VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTERAKTIVNÍ PRŮVODCE PRO ZOOLOGICKÉ ZAHRADY

INTERACTIVE GUIDE FOR ZOOLOGICAL GARDENS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN MACHÁČEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Macháček Martin**
Program: Informační technologie
Název: **Interaktivní průvodce pro zoologické zahrady**
Interactive Guide for Zoological Gardens
Kategorie: Informační systémy

Zadání:

1. Prostudujte problematiku prezentace průvodních informací návštěvníkům zoologických zahrad. Prozkoumejte současné možnosti, které mají zoologické zahrady k dispozici pro daný účel a zhodnoťte jejich výhody a nevýhody.
2. Prostudujte principy tvorby použitelných informačních systémů a mobilních aplikací. Zaměřte se možnosti tvorby mapových aplikací.
3. Analyzujte požadavky zoologických zahrad na snadnou tvorbu průvodních prezentací. Zohledněte požadavky jejich návštěvníků.
4. Navrhněte informační systém, který umožní pracovníkům zoologických zahrad snadno vytvářet průvodce danou zoologickou zahradou s využitím mapy zobrazitelného na mobilním telefonu.
5. Navržené řešení implementujte. Klaďte důraz na jednoduchost a uživatelskou přívětivost.
6. Výsledný systém otestujte ve spolupráci s vybranou zoologickou zahradou. Zhodnoťte výsledky.

Literatura:

- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers/Elsevier, 2010, ISBN: 9780123750303.
- Preece, J., Sharp, H. a Rogers, Y.: *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 2015, ISBN: 9781119020752.
- Leaflet: *Leaflet API reference* [online]. 2019 [cit. 2020-09-16]. Dostupné z: https://leafletjs.com/reference-1.7.1.html

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hynek Jiří, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 23. října 2020

Abstrakt

Velké množství zoologických zahrad spadá pod unii CSZOO. I přesto doposud neexistoval sjednocený způsob správy těchto informací a jejich prezentace návštěvníkům. Tato práce řeší tento problém vytvořením sjednocujícího informačního systému pro jakoukoliv zoologickou zahradu, spolu s jednotnou, moderní mobilní aplikací, sloužící jako kapesní průvodce a přehledný zdroj informací ze systému. Mimo funkcionality jako kalendář událostí, zvířecí druhy, expozice nebo oznámení je velmi důležitou součástí interaktivní mapa. Vytořený informační systém je spolu s veřejnou bezstavovou API založen na frameworku Lumen. Konečná mobilní aplikace je multiplatformní a založena na technologii Cordova. Webové rozhraní informačního systému i mobilní aplikace využívají pro frontend architekturu framework Vue.js. Interaktivní mapa je realizována skrze knihovnu Leaflet v kombinaci s technologiemi Mapbox. Vytvořená řešení figurují jako základní stavební kameny pro budoucí expanze.

Abstract

Even though a large number of zoological gardens fall under CSZOO union, there has been no unified way of presenting and managing this information. This project solves this problem by creating a unifying information system for any zoological garden, together with a modern mobile application, which serves as a pocket guide and a clear source of information from this system. In addition to functionalities such as events calendar, species, exhibitions or announcements, an interactive map is a very important part. Implemented system together with public stateless API is based on the Lumen framework. Mobile application is cross-platform and is based on Cordova technology. In order to establish certain form of frontend architecture, both the information system web interface and mobile application use framework Vue.js. Interactive maps are implemented through library called Leaflet in combination with Mapbox technologies. The created solutions exist as basic building blocks for future expansions and development.

Klíčová slova

zoo, zoologická zahrada, informační systém, sjednocující systém, webová aplikace, mobilní aplikace, interaktivní mapa, Lumen, Cordova, Vue.js, bezstavová API, Leaflet, SASS, MongoDB

Keywords

zoo, zoological garden, information system, unifying system , web application, mobile application, interactive map, Lumen, Cordova, Vue.js, stateless API, Leaflet, SASS, MongoDB

Citace

MACHÁČEK, Martin. *Interaktivní průvodce pro zoologické zahrady.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek, Ph.D.

Interaktivní průvodce pro zoologické zahrady

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Hynka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Macháček
9. května 2021

Poděkování

Děkuji vedoucímu práce panu Ing. Jiřímu Hynkovi, Ph.D. za jeho odborné vedení, cenné rady, věcné připomínky a kladnou oporu během vypracovávání této práce. Chci taktéž poděkovat své rodině a blízkým za jejich nenahraditelnou podporu.

Obsah

Kapitola 1

Úvod

Tato práce vznikla za účelem vylepšení prezentace informací návštěvníkům zoologických zahrad a k zavedení vhodného informačního systému pro správu těchto informací. Způsob a kvalita sdělování informací, ať už navigačních, nebo třeba vzdělávacích, je velice důležitou součástí celkového zážitku návštěvníka. Drtivá většina zoologických zahrad (dále jen zoo) spoléhá na plánky nebo informační tabule. Velmi často se stává, že spousta informací se u těchto způsobů ztrácí, plánek není dostatečně přehledný, nebo v něm návštěvník nenajde to, co hledá, ať už se jedná o navigaci uvnitř zoo, či plánované časy událostí. Nesmíme opomenout fakt, že zoo neslouží jen jako způsob pobavení návštěvníků, ale nesou i určité ochranářské poslání, které mohou naplňovat různými způsoby.

Rychlý vývoj mobilních telefonů a informačních technologií nabádá k jejich využití, a to nejen pro řešení existujících problémů, ale i k vytvoření systému pro efektivní přístup a správu informací. Pokud se práce bude zaměřovat na tyto technologie, je také potřeba zdůraznit, že většina zoo, ať už velkých nebo menších, má relativně špatné zkušenosti s již existujícími řešeními. Externí firmy, které tyto aplikace nabízí, si většinou účtují peníze za vytvoření obyčejné, vzhledem neatraktivní, statické aplikace, která neposkytne návštěvníkům důležité ani zajímavé, a v některých případech, ani užitečné informace. Návštěvníci si navíc ještě musí často instalovat do každé zoo aplikaci zvlášť.

Cílem této práce je tedy návrh a tvorba sjednocujícího informačního systému, který bude poskytovat zaměstnancům efektivní rozhraní pro správu informací v jejich zoo. Dále je cílem návrh a implementace moderní mobilní aplikace, která bude čerpat data ze zmiňovaného systému. Aplikace bude také poskytovat důležité a zajímavé informace návštěvníkům a sloužit jako kapesní průvodce v zoo.

Práce je rozdělena do několika kapitol. Kapitola 2 popisuje hlavní způsoby prezentace informací návštěvníkům, ukazuje konkrétní příklady a okrajově je hodnotí. Kapitola 3 se věnuje procesu vytváření vhodného informačního systému spolu s existujícími možnostmi pro tvorbu klientských aplikací. Taktéž představuje možnosti tvorby interaktivních map v těchto aplikacích. Kapitola 4 je věnována provedení průzkumu trhu a analýze požadavků nejen zoo, ale hlavně i návštěvníků. Kapitola 5 je zaměřena návrhu systému, diagramů a databázových schémat, prezentuje také grafické návrhy uživatelského rozhraní jak informačního systému, tak mobilní aplikace. Implementaci celého systému včetně administrátorského rozhraní a mobilní aplikace je věnována kapitola 6. Kapitola 7 je soustředěna na testování a zmiňuje plán do budoucna. Závěrečná kapitola 8 obsahuje shrnutí této práce, implementovaných řešení a dosažených cílů.

Kapitola 2

Existující možnosti pro prezentaci informací návštěvníkům

Tato kapitola se nejdříve v sekci 2.1 a 2.2 věnuje krátkému popisu významu dat a informací v zoo. V sekcích 2.3 a 2.4 se poté zabývá představením možností, které zoo aktuálně mají k dispozici a které využívají k prezentaci informací návštěvníkům. Taktéž nabízí ukázku a krátké zhodnocení konkrétních příkladů.

2.1 Účel informací v zoo

Pojem informace je používán v mnoha oborech a disciplínách. Podle knihy [17] ho lze chápat jako obyčejná data v určitém kontextu s vhodnou formou reprezentace. V rámci zoo se jedná o nezbytnou součást zvyšující zážitek z celé návštěvy. Informace, které zoo poskytují, nemusí sloužit pouze k obeznámení s příběhy zvířat, ale také mohou zpřehlednit orientaci uvnitř zoo prostřednictvím předpřipravených tras, lokací jednotlivých restaurací, expozic, toalet, obchůdků. Také mohou sdělovat historii zoo, aktuality nebo zajímavé události.

2.2 Způsoby sdělování informací

Zoologické zahrady v současné době používají tři základní způsoby prezentace dat.

- **Mapová vizualizace** – v případě, že chce zoo svým návštěvníkům sdělit například lokality jednotlivých zvířat, musí využít efektivního způsobu pro sdělení těchto lokalit. V takovéto situaci je vhodná mapová vizualizace, dokáže totiž velice jednoduše, přehledně a hlavně rychle návštěvníkům lokalizační data prezentovat.
- **Tištěný text** – tento způsob je vhodný tehdy, když zoo potřebuje sdělit například příběh, nebo pokud se jedná o data, která nelze jednoduše prezentovat ostatními způsoby. Mohou se také prezentovat data otevírací doby, kontaktů a mnoho dalších.
- **Verbální sdělení** – v určitých případech je nejlepší pro transformaci dat pouhá verbální komunikace (mluvené slovo). Tato možnost se projevuje v situaci, při které je příliš náročné nebo neefektivní data prezentovat tištěným textem či mapovou vizualizací. Konkrétním příkladem této situace může být organizovaná prohlídka.

Tyto způsoby prezentace jsou v zoo využívány jak ve fyzické 2.3, tak v elektronické 2.4 formě.

2.3 Fyzické nosiče informací

Fyzické nosiče se v zoo využívají již po několik desetiletí, tvoří základní bloky pro prezentaci jakéhokoliv druhu informací. Patří mezi ně informační tabule, plánek a nebo speciální kategorie – komentovaná prohlídka, kde je nosičem označován samotný průvodce.

Informační tabule

Na informačních tabulích bývají uvedené obecné informace o oblasti, ve které je tabule umístěna a novinky související s touto oblastí. V některých případech na těchto tabulích může být umístěna i mapa. Naskytají se dva hlavní problémy, a tedy náročnost udržení aktuálnosti informací a množství prostředků, které by mohlo být soustředěno jiným směrem (ať už se jedná o čas, pracovní sílu, či peníze).

I přesto, že obecné informace o oblasti jsou napříč roky relativně stejné, tak při jakékoliv změně, například porod mláděte nebo jiné události v dané oblasti, musí zaměstnanci tisknout úplně nový list a jít mezi návštěvníky plakát na tabuli vyměnit. Tyto plakáty, někdy i celé tabule, bývají často zanedbané a vzhledově ne nijak poutavé (například tabule na obrázku 2.1).

Obrázek 2.1: Konkrétní příklad informační tabule (Zoo Liberec)[1].

Plánek zoo

Informační plánky mají za úkol návštěvníkům co nejjednodušeji představit danou zoo, poskytnout přehlednou mapu s lokacemi jednotlivých zvířecích druhů, expozicí, stánků apod. Také by měly sloužit jako kapesní průvodce v průběhu návštěvy. Tyto plánky většinou bývají dostupné u vchodu, v některých zoo v ceně vstupenky, ovšem najdou se i takové zoo, které ho nabízí za malý poplatek. Podobně jako u tabulí se zde naskytá problém aktuálnosti, ale také je nutno uvést, že jakmile návštěvník zoo opouští, tak tento plánek většinou ihned vyhazuje, což vede k velkému množství zbytečného odpadu. Tyto plánky mohou být

[1]Obrázek převzat z https://zena-in.cz/clanek/zoo-liberec-nenechte-si-ujit-unikatni-bile-tygry

velice nepřehledné, a to hlavně v případě, kdy zoo má mnoho zvířat a dobrovolných aktivit (obrázek 2.2).

Obrázek 2.2: Příklad plánku v Zoo Liberec[2].

Komentovaná prohlídka

Jedná se o průvodcovskou službu, díky které mají návštěvníci možnost poznat zoo z jiného, nového úhlu. Mohou se pod vedením zkušeného průvodce dozvědět vztahy mezi druhy zvířat, příběhy a další zajímavosti. V některých zoo tato služba musí být domluvena předem, její cena se odvíjí primárně podle délky prohlídky, rozsahu a kvality informací. Tento způsob získávání informací může být často velice zajímavý, ovšem ne všechny zoo ho nabízí a poplatek za tuto službu nemusí některým návštěvníkům vyhovovat.

2.4 Digitální nosiče informací

Nosiče digitální podoby se naskytly spolu s technologickým pokrokem, a to konkrétně webových a mobilních technologií. Současně se s tímto rozvojem totiž objevují příležitosti, díky kterým mají zoo mnohem větší potenciál na zlepšení efektivity prezentace informací, kvalitnější strukturalizaci a také zmenšení nákladů na správu. Množství informací, které zoo mohou prezentovat, již není omezeno velikostí plánku nebo informační tabule.

Z pohledu potenciálního návštěvníka tyto možnosti poskytují mnohem větší pohodlí. Dohledání žádoucích informací může být nyní otázka několika vteřin, což se ale v některých případech může stát i nevýhodou, neboť nadměrné množství dostupných informací a případné zavedení bezplatných virtuálních prohlídek nijak člověka nemotivuje k samotné návštěvě zoo.

[2]Obrázek převzat z https://www.zoo-ostrava.cz/cz/zoo/novinky/1172-aktualizovane-vydani-tistene-mapy-zoo

Webové stránky

Tato možnost je v dnešní době standardem pro každou zoo. Mohou využívat webové stránky nejen pro jednoduchou prezentaci základních informací jako otevírací doby, lokace, ceny vstupného, seznam zvířat, ale také pro již zmíněnou virtuální prohlídku, online nákup vstupenek, nebo skladiště oficiálních dokumentů, událostí, historie a mnoho dalších.

Uvádí se, že pro utvoření prvního názoru o webové stránce stačí pouhých 50ms [19], s tím, že tento názor je z 94% založený na jejím designu, nehledně na kvalitu informací. Mnoho zoo již adaptovalo modernizaci webové prezentace, jedná sa ovšem o službu, která je v drtivé většině případů finančně i časově náročná, a tedy pro vybrané zoo téměř nedosažitelná.

Na obrázcích 2.3 a 2.4 lze vidět rozdíly v aktuálnosti vzhledu a atraktivity webové prezentace Zoo v San Diegu a Zoo Brno. Z tohoto porovnání je ze strany Zoo Brno zjevná určitá zastaralost designu, naopak web Zoo San Diego disponuje moderním vzhledem, jež dokáže potenciálního návštěvníka dostatečně na první pohled zaujat.

Obrázek 2.3: Úvodní stránka webu San Diego Zoo[3].

