Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Vývoj multiplatformní mobilní aplikace s využitím frameworku Flutter

Bakalářská práce

Autor: Jan Stehno
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Jakub Beneš

Hradec Králové duben 2023

Prohlášení:

Prohlašuji, že jsem bakalářskou/diplomovou práci zpracoval/zpracovala samostatně a s použitím uvedené literatury.

……

V Hradci Králové dne 25.4.2023 Jan Stehno

Poděkování:

Děkuji vedoucímu bakalářské/diplomové práce Ing. Jakubovi Benešovi za metodické vedení práce a za veškerou pomoc při její tvorbě.

Anotace

Bakalářská práce se zabývá popisem vývoje multiplatformní mobilní aplikace ve vývojovém prostředí Flutter. Nejprve se práce zabývá obecným popisem vývoje multiplatformních mobilních aplikací a seznamuje čtenáře s možnými řešeními tohoto problému v podobě vývojových prostředí. Dále práce rozebírá podrobněji jednu z porovnávaných technologií - Flutter. Pomocí této technologie bude vyvinuta multiplatformní mobilní aplikace. V rámci bakalářské práce je aplikace průběžně testována a nasazena v mobilních obchodech aplikací Google Play a App Store.

Annotation

Title: Development of a multi-platform mobile application using the Flutter framework

The purpose of the bachelor thesis is to describe the development of a multi-platform mobile application using the Flutter framework. First, the thesis deals with a general description of the development of multi-platform mobile applications and introduces the reader to possible solutions to this problem in the form of development environments. Furthermore, the thesis will analyze one of the compared technologies, Flutter, in more detail. A multi-platform mobile application will be developed using this technology. As a part of the thesis, the application is continuously tested and deployed in the Google Play an App Store mobile application stores.

Obsah

Seznam obrázků

Seznam ukázek kódu

1 Úvod

V době moderních digitálních technologií má skoro každý k dispozici mobilní telefon. Mobilní telefon se stal vlastně takovým přenosným počítačem. Uživatelé je mají neustále po ruce a tím jim umožňují mimo volání a posílání zpráv také možnost připojení k internetu a tím i přístup k obsahu různých článků, informací, služeb a produktů, firemní nebo osobní online komunikaci, anebo zábavě. Tímto snadným a rychlým přístupem umožňují zvyšovat produktivitu a zjednodušovat mnoho činností. Uživatel může všechny tyto možnosti využívat například pomocí webové stránky spuštěné v nainstalovaném prohlížeči. Druhou možností jsou mobilní aplikace.

Mobilní aplikace se staly nedílnou součástí každého mobilního zařízení a mohou najít uplatnění v mnoha oblastech. Od běžné kalkulačky nebo poznámkového bloku, přes počasí, aktuální novinky, blogy a doprovodné aplikace, ke streamovacím aplikacím a hrám.

Herní průmysl je obrovský a nejen ten, který se zaměřuje na mobilní zařízení. Hry na stolní počítače nebo laptopy bývají velmi často zábavou a výplní volného času. Mnoho her je vyvíjeno a aktualizováno po dlouhou dobu. Propracované hry potřebují v určitém bodě vývoje i doprovodný dokument, webovou stránku, nebo aplikaci s informacemi, které se jí týkají. Tato stránka obsahuje důležité informace zaměřené na hratelnost a usnadnit tak uživateli jeho herní zážitek.

Jednou takovou hrou se stal i lovecký simulátor theHunter: Call of the Wild, který je v momentální době vyvíjen už 6 let pro stolní počítače, PlayStation a Xbox. Vzhledem k množství informací, které hra obsahuje, zatím nenabízí zmíněnou webovou stránku nebo oficiální doprovodnou mobilní aplikaci, která tyto informace uživatelům sdílí. Tato bakalářská práce bude zaměřena právě na vývoj zmíněné doprovodné aplikace pro tuto hru.

2 Cíl práce

Nejprve se bude potřeba zamyslet nad možnými způsoby vývoje multiplatformní mobilní aplikace. Tyto způsoby porovnat a poté vybrat jednu z těchto technologií. Technologie bude umožňovat přístup k přídavným modulům a knihovnám. Tyto moduly a knihovny budou potřeba pro jednodušší vývoj a zároveň rozšíření aplikace o důležitou funkčnost a ovládaní.

Cílem práce bude vytvořit multiplatformní mobilní aplikaci za pomocí právě vybrané technologie. Aplikace bude umožňovat rychlý a přehledný přístup k informacím. V aplikaci bude kladen důraz na vícejazyčnost, jednoduchost a intuitivnost ovládaní, nastavitelnost rozhraní, včetně možnosti rozšíření o nové funkce a informace. Výsledná aplikace bude publikována a nadále aktualizována v obchodech Google Play a App Store.

3 Mobilní aplikace

Mobilní aplikace je specializovanou softwarovou aplikací vytvářenou speciálně pro chytré telefony, tablety, emulátory a další mobilní zařízení. Mobilní aplikace se staly každodenní samozřejmostí. Mezi první patří především aplikace pro prohlížení e-mailu, zpráv a počasí, nebo kalendář a další základní funkce.

Po zvyšující se poptávce po nových a sofistikovanějších aplikacích se začaly vyvíjet mobilní hry, firemní aplikace nebo praktické aplikace využívající GPS, biometrické senzory pro rozpoznání otisku prstu nebo obličeje, nebo aplikace pro měření životních funkcí.

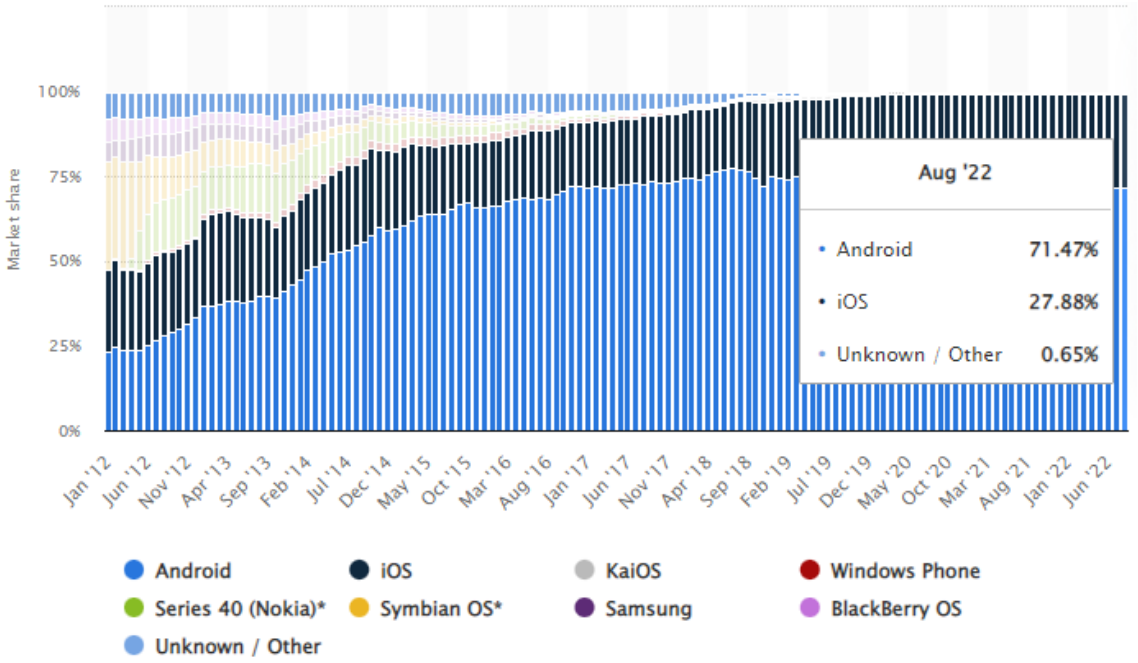
Mobilní aplikace firem, obcí nebo měst jsou velmi silným marketingovým nástrojem, do kterého se určitě vyplatí investovat. Mobilní aplikace nabízí relativně levný způsob propagace firemních služeb, e-shopů, kaváren nebo obcí a měst. Mobilní aplikace jsou moderní, rychle dostupné a umožňují lehký přístup k informacím.

3.1 Vývoj mobilních aplikací

Vývoj aplikace může být rozdělen do několika kroků. Analýzy požadavků, při které je potřeba si vytyčit, co přesně se po aplikaci chce a co bude nabízet. Návrhu uživatelského rozhraní, které má být uživatelsky přívětivé a moderní. Programování, validace a testování, bez kterého aplikace na některých mobilních platformách nebude dělat nebo vypadat tak, jak má. Mezi poslední kroky patří distribuce, bez které se aplikace nedostane mezi cílové uživatele. A správy spojené s dalším vývojem, který je potřeba pro udržení aplikace v co možná nejaktuálnějším a nejstabilnějším stavu. Důležitý je také výběr koncového operačního systému, nebo mobilní a vývojové platformy. [3][4]

3.2 Mobilní operační systémy

Při výběru koncového operačního systému je důležitá hlavně jeho rozšířenost a známost. Vzhledem k tomu, že podle statista.com má Android zhruba 72% podíl na trhu a iOS zhruba 28%, se vývoj aplikace zaměří právě na kompatibilitu pouze pro tyto dva operační systémy. Dalšími operačními systémy, které jsou vidět na obrázku níže, jsou například KaiOS, Windows Phone, Samsung a další. [11]



Obrázek 1: Podíl mobilních systémů na trhu 2014 - 2022

3.2.1 Android

Android je mobilní operační systém, který vyvíjí firma Google. Má zatím největší zastoupení na světě od roku 2013. Od roku 2008, kdy byla vydána úplně první verze operačního systému Android, se operační systém postupně měnil a vyvíjel až do nynější verze Android 13. Android v současné době běží na telefonech, tabletech, televizích, autorádiích nebo hodinkách a zařízeních pro virtuální realitu. [7][13]

Pro vývoj aplikací pro tento operační systém je podporované vývojové prostředí Eclipse včetně, v roce 2014, odkoupeného prostředí Android Studio, které se stalo

hlavním vývojovým prostředím. Nativními jazyky pro vývoj aplikací pro Android jsou Java a Kotlin. [13]

Všechny nástroje pro vývoj pro platformu Android jsou obsaženy v sadě Software Development Kit (dále jen jako SDK). Sada SDK je rozdělena na tři druhy. Základní, která obsahuje nástroje pro debugging, nástroje pro testování aplikace, emulátor umožňující testování aplikace bez fyzického zařízení, nástroje pro přenos souborů do zařízení a mobilní platformy. Doporučená, která navíc od základní obsahuje i USB driver, příklady kódů a dokumentaci. A plná, která zpřístupňuje Google API a další mobilní platformy. [7][13]

3.2.2 iOS

iOS je mobilní operační systém vyvíjený společností Apple. Je po operačním systému Android druhým nejrozšířenějším na světě. iOS je na rozdíl od systému Android více uzavřený a může být nainstalován pouze na zařízeních od společnosti Apple. Dalším velkým rozdílem je nemožnost nainstalovat do zařízení se systémem iOS jiné než Apple společností schválené aplikace v obchodě App Store. Tím, že je iOS tolik omezen a kontrolován ho činí mnohem bezpečnějším oproti Androidu. Nejnovější verzí systému iOS je verze 16. [7][12]

Vyvíjet aplikace pro systémy iOS je možné v Apple zdarma nabízené aplikaci XCode. Nativními jazyky pro vývoj aplikací pro iOS jsou Objective-C a Swift. [12]

Stejně jako u vývoje aplikací pro systém Android zde existuje sada SDK. Stejně tak je i rozdělena do několika částí. Části Cocoa Touch, která umožňuje práci s multi-dotykem, lokalizací nebo kamerou. Media části zajišťující zvuk a práci s obrázky a videem. Práci se sítí, vnitřní databází, vlákny a jádrem umožňuje část Core Services. A Mac OS X Kernel zpřístupňuje operovat s protokoly TCP/IP, sokety, souborovým systémem, řízením spotřeby nebo zabezpečením. Stejně jako sada SDK pro Android i ta pro iOS umožňuje využít iPhone emulátor pro testování aplikace bez fyzického zařízení. [7][12]

4 Vývojová prostředí

Vývoj aplikací pro Android pomocí Android SDK je omezen pro zařízení, na kterých běží systém Windows. Stejně tak je vývoj aplikací pro iOS pomocí iOS SDK omezen pro zařízení, na kterých běží systém MacOS. Protože je potřeba vyvinout aplikaci pro oba systémy zároveň, je zapotřebí také nadstavbu vývojových prostředí zmíněných výše, které takový vývoj umožní pomocí společného SDK. V tomto případě se je možné potýkat s jiným programovacím jazykem a omezenými nebo alternativními funkcemi.

