Studentská tvůrčí a odborná činnost - 2023

Mobilní aplikace pro rozpoznávání zvířat v zoologických zahradách

Jiří Daberger

ÚVOD

Chytré mobilní telefony a jejich aplikace jsou trendem dnešní společnosti a většina z nás, si život bez chytrého mobilního telefonu nedokáže ani představit. Dnešní telefony již neslouží pouze jako prostředek ke komunikaci s našimi přáteli pomocí telefonování a SMS, ale zejména k využívání chytrých aplikací, které mohou udělat náš život jednodušší. Tímto posláním se zabývá i aplikace vzniklá v této bakalářské práci s důrazem na usnadnění návštěv zoologických zahrad formou interaktivního pozorování zvířat pomocí mobilního fotoaparátu a moderních technologií dnešního světa.

Výsledná aplikace je vytvořena za pomoci umělých neuronových sítí, které jsou díky své schopnosti „trénování“, vhodná pro řešení komplikovaných úloh v oblastech, jako je například klasifikace obrazových dat. Důležitou částí práce byl vývoj mobilní aplikace, jakožto prostředek pro detekci zvířat s využitím naučeného modelu konvoluční neuronové sítě za použití vlastního trénovacího datasetu.

Praktická část začíná návrhem mobilní aplikace a jejími funkcionálními a nefunkcionálními požadavky. Z návrhu přejdeme k samotnému vývoji aplikace, kde se seznámíme s použitými technologiemi. Zde nás dále čeká proces vývoje náhledu kamery v aplikaci, ukládání a načítání dat zvířat na jejich obrazovce detailu. Dozvíme se, k čemu můžeme v takové aplikaci využít polohovací technologii GPS a co čeká uživatele při objevování zvířat a zoologických zahrad. Od vývoje aplikace přejdeme k vývoji datasetu, bez kterého by aplikace nemohla ani vzniknout. Dále si představíme možnost trénování modelu konvoluční neuronové sítě pro rozpoznávání zvířat z vytvořených datasetů s detailem na finální testování těchto modelů.

Obrázek 1. QR kód s odkazem na stažení aplikace z obchodu Google Play.

1 NÁVRH MOBILNÍ APLIKACE

V začátku vývoje mobilní aplikace bylo zapotřebí vytvořit základní návrh celkové aplikace s otázkou, jaké funkcionální požadavky bude obsahovat a jak bude jejich funkčnost provedena. Hlavní myšlenkou bylo vytvořit mobilní aplikaci, která by obsahovala seznam několika zvířat a možnost některé z nich identifikovat za pomocí mobilní kamery přímo v dané aplikaci s důrazem na pohodlné a intuitivní ovládání.

Postupem času se aplikace rozrůstala o nové funkce přidávající uživateli větší přehled a bližší seznámení se zvířaty a zoologickými zahradami v České republice.

1.1 Funkcionální požadavky

Mezi funkcionální požadavky můžeme zařadit to, co se od aplikace jako takové očekává. Většina požadavků byla určena při prvotním návrhu aplikace, ale podstatná část nových požadavků byla doplněna později již během samotného vývoje aplikace.

- Systém aplikace umožňuje zobrazit seznam zvířat rozdělených do následujících kategorií: Savci, Ptáci, Plazi
- Systém aplikace umožňuje zobrazení informací o jednotlivém zvířeti
- Systém aplikace umožňuje povolení kamery mobilního zařízení pro možnost zobrazení jejího náhledu
- Systém aplikace umožňuje uživateli přibližovat a oddalovat náhled kamery pomocí gesta na obrazovce nebo posuvníkem
- Systém aplikace je schopen rozpoznat několik druhů zvířat z náhledu mobilní kamery a případně rozpoznávání i pozastavit
- Systém aplikace je schopen během náhledu fotoaparátu vytvořit snímek a uložit jej do mobilního zařízení s možností tento snímek odstranit do určitého času od vyfocení
- Systém aplikace je schopen rozpoznat několik druhů zvířat z vybrané fotografie uložené v mobilním zařízení
- Systém aplikace umožňuje vyhledat zvířata na základně jejich jména
- Systém aplikace umožňuje zobrazit zoologické zahrady v sekci „Objevy“ s možností filtrování Vše/Objevené/Neobjevené a jejich status objevení
- Systém aplikace umožňuje uložit „objevení“ zoologické zahrady do úložiště v aplikaci
- Systém aplikace umožňuje načtení informací ohledně časové dostupnosti zoologické zahrady pomocí internetu a Places API a zobrazení její polohy na mapě pomocí Maps API od společnosti Google