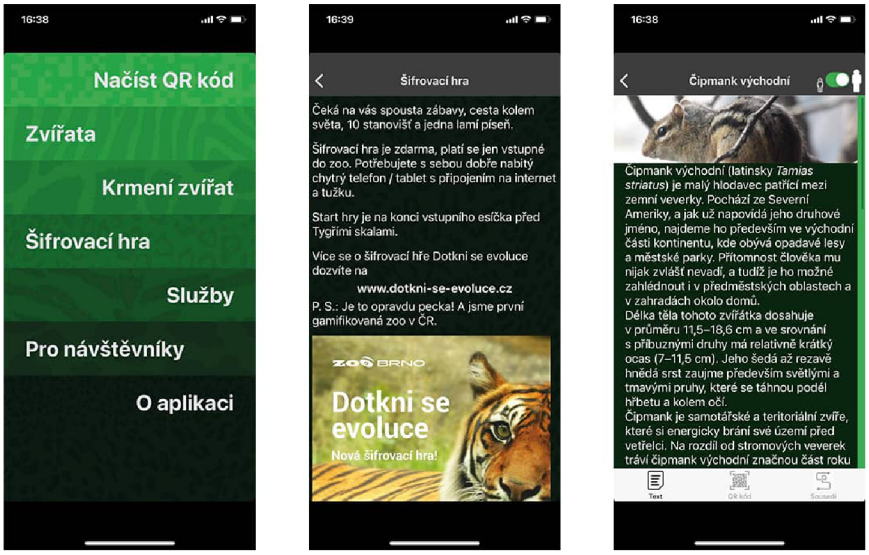
Obrázek 2.4: Úvodní stránka webu Zoo Brno[4].

[3]Webové stránky San Diego Zoo https://zoo.sandiegozoo.org/

Mobilní aplikace

Mobilní aplikace se dostávají do popředí až posledních pár let, nabývají ovšem čím dál více popularity [14]. Mobilní aplikace mohou lidem sloužit jako prostředek pro komunikaci, zábavu, výuku, organizaci, vyplnění volného času, a mnoho dalšího. Většina zoo svým návštěvníkům aktuálně mobilní aplikace neposkytuje, ať už z peněžních či jiných důvodů. Pro jejich krátké představení a uvedení do stávající situace tato práce používá mobilní aplikaci Zoo Brno, Safari Park Dvůr Králové a NaviZoo.

Mobilní aplikace Zoo Brno (obrázek 2.5) disponuje několika funkcemi, patří mezi ně informace o zvířatech, čtečka QR kódů pro rychlé získání informací uvnitř zoo, časy krmení, šifrovací hra a jednoduchý popis služeb. Aplikace je velmi jednoduchá, hlavní kategorie jsou vyobrazené přímo po spuštění. V čem tato aplikace ovšem zaostává, je nepochybně design, taktéž rozsah aplikace a také celková užitečnost. Návštěvník nemá důvod si aplikaci stáhnout, protože všechny informace, které aplikace nabízí, jsou dostupné ve stejně statické podobě i na webových stránkách.



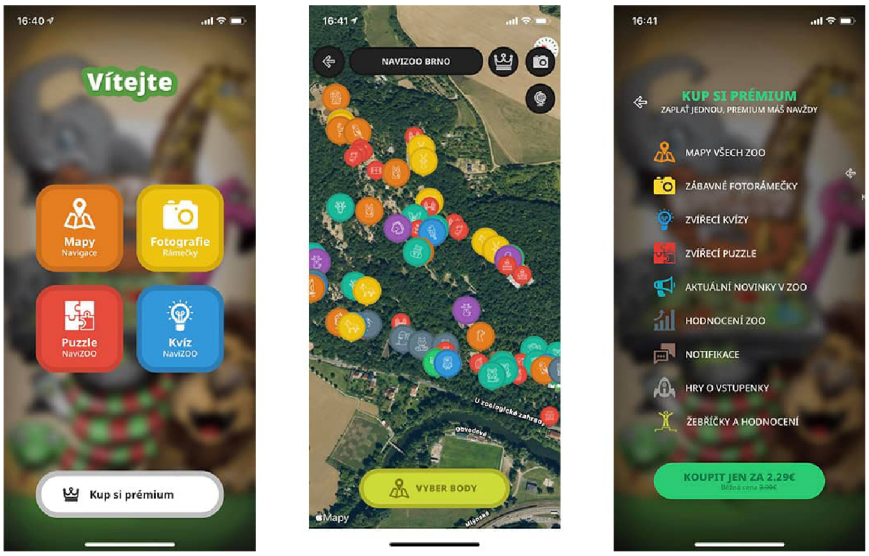
Obrázek 2.5: Mobilní aplikace Zoo Brno – zoobrno[5].

Aplikace Safari Park Dvůr Králové (obrázek 2.6) využívá k prezentaci informací svým návštěvníkům primárně interaktivní mapu. Aplikace si drží svůj osobitý vzhled, který je jednoznačně inspirován vzhledem webových stránek[6] této zoo. Bohužel, až na interaktivní mapu a krátký popis zvířat tato aplikace více nenabízí. Stručně řečeno, jedná se o malou aplikaci, která ale pomocí interaktivní mapy dokáže sdělit více, než deset stran neudržovaného textu.

[4]Webové stránky Zoo Brno https://www.zoobrno.cz/
[5]Zoo Brno aplikace https://play.google.com/store/apps/details?id=com.zoobrno
[6]Webové stránky Safari Park Dvůr Králové https://safaripark.cz/

Obrázek 2.6: Mobilní aplikace Dvůr Králové – Safari Park Dvůr Králové[7].

Aplikace NaviZoo (obrázek 2.7) se liší od předchozích dvou, a to tím, že není limitovaná na jednu konkrétní zoo. Díky tomu, že uživatel nemusí stahovat do každé zoo aplikaci zvlášť, se nyní naskytuje spousta možností pro sjednocení a univerzální prezentaci informací v jakékoliv zoo na světě. NaviZoo poskytuje návštěvníkům jednoduchou interaktivní mapu, skrze kterou lze prezentovat jakýkoliv typ informací, fotogalerii s různými rámečky, malé hry, krátký kvíz a možnost vyhrát vstupenku do zoo. Po otestování této aplikace se bohužel objevila spousta zásadních problémů. Aplikace je velmi nestabilní (iOS11), seká se a drtivá většina funkcí je zamknuta za takzvaným premium členstvím, které uživatele neustále vyzývá ke koupi.



Obrázek 2.7: Mobilní aplikace napříč několika zoo – NaviZoo[8].

[7]Safari Park Dvůr Králové aplikace https://play.google.com/store/apps/details?id=cz.as4u.dk.zoo
[8]NaviZoo aplikace https://play.google.com/store/apps/details?id=com.navizoo.navizooapp

Kapitola 3

Vývoj použitelných informačních systémů

Pro vytvoření vhodného informačního systému bylo nutné prostudovat fáze jeho vývoje. Výsledný systém je určen široké veřejnosti, je tedy nutné dbát na míru použitelnosti, kvalitní analýzu požadavků uživatelů, dobrý návrh, správný výběr technologií a testování. Sekce 3.1 definuje použitelnost informačního systému, zatímco 3.2 je zaměřena na uvedení metod pro analýzu potřeb a existujícího stavu. Sekce 3.3 popisuje fáze návrhu systému a sekce 3.4 se věnuje technologiím pro implementaci serverové a klientské části systému, spolu s možnostmi tvorby interaktivních map. Poslední sekce 3.5 této kapitoly je krátce věnována metodám testování.

3.1 Definice a použitelnost informačního systému

Definici informačního systému lze vyjádřit několika způsoby. Například Vladimír Šmíd na svých webových stránkách [22] v rámci výuky uvádí, že informační systém lze chápat jako strukturovaný souhrn informací, které sbíráme, zpracováváme a využíváme k práci nebo k rozhodování. Použitelnost informačního systému lze vyjádřit skrze definici samotného slova použitelnost. Toto slovo je definováno Mezinárodní organizací pro normalizaci (ISO) jako míra, do které může být produkt používán konkrétními uživateli, aby efektivně, účinně a uspokojivě dosáhli stanovených cílů v daném kontextu užití. Jakob Nielsen [11] tento pojem upřizpůsobuje kontextu software a popisuje ho pomocí množiny kvalitativních atributů:

- **Naučitelnost** – jak snadné je pro uživatele zvládnutí základních úkolů při prvním setkání s programem. Taktéž značí, jak jednoduché je se s programem naučit pracovat efektivně.
- **Efektivita** – jak rychle je uživatel seznámený s programem schopen plnit zadané úkoly.
- **Zapamatovatelnost** – jak jednoduché je efektivně pracovat s programem po dlouhém časovém období bez používání programu.
- **Chybovost** – kolik chyb uživatelé při používání programu dělají, jak jsou chyby závažné a jak složité je řešení chyb.
- **Spokojenost** – jak příjemné je s programem pracovat.

3.2 Analýza požadavků a aktuálního stavu

Důkladná analýza aktuálního stavu a potřeb potenciálních uživatelů je nezbytnou součástí při vývoji. Cílem analýzy jsou čtyři hlavní výstupy. Vymezení funkčnosti, odhad množství práce, vyjasnění zadání a zachycení omezení. Existuje mnoho metod pro analýzu požadavků [18], lze je ale seskupit podle dvou hlavních typů – kvalitativní a kvantitativní. Detailním popisem analýzy požadavku se věnuje Jiří Reichel ve své knize [15]. V tabulce 3.1 lze vidět shrnutí nejdůležitějších vlastností obou typů.

- **Kvalitativní** – umožňuje detailně pochopit chování, přístup a schopnosti uživatelů, poznat kontext ve kterém se systém bude vyvíjet a efektivně identifikovat existující řešení.
- **Kvantitativní** – jedná se o poměrně rychlý a přímý způsob, poskytuje možnost testovat a validovat existující hypotézy, a zobecňovat výsledky na populaci.

	Kvalitativní	**Kvantitativní**
Rozsah	velké množství informací o malém počtu lidí	omezený rozsah informací o velkém počtu lidí
Redukce	silná redukce sledovaných jedinců	silná redukce sledovaných proměnných a vztahů
Generalizace	nelze zobecnit na celou populaci, ale umožňuje zjistit příčiny situace	lze zobecnit na celou populaci, ale neodhalí příčiny situace
Výstup	vytváří nové hypotézy a teorie, snaha porozumět situaci	ověřuje existující hypotézy

Tabulka 3.1: Shrnutí hlavních vlastností kvalitativních a kvantitativních přístupů.

Pro kvalitnější analýzu je vhodné aplikovat více metod z obou typů. Tato práce využívá konkrétně tři s ohledem na jejich pořadí:

1. **Výzkum od stolu** – relativně jednoduchý koncept pro rychlou počáteční analýzu. Nejedná se o metodu vlastního výzkumu, nýbrž o sběr již existujících dat a informací. Je vhodný pro seznámení s aktuálním stavem a konkurencí (například za pomoci internetu). Také je využíván k prozkoumávání potenciálních problémů a řešení skrze existující články, studie nebo diskuze [6].
2. **Dotazníky** – kvantitativní metoda pro levný sběr relevantních dat od velkého množství lidí. V knize [13] jsou dotazníky považovány za velmi kvalitní metodu. Lze je využít skrze fyzickou i digitální formu. Jsou vhodné v případě, kdy existují určité hypotézy, které je potřeba ověřit. Anonymita respondentů se dá považovat za výhodu z pohledu dotazovaných, ale z pohledu dotazujícího se jedná spíše o nevýhodu, neboť nelze přesně detekovat klamné respondenty.
3. **Strukturované a hloubkové rozhovory** – pro bližší porozumění konkrétních potřeb a vcítění do konkrétní situace potenciálních uživatelů [18]. Spadá do skupiny kvalitativních metod. Princip spočívá v připravení struktury otázek, na které budou dotazovaní při osobním rozhovoru odpovídat. Pro maximalizaci kvality výstupu této metody je vhodné pokládat otevřené otázky, co nejvíce naslouchat a zapisovat si poznámky během rozhovoru.

3.3 Návrh informačního systému

Po prvotním sběru informací a analýze požadavků následuje navrhování systému. Vzhledem k tomu, že během navrhování se mohou uživatelské požadavky měnit, je časté a někdy nevyhnutelné tyto činnosti (analýza a navrhování) prolínat. Fázi navrhování systému lze v tomto případě rozdělit na návrh databáze, komunikaci mezi serverem a klientem, a navrhování uživatelského rozhraní.

Databáze

Databáze uchovává data, se kterými se v systému pracuje a odpovídá na požadavky. Návrh databáze lze reprezentovat skrze diagramy, přičemž typickým příkladem je entitně relační diagram [5], který má za úkol přehledné vyobrazení struktury dabatáze, jednotlivých entit a vzájemných relací.

Komunikace klient-server

Serverová část by měla poskytovat rozhraní, které zpracovává požadavky od klientských aplikací nebo úzce svázaného administračního prostředí. Tyto požadavky by měla vyhodnocovat a žadateli posílat odpověď na základě obsahu daného požadavku. Serverová komunikace s klientem může být založena na dvou principech[1]:

- **Stavový (Stateful)** – jak již název napovídá, jedná se o způsob komunikace, kde si server uchovává aktuální stav uživatele. Po přihlášení do systému si server ukládá a posílá klientovi takzvané *Session ID*, které ho identifikuje při budoucí komunikaci.
- **Bezstavový (Stateless)** – pro ověření klienta používá speciální token nebo cookies[2]. Jednotlivé požadavky jsou na sobě plně nezávislé a server si neuchovává aktuální stav klienta. Výhodou cookies je, že se posílají automaticky v požadavku, zatímco token musí být přiřazen k požadavku manuálně. Naopak výhodou tokenu oproti cookies je například nezávislost na doméně.

Pro úplnost je vhodné zmínit pojem REST[3]. Jedná se o architekturu využívající stateless principu, pomocí které lze přehledně a hlavně jednotně komunikovat se serverem. Zdroje tohoto serveru mají svůj vlastní webový identifikátor (URI) a přistupuje se k nim čtyřmi hlavními HTTP metodami – GET, POST, PUT a DELETE.

Uživatelské rozhraní

Po návrhu serverové části systému je nyní možné přistoupit k navrhování uživatelského rozhraní. Grafickému návrhu by mělo předcházet definování množiny úkonů, které může daný uživatel v rozhraní realizovat. Tyto úkony lze velice přehledně znázornit například skrze diagram případů užití [5], který pracuje pouze s úkony uživatele v rozhraní, zatímco konkrétní logika a způsoby realizace zůstavají ukryty. Konkrétní diagram případů užití je v této práci sestaven pro webovou aplikaci správy zoo (sekce 5.3).