4.1 Ionic

Ionic je open-source software framework vydaný v roce 2013. Ionic poskytuje nástroje a služby pro vývoj hybridních mobilních, desktopových a progresivních webových aplikací založených na moderních technologiích a postupnech pro vývoj webových aplikací s využitím webových technologií jako HTML5, CSS/SASS a JavaScript. Umožňuje také integraci frameworků Angular, React nebo Vue. [10]

Ionic používá moduly Cordova a Capacitor k získání přístupu k funkcím hostitelských operačních systémů, jako je fotoaparát, svítilna nebo GPS. Pomocí Web Components poskytuje Ionic vlastní komponenty a metody pro interakci. Kromě sady SDK umožňuje Ionic také služby pro automatické sestavení aplikace nebo nasazení kódu. Ionic má také své vlastní vývojové prostředí, Integrated Development Environment (dále jen jako IDE), Ionic Studio. [10]

Ionic podporuje vývoj pro Android 4.4 a výše a pro iOS 10 a výše. Existuje také Ionic 2, který umožňuje vývoj aplikací pro Windows 10. Aplikace poté běží za použití nativního i webového kódu, což umožňuje přístup k nativním funkcím, přičemž většina uživatelského rozhraní využívá standardní webové technologie. [10]

4.2 NativeScript

NativeScript je open-source Framework vydaný v roce 2014. Je používán pro tvorbu aplikací pro Android a iOS. Aplikace NativeScript jsou psány pomocí JavaScriptu nebo pomocí jazyka, který lze přetransformovat do jazyka JavaScript. NativeScript stejně jako Ionic podporuje Angular, React a Vue. [10]

Komponenty a rozhraní jsou definovány pomocí souborů XML. Tyto soubory jsou použity pro spuštění nativního kódu. Zdrojový kód není nijak kompilován a běží přímo na zařízení. NativeScript aplikace nijak nemanipuluje s DOM ani nijak neiteraguje s prohůížečem. [10]

4.3 React Native

React Native je open-source aplikační framework vydaný v roce 2015. Je používán pro tvorbu aplikací pro Android, iOS, macOS, web, Windows nebo Univerzální Windows Platformu (společné aplikace pro Windows, Windows Mobile, Xbox a HoloLens). [6][10]

Funkčně se React Native velice podobá klasickému Reactu. Běží v procesu na pozadí a komunikuje s nativní platformou prostřednictvím serializovaných dat. Komponenty React obalují stávající nativní kód. Přestože je syntaxe velice podobná, React Native HTML ani CSS nepoužívá. Místo toho se k manipulaci nativního zobrazení používají zprávy JavaScriptu. [6][10]

4.4 Flutter

Flutter je open-source aplikační framework vydaný v roce 2017. Podobně jako React Native, Flutter je používán pro tvorbu aplikací pro Android, iOS, macOS, Linux, Windows nebo web. [6][8][10]

Flutter obsahuje správce balíčků Dart, které umožňují uživatelům sdílet a využívat veřejné balíčky a pluginy Flutter. Flutter mimo jiné obsahuje dvě sady komponent. Material Design balíček obsahující schémata vzhledu a jazyka od Google. A Cupertino Design balíček obsahující schémata vzhledu a jazyka od Apple.

Flutter podporuje vývoj pro Android 4.1 a výše a pro iOS 11 a výše. Kromě zmíněného IDE - Android Studio, je možné pro vývoj Flutter aplikací využít také IntelliJ IDEA nebo Visual Studio Code. [4][6][8][10]

5 Flutter

Flutter namísto využívání webových zobrazení nebo využívání OEM widgetů zařízení vykresluje Flutter všechny komponenty obrazu pomocí vlastního vysoce výkonného renderovacího jádra. Tato výhoda umožňuje tvorbu aplikací, které jsou stejně výkonné jako ty nativní. [4][5][8][10]

Aplikace Flutter jsou psány v jazyce Dart. Pro lepší výkon využívá Flutter tzv. „ahead-of-time (AOT) compilation“ (kromě webu je pomocí této kompilace kód přetransformován do jazyka JavaScript), která umožňuje snížit množství procesů a výkonu při běhu aplikace tím, že přetransformuje kód vyšší úrovně do toho s nižší úrovní. [4][5][8][10]

5.1 Hot-Reload

Flutter při vývoji podporuje tzv. hot-reload. Tato funkce je považována za jeden z hlavních faktorů pro urychlení vývojového cyklu. Hot-reload je v podstatě implementováno vložením aktualizovaného zdrojového kódu do běžícího kódu na virtuálním zařízení beze změny vnitřní struktury aplikace. Tzn., že všechny přechody a akce aplikace budou zachovány i po hot-reloadu. [4][5][8]

5.2 Dart

Všechny aplikace Flutter jsou psané za pomoci Dart. Dart je programovací jazyk, který byl vyvinut a je stále spravován společností Google. Ve firmě Google je jazyk velmi rozšířen a pomocí něj byla schopna firma Google vyvinout například aplikaci AdWords (aplikace/služba pro sdílení/inzerování reklam) a další velké aplikace. [4][14]

Dart byl původně vyvinut jako nástupce JavaScriptu. Jazyk tudíž implementuje jeho důležité a standardizované charakteristiky. Pro programátory má Dart syntax podobný Javě. Vzhledem k tomu, že zobrazení Flutter aplikace je pravidelně obnovováno/vykreslováno, je Dart optimalizován pro náročnou práci s pamětí pomocí „Generational Garbage Collection“. [4][14]

5.3 Stav

Podle definice nalezené ve Flutter API docs: „Stav je informace, která může být čtena synchronně, když je widget sestaven a může se změnit v průběhu života widgetu.“ Stav reprezentuje dynamická data, která ovlivňují to, co se zobrazuje. Stav je považován za vrstvu reprezentující vnitřní strukturu widgetu. Jednodušeji řečeno, stav popisuje, jak bude widget reagovat na interakce od uživatele. [1][2][5][8]

V základu je každý widget neměnný, ale mohou mít stavový objekt. Kdykoliv se stav widgetu změní, tento widget je znovu sestaven a vykreslen. To znamená, že namísto změny vlastností widgetu, je vytvořen widget s vlastnostmi novými. Změna stavu v jednom widgetu většinou neovlivňuje widgety další (předchůdce i potomky). To dělá Flutter velmi efektivním z pohledu výkonu. [1][2][5][8]

5.4 Widget

Widgety jsou nejdůležitější součástí Flutter aplikací. Nejenom, že kontrolují a mění to, co se zobrazuje, ale také vyřizují a odpovídají uživateli na jeho akce. Proto je důležité, aby widgety pracovaly co možná nejrychleji (včetně renderování a animování). [4][5][8]

Namísto znovupoužívání OEM widgetů (jako má například React Native), Flutter má svoje vlastní widgety. To znamená, že Flutter sám rozhoduje o tom, jak a kdy bude který widget zobrazen. Flutter určitým způsobem přesouvá widgety a vykreslovač ze systémové úrovně do aplikační. To jim umožňuje být více upravitelnými a rozšiřitelnými. Na druhou stranu je díky tomu aplikace o něco větší. Flutter má dva typy widgetů. Stateless Widget (bezstavový, dále jako SLW) a Stateful Widget (stavový, dále jako SFW). SFW používá odpovídající stavový objekt, který

reprezentuje jeho stav. Na druhé straně je SLW, který žádný takový objekt nepotřebuje, protože se od něj žádná reakce neočekává. SLW je neměnný, což zabraňuje jeho, výše zmíněnému, pravidelnému překreslování. [4][5][8]

5.4.1 Container

Jeden ze základních widgetů Flutteru je Container. Containeru lze definovat velké množství parametrů. Barvu, velikost, odsazení, nebo tvar. Vzhledem k velkému počtu možností, lze Container považovat za nejobecnější widget. Jeho poté více specifické nadstavby jsou například AnimatedContainer, SizedBox, Padding nebo Opacity. [8]

5.4.2 Column a Row

Dalšími velmi používanými widgety jsou Column a Row (neboli sloupec a řádek). Tyto widgety se většinou využívají na vykreslování vertikálních i horizontálních listů. Nebo je lze použít pro specifické rozdělení widgetů ve sloupci a řádku. [8]

5.4.3 Button

Widget Button a jeho varianty jsou nedílnou součástí každé aplikace. Přesměrování na další nebo předchozí aplikační stránku, použití možnosti v nastavení, uložení nebo načtení informací a mnoho dalších funkcí lze vyvolat pouhým poklepáním na jeden Button. [8]

5.4.4 GestureDetector

Jedním ze složitějších widgetů ke GestureDetector. Jak už jeho název napovídá, dokáže detekovat gesta uživatele na jejich základě odpovědět. Mezi ty nejdůležitější patři např.: tap, double tap, pan, nebo drag. [8]

5.4.5 FutureBuilder

FutureBuilder je zapotřebí při vykreslování widgetů a informací, které jsou závislé na předchozím asynchronním získání. Pokud například aplikace potřebuje nejprve získat data z internetu, dělá tuto akci asynchronně vůči ostatním úlohám. Vzhledem k tomu, že se tyto informace musí vykreslit pomocí widgetu, je vhodné využít

FutureBuilder. Ten zajistí, že se widget vykreslí s potřebnou informací až po jejím nalezení. [8]

5.5 Balíčky a PUB.DEV

Pub.dev je manažer balíčků pro programovací jazyk Dart. Obsahuje znovupoužitelné knihovny a balíčky pro Flutter, AngularDart a Dart aplikace. Vytvářená aplikace využívá mnoha těchto balíčků pro její snadnější programování a práci se složitějšími funkcemi programovacího jazyka. Balíčky pro vyvíjenou aplikaci lze „doinstalovat" pomocí seznamu přídavných balíčku v souboru pubspec.yaml pod parametr dependencies. [8]

5.5.1 Lokalizace aplikace

Pro snadnější lokalizaci aplikace a práci s jazyky a překladem využívá aplikace balíček easy_localization. Balíček umožňuje definovat pole jazyků, ve kterých je možné aplikaci mít. Balíček také umí uložit nastavený jazyk za běhu aplikace a při vykreslování textových částí jeho zpětné načtení.

