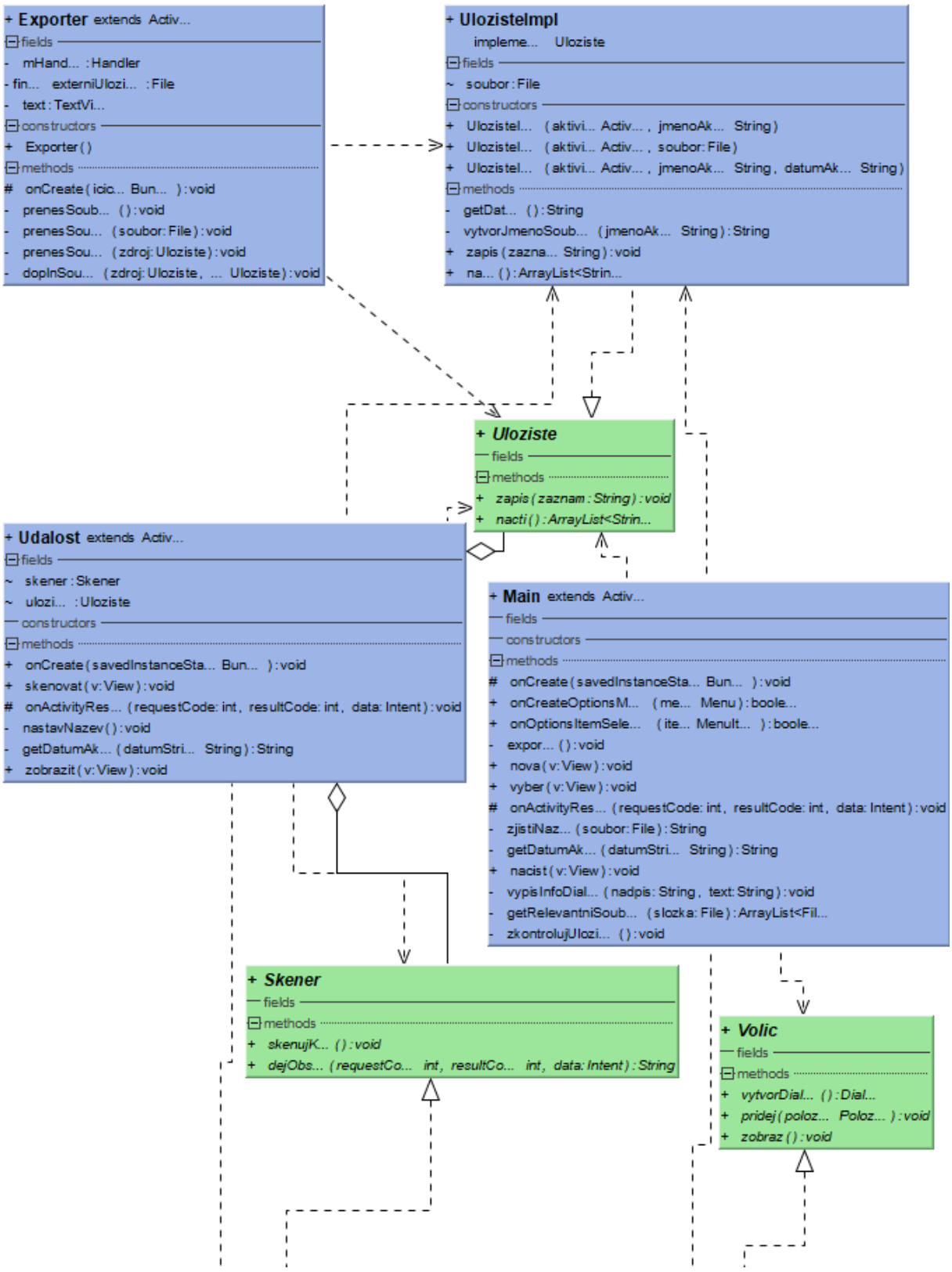
15 Save files on device storage. *Android Developers* [online]. 2007- [cit. 2018-05-10]. Dostupné z: https://developer.android.com/training/data-storage/files

16 *Stack Overflow: Where Developers Learn, Share, & Build Carreers* [online]. 2018 [cit. 2018-5-10]. Dostupné z: https://stackoverflow.com/