Po definování množiny úkonů lze přistoupit k navrhování grafického rozhraní, neboli takzvanému prototypování. Prototypování lze do určité míry považovat za způsob analýzy

[1] Hussein Nasser o stateful a stateless principech https://www.youtube.com/watch?v=nFPzI_Qg3FU
[2] Více o cookies na http://www.whatarecookies.com/
[3] Více o REST https://www.codecademy.com/articles/what-is-rest

požadavků, neboť při každé fázi návrhu lze získávat zpětnou vazbu. Zahrnuje čtyři hlavní fáze [2], přičemž poslední fáze se nejvíce podobá finálnímu rozhraní, se kterým bude konečný uživatel pracovat. Každá z uvedených fází nabývá určité důvěryhodnosti (*fidelity*), která stoupá s časovou náročností konkrétní fáze, úrovní podobnosti s výsledným produktem a nabízející interaktivitou (obrázky 3.1 a 3.2). Mezi nástroje používaných v jednotlivých fázích patří:

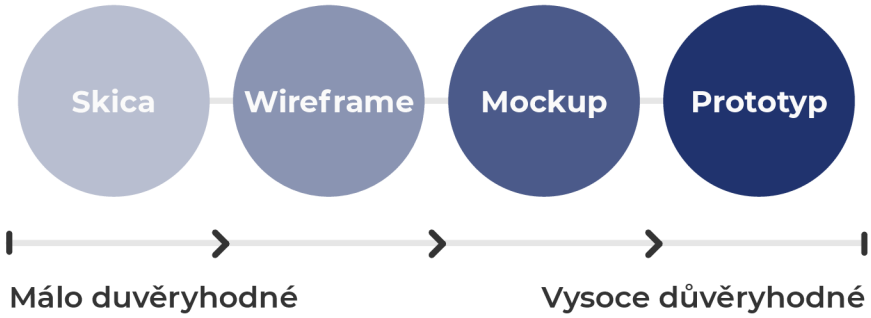
- **Skica** – jedná se o nejjednoduší fázi rychlého navrhování UI. Je velmi levná, protože k její realizaci stačí obyčejný papír a tužka. Nabývá tedy podoby fyzického náčrtu, skrze který lze sdělit základní vizuální ideu potenciálního UI. Tato fáze spadá do skupiny málo důvěryhodných, protože uživatelům nemůže s dostatečnou jistotou reprezentovat rozhraní, se kterým budou pracovat.

- **Drátěný model** – definuje rozmístění jednotlivých prvků na konkrétní stránce. Nejedná se o propracovaný grafický návrh, je totiž tvořen pouze za pomoci jednoduchých čar a obyčejného nestylovaného textu. Může mít digitální či fyzickou formu. Podobně jako u náčrtu se jedná o málo důvěryhodnou fázi, neboť stále nedisponuje dostačujícími charakteristikami důvěryhodného návrhu. I přesto, že drátěný model lze vytvořit i s nástroji zmiňovanými v dalších fázích, existuje spousta dedikovaných nástrojů čistě pro tvorbu drátěného modelu. Patří mezi ně například Frame Box[4] nebo Wireframe.cc[5].

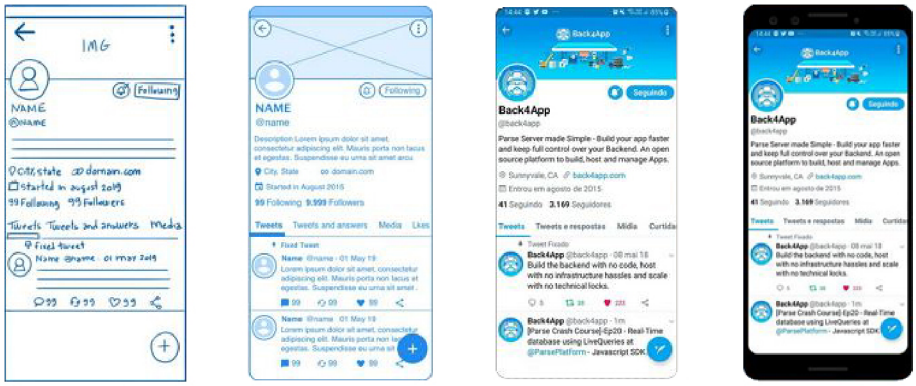
- **Mockup** – propracovaný design, jež reprezentuje téměř finální podobu jednotlivých prvků rozhraní, bez možnosti uživatelské interakce. Narozdíl od drátěného modelu již obsahuje barvy, fonty, ikonky apod. Cílem mockupu je navození emocí a prezentace designu v různých situacích a stavech pro jednodušší následný vývoj. Mockup lze tvořit v grafikých programech jako Adobe Photoshop[6], ale často se tvorba mockupů pojí s tvorbou prototypů, a proto se převážně využívají nástroje uvedené v další fázi. Díky propracovanosti a zaměření na detaily se tento typ považuje za vysoce důvěryhodný.

- **Prototyp** – svými vlastnostmi je nejpřesnější reprezentací návrhu finálního systému. Nabývá designu převážně z mockupu, ale navíc také disponuje funkčními přechody mezi jednotlivými prvky návrhu (uživatel si může proklikat jednotlivé stránky). Mezi nástroje pro tvorbu prototypů patří například Adobe XD[7] nebo Figma[8]. Díky možnosti interakce s prvky návrhu je definitivně zařazen mezi vysoce důvěryhodné.

[4]Frame Box http://framebox.org/
[5]Wireframe.cc https://wireframe.cc/
[6]Adobe Photoshop https://www.photoshop.com
[7]Adobe XD https://www.adobe.com/products/xd.html
[8]Figma https://www.figma.com/



Obrázek 3.1: Výše důvěryhodnosti v závislosti na fázích návrhu.



Obrázek 3.2: Jednotlivé fáze návrhu uživatelského rozhraní – náčrt, drátěný model, mockup a prototyp[9].

3.4 Implementace

Serverová část

Podle potřeb modelu skladovaných dat a zamýšleného účelu těchto dat existují různé typy databází, přičemž je lze rozdělit na dvě hlavní skupiny [7]:

- **Relační** – tradiční způsob tvorby databáze, data v ní jsou reprezentována skrz sloupce a řádky uvnitř tabulek. Mezi výhody patří kvalita robustních technologií, spolu s tím, že spousta z nich je spravována velkými známými společnostmi (například MySQL[10], Oracle[11] nebo PostgreSQL[12]). Tyto technologie velice často splňují ACID[13] požadavky, jsou většinou vnímány jako stabilní a kvalitně zdokumentované možnosti.
- **Nerelační** – také často označované jako NoSQL[14] databáze, začínají nabývat popularity spolu s komplexitou webových aplikací. Uchovávané data nemají stritkně předem

[9] Jednotlivé části převzaty s obrázku na stránce https://www.mockplus.com/blog/post/website-mockup
[10] MySQL https://www.mysql.com
[11] Oracle https://www.oracle.com
[12] PostgreSQL https://www.postgresql.org
[13] ACID https://retool.com/blog/whats-an-acid-compliant-database/
[14] NoSQL https://azure.microsoft.com/cs-cz/overview/nosql-database

předepsanou formu, tudíž jsou více flexibilní. Tento typ databází se vyplatí v případě, že databáze nemá nutně předepsanou podobu a může se za chodu měnit. Dále se rozdělují podle realizace struktury uchovávaných dat na klíč-hodnota (Redis[15], Riak[16]), dokumentové (MongoDB[17]), sloupcové (Apache Hadoop[18]) a nebo grafové (Neo4j[19]).

K implementaci serverové části a zajištění komunikace s databází existuje velké množství technologií a programovacích jazyků. Patří mezi ně například jazyk PHP[20], Python[21], C#[22] nebo Ruby[23]. Na základě těchto jazyků jsou postavené webové frameworky [10]. Jsou to velmi užitečné nástroje pro usnadnění a zrychlení vývoje, neboť poskytují předpřipravené konstrukce pro řešení častých problémů, například:

- práce s HTTP požadavky, odpověďmi a jejich zpracovávání,
- kontrolovaný přístup k datům,
- abstrakce přístupu databáze a práce s ní.

Proces rozhodování, jaký jazyk a framework použít, by měl podléhat zhodnocením žádoucích kritérií jako výkon, důvěryhodnost, popularita mezi komunitou, rozšiřitelnost nebo například bezpečnost. Mezi nejčastější frameworky patří kupříkladu Django[24] (Python), Express[25] (Javascript), Laravel[26] (PHP) nebo Ruby on Rails[27] (Ruby).

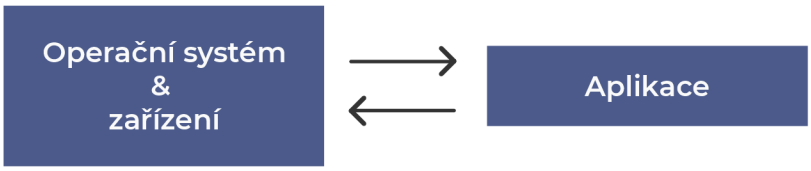
Klientská část

Klientskou část systému si lze představit jako jakoukoliv aplikaci, která komunikuje se systémem, ať už externí aplikace, či úzce svázané administrační prostředí. Klientské části čerpají data ze systému a určitým způsobem je prezentují uživatelům. Obecně je lze označit jako aplikace bez ohledu na podobu fyzického zařízení (mobilní, stolní apod). Pro jejich tvorbu je nutné úvést jejich konkrétní rozdělení a krátké porovnání. Finální rozhodnutí o konkrétním přístupu je velmi důležité, neboť může výrazně ovlivnit průběh celého vývoje. Existují tedy 3 hlavní typy aplikací (jejich sumarizační porovnání je v tabulce 3.2):

- **Nativní aplikace** – jsou vyvíjeny pomocí technologií, kterým je optimalizován operační systém koncového zařízení (například Kotlin[28] – Android, C# – Windows, Swift[29] – iOS). Díky takovéto optimalizaci může být ve výsledku aplikace velice rychlá, stabilní a také povoluje vývojářům snadný přístup k hardwarovým prvkům zařízení (kamera, poloha, mikrofon apod). Velkou výhodou tohoto typu aplikací je také možnost jednoduchého přizpůsobení uživatelského rozhraní grafickým standardům dané

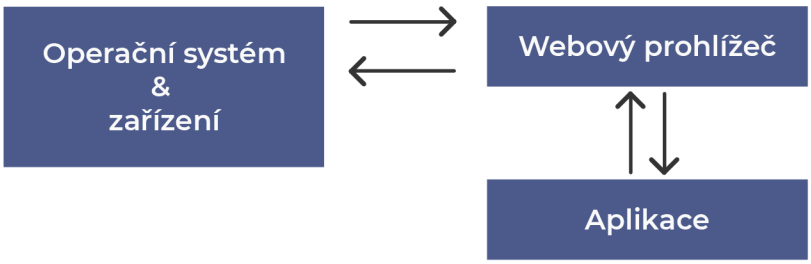
[15] Redis https://redis.io
[16] Riak https://riak.com
[17] MongoDB https://www.mongodb.com
[18] Apache Hadoop https://hadoop.apache.org
[19] Neo4j https://neo4j.com
[20] PHP https://www.php.net
[21] Python https://www.python.org
[22] C# https://docs.microsoft.com/cs-cz/dotnet/csharp
[23] Ruby https://www.ruby-lang.org
[24] Django https://www.djangoproject.com
[25] Express https://expressjs.com
[26] Laravel https://laravel.com
[27] Ruby on Rails https://rubyonrails.org
[28] Kotlin https://kotlinlang.org
[29] Swift https://swift.org

platformy. S výhodami tohoto způsobu se ovšem objevují i určité nevýhody, například neumožnění využití ekvivalentního zdrojového kódu napříč více platformami. Na obrázku 3.3 lze vidět velmi zjednodušenou reprezentaci nativní aplikace v zařízení.



Obrázek 3.3: Zjednodušený diagram nativní aplikace.

- **Webové aplikace** – jedná se o velice jednoduchý koncept. I přesto, že název obsahuje slovo aplikace, se spíše jedná o webovou stránku disponující některými charakteristikami aplikace. Pro jejich vývoj je potřeba prohlížeč a podporující webové technologie jako HTML, CSS a JavaScript (znázorněno na obrázku 3.4). Narozdíl od nativních aplikací nemají stanovené omezení vůči operačnímu systému a mohou tedy být spustitelné na jakémkoliv zařízení disponující webovým prohlížečem. Zároveň ale fakt, že tyto aplikace jsou spuštěné v prohlížeči, s sebou přináší několik nevýhod, mezi které patří hlavně nízká stabilita, závislost na prohlížeči (verze a podpora) a velice omezené možnosti pro práci s hardwarovými prvky zařízení.



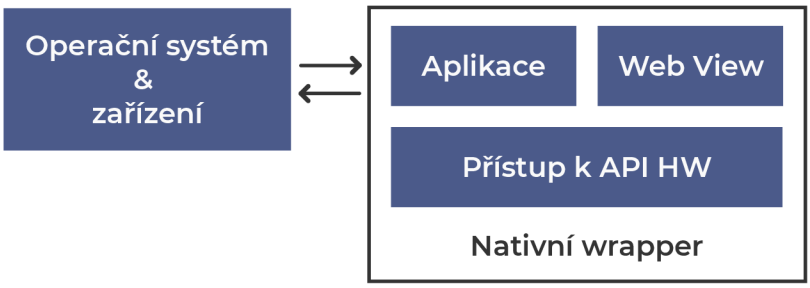
Obrázek 3.4: Zjednodušený diagram webové aplikace.

Pro úplnost tohoto typu je vhodné zmínit i takzvané *progresivní webové aplikace* (PWA[30]), které poskytují lepší podmínky pro používání webových aplikací, jako například lepší optimalizace, možnosti offline používání, či možnost přidání webové aplikace na plochu zařízení. Podrobnějším rozborem se zabývá Tal Ater ve své knize [1].

- **Hybridní aplikace** – kombinují kladné vlastnosti z webových a nativních aplikací. Pro vykreslování využívají jádro webového prohlížeče (*Web View* na obrázku 3.5), tudíž se nejedná o obyčejnou webovou stránku s URL. Pro vývoj se taktéž používají webové technologie. Výsledné aplikace se odlišují díky obalující *web-to-native* vrstvě (na obrázku 3.5 takzvaný *Native wrapper*). Tato vrstva poskytuje nádstavbu pro komunikaci mezi zmíňovaným zabudovaným prohlížečem a API[31] (takzvané *application programming interface*) komponentami zařízení, ke kterému by obyčejná webová aplikace neměla přístup (například úložiště). Podobně jako u webových aplikací se jedná o aplikace multiplatformní a díky poskytnuté provázanosti jazyka Javascript a obalu-

[30]PWA https://web.dev/progressive-web-apps/
[31]API https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

jícího frameworku (jako například Electron[32] nebo Cordova[33]) se jedná o velice pevný základ pro tvorbu multiplatformních aplikací.



Obrázek 3.5: Zjednodušený diagram hybridní aplikace.

	Nativní	**Webové**	**Hybridní**
Multiplatformní	ne	ano	ano
Distribuční platforma	ano	ne	ano
Výkon a optimalizace	úplně	málo	téměř úplně
Offline	ano	ne	ano
Nativní API	úplně	málo	téměř úplně

Tabulka 3.2: Sumarizace porovnání tří typů aplikací.