5.5.2 Uložiště aplikace

Pro uložení nastavení, jazyka a některých uživatelsky vytvořených informací je zapotřebí uložiště. Toto uložiště se chová podobně jako cache. Na rozdíl od ní je informace v uložišti uchována i při vypnutí aplikace. Uložiště se odstraní až po odinstalování aplikace ze zařízení. Pro využití uložiště používá aplikace balíček shared_preferences. Balíček umožňuje definování názvu ukládaných souborů nebo informací, jejich samotné uložení a načtení.

5.5.3 Práce s externími soubory

I přesto, že aplikace ukládá vytvořené uživatelovy informace ve výše zmíněném uložišti, je vhodné, aby si mohl uživatel své soubory z aplikace vytáhnout a zálohovat. Pro práci s externími soubory, respektive s uložištěm zařízení využívá aplikace balíčku file_picker. Balíček umí ukládat a načítat soubory z vybraného uložiště.

6 Vývoj

Pro demonstraci vývoje multiplatformní aplikace je v rámci této práce vytvářena právě taková aplikace. Aplikace bude doprovodnou aplikací pro počítačovou hru theHunter: Call of the Wild. Jedná se o loveckou hru hranou z první osoby. Aplikace je zaměřena na vyhledávání detailních a důležitých herních informací. Mezi ně patří informace o loveckých rezervacích, lovných zvířatech, použitelných vábničkách, zbraních a dalšího vybavení. Dalšími rozšiřujícími funkcemi jsou mapy s oblastmi lovu a lokacemi zvířat, tabulka s informacemi o právě probíhající činnosti každého zvířete v rezervaci, nebo záznamník úlovků.

6.1 Požadavky

6.1.1 Navigace

Vzhledem k počtu výše uvedených informací a funkcí je důležité vhodné rozvržení funkcí a stránek aplikace. Navigační menu bude obsahovat odkazy na každou z výše uvedených funkcí. Podle potřeby budou odkazy rozděleny stromově. Na stránkách uvnitř stromu nebo na konci (tam, odkud už se nebude dál kam odkázat) bude vždy zobrazena hledaná informace.

6.1.2 Lokalizace

Lokalizace je pro aplikaci jednou z nejdůležitějších částí. Umožňuje uživatelům, kteří nemluví nebo nerozumí například anglicky, využívat jiného implementovaného jazyka. Vytvářená aplikace bude umožňovat výběr jazyka a její uložení pro příští použití. Každý přeložený text bude co nejvěrohodnější. Pro konzistenci informací mezi aplikací a hrou bude text ze hry přeložen přesně tak, jak je v ní. Zároveň bude aplikace přeložena do většiny jazyků, do kterých je přeložená hra – angličtiny, češtiny, ruštiny, němčiny, polštiny, španělštiny, francouzštiny, jihoamerické portugalštiny a japonštiny.

6.1.3 Nastavení

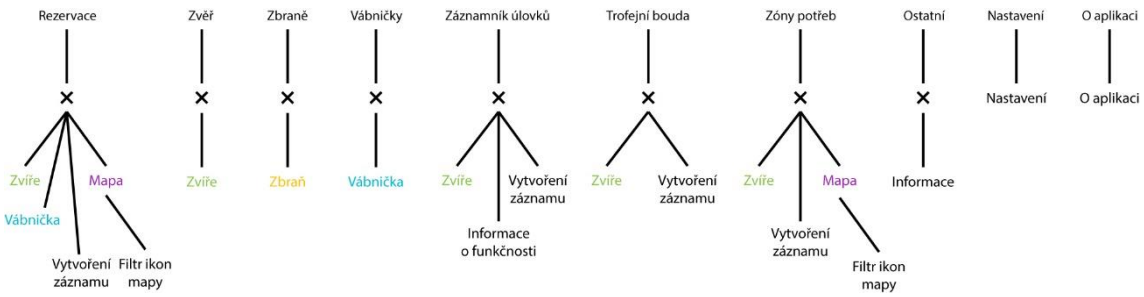
Nastavení aplikace bude umožňovat vybrat jazyk a téma aplikace (světlá, tmavá). Mezi další nastavení bude patřit velikost textu, primární barva a některé vybrané informace nebo možnosti například k funkcím aplikace.

6.1.4 Další požadavky

Aplikace bude přehledná, se snadnou navigací k požadovaným funkcím. Aplikace bude použitelná a správně zobrazená na všech mobilních zařízeních – telefonech, tabletech. Nebude obsahovat nevhodný nebo zavádějící obsah, který nesouvisí se hrou. V případě potřeby bude u složitějších funkčních stránek vysvětlivka, jak s nimi pracovat. Bude obsahovat seznam změn, informace o aplikaci, autorovi a zdroji informací.

6.2 Rozvržení aplikace

Jak již bylo popsáno výše, aplikace v pozadí funguje jako určitý strom. Při zapnutí aplikace je zobrazena uživateli úvodní stránka s navigačním menu a odkaz na list změn. Navigační menu pak obsahuje několik odkazů, které buď vedou rovnou na cílovou funkční stránku nebo informaci. Z každé takové stránky lze postupovat pouze vpřed nebo zpět. Graf níže zobrazuje, odkud kam se dá v aplikaci odkázat a které stránky se chovají jako konečné.



Obrázek 2 Rozvržení aplikace – strom s odkazy a jejich propojení

Každý křížek v obrázku výše znázorňuje list. Tento list je generován z dostupných dat o příslušné informaci. Z grafu je také možné vidět, že na některé koncové stránky se lze dostat z více jiných stránek. Toto je dáno logikou propojení těchto informací.

Aplikační stránky se načítají v podobě zásobníku. Ve spod zásobníku je úvodní stránka. Po odkázání se na jinou se stránka zobrazí nad tou, ze které se odkazovalo. Pokud uživatel zadá, že se ze zobrazené stránky chce dostat pryč, tato stránka je odebrána ze zásobníku a zobrazí se ta, co byla pod ní. Proto se lze z každé stránky dostat „pouze“ vpřed nebo vzad. Vpřed se dá dostat např. pomocí textových tlačítek nebo přímo tlačítek s ikonou znázorňující informaci, na kterou odkaz vede. Pro přepnutí na stránku původní lze použít u zařízení se systémem Android interní navigační tlačítko zpět. Na systému iOS pak lze použít pohyb stránky z leva do prava. U obou systémů je pak možné využít naprogramované tlačítko zpět v hlavičce každé stránky.

6.3 Grafika, barevné zpracování aplikace

6.3.1 Celkové zobrazení

Pro přehledné zobrazení dat a informací je důležité, aby jednotlivé prvky rozhraní byly jasně srozumitelné a viditelné. Aplikace využívá dvou základních témat – světlé a tmavé. Obě témata využívají materiálních barev pro příjemnost a kontrastu pro čitelnost zobrazení. Celá aplikace má pak jednu nastavitelnou primární barvu.

6.3.2 Zobrazení widgetů

Pro styl widgetů je zachován, jako u barev, materialismus. Nedílnou součástí je střídání ostrých a oblých rohů prvků rozhraní, zvýraznění důležitých textů nebo tlačítek a využívání obrázků a ikon.

6.4 Porozumění aplikaci a jejím funkcím

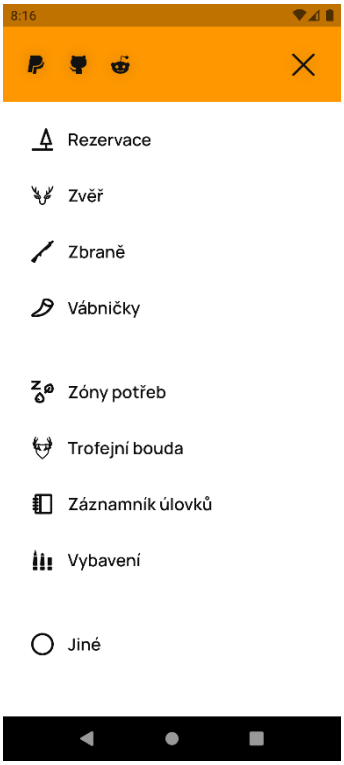
Aplikace je doprovodnou aplikací ke hře. Doprovodná aplikace má za úkol pomoci uživateli s nalezením důležitých informací, které jsou ve hře buď náročné na nalezení nebo má hráč tyto informace k dispozici až později v průběhu hrání či se mu zobrazí pouze za určitých podmínek. Aplikace má všechny tyto informace dostupné pro všechny uživatele ihned od jejího nainstalování.

6.4.1 Jak funguje hra

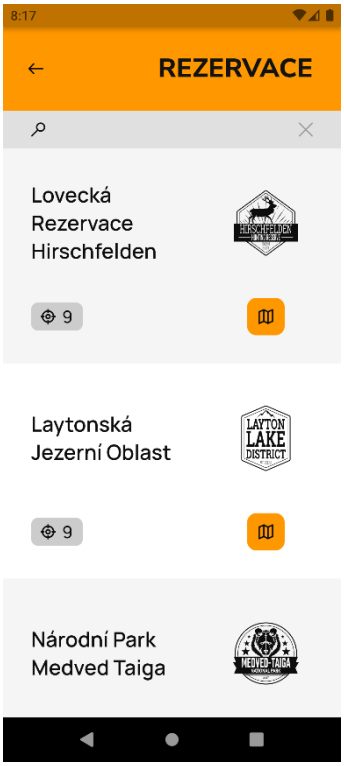
Pro zjednodušení pochopení funkcí funguje hra tak, že v každé rezervaci je určitý počet zvířat, která lze na mapě najít v určitých místech a dělající specifickou aktivitu na základě času – krmení, pití a odpočinek. Tyto zvířata lze pak nalákat vybranými vábničkami a střílet eticky i neeticky nabízenými zbraněmi.

6.4.2 Hlavní statické informace

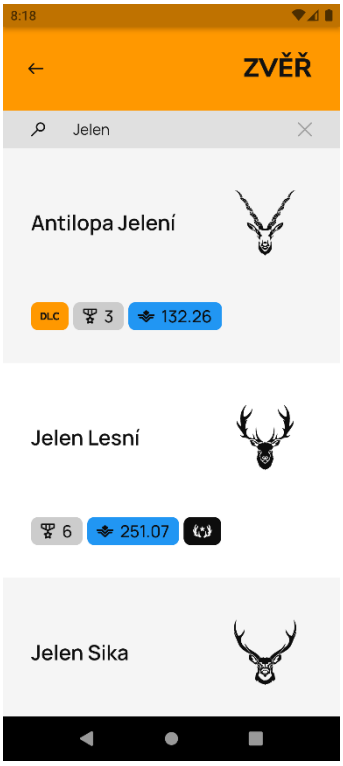
Aplikace obsahuje informace o všech rezervacích, zvířatech, vábničkách a zbraních. K těmto informacím se uživatel dostane pomocí hlavního menu a výběru jednoho z výše zmíněných seznamů. V seznamu pak stačí hledaný prvek najít a vybrat. K rychlejšímu vyhledávání a filtraci výsledků je možnost použít vyhledávací pole v hlavičce seznamu.



