ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA EKONOMICKÁ

Bakalářská práce

Vývoj aplikace pro mobilní telefony

Mobile application development

Zdeněk Borský

Plzeň 2017

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 panu RNDr. Mikuláši Gangurovi, Ph.D. za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat.

Obsah

Úvod

Mobilní telefony se staly nejrychlejším komunikačním prostředkem dnešní doby. S nástupem chytrých mobilních telefonů mohou lidé nejen využívat základní funkce telefonů jako je volání a posílání zpráv, ale také nově rozšířené, jako je využívání internetu, GPS, rozpoznávání majitele telefonu na základě tváře, počasí, videohovory, focení, hry. Díky těmto a dalším výhodám pro každodenní život vlastní mobilní telefon většina populace vyspělých zemí [1]. Z tohoto důvodu je řešení následující problematiky realizováno právě pro mobilní telefony.

Při nálezu určité věci platí, že dle právního hlediska je nálezce povinen nalezenou věc předat obci, na jejímž území byla tato věc nalezena. Český zákon nezná přisvojení jako obecný nabývací způsob. Vlastník má poté možnost se o příslušnou věc přihlásit do 1 roku [2]. Realita ovšem většinou bývá zcela jiná. Ve většině případech lidé, kteří naleznou nějakou věc, využijí k jejímu předání místo nebo osobu spojenou s místem nálezu. Pokud například najdou peněženku v autobuse, odevzdají ji řidiči, na centrálu dopravních podniků, popřípadě na policii. Málokdo chce totiž věnovat svůj čas na cestu na obecní či městský úřad. Nálezce může věc odevzdat i v jiné městské části než ji našel. O tento fakt se opírá aplikace, která bude vytvořena v rámci této bakalářské práce. Aplikace má za cíl usnadnit dohledání ztracené věci a ušetřit tak čas jak nálezci, tak osobě, která danou věc ztratila.

- **Hlavní cíl práce:**
  - Hlavním cílem této práce je vytvoření funkční aplikace pro mobilní zařízení na zvolené platformě, která bude řešit problematiku ztrát a nálezů
- **Vedlejší cíle práce:**
  - Stanovení funkcionalit a využití mobilní aplikace
  - Analýza používaných operačních systémů a výběr OS spolu s vývojovým prostředím pro vybraný OS
  - Návrh loga aplikace a layoutu
  - Návrh datových objektů a vazeb mezi nimi
  - Vývoj API

- Implementace navržené aplikace
- Uživatelské testy aplikace, návrhy na změny, zhodnocení využití

V první kapitole je provedena analýza daného problému, který má mobilní aplikace řešit. Zároveň zde nalezneme návrh funkcionalit mobilní aplikace. Druhá kapitola se zabývá analýzou různých mobilních platforem a jejich výhod či nevýhod. Třetí kapitola se soustředí na výběr operačního systému, vývojového prostředí a seznámení s tímto prostředím. Čtvrtá kapitola obsahuje návrh designu loga, návrh layoutu, návrh relačního modelu a návrh UML diagramu. V páté kapitole je rozebrán samotný vývoj mobilní aplikace a API, to znamená použité komponenty, vazby, prvky a architektura. Šestá kapitola se soustředí na stanovení návrhů na změny, popisuje průběh testování, pilotáž mobilní aplikace a zhodnocení výsledků.

1 Návrh funkcionality a využití mobilní aplikace

1.1 Průzkum trhu

Po prozkoumání výsledků z různých prohlížečů není autorovi této práce známo, že by existovala aplikace, která by bez omezení na místo poskytovala službu ztrát a nálezů. Existují spousty webových aplikací, které jsou obvykle zaměřeny pouze na jedno město, vesnici či objekt.

- Ztráty a nálezy města Plzeň: http://umo3.plzen.eu/urad-a-samosprava/ztraty-a-nalezy/
- Ztráty a nálezy v Praze: http://www.praha.eu/jnp/cz/o_meste/zivot_v_praze/sluzby/ztraty_a_nalezy/
- Ztráty a nálezy města Liberec: http://www.liberec.cz/ztraty-nalezy/
- Ztráty a nálezy autobusů FlixBus: https://www.flixbus.cz/servis/ztraty-a-nalezy

Další velmi častou možností jsou webové aplikace na principu ztrát a nálezů, které poskytují různé instituce, například poskytovatelé autobusové dopravy nebo budovy jako divadla, stadiony a kina. Aplikace nemusí sloužit pouze pro civilní osoby, ale při jejím potencionálním rozšíření ji budou moct využívat i státní složky, například PČR, obec či město. Tyto instituce jsou povinny dle nového občanského zákoníku, předpisu č. 89/2012 § 1057

nalezenou věc uschovat po dobu 1 roku. Po uplynutí 1 roku mohou tyto instituce s věcí manipulovat jako poctiví držitelé. Pokud se ovšem právoplatný majitel nalezené věci najde i v době až 3 let od nalezení, musí mu instituce věc vrátit nebo mu vrátit peněžní kompenzaci, kterou získala v dražbě nalezené věci [2].

1.2 Využití aplikace

Hlavní využití aplikace bude spočívat v šetření času lidem, kteří něco ztratili. Tito lidé by nemuseli složitě hledat, kde byla jejich věc po nalezení zanechána. Je dáno zákonem, že nálezce je povinen nalezenou věc odevzdat na příslušný úřad, nebo provozovateli veřejné budovy či veřejného prostředku, dle nového občanského zákoníku, předpisu č. 89/2012 § 1052 [2]. Zpravidla tomu tak není a tyto věci končí na místech, které jak již bylo řečeno v úvodu, souvisí s místem nálezu, ale nemusí to být místa, která udává nový občanský zákoník. Z tohoto důvodu je obvykle obtížné ztracenou věc dohledat, i když byla nalezena a poctivě odevzdána. Někteří lidé se mnohdy smíří se ztrátou, než aby absolvovali hledání místa, kde byla věc odevzdána. Aplikace s názvem „Ztrátoše“ bude tedy sloužit pro snadnější nalezení místa, kde byla ztracená a znovu nalezená věc odevzdána. Vydání této věci bude záležet čistě na lidském faktoru.

1.3 Návrh aplikace a funkcionalit

Přihlášení do aplikace nebude nutné, ale bude umožněno. Toto přihlášení by dle očekávání mělo sloužit převážně pro státní složky, veřejné objekty a jiné organizace. Bude ovšem možné i pro civilní osoby. Přihlášení plní účel při vyhledávání a vkládání. Pokud uživatel vloží příspěvek po přihlášení, bude poté jeho jméno uvedeno i ve vyhledávání. Předání věci osobě, která věc ztratila, bude záležet vždy na osobě, které byla tato věc předaná jako nalezená. Aplikace bude sloužit jako jednotný portál pro nalezené věci.

1.3.1 Seznam funkcionalit aplikace

- **Přihlášení do systému** – Slouží jako další informativní prvek při vyhledávání.
- **Registrace** – Registrace slouží pro následné přihlášení.
- **Ukládání informací o nalezené věci do databáze** - Funkce bude ukládat informace od uživatelů do databáze.

- **Filtrování nalezených věcí dle specifikací uživatele** – Hlavním rozdílem od všech ostatních aplikací (převážně webových) bude možnost tuto aplikaci používat v rámci celé České republiky. Proto bude možné filtrovat si informace pouze pro určitý kraj, okres a kategorii.

1.3.2 Uživatelské rozhraní aplikace

Uživatel bude mít možnost seznámit se s pravidly využívání aplikace a s platným zákonem České republiky, který se zaobírá nálezem či ztrátou a to zákon č. 89/2012 Sb., občanský zákoník.

Například tedy:

- č. 89/2012 Sb., občanský zákoník § 1051.
  „Má se za to, že si každý chce podržet své vlastnictví a že nalezená věc není opuštěná. Kdo věc najde, nesmí ji bez dalšího považovat za opuštěnou a přivlastnit si ji“ [2].
- č. 89/2012 Sb., občanský zákoník § 1052.
  „(1) *Ztracenou věc vrátí nálezce tomu, kdo ji ztratil, nebo vlastníkovi proti úhradě nutných nákladů a nálezného* [2].
  (2) *Nelze-li z okolností poznat, komu má být věc vrácena, a nepovažuje-li se věc za opuštěnou, oznámí nálezce bez zbytečného odkladu nález obci, na jejímž území byla nalezena, zpravidla do tří dnů; byla-li však věc nalezena ve veřejné budově nebo ve veřejném dopravním prostředku, odevzdá nálezce nález provozovateli těchto zařízení, který se zachová podle jiných právních předpisů, a není-li jich, postupuje, jako by byl nálezcem“* [2].
- Uživatel bude mít možnost přihlášení.
- Uživatel, který bude postrádat určitou věc, bude mít možnost si pomocí aplikace zobrazit věci nalezené v určitém kraji a okresu pro určitou kategorii.
- Každá věc uložená do databáze ponese informaci, kde byla nalezena, kam byla odevzdána, o jakou věc se jedná, jakého výrobce daná věc je, kdy byl vložen záznam a kým, pokud byl uživatel při vkládání přihlášen.

2 Analýza používaných operačních systémů a vývojových prostředí

V dnešní době existuje spousta operačních systémů pro mobilní telefony, například iOS, Android, Windows Phone. Mezi nejvíce využívané patří právě zmíněné platformy [3].

2.1 Windows Phone

Windows Phone je platforma, která spadá pod společnost Windows. Je nástupce předchozí platformy Windows Mobile, se kterou ale není zpětně kompatibilní. Počátky Windows Phone se datují ke dni 21. října 2010. V dnešní době se používají především systémy Windows Phone 7 a 8, které však nejsou navzájem kompatibilní a dokonce obsahují interně různé platformy Windows CE a Windows NT [4].

Úspěch Windows Phone 8 je postaven na dostupnosti kvalitních aplikací za přiměřenou cenu, s vyhovujícím prodejním a licenčním modelem. Od 18. 10. 2013 je k dispozici zdarma upgrade na verzi 8.1 [4].

Nejnovější platformou je Windows 10 Mobile. Windows prohlásil, že toto je poslední pokus o „restart“. Windows 10 Mobile má snahu o propojení veškerých zařízení běžících na této platformě, ale také propojení s osobními počítači a dalšími zařízeními běžícími na operačním systému Windows. Společnost se rozhodla, že chce dělat univerzální aplikace. Bohužel dle posledních průzkumů a recenzí jsou tyto aplikace mnohem pomalejší, a to hlavně z důvodu změny logiky v operačním systému. Společnost se rozhodla vynechat veškeré huby a centra, která byla určena pro konkrétní druh obsahu. Místo toho jsou veškeré části systému samostatnou aplikací, což znamená aplikaci pro hudbu, hry, rádio atd. To sice umožňuje Microsoftu pružně reagovat na případné chyby a vývoj dílčích vylepšení, ale zároveň to zpomaluje danou platformu [6].

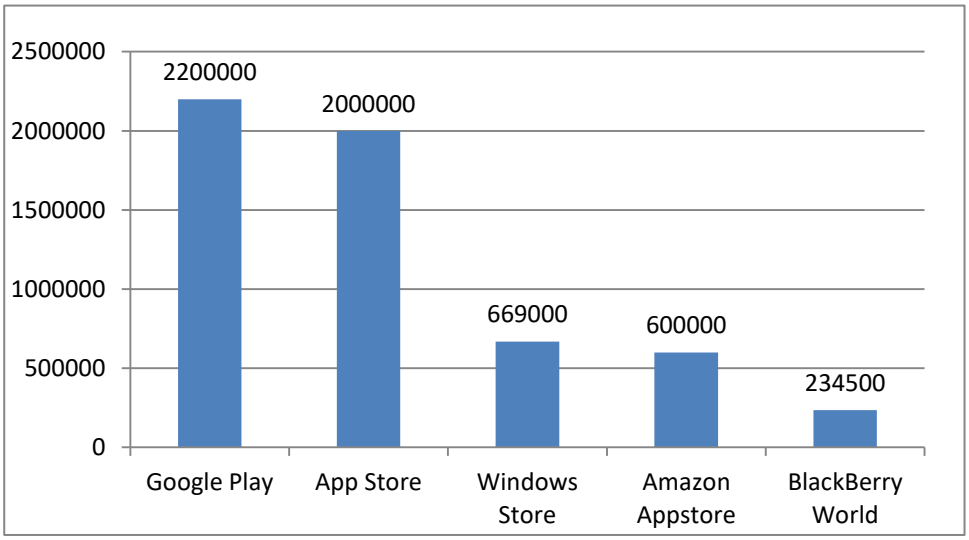
2.1.1 Vývoj pro Windows Phone

Vývoj probíhá ve vývojovém prostředí Microsoft Visual Studio 2013 a návrhy designu se realizují pomocí Blend for Visual Studio 2013. Toto vývojové prostředí je možno získat v komerční placené verzi ale i v „odlehčené“ verzi

Express pro studenty nebo hobby vývojáře. Tato verze je zdarma. K vývoji lze ovšem použít i jazyky XAML na návrh prezentačního rozhraní v kombinaci s C#, VB.NET a C++. Dále lze využít i HTML5 pro návrh prezentačního rozhraní a programovací jazyk JavaScript pro aplikační logiku, což otevírá prostor pro webové vývojáře [4].

Poslední možností je Project Siena. Tato aplikace, která je volně ke stažení, umožňuje lidem bez předchozích znalostí v programování intuitivně vytvářet jednodušší aplikace, například pro nabídku služeb, prezentování blogových aktualizací a podobně. Project Siena by mohl pomoci vyřešit obtíže, se kterými se Microsoft potýká. Tyto obtíže spočívají v menším počtu aplikací oproti hlavní konkurenci [9][7]. *Sloupcový Graf 1.*

Sloupcový graf 1 Počet aplikací dle platforem



Zdroj: *[7] Zpracování s využitím Microsoft Excel 2010*

Lze tedy říci, že vývoj aplikace pro Windows Phone neobnáší žádnou počáteční investici.

2.1.2 Publikování aplikace pro Windows Phone

Windows Store je oficiálním zdrojem pro aplikace na danou platformu. Umožňuje distribuovat jak aplikace placené, tak aplikace, které jsou zdarma. U placených aplikací je možnost takzvané zkušební doby, která se zadává přímo na Windows Store a umožňuje používání aplikace pouze na určitou dobu bez

zakoupení. Je možné kromě omezení doby nastavit i omezení funkcionality, například otevření jen pár úrovní pokud se jedná o hru.

2.1.2.1 Podmínky pro průchod řízením schválení aplikace

- Aplikace musí přinášet přidanou hodnotu zákazníkům
- Aplikace může mít reklamy, ale musí poskytovat hlavně funkcionality než zobrazování reklam
- Aplikace se musí chovat předvídatelným způsobem
- Aplikace musí být řízena uživatelem
- Aplikace musí být vhodná ke globálnímu nasazení
- Aplikace musí být snadno rozpoznatelná a pochopitelná [4]

Pokud splní aplikace základní podmínky pro její publikování, musí si vývojář zakoupit účet a to buď individuální účet v ceně $19USD nebo firemní účet v ceně $99 USD [8]. Poté musí vývojář zvolit a zároveň ověřit název pro aplikaci, ať již ověřuje duplicitu názvu nebo korektnost názvu. V druhém kroku je zapotřebí zvolit prodejní model aplikace a nastavit případné atributy při zvolení placené aplikace, jako cenu, region pro distribuci a podobné. Je potřeba zvolit i věkové omezení aplikace. Kontrolu všech potřebných nastavení lze provádět pomocí programu App Certification Kit.

Společnostem nadále Microsoft nabízí možnost takzvaného sideloadingu, tedy aplikace vytvořené pro firmu, která se nebude volně prodávat a bude dostupná pouze zaměstnancům. Tyto aplikace nejsou vystaveny na Windows Store, ale je zapotřebí, aby byly také ověřeny pomocí Windows App Certification Kitu a byly kryptograficky podepsané. Toto umožňuje šířit firemní aplikaci pouze v rámci firemních zařízení a udržovat ji neustále aktualizovanou na daných zařízeních [4].

2.1.3 Shrnutí a výhody / nevýhody

Microsoft vyvíjí snahu o neustálé zlepšování a komplexní propojení svých zařízení, ať už pro usnadnění práce uživatelům nebo vývojářům. Bohužel to Microsoft stálo již pár nekompromisních „restartů“ ve smyslu vývoje zcela nových platforem. Neustálé změny bezpochyby odradily spousty zákazníků. Samotný vývoj se ovšem může realizovat hned několika způsoby a to i velice uživatelsky přívětivou cestou pomocí projektu Siena, ke stažení:

https://www.microsoft.com/cs-cz/store/p/microsoft-project-siena/9wzdncrfj3pp

Microsoft ovšem stále čeká velmi dlouhá cesta k dosažení svého cíle s novou platformou Windows 10 Mobile.