- Systém aplikace umožňuje podobně jako u zoologických zahrad filtrovat a ukládat „viděné“ zvířata v sekci „Objevy“ aplikace
- Systém aplikace umožňuje odstranit zoologickou zahradu nebo zvíře z „objevů“
- Systém aplikace umožňuje zobrazení informací o aplikaci obsahující text a seznam odkazů na stránky, které poskytly aplikaci cenná data během vývoje
- Systém aplikace umožňuje zobrazit různé informace o zvířatech z konkrétních zoologických zahrad v detailu zvířete v sekci „V zoo“
- Systém aplikace umožňuje po povolení lokace získat pomocí internetu a GPS aktuální pozici uživatele, a tak nastavit nejbližší zoologickou zahradu v sekci „V zoo“ u informací zvířete
- Systém aplikace umožňuje během prvotního zapnutí aplikace zobrazit uvítací obrazovku s několika informacemi na použití aplikace

1.2 Nefunkcionální požadavky

Do nefunkcionálních požadavků můžeme zařadit spíše technické specifikace a vlastnosti aplikace, které nejsou spojeny přímo s uživatelem aplikace, ale právě s vývojem aplikace, výběrem technologií, určení kompatibilních zařízení, stabilitou a například i bezpečností při jejím používáním.

1.2.1 Kompatibilita

Jedním z hlavních nefunkcionálních požadavků je minimální verze systému Android, na kterém je aplikace podporována. Jelikož byl pro programování vybrán jazyk Kotlin spolu s frameworkem Jetpack Compose, nezbývalo nic jiného, než začít na nejnižším podporovaném SDK levelu Jetpack Compose 21. Postupně ale byla potřeba minSdk zvýšit na level 24, jehož využití ale ve světě bezproblémově přesahuje 96 %, takže se nejednalo o závažný problém. Tím bylo zajištěné široké spektrum kompatibilních zařízení, na kterých může být aplikace spuštěna. UI design je optimalizován spíše na mobilní zařízení, ale díky responzivitě frameworku Jetpack Compose je možné aplikaci ovládat i na tabletech s větší velikostí displeje bez nutnosti vytvoření nového designu.

1.2.2 Forma ukládání dat

Jedním z dalších konkrétních nefunkcionálních požadavků při vývoji aplikaci bylo zajistit, aby veškeré texty aplikace byly uloženy prostřednictvím Android **Resources Values** a tím vytvořit ucelenou strukturu všech textů v jednotném jazyce. Díky tomu je možné jednoduše vytvořit překlady, a tak aplikaci nabídnout celému světu.

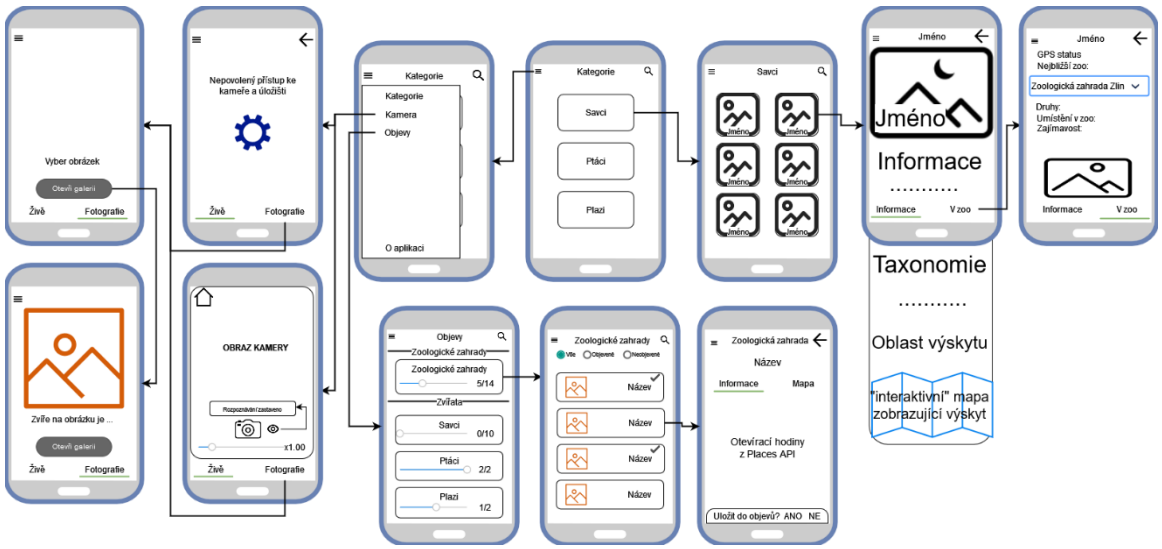
Pro ukládání dat, jako je například status objevení zoologické zahrady nebo zvířete, je v aplikaci využité jednoduchého API **SharedPreferences**.

Toto API dovoluje uložení tzv. *key-value* párových hodnot do úložiště aplikace, které je zabezpečené před dostupností v jiných aplikací.