Interaktivní mapy v aplikacích

Tato podsekce je věnována možnostem pro interagraci interaktivních map do aplikací. V rámci obecné implementace se sice jedná o velmi specifický okruh, nicméně v této práci figuruje jako relativně podstatný subjekt, proto je vhodné prostudovat existující možnosti a jejich vlastnosti. Mezi nejznámější řešení poskytující práci s mapou patří [3]:

- **Google Maps**[34] – tato technologie se může chlubit 99% pokrytím světa s více než jednou miliardou aktivních uživatelů. Dlouhodobě se jedná o průmyslový standard pro integraci map do aplikací. Jejich API je velice kvalitně zdokumentované, funkce, které nabízí, jsou rozsáhlé a kvalitně propojené v jejich ekosystému. Nabízí taktéž knihovny pro integraci map na různých platformách (iOS, Android, web).
- **Mapbox**[35] – jeden z největších zprostředkovatelů nástrojů pro práci s mapami. Většina podkladů, které Mapbox používá, je veřejná. Mezi zdroje těchto podkladů patří i mimo jiné například OpenStreetMap. Nabízí komplexní sadu funkcí a nástrojů pro integraci mapových služeb. Mezi jejich nabídku spadají knihovny pro práci s mapou s využitím jazyka Javascript, Mapbox Studio[36] pro přizpůsobení grafické podoby mapy nebo například mapové SDK[37] pro nativní mobilní aplikace (Android, iOS).

[32] Electron https://www.electronjs.org/
[33] Cordova https://cordova.apache.org/
[34] Google Maps https://developers.google.com/maps/apis-by-platform
[35] Mapbox https://www.mapbox.com
[36] Mapbox Studio https://www.mapbox.com/mapbox-studio
[37] Mapbox SDK https://www.mapbox.com/mobile-maps-sdk

- **OpenStreetMap**[38] – *open-source* projekt, řízen svojí otevřenou komunitou, která dobrovolně přidává a spravuje mapová data. Velkou výhodou této možnosti je její absolutní volnost při používání (je zcela zdarma). Narozdíl od konkurentů neposkytuje ve svém základu žádnou API pro tvorbu map v aplikacích, nicméně existuje velmi velké množství knihoven, které tuto funkčnost zařídí. Nejčastějším zástupcem těchto knihoven je knihovna Leaflet[39], která poskytuje velmi kvalitní dokumentaci, bohatou komunitu a API pro tvorbu interaktivních map. Velkou výhodou této knihovny je, že s ní lze čerpat mapové podklady a nástroje z různých mapových API (například i Mapbox).

Rozhodování o technologii pro integraci interaktivních map by se mělo odvíjet od požadavků a cílené podoby. U zástupců jako Google Maps a Mapbox lze v případě zájmu využít jejich propracované a úzce svázané nástroje, které poskytnou velmi kvalitní a jednoduchý vývoj. Na druhou stranu knihovnoa Leaflet (v základu využívá OopenStreetMap) dokáže poskytnout bohatou *open-source* komunitu, velké množství dodatečných knihoven a kompletní volnost v případě budoucí výměny zprostředkovatele určitých částí mapy (například v případě výměny zdroje mapových podkladů).

3.5 Metody testování použitelnosti

Při vývoji jakéhokoliv software se využívá testování jako nástroj pro zhodnocení míry naplnění stanovených cílů. Množství metod, kterými lze toto testování realizovat, je opravdu velké, ale všechny mají společný prvek – uživatele a jejich zpětnou vazbu. Může mezi ně patřit například [18]:

- **A/B testování** – testování variant určité komponenty, například design či rozložení modulů. Výsledky zakládá na přímé zpětné vazbě od uživatelů podílejících se na testování.
- **Párové testování** – testování skrze existující seznam úkolů a otázek, které jsou zadány uživateli, zatímco druhá osoba tohoto uživate sleduje a zapisuje si poznámky. Na závěr tyto poznámky analyzuje a vyhodnocuje.
- **Eye tracking** – forma testování, která se zaměřuje na viditelnost prvků, rozložení textu apod. K tomuto testování je využívána kamera, která snímá pohyb očí uživatele a následně dle tohoto pohybu generuje takzvané teplotní mapy (heatmap).

[38] OpenStreetMap https://www.openstreetmap.org
[39] Leaflet https://leafletjs.com

Kapitola 4

Analýza potřeb zoo a návštěvníků

Je žádoucí, aby systém vycházel uživatelům vstříc a napomáhal jim v co nejvíce aspektech. Proto je nutné prozkoumat požadavky potenciálních uživatelů. Tyto požadavky jsou klíčovým prvkem pro systémový návrh, neboť bez nich se jedná pouze o osobní, zkreslený náhled, který nemusí být v souladu s názory konečných uživatelů. Záměrem analýzy v této práci není přemluvit všechny zoo ke spolupráci, ale spíše se blíže seznámit s aktuálním stavem, získat podvědomí několika zoo a podle této analýzy poskytnout případný dlouhodobý plán, skrze který by šla celá situace vylepšit. Počáteční spolupráce s menšími zoo může poskytnout více důležitých informací, a umožní systému nabýt většího rozsahu. V případě analýzy potřeb a požadavků návštěvníků je nutné provést průzkum detailněji než u jednotlivých zoo, protože bez uživatelů koncové mobilní aplikace nemá tento systém smysl a jednotlivé zoo nemají motivaci k jeho používání.

Konkrétní analýze se věnují sekce 4.1 a 4.2. Sekce 4.1 se zaměřuje na analýzu požadavků a potřeb uživatelů administračního systému (jednotlivých zoo), zatímco sekce 4.2 se věnuje požadavkům potenciálních uživatelů mobilní aplikace (návštěvníků zoo).

4.1 Požadavky zoo

Je velmi důležité pochopit aktuální situaci zoo, seznámit se s jejich názorem na aktuální stav a sesbírat co nejvíce relevantních dat, které usnadní navrhování systému. Na sběr těchto dat je využit dotazník (dostupný na přiloženém datovém nosiči v souboru `dotazniky/dotaznik_zoo.pdf`) a následně hloubkový a strukturovaný rozhovor.

Statistiky z dotazníku pro zoo

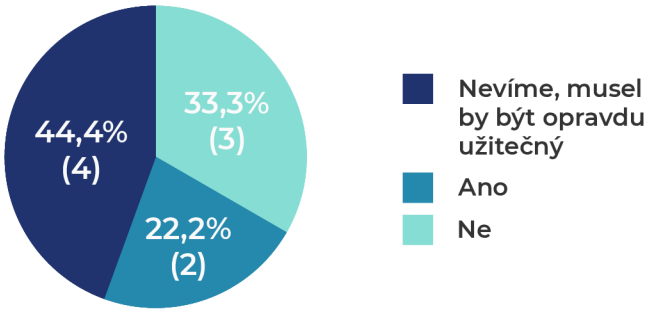
Dotazník byl rozeslán celkem 15 zoo. Vyplnilo ho k datu odevzdávání této práce 9 zoo, konkrétně Zoo Bojnice, Zoo Bratislava, Zoo Děčín, Zoo Jihlava, Zoo Hluboká, Zoo Olomouc, Zoo Liberec, Zoo Chleby a Zoo Spišská Nová Ves. Nereprezentuje tedy názor všech zoo a může se jednat do určité míry o zkreslená data. Bližší spolupráce během analýzy potřeb a požadavků byla vedena se Zoo Brno, která dotazník nevyplňovala, ale účastnila se hloubkového a strukturovaného rozhovoru.

Podle statistik první fáze dotazníku mají návštěvníci k čerpání žádoucích informací a orientaci uvnitř zoo nejvíce k dispozici informační tabule, letáky či webové stránky. Pokud se jedná o zvířata, tak všechny zoo disponují alespoň 200 druhy. Při tak vysokém počtu je během navrhování rozhraní vhodné tuto informaci brát v potaz a zaměřit se na jednoduchý způsob seskupování zvířat a jejich efektivní filtrování.

Na otázku, zda zoo nabízí svým návštěvníkům mobilní aplikaci, všechny dotazované zoo odpověděly ne. Je vhodné zdůraznit, že tento výsledek nereprezentuje všechny zoo (například v průzkumu existujících aplikací lze vidět několik zoo s mobilními aplikacemi). Jedná se spíše o situaci, kdy zoo, jež disponují mobilní aplikací, nemají zájem o účast v tomto dotazníku, nebo neměly zájem odpovídat na úvodní kontaktní email.

I přesto, že na dotazník odpovědělo pouze 9 zoo, je zde zjevný prostor pro vývoj zmiňovaného systému. Tento předpoklad je také podložen grafy 4.1 a 4.2, přičemž v grafu 4.1 lze vidět četnosti hlavních důvodů, proč zoo nemají podobný systém či mobilní aplikaci (bylo možné zvolit více možností). Graf 4.2 ukazuje, že 2 zoo by se již teď chtěly podílet na navhovaném systému, a 4 zoo si zatím nejsou jisté (ať už kvůli špatným předchozím zkušenostem či jiným důvodům).

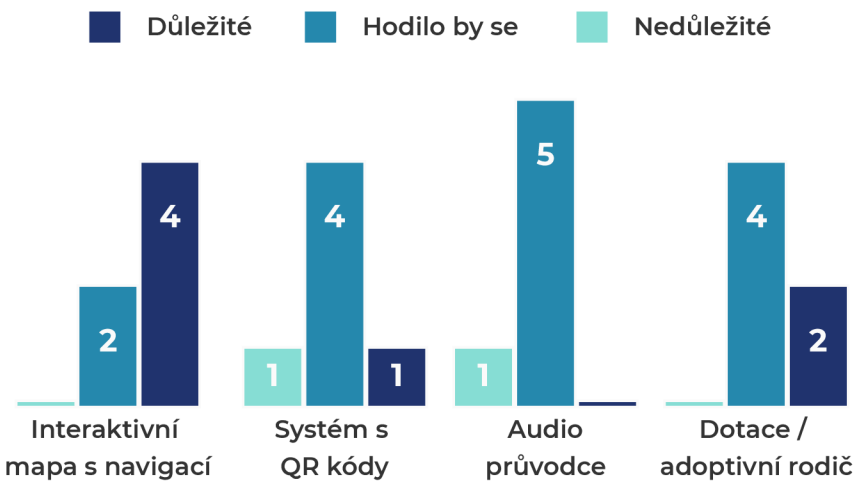
Obrázek 4.1: Četnosti důvodů, proč určitá zoo nemá k dispozici mobilní aplikaci či zmiňovaný systém.



Obrázek 4.2: Statistika prvotního zájmu o účast v navrhovaném systému.

Dále bylo nutné zjistit, zda zoo disponují fotkami zvířat a expozic. Všechny zoo, které se účastnili dotazníku, mají dostupné kvalitní fotky zvířat.

Důležitou částí je hodnocení potenciální funkčnosti výsledného systému. Byly uvedeny 4 hlavní funkcionality, přičemž je šlo ohodnotit jednou ze tří možností – nedůležité, hodilo by se, důležité. Výsledný graf 4.3 znázorňuje četnosti hodnocení jednotlivých funkcionalit.



Obrázek 4.3: Četnosti hodnocení důležitosti jednotlivých funkcionalit.

Ke každému sloupci je při výpočtu důležitosti přiřazena váhová hodnota:

- nedůležité = 0,
- hodilo by se = 1,
- důležité = 2.

Finální důležitost funkcionality je v práci počítána vynásobením četností jednotlivých sloupců s jejich přiřazenou hodnotou a následným součtem výsledných hodnot sloupců. Například u systému s QR kódy se jedná o výpočet $D = 0 * 1 + 1 * 4 + 2 * 1$. Z porovnání výsledků je nejdůležitější interaktivní mapa s navigací v zoo.

Dotazované zoo měly také možnost napsat, co si myslí, že je pro jejich návštěvníky v mobilní aplikaci důležité. Mezi časté patří hlavně seznam událostí a aktualit, nebo časy komentovaného krmení. Hypotézy o důležitosti funkctionalit se ověří dotazníkem pro návštěvníky, neboť zrovna oni budou mobilní aplikaci využívat, a aplikace musí odrážet hlavně jejich představy.

Hloubkový a strukturovaný rozhovor se Zoo Brno

Pro hlubší pochopení aktuálního stavu a požadavků byl domluven osobní rozhovor s vedením Zoo Brno. Z tohoto rozhovoru bylo možné ověřit určité hypotézy, ale také získat informace, které by obyčejný dotazník neodhalil.

Z pohledu brněnské zoo je nápad aplikace pro návštěvníky a sjednocujícího systému pro jakoukoliv zoo vnímán pozitivně. Ihned bylo zřejmé, že většina zoo v České republice nemá dostupnou finanční částku a čas k najmutí externí firmy na vytvoření aplikace pro návštěvníky, natož komplexního informačního systému. Celkové vylepšení či výměna dosavadního systému a mobilní aplikace jsou věci, které tato zoo chtěla realizovat již dlouho, ale doposud se nenaskytly vhodné možnosti. Při rozhovoru byly zmíněny jednotlivé funkcionality (podobně jako u dotazníku).

Očima Zoo Brno bylo pořadí důležitostí stejné jako u statistik z dotazníku, přičemž největší důraz byl kladen na zmiňovanou interaktivní mapu s interní navigací. Tuto funkcionalitu Zoo Brno považuje za nejdůležitější. Pro funkční implementaci takovéto mapy jsou potřebná mapová data. Zoo Brno je na veřejných zdrojích do určité míry zmapované (například v OpenStreetMap), ale úzké cestičky či nová zákoutí zde chybí.

4.2 Požadavky návštěvníků

Hlavní hypotéza, kterou vynesl předchozí dotazník (v sekci 4.1), je o důležitosti jednotlivých funkctionlit v mobilní aplikaci. Nyní je vhodné tyto hypotézy korespondovat s požadavky návštěvníků, ale hlavně tuto situaci využít k zjištění jejich názoru na různé aspekty při návštěvě zoo.