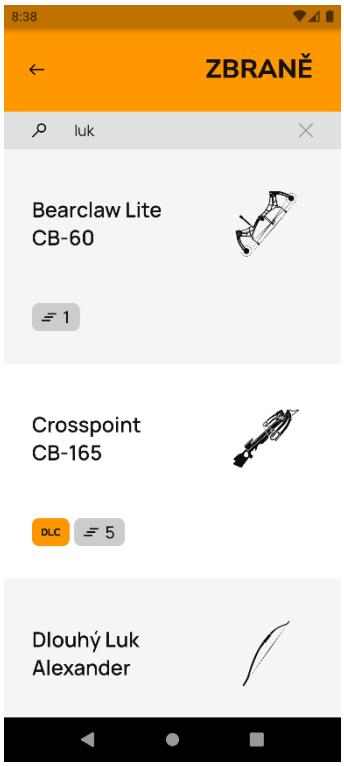
Obrázek 3 Hlavní menu



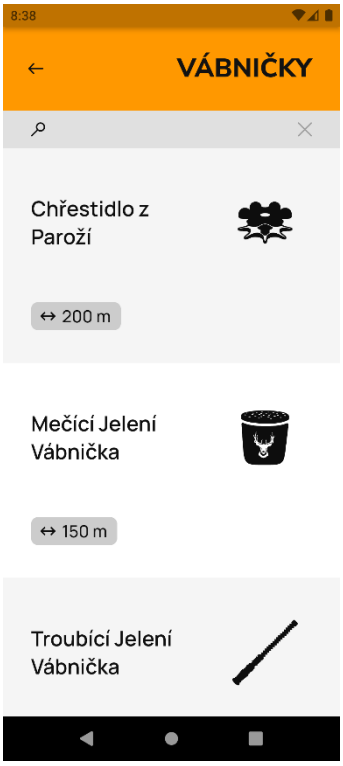
Obrázek 7 Seznam rezervací



Obrázek 6 Vyfiltrovaný seznam zvířat



Obrázek 5 Vyfiltrovaný seznam zbraní

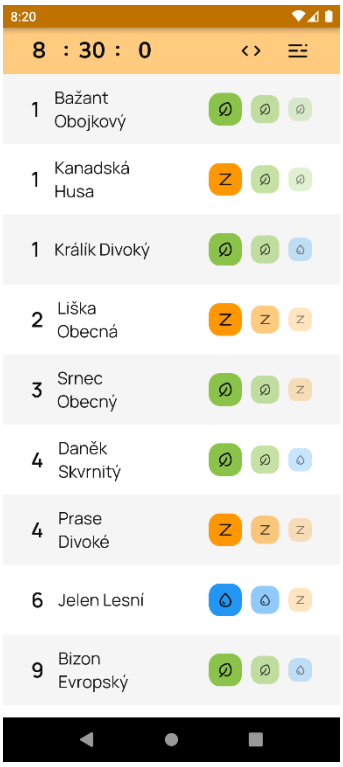


Obrázek 4 Seznam vábniček

6.4.3 Zóny potřeb

Další funkcí jsou zóny potřeb. Tato funkce nabízí tabulku, která zobrazuje aktivity zvířat. Výsledné informace jsou závislé na vybraném čase a rezervaci zadané uživatelem. Čas sám postupuje (pokud není ručně zastaven) a postupně mění zobrazené aktivity na jiné. Kvůli většímu počtu zvířat v rezervaci je možné využít kompaktní zobrazení. Na funkci se lze přesměrovat z hlavního menu.

Obrázek 9 Funkce zón potřeb



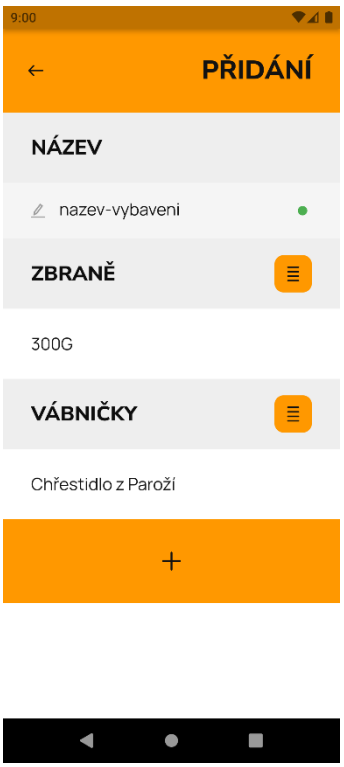
Obrázek 8 Funkce zón potřeb - kompaktní zobrazení

6.4.4 Seznam vybavení

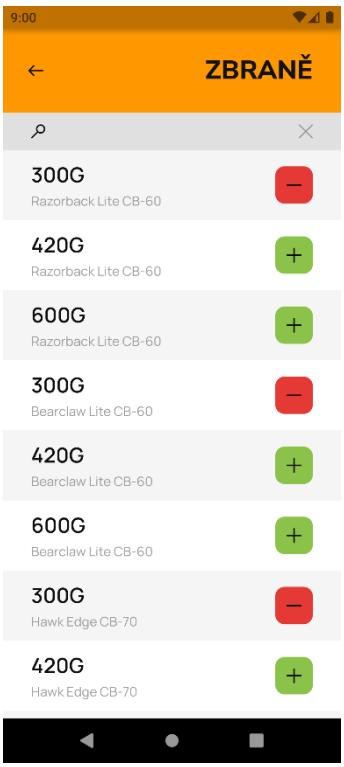
Seznam vybavení je jednou z nejmladších funkcí přidanou do aplikace. Uživatel je schopen vytvořit seznam vybavení podle toho, co používá za set vybavení ve hře. Aplikace nabízí tvorbu seznamu ze zbraní a vábniček. Vytvořený seznam je poté možné vybrat a následně bude pasivně využíván pro zobrazení možnosti zvíře přivábit nebo eticky zastřelit. Ikony pro tyto možnosti jsou zobrazené v seznamu

aktivit ve funkci zón potřeb a v seznamu zvířat v detailu o rezervaci. Seznamy vybavení lze upravit a smazat. Funkce je dostupná pod tlačítkem v hlavním menu.

Obrázek 11 Seznam vybavení



Obrázek 12 Vytvoření seznamu vybavení

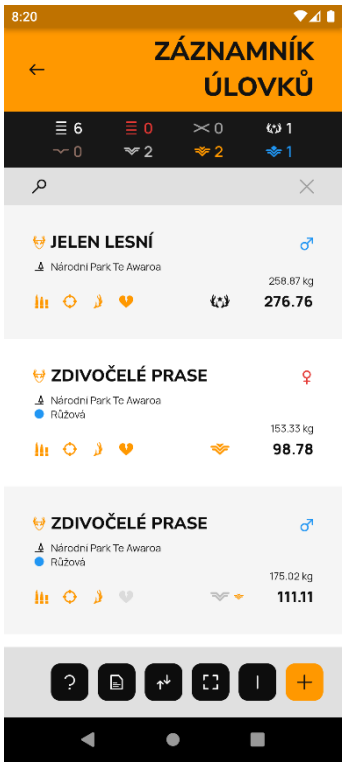


Obrázek 10 Přidání a odebrání jednotlivého vybavení

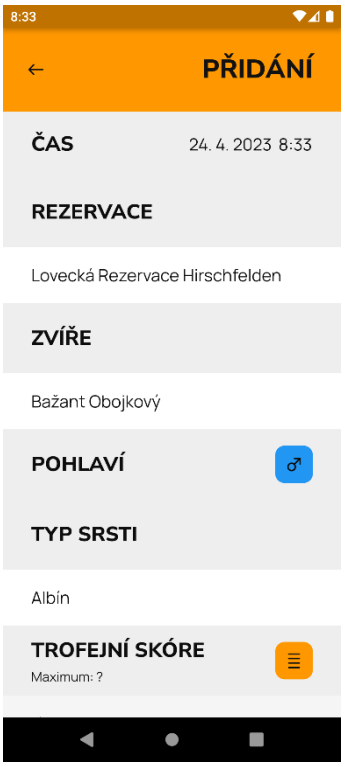
6.4.5 Záznamník úlovků a trofejní bouda

Další velkou a propracovanou funkcí je záznamník úlovků. Záznamník úlovků umožňuje uživateli ukládat své úlovky do přehledného seznamu. Uživatel při vytváření záznamu zadává čas ulovení, rezervaci, zvíře, jeho pohlaví, typ srsti, váhu a trofejní skóre vypočítané hrou. Posledním údajem je tzv. kontrola úlovku opět daná hrou. Jedná se o kontrolu, zda hráč zastřelil zvíře za určitých „etických“ podmínek. Záznam lze vytvořit přímo v záznamníku nebo lze využít možnosti vytvoření záznamu přímo ze seznamu zvířat v detailu o rezervaci nebo z tabulky ve funkci zón potřeb. Alternativou k záznamníku je trofejní bouda. Trofejní bouda slouží pro zobrazení úlovků, které byly hráčem přidány a vystaveny v herní trofejní boudě. Herní bouda neumožňuje zobrazení některých informací, které se zadávají při vytváření záznamu v záznamníku. Tudíž aplikace při vytváření záznamu pouze pro trofejní boudu tyto informace nepotřebuje. Záznamy vytvořené v záznamníku

lze přesouvat mezi záznamníkem a trofejní boudou (imitace přesunutí herního úlovku do boudy). Všechny záznamy je možné upravit a smazat. K oběma těmto funkcím se lze opět dostat z hlavního menu.