1.3 Wireframe aplikace

Celá aplikace obsahuje několik oken, které jsou mezi sebou propojeny navigačním panelem, který je umístěn pod tlačítkem v levém horním rohu, popřípadě je dostupný táhnutím prstu zleva doprava kdekoliv na obrazovce. Rozvržení a základní podoba oken aplikace byla vytvořena webovým nástrojem draw.io dostupným na následující adrese: https://app.diagrams.net/.

Tento návrh hrál důležitou roli během reálného vývoje aplikace, jelikož autorovi poskytoval potřebnou předlohu pro naprogramování UI.



Obrázek 2. Návrh drátového modelu aplikace.

Na obrázku výše (Obrázek 2) můžeme vidět poslední verzi drátového modelu obsahující téměř všechny dostupná okna v reálné aplikaci, která byla na základě tohoto vytvořena. Startovací obrazovkou je obrazovka označena názvem „Kategorie“ uprostřed horní řady. Tato obrazovka je spuštěna po startu aplikace a uživatel se má možnost pomocí navigace dostat na kteroukoliv jinou obrazovku v aplikaci.

Tento wireframe neobsahuje uvítací obrazovku, jelikož ta byla vytvořena téměř v samotném závěru programování aplikace a jelikož je její UI stavba poměrně nenáročná, obešla se bez dodatečného návrhu.

2 VÝVOJ MOBILNÍ APLIKACE

Samotný vývoj aplikace může začít až po jejím, alespoň základním, návrhu, zvolením platformy
a výběru architektury spolu s programovacím jazykem.

Celý vývoj a testování aplikace probíhalo na fyzickém zařízení Xiaomi Mi 9T s operačním systémem Android ve verzi 11 a rozlišením obrazovky 1080x2340.

2.1 Zvolené technologie

Mobilní aplikace tedy vznikla pro mobilní platformu Android použitím programovacího jazyka Kotlin ve verzi 1.7.10. Pro tvorbu UI byl použitý framework Jetpack Compose, který byl vždy postupně aktualizován, jelikož je stále ve vývoji. Na samotném začátku byl jako grafický standard použitý Material Design 2, který byl po uvedení nové verze nahrazen za Material Design 3. S přechodem vzniklo několik, ať už méně nebo více závažných problémů, které ale byly postupem času odstraněny. Pro veškeré naprogramování aplikace bylo využito vývojové prostředí Android Studio.

2.1.1 Náhled kamery

Po množství optimalizace i přidání nových funkcí byl finální náhled kamery téměř hotový. Funkce jako taková se ze 323 řádků zmenšila na přibližně 153 řádků, a přitom výsledek obsahuje mnoho nových funkcí. Hlavní optimalizační část byla docílena díky ViewModelu, který nově uchovává veškerá data kamery, její nastavení, stav i funkci na samotné detekování. Pro zpracování obrazu byla vytvořena třída **ImageClassfier**, která obsahuje funkci **detect** pro načtení obrazu z kamery, jeho transformaci a vložení do modelu. Pokud model vyhodnotí, že se v obrazu nachází nějaké zvíře s jistotou přesahující 50 %, pak vrátí jeho název, v opačném případě vrací prázdný řetězec. Pomocí tohoto výstupu se buď získá objekt **Animal** obsahující informace o detekovaném zvířeti, nebo prázdná hodnota **null**. Tato informace je dostupná ve ViewModelu v položce **classifiedAnimal.value**.

2.1.1.1 Informace o detekovaném zvířeti

Veškerá „logika“ se děje v hlavní **@composable** funkci pro zobrazení kamery. Zde se v pravidelných intervalech 800 ms kontroluje hodnota **classifiedAnimal** a pokud je po tomto čase stejná, můžeme s jistotou říct, že pozorujeme stejný objekt (zvíře nebo nic). Pokud je tedy tato hodnota zvíře, zobrazíme ji uživateli tak, že se z pod stránky vysune tzv. **ModalBottomSheet** neboli „modální“ okno, jehož charakteristikou je vyobrazení na nejvyšší úrovni – UI prvky pod ním nejsou aktivní. Jakmile dojde k tomuto okamžiku, aplikace dále pozastaví detekování a v modálním okně zobrazí obrázek detekovaného zvířete s tlačítkem, kterým se uživatel může dostat do jeho detailu. Jakmile uživatel okno zavře (kliknutím mimo okno nebo jeho stažením dolů), popř. se vrátí z detailu zvířete, aplikace pokračuje v detekování.