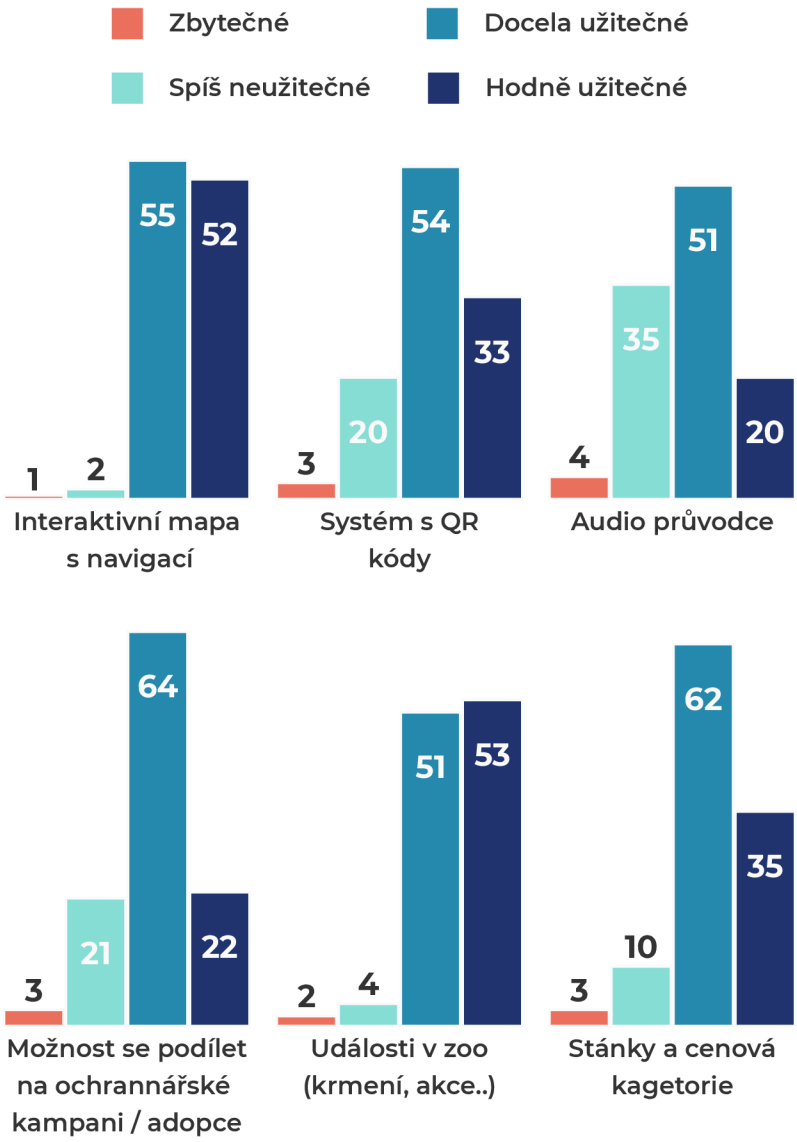
Výsledky dotazníku pro návštěvníky

Na anonymní dotazník pro návštěvníky (dostupný taktéž na přiloženém datovém nosiči v souboru `dotazniky/dotaznik_navstevnici.pdf`) zoo odpovědělo 110 lidí. Tato podsekce pojednává pouze o jeho nejdůležitějších částech. Ze 110 dotazovaných se 80,9% přiznalo, že používají mobilní telefon při návštěvě zoo. Celkem 77.3% také souhlasí, že navigování uvnitř zoo by mohlo být značně přehlednější. Před výsledky tohoto dotazníku se do určité míry předpokládalo, že většina návštěvníků si aspoň jednou zkusila nainstalovat aplikaci některé zoo do svého telefonu. Tento předpoklad byl ovšem velice milný, neboť 99% dotazovaných odpovědělo, že si nikdy žádnou aplikaci do zoo ani nezkusili nainstalovat. Naskytá se zde možnost, že drtivá většina dotazovaných nemá vůbec o mobilní aplikace v zoo zájem, tudíž je nutné zjistit, jaký byl důvod k nenainstalování. Další otázka tedy zněla „Jaký je důvod, že jsi si aplikaci nikdy nezkusil/a nainstalovat?“ (bylo povoleno vybrat více možností). Četnost odpovědí je následující:

- **31**× – Nechce se mi instalovat aplikace pro každou zoo zvlášť.
- **74**× – Nevěděl/a jsem, jestli/že ta konkrétní zoo má mobilní aplikaci.
- **5**× – Nemám místo v telefonu.
- **39**× – Přijde mi to zbytečné.

Podle uváděných četností se opět odhaluje prostor pro vývoj a zlepšení aktuální situace. Většina dotazovaných neví, že nějaké aplikace v zoo existují. V budoucnu je tedy žádoucí se zaměřit na vhodnou reklamu či marketing cílovým skupinám a nezanedbat důležité formy propagace. Návštěvníkům, kteří si nechtějí instalovat do každé zoo aplikaci zvlášť, plánovaný systém nahrává přímo do karet, neboť se od základu jedná o sjednocující systém s jednou společnou mobilní aplikací.

Otázka o důležitosti jednotlivých funkcionalit (stejně jako u dotazníku pro zoo) byla položena také návštěvníkům. V grafech na obrázku 4.4 lze vidět jejich hodnocení.



Obrázek 4.4: Četnosti hodnocení důležitosti jednotlivých funkcionalit podle návštěvníků.

Pro zjištění důležitosti je použit stejný výpočet jako pro 4.3, s tím, že váhové hodnoty jsou nyní: zbytečné = -1, spíš neužitečné = 0, docela užitečné = 1 a hodně užitečné = 2. Z porovnání výsledků je pro návštěvníky nejužitečnější seznam událostí a plánované akce, poté následuje interaktivní mapa s navigací, na třetím místě jsou lokace stánků a jejich cenové kategorie. Následuje systém s QR kódy, hned za ním je možnost zapojit se do ochranářské kampaně. Ukázalo se, že ze všech nabízených je pro návštěvníky audio průvodce nejméně důležitou funkcionalitou. Zároveň ale to, že se některá funkcionalita umístila mezi posledními, neznamená, že by v systému neměla existovat. Výsledná statistika spíše napovídá, na jakou část by měl být v návrhu a implementaci kladen největší důraz.

K návrhu mobilní aplikace bylo také důležité zjistit procentuální zastoupení operačních systémů v telefonech návštěvníků. Operační systém iOS využívá 40,6% návštěvníků, zatímco Android převládá se svými 58,2%. Není tedy vhodné se soustředit pouze na jeden z nich, ale spíše se zaměřit na multiplaformní řešení.

4.3 Shrnutí zjištěných informací

Podle výsledků dostazníků a rozhovoru se zoo lze konstatovat následující:

- Zoo v České republice často nemají k dispozici vhodné řešení pro správu informací.
- Taktéž nemají přebytečné prostředky (finanční, časové apod) na vývoj svého vlastního systému a mobilní aplikace.
- Lze určit hlavní skupiny informací, které bude možné v systému spravovat (více v následující kapitole, konkrétně sekce 5.1).
- Ze strany několika zoo již existuje určitý zájem o podílení na navrhovaném systému.
- Některé zoo nejsou úplně zmapované, například Zoo Brno chybí nové či krátké uličky. Pokud má aplikace disponovat touto mapou s navigací, je nezbytné tyto data ve vybraných zoo aktualizovat.
- Značná část návštěvníků neví, že v některých zoo nějaká aplikace existuje. V některých případech se o ně ani nezjímají, neboť jim vadí nutnost ji hledat a instalovat v každé zoo zvlášť.
- Potenciální skupina uživatelů aplikace není nijak přísně vyhrazena, naopak se může jednat o jakéhokoliv návštěvníka zoo.
- Návštěvníci nejvíce vyžadují v aplikaci aktuální události a interaktivní mapu, neboť se díky těmto funkcionalitám zjednoduší hledání aktualit a zároveň zvýší přehlednost orientace uvnitř zoo.
- Je potřeba se zaměřit na multiplatformní řešení mobilní aplikace, neboť vynechání jednoho řešení, ať už pro Android či iOS, by vedlo ke ztrátě téměř poloviny potenciálních uživatelů.

Kapitola 5

Návrh řešení

Návrh je rozdělen do několika částí. První část 5.1 je věnována modelu dat, v němž jsou definovány hlavní pravidla, kterých se návrh bude držet. Druhá část 5.2 je zaměřena návrhu serverové části systému. Třetí sekce 5.3 se věnuje návrhu webové aplikace pro zoo (administrační aplikace). Poslední sekce 5.4 je zaměřena na mobilní aplikaci pro návštěvníky.

5.1 Základní model dat

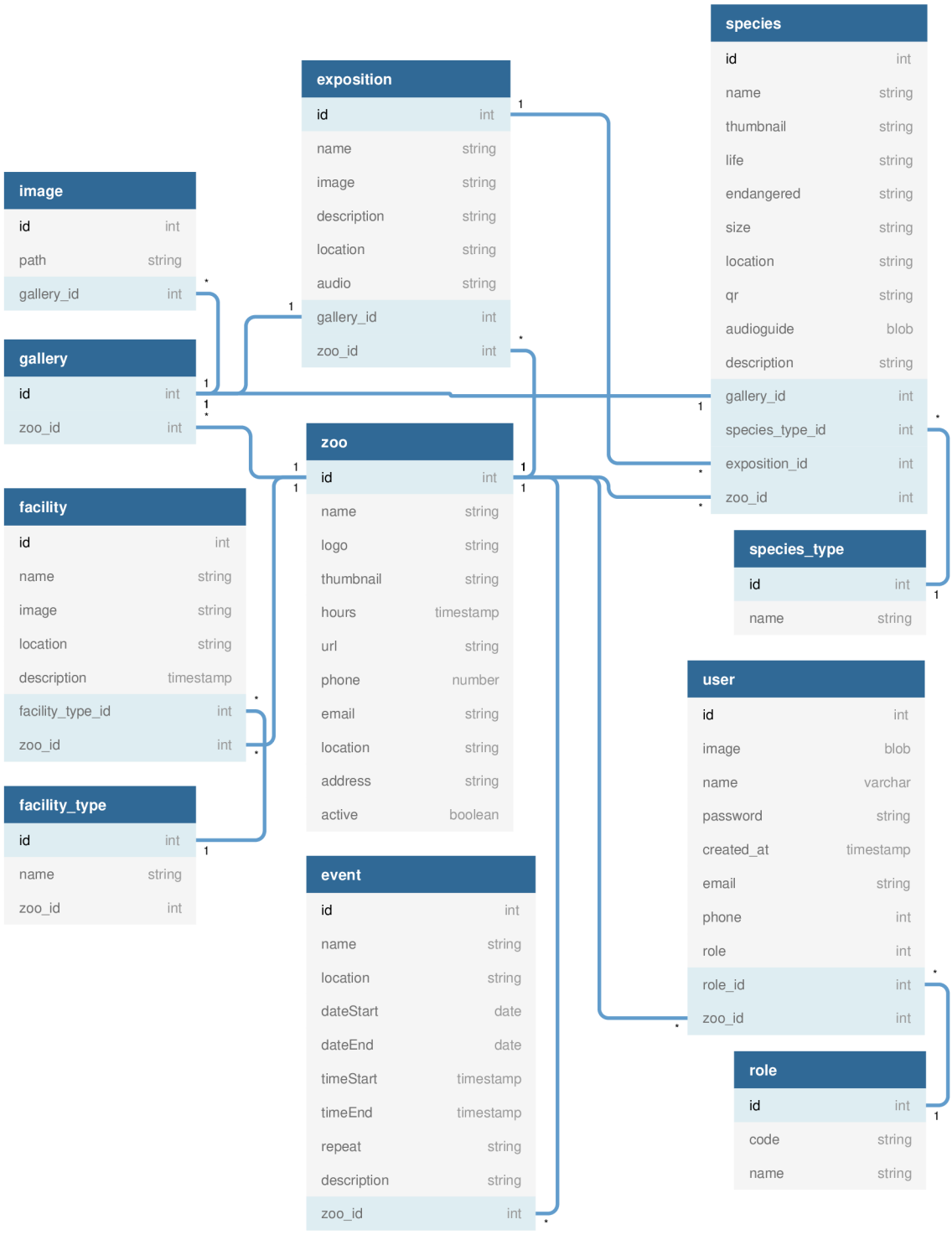
Jak již práce několikrát zmiňovala, navrhované řešení nabývá podoby sjednocujícího systému pro jakoukoliv zoo. Návštěvníci z něj budou získávat informace skrz dedikovanou mobilní aplikaci, která je bude atraktivním způsobem prezentovat.

Skupiny informací, se kterými se bude v systému moci zacházet jsou následující – informace o zoo, zvířata, expozice, události a zařízení (stánky, toalety apod). Je nutné, aby si jednotlivé zoo nemohly editovat záznamy navzájem. Proto je nutné systém vytvořit tak, aby uživatelské účty zaměstnanců byly svázané pouze s jednou zoo, a nemohly si editovat informace navzájem. Je vhodné jednotlivé zoo spolu s uživateli od sebe co nejvíce izolovat. Každý uživatel má také takzvanou roli, která určuje úroveň jeho oprávnění. Toto oprávnění udává, co může daný uživatel ve své zoo spravovat. Uživatelské účty pro návštěvníky nejsou zatím v plánu, ale systém je navržen tak, aby se v případě potřeby dal o tyto uživatelské účty jednoduše rozšířit (například v případě gamifikace). Konkrétní informace spadající do zmiňovaných komponent jsou vyobrazené v entitně vztahovém diagramu 5.1.

5.2 Serverová část systému

Součástí serverového návrhu je specifikace konceptuálního modelu, který dokáže zjednodušit a vyjasnit určité spojitosti. K tomu je využit entitně vztahový diagram na obrázku 5.1. Tento diagram definuje všechny potřebné relace a atributy jednotlivých částí v databázi, z čehož lze následně vycházet při návrhu uživatelského rozhraní (díky konečné množině informací, se kterými se bude pracovat) a implementaci (při implementaci je možné, že se podoba databáze lehce změní). Vytvořený návrh je dostupný taktéž v nástroji dbdiagram.io[1].

[1] ER diagram v dbdiagram.io https://dbdiagram.io/d/5f8dcd8d3a78976d7b78449e



Obrázek 5.1: Entitně relační diagram navrhovaného systému.

Návržená komunikace klient-server

Celé rozhraní pro komunikaci mezi serverem a aplikacemi je navrženo jako plně bezstavové RESTful API. Toto rozhodnutí je provedeno na základě kladeného důrazu na budoucí rozšiřitelnost systému a jednotnou formu komunikace. Kompletní návrh API je příliš obsáhlý na zdejší výpis, proto je zde uveden spíš abstraktní popis. Navrženou API lze rozdělit na dvě části:

- **Nechráněné přístupové body** – nabízí skupinu bodů nevyžadující žádnou formu verifikace či ověření žadatele. V aktuálním návrhu API slouží všechny prvky této skupiny pouze ke čtení dat. Nevyžadují ani žádné tělo HTTP požadavku, nýbrž pouze správný tvar URI. Příklad takovéhoto přístupového bodu může být následující:

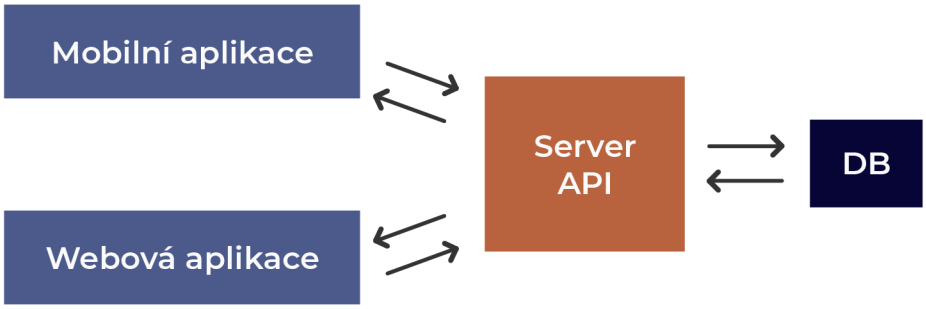
  ```
  GET api-v1/zoos/7/species/19
  ```

  kde číslo `7` je identifikátor konkrétní zoo a `19` je identifikátor konkrétního zvířecího druhu.