Obrázek 14 Seznam zaznamenaných úlovků

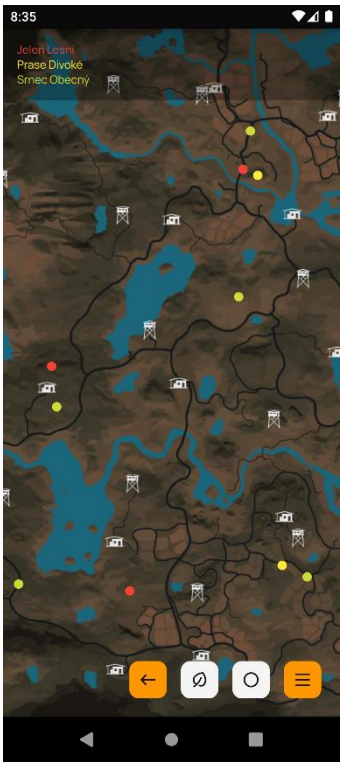


Obrázek 13 Vytvoření záznamu

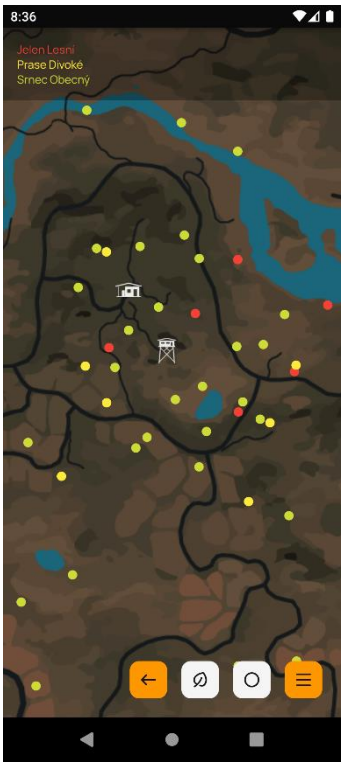
6.4.6 Mapy rezervací

V aplikaci jsou pro uživatele připraveny také mapy rezervací. Na mapu se lze dostat v seznamu rezervací nebo ve funkci zón potřeb. Mapa zobrazuje pozice chat, rozhleden, pozemních úkrytů a lokace výskytu zvířat. Mapa umožňuje vybrat si specifické pozice k zobrazení a dává uživateli také možnost přiblížení. Vzhledem k tomu, jak jsou mapy naprogramovány, jsou v první úrovni přiblížení viditelné pouze chaty, rozhledny a velmi obecné pozice zvířat. V druhé úrovni lze vidět chaty, rozhledny, pozemní úkryty a přesnější pozice zvířat. V poslední úrovni, tedy největším přiblížení, jsou uživateli zobrazeny chaty, rozhledny, pozemní úkryty a přesné pozice zvířat. V této úrovni pak může uživatel zobrazit navíc typ aktivity,

kterou zvíře na pozici většinou provádí. Typ aktivity je dán barvou bodu – zelená pro krmení, modrá pro pití a oranžová pro odpočinek.

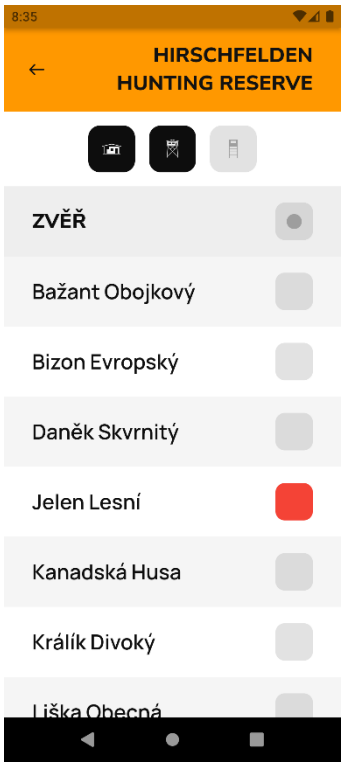


Obrázek 17 Mapa se zobrazenými body



Obrázek 16 Mapa po úplném přiblížení

Obrázek 15 Mapa po zapnutí typu aktivit



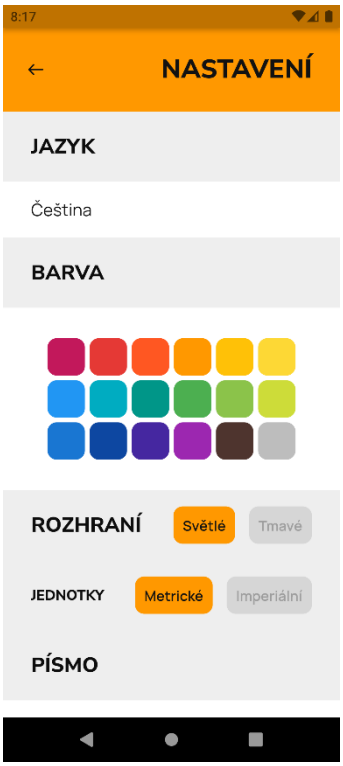
Obrázek 18 Seznam pro zobrazování bodů na mapě

6.4.7 Doplňující herní informace

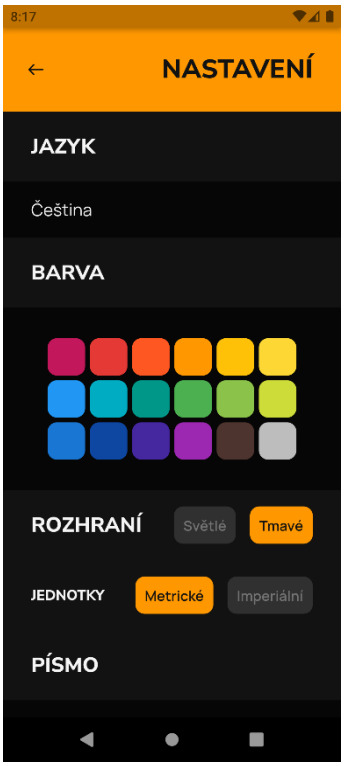
V neposlední řadě aplikace obsahuje informace ke hře, které se chovají spíše jako informace doplňující. Prozatím je do tohoto seznamu zařazen pouze seznam všech herních DLC (downloadable content, přídavný obsah pro hru). Mezi další, které tento seznam v budoucnu naplní, jsou např. informace o schopnostech lovce, možnostech vystavování skupin zvířat v trofejní boudě nebo seznam herních misí. Na tento obsah se lze dostat z hlavního menu.

6.4.8 Další důležité funkce a informace aplikace

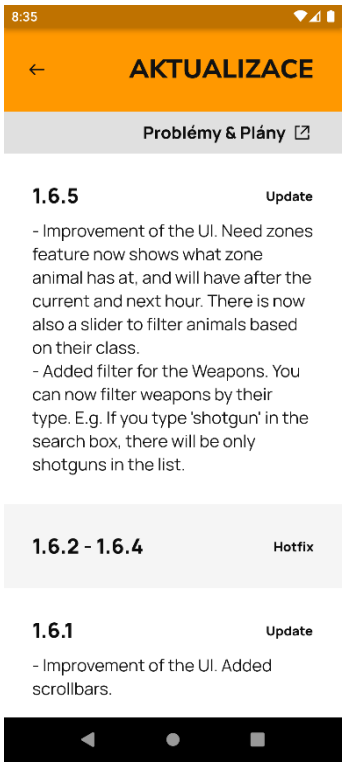
Posledními, často opomíjenými, částmi aplikace jsou nastavení, informace o aplikaci a změny aplikace podle verzí. Nastavení umožňuje celou škálu možností. Uživatel v něm dokáže nastavit jazyk nebo vzhled aplikace. Informace o aplikaci obsahují důležité základní informace o aplikaci včetně kontaktních údajů na vývojáře. Seznam změn zobrazuje změny v aplikaci podle verzí. Mimo to může uživatel využít externího odkazu na seznam chyb a budoucích plánovaných vylepšení aplikace. Ke všem těmto třem částem aplikace se lze dostat z hlavního menu.



Obrázek 19 Nastavení



Obrázek 20 Tmavý režim



Obrázek 21 Seznam změn

6.5 Uložení dat a informací

Data, obrázky a texty jsou pro Flutter uložené ve složce assets. Obsah této složky a jejích podsložek, který je pak použit v aplikaci, je nutné definovat v souboru pubspec.yaml pod parametr assets. Vzhledem k použití zabudovaných statických souborů je aplikace použitelná zcela offline. [8]

6.5.1 Herní data

Pro uložení herních dat využívá aplikace zmíněné statické datové soubory typu JSON. Soubory jsou vytvořeny pro jednotlivé informační stránky – rezervace, zvířata atd. Data jsou uložena podobně jako v databázi. Každý záznam má své ID a další informace. Pro lokalizaci jsou názvy záznamů uloženy pro každý jazyk jako jeden řádek informace o záznamu. V aplikaci jsou jednou načteny při zapnutí a využívány podle potřeby jako databáze po dobu běhu.

```
{
    "ID": 35,
    "EN": "Red Deer",
    "RU": "Благородный Олень",
    "CS": "Jelen Lesní",
    "PL": "Jeleń Szlachetny",
    "DE": "Rothirsch",
    "FR": "Cerf Èlaphe",
    "ES": "Ciervo Común",
    "PT": "Veado-vermelho",
    "JA": "アカシカ",
    "LEVEL": 6,
    "DIFFICULTY": 9,
    "SILVER": 90.5,
    "GOLD": 182.25,
    "DIAMOND": 251.07,
    "TROPHY": 272.0,
    "TROPHY_GO": 285.0,
    "WEIGHT_KG": 240.0,
    "WEIGHT_LB": 529.15,
    "WEIGHT_GO_KG": 260.0,
    "WEIGHT_GO_LB": 573.2,
    "SIGHT": 0,
    "HEARING": 0,
    "SMELL": 3,
    "DLC": 0
}
```

Ukázka kódu 1 Struktura zvířete

6.5.2 Uložení textů rozhraní aplikace

Texty rozhraní jsou také uloženy v souborech JSON. Na rozdíl od herních dat jsou texty uloženy jako nový soubor pro každý jazyk zvlášť. Tato struktura je vyžadována balíčkem easy_localization, který pak z těchto souborů načítá a předává požadovaný přeložený text widgetům.

```
{
  "about": "About",
  "animal": "Animal",
  "animal_anatomy": "Anatomy",
  "trophy_lodge": "Trophy lodge",
  "trophy_rating": "Trophy rating",
  "units": "Units",
  "version": "Version:",
  "weapon_ammo": "Ammo",
  "weapon_type": "Type",
  "weapons": "Weapons",
  "weapons_rifles": "Rifles",
  "weapons_shotguns": "Shotguns",
  "wildlife": "Wildlife",
  "yards": "yd"
}
```

Ukázka kódu 2 Struktura textů rozhraní - zkrácené

6.6 Programová část aplikace

6.6.1 Struktura aplikace

Struktura aplikace je rozdělena do několika částí. Data, obrázky a ikony aplikace popsané výše jsou, jak je zmíněno, v části assets. První programovou část tvoří widgety, které něco zobrazují a mají určité doplňující funkce. K tomu, aby mohla aplikace na těchto stránkách něco zobrazit, potřebuje k tomu data. Tyto data načítají a předávají buildery, ty tvoří druhou část struktury. Třetí část tvoří modely těchto dat. Mezi další část struktury patří programátorem vytvořené, znovupoužitelné a funkční widgety. Poslední částí jsou pomocné třídy a funkce pro práci např. s externí pamětí, vzhledem aplikace, nebo filtrací listů dat. Struktura projektu je následující:

- assets
    - graphics (složka pro obrázky a ikony)
    - fonts (složka pro písmo)
    - raw (složka pro datové soubory)
    - translations (složka pro texty rozhraní)
- lib
    - activities (hlavní widgety)
    - builders
    - miscellaneous (pomocné widgety a třídy)
    - model (modely dat)
    - widgets (znovupoužitelné widgety)
    - main (třída, inicializace hodnot aplikace, nastavení a jazyka)
- pubspec.yaml (soubor pro upřesnění verze aplikace, definice doplňků a balíčků, zpřístupnění složky assets pro využití obrázků a datových souborů)