2.1.3.1 Výhody – uživatel

- Zkušební doby na Windows Store pro placené aplikace
- Známé prostředí
- Jednoduchost
- Kontrola aplikace před publikováním na Windows Store
- Možnost vývoje vlastní aplikace pomocí projektu Siena i pro uživatele, kteří nedisponují znalostmi programovacího jazyka
- Firmy mohou využít sideloading

2.1.3.2 Nevýhody – uživatel

- Nejistota, historicky několik razantních změn platforem
- Menší počet dostupných aplikací
- V rámci mobilních zařízení málo rozšířená platforma

2.1.3.3 Výhody – vývojář

- Nízké poplatky pro zveřejnění
- Možnost volby několika programovacích jazyků
- Existence nástroje Windows App Certification Kit
- Málo aplikací na trhu, tedy menší konkurence

2.1.3.4 Nevýhody – vývojář

- Málo rozšířená platforma, tedy málo potencionálních uživatelů
- Existence projektu Siena, tedy přiliv nové konkurence pro stávající vývojáře z řad běžných uživatelů
- Nejistota, historicky několik razantních změn platforem
- Nekompatibilnost v rámci starších platforem

2.2 Android

Android je další z řady operačních systémů pro mobilní telefony, založený na jádře Linuxu.

Tento operační systém vyvíjí uskupení výrobců mobilních telefonů a to Open Handset Alliance. Historie Androidu sahá do roku 2003, kdy jej vlastnil Android Inc. V roce 2005 Android odkoupila společnost Google, která také jako první v roce 2007 vydala nekomerční verzi systému Android. Komerční verze Android 1.0 vyšla až v roce 2008. Nejnovější verzí je nyní Android 7.0 Nougat.[10], [11], [12].

Nejpoužívanější verzí je ovšem verze Lollipop z roku 2014, kterou používá 36,1% zařízení a verze Marshmallow z roku 2015, kterou používá 34.3% zařízení [13].

Android je nejpoužívanějším operačním systémem dnešní doby pro mobilní telefony [1]. Tento systém se využívá jak v mobilních telefonech, tak v hodinkách, televizích, autech a podobně [16].

2.2.1 Vývoj pro Android

Operační systém Android se snaží být velmi otevřený pro vývojáře z řad uživatelů i profesionálů. Existuje spousta softwarů pro vývoj aplikací právě pro Android. Kromě oficiální verze vývojového prostředí, která bude zmíněna později, existuje spousta specializovaných možností jako:

- SalesForce1, specializovaný software pro zlepšení prodejů.
- Canvas zaměřený především na zákazníky, kteří jsou neustále v pohybu a potřebují stále vyplňovat papíry.
- Jscrambler, který je kombinací JavaScriptu a html5.
- Zengine, specializovaný na technické využití.
- Cordova, která se specializuje na využití JavaScriptu, Html5 a CSS pro tvorbu aplikací na různá zařízení. [14]

Oficiálním vývojovým prostředím ale stále zůstává Android Studio, které je volně dostupné z oficiálních stránek společnosti Android. Syntaxe jazyka využívaná v prostředí Android Studia vychází ze syntaxe Javy. Android myslí i na uživatele, kteří chtějí programovat aplikace pro tuto platformu, ale nevlastní

fyzicky žádné zařízení. Umožňuje otestovat aplikace na virtuálním mobilním zařízení v rámci Android Studio [17].

2.2.2 Publikování aplikace pro Android

Publikování na Google Play Store vyžaduje stejně jako u konkurenčních platforem několik kroků. Prvním krokem k úspěšnému publikování aplikace je registrace vývojářského účtu. Vytvoření tohoto účtu je zpoplatněno $25USD. Dalším krokem, pokud má být aplikace zpoplatněna, je nastavení platebního účtu u Google. Pokud budou tyto kroky hotovy, může se uživatel přihlásit do vývojářské konzole, což je místo, kde je spravována aplikace na Google Store. Tato testovací konzole nabízí hned několik služeb [17].

2.2.2.1 Služby poskytované vývojářskou konzolí před zveřejněním aplikace

- **Cloud Test Lab** – Automaticky otestuje aplikaci na většině zařízeních od různých výrobců. Napomáhá ověřit kompatibilitu před dalšími kroky.
- **Alpha / Beta test** – Před samotným zveřejněním aplikace je možnost vydat beta verzi k otestování, a to buď vybrané skupině uživatelů, nebo uživatelům na Google Play. Pokud zvolíme neveřejné testování, testeři nebudou schopni vkládat veřejné recenze k testované aplikaci. Proto je zapotřebí sdělit testerům jakým způsobem mají provádět zpětnou vazbu.
- **Promote App** – Možnost sledování počtu instalací aplikace, ale taky kolikrát byla otevřena v Google Play a Google Store. Tato služba ovšem nabízí hlavně možnost měnit námi poskytnuté screenshoty z aplikace, text u aplikace, barvy a další. Díky tomu můžeme sledovat jaký text, barva, popis má největší vliv na uživatele a postupně nalézt nejefektivnější kombinaci k přilákání dalších uživatelů.
- **Statistika** – Možnost využít různé statistické informace z používání aplikace. Tyto statistiky nabízí přehled o nejčastějším času využívání aplikace, o peněžních transakcích v rámci aplikace, o chování uživatelů, pádů aplikace a změnách nastavení aplikace od uživatele.
- **Správa ceny** – Samozřejmostí je i správa ceny aplikace, možnost předplatného nebo nastavení ceny pro různé země.

2.2.3 Další možnosti distribuce u Androidu

Kromě publikování aplikace pro mobilní zařízení máme možnost publikovat aplikaci i na další zařízení, která fungují na platformě Android a to Android TV, Android Wear, Android Auto a Cardboard VR. Dále nám Google Store umožňuje publikovat aplikaci ve speciálních okruzích. Třemi největšími okruhy jsou okruh pro rodinu, pro práci a pro vzdělávání.

2.2.3.1 Android TV, Wear, Auto, VR

Programování pro další zařízení, která fungují na platformě Android, podléhá kromě základního testování a schvalování i nadřazenému testování. Pokaždé je zapotřebí souhlasit se specifickými podmínkami ze strany Androidu a Google Play. Je nutné využít specifické knihovny. Je povinnost otestovat funkčnost této aplikace pro zvolený produkt. V tomto směru je nám nabídnuto testování na virtuálním produktu. Společnost Android i Google v tomto směru vychází programátorům velmi vstříc a nabízí podrobný popis jak zprovoznit aplikaci na těchto netradičních zařízeních.

2.2.3.2 Okruhy v Google play

Při zveřejnění aplikace je možnost přiřadit ji do určitého okruhu v Google Play Store. Tyto okruhy jsou.

- **Families** – Pokud je aplikace pro děti, přiřazením této aplikace do okruhu rodina získáme větší propagaci aplikace k rodičům. Pokud budou rodiče hledat obsah na Google Play, který je „family-friendly“, bude se mu vložená aplikace v tomto okruhu zobrazovat přednostně. Dále rodiče, kteří byli spokojeni s aplikací, ji mohou ohodnotit a poté se u této aplikace bude zobrazovat hvězdička, jako doporučená aplikace ostatními rodiči [15].
- **Work** – Tato možnost je zde především pro společnosti. Nabízí společnosti možnost organizovat, které aplikace budou dostupné pro jejich zaměstnance a které aplikace jim z Google Play dostupné v rámci firmy nebudou [15].
- **Education** – V sekci vzdělávání se nachází aplikace vytvořené pro učitele. Tyto aplikace mohou být využívány ve školách a umožňují například učitelům rozeslat zadání do všech ostatních zařízení v učebně.

Jsou tedy schopni studentům zobrazit to, co je zapotřebí v jejich zařízeních [15].

2.2.4 Shrnutí a výhody / nevýhody

Operační systém Android je nejvíce využívaný systém pro mobilní telefony [1]. Android umožňuje svým uživatelům stahovat veliké množství aplikací [7] a to hlavně díky jeho politice v tomto směru. Nepokládá před vývojáře zbytečné překážky. Naopak, Android se snaží vytvořit prostředí, ve kterém se bude každý vývojář cítit dobře a bude vědět, že nemá žádné překážky ve své tvorbě.

2.2.4.1 Výhody – uživatel

- Velké množství aplikací
- Většina aplikací bezplatná
- Intuitivní ovládání

2.2.4.2 Nevýhody – uživatel

- Nekompatibilita mezi verzemi
- Možnost výskytu nekvalitních aplikací

2.2.4.3 Výhody – vývojář

- Otevřenost systému pro vývoj
- Kvalitní vývojářská podpora
- Přehledné vývojové prostředí
- Zpřístupněn vývoj pro Android TV, Wear, Auto, VR

2.2.4.4 Nevýhody – vývojář

- Existence velkého množství vývojářů, tedy konkurence pokud se tím chce vývojář živit
- S velkým množstvím vývojářů přichází i veliké množství aplikací, tedy je složitější dostat se do povědomí uživatelů

2.3 Apple IOS

Operační systém IOS je vyvíjen společností Apple Inc. První operační systém pod touto značkou byl vypuštěn v roce 2007. Od tohoto roku Apple vydal již 9 verzí své mobilní platformy. Poslední verze IOS 9 byla vypuštěna v roce 2015 a svého posledního vylepšení se dočkala v srpnu 2016 [18]. Z uživatelského hlediska a z osobní dlouholeté zkušenosti s produkty Apple je možné konstatovat následující fakta.

Vývoj pro zařízení Apple je oproti ostatním platformám velmi uzavřený. Naproti tomu z uživatelského hlediska je IOS velmi nestranný. U konkurence máme velmi často pouze jednu možnost například pro výběr softwaru, kterým budeme číst emaily, prohlížet webové stránky nebo například upravovat textové dokumenty. Toto je na platformě IOS naprosto nepodstatné, jelikož si zde můžeme zvolit, který software budeme používat. Například balíček Microsoft Office jako Word, Excel, Outlook a další programy společnosti Windows běží na této platformě někdy lépe jak na samotném operačním systému Windows.

2.3.1 Vývoj pro IOS

Vývoj pro IOS probíhá pomocí programovacího jazyka Swift, který vzešel z dříve používaného balíčku frameworku Cocoa (objective C). Samotný Swift byl vypuštěn v roce 2014 a je určen výhradně pro vývoj na platformách Mac OS X. Z programového hlediska je jazyk Swift obdobou objective-C, pouze má upravené syntaxe a celkově využívá modernější zápis. Swift stále umí spolupracovat s jazykem Cocoa i Cocoa Touch [19]. Jak již bylo zmíněno, je zapotřebí vlastnit zařízení Mac, aby bylo možné programovat aplikace pro IOS. Existuje několik způsobů, jak toto vcelku velké omezení obejít. Jedním ze způsobů je instalace virtuálního počítače. Na virtuální počítač je zapotřebí nainstalovat OS pro Mac. Tento software je možno získat ze stránek společnosti Apple: https://support.apple.com/downloads/macos a to i v nejnovější verzi MacOS Sierra 10.12.2. Bude potřeba nainstalovat i vývojové prostředí pro IOS s názvem Xcode.

Další možností je obstarat si službu podobnou službě MacinCloud, která nabízí pronajmutí vlastního cloudového přístroje Mac. Tato služba vcelku

jednoduchou cestou nabízí možnost programovat pro IOS i z platformy Windows či Linux.

Poslední možností je zvolit některý z multiplatformních nástrojů. Pokud má uživatel například blíže k JavaScriptu, je možné použít software SmartFace. Pokud má blíže k C, je možné použít software Xamarin. Tyto alternativní způsoby mají ovšem jednu nevýhodu. Pro publikování a schválení aplikace na Apple Store je stále potřeba schválení z platformy Mac OS [20].

2.3.2 Publikování aplikace pro IOS

Základním krokem k publikování aplikace na Apple Store je přístup na vývojářský účet, který je zpoplatněn $99 USD na rok. Následně je nutné získat certifikát. Pro vlastníky počítačů Mac je tento krok vcelku jednoduchý. Stačí si jej pouze vytvořit pomocí Keychain Access application. Pokud chceme tento certifikát získat z jiné platformy, je tento proces mnohem složitější a obnáší vyžádání si daného certifikátu a zdlouhavý proces. Existují dva druhy certifikátů a to certifikát pro testování aplikace a certifikát pro uvedení na Apple Store [20].

Pokud byla naše aplikace schválena a máme certifikát, počítač s Mac OS a vývojářský účet, můžeme naši aplikaci nahrát na Apple Store. Pro uživatele jiných platforem bude tento krok nemožný a nelze jej nijak obejít. A to z důvodu, že aplikace se na Apple Store vkládají pomocí application uploader, který funguje pouze na Mac OS X.

Jedinou nadějí je proto půjčení si zařízení Mac nebo si pronajmout již zmiňovanou službu MacinCloud a uploadovat aplikace skrze tuto službu [20]. Stejně jako u aplikací pro Windows či Android je zapotřebí, aby aplikace splňovala určitá kritéria.

2.3.2.1 Podmínky pro průchod schvalovacím řízením aplikace

- Aplikace musí být otestovaná na chyby a pády
- Všechny informace a metadata musí být kompletní a správně
- Kontaktní údaje musí být správné a aktuální
- Poskytnutí aktivní demoverze a účtu společnosti Apple, skrze který bude umožněn přístup k této demoverzi

- Poskytnutí dokumentace k aplikaci, kde jsou vysvětleny funkce, které nemusí být zcela zřetelné na první pohled, popřípadě poskytnutí videozáznamu funkčnosti, pokud není možné využít aplikaci nebo část aplikace kvůli geo-zámku
- Aplikace musí splňovat kritéria stanovená v ustanoveních společnosti Apple, která jsou dostupná ze stránky [21]: https://developer.apple.com/app-store/review/guidelines/

2.3.3 Shrnutí a výhody / nevýhody

Společnost Apple u svých produktů sází spíše na ověřenou kvalitu. Je faktem, že produkty, a hlavně tedy operační systém IOS, jsou perfektně fungující. Není tedy ze strany společnosti Apple zapotřebí vyvíjet velké úsilí na změny v tomto operačním systému. Většina aplikací, ať již továrních nebo stažených z Apple Store, funguje bezchybně [22].

2.3.3.1 Výhody – uživatel

- Uživatelsky přívětivé a intuitivní prostředí operačního systému
- Možnost využívání různých aplikací z ostatních platforem, pokud jsou dostupné
- Druhý největší obchod s aplikacemi App Store [23]
- Aplikace jsou vydávány pouze z placených účtů, vysoká eliminace špatných aplikací
- Asistent Siri
- Dobré propojení různých zařízení Apple

2.3.3.2 Nevýhody – uživatel

- Velká část aplikací je placena
- Zařízení je celkově dražší

2.3.3.3 Výhody – vývojář

- Uživatelé jsou zvyklí za aplikace platit
- Vývojový jazyk je velice přehledný
- Možnost implementovat aplikaci na spoustu verzí operačního systému IOS

2.3.3.4 Nevýhody – vývojář

- Nutnost vlastnit vývojářský účet
- Omezení z hlediska potřebného hardwaru pro vývoj aplikací

3 Výběr operačního systému a vývojového prostředí

Na základě analýzy různých operačních systémů pro mobilní zařízení byl vybrán operační systém Android. Hlavními důvody pro výběr tohoto systému oproti konkurenci jsou:

- Dostupnost vývojového prostředí
- Dostupnost testovacího zařízení, ať již fyzického nebo virtuálního
- Velký počet literatury zaměřené na tuto problematiku

Nadále se tato kapitola bude zaobírat instalací různého softwaru pro vývoj aplikací a výběrem nejlepší varianty.

3.1 Instalace a seznámení s vývojovým prostředím pro Android

Vývojové prostředí představuje software, který slouží pro zjednodušení programování výsledného softwaru. Zpravidla je každé vývojové prostředí orientováno na jeden vývojový jazyk nebo na skupinu více či méně souvisejících programovacích jazyků. Dále tato vývojová prostředí obsahují nástroje, které mohou například hlídat za programátora správnou formu syntaxe, obsahují různé knihovny, mohou hlídat kvalitu softwaru pomocí dodržování různých návrhových vzorů, našeptávání, refaktoring, navigaci v kódu a analýzu kódu [36].

Seznámení s prostředím bude probíhat pouze okrajově, jelikož není obsahem této kapitoly rozebírat programování určité aplikace [24].

3.1.1 Android Studio

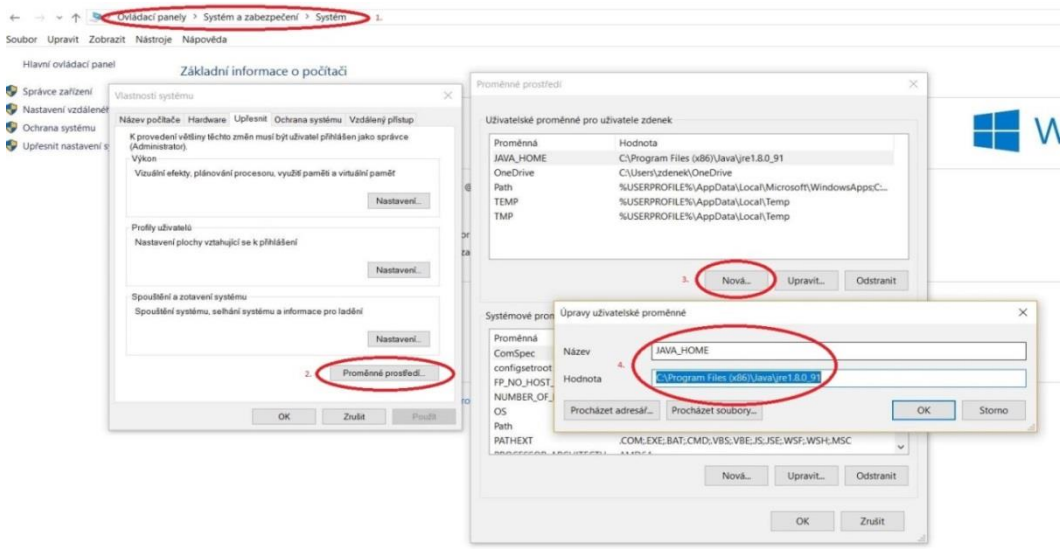
Software Android Studio představuje oficiální nástroj pro vývoj aplikací na platformu Android. Android Studio je volně dostupné ze stránky: https://developer.android.com/studio/index.html.