- **Chráněné přístupové body** – vybrané body vyžadující nejen správné URI, ale i ověření žadatele. Toto ověření je navrženo skrze speciální token v hlavičce HTTP požadavku. Server posílá token při přihlášení do systému a slouží k přímé identifikaci žadatele. Aplikace žadatele tedy musí zařídit uchování tokenu pro další požadavky spadající do této skupiny a jeho automatické přikládání do hlavičky HTTP požadavku.

V případě práce se soubory je potřeba navrhnout způsob jejich uchovávání. V tomto systému budou jména souborů generované tak, aby bylo co nejnáročnější je uhodnout. Adresáře, ve kterých se nachází, nebudou přístupné veřejnosti, tudíž nemůže dojít k jejich automatizovanému prohledávání. Velice podobný systém využívá například Facebook – každý obrázek má svoji vlastní veřejná URL složenou takovým způsobem, aby ji bylo prakticky nemožné uhádnout (a to i v případě, že fotka je součástí soukromé konverzace).

Navržená API nyní poskytuje dostatečnou úroveň rozšiřitelnosti, díky které lze z jakékoliv budoucí aplikace plně číst a spravovat veškeré informace v systému (záleží pouze na účelu a potřebách dané aplikace; ať už je určena pro správu či jako další způsob prezentace). V tento moment je tedy vhodné detailněji představit dvě separátní aplikace, které budou s touto API komunikovat. Jsou jimi webová aplikace pro správu zoo a mobilní aplikace pro návštěvníky. Na obrázku 5.2 níže lze vidět velmi zjednodušený diagram architektury.



Obrázek 5.2: Zjednodušený diagram architektury komunikací.

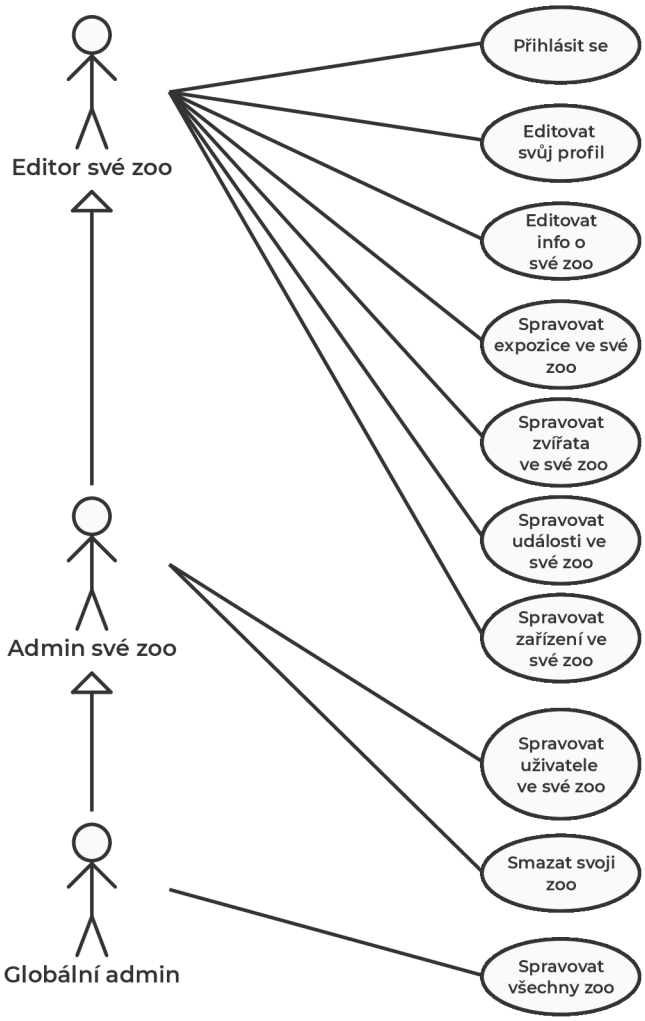
5.3 Webová aplikace pro zoo

Hlavní myšlenka této aplikace je správa informací pro jakoukoliv zoo. V drtivé většině případů aktuálně zoo spravují své informace skrz webové aplikace. Tato možnost nepředstavuje žádné překážky, naopak jí plně vychází vstříc, protože nabízí platformovou nezávislost a pro spuštění nevyžaduje žádný instalační proces. Také vzhledem k tomu, že mezi zaměstnanci zoo již existuje určitý zvyk, co se týká rozhraní správy, je vhodné se tomuto faktu přizpůsobit a tedy navrhnout tuto aplikaci jako webovou.

Pro přístup k této aplikaci je nutné přihlášení, neboť disponuje například možnostmi mazání uživatelů v zoo nebo změnu informací zvířecích druhů. Tyto funkce nesmí být přístupné veřejnosti, ale pouze autorizovaným uživatelům svázaných s konkrétní zoo.

Případy užití

Pro jednodušší navrhování uživatelského rozhraní je vhodné sestavit úkony, které lze v aplikaci realizovat. K tomu je využit diagram případů užití na obrázku 5.3. Slovo spravovat je v diagramu využito jako vyjádření kombinace tří úkonů – přidávat, upravovat a mazat.



Obrázek 5.3: Diagram případů užití webové aplikace.

Uživatelské rozhraní

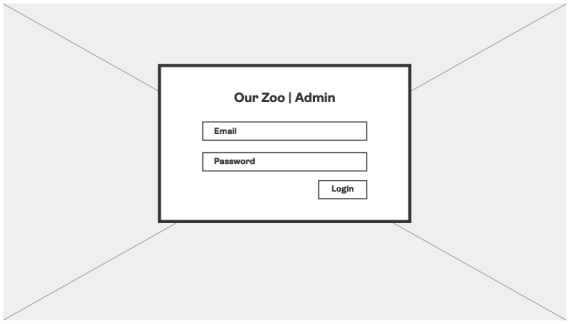
Při navrhování uživatelského rozhraní byl kladen velký důraz na jednoduchost, přehlednost a moderní design. Je nutné podotknout, že v případě této práce je vzhled rozhraní stejně důležitý jako jeho funkce. Toto tvrzení je založeno na předpokladu, že vzhled rozhraní je velkým faktorem pro jeho zapamatovatelnost. Zároveň dokáže přinést nový nádech do nepřehledného prostoru existujících řešení a následně tak vzbudit širší zájem.

Návrh se drží fází drátěného modelu a mockupu. Skica se neprováděla, neboť drátěný model dokáže poskytnout více důležtiých informací. Před návrhem rozhraní byl sestaven diagram případů užití, který poskytl dostatečný přehled o potřebných komponentách. Na celý návrh byl použit nástroj Adobe XD[2]. Grafický návrh této webové aplikace je k nalezení na přiloženém datovém nosiči (soubor `navrhy/design_web.xd`). I přesto, že v následujících fazích návrhu rozhraní je použit anglický jazyk, tak výsledné rozhraní bude implementováno s češtinou, neboť je aplikace k datu odevzdání práce mířena primárně na české zoo.

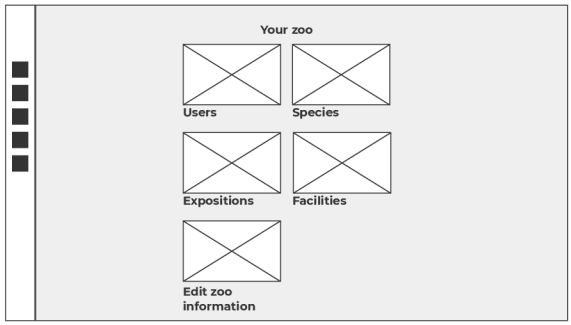
Drátěný model

Prvním krokem v aplikaci je přihlášení (obrázek 5.4). Po úspěšném přihlášení je uživatel přesměrován na domovskou stránku zoo, se kterou je svázán svým účtem (obrázek 5.5). Na této stránce má k dispozici hlavní komponenty aplikace. Tyto komponenty se drží diagramu případů užití, a každé je přiřazena vlastní stránka – uživatelé, zvířecí druhy, expozice, události, zařízení (např toalety a stánky) a informace o konkrétní zoo. Do těchto komponent se uživatel může dostat skrz menu nebo přes zmiňovanou domovskou stránku. Úkony, které uživatel může realizovat s jednotlivými komponenty jsou stejné – přidávat, upravovat nebo mazat (s ohledem na jeho oprávnění).

Konkrétní použití lze představit na situaci, kdy uživatel chce přidat nový zvířecí druh do své zoo. Přejde do komponenty zvířecích druhů (obrázek 5.6) a zvolí tlačíko „Přidat nový druh“, poté je přesměrován na stránku, kde vyplní náležící informace k jeho zvířecímu druhu (obrázek 5.7). V této fázi může uživatel buď tento druh přidat, nebo se může vrátit zpět do přehledu. Pokud se rozhodne, že chce zvířecí druh upravit nebo smazat, stačí aby vybral konkrétní druh, na další stránce změnil příslušné informace a potvrdil své změny. Toto schéma sdílí všechny komponenty v celé webové aplikaci. Rozhraní je díky tomu zapamatovatelné a snadno naučitelné.

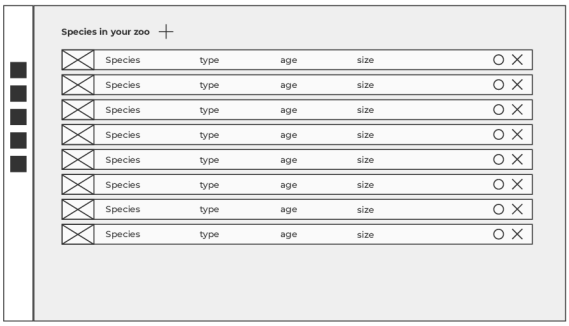


Obrázek 5.4: Drátěný model přihlašovací stránky.

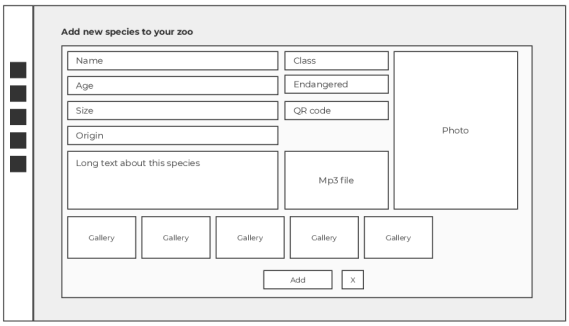


Obrázek 5.5: Drátěný model domovské stránky.

[2]Adobe XD https://www.adobe.com/cz/products/xd.html



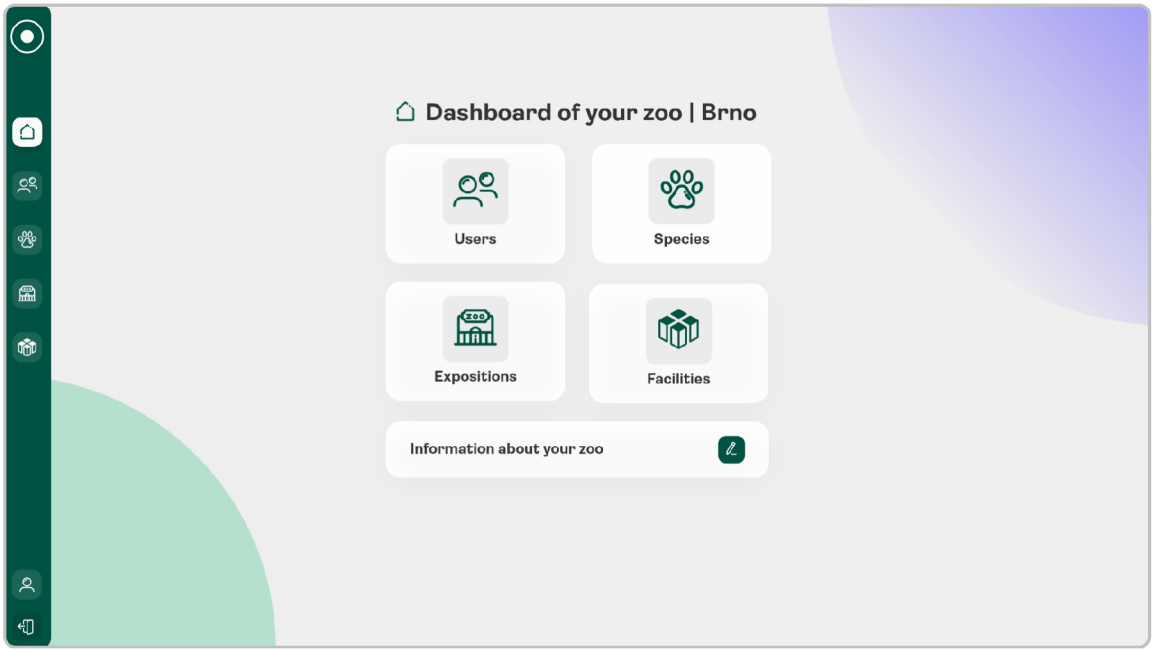
Obrázek 5.6: Drátěný model přehledu zvířecích druhů.



Obrázek 5.7: Drátěný model přidání zvířecího druhu.

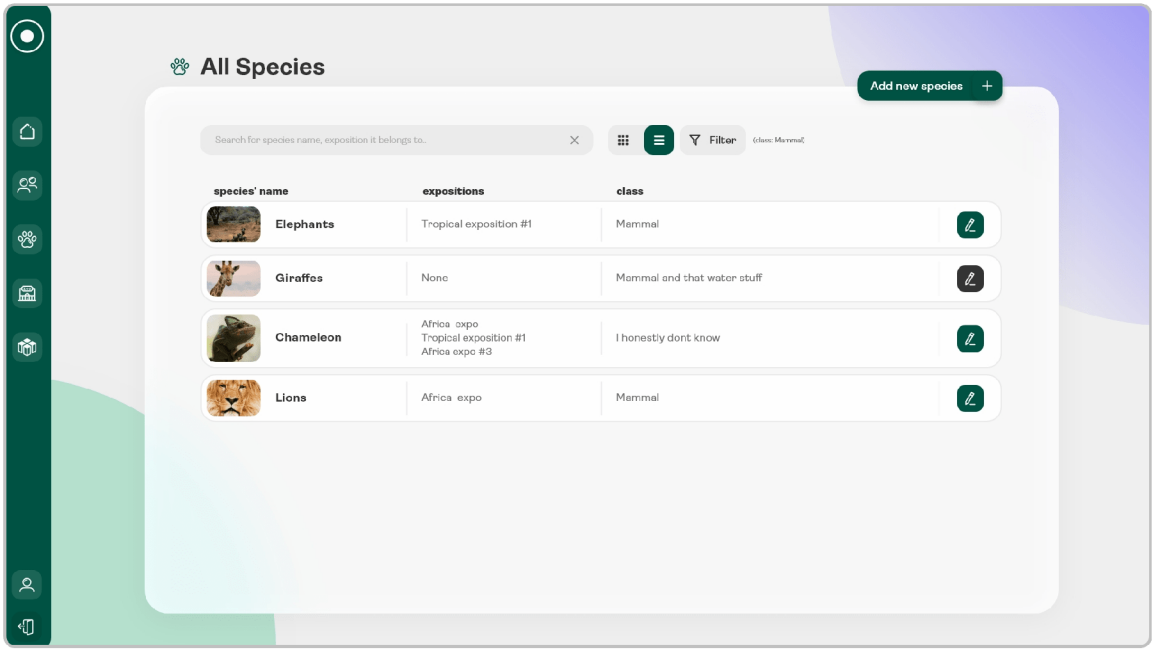
Mockup

Styl celého rozhraní je založen na kombinaci dvou aktuálních trendů – Flat design[3] a občasný Glassmorphism[4], nabývá velice jemného barevného schématu. Je vhodné, aby tyto barvy odpovídaly kontextu informací, kterými jsou obklopeny, proto jsou ve finálním návrhu jako primární barvy zvoleny odstíny zelené v kombinaci s velmi jemnou modrou. Na obrázcích níže lze vidět konkrétní mockup pro domovskou stránku (5.8), komponentu pro seznam zvířecích druhů (5.9) a editační komponentu zvolené expozice (5.10).