6.6.2 Zobrazovací a funkční widgety

Tyto widgety tvoří základní vrstvu, která je zobrazována uživateli. Jedná se o widgety, které jsou zobrazeny v obrázku rozvržení v části Rozvržení aplikace. Těmto widgetům jsou předávána data, která pak zobrazují. Jejich úkolem není jen tyto data zobrazit, ale často je s těmito daty na stránce manipulováno. Proto je většina těchto widgetů stavových.

```
class ActivityAnimalInfo extends StatefulWidget {
  // předávané informace z builderu
  final int animalId;
  const ActivityAnimalInfo({
    Key? key,
    required this.animalId,
  }) : super(key: key);

  @override
  // vytvoření stavu widgetu
  ActivityAnimalInfoState createState() => ActivityAnimalInfoState();
}

class ActivityAnimalInfoState extends State<ActivityAnimalInfo> {
  late final Animal _animal;
  late final bool _imperialUnits;
  int _toggledRarity = -1;

  @override
```

```
  void initState() { // inicializace proměnných _animal a _imperialUnits }

  void _toggleRarity(int rarity) {
     // používá funkci setState, mění proměnnou _toggledRarity
  }

  // rozdělení stránky na malé widgety

  Widget _buildWidgets() { // vytvoření celého widgetu }

  @override
  Widget build(BuildContext context) => _buildWidgets();
}
```

Ukázka kódu 3 Stránka s informacemi o zvířeti

6.6.3 Buildery

Builder je prostředníkem mezi widgety. Jedná se také o widget, ale jeho jedinou funkcí je najít, vyfiltrovat, předem upravit nebo částečně změnit načítaná data a předat je widgetu, který je zobrazí. Pokud je to nutné, builder využívá widget FutureBuilder (částečně popsán v sekci FutureBuilder). Jedná se většinou o větší množství dat nebo data, která je nutná načíst předem z uložiště aplikace. Jedním z příkladů je prvotní spuštění aplikace, kdy se načítají datové JSON soubory. V průběhu načítání takových dat je uživateli zobrazena příslušná načítací animace. Doba zobrazení záleží na množství načítaných dat. Tam, kde jsou data již načtena od spuštění, není FutureBuilder nutný.

```
class BuilderLogs extends StatelessWidget {
  // definice parametrů naplňovaných v konstruktoru
  const BuilderLogs({
    Key? key,
    required this.trophyLodge,
  }) : super(key: key);

  // načítaní dat může být rychlé, pro pocit načítaní je přidané zpoždění
  Future<Widget> _forcedDelay() async {}

  Widget _buildWidgets() {
    // data se načítají asynchronně ze souboru v paměti aplikace
    // nutný FutureBuilder
    return FutureBuilder();
  }

  @override
  Widget build(BuildContext context) => _buildWidgets();
}
```

Ukázka kódu 4 Builder pro záznamník úlovků

6.6.4 Modely

Jedná se o jednoduché třídy, které umožňují příjemnější a rychlejší práci s daty. V aplikaci jsou modelové třídy využívány pro načtení a přetransformování dat z JSON souborů do modelů. Hlavním úkolem modelu je předávání dat pro zobrazení.

```
class Zone {
  // definice parametrů naplňovaných v konstruktoru
  Zone({
    required id,
    required animalId,
    required reserveId,
    required zone,
    required from,
    required to,
  })  : _id = id,
        ...

  // nutné gettery, settery
  // a další potřebné funkce modelu

  // pro transformaci dat z JSON souborů do modelů funkce fromJson
  factory Zone.fromJson(Map<String, dynamic> json) {
    return Zone(
      id: json['ID'],
      ...
    );
  }
}
```

Ukázka kódu 5 Model pro zónu potřeb

6.6.5 Pomocné widgety a třídy

Pomocné widgety v sobě ukládají mnoho funkcí, které je potřeba využít ve větším množství případů. Jedná se například o filtraci, vyhledávání a víceúrovňové řazení dat, práci s lokálními i externími soubory nebo složitější struktury, které pomáhají ukládat a pracovat specifickými daty.

```
class Settings extends ChangeNotifier {
  final List<String> _languages = [];
  final List<String> _languageCodes = [];
  // definice parametrů naplňovaných v konstruktoru
  Settings(
      {language,
       ...
    }) {
    _language = language;
    ...
  }
```

```
  // nutné gettery, settery a další potřebné funkce pomocné třídy

  // vybraná funkce pro změnu číselných jednotek
  Future<void> changeUnits() async {
    _sharedPreferences = await SharedPreferences.getInstance();
    if (_imperialUnits == true) {
      _imperialUnits = false;
      await _sharedPreferences.setBool("imperialUnits", false);
    } else {
      _imperialUnits = true;
      await _sharedPreferences.setBool("imperialUnits", true);
    }
    // oznámení o změně
    notifyListeners();
  }

  // vybraná funkce pro změnu jazyka
  Future<void> changeLanguage(int languageId) async {
    _sharedPreferences = await SharedPreferences.getInstance();
    _language = languageId;
    await _sharedPreferences.setInt("language", languageId);
    notifyListeners();
  }
}
```

Ukázka kódu 6 Pomocná třída pro nastavení aplikace

6.6.6 Znovupoužitelné widgety

Aplikace často využívá widget jednoho typu na více stránkách. Pro tyto případy jsou využívány vhodně vytvořené widgety. Jednou z hlavních výhod je stejný vzhled v celé aplikaci, stejně tak jako jeho funkčnost. V aplikaci se jedná především o tlačítka, prvky seznamů, nebo hlavičky stránek. Většina těchto widgetů je strukturována jako SLW, widgety jako například vyhledávací pole, který si pamatuje a mění svůj stav, jsou naprogramovány v podobě SFW.

```
class WidgetAppBar extends StatelessWidget {
  // definice parametrů naplňovaných v konstruktoru
  const WidgetAppBar({
    Key? key,
    // parametry widgetu
  }) : super(key: key);

  Widget _buildBackButton() {
    return WidgetButton.withIcon(
      icon: "assets/graphics/icons/back.svg",
      color: Interface.accent,
      background: Colors.transparent,
      onTap: () {
```

```
        // funkce pop zajistí opuštění momentální stránky na předchozí
        Navigator.pop(context);
      });
  }

  Widget _buildWidgets() { // vytvoření celého widgetu }

  @override
  Widget build(BuildContext context) => _buildWidgets();
}
```

Ukázka kódu 7 Bezstavový widget - hlavička stránky

```
class WidgetSearchBar extends StatefulWidget {
  // definice parametrů naplňovaných v konstruktoru
  const WidgetSearchBar({
    Key? key,
    this.color,
    this.background,
    required this.controller, // pro vyhledávací pole, mění stav widgetu
  }) : super(key: key);

  @override
  // vytvoření stavu widgetu
  WidgetSearchBarState createState() => WidgetSearchBarState();
}

class WidgetSearchBarState extends State<WidgetSearchBar> {
  Widget _buildWidgets() {
    // vytvoření widgetu
    // přiřazení kontroleru k vyhledávacímu poli
  }

  @override
  Widget build(BuildContext context) => _buildWidgets();
}
```

Ukázka kódu 8 Stavový widget - vyhledávací pole

6.7 Funkce aplikace

6.7.1 Přesměrování

Mezi nejzákladnější a nejjednodušší funkci aplikace patří přesměrování mezi stránkami. Přesměrování v aplikaci je nahrazení původně zobrazené stránky stránkou novou. Při přesměrování je nová stránka zobrazena pomocí určitého efektu. Ten závisí na typu operačního systému. Pokud v aplikaci nebude nastaveno jinak, původní stránky jsou ukládané v paměti a při zavření zobrazené stránky se v aplikaci vykreslí ta předchozí. Přesměrování zajistí třída Navigator a její funkce

push. Parametrem je pak BuildContext, který říká, kterou stránku vlastně „přepisujeme“, a objekt MaterialRoutePage, který jako parametr potřebuje builder. Při přesměrování dojde k použití předaného builderu a následně se zobrazí stránka s daty.

Uživatel má v aplikaci nejen možnost se dostat na další stránku, ale i na tu předchozí. Vrácení se zpět zajistí integrovaná navigační funkce zařízení. Druhou možností je definované tlačítko uvnitř hlavičky každé stránky. Třída Navigator a její funkce pop, která opět potřebuje parametr BuildContext, a která uživateli dovolí „vyskočit“ ze zobrazené stránky.

Přesměrování ze stránky na stránku je uživateli umožněno pomocí tlačítek s ikonami nebo textem s funkcí přesměrování. Aplikace využívá zabudovaných widgetů typu Button nebo GestureDetector. Do parametru onPressed nebo onTap je pak přidána funkce Navigatoru pro přesměrování.

```
WidgetTextIcon.withTap(
    text: tr('reserves'),
    icon: "assets/graphics/icons/reserve.svg",
    color: Interface.dark,
    onTap: () {
      Navigator.push(context, MaterialPageRoute(builder: (context) => const
BuilderRAWC(text: "reserves", type: ObjectType.reserve)));
    })
```

Ukázka kódu 9 Textové tlačítko odkazující na seznam rezervací

6.7.2 Uživatelské nastavení

Pro uložení uživatelského nastavení využívá aplikace balíček shared_preferences. Tento balíček umožňuje využít paměť aplikace pro uložení jednoduchých dat. Paměť je podobná paměti cache. Jediným rozdílem je „částečná“ perzistence dat i po vypnutí aplikace. Balíček a paměť nezajišťuje jejich úplnou perzistenci, tzn. že není dobré ukládat do této paměti důležitá data. Balíček je schopen uložit data typu int, bool, double, string a string list.

Aplikace pomocí třídy SharedPreferences a funkce getInstance přistupuje k uložišti aplikace. Poté lze do této paměti pomocí setterů nastavit hodnoty, které chce

aplikace uchovat. Ukládají se hodnoty jako primární barva, velikost písma, téma aplikace, metrické nebo imperiální jednotky, jazyk a další důležité informace, které uživatel v aplikaci nastavil.

6.7.3 Uložení větších dat

Pro uložení většího objemu dat bez využití paměti zařízení je možné využít dokumentové uložiště aplikace. Jedná se o jiné uložiště než to, které využívá aplikace pro uložení nastavení. Tato paměť je perzistentní do doby odinstalování aplikace.

Přístup do uložiště je zajištěn pomocí funkce getApplicationDocumentsDirectory z balíčku path_provider. Pro uložení stačí aplikaci specifikovat název souboru, do kterého má data uložit. Data lze samozřejmě z uložiště také číst. Všechny procesy prováděné nad tímto uložištěm se dějí asynchronně.

```
// nalezení cesty k dokumentovému uložišti
static Future<String> get _localPath async {
  final directory = await getApplicationDocumentsDirectory();
  return directory.path;
}

// v uložišti vytvořen soubor, zde soubor pro uložení záznamníku úlovků
static Future<File> get _localFile async {
  final path = await _localPath;
  return File('$path/logs.json');
}

// čtení dat ze souboru
static Future<String> _readFile() async {
  try {
    final file = await _localFile;
    final String contents;
    await file.exists() ? contents = await file.readAsString() : contents =
"[]";
    return contents;
  } catch (e) {
    return e.toString();
  }
}
```

Ukázka kódu 10 Uložení záznamníku úlovků do uložiště aplikace

6.7.4 Export a import souborů

Záznamník úlovků ukládá úlovky zadané uživatelem do souboru, který je uložen v uložišti popsaném výše. Vzhledem k velikosti souboru a potenciálně velkému počtu záznamů je na místě možnost zálohování. Popřípadě možná migrace na jiné

zařízení. Aplikace umožňuje soubor záznamníku exportovat do uložiště zařízení a následně zpět importovat z uložiště do aplikace. Uživatel je předem dotázán, zda může aplikace s externím uložištěm pracovat.