3.1.1.1 Instalace potřebného softwaru

Instalace vývojového prostředí Android studio je velmi jednoduchá. Stačí pouze stáhnout instalační soubor pro patřičný operační systém, na kterém pracuje používaný počítač, poté jej spustit a projít instalací. Není třeba nadále

nic konfigurovat či nastavovat. Tento instalační balíček již obsahuje veškerý potřebný software, kromě softwaru pro podporu Javy. Tento software na podporu Javy je zapotřebí stáhnout zvlášť. Jedná se o takzvané JDK neboli Java Development Kit. Jedinou komplikaci zde tvoří přidání systémové proměnné pro JDK, které probíhá následovně. Po nainstalování JDK je zapotřebí nastavit systémovou proměnnou. V rámci OS Windows v těchto krocích *Ovládací panely -> Systém a zabezpečení -> Systém -> Upřesnit nastavení systému -> Upřesnit* a zde ve spodní části zvolit *Proměnné prostředí* kde vytvoříme novou proměnnou s názvem JAVA_HOME a s cestou do adresáře s JDK [25].

Obrázek 1 Nastavení systémové proměnné



Zdroj: [26] *Zpracování pomocí OS Windows 10*.

3.1.1.2 Seznámení s prostředím Android Studio

Než si představíme prostředí Android Studio, je potřeba pro budoucí bezproblémový chod povolit virtualizaci v nastavení BIOS. Tato virtualizace slouží pro vytvoření virtuálního zařízení pomocí emulátoru, který již je součástí Android Studio. Nejdříve je zapotřebí se dostat do nastavení systému BIOS. Do tohoto nastavení se dostaneme následujícími kroky [26]:

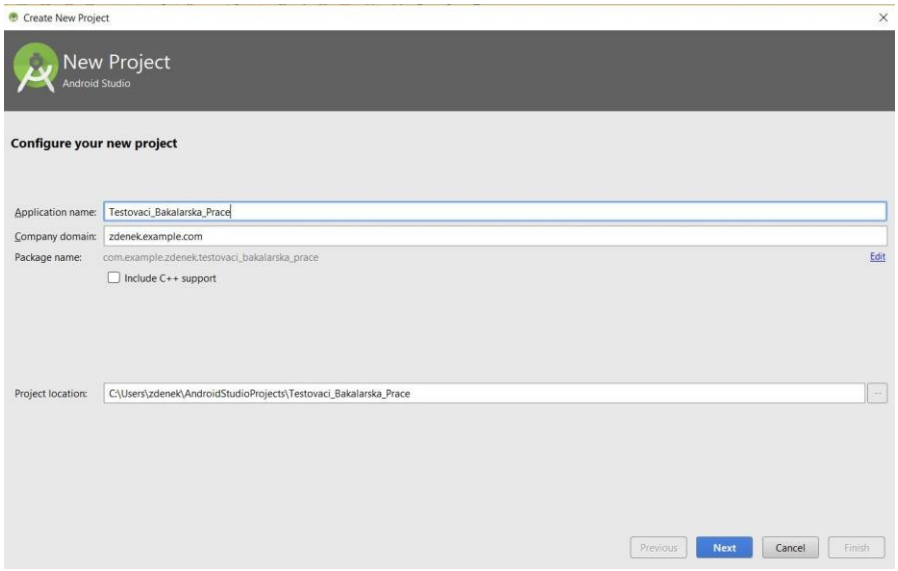
- Při spuštění počítače kliknutím buď na F2, F10 nebo Delete. Tyto vstupy by měly pokrýt velké množství typů BIOS systémů. Pokud nebudou tyto vstupy fungovat, sledujte při zapínání vašeho počítače, kterým tlačítkem vstupujete do systému BIOS. Například „*Press F2 to enter setup*„.

- V nastavení BIOS najdeme Virtualization technology a nastavíme její hodnotu na „Enabled“.
- Změny uložíme a pokračujeme ve spuštění operačního systému.

Nyní již přejdeme k seznámení s vývojovým prostředím Android studio. Po spuštění je nutné vytvořit nový projekt a nastavit určité parametry pro tento projekt.

Vše začíná zvolením si názvu daného projektu. Tento projekt založíme skrze navigaci ve vývojovém prostředí a to *File ->New ->New Project…*

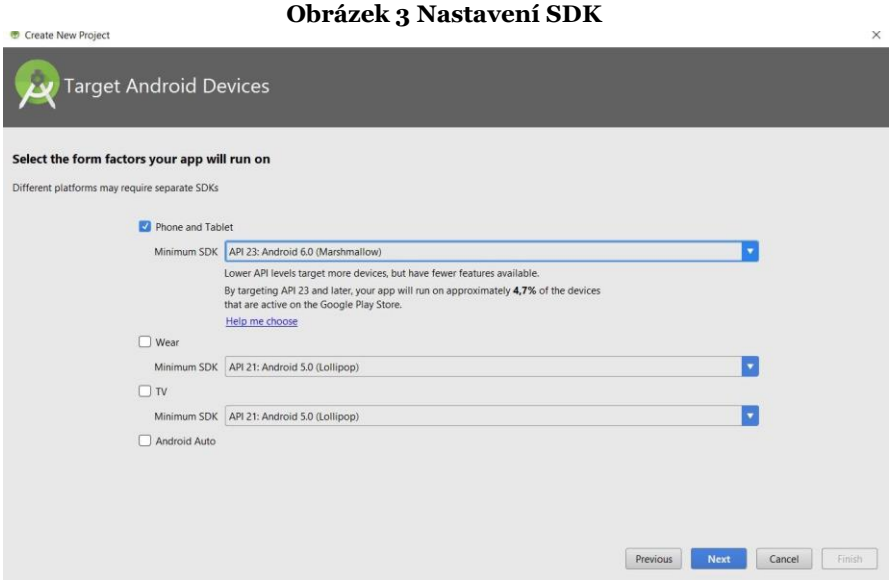
Obrázek 2 Vytvoření projektu



Zdroj: *Zpracování pomocí Android Studio [26].*

V dalším kroku je zapotřebí nastavit, na které verzi operačního systému Android chceme programovat naší aplikaci. Již samotné vývojové prostředí nám oznamuje, kolik procent zařízení zvolená verze pokryje. Doporučuji zvolit verzi Marshmallow Android 6.0. Snížením verze sice pokryjeme větší množství zařízení, ale bude způsobena ztráta mnoha funkcionalit, které ve starších verzích nejsou [26].

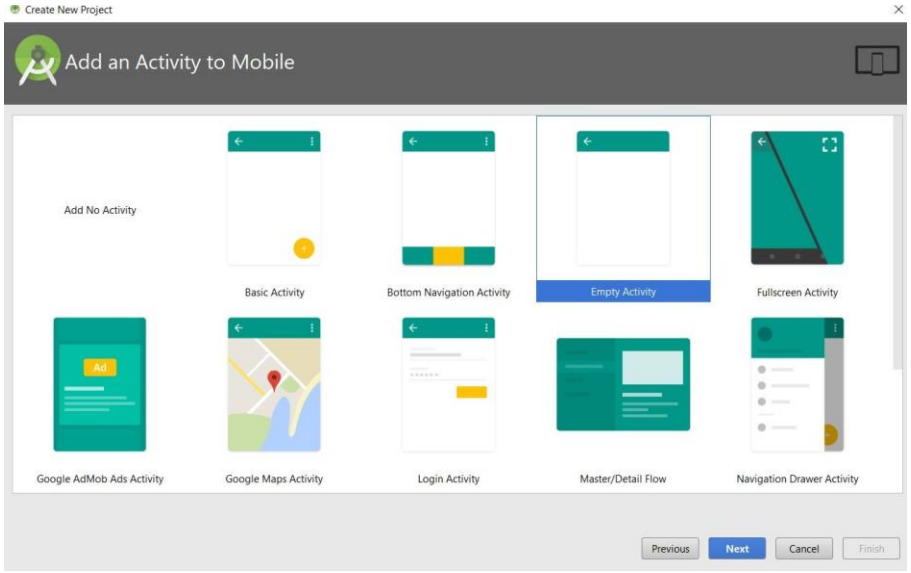
Obrázek 3 Nastavení SDK



Zdroj: *Zpracování pomocí Android Studio [26].*

Android Studio nám již samo bude nabízet přednastavené vzory pro aplikaci. Tyto vzory jsou přednastaveny pro nejčastěji používané problematiky, jako například přihlašovací plocha, navigační menu, mapy, Google ad aktivity a podobné.

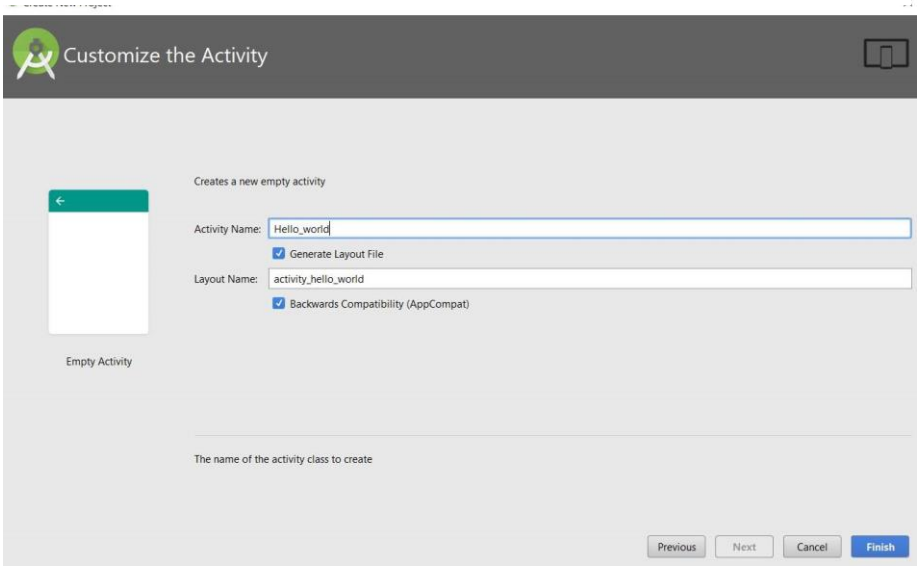
Obrázek 4 Vybrání šablony



Zdroj: *Zpracování pomocí Android Studio [26].*

Finálním krokem je zvolení názvu první plochy neboli aktivity dané aplikace.

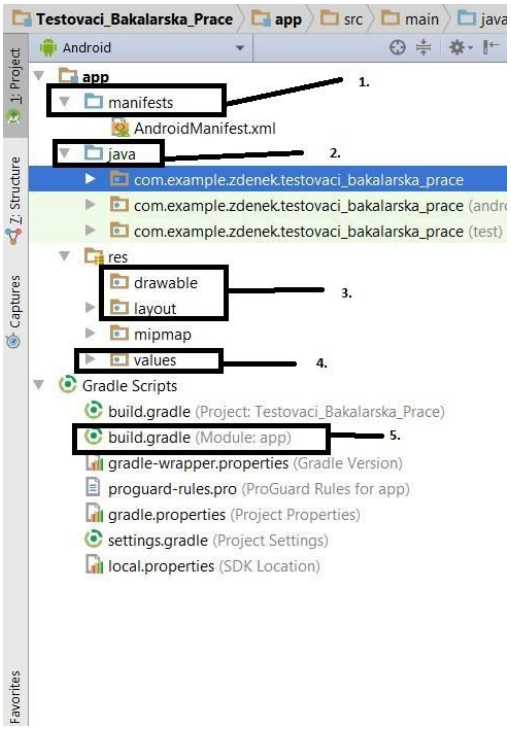
Obrázek 5 Zvolení názvu projektu



Zdroj: *Zpracování pomocí Android Studio [26].*

Nyní je projekt založen. V další části bude vysvětlena anatomie aplikace. Anatomie aplikace je zobrazena v levé části vývojového prostředí a obsahuje veškerou strukturu aplikace.

Obrázek 6 Struktura projektu Android Studio

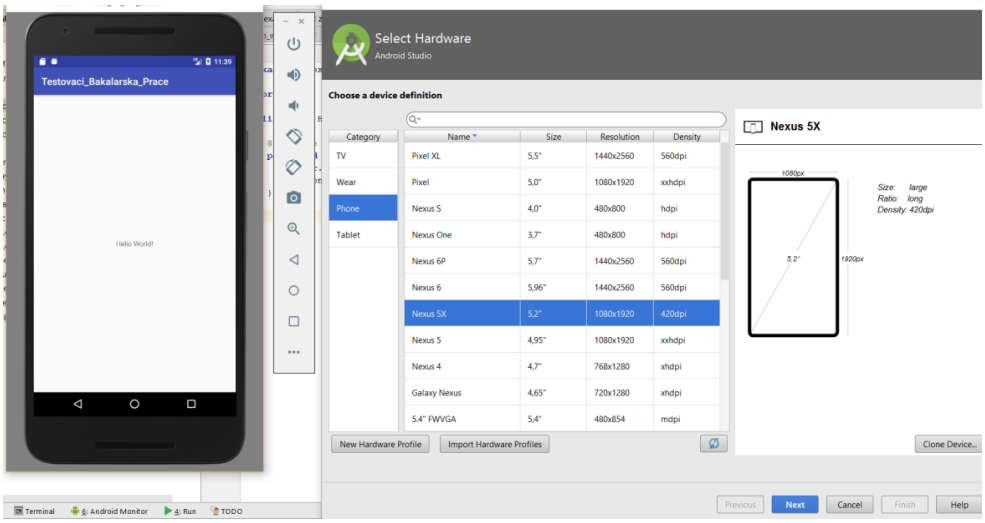


Zdroj: *Zpracování pomocí Android Studio [26].*

1. **AdroidManifest.xml** – Tento soubor popisuje fundamentální charakteristiky dané aplikace a definuje jednotlivé komponenty v aplikaci. Je uložen ve složce manifests.
2. **Java** – Tato složka v sobě obsahuje soubory s příponou .java. Tyto soubory v sobě mají třídu, která se spustí při zapnutí aplikace, například kliknutím na logo aplikace na mobilním zařízení.
3. **Drawable a layout** – Drawable v sobě obsahuje objekty, které jsou navrhnuté pro vysoké rozlišení. Layout v sobě obsahuje informace, které definují uživatelské prostředí.
4. **Values** – Tento xml soubor obsahuje kolekci zdrojů, jako řetězce string, definování barev a další.
5. **Build.grandle** – Tento soubor se generuje automaticky a obsahuje například VersionCode, versionName, applicationID, buildToolsVersion a podobné informace.

Spouštění aplikace probíhá v prostředí Android Studio, lze spustit klávesovou kombinací Shift+F10. Po prvním spuštění nám bude nabídnuto vybrat si přednastavená zařízení a popřípadě stáhnout na toto zařízení určitou verzi systému. Po tomto výběru se spustí emulátor s vybraným zařízením a softwarem. Na tomto virtuálním zařízení se zobrazí spuštěná aplikace [26].

Obrázek 8 Výběr virtuálního zařízení Android Studio



Zdroj: *Zpracování pomocí Android Studio [26].*

3.1.2 Eclipse

Vývojové prostředí Eclipse bylo velmi oblíbené prostředí pro programování aplikací před vznikem Android Studia. Jeho instalace je ovšem náročnější než instalace oficiálního vývojového prostředí a to především z nutnosti doinstalovat i další potřebný software [27].

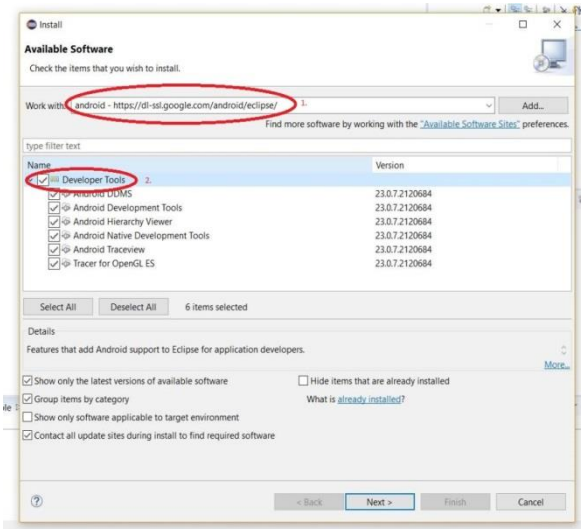
3.1.2.1 Instalace potřebného softwaru

Vývojové prostředí je potřeba doplnit o další software a to:

1. **Java Development Kit (JDK) -** Je produktem Oracle Corporation a jsou v něm obsaženy základní nástroje pro vývoj aplikací s platformou Java. Java Development Kit je volně dostupný ze stránky [27]: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.
2. **Software Development Kit (SDK) -** Je také produktem Oracle Corporation a je potřebný k testování a tvorbě aplikací. Software Development Kit je volně dostupný ze stránky [27]: http://developer.android.com/sdk/index.html.