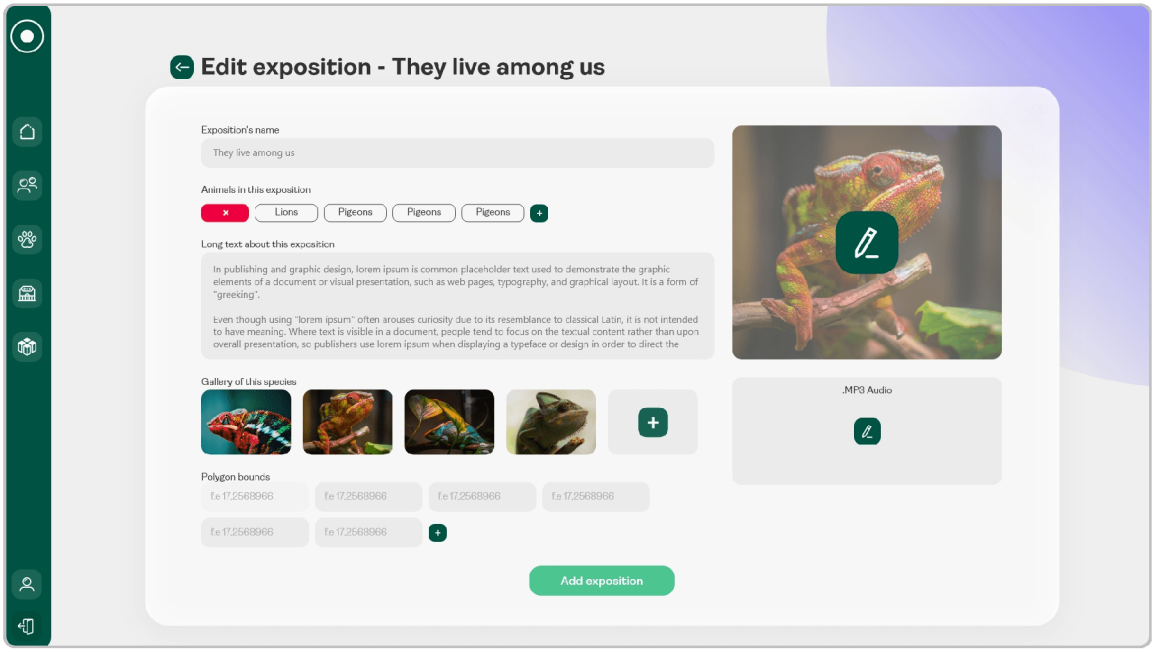


Obrázek 5.8: Domovská stránka administračního rozhraní.

[3]Flat design https://www.interaction-design.org/literature/topics/flat-design
[4]Glassmorphism https://uxdesign.cc/glassmorphism-in-user-interfaces-1f39bb1308c9



Obrázek 5.9: Stránka se zvířecími druhy.



Obrázek 5.10: Stránka pro úpravu vybrané expozice.

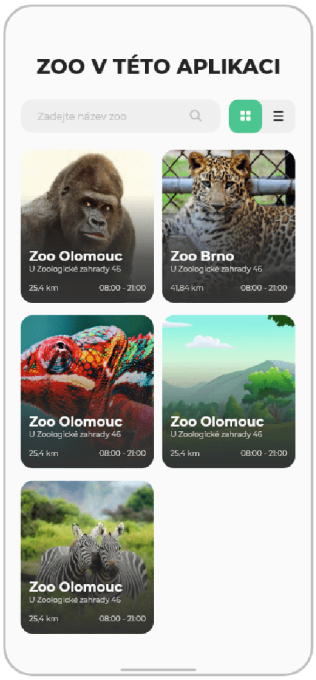
5.4 Mobilní aplikace pro návštěvníky

Tato aplikace je zamýšlena pro návštěvníky zoo. Bude existovat jako kapesní průvodce a zdroj zajímavých informací jakékoliv zoo ze zavedeného systému. Bude přizpůsobena co nejvíce platformám a zařízením, a proto je navržena jako hybridní, volně dostupná ke stažení pomocí nativních distribučních nástrojů. Aplikace nevyžaduje žádné přihlášení a nijak nepracuje s uživatelskými účty, a proto v aktuálním návrhu využívá pouze nechráněné koncové body API ze sekce 5.2 (tedy pouze čtení informací ze serveru).

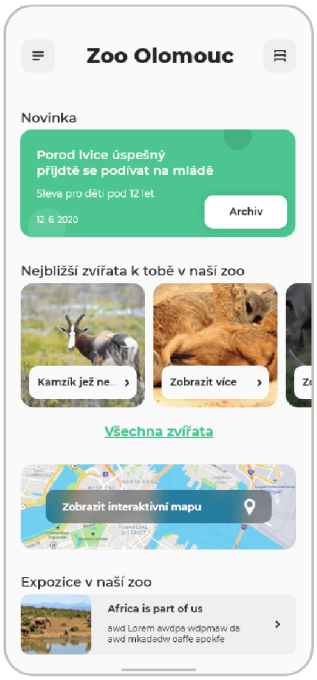
Uživatelské rozhraní a funkce aplikace

Podobně jako u webové aplikace, i zde se využívá jednoduché barevné schéma jemně zelené a modré barvy. Hned po funkcionalitě je moderní estetika a přehledné rozhraní klíčovým prvkem pro zaujmutí návštěvníků, proto byl na tuto část kládán velký důraz (celé grafické návrhy mobilní aplikace jsou taktéž k dispozici na přiloženém datovém nosiči v souboru `navrhy/design_mobil.xd`). Celá mobilní aplikace se drží stylu zaoblených rohů. Dříve na tento styl bylo nahlíženo pouze jako na dočasný trend, ale postupem času se čím dál více ukazuje jeho pozitivní vliv [16] na celý uživatelský zážitek. U každé komponenty je kladen velký důraz na snadné použití na mobilním zařízení. Název stránky či sekce, ve které se uživatel nachází, je vždy zobrazen v horní části stránky. Na většině stran jsou vedle tohoto nadpisu umístěny také ovládací prvky stránky jako například tlačítko menu, nebo tlačítko pro krok zpět.

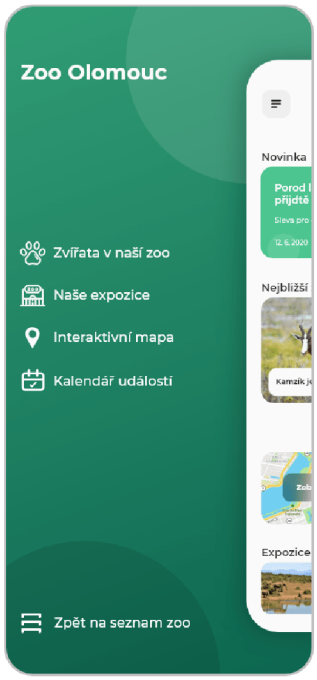
Narozdíl od webové aplikace, kde je využit diagram případů užití, drátěný model a mockup, je v případě této aplikace vhodnější kombinace mockupů uživatelského rozhraní s následným slovním popisem (jednotlivé popisy jsou doplňovány o ilustrace referované v závorkách):



Obrázek 5.11: Výběr konkrétní zoo.



Obrázek 5.12: Hlavní stránka zoo.



Obrázek 5.13: Menu vybrané zoo.

- **Výběr zoo** – při zapnutí aplikace je k dispozici seznam zoo, které jsou součástí navrhovaného systému (5.11). Jednotlivé zoo jsou v tomto seznamu v podobě velkých dlaždic, a díky tomu, že se řadí podle vzdálenosti od zařízení, je velmi zjednodušeno hledání žádoucí zoo. V každé dlaždici je úvodní obrázek zoo, její název, ale také lokace, otevírací doba a vzdálenost. Po výběru konkrétní zoo je uživatel přesměrován na domovskou stránku vybrané zoo. Všechny další interakce s aplikací jsou vázány pouze na tuto zoo.

- **Domovská stránka zoo** – zde je vyobrazen přehled všech důležitých komponent (5.12). Konkrétně se jedná o nejnovější událost, nejbližší zvířecí druhy, nejbližší expozice a odkaz na interaktivní mapu. Z domovské stránky je taktéž dostupné menu (5.13) a možnost vrátit se na seznam všech zoo. Všechny prvky tohoto přehledu fungují nejen jako odkaz na danou komponentu, ale hlavně jako způsob rychlého představení prioritních informací. Ve většině případů chtějí návštěvníci zjistit aktuální stav v zoo aniž by museli procházet celý archiv, proto se zde vždy zobrazuje jen jedna nejnovější aktualita. Seznam nejbližších zvířecích druhů je zobrazen přes horizontální posuvník, díky němuž lze vyobrazit více zvířat, aniž by zabíraly veškeré místo na obrazovce. Nejbližší expozice jsou umístěny hned za odkazem na interaktivní mapu. Jsou ve stylu širokých panelů, neboť je u nich nutno vypsat krátký popisek k jejich rychlému představení bez nutnosti je otevírat.

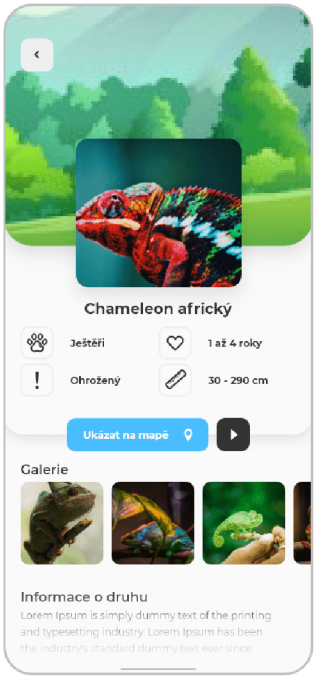
Obrázek 5.14: Seznam událostí v zoo.

Obrázek 5.15: Seznam zvířecí druhů v zoo.

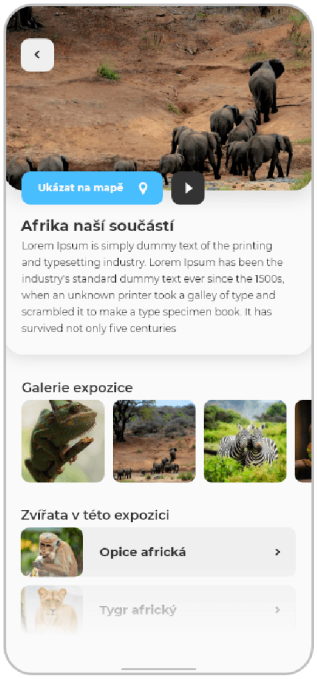
Obrázek 5.16: Seznam expozic v zoo.

- **Události** – slouží jako seznam nadcházejících událostí v konkrétní zoo a jejich detailní popis (5.14). Zároveň lze tuto komponentu využít jako archiv událostí. Tento seznam je spojen s vertikální časovou linií znázorňující datum a čas daných událostí, což výrazně zjenodušuje hledání podle žádoucího data.

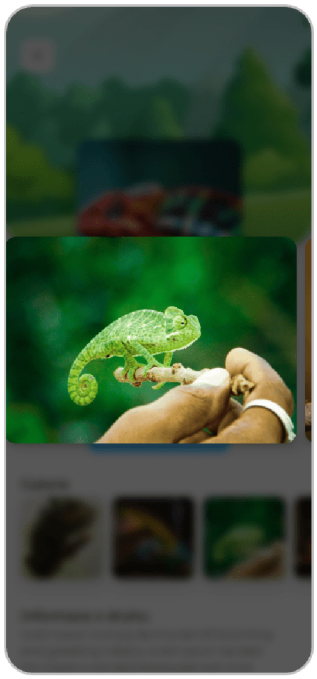
- **Zvířecí druhy** – seznam všech zvířecích druhů v dané zoo (5.15). Lze vybrat, jestli se mají řadit podle vzdálenosti nebo podle abecedy. Jelikož je v jedné zoo často více než 200 zvířecích druhů, je zde umístěno vyhledávací pole. Spolu s tímto vyhledávacím polem je zde přítomen také přepínač způsobu zobrazení tohoto seznamu (dlaždice nebo panely), přičemž ve výchozím nastavením jsou aktivované dlaždice (tři na jednu řádku). U každé dláždice je zobrazen pouze název druhu a jeho obrázek. Při výběru konkrétního zvířecího druhu se aplikace přesune na komponentu konkrétního zvířecího druhu (5.17).
- **Expozice** – seznam všech expozicí v konkrétní zoo (5.16). Stejně jako u zvířecích druhů se zde naskytují možnosti řazení dle vzdálenosti k zařízení a vyhledávání. Po zvolení konkrétní expozice se aplikace dostává do komponenty konkrétní expozice (5.18). Podobně jako u domovské stránky jsou jednotlivé expozice vyobrazené skrz panely (název expozice s krátkým popiskem).



Obrázek 5.17: Konkrétní zvířecí druh.



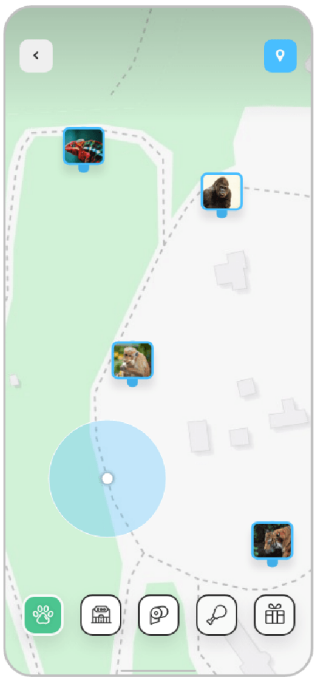
Obrázek 5.18: Konkrétní expozice.



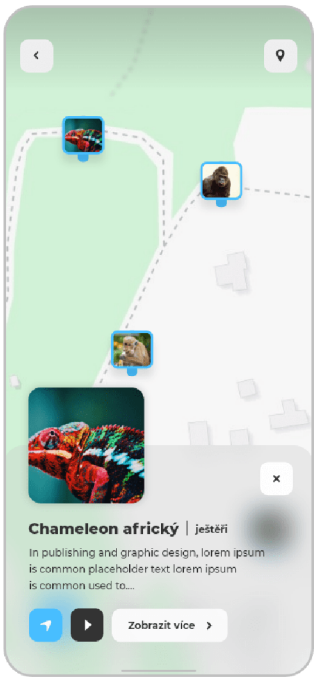
Obrázek 5.19: Příklad galerie.