Opět aplikaci pomáhá balíček path_provider a funkce getExternalStorageDirectory. K této cestě uložiště je nutné specifikovat do jaké složky a pod jakým názvem se má soubor uložit. V tomto případě aplikace ukládá takto vyexportované soubory do složky Downloads. To, z jaké složky se má soubor zpětně naimportovat, si uživatel volí sám. Soubory jsou ukládané ve formátu JSON.

6.7.5 Filtrace a řazení dat

Velké seznamy mají v aplikaci možnost filtrace, respektive vyhledávání podle názvu záznamu. V případě záznamníku úlovků se jedná o víceúrovňovou filtraci a řazení. Obě možnosti spravuje třída HelperFilter vytvořená právě pro specifickou filtraci dat v této aplikaci. Třída HelperFilter má několik funkcí, které aplikace využívá pro filtraci seznamu dat se složitější nebo více specifickou strukturou. V aplikaci je třída použita na:

- filtraci seznamu, který obsahuje dynamická data
- filtraci zbraní podle názvu a/nebo typu
- filtraci podle názvu seznamu obsahujícího lovecké vybavení
- filtraci a řazení záznamů úlovků podle více kritérií

Třídu doplňuje ve dvou případech upravená část balíčku multi_sort dokáže seřadit list dat podle více kritérií v závislosti na jejich pořadí. V případě zbraní seřazuje aplikace seznam nejprve podle typu zbraně a poté podle názvu. Druhé místo, kde je multi_sort využit je při řazení seznamu záznamů úlovků. Tento seznam je řazen podle preferencí uživatele. Možnou preferencí je řazení podle data záznamu, hodnoty trofeje zaznamenaného úlovku, nebo názvu uloveného zvířete. Tyto preference lze zadat v jakémkoliv pořadí.

6.7.6 Specifické funkce aplikace

6.7.6.1 Mapa

Mapa každé rezervace obsahuje mnoho dat k zobrazení. Některé zařízení nemusí počet widgetů na jedné stránce zvládnout vykreslit. Mapa navíc umožňuje zoom. Vzhledem k těmto podmínkám aplikace používá balíček map. Tento balíček zajišťuje specifické chování mapy v závislosti na úrovni přiblížení. Prakticky napodobuje mapy, které jsou známé např. od Seznamu nebo Googlu.

Každá úroveň přiblížení je rozdělena na čtverce. Největší oddálení má čtverce největší. Čím více je mapa přiblížena, tím je také rozdělena na více čtverců. Každý čtverec je vždy při větším přiblížení rozdělen do čtyř. To zajistí plynulejší vykreslování, ale hlavně zamezí zbytečnému vykreslování nepotřebných, respektive neviditelných částí mapy.

Balíček používá pro zachycení bodů a ikon na mapu hodnot zeměpisné šířky a délky. Ty poté přepočítává na souřadnice na obrazovce zařízení. Je zaměřen na vykreslování zeměkoule, tj. zploštělé koule. Aplikace potřebuje vykreslit mapu, která je čtvercová, respektive koule. Pomocná třída MapProjection tudíž přepisuje přepočet šířky a délky v obou osách stejně, aby se zajistilo správné vykreslení všech bodů na mapě.

```
class MapProjection extends Projection {
  const MapProjection();

  @override
  LatLng toLatLng(TileIndex tile) {
    final x = tile.x;
    final y = tile.y;

    final lat = 360.0 * (x - 0.5);
    final lng = 360.0 * (y - 0.5);

    return LatLng(lat, lng);
  }

  @override
  TileIndex toTileIndex(LatLng location) {
    final lng = location.longitude;
    final lat = location.latitude;
```

```
    double x = (lng + 180.0) / 360.0;
    double y = (lat + 180.0) / 360.0;

    return TileIndex(x, y);
  }
}
```

Ukázka kódu 11 Přepsané metody pro výpočet souřadnic

Mapa v aplikaci má tři úrovně přiblížení. Pro každou úroveň jsou vytvořené čtvercové obrázky pozadí. První úroveň využívá 4x4, druhá 8x8 a třetí 16x16 obrázků. Podle toho, kolik se jich vejde na obrazovku, tolik se jich vykreslí. Stejný systém využívají i body na mapě. Za body na mapě se považují pozice chat, rozhleden, pozemních úkrytů a pozice výskytu specifické zvěře. Aby aplikace fungovala co možná nejplynuleji a zamezilo se sekání nebo pádu aplikace, jsou body zobrazovány s různou hustotou podle úrovně přiblížení. Nejřidší při největším oddálení, nejhustší při největším přiblížení.

6.7.6.2 Zóny potřeb

Funkce zón potřeb zobrazuje uživateli v přehledné tabulce právě probíhající aktivitu zvěře. Aktivita je závislá na zvířeti, rezervaci a času. Aplikace umožňuje nastavit čas a rezervaci. Seznam zvířat je pak vykreslen společně s typem aktivity. Čas poté automaticky postupuje a zobrazuje tak změnu aktivit bez dalšího zásahu uživatele.

```
class ActivityNeedZones extends StatefulWidget {
  const ActivityNeedZones({
    Key? key,
  }) : super(key: key);

  @override
  // tento widget mění zobrazení aktivit v čase, musí být stavový
  ActivityNeedZonesState createState() => ActivityNeedZonesState();
}

class ActivityNeedZonesState extends State<ActivityNeedZones> {
  // definice potřebný proměnných
  @override
  void initState() {
    // balíček wakelock umožňuje „vypnout“ usínání obrazovky
    Wakelock.enable();
    _timer = RestartableTimer(Duration(microseconds: 100000), () =>
_changeTime());
    super.initState();
  }
```

```
  @override
  void dispose() {
    // usínání je vypnuto pouze pro tento widget, po jeho opuštění je usínání
obrazovky opět zapnuto
    Wakelock.disable();
    super.dispose();
  }

  // změna a pozastavení času, minuta ve hře je reálných ~15 vteřin
  void _changeTime() {
    setState(() {
      if (!_stopped) {
        _timer.reset();
        _second += 1;
        if (_second == 15) {
          _second = 0;
          _minute += 1;
          if (_minute == 60) {
            _minute = 0;
            _hour += 1;
            if (_hour == 24) {
              _hour = 0;
            }
          }
        }
      }
    });
  }

  // předání informací builderu, který poté zobrazí tabulku
  Widget _buildNeedZones() {
    return
      BuilderReserveNeedZones(
        reserveId: _reserveId,
        hour: _hour,
        min: _min,
        max: _max,
        compact: _compact,
        classSlider: _classSlider,
      );
  }

  @override
  Widget build(BuildContext context) => _buildWidgets();
}
```

Ukázka kódu 12 Funkce zón potřeb

6.7.6.3 Záznamník úlovků

Záznamník úlovků je pravděpodobně nejsložitější části celé aplikace. Záznamník umožňuje uživateli ukládat záznamy o úlovcích. Každý vytvářený záznam se skládá z času, rezervace, zvířeti, pohlaví, typu srsti, trofejního skóre, váhy a tzv. kontroly

úlovku. Zvíře je závislé na rezervaci, typ srsti na zvířeti a pohlaví, a trofejní skóre a váha na zvířeti. Záznam lze navíc vytvářet z více míst v aplikaci.

Seznam záznamů lze filtrovat a řadit. Filtraci umožňuje vyhledávací pole pod hlavičkou stránky. Tato filtrace umožňuje využití více parametrů najednou. Parametry musí být odděleny svislou čárou a musí mít náležitou strukturu. Řazení je pak uživateli umožněno pomocí tlačítek ve spodním menu. Řadit lze pomocí tří kritérií v jakémkoliv pořadí.

Ze seznamu lze záznamy také odstranit, jednotlivě nebo všechny najednou. Možnost zálohování nebo migrace je umožněna exportem a importem celého seznamu do paměti zařízení. Vzhledem k tomu, že se jedná o velmi složitou funkční část aplikace, je uživateli umožněno nahlédnout do pomocné stránky s návodem, jak funkci používat.

7 Testování

Pro testování aplikace je využíváno emulátorů. Aplikace je publikována pro zařízení Android a iOS. Je tedy nutné testovat aplikaci jak na telefonu, tak tabletu při různých rozlišeních. V testování pomáhá hlavně tzv. hot-reload, popsaný v sekci Hot-Reload. Aplikaci je možné za běhu změnit a použít zmíněný hot-reload. Aplikace se neinstaluje na zařízení znovu, ale pouze se opakovaně spustí se změněnými hodnotami. To umožňuje rychlé změny v grafickém rozhraní nebo změny hodnot funkcí. [8]

Pro složitější problémy je nutné využít debugger. Debugger umožňuje nahlédnout do stavu paměti a ovládat kroky funkcí. Dokáže tak řešit problémy, které nejsou na první pohled zjevné. Vzhledem k tomu, jak moc silná je možnost hot-reloadu, debugger jako takový je zapotřebí jen velmi zřídka.

8 Publikování a ukázka

Jak již bylo zmíněno, aplikace je publikována pro operační systémy Android a iOS. Oba systémy mají své vlastní obchody, kde lze aplikaci najít pod názvem COTW Companion.

8.1 Android, Google Play

Celá práce a aplikace je napsána na zařízení se systémem Windows. To umožňuje jednoduché vytvoření a sestavu aplikace pro Android. Stejně tak jeho nahrání na obchod Google Play. Pro publikování aplikace na Google Play je nutností developerský účet Google. Za tento účet je účtován jednorázový poplatek. Účet pak umožní vytváření a sdílení aplikací pro systémy Android skrz obchod Google Play.

Pro sdílení aplikace je nutné vytvořit si unikátní certifikát a klíč. Ty jsou pak používány pro ověření aplikace. U aplikace je také nutné specifikovat všechny potřebné požadavky, které aplikace bude využívat a které musí uživatel instalující aplikaci odsouhlasit. Tato aplikace používá přístup k internetu a přístup k externímu uložišti.

Pro sdílení aplikace na Google Play je nutností vytvoření aplikační ikony, náhledů aplikace pro telefon a tablet, popis a balíček aplikace. Google po vývojáři také vyžaduje vyplnění důležitých formulářů tykajících se například věkového omezení. Zmíněný balíček lze vytvořit pomocí příkazů frameworku Flutter, který se postará o správnou transformaci kódu a zabalení aplikace do balíčku. Balíček je pak nahrán pomocí rozhraní pro ověření. Ověření trvá v řádu hodin. Pokud se jedná o malou aktualizaci, tak i minut. Poté lze aplikaci ručně nebo automaticky vypustit na Google Play.

8.2 iOS, App Store

Vzhledem k nemožnosti přístupu k zařízení společnosti Apple, je publikování aplikace pro systém iOS relativně složitější. Stejně jako u Googlu, je i pro publikování aplikací pro iOS nutný developerský účet. Za tento účet je účtován roční poplatek. Účet pak umožňuje podobné funkce jako ten od Googlu.