2.1.1.2 Pohled na UI

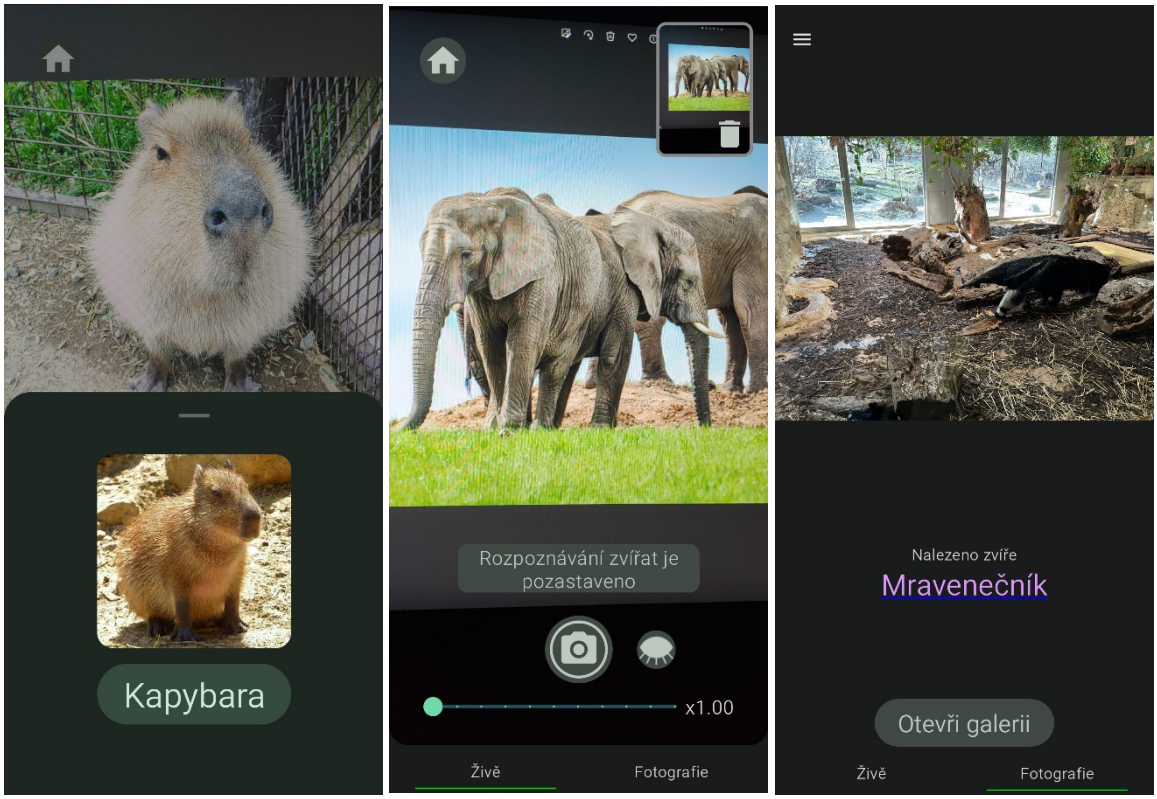
Obraz z kamery zabírá téměř celou plochu stránky. Opakem je míso ve spodní části, které obsahuje rozdělení detekování zvířat na záložky „Živě“ a „Fotografie“.

Záložka **Živě** je automaticky spuštěna při vstupu do této stránky aplikace a značí zobrazení „živého“ obrazu z kamery, a naopak v záložce **Fotografie** má uživatel možnost vybrat fotografii z úložiště mobilního telefonu, na které si přeje detekovat zvíře.

Na hlavní stránce (živě) navíc přibyli funkce pro **pozastavení detekce**, **vytvoření snímku** a **zoom** kamery. Horní menu aplikace bylo odstraněno a pro umožnění uživateli tuto stránku

opustit bylo v levém horním rohu doplněné tlačítko se symbolem domečku, kterým se po stisku dostane uživatel na prvotní stránku aplikace – Kategorie.

Pokud uživatel vyfotí fotografii, její zmenšenina se objeví v pravém horním rohu. Zde ji má ještě po dobu 3 vteřin možnost odstranit. Po uplynutí této doby se zmenšenina fotky vytratí a snímek zůstane uložený přímo v globálním úložišti, kde si jej může uživatel zobrazit.



Obrázek 3. Na levém obrázku můžeme vidět vysunovací okno obsahující detekované zvíře. Prostřední obrázek obsahuje design zobrazení kamery spolu s pořízeným snímkem. Obrázek vpravo ukazuje záložku „Fotografie“ s detekovaným mravenečníkem v obrazu.

2.2 Náhled detailu zvířete

Z drátového modelu aplikace (Obrázek 2.) si lze povšimnout velkého obrázku zvířete zabírajícího téměř půl strany. Ve spodní části tohoto obrázku je pak název daného zvířete. Pod obrázkem už jsou v přehledné formě zobrazeny základní informace a dále delší informační text, který je rolovatelný pro šetření místa. Tímto způsobem byl detail zvířete zobrazen v prvních verzí aplikace.