3. **Android Development Tools -** Po stažení veškerého potřebného softwaru je zapotřebí zajistit, aby tento software spolu komunikoval. K tomuto účelu slouží Android Development Tools, což je plugin pro vývojové prostředí Eclipse. Instalace tohoto pluginu probíhá přímo ve vývojovém prostředí Eclipse a to v těchto krocích [27]:
   - Klikneme na nabídku Help -> Instal New Software
   - Po otevření okna zvolíme za Work with adresu `https://dl-ssl.google.com/android/eclipse/`
   - Ve spodním okně se objeví Developer Tools - tuto možnost zaškrtneme. Dále nic nenastavujeme a pouze plugin stáhneme a nainstalujeme do Eclipse.

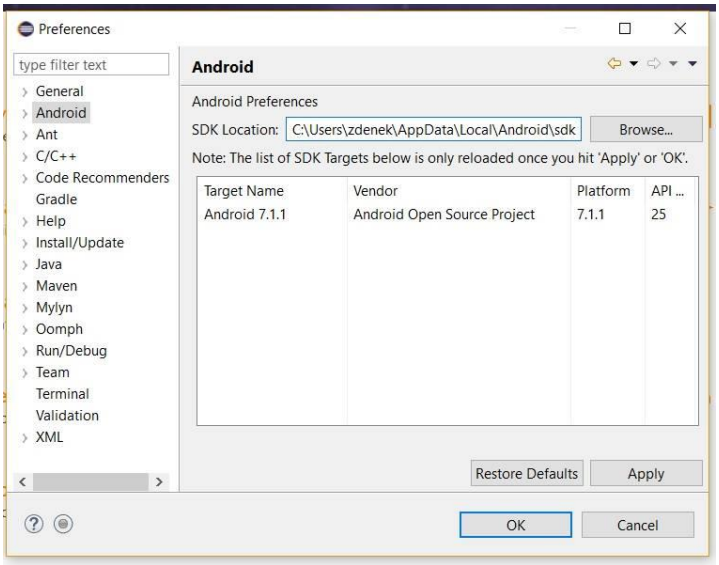
Obrázek 9 Instalace plugin Eclipse



Zdroj: *Vlastní zpracování s využitím Eclipse*

- Posledním krokem je konfigurace pluginu ADT. Tato konfigurace probíhá v následujících krocích. *Windows -> Preferences -> Android* a do políčka na pravé straně zadáme cestu ke složce s SDK [27].

Obrázek 10 Konfigurace pluginu ADT



Zdroj: *Vlastní zpracování s využitím Eclipse*

Již z tohoto popisu je zřetelné, že pouze samotná instalace softwaru pro Eclipse je mnohem náročnější než instalace Android Studia.

3.1.3 MIT App Inventor

Prostředí App Inventor je nástrojem společnosti Google. Toto prostředí je vhodné pro uživatele, kteří nemají zkušenosti s programováním, jelikož je toto prostředí založeno na používání „blokového“ programování.

App Inventor je vizuálně zpracovaný přesunovací nástroj pro platformu Android. Programování probíhá sestavením uživatelského rozhraní a následného sestavení bloků, které určují funkcionalitu dané aplikace.

3.1.3.1 Instalace potřebného softwaru

Instalace zde žádná není, jelikož se jedná o prostředí, které běží v cloudu. Je zapotřebí se pouze přihlásit pomocí účtu Google. To znamená, že samotná aplikace se nachází na serverech společnosti Google. Tato webová aplikace je volně dostupná ze stránky: http://ai2.appinventor.mit.edu/ [39].

3.2 Zhodnocení a výběr vývojového prostředí

Při vývoji aplikace pro platformu Android je uznáváno pouze jedno oficiální vývojové prostředí a tím je Android Studio. Toto prostředí je navíc volně dostupné. Z těchto důvodů je vybráno pro tvorbu aplikace. Možnost vývoje v Eclipse nebyla vybrána hlavně z důvodů většího množství procesů nastavování. Možnost MIT App Inventor nebyla vybrána hlavně z důvodu blokové tvorby zdrojového kódu, která neposkytuje dostatečnou svobodu při tvorbě složitější aplikace.

4 Návrh uživatelského prostředí a datových objektů

V této kapitole je představeno logo aplikace a návrh uživatelského prostředí.

4.1 Název a logo

Název aplikace byl postupem času měněn. Cílem bylo pojmenovat aplikaci jménem, které souvisí s účelem aplikace. Ujal se název Ztrátoše. Tento název byl zvolen hned z několika hledisek:

- Souvisí s účelem aplikace.
- Je snadno zapamatovatelný.
- Je jedinečný oproti klasickému názvu „ztráty a nálezy“.
- Název může označovat určitou osobu či ikonu aplikace, ve smyslu například známého mimozemšťana českého portálu s elektronikou Alza.cz. Což může v budoucnosti vést k navržení maskota aplikace.

Z hlediska loga byl zvolen otazník, který slouží jako světově používané interpunkční znaménko pro tázací větu, ale zároveň slouží jako běžně používaný znak pro chybějící, postrádající se nebo nezvěstnou věc. Jako pomoc při výběru barvy posloužila webová stránka http://paletton.com. Byla vybrána modrá kombinace.

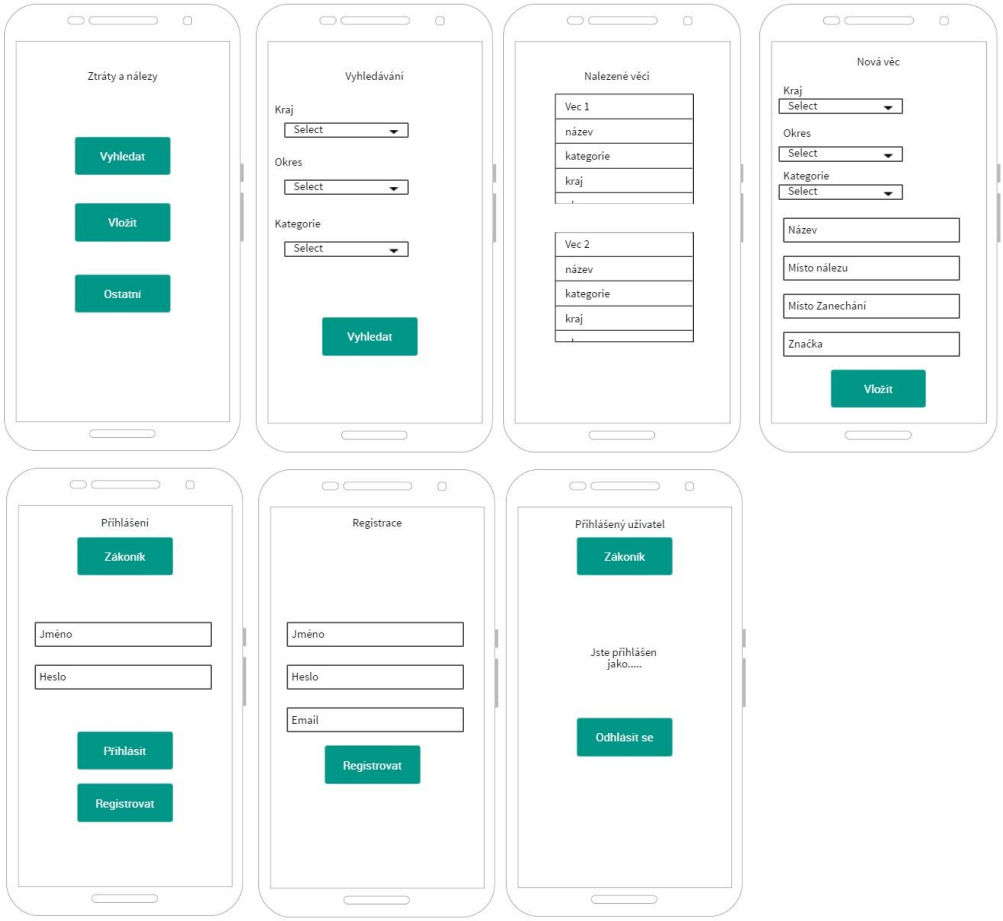
Obrázek 11 Návrh loga

Zdroj: *Vlastní zpracování pomocí Adobe Photoshop CS6[33].*

4.2 Návrh layoutu

Správně navržený layout usnadňuje uživateli používání výsledné aplikace. Aplikace vytvořená v rámci této práce má 7 ploch. Na následujícím obrázku [12] jsou jednotlivé plochy představeny. Layout byl vytvořen pomocí stránky mockflow.com. Na této stránce je možné po registraci získat přístup na 30 dní užívání zdarma. Stránka je dostupná z: https://mockflow.com.

Obrázek 12 Layout aplikace



Zdroj: *Zpracování pomocí stránky mockflow.com*

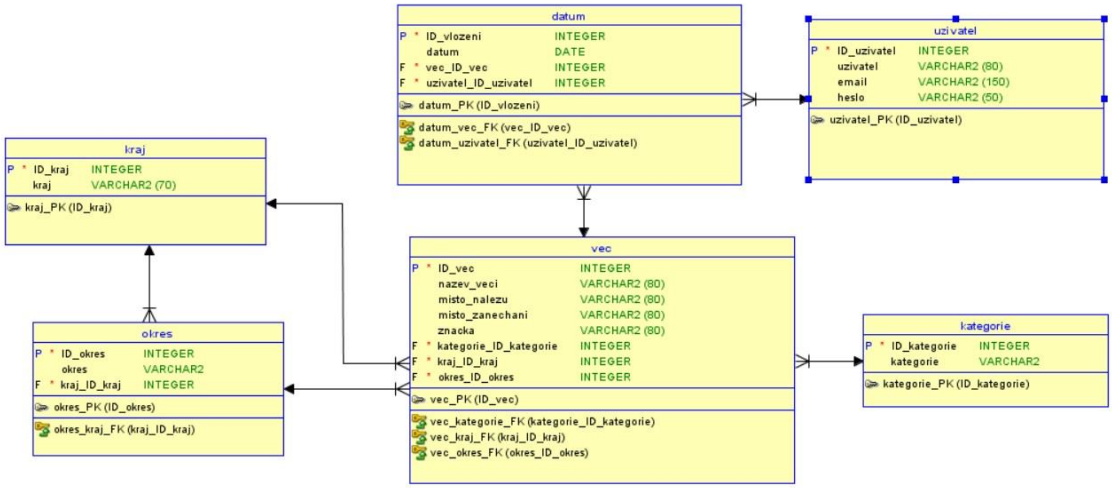
4.3 Datové objekty a vazby mezi nimi

V této kapitole je představen relační model aplikace a UML diagram aktivit v API.

4.3.1 Relační model

Relační model byl vytvořen pomocí software Oracle SQL Developer, který je dostupný ze stránek: http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html.

Obrázek 13 Relační model



Zdroj: *Vlastní zpracování pomocí Oracle SQL Developer*

1. **Uživatel –** Tabulka *uživatel* v sobě uchovává informace o uživateli.
2. **Datum –** Tabulka *datum* v sobě uchovává informace o datu vložení záznamu. Zároveň se pomocí této tabulky propojuje vložená věc s uživatelem pomocí cizích klíčů z tabulky *věc* a *uživatel*.
3. **Kraj –** V tabulce *kraj* jsou uložené kraje České republiky.
4. **Okres –** Tabulka *okres* v sobě uchovává informaci o okresech. Tabulka přebírá cizí klíč z tabulky *kraj*. To zaručuje, že každý okres je přirazen ke správnému kraji.
5. **Kategorie –** Tabulka *kategorie* v sobě uchovává základní možnosti rozdělení věcí do různých kategorií.
6. **Věc –** Tabulka *věc* v sobě obsahuje informace o nalezené věci. Zároveň tato tabulka přebírá cizí klíč z tabulek *kraj, okres* a *kategorie*. V tabulce jsou povoleny duplicitní záznamy.

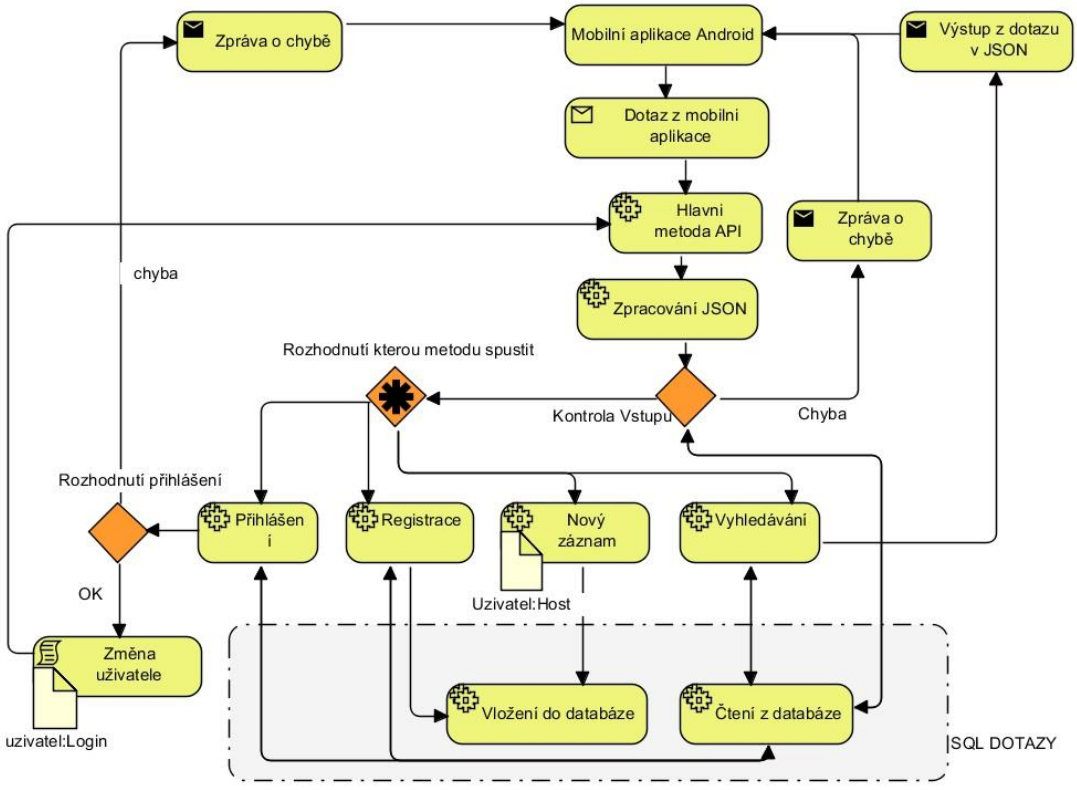
Relační model splňuje náležitosti třetí normální formy. Pro splnění třetí normální formy je zapotřebí splnit tyto náležitosti:

1. **První normální forma** – Každý z atributů obsahuje pouze atomické hodnoty, tedy hodnoty, které nejsou z pohledu databáze dále dělitelné [32].
2. **Druhá normální forma** – Je splněna, pokud je splněna první normální forma a atributy jsou plně závislé na primárním klíči. U druhé normální formy musí být atribut závislý na celém primárním klíči, nejen na jeho podmnožině [32].
3. **Třetí normální forma** – Musí být splněny předešlé dvě normální formy a zároveň žádný atribut není tranzitivně závislý na klíči. Třetí normální forma povoluje tranzitivní závislost u klíčových prvků. Takže závislost tabulky *okres* na *kraj* a jejich použití v tabulce *věc* je v pořádku [32].

Oracle SQL Developer umožňuje na základě vytvořeného relačního modelu vygenerovat SQL příkaz pro vytvoření a propojení všech tabulek. Tímto postupem byla vytvořena i databáze využívaná v této bakalářské práci.

4.3.2 UML Diagram

Obrázek 14 UML Diagram



Zdroj: *Vlastní zpracování v softwaru Visual Paradigm Standart*

Výše uvedený UML diagram byl vytvořen pomocí softwaru Visual Paradigm, který je dostupný jako zkušební verze na 30 dnů a je dostupný ze stránek:

https://www.visual-paradigm.com/download/

UML diagram na obrázku [14] představuje přibližné fungování API.

Průběh UML diagramu je následovný. Z mobilní aplikace je odeslán dotaz ve formátu JSON. Tento dotaz obdrží hlavní metoda v API, která jej zpracuje na pole. Toto pole následně prochází kontrolou hodnot. Pokud je vše v pořádku, API zjistí na základě složení vstupu jaká metoda je vyžadována mobilní aplikací.