Pro sdílení aplikace je využito Codemagic. Codemagic je prostředník mezi aplikací na systému Windows a aplikací na App Store. Umožňuje vytváření aplikací např. pro Android, iOS, nebo PC bez nutnosti zařízení k tomu určenému.

Stejně jako na Googlu, ke sdílení aplikací je nutný certifikát a klíč, který je použit pro ověření aplikace. Stejně jako u aplikace pro Android je v některých případech nutná specifikace některých funkcí, které aplikace využívá a ke kterým potřebuje souhlas uživatele.

Pro vytvoření balíčku aplikace pomocí Codemagic, je nutné ji nahrát na jeden z repositářů. Tato aplikace je nahrána na GitHub. Codemagicu poté stačí říct, kde se aplikace nachází, jaký má použít klíč a certifikát a kam aplikaci nahrát. Codemagic následně imituje sestavu aplikace jako na zařízení Apple. Vytvoří příslušné soubory a ty pak odešle na základě preferencí na účet, ze kterého se má aplikace nahrát na App Store. [9]

Pro sdílení aplikace na App Store je nutností vytvoření aplikační ikony, náhledů aplikace pro telefon a tablet, popis a balíček aplikace. Navíc App Store potřebuje další detailní informace o verzi, autorovi atp. Apple aplikaci trvá ověřit výrazně déle než na Googlu. Největším faktorem je bezpečnost. Ověření trvá v řádu dní, záleží na vytíženosti a velikosti aktualizace.

9 Zpětná vazba uživatelů

9.1 Recenze

Recenze aplikace jsou nedílnou součástí aplikačního vývoje. Po vydání první verze aplikace nějakou chvíli trvá, než uživatelé začnou sdílet recenze. Můžou to být dny, nebo i měsíce, záleží na rozšířenosti a povědomí o aplikaci. Tato aplikace vyšla bez jakéhokoliv předběžného upozornění nebo jiného způsobu uvědomění uživatelů. Tudíž první recenze přicházely až po nějaké době. Recenze jsou první zpětnou vazbou, kterou vývojář dostane. Zprvu to jsou pouze čísla. Textové recenze jsou ty,

na které je důležité se zaměřit. Chyby, nedostatky, ale i pochvaly a vyzdvihování některých funkcí je to, co začalo tuto aplikaci postupně posouvat určitým směrem.

9.2 Soukromé zprávy

Soukromé zprávy jsou jakousi lepší formou recenze. Uživatel je schopen popsat svůj problém hlouběji a podrobněji (včetně např. obrázků), než s omezeným počtem písmen v recenzi. Navíc dá vývojáři stejným způsobem možnost rozebrat podrobněji řešení problému. Soukromé zprávy také byly jedním z prvků, které zajistili rozšíření aplikace o nové jazyky a také úpravu a korektnost stávajících překladů.

9.3 Řešení problémů

Přestože můžou být některé recenze a zprávy velmi pozitivní, najdou se i takové, které jsou velmi negativní. Negativita vzkvétá většinou z prvku v aplikaci, který se uživateli nelíbí nebo mu nefunguje, respektive ho nechápe. V takových situacích je důležité najít takové řešení, které pomůže uživateli, který má problém a nepoškodí uživatele, který s danou věcí problém nemá.

10 Shrnutí výsledků

V rámci práce byla za pomocí technologie Flutter vyvinuta multiplatformní aplikace COTW Companion. Po předchozím srovnání technologií, byla technologie Flutter vybrána jako nejvhodnější vzhledem k požadavkům aplikace. K vývoji aplikace pro systémy Android a iOS bylo využito prostředí Android Studio a Codemagic. Aplikace je nahrána a spravována na uložišti GitHub. Je přehledná, rychlá a obsahuje úměrné množství informací a funkcí, které jsou věrohodné a nezavádějící. Aplikace je k dispozici v obchodech Google Play a App Store.

11 Závěry a doporučení

V rámci práce byla za pomocí technologie Flutter vyvinuta multiplatformní aplikace COTW Companion. Po předchozím srovnání technologií, byla technologie Flutter vybrána jako nejvhodnější vzhledem k požadavkům aplikace. K vývoji aplikace pro systémy Android a iOS bylo využito prostředí Android Studio a Codemagic. Aplikace je přehledná, rychlá a obsahuje úměrné množství informací a funkcí, které od ní potencionální uživatel čeká. Aplikace je k dispozici v obchodech Google Play a App Store. Tato práce se zabývala volbou nejvhodnější technologie pro vývoj multiplatformní mobilní aplikace jejím následným vývojem. Aplikace COTW Companion je doprovodnou aplikací ke hře theHunter: Call of the Wild, která je loveckým simulátorem.

První část práce popisovala problematiku mobilní platforem, aplikací a byly popsány nejrozšířenější operační systémy Android a iOS. V další části se práce zabývala porovnáním technologií pro vývoj multiplatformních aplikací. Vybranou technologií byl, díky své škálovatelnosti, podpoře multiplatformního vývoje a rychlé možnosti vývoje, vybrán Flutter. Flutter má velké množství zabudovaných komponent, ke kterým vede velmi rozšířenou a detailní dokumentaci, včetně videí a ukázek kódu. Flutter dokázal naplnit požadavky dané aplikací a zjednodušit její vývoj hlavně díky veřejně dostupným propracovaným balíčkům.

Aplikaci je, díky zvolené technologii, možné dále rozšiřovat a aktualizovat. Vhodně vytvořené komponenty umožňují snadné znovupoužití, a tak i snadné rozšíření o další funkcionality. Aplikace byla otestována na zařízeních Android i iOS. Pro oba systémy je nyní k dispozici na Google Play a App Store. Aplikace se doposud pyšní velkou oblibou a je považována mnohými uživateli za zpříjemnění herního zážitku.

12 Seznam použité literatury

[1] CHENG, Fu. State Management. In: CHENG, Fu. Flutter Recipes [online]. Berkeley, CA: Apress, 2019, 2019-10-11, s. 365-412 [cit. 2023-04-20]. ISBN 978-1-4842-4981-9. Dostupné z: doi:10.1007/978-1-4842-4982-6_10

[2] SLEPNEV, Dmitrii. State management approaches in Flutter [online]. 2020 [cit. 2023-04-20]. Dostupné z: https://urn.fi/URN:NBN:fi:amk-2020121829659. Bachelor's thesis. South-Eastern Finland University of Applied Sciences. Vedoucí práce Timo Hynninen.

[3] BOUKHARY, Shady a Eduardo COLMENARES. A Clean Approach to Flutter Development through the Flutter Clean Architecture Package. In: 2019 International Conference on Computational Science and Computational Intelligence (CSCI) [online]. IEEE, 2019, 2019, s. 1115-1120 [cit. 2023-04-20]. ISBN 978-1-7281-5584-5. Dostupné z: doi:10.1109/CSCI49370.2019.00211

[4] PAYNE, Rap. Beginning App Development with Flutter [online]. Berkeley, CA: Apress, 2019 [cit. 2023-04-20]. ISBN 978-1-4842-5180-5. Dostupné z: doi:10.1007/978-1-4842-5181-2

[5] ZAMMETTI, Frank. Flutter: A Gentle Introduction. In: ZAMMETTI, Frank. Practical Flutter [online]. Berkeley, CA: Apress, 2019, 2019-07-20, s. 1-36 [cit. 2023-04-20]. ISBN 978-1-4842-4971-0. Dostupné z: doi:10.1007/978-1-4842-4972-7_1

[6] WU, Wenhao. React Native vs Flutter, Cross-platforms mobile application frameworks [online]. [cit. 2023-04-20]. Dostupné z: https://urn.fi/URN:NBN:fi:amk-201805158156*. Thesis. Metropolia University of* Applied Sciences. Vedoucí práce Kari Salo.

[7] BAREA, Abimael, Xavier FERRE a Lorenzo VILLARROEL. Android vs. iOS Interaction Design Study for a Student Multiplatform App. In: STEPHANIDIS, Constantine, ed. HCI International 2013 - Posters' Extended Abstracts [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, 2013, s. 8-12 [cit. 2023-04-24]. Communications in Computer and Information Science. ISBN 978-3-642-39475-1. Dostupné z: doi:10.1007/978-3-642-39476-8_2

[8] Flutter Documentation [online]. [cit. 2023-04-20]. Dostupné z: https://docs.flutter.dev

[9] Codemagic Documentation [online]. [cit. 2023-04-20]. Dostupné z: https://docs.codemagic.io

[10] SRIVASTAVA, Sudeep. Top 10 Best Cross-Platform App Development Frameworks. Appinventiv [online]. 2023 [cit. 2023-04-20]. Dostupné z: https://appinventiv.com/blog/cross-platform-app-frameworks/

[11] TAYLOR, Petroc. Mobile operating systems' market share worldwide from 1st quarter 2009 to 4th quarter 2022. Statista [online]. 2023 [cit. 2023-04-20]. Dostupné z: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[12] Wikipedia contributors. IOS. Wikipedia [online]. Wikipedia, The Free Encyclopedia, 2023 [cit. 2023-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=IOS&oldid=1151495035

[13] Wikipedia contributors. Android. Wikipedia [online]. Wikipedia, The Free Encyclopedia, 2023 [cit. 2023-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=1151338650

[14] Wikipedia contributors. Dart. Wikipedia [online]. Wikipedia, The Free Encyclopedia, 2023 [cit. 2023-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Dart_(programming_language)&oldid=1145322174

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **Jan Stehno**

Studium: I1900255

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Vývoj multiplatformní mobilní aplikace s využitím frameworku Flutter**

Název bakalářské práce AJ: Development of a multi-platform mobile application using the Flutter framework

Cíl, metody, literatura, předpoklady:

Cílem práce je prozkoumat možnosti vývoje multiplatformních mobilních aplikací. Vybrané technologie porovnat podle vybraných kritérií. V praktické části práce navrhnout a vyvinout vlastní multiplatformní aplikaci s využitím frameworku Flutter.

Osnova:

- Úvod
- Vývoj multiplatformních mobilních aplikací
- Porovnání vybraných frameworků
- Flutter
- Vývoj multiplatformní mobilní aplikace
- Shrnutí výsledků
- Závěry a doporučení
- Seznam použité literatury

1. PAYNE, Rap. *Beginning App Development with Flutter* [online]. Berkeley, CA: Apress, 2019 [cit. 2022-08-29]. ISBN 978-1-4842-5180-5. Dostupné z: doi:10.1007/978-1-4842-5181-2
2. IŞITAN, Mehmet a Murat KOKLU. Comparison and Evaluation of Cross Platform Mobile Application Development Tools. *International Journal of Applied Mathematics Electronics and Computers* [online]. 273-281 [cit. 2022-08-29]. ISSN 2147-8228. Dostupné z: doi:10.18100/ijamec.832673
3. CHEON, CC Yoonsik; CHAVEZ, Carlos. Creating Flutter Apps from Native Android Apps. Department of Computer Science The University of Texas at El Paso, El Paso, Texas, 2020.

Zadávající pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: Ing. Jakub Beneš

Datum zadání závěrečné práce: 15.10.2021