- **Konkrétní zvířecí druh** – v této komponentě jsou všechny informace o vybraném druhu (5.17). Obsahuje základní informace jako ohroženost, velikost, délku života, a krátký popis, ale také je zde k dispozici krátká galerie věnována pouze vybranému druhu. Tato komponenta nabízí taktéž možnost lokalizace daného druhu na interaktivní mapě v rámci zoo a možnost spustit krátkou audio nahrávku o tomto druhu. Hlavní obrázek zvířete je spolu s jeho názvem umístěný na vrchní části stránky, neboť je důležité, aby uživatel ihned poznal kde se nachází a na co se dívá.
- **Konkrétní expozice** – podobně jako u konkrétního zvířecího druhu je zde popisek a krátká galerie (5.19). Taktéž ale obsahuje seznam zvířat, které k této expozici patří. Skrz tento seznam se lze přepnout na komponentu konkrétního zvířecího druhu.

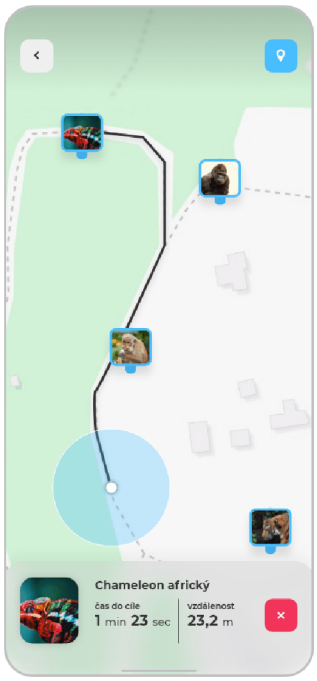
Expozici lze samozřejmě také lokalizovat na mapě nebo k ní spustit krátkou audio nahrávku (celá stránka je na obrázku 5.18).



Obrázek 5.20: Filtry v interaktivní mapě.



Obrázek 5.21: Vyjížděcí okno u interaktivní mapy.



Obrázek 5.22: Navigace v interaktivní mapě.

- **Interaktivní mapa** – při přepnutí do této komponenty je mapa zaměřena na lokaci vybrané zoo. Vyobrazuje odpovídající lokalitu zvířecích druhů, expozic a zařízení za pomoci ukazatele na mapě. Na tyto ukazatele je možnost kliknout a tím aktivovat vyjížděcí okno se stručnými informacemi a odkazem na celou komponentu (5.21). Taktéž je zde možnost spustit navigaci k vybranému ukazateli (5.22). Mezi ovládací prvky této stránky patří také aktivační tlačítko geolokace zařízení, díky kterému se na mapě zobrazí aktuální poloha mobilního zařízení. Při spuštění navigace k určitému ukazateli se toto tlačítko aktivuje automaticky. Je žádoucí, aby vyjížděcí okna zabíraly co nejméně místa na obrazovce, ale také musí poskytovat vhodné množství informací. Je předvídatelné, že množství ukazatelů se může stát příliš velkým a mapa tedy nepoužitelnou. Proto jsou navrženy filtry na každou zmíněnou skupinu, přičemž vždy musí být aktivován právě jeden filtr (5.20). Užívání této aplikace má sloužit mimo jiné jako motivace k návštěvě zoo. Z tohoto důvodu je interaktivní mapa dostupná pouze mobilním zařízením, které se fyzicky nachází blízko vybrané zoo. V opačném případě by totiž sloužila jako další argument proč zůstat doma.

Kapitola 6

Implementace navržených řešení

Tato kapitola se věnuje zvoleným technologiím a implementačním detailům. Každá sekce popisuje implementaci jednoho celku systému. První sekce 6.1 je zaměřena na technologie využité pro implementaci serverové části spolu s API, druhá sekce 6.2 je orientována na technologie a implementační poznatky administrační webové aplikace. Sekce 6.3 se věnuje implementaci mobilní aplikace pro návštěvníky. Tyto sekce spolu s představováním využitých technologií taktéž krátce zmiňují i obecný popis jejich účelu.

6.1 Serverová část

Sekce 3.4 se krátce věnovala úvodu do možných technologií pro implementaci serverové části. Na základě předešlých zkušeností s informačními systémy byl zvolen jazyk PHP. Jedná se o skriptovací programovací jazyk, určen přednostně pro tvorbu dynamických webových aplikací. I přesto, že ostatní jazyky jako například Python či systém Node.js[1] nabývají rychlého vývoje a dostávají se do popředí, je vhodné uvést, že PHP je celosvětově stále primárním jazykem pro programování serverové části [21]. Největším důvodem pro volbu PHP je ovšem velké množství ověřených a stabilních frameworků, které dokážou posunout PHP na jinou úroveň v realizaci architektury a rozšiřitelnosti – například Laravel, Symfony[2], CodeIgniter[3] apod.

Hlavní využité technologie

V implementaci serverové části systému je spolu s jazykem PHP využit framework Lumen[4]. Tento framework existuje jako alternativa a mikro verze populárního frameworku Laravel. Ve svém základu obsahuje pouze nezbytné a základní komponenty pro vývoj API, s tím, že v případě potřeby lze tyto komponenty a funkcionality manuálně přidat. Rozhodování o konečném PHP frameworku bylo ovlivněno vlastní preferencí, předchozími zkušenostmi s ostatními frameworky a kladeným důrazem na jednoduchost tvorby a rychlost tvořené API. Databáze výsledného systému je založena na NoSQL technologii MongoDB, vytvořena podle návrhu entitně vztahového diagramu na obrázku 5.1, s tím, že v průběhu implementace bylo nutné pozměnit pár detailů jako například úprava některých názvů, přidání dodatečných či vymazání zbytečných atributů.

[1] Node.js https://nodejs.org/en
[2] Symfony https://symfony.com
[3] CodeIgniter https://codeigniter.com
[4] Lumen https://lumen.laravel.com

Framework Lumen poskytuje, stejně jako Laravel, velmi kvalitní programátorské rozhraní pro práci s databázemi spolu s Eloquent ORM[5], díky kterému lze napojit model (z MVC) na konkrétní databázovou kolekci (tabulku) a poté k ní přímo přistupovat bez náročného napojovacího objektu či obav o bezpečnost způsobu zvoleného přístupu. Toto rozhraní umožňuje taktéž velmi jednoduchý a bezpečný způsob napojení na databázi skrz kombinaci konfiguračních souborů (`.env` a `config/database.php`) a rychlou tvorbu databázových dotazů. Velkou výhodou frameworku Lumen (i například Laravel či Symfony) je sjednocený formát konfiguračních souborů k různým typům databáze (databáze lze jednoduše vyměnit bez hlubšího zásahu do struktury projektu). Framework Lumen ve svém základu nepodporuje zvolenou technologii MongoDB, proto je nainstalován balíček laravel-mongodb[6], jež umožňuje pracovat s databázemi MongoDB jako s relačními databázemi skrze zmiňovaný Eloquent s identickou syntaxí.

Architektura serverové části s API

Architektura vytvořeného systému je strukturována jako modifikovaný MVC za cílem snadného vyhodnocování API požadavků. Tyto požadavky jsou nejdříve zpracovány skrz takzvaný *router* (soubor `routes.php`), kde se podle jeho typu a obsahu rozhodne, který controller jej bude zpracovávat. Těsně před předáním požadavku tomuto controlleru se požadavek přiřadí do příslušných middlewarů, kde probíhá primárně autentizace žadatele či například povolení sdílení zdrojů ze systému pro žadatele mimo lokální adresu serveru. Při návratu z middlewarů se požadavek přepošle do zmiňovaného controlleru. Zde probíhá většina logiky systému. Aplikuje na tělo požadavku takzvanou *policy*, která má na starost čistě autorizaci žadatele s příslušnými akcemi. V případě, že žadatel nemá mít k určitým datům přístup, je mu tento přístup odepřen. Controller nadále zvaliduje jednotlivé atributy těla požadavku. Zpracuje je a ve finální podobně je přiřadí odpovídajícímu modelu, který je díky zmiňované Eloquent ORM nádstavbě úzce svázán s databázovou kolekcí. Skrze tento model se vykoná požadavaná akce, načež se žadateli vrací odpověď dle výsledku vyhodnocení zmiňované akce. Celé schéma tohoto procesu lze vidět na zjednodušeném diagramu na obrázku 6.1.

Middlewary nejsou svázané s konkrétními controllery, ale s vybranými přístupovými body – před využitím controlleru lze tedy využít několik middlewarů za sebou. Je vhodné zde uvést ty nejdůležitější, které systém využívá (tabulka 6.1).

Middleware	**Účel**
`Authenticate`	Prvotní zpracovávač požadavků, které vyžadují autentizovaného žadatele
`CompressionMiddleware`	Automatická komprese odpovědí na požadavky
`CorsMiddleware`	Povolení přístupu k datům pro žadatele mimo síť serveru (CORS[7])
`ZooMiddleware`	Kontrola svázanosti autentizovaného uživatele s vybranou zoo

Tabulka 6.1: Middlewary s jejich účely.

[5]Eloquent ORM https://laravel.com/docs/5.0/eloquen
[6]Balíček laravel-mongodb https://github.com/jenssegers/laravel-mongodb

Každý controller má na starost svoji vlastní entitu. Téměř každý controller má v rámci přiřazené entity čtyři hlavní CRUD operace (Create, Read, Update a Delete), rozšířené o metody pro získání všech instancí přiřazené entity v dané zoo. I přesto, že CRUD je svojí podobou určen primárně pro práci s relačními databázemi [9], tak se nejedná o žádnou překážku, a to díky zmiňované knihovně laravel-mongodb spolu s technologií Eloquent. Mimo zmíněné CRUD operace se mohou v controllerech objevit i speciální případy, například controller `AuthApiController` se vůbec neřídí CRUD operacemi, nýbrž poskytuje například metody pro přihlášení, resetování hesla apod. Seznam všech controllerů s jejich úkony lze vidět v tabulce 6.2.

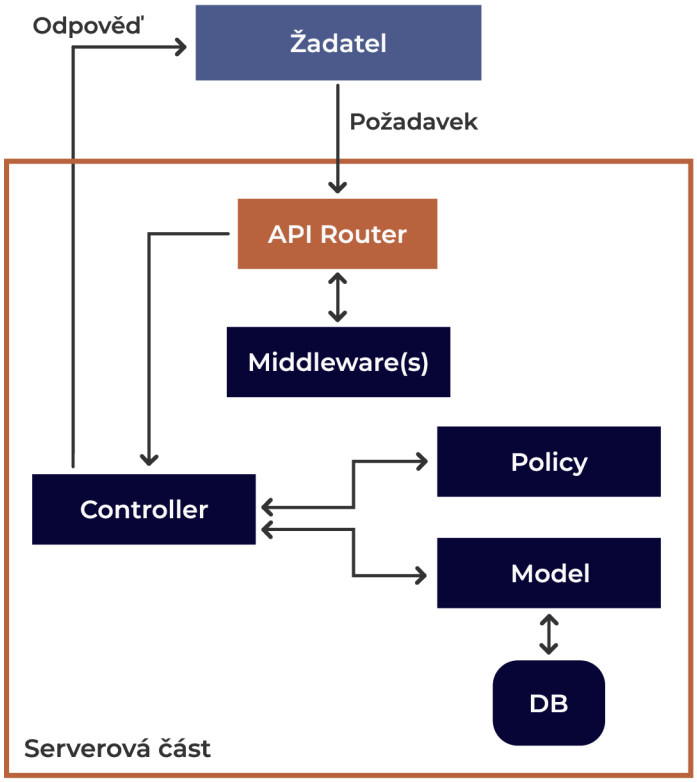
Controller	**Účel**
`ZoosApiController`	Správa informací o zoologických zahradách
`UsersApiController`	Správa uživatelů v systému
`AuthApiController`	Přihlašování, registrace, hesla apod
`SpeciesApiController`	Správa zvířat
`ExpositionsApiController`	Správa expozic
`GalleriesApiController`	Správa galerií zvířat a expozic
`FacilitiesApiController`	Správa zařízení
`EventsApiController`	Správa událostí
`AnnouncementsApiController`	Správa oznámení

Tabulka 6.2: Seznam controllerů s jejich účely.

Spolu s výše uvedenými controllery a middlewary jsou vytvořeny pomocné třídy (takzvané *helpers*). Tyto pomocné třídy poskytují metody, jež se opakovaně využívají napříč controllery při zpracovávání požadavku (tabulka 6.3).

Helper	**Účel**
`AudioHelper`	Poskytuje metody `upload` a `remove`, jež se starají o přidávání a odebírání audio stop
`EmailHelper`	Obsahuje metodu pro sjednocené odesílání emailů ze serveru (přidání uživatele a zapomenuté heslo)
`ImageHelper`	Nabízí metody pro správu obrázkových souborů spolu s jejich verzemi. Pro každý obrázek v systému jsou vytvořeny zkompresované kopie v různých velikostech - 10×Y, 50×Y, 200×Y, 350×Y, 600×Y (hodnoty jsou v pixelech, Y značí přizpůsobenou výšku podle uvedené šířky pro zachování poměru stran)

Tabulka 6.3: Pomocné třídy s jejich účely.



Obrázek 6.1: Diagram vyhotovení požadavku od žadatele.

Komunikace s API a přístupové body

Komunikace se serverem je implementována jako plně bezstavová RESTful API. Její implementace se drží návrhu v podsekci 5.2. Seznam všech přístupových bodů, které systém nabízí, je v příloze této práce (A). Je zde ale vhodné vysvětlit základní strukturu a způsob sestavování takovýchto přístupových bodů. Každý přístupový bod se skládá z vlastní URL, přístupové metody, množiny middlewarů a přiřazeného controlleru. Jako příklad formátu lze uvést skladbu URL pro správu zmiňovaných entit (zoo, zvířata, expozice, zařízení, události a oznámení).

```
https://{address}/api/zoo/{zooId}/{entity}/{entityId}
```

kde **`address`** je doménová adresa pro přístup na serverovou API, dále **`zooId`** je unikátní identifikátor zoologické zahrady v systému, **`entity`** je prvek seznamu species, expositions, announcements, facilities a events (prvky v anglickém překladu odpovídají představeným entitám). Položka **`entityId`** je unikátní identifikátor prvku z množiny **`entity`**. Metody, pod kterými jsou požadavky odesílány, mohou nabývat tří podob (GET, POST nebo PATCH), přičemž každý přístupový bod s URL patří právě jedné metodě.

Vzhledem k tomu, že je tato API bezstavová, bylo nutné vytvořit způsob vhodné komunikace se soubory. Vzorová komunikace předpokládá tělo požadavku ve formátu JSON, do kterého lze vkládat pouze řetězce, čísla, další JSON objekty, pole