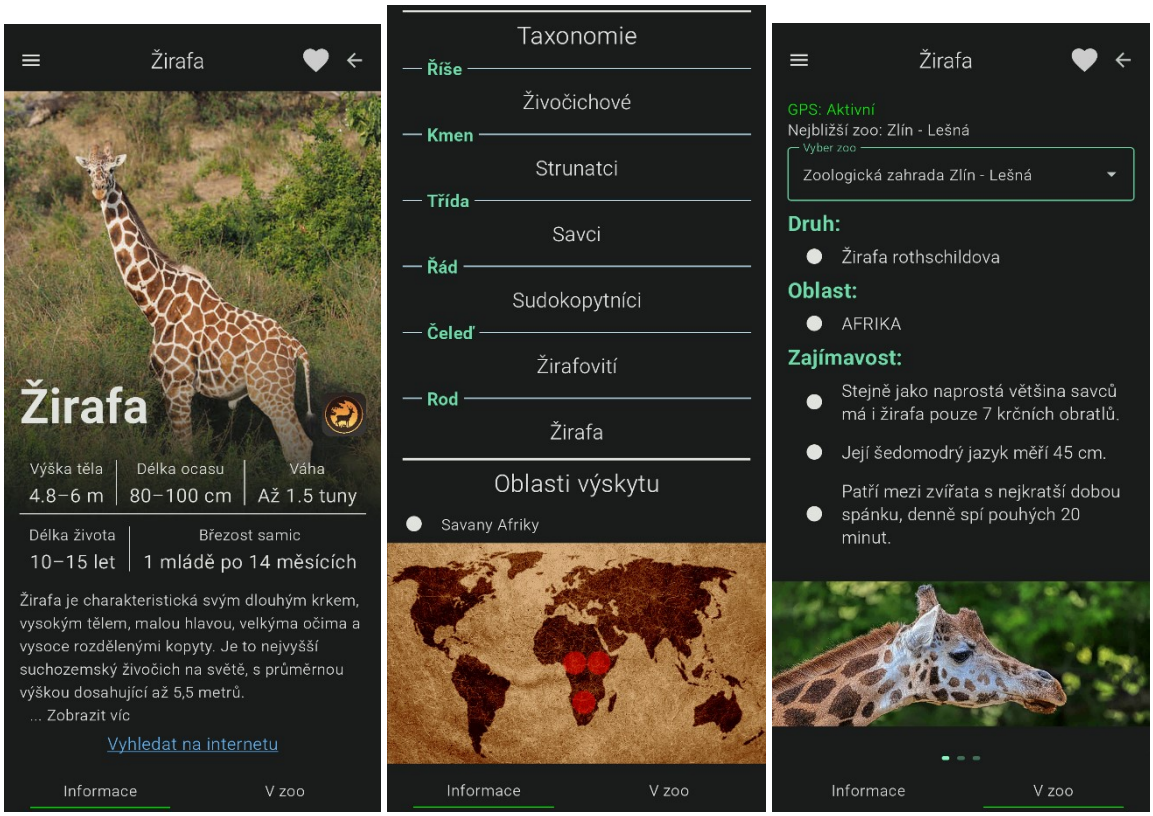
2.2.1 Ukázka

Na následujícím obrázku (Obrázek 4) můžeme vidět obě části detailu zvířete, které v aplikaci jsou. Téměř veškeré komponenty jsou při vstupu do detailu pro příjemnější vzhled animovány a stejně tak je animován i přechod mezi záložkami „Informace“ a „V zoo“.

Název zvířete má aplikovanou vertikální animaci, kdy text sjede shora dolů a řádky „tabulky“ mají animaci horizontální k sobě opačným směrem – horní řada textu přechází zleva doprava a spodní řádek zprava doleva. Dále si můžeme povšimnout ikonky na pravé straně od názvu zvířete, tato ikonka udává, že je aplikace schopná toto zvíře pomocí kamery detekovat.

Informační text je v základu limitován na zobrazení 6 řádků, na které když uživatel klikne (kdekoliv v textu), tak se stránka „natáhne“ a zobrazí veškerý dostupný text. Toto zobrazení je taktéž animováno plynulým přechodem a lehkým poskočením.

Položky Taxonomie mají opět horizontální animaci vysunutí zleva doprava, a i tečky na mapě mění plynule svou velikost.



Obrázek 4. Detail žirafy v aplikaci.

Když se přesuneme na záložku „V zoo“, dostaneme výsledek první dostupné zoologické zahrady v seznamu, která dané zvíře chová. Pokud máme připojení GPS, automaticky se zobrazí nejbližší zoologická zahrada, ale jen v případě, že dané zvíře chová. Můžeme vidět, že na stránce máme informace ohledně druhu chovaných zvířat daného rodu, informace zoologické zahrady o jejím chovu a popřípadě zajímavost. Veškeré texty jsou pořízeny ze stránek každé ze zoologických zahrad. Ve spodní části si můžeme zobrazit až 3 obrázky (pokud jsou na jejich stránkách dostupné) zvířete z dané zahrady. Po kliknutí na obrázek se jeho podoba zvětší na celou stránku, opětovným kliknutím se zmenší do základní podoby.

2.2.2 Objevy zoologických zahrad

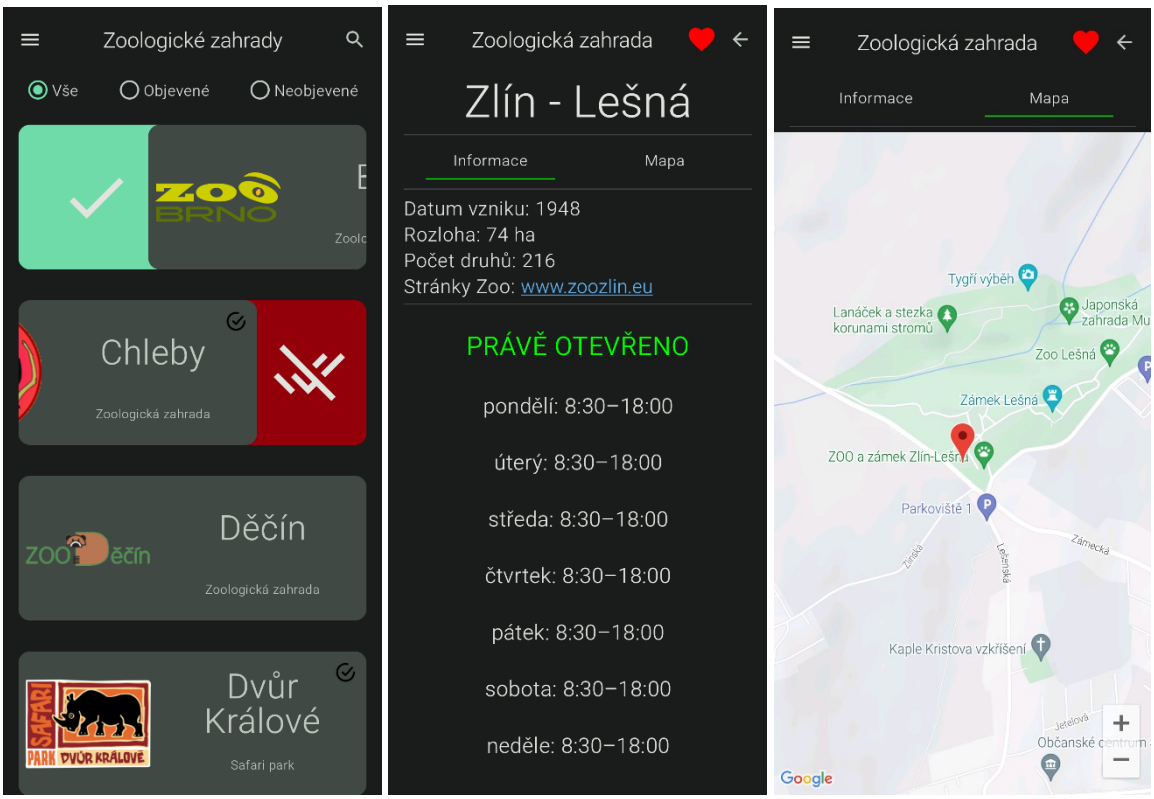
V detailu u objevů zoologických zahrad nalezneme seznam obsahující 14 zoologických zahrad České republiky a filtr pro jejich zobrazení – vše, objevené, neobjevené. Každá

položka seznamu obsahuje logo zahrady, její typ, název / město a status, zda byla objevena. K objevení můžeme využít **swipe gesta**, díky kterým můžeme zahradu uložit do objevů (zleva doprava) nebo ji z objevů odstranit (zprava doleva).

Při prokliku do zoologické zahrady se nám načte záložka „Informace“ s velkým nápisem obsahující název zahrady a 4 základní informace: datum vzniku, rozloha, počet druhů zvířat a odkaz na webové stránky. Pokud má uživatel přístup k internetu, aplikace mu dále načte pomocí **Google Places API** informace o otevíracích hodinách na celý týden a dále poskytne informaci, zda je zahrada aktuálně otevřena. Pro tuto potřebu byl naprogramován ViewModel, který se o načtení informací postará a zároveň si jejich stav uchovává.

Ve druhé záložce „Mapa“ se téměř přes celou obrazovku vykreslí Google mapa s nastavenou polohou dané zoo. Mapa je vykreslena opět pouze s dostupným internetovým připojením a o vykreslení se stará **Google Maps API**.

Pro získání API klíče k těmto dvěma API nástrojům je nutné vlastnit **fakturační účet** na developers.google stránce propojený s kreditní kartou, jelikož může být jejich použití při mnoha dotazech zpoplatněno.



Obrázek 5. Pohled na obrazovku zoologických zahrad v Objevech a detail obrazovky obsahující informace a mapu konkrétní zoologické zahrady Brno.

3 TVORBA DATASETU

Dataset tvoří jádro celé práce, jelikož bez něj by se nedala natrénovat neuronová síť, která by byla schopná zvířata rozpoznat. Pro jeho přípravu bylo využito několik technik a nástrojů, které pomohli velkou část jeho tvorby urychlit automatizací. Celkem byly vytvořeny 2

rozdílné datasety pro vývoj modelu zaměřeného na detekci objektů v obraze a na pouhé klasifikování obrazu. Tvorba datasetů a jejich úprav zabrala v práci nespočet (stovky) hodin a bez naprogramovaných skriptů by je ani nebylo možné vytvořit. K tomu byl využitý programovací jazyk Python ve verzi 3.7.9. Celkem vzniklo asi **9 skriptů** s následujícími vlastnostmi:

- Stahování obrázků ze stránky Flickr pomocí jeho API
- Augmentace dat – vytvoření několika kopií originálních obrázků s aplikováním různých transformací pro zajištění většího počtu trénovacích dat
- Kontrola vadných obrázků, které zapříčiňovali chybu během trénování
- Přejmenovávání obrázků a jejich přidružených anotačních souborů
- Skript pro editaci anotačních souborů XML – odstranění redundantních informací či jejich změna
- Skript pro odstranění příliš malých a nekvalitních obrázků a také k odstranění příliš malých objektů zvířat z anotačního souboru
- Skript pro generování „background“ obrázků a jejich anotací – **Background** dataset slouží pro zpřesnění modelu
- Skript na rozdělení celého datasetu na složky **test** a **train** – Train slouží pouze pro učení sítě a Test pro její kontrolu
- Skript ke generaci souboru **tfrecord**, který právě obsahuje data ze složek train a test. Tento soubor je vstupním bodem k trénování modelu

S těmito skripty a dlouhé manuální práci – ruční filtrování všech obrázků (auto značky Jaguár není zvíře jaguár), ale taky ručním značení objektů (zvířat) v obrázku, byly datasety připraveny k použití trénování konvoluční neuronové sítě.

4 TRÉNOVÁNÍ MODELU

Model konvoluční neuronové sítě hraje nejdůležitější roli této aplikace. Abychom mohli síť naučit, bylo potřebné nasbírat a vytřídit velký počet obrázků, na kterých se síť učí.

Trénování navíc od datasetu vyžaduje i vysoký hardwarový výkon a spoustu času. První vytvořené modely byly trénovány na osobním notebooku **MSI GE62 VR 7RF Apache Pro** s 16 GB pamětí RAM, grafickou kartou NVIDIA GTX1060 s 3 GB pamětí a CPU i7-7700HQ. Rychlost trénování modelů obvykle závisí na dostupném HW, složitosti modelu a počtu trénovacích dat. Později byla možnost použití grafických karet virtuální organizace **Cesnet MetaCentrum**, které díky jejich výkonu dovolili plné trénování modelů v relativně krátkém čase. K trénování autor využil dostupnou **TensorFlow Object Detection API**, která efektivně pomohla s trénováním modelu. Díky ní lze jednoduše nastavit konfigurační soubor před-trénovaného modelu a využít jej tak jako základ nového modelu pro tuto práci.

5 TESTOVÁNÍ MODELU

Pro otestování exportovaného **tflite** modelu byl vytvořen skript, který jej prověří na testovacích datech.
Tento skript byl převzatý z GitHubu (https://github.com/ChrystleMyrnaLobo/tflite-object-detection/blob/master/inference.py) a následně upraven pro potřeby autora. Tímto způsobem byly porovnány 4 poslední vytvořené modely, které se mezi sebou lišili zejména způsobem trénování a lehkými změnami datasetu. Dále byl do testování zařazen v pořadí 13. model, jež byl posledním modelem neobsahující background obrázky v datasetu.

Tabulka 1. Porovnání natrénovaných modelů.

Číslo modelu	Přesnost klasifikace	Přesnost průniku detekce	Doba měření
13	16,10 %	15,72 %	32 minut
16	76,56 %	71,75 %	42 minut
18	84,56 %	73,90 %	43 minut
19	85,89 %	75,07 %	42 minut
20	86,07 %	76,20 %	43 minut

Z tabulky (Pro otestování exportovaného **tflite** modelu byl vytvořen skript, který jej prověří na testovacích datech.
Tento skript byl převzatý z GitHubu (https://github.com/ChrystleMyrnaLobo/tflite-object-detection/blob/master/inference.py) a následně upraven pro potřeby autora. Tímto způsobem byly porovnány 4 poslední vytvořené modely, které se mezi sebou lišili zejména způsobem trénování a lehkými změnami datasetu. Dále byl do testování zařazen v pořadí 13. model, jež byl posledním modelem neobsahující background obrázky v datasetu.

Tabulka 1.) můžeme vyčíst různé hodnoty měření každého modelu i přes použití stejného testovacího datasetu (pro modely č. 16–20). Samozřejmě je to díky tomu, jak byly jednotlivé modely natrénovány. Tyto modely oproti modelu č. 13 obsahují ve svém trénovacím datasetu background obrázky, díky čemu můžeme vidět extrémní rozdíl v přesnosti modelů. Skript ovšem nefiltruje přesnost každé detekce, tudíž se do celkového průměru počítají i velmi špatné detekce například pod 50 %. Stejně tak to je ale i u zbylých modelů, které dosahují mnohem lepší výsledek, což je zajímavé sledovat.

Modely číslo **16 a 18** jsou trénovány na shodném datasetu se shodně nastavenými parametry konfiguračního souboru. Jediný rozdíl je ten, že na **modelu 16** byl aplikován experiment **trénování od nuly**. To znamená, že byl učen pouze na autorem vytvořeném datasetu a nebylo tedy využito techniky *transfer learning.* Učení **modelu 18** naopak vycházelo s před-učeného modelu, a tak můžeme vidět rozdíl přesně 8 % v přesnosti klasifikace na nových datech trénovacího datasetu.

Model č. 19 je téměř shodný jako model 16/18. Jedinou změnou jsou nové obrázky datasetu kočky a psa pro trénování. Následné trénování probíhalo totožně jako model 18, tedy s využitím *transfer learning* metody. Díky tomu můžeme pozorovat přes 1 % zlepšení v přesnosti klasifikace i lokalizace.

Dalším **experimentem** byl model číslo **20**. Trénování probíhalo na totožném datasetu jako u modelu 19, ale změnou prošel konfigurační soubor trénování, což podle našeho skriptu přináší ještě o něco málo lepší výsledek.

ZÁVĚR

Cílem této práce bylo vytvořit kompletní mobilní aplikaci na operační systém Android, která by byla schopná rozpoznat vybrané druhy zoologických zvířat. Aby byla aplikace této činnosti schopná, musel být natrénován model konvoluční neuronové sítě. K natrénování takového modelu bylo navíc potřebné vytvořit kvalitní dataset obsahující tisíce obrázků obsahující dané zvířata. Aby aplikace nebyla chudá na funkcionalitách, bylo v aplikaci za cíl vytvořit i seznam těchto zvířat spolu s jejich detailním náhledem a umožnění uživateli uložení těchto zvířat, a navíc i zoologických zvířat, do tzv. objevů.

Během přibližně 10 měsíců vznikla mobilní aplikace schopná rozpoznat 13 druhů zoologických zvířat pomocí obrazu mobilní kamery. Dalších 6 druhů zvířat, celkově tedy 19, je aplikace schopna rozpoznat pomocí odlišně naučeného modelu na vybrané fotografii z galerie mobilního zařízení. Aplikace taktéž dokáže pozastavit detekci zvířat, vyfotit snímek nebo zoomovat kamerou. Mezi její další funkce patří vyhledání a zobrazení detailních informací zvířete, vyhledání nejbližší zoologické zahrady pomocí GPS nebo uložení navštívení zoologických zahrad a zvířat do statistik nazvaných v aplikaci jako Objevy. Ve finále byla aplikace řádně otestována a porovnaná s dvěma populárními aplikacemi Seek a Google Lens.

Aplikaci je možné využít na jakémkoliv Android zařízení ve verzi 7.0 a výš. Může sloužit jako učební pomůcka dětem nebo jako interaktivní průvodce zoologickou zahradou. Díky vyhledávání zvířat může sloužit i jako rychlá encyklopedie nebo přehledný nástroj sloužící k zobrazení [v budoucnu] všech zoologických zahrad minimálně v České republice.

V budoucnu by aplikaci nejvíce prospělo přidání nových zvířat i zoologických zahrad. Zasloužila by si hezčí uživatelské prostředí a volbu nastavení, kde by si uživatel mohl nastavit například domovskou obrazovku, jazyk aplikace nebo funkce mobilní kamery. Mezi nové funkcionality by se mohlo zařadit rozpoznávání zvířat i podle zvuku, možnost zvíře zobrazit pomocí augmentové reality, možnost ukládání více informací o objevu nebo nastavit jejich objevení při rozpoznání kamerou či fotografií.