- Pokud je voláno vyhledávání provede se čtení z databáze.
- Pokud je volána registrace, provede se nejdříve čtení z databáze pro kontrolu duplicity hodnot. Na základě kontroly se buď uživatel vloží do databáze, nebo ne.
- Pokud je volána metoda přihlášení provede se nejdříve čtení z databáze pro porovnání přihlašovacích údajů. Je-li vše v pořádku, odešlou se zpět do aplikace zadané přihlašovací údaje.
- Pokud je volána metoda pro vložení nového záznamu, provede se zápis do databáze buď pod uživatelem host, nebo pod přihlášeným uživatelem.

5 Vývoj mobilní aplikace

Pro vývoj aplikace v rámci této bakalářské práce byly využity následující nástroje a služby:

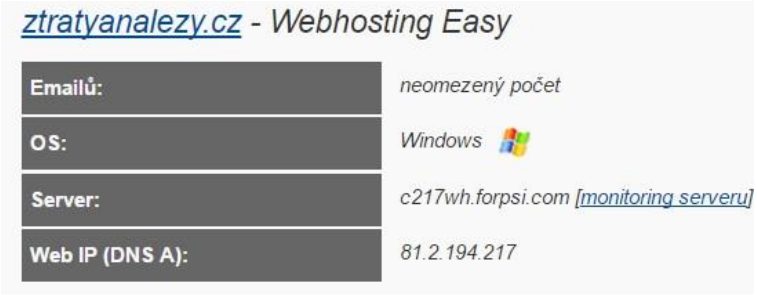
- Vývojové prostředí – Microsoft Visual Studio https://www.visualstudio.com/cs/downloads/?rr=https%3A%2F%2Fwww.google.cz%2F
- Vývojové prostředí – Android Studio https://developer.android.com/studio/index.html
- Prostředí pro správu databáze MylittleAdmin od poskytovatele Forpsi.com
- Databázový prostor od poskytovatele Forpsi.com
- Doménová služba od poskytovatele Forpsi.com
- Webhosting od poskytovatele Forpsi.com

5.1 Nastavení webhostingu a domény

Prvním krokem, který je zapotřebí vykonat, je nastavení webhostingu a domény. Následující ukázky jsou prováděny u poskytovatele Forpsi. Je možné, že se uživatelská administrace bude lišit v rámci různých poskytovatelů. Ovšem pro správnou funkčnost aplikace je nutné následující kroky provést u všech poskytovatelů.

Nejdříve je potřeba si u svého poskytovatele zajistit služby webhostingu, domény a databázového prostoru. Vytváření zmíněných služeb je u většiny poskytovatelů doprovázeno návodem co a jak vytvořit. Z tohoto důvodu zde není tento postup zmíněn. Nicméně je důležité si ověřit, zda má doména nastavené své DNS na stejnou IP adresu jako webhosting. Tento krok je důležitý především pokud doména a webhosting byli zakoupeny zvlášť. Na obrázku [15] jsou zobrazené informace k webhostingu, který nese název ztratyanalezy.cz a jeho DNS je 81.2.194.217.

Obrázek 15 Webhosting



Zdroj: Vlastní zpracování

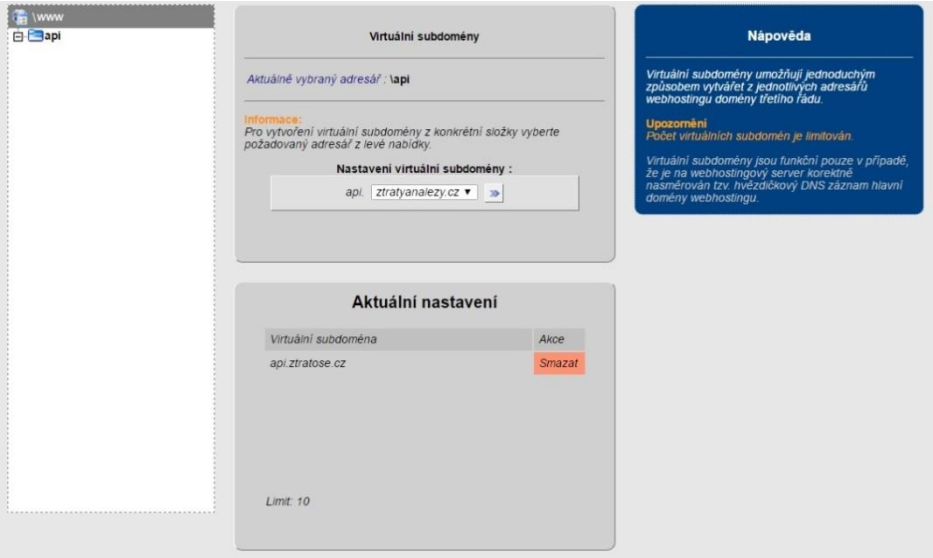
Na obrázku [16] je zobrazené nastavení domény, kde si můžeme všimnout, že její název je odlišný od názvu webhostingu. A bylo nutné zde nastavit správně DNS a to na DNS webhostingu.

Obrázek 16 Nastavení domény

Zdroj: Vlastní zpracování

Potřeba procesu nastavování záleží na poskytovateli. V tomto případě byla doména a webhosting zakoupeny zvlášť. V ukázce bylo dosaženo propojení domény a webhostingu. Nyní je potřeba vytvořit u naší domény její virtuální subdoménu, která bude sloužit pro přístup k API. Tento proces nastavování je možný skrze nastavení webového rozhraní našeho webhostingu, obrázek [17].

Obrázek 17 Nastavení virtuální subdomény



Zdroj: Vlastní zpracování

Posledním závěrečným krokem v nastavování je vytvoření aplikačního adresáře. Tento adresář umožní provozování více na sobě nezávislých aplikací. Každá z těchto aplikací ovšem musí obsahovat vlastní soubor web.config dle různých adresářů webhostingu, obrázek[18].

Obrázek 18 Nastavení aplikačního adresáře

Zdroj: Vlastní zpracování

Nyní je již vše připravené a nastavené pro napojení API.

5.2 API

Tvorba API probíhá v prostředí Microsoft Visual Studio. Toto prostředí bylo vybráno především z důvodu vytvoření testovací aplikace, která umožňuje rychlejší testování změn v API. API je zkratka pro Application Programming Interface. Tato zkratka označuje sbírku různých procedur, tříd, protokolů či funkcí určité knihovny, programu či operačního systému. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu [29].

5.2.1 Formát vstupu JSON

Pro výměnu dat je zvolen formát JSON. Jedná se o upravený formát pro výměnu dat. Tento formát je snadněji zapisovatelný a čitelný pro člověka. Zároveň je jednoduší ho analyzovat i generovat pomocí strojů, na rozdíl například od XML. JSON má formát textu a je na použitém jazyce zcela nezávislý. Využívá konvence jazyků rodiny C [28]. Formát JSON využitý v API je následující:

{“nazev1“:“hodnota1“,“nazev2“:“hodnota2“,“nazev3“:“hodnota3“}

- *{}* značí konec a začátek jednoho objektu v rámci JSON
- ““ mezi kterýma jsou uloženy názvy a hodnoty proměnných
- : oddělují název a hodnotu proměnné
- , oddělují jednotlivé dvojice název a hodnota proměnné

5.2.2 Formát výstupu JSON

Je zapotřebí, aby metody, které vrací údaje do aplikace, odesílaly data ve formátu JSON. Tento formát se od formátu na vstupu liší. Tato odlišnost je způsobena vývojovým prostředím. Microsoft Visual Studio neobsahuje v základní verzi knihovnu na zpracování vstupu ve formátu JSON. Bylo tedy nezbytné naprogramovat patřičné metody, které zpracují vstup. Z tohoto důvodu byl zvolen odlehčený formát JSON na vstupu do API. Android Studio obsahuje knihovny na zpracování formátu JSON, tudíž jsou data do aplikace odesílána v plné podobě JSON.

- [] značí konec a začátek pole, před hranaté závorky se vkládá název pole oddělen od závorek dvojtečkou
- *{}* značí konec a začátek jednoho objektu v rámci JSON pokud jsou

na krajích a zároveň značí jednotlivé prvky pole, pokud jsou v hranatých závorkách

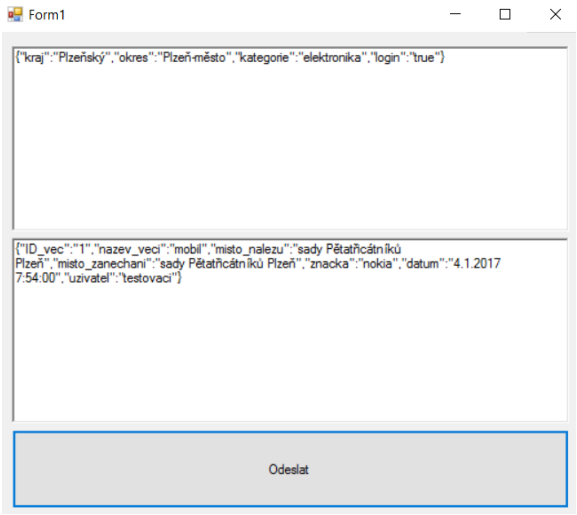
- “” mezi kterými jsou uloženy názvy proměnných a hodnoty
- : oddělují název proměnné a hodnotu
- , oddělují jednotlivé dvojice název proměnné a hodnota

{“VEC“:[{“nazev1“:“hodnota1“},{ nazev2“:“hodnota2“}]}

5.2.3 Vytvoření testovací aplikace na API

Pro snadnější testování provedených změn při programování API bylo vytvořeno jednoduché prostředí pomocí Microsoft Visual Studio.

Obrázek 19 Testování API



Zdroj: *Vlastní zpracování v Microsoft Visual Studio*

Toto prostředí obsahuje pouze dvě textová pole a jedno tlačítko. Do prvního textového pole se vkládá vstup ve formátu JSON. V druhém textovém poli se vypisuje výstup z API, také ve formátu JSON nebo pouze jako řetězec dle odpovídající metody. Finální mobilní aplikace funguje na naprosto stejném principu jako tato jednoduchá aplikace.

V předchozí kapitole bylo ukázáno, jak propojit všechny náležitosti v rámci webhostingu a domény. Následující úsek kódu ukazuje, jakým způsobem jsou odesílána data z aplikace.

```
var result = client.PostAsync("http://api.ztratose.cz/v1/Data/zapsaniveci", content).Result;
```

Můžeme si povšimnout, že metoda *PostAsync* odesílá na stránku, kde je uložené API, parametry, obrázek [17]. První parametr je tedy webová stránka, kde je uložené API. Na konci této webové stránky je název přijímací metody,

která přebírá data od aplikace *zapsaniveci*. Této přijímací metodě je předán obsah prvního textového pole.

5.2.4 API propojení s databází.

Pro správné vykonávání metod a zajištění potřebných funkcionalit je důležité, aby API umělo komunikovat se svým okolím.

Komunikace mezi API a databází je definována v souboru *Web.config*, který je ve formátu XML. Představuje soubor pro aplikaci technologie ASP.NET s rozhraním NET Framework a ASP.NET. Tento soubor slouží jako konfigurační soubor pro komunikaci s databází. Je součástí kořenového adresáře projektu v Microsoft Visual Studio, stejně jako soubor, který obsahuje zdrojový kód API [31].

Web.Config v našem případě určuje hlavně připojení na databázi. Toto připojení je uloženo pod názvem *ConnectionString*. *ConnectionString* obsahuje hodnoty jako adresu, na které lze nalézt databázi, název databáze, uživatele a heslo. Tyto informace jsou vyžadovány v souboru se zdrojovým kódem API a to u metod pro zápis a čtení z databáze.

5.2.5 Čtení a zápis nad databází

Metody pro čtení záznamů z databáze jsou celkově označovány jako *SQLReader…*, kde jsou tečky nahrazeny názvem metody pro čtení. V API se využívají dva druhy metod pro čtení. První druh vyžaduje jako parametr pouze SQL příkaz a vrací všechny záznamy. Druhý druh vrací pouze hodnoty jednoho sloupce a vyžaduje parametr navíc, který představuje název sloupce pro vypsání.

Metody, které určitým způsobem přidávají či mění hodnoty v databázi, jsou značeny jako *SQLWriter…*, kde jsou stejně jako v případě metod pro čtení tři tečky nahrazeny názvem metody. Hlavním rozdílem mezi metodou obstarávající změny v databázi a výpisem z databáze je, že metody provádějící změny nevrací žádné informace z databáze. Těmto metodám je na výstupu přidělen textový řetězec pro kontrolu průběhu.

5.2.6 Využívané funkce jazyka C#

V této podkapitole budou rozebrány nejvíce využívané funkce, které poskytuje jazyk C# a byly využity.

- **Contains()** – Určení, zda řetězec či pole nad kterým je tato funkce volaná, obsahuje řetězec vložen jako parametr funkce [30].
- **DateTime.Now.ToString("M/d/yyyy H:mm:s")** – Funkce získává aktuální datum ve formátu „28.12.2017 18:05:56“.
- **Trim()** – Tato funkce odstraní z konce a začátku řetězce, nad kterým je volaná, znaky vložené jako parametr funkce [30].
- **Split()** – Funkce vloží do pole řetězec, který rozdělí na jednotlivé indexy. Dělení řetězce probíhá podle znaku, který se do funkce vloží jako parametr.
- **Replace()** – Tato funkce nahradí z řetězce znaky, které má jako první parametr a nahradí je znaky, které má jako druhý parametr [30].
- **Regex *XY*.IsMatch()** – U této funkce se vytvoří takzvaný „Regex“, který představuje určitý vzor pro řetězec. Může určovat, které znaky jsou povolené pro řetězec, ale také formát řetězce. *XY* poté představuje název předlohy a do parametru se vkládá řetězec pro otestování. [30].
- **Array.FindIndex()** – Tato funkce je volána nad polem. Jako parametr očekává řetězec, u kterého chceme v poli najít jeho index. Pole projíždí od začátku do konce a vrací první nalezenou shodu. Po nalezení se ukončí [30].

5.2.7 Zpracování vstupu JSON

Prvním krokem ve zdrojovém kódu API je rozlišení vstupních hodnot. API očekává vstup ve zvoleném formátu JSON. Převezme vstup od aplikace v podobě textového řetězce. Tento řetězec je následně zpracován pomocí funkcí *Replace()*. Jsou ponechány pouze znaky *(:) a (,)*, které oddělují parametry a dvojice parametrů. Dle těchto znaků funkce *Split()* následně rozdělí upravený řetězec JSON do pole. Po tomto kroku má pole následující strukturu:

[nazev1 hodnot1 nazev2 hodnot2 nazev3 hodnot3…]

5.2.8 Zaručení dynamiky pořadí na vstupu

Jelikož jednotlivé dvojice název a hodnota mohou být ve formátu JSON v libovolném pořadí, je důležité rozlišit, kde se jaký název nachází. Metoda obdrží zpracované pole názvů a hodnot, a obsahuje své vlastní pole názvů, se kterými má pracovat. Každá metoda pracující se vstupem JSON obsahuje cyklus *foreach,* který prochází pole názvů své metody. V tomto cyklu se pomocí *Array.FindIndex()* hledá index, na kterém se nachází potřebný název ve vstupním poli. Jelikož jsou název a hodnota vždy ve dvojici, API ví, že hodnotu daného názvu má očekávat na indexu o jedna větší. Nalezené indexy jak názvu, tak hodnoty se ukládají do dvojrozměrného pole.

5.2.9 Kontrola vstupu

Kontrola probíhá dvěma způsoby. Z tohoto důvodu zde existují dvě pole pro očekávané názvy, viz kapitola 5.2.8.

První způsob porovnává statické vstupy, tedy vstupy, které jsou uložené v databázi *kraj, okres, kategorie*. Porovnává, zda hodnota, kterou dostala pod názvem například kraj, se opravdu vyskytuje i v tabulce *kraj* uložené v databázi. Tato kontrola probíhá pro každý z názvů pomocí cyklu foreach nad polem názvů kontrolní metody, kde se má provádět kontrola prvním způsobem. Výsledkem každé kontroly je naplnění kontrolní proměnné, pokud se záznam vyskytuje v databázi. Pokud kontrola neprošla, kontrolní proměnná zůstává prázdná. Na základě kontrolní proměnné se do pole výsledku vkládá buď řetězec „dobře“ nebo „špatně“. Se vkládáním řetězce „špatně“ se do pole chyb vloží název, jehož hodnota neprošlo kontrolou.

Druhý způsob kontroly se vyhodnocuje obdobným způsobem, akorát je zaměřen na vstupy od uživatele. Rozdílný je způsob kontroly, který je realizován pomocí *Regex XY.IsMatch()*. Zde se nekontroluje, zda se hodnota v názvu nachází v databázi, ale kontroluje se, zda neobsahuje speciální znaky. V případě emailu se kontroluje, zda splňuje formát. V případě hesla se kontroluje, zda je vyplněné. Po splnění či nesplnění kontroly se postupuje stejně jako v prvním způsobu.

Posledním krokem kontroly je rozhodnutí o výstupu. Toto rozhodnutí probíhá na základě pole výsledků. Pokud toto pole výsledků obsahuje řetězec

„špatně“, kontrola vypíše ve formátu JSON chybovou zprávu o názvech, které neprošly kontrolou. Tyto názvy jsou uloženy v poli chyb. Pokud vše proběhlo v pořádku, vrací se řetězec „OK“. Řetězec „OK“ se vrací pouze do hlavní metody, nikoli do aplikace. Hlavní metoda na základě výstupu z kontroly pomocí klauzule *if* zjistí, jestli může pokračovat dále nebo vypsat pole chyb.

5.2.10 Rozhodnutí, která metoda se má volat

Rozhodnutí o tom, která metoda se má na základě vstupu volat, probíhá v hlavní metodě, která přijímá data z aplikace. Toto rozhodování probíhá pouze pomocí klauzule *if*, jelikož jsou vstupy postavené tak, že neexistuje jeden formát vstupu pro dvě metody. Existují ovšem metody, které očekávají stejné hodnoty, viz obrázek [19].

Obrázek 20 Metody se stejnými parametry

Zdroj: *Vlastní zpracování*

Na obrázku [19] je možné si všimnout, že jak metoda pro hledání, tak metoda pro vložení sdílí hodnoty *kraj, okres, kategorie*. Metoda pro vložení obsahuje i další názvy. To ovšem klauzule *if* neumí rozlišit, proto se musí doplnit o negaci funkce *Contain()*. U klauzule *if* pro hledání bude tedy negace funkce *Contains()* na jeden z prvků, který je u metody pro vložení navíc. Tímto způsobem se zaručí spuštění správné metody.

5.2.11 Přihlášení uživatele

Metoda obstarávající přihlášení očekává na vstupu přihlašovací jméno a heslo. Tyto údaje následně zkontroluje s údaji, které má pod zadaným přihlašovacím jménem v databázi. Pokud je vše v pořádku, vrátí se uživateli

heslo zpět. Pro vkládání dat pod přihlášeným uživatelem bude aplikace používat vrácené heslo jako token. Tento token v kombinaci se jménem je kontrolován v API pro každé vložení nebo jinou operaci. Uživatel má i možnost se odhlásit. V tom případě se jeho vkládání věcí bude provádět na defaultním uživateli host.

5.2.12 Vyhledávání

Vyhledávání je uživateli umožněno na základě názvů *kraj, kategorie, okres*. Vyhledávání nejdříve zpracuje vstup, viz kapitola 5.2.8. Následuje segment klauzulí *if*. První kontrola zjistí, zda jsou všechny názvy prázdné. Pokud ano, vypíše všechny věci z databáze. Pokud je alespoň jeden název vyplněn, je zapotřebí přidat do SQL příkazu *WHERE*. Za toto *WHERE* se následně budou přiřazovat jednotlivé úseky SQL příkazu dle toho, jaké názvy jsou či nejsou vyplněné. U prvního názvu se kontroluje pouze to, zda je vyplněný. Pokud ano, přidá se patřičná část do SQL příkazu. U druhého prvku se již musí kontrolovat nejen to, zda je vyplněný, ale také zda je předchozí kontrolovaný název vyplněný. Pokud je vyplněn právě kontrolovaný název a zároveň i předchozí název, musí se vložit spojka *AND* do SQL příkazu. Výsledný SQL příkaz se vrátí do hlavní metody, ve které se zavolá metoda po čtení.

5.2.13 Registrace

U registrace nového uživatele je nutné ověřit, zda se jméno a email uživatele již v databázi nevyskytuje. Toto ověření probíhá pomocí SQL příkazu, který hledá v tabulce uživatelů shodu. Pokud ji najde, vrátí záznam do kontrolní proměnné. Pokud ji nenajde, proměnná zůstává prázdná. Na základě obsahu kontrolní proměnné se rozhodne, zda může být uživatel vložen nebo ne. Kontrola zadaných hodnot probíhá v metodě obstarávající kontrolu všech vstupů do API, viz kapitola 5.2.9.

5.2.14 Smazání záznamu

Metoda pro smazání záznamu je pouze součástí API. Tato metoda není naimplementovaná do mobilní aplikace a spadá do návrhů změn do budoucna. Metoda smaže záznamy z databáze, které jsou starší 30 dnů. Pro toto smazání si nejdříve vyžádá z databáze identifikátory data z tabulky *datum*, kde je podmínka pro smazání splněna. Na základě těchto identifikátorů data zjistí, o kterou věc se jedná a uloží si také identifikátor dané věci. Následně si před smazáním

záznamů a zrušení vazby skrz tabulky *datum* nechá vypsat emailové adresy uživatelů, kterých se toto smazání týká a uloží je do pole. Poté probíhá smazání záznamů nad tabulkou *datum* a až následně nad tabulkou *věc*. V budoucnu bude zapotřebí na získané emailové adresy zaslat informativní zprávu uživatelům o smazání jejich záznamů.

5.3 Vývoj Aplikace – teoretická část

Kapitola zabývající se vývojem aplikace se skládá ze dvou částí. První částí je část teoretická. Na teoretickou část navazuje část praktická. V teoretické části jsou popsány důležité třídy, služby, soubory či adresáře využívané v rámci Android Studio. V praktické části jsou popsány jednotlivé aktivity mobilní aplikace.

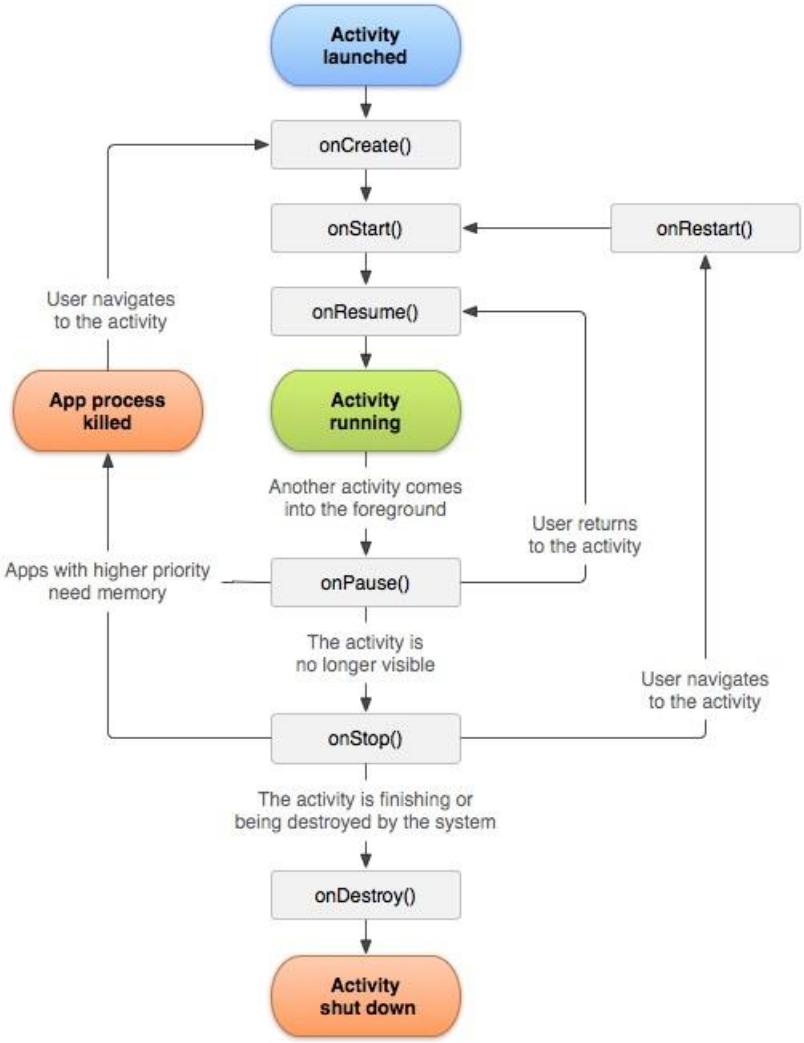
5.3.1 Aktivity

Každá aktivita v aplikaci představuje jednu obrazovku s uživatelským rozhraním. Každá aktivita má svůj životní cyklus, který určuje pořadí, kdy se má která aktivita spustit nebo při jaké situaci se má spustit. Když je například spuštěna nová aktivita, je vložena na začátek pomyslné fronty aktivit, kde aktivita na začátku je ta, která právě probíhá. Ostatní aktivity jsou ve frontě. Aktivity, které čekají ve frontě, se nespustí, dokud aktivita, která je ve frontě před nimi, právě běží. Aktivity můžeme dle životního cyklu rozdělit do tří stavů [36]:

- **Právě běžící aktivita**
  Je na počátku pomyslné fronty. Tato aktivita dostává informace o vstupech. S touto aktivitou právě pracuje uživatel [36].
- **Pozastavená aktivita**
  Aktivita může být pozastavena například z důvodu překrytí jinou, právě běžící aktivitou. Aktivita, která je pouze pozastavena, může být stále částečně vidět, ale nedostává informace ze vstupů [36].
- **Zastavená aktivita**
  Aktivita, která je zastavena, již není vidět, ale nedošlo zatím k jejímu zničení. Ke zničení zastavené aktivity nedochází z důvodu možnosti, že se k ní

uživatel bude chtít vrátit. Zastavená aktivita, stejně jako pozastavená, nepřijímá informace ze vstupu [36].

Obrázek 21 Životní cyklus aktivity



Zdroj: *Developer Android - Activity* [*36*].

Jednotlivé stavy popsány výše jsou realizovány pomocí metod životního cyklu aktivit.

Tabulka 1 Metod životního cyklu

onCreate()	Metoda je volána pro vytvoření aktivity. Zároveň se zde nastavuje načítání dat a vytváří pohledy.
onRestart()	Metoda je volána nad aktivitou, která byla pozastavena a má se znovu spustit.
onStart()	Metoda je volána v momentě, kdy se má aktivita stát viditelnou pro uživatele.
onResume()	Metoda je volána, když má aktivita přijímat informace od uživatele. Při volání této metody je aktivita první ve frontě.
onPause()	Metoda je volána, když má dojít k obnovení předchozí aktivity. Je doporučeno pozastavit animace, uložit dosud neuložená data a další procesy, které mohou využívat zdroje zařízení a tím jej zpomalovat.
onStop()	Metoda je volána, když aktivita již není viditelná pro uživatele.
onDestroy()	Metoda je volána před zničením aktivity. Buď z důvodu dokončení aktivity finish(), nebo ji systém sám zničí pro potřebu využití zdrojů zařízení.

Zdroj: *Vlastní zpracování pomocí MS Excel* [*36*].

5.3.2 Intent

Třída intent zprostředkovává přechod mezi aktivitami. Objekt třídy intent může obsahovat data, která předá cílovému komponentu. Intent by se dal považovat za most mezi jednotlivými prvky, ať již aktivitami, broadcasty, receivery nebo services. Existují dva typy intentů a to explicitní a implicitní [38].

- **Explicitní intenty -** Explicitní intenty jsou využívaný v případě, kdy přímo víme, kterou třídu spustit [38].
- **Implicitní intety -** Implicitní intenty jsou využívány pro všeobecnou akci. Tento typ umožňuje popsat záměr, ale nevyžaduje přesný způsob. Lze tedy říci, že pomocí implicitního intentu je možné říct aplikaci o otevření webové stránky a nemusí se nic víc řešit. Naopak u explicitního

intentu by bylo zapotřebí hledat například webový prohlížeč a další náležitosti pro otevření webové stránky [38].

5.3.3 Obsah kořenového adresáře

V této kapitole budou popsány důležité části kořenového adresáře, které bylo potřeba nějakým způsobem modifikovat. Nejsou zde zahrnuty adresáře, které byly automaticky vygenerovány a nebyly modifikovány.

- **AndroidManifest.xml** - Jedná se o soubor XML, který popisuje aplikaci a komponenty dodávané touto aplikací. Může se jednat například o služby nebo aktivity. Soubor v sobě obsahuje následující informace:
  - Veškerá oprávnění pro aplikaci
  - Uvádí se zde, jak jsou součásti aplikace propojeny s operačním systémem. Definuje se zde tedy například, jaká aktivita se má zobrazit v hlavní nabídce, neboli při spuštění
  - Dále je zde seznam všech komponentů
  - Názvy externích knihoven, které jsou v rámci aplikace využívány [37],[34]
- **app/src/main/java –** Obsahuje jednotlivé Java soubory, které jsou v aplikaci využívány.
- **App/src/main/res –** V tomto adresáři se nachází zdrojové soubory grafického typu, ale i XML soubory, které popisují jednotlivá tlačítka, textboxy a další elementy jednotlivých aktivit.
  - **App/src/main/res/drawable -** V této podsložce se nachází veškeré obrázky využívané v aplikaci.
  - **App/src/main/res/layout -** V této složce se nachází XML soubory, které ovlivňují uživatelské rozhraní. Mohou ovlivňovat například vzhled tlačítek, jejich rozměry a další [37].
  - **App/src/main/res/mipmap -** V mipmap je v našem případě uloženo pozadí aplikace. Všeobecně se zde ukládají například pozadí či ikony aplikace. V tomto projektu jsou ovšem ikony uloženy v drawable.

- **App/src/main/res/values -** Ve values se uchovávají řetězce. V příkladu této aplikace se zde uchovávají informace k menu pro *kraj, kategorii, okres*.

5.3.4 Oprávnění

Oprávnění stanovuje aplikaci, co může vykonávat. Toto oprávnění se nachází v souboru AndroidManifest.xml. Na vypsání oprávnění se používá segment <*uses-pemission*>, ve kterém je jediný atribut *android:name*. Tomuto atributu je poté možno přiřadit celou řadu oprávnění. Například pro povolení užívání internetu musí být v atributu uložen následující řetězec kódu [38].

```
„android.permission.INTERNET"
```

5.3.5 R.Java – získávání stylování

Soubor *R.java* je souborem struktury Android projektu. Tento soubor umožňuje dědit styl nastavený z XML souborů přímo ve zdrojovém kódu aktivity a je automaticky generován při založení projektu. Tento soubor obsahuje ID na všechny zdroje v rámci adresáře *res/* [43].

5.3.6 SharedPreferences – sdílena data

Android poskytuje spoustu možností jak ukládat data v aplikaci. Jednou z těchto možností je *Shared Preferences*. Na rozdíl od *Preferences* je toto sdílené ukládání dostupné v rámci celé aplikace. *Shared Preferences* umožňují ukládat data ve stylu název a hodnota. V aplikaci jsou použity názvy *login-name* a *login-pass*. K těmto dvěma názvům se přiřazují hodnoty uživatele a hesla [45]. Pro získání hodnot se používá následující segment kódu:

```
SharedPreferences myPrefs2 =
getApplicationContext().getSharedPreferences("login", 0);
String prefName = myPrefs2.getString("login-name","host");
String prefPass = myPrefs2.getString("login-pass","");
```

5.3.7 onItemSelected() – kontrola vybrané položky

Definovaná metoda rozhraní, která se volá pro zjištění, která položka byla vybrána v určitém zobrazení. Metoda je využívána pro kontrolu nad *spinnerem* označující kraj. Obsahuje následující parametry [46]:

- **Parent (rodič)** – AdapterView: kde se má odehrát výběr.
- **View (pohled)** – Pohled na AdapterView kam bylo kliknuto.
- **Position (pozici)** – Pozice pohledu v adapteru.

- **ID** – ID řádku položky, která byla vybrána.

5.3.8 Toast – zobrazení zpráv

Pro zobrazení zpráv, jak chybových, tak stavových, se využívá funkce toast. Tato funkce umožňuje zobrazit zprávu v malém vyskakovacím okně. Toto okno zabere automaticky vždy pouze tolik místa, kolik je zapotřebí pro jeho obsah. Vyskakovací okno automaticky zmizí po vypršení určité doby.

5.4 Vývoj aplikace - Praktická část

V této části je ukázán vývoj mobilní aplikace pro operační systém Android ve vývojovém prostředí Android Studio. Kapitola je rozdělena na podkapitoly dle toho, jaká aktivita se rozebírá.

5.4.1 Hlavní aktivita

Hlavní aktivita je obsažena v souboru *MainActivity.java*. Z hlavní aktivity je možné spustit tři vedlejší aktivity. Ke spuštění těchto vedlejších aktivit jsou zde tři tlačítka. Každé z těchto tlačítek obsahuje segment kódu s *intent* pro odkázání na patřičnou aktivitu, která se pod tlačítkem ukrývá.

Obrázek 22 Hlavní aktivita

Zdroj: *Vlastní zpracování pomocí Android Studio*

Ikony byly staženy ze stránky http://www.flaticon.com/ od těchto autorů.

- **Horní ikona lupy** – http://www.freepik.com/

- **Prostřední ikona přidání** – http://www.flaticon.com/authors/roundicons
- **Spodní ikona pro ostatní možnosti** - http://www.flaticon.com/authors/vectors-market

Při kliknutí na poslední tlačítko pro ostatní možnosti se kontrolují sdílená data. Tato sdílená data obsahují *login-name* a *login-pass*. Pokud je zde uživatelské jméno nastaveno na *host,* otevře tlačítko aktivity pro přihlášení. Pokud není uživatel *host,* otevře se aktivita, která nabízí odhlášení.

Součástí hlavní aktivity je také segment klauzulí *if,* které slouží pro kontrolu proměnné *hlaska-odeslani,* která je obdržená z aktivity pro vložení nového záznamu. Na základě této proměnné se vypíší informace o vložení.

5.4.2 Odesílání do API

Pro komunikaci s API se využívá soubor *ZdrojakCore.java*. Obsahuje metody pro funkcionality poskytované aplikací. Nejdříve je důležité nastavit správnou komunikační adresu na API. Vytvoření první části adresy *api.ztratose.cz* je možné vidět v kapitole 5.1 na obrázku [16]. Druhá část *v1/Data/Postzapsaniveci* je definována v samotném API.

```
private static final String ACTION =
"http://api.ztratose.cz/v1/Data/Postzapsaniveci";
```

Následně v souboru probíhá v každé metodě vytvoření připojení.

```
URL url = new URL(ACTION);
URLConnection connection = url.openConnection();
```

Po těchto krocích každá metoda převezme data na svém vstupu a z těchto dat složí výstup pro API ve formátu JSON. Vytvoření probíhá vytvořením objekt třídy *JSONObject*, do kterého se nahrají data pomocí

jsonobject.accumulate(nazev:hodnota).

Poté je celý JSON výstup vložen do proměnné a ta je odeslána na zpracování do API. Obdobný postup probíhá ve všech metodách souboru *ZtratoseCore*. Tyto metody jsou.

- **CallApi_Hledat –** Metoda využívaná v aktivitě pro vyhledávání.
- **CallApi_Pridat –** Metoda využívaná v aktivitě pro přidání nového záznamu.
- **CallApi_Login –** Metoda využívaná k provedení přihlášení.

- **CallApi_Registrovat -** Metoda pro registraci uživatele.

5.4.3 Vyhledávání záznamů

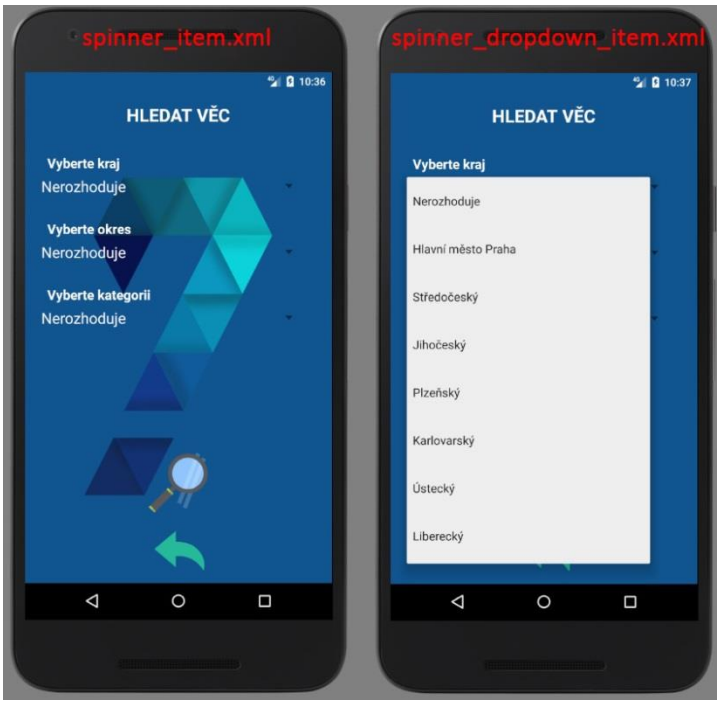
Vyhledávání probíhá na základě názvů kategorie, okres a kraj. Princip tohoto vyhledávání byl již popsán v kapitole o API 5.2.12. Přístup k vyhledávání je z hlavní aktivity v souboru *MainActivity*. Je zde nastaven *intent* na patřičnou aktivitu, která provádí vyhledávání.

```
public void buttonSearch_onClick(View e) {
    Intent intent = new Intent(this,SearchActivity.class);
    startActivity(intent);
}
```

Hodnoty pro vyhledávání jsou uložené ve *spinnerech* a jsou v aplikaci stanoveny staticky v adresáři *app/src/main/res/values/styles.xml*. *Spinnery* přebírají pomocí R.java svoje styly ze souborů.

- *spinner_item.xml*, tento soubor definuje stylování, pokud *spinner* není rozkliknut. Je zde nastavena barva textu na bílou, jelikož je pozadí aplikace tmavé
- *spinner_dropdown_item.xml*, zde je naopak nastavena barva textu na černou, jelikož samotný *spinner* má po rozkliknutí bílé pozadí

Obrázek 23 Styl spinnerů vyhledávání



Zdroj: *Vlastní zpracování pomocí Android Studio*

V souboru *styles.xml* jsou definované hodnoty pro *spinnery*, a to jak hodnoty pro kraj a kategorii, tak pro okres. Na následující ukázce segmentu je pole okresů pro Karlovarský kraj. Pro každý kraj je zde obdobné pole jeho okresů.

```
<string-array name="okres_Karlovarsky">
    <item>Nerozhoduje</item>
    <item>Cheb</item>
    <item>Karlovy Vary</item>
    <item>Sokolov</item>
</string-array>
```

V aktivitě pro vyhledávání probíhá mimo jiné i nastavení *spinneru* pro okres na základě vybraného kraje. Do aktivity je nejdříve nahrán *spinner* pro kraj a následně i pro okres. Pro každý z krajů je zde vytvořen *adapter* v rámci *ArrayAdapter*, který je nastaven na patřičný zdroj dat pro *spinner* okres. Zjištění, který kraj je právě aktivní, probíhá pomocí *onItemSelected()*. Tato funkce získá pozici vybraného kraje a předá ji *switchi*, který má jednotlivé kraje uložené. Díky tomu se rozpozná, který *adapter* se má použít pro *spinner* okresů.

Po kliknutí na tlačítko pro provedení vyhledávání se získaná data ze *spinnerů* nahrají do proměnných. Následně se volá metoda pro hledání ze souboru *ZtratoseCore*, která převezme patřičná data a vytvoří JSON, který odešle do API. Zpět dostane z API JSON s výsledky nebo chybovou zprávu pro vypsání chybového stavu. JSON s výsledky se předává aktivitě pro vypsání dat.

```
Intent intent = new Intent(this,NalezenoActivity.class);
intent.putExtra("json-object",data);
startActivity(intent);
```

5.4.4 Vypsání nalezených záznamů

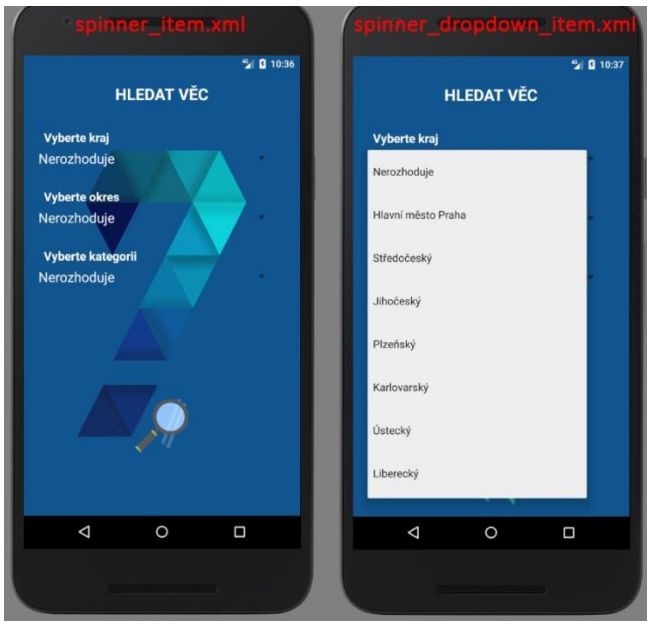
Aktivita převezme JSON s výsledky od aktivity s vyhledáváním. Tento JSON zkontroluje, zda není prázdný. Pokud ano, zobrazí se pouze jedna položka se zprávou, že se pod danými kritérii nenachází žádná data.

```
list.add(new ListRow("","","","","Nejsou žádné nalezené
věci!",""));
```

Tato aktivita má nastavený svůj vzhled jako listview, který ovšem není vhodný pro vypsání výsledných dat. Bylo zapotřebí zajistit, aby každá položka v listview byla vypsána jako tabulka několika položek. Je zde tedy vytvořen objekt adapter třídy *ItemAdapter*, který definuje šablonu pro položky listview

[43]. Toto probíhá v rámci *onCreate()*. Následuje metoda, která přijatý JSON zpracuje a rozdělí jej na potřebné části. Tyto části jsou následně vkládány do listview s využitím stylu šablony z *ItemAdapter*. Na obrázku [24] je zobrazeno v levé části, jak vypadá nastavený listview a v pravé části, jak vypadá šablona v *IntemAdapter* pro jednotlivé položky z listview.

Obrázek 24 Aktivita vypsání nálezů



Zdroj: *Vlastní zpracování pomocí Android Studio*

5.4.5 Přihlášení

Aktivita přihlášení je dostupná z hlavní aktivity. Tato aktivita se zobrazí po kliknutí na tlačítko, pokud uživatel není přihlášen. Nacházejí se zde tři tlačítka a dvě textová pole pro jméno a heslo. Každé z tlačítek spouští další aktivitu.

- **Tlačítko 1** - Nachází se v horní části a slouží pro vypsání části občanského zákoníku, která se zabývá problematikou nálezu.
- **Tlačítko 2** – Tlačítko pro přihlášení.
- **Tlačítko 3** - Tlačítko pro registraci.

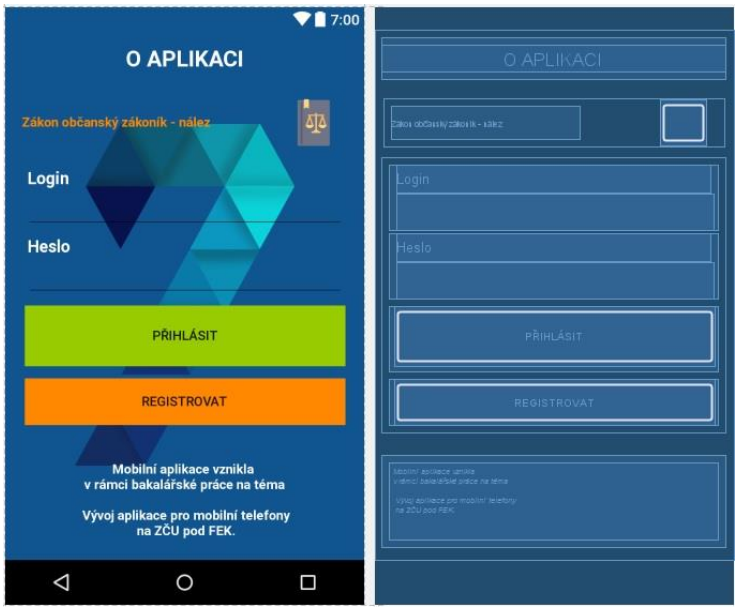
U *Tlačítka 2* se nejdříve převezmou hodnoty z textových polí pro jméno a heslo. Dochází ke kontrole, zda nejsou textová pole prázdná. Pokud ano, zobrazí se zpráva o chybě. Jsou-li pole naplněná, volá se metoda pro přihlášení ze souboru *ZtratosCore*. Této metodě jsou předány vyplněné údaje, které jsou odeslány ve formátu JSON na zpracování do API. Na základě odpovědi z API se vypíše uživateli buď zpráva o chybě, nebo je přesměrován na plochu pro

přihlášené uživatele, která je představena v kapitole 5.4.7. Zároveň se do sdílených dat uloží přihlašovací údaje, aby mohl uživatel vkládat věci pod svým jménem. Důležitým krokem je segment kódu

```
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK),
```

který vymaže historii průchodů aplikací. Kdyby tato část kódu neproběhla, uživatel by musel pro vypnutí tlačítkem zpět na svém zařízení procházet v aplikaci zpět tak dlouho, dokud by nedošel v historii průchodu na začátek.

Obrázek 25 Přihlášení



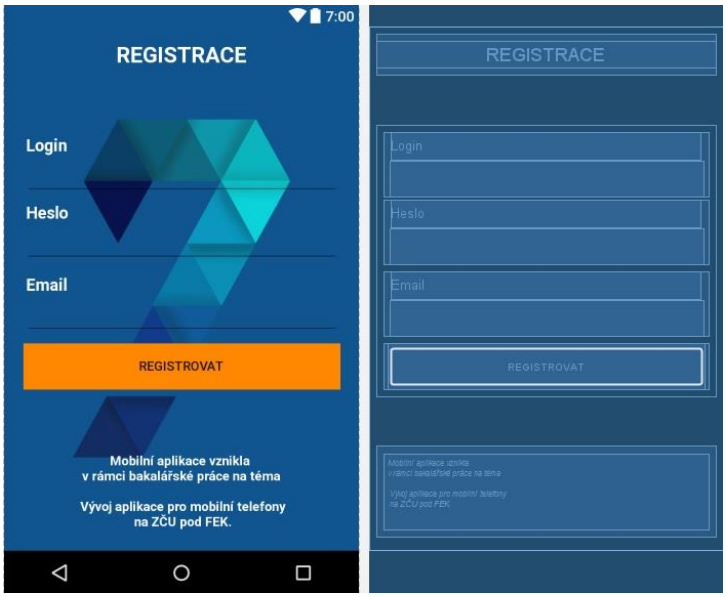
Zdroj: *Vlastní zpracování pomocí Android Studio*

5.4.6 Registrace

Plocha registrace obsahuje tři textová pole pro jméno, heslo a email. Dále obsahuje jedno tlačítko pro provedení registrace. Po stisknutí tlačítka se nahrají hodnoty z polí do proměnných. Proběhne kontrola v rámci aktivity, zda jsou jednotlivá pole vyplněná. Pokud ano, volá se metoda pro registraci ze souboru *ZtratoseCore*. V této metodě proběhne složení JSON výstupu z aplikace a odeslání na zpracování do API. Následně metoda obdrží od API JSON vstup. Tento vstup předá zpět do aktivity pro registraci. V aktivitě pro registraci se JSON vstup vyhodnotí, zda proběhl v pořádku nebo zda je zapotřebí ukázat chybovou zprávu. Pokud je vše v pořádku, JSON vstup se zpracuje. Jméno a heslo je uloženo do sdílené proměnné a uživatel je stejných způsobem jako po

úspěšném přihlášení odkázán na plochu pro již přihlášené uživatele. Než je uživatel odkázán, probíhá zde smazání historie průchodu aplikací.

Obrázek 26 Registrace



Zdroj: *Vlastní zpracování pomocí Android Studio*

5.4.7 Přihlášený uživatel

Tato aktivita slouží pro informování uživatele, že je přihlášen. Nyní budou veškeré příspěvky vkládány pod jeho jménem. Na této aktivitě se nachází *textView,* který informuje uživatele pod jakým jménem je přihlášen. Dále je zde tlačítko, které umožní uživateli odhlásit se ze svého účtu a vkládat data pod anonymním uživatelem *host*. Po kliknutí na tlačítko pro odhlášení se nastaví hodnoty sdílených proměnných *login-name* a *login-pass* zpět na uživatele *host*.

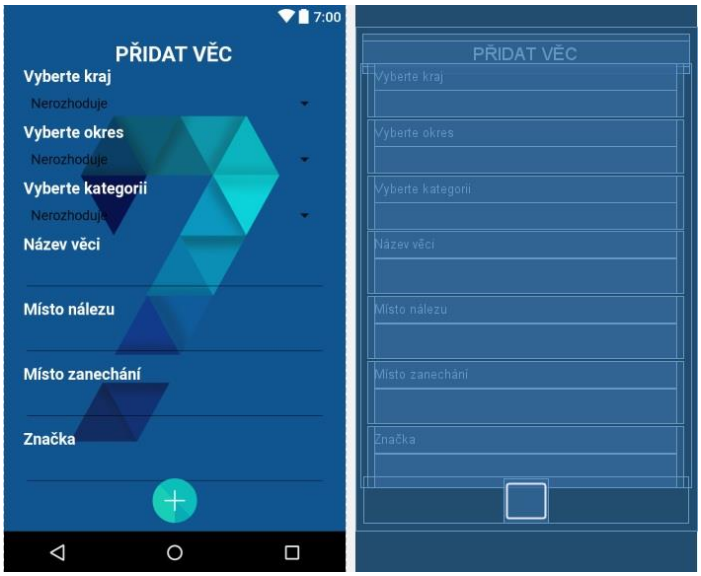
Obrázek 27 Přihlášený uživatel

Zdroj: *Vlastní zpracování pomocí Android Studio*

5.4.8 Vkládání nových záznamů

Aktivita pro vkládání nových záznamů obsahuje trojici *spinneru* kraj, okres a kategorie. *Spinnery* fungují identicky jako v aktivitě pro vyhledávání, viz kapitola 5.4.3. Dále jsou zde textová pole pro hodnoty názvu věci, místa nálezu, místa zanechání a značky věci. Aktivita si na stisknutí tlačítka vložit převezme z elementu hodnoty a uloží je do proměnných. Následně zkontroluje, zda se ve sdílených datech *login* na pozicích *login-name* a *login-pass* nachází uživatel *host*. Pokud ano, nastaví se proměnné pro uživatele a heslo na uživatele host. Pokud se ve sdílené proměnné nenachází host, nastaví se proměnné na hodnoty pod *login-name* a *login-pass*. Všechna data jsou poslána do metody pro vkládání nových věcí v souboru *ZtratoseCore*. Zde jsou data zpracována do formátu JSON a odeslána do API pro zpracování. Z API se obdrží zpráva o vložení, na základě které se uživateli zobrazí zpráva s chybou nebo s informací o úspěšném vložení.

Obrázek 28 Přidání věci



Zdroj: *Vlastní zpracování pomocí Android Studio*

5.4.9 Vypsání zákonů

Na aktivitu, která v sobě obsahuje aktuální zákon, který se zabývá problematikou nálezu věci, je možné se dostat pomocí tlačítka na ploše pro přihlášení, registraci a na ploše pro již přihlášené uživatele. Tlačítko se vždy nachází v horní části obrazovky, viz obrázek [24]. Zdroj ikony pro vypsání zákonů je dostupný z následující stránky: http://www.flaticon.com/free-icon/law_186327#term=law&page=1&position=70.

Tato aktivita obsahuje pouze jeden *litview,* který je nastavený na zdroj ze souboru *styles.xml*.

6 Pilotáž, testování a návrhy změn pro aplikaci

6.1 Testování

V této kapitole bude představeno, jak probíhalo testování různých částí projektu.

6.1.1 Testování API

Testování bylo realizováno v rámci vývoje. Díky postupnému rozšiřování a zlepšování API je tato metoda nejpříjemnější pro vývojáře. Vývoj a testování API probíhal následovně:

1. Testování předávání řetězců mezi aplikací a API.
2. Testování čtení a zápisu z databáze.

3. Rozšíření API o další funkcionality, jako vkládání nového záznamu. API bylo v tuto chvíli velmi statické a vstup musel být napsán přesně tak, jak jej API očekává.
4. Po vytvoření každé funkcionality jsem se zaměřil na její testování a zkoušení různých vstupů, jak platných, tak neplatných.
5. Přechod na dynamické čtení vstupu JSON a dynamické testování vstupu.
6. Testování dynamických úprav v metodách.
7. Rozšíření o přihlášení.
8. Testování přihlášení.
9. Testování funkcionalit po přihlášení.
10. Celkové testování všech funkcionalit.

6.1.2 Uživatelské testování aplikace - Pilotáž

První uživatelské testování proběhlo svépomocí. V tomto testování jsem se snažil přemýšlet jako uživatel, který aplikaci vidí poprvé v životě a nemá o aplikaci žádné informace. Celé toto testování jsem ze svého pohledu uchopil jako testování aplikace velmi nedisciplinovaným uživatelem.

Dále jsem nechal aplikaci otestovat několika dalšími uživateli ze dvou skupin. První skupinu tvořili uživatelé, kteří znali záměr této aplikace a druhou skupinu tvořili uživatelé, kteří nevěděli, co má aplikace vykonávat nebo k čemu slouží. U první skupiny, která znala účel aplikace, byl průchod aplikací plynulý. Druhá skupina díky nadpisu na úvodní ploše ihned zjistila účel aplikace. Díky všeobecné znalosti principu ztrát a nálezů uživatelé tušili, co mohou přibližně od aplikace očekávat. V obou případech bylo několik připomínek na rozšíření funkcionalit nebo vylepšení používané aplikace. Snažil jsem se od uživatelů získat názor týkající se jak vzhledové stránky, tak funkcionalit. Tyto a další návrhy jsou zmíněny v následující kapitole 6.2.

6.2 Návrhy změn

Návrhy změn zde vypsané se týkají nejen rozšíření funkcionalit aplikace, ale také API. Cílem této kapitoly je stanovit potřebná rozšíření a vylepšení do budoucna, a to jak z pohledu uživatele, tak z pohledu vývojáře, který ví o vnitřních problematikách.

- **Zlepšení vyhledávání** – Část uživatelů by uvítala přidávání místa nálezu a místa zanechání pomocí našeptávače adresy. Díky tomu by bylo zajištěno, že bude vložena existující adresa. Tato informace by se dále dala použít například k navigaci na místo zanechání ztracené věci.
- **Změna ikony pro přihlášení** – Uživatelé by uvítali změnu designu třetího tlačítka. Tlačítko slouží na přesun do přihlášení a registrace nebo na přístup k tlačítku pro vypsání zákonů. Ikona použita u tohoto tlačítka ovšem všeobecně značí určité nastavení aplikace. Uživatelé očekávali pod tímto tlačítkem jiné funkcionality.
- **Oznámení o smazání –** V budoucnu bude nutné dodělat propojení API se službou POP3 nebo IMAP, která zašle zaregistrovaným uživatelům email, pokud jejich záznam přesáhl dobu pro uchování záznamu a byl vymazán.
- **Možnost smazat svůj záznam –** Přihlášení uživatelé by uvítali možnost smazat svůj záznam sami.
- **Možnost navigace –** Při ztrátě věci v cizím městě by uživatelé uvítali možnost navigace na místo, kam byla věc odevzdána. Podmínkou této funkcionality je zadávání místa nalezu pomocí našeptávače.

Na základě připomínek od uživatelů bylo rozšířeno API o další funkcionality, které v rámci bakalářské práci již nebyly implementovány a vytvořeny v mobilní aplikaci. Jedná se o vypsání záznamu, které vložil přihlášený uživatel, na základě kterých je možné záznamy změnit a smazat. Aplikace zatím nebyla nikde zveřejněna. Nejdříve je důležité aplikaci dokončit i se všemi doplňkovými funkcemi, aby byla schopna obstát na trhu. Zároveň se jedná o originální aplikaci, kde by při zveřejnění v nedokončeném stavu mohla hrozit hrozba ze strany konkurence, která by mohla převzít nápad na aplikaci a dokončit jí v plné podobě dříve.

Závěr

Hlavním cílem této práce bylo vytvořit aplikaci, která by pomohla vyřešit problematiku ztrát a nálezů.

Byl proveden průzkum trhu na již existující aplikace a webové služby na téma ztrát a nálezů. Následně bylo definováno předpokládané využití aplikace, její funkcionality a možnosti uživatelského rozhraní.

Proběhlo srovnání různých mobilních platforem. U každé platformy byly uvedeny podmínky pro publikování, možnosti vývoje a výhody či nevýhody z pohledu uživatelů a vývojářů. Byl proveden výběr operačního systému Android pro vývoj aplikace. Dále proběhlo seznámení s oficiálním vývojovým prostředím pro Android a to Android Studio. Byly zde nastíněny i další možnosti vývoje. Na základě těchto informací bylo vybráno Android Studio jako vývojové prostředí pro tvorbu aplikace.

Došlo k vytvoření loga aplikace, návrhu uživatelského layoutu, UML diagramu a relačního modelu pro databázi, na základě kterého byla vygenerována databáze.

Implementaci mobilní aplikace předcházelo seznámení s API. Důraz byl kladen na vysvětlení principů fungování v API, na které je následně odkazováno v části vývoje mobilní aplikace. Při popisu samotného vývoje mobilní aplikace byly nejdříve vysvětleny potřebné teoretické znalosti. Následně bylo popsáno fungování jednotlivých aktivit a jejich vzájemné propojení.

Navržené API i mobilní aplikaci jsem průběžně testoval. Po dokončení vývojářských testů byla provedena pilotáž mezi vybrané uživatele. Závěrem lze tedy říci, že byla vytvořena mobilní aplikace, která řeší problém ztrát a nálezů.

Seznam grafů

Seznam tabulek

Seznam obrázků

Seznam použitých symbolů a zkratek

GPS - Globální polohový systém

OS – Operační systém

HTML 5 - HyperText Markup Language

C# - Objektově orientovaný programovací jazyk

VB.NET - Objektově orientovaný jazyk

C++ - Multiparadigmatický programovací jazyk

JDK - Java Development Kit

SDK - Software development kit

BIOS - Basic Input-Output Systém

UML - Unified Modeling Language

SQL - Structured Query Language

API - Application Programming Interface

DNS - Domain Name Systém

IP – Internet protocol

JSON - JavaScript Object Notation

XML - Extensible Markup Language

ASP- Active Server Pages

POP3 – Post Office Protocol

IMAP – Internet Message Access Protocol

Seznam použité literatury

1. Statista – *Number of mobile phone users wordlwide from 2013 to 2019 (in billions)* [Online] [Citace 1.4.2017] https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/
2. Podnikatel – *Nový občanský zákoník : Nález* [Online] [1.4.2017] http://www.podnikatel.cz/zakony/novy-obcansky-zakonik/f4582281/#p1057
3. Statista – *Global mobile OS market share*. [Online] [Citace 10.12.2016] https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/
4. LACKO, Ľuboslav. *Vývoj aplikací pro Windows 8.1 a Windows Phone*. Brno: Computer Press, 2014. ISBN 978-80-251-3822-9.
5. Wikipedie – *Windows phone* [Online] [Citace 10.12.2016] *https://cs.wikipedia.org/wiki/Windows_Phone*
6. MobilMania – *Windows 10 jeden systém vládne všem* [Online] [Citace 10.12.2016] http://www.mobilmania.cz/windows-10-mobile-jeden-system-vladne-vsem-recenze/a-1333451/default.aspx
7. Statista – *number of apps available in leading app stores* [Online] [Citace 10.12.2016] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

8. Msdn.Microsoft – *Account types,locations, and fees* [Online] [Citace 10.12.2016] https://msdn.microsoft.com/en-us/windows/uwp/publish/account-types-locations-and-fees
9. Zive – *Project Siena: Vytvořte si sami aplikaci pro Windows* [Online][Citace 1.4.2017] http://www.zive.cz/clanky/project-siena-vytvorte-si-sami-aplikaci-pro-windows/sc-3-a-172300/default.aspx
10. Wikipedie – *Android* [Online] [Citace 12.12.2016] https://cs.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%AD_syst%C3%A9m%29
11. Wikipedie - *Historie verzí Android* [Online] [Citace 12.12.2016] https://cs.wikipedia.org/wiki/Historie_verz%C3%AD_Android
12. Androworks – Vyzkoušejte Android mobil [Online] [Citace 12.12.2016] http://www.androworks.org/operacni-system-android/
13. Engadget – *Lolipop becomes the most popular version of Android* [Online] [Citace 12.12.2016] https://www.engadget.com/2016/03/08/lollipop-leads-android-usage-share/
14. AppStorm - *15 Apps for programming on Android* [Online] [Citace 12.12.2016] http://android.appstorm.net/roundups/developer/15-apps-for-programming-on-android/
15. Developer.Android – *Get Started with publishing* [Online] [Citace 12.12.2016] https://developer.android.com/distribute/googleplay/start.html
16. Android – *Menu: Browse devices* [Online][Citace 1.4.2017] https://www.android.com/
17. LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.
18. Wikipedie –*Ios (Apple)* [Online] [Citace 5.12.2016] https://cs.wikipedia.org/wiki/IOS_(Apple)
19. Wikipedie – *Swift (programovací jazyk)* [Online] [Citace 5.12.2016] https://cs.wikipedia.org/wiki/Swift_(programovac%C3%AD_jazyk)
20. Bluecloudsolution - *develop apps on windows pc* [Online] [Citace 5.12.2016] http://www.bluecloudsolutions.com/blog/develop-apps-on-windows-pc/
21. Developer Apple – *app store/review/guidelines* [Online] [Citace 5.12.2016] https://developer.apple.com/app-store/review/guidelines/
22. LetemSvetemapplem – *9 důvodů, proč je iPhone lepší než telefony s Androidem* [Online] [Citace 4.4.2017] https://www.letemsvetemapplem.eu/2015/02/04/9-duvodu-proc-je-iphone-lepsi-nez-telefony-s-androidem/
23. Androidauthority – *Google Play Store vs Apple App store: by the numbers (2015)* [Online][Citace 4.4.2017] http://www.androidauthority.com/google-play-store-vs-the-apple-app-store-601836/
24. Wikipedie – *Vývojové prostředí* [Online][Citace 7.4.2017] https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD

25. Zdrojak – *Android Studio – nové vývojové prostředí* [Online][Citace 5.4.2017] https://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/
26. Tutorialspoint – *Android Basics: Hello World Example* [Online][Citace 2.4.2017] https://www.tutorialspoint.com/android/android_hello_world_example.htm
27. Itnetwork – *Adnroid programování - vývojové prostředí* [Online][Citace 2.4.2017] http://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-vyvojove-prostredi
28. JSON- *Úvod do JSON* [Online][Citace 13.4.2017] http://www.json.org/json-cz.html
29. Wikipedie – *API* [Online][Citace 13.4.2017] https://cs.wikipedia.org/wiki/API
30. Microsoft Visual Studio – *Documentation* [Online][Citace 13.4.2017] https://msdn.microsoft.com/en-us/library/dd831853(v=vs.110).aspx
31. Microsoft Visual Studio – *Web.config* [Online][Citace 14.4.2017] https://msdn.microsoft.com/en-us/library/aa306178.aspx
32. Manualy – *Teorie relačních databází: Normalizace* [Online][14.4.2017] http://www.manualy.net/article.php?articleID=13
33. Stahnu – *Adobe photoshop CS6* [Stažení][15.3.2017] https://stahnu.cz/bitmapove-editory/adobe-photoshop/download/65220
34. ALLEN, Grant. *Android 4: průvodce programováním mobilních aplikací*. Brno: Computer Press, 2013. ISBN 978-80-251-3782-6.
35. Developer Android – *Start Another Acitivity* [Online][17.4.2017] https://developer.android.com/training/basics/firstapp/starting-activity.html
36. Developer Android – *Acitivity* [Online][17.4.2017] https://developer.android.com/reference/android/app/Activity.html
37. Developer Android – *Acitivity* [Online][17.4.2017] https://developer.android.com/guide/topics/manifest/manifest-intro.html
38. Zdrojak – *Vyvíjíme pro Android: Intenty, intent filtry a pesmission* [Online][17.4.2017] https://www.zdrojak.cz/clanky/vyvijime-pro-android-intenty-intent-filtry-a-permissions/
39. WOLBER, David. App inventor. Brno: Computer Press, 2014. ISBN 978-80-251-4195-3.
40. Wikipedie – *Vývojové prostředí* [Online][10.4.2017] https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%A9_prost%C5%99e5%99ed%C3%AD

42. Stackoverflow – *What is the R.java file in Android Studio*

[Online][Citace 17.4.2017]
http://stackoverflow.com/questions/28522144/where-is-the-r-java-file-in-android-studio
43. StackOverflow – *Android Custom row for listview* [Online][18.4.2017] http://stackoverflow.com/questions/15832335/android-custom-row-item-for-listview
44. Flaticon – *Law Free Icon* [Zdroj][17.4.2017]
45. Developer Android – *Shared Preferences* [Online][16.4.2017] https://developer.android.com/reference/android/content/SharedPreferences.html
46. Developer Android – *AdapterVied.OnItemSelectedListener* [Online] [16.4.2017] https://developer.android.com/reference/android/widget/AdapterView.OnItemSelectedListener.html

Příloha A

- Přiložené CD
  - Zdrojový kód API
  - Zdrojový kód testovací aplikace na API
  - Soubor se vstupy pro testovací aplikaci
  - Mobilní aplikaci ve formátu apk
  - Projekt mobilní aplikace pro Android Studio

Abstrakt

Zdeněk Borský, *Vývoj aplikace pro mobilní telefony*. Bakalářské práce. Plzeň: Západočeská univerzita v Plzni. Fakulta ekonomická. 69s, 2017

Klíčová slova: Vývoj aplikací, Android, ztráty a nálezy, ztrátoš, mobilní aplikace.

Tato bakalářská práce se zabývá vývojem mobilní aplikace na principu ztrát a nálezů. Práce se skládá z části teoretické a praktické. V teoretické části se porovnávají vybrané platformy, publikování aplikace pro vybrané platformy a možnosti vývoje pro jednotlivé platformy. Jsou zde zmíněny výhody a nevýhody jednotlivých platforem z pohledu vývojáře i uživatele. Je zde popsán výběr vhodné platformy a seznámení s vybraným vývojovým prostředím. V praktické části, je vytvořeno logo aplikace, návrh layoutu, relační model aplikace a UML diagram aplikace. Dále je zde popsán postup zprovoznění webhostingu a domény. Vývoj API pro vytvořenou aplikaci je umístěn do vývojového prostředí Microsoft Visual Studio a vývoj samotné aplikace do vývojového prostředí Android Studio. V závěru práce je shrnuto, jak probíhalo testování a náležitosti a jaké aspekty je zapotřebí v rámci API i uživatelského prostředí vylepšit.

Abstract

Zdeněk Borský, Mobile application development. Bachelor thesis. Pilsen: University of West Bohemia. Faculty of Economics. 69p, 2017

Key words: application development, operating system Android, lost and found, smartphone application

This bachelor thesis is occupied with a development of a smartphone application based on the lost-and-found principle. The thesis is composed of a theoretical and a practical part. In the theoretical part, there are compared the picked platforms, publications relating to the picked platforms and the development options for individual platforms. In this part, benefits and drawbacks of the individual platforms from the developer´s and the user´s point of view are mentioned. There are described a theoretical selection of a suitable platform and an introduction to a selected development environment. In the practical part, a logo, a relational model and a UML diagram of the

application have been created. Furthermore, procedures of setting up a webhosting and a domain are described in this part. A development of an API for the created application is placed in the Microsoft Visual Studio development environment and a development of the application in the Android Studio