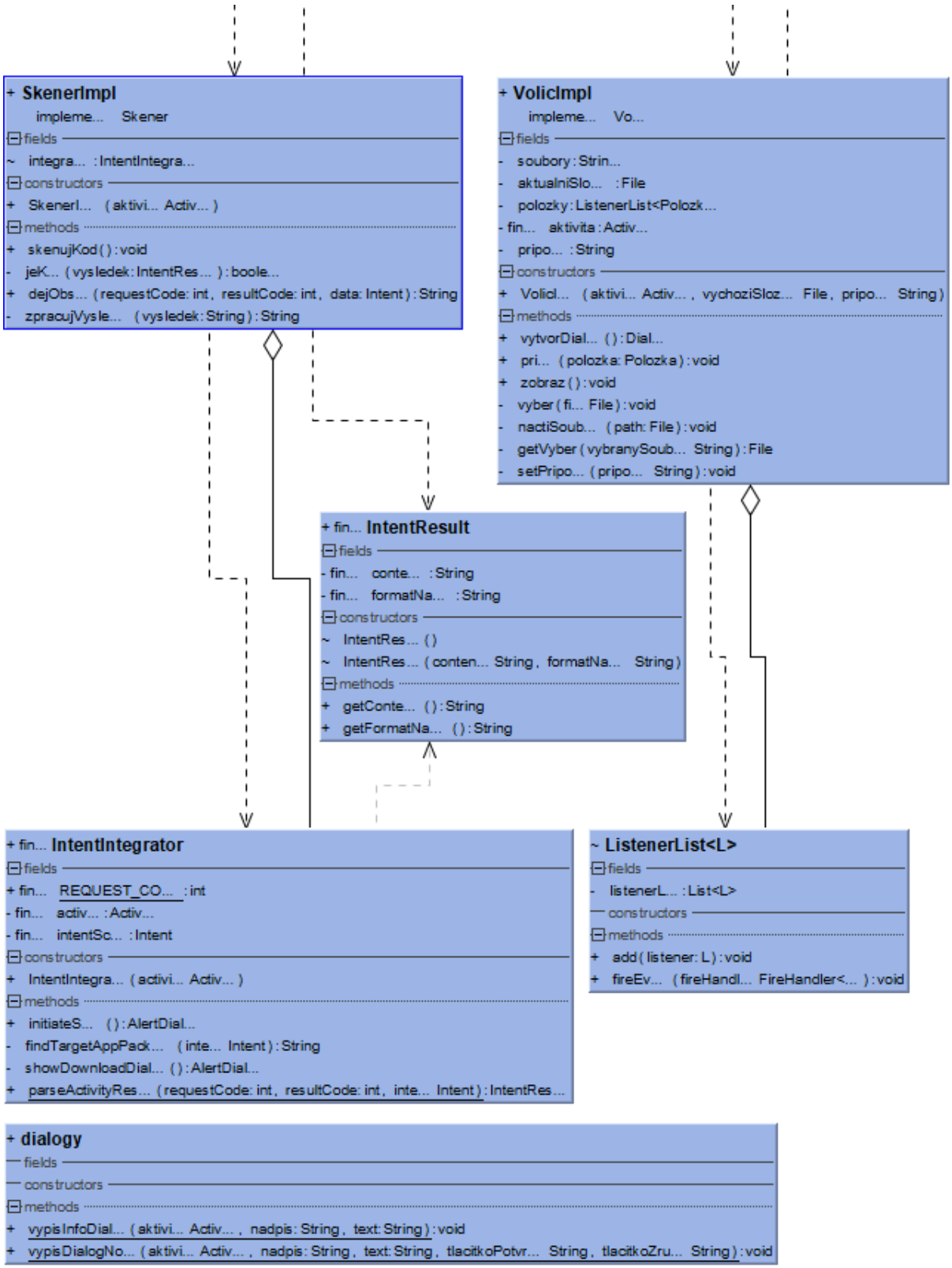
17 VÁCLAVÍK, Jan. Jak vyvíjet mobilní aplikace. *Jan Václavík* [online]. 2018, 4.6.2015 [cit. 2018-05-10]. Dostupné z: http://janvaclavik.cz/jak-vyvijet-mobilni-aplikace

9 PŘÍLOHY

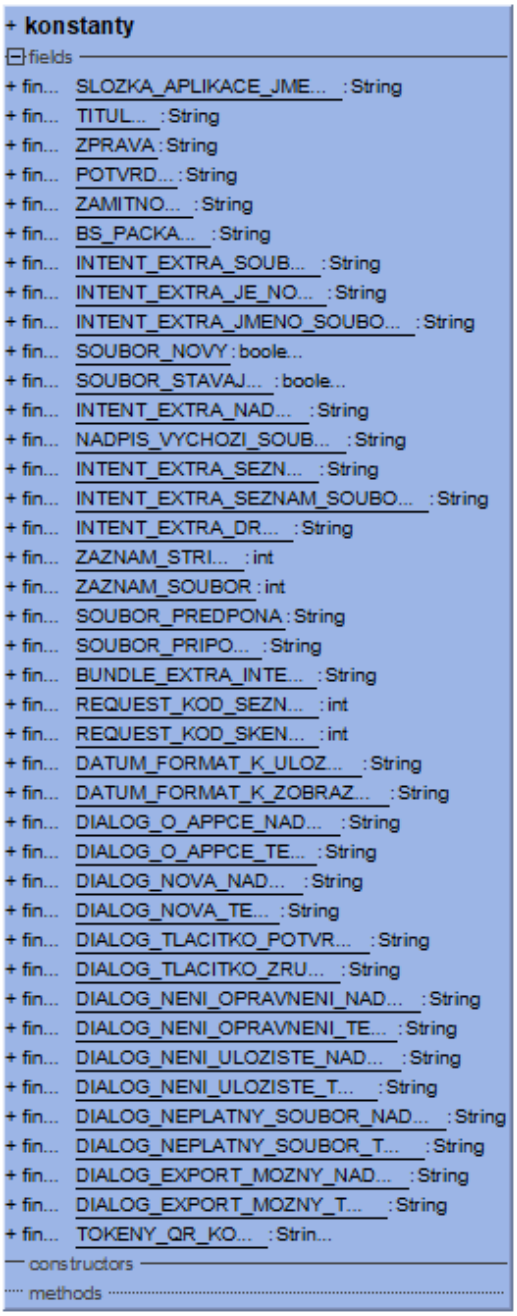
Příloha A – UML diagram tříd mobilní aplikace

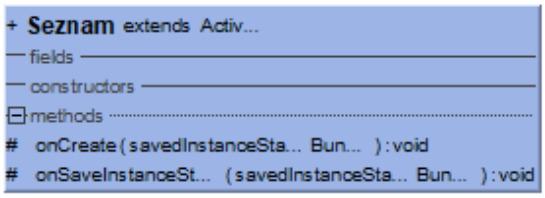


Příloha A – UML diagram tříd mobilní aplikace (pokračování)



Příloha A – UML diagram tříd mobilní aplikace(pokračování)





Příloha B – UML diagram tříd aplikace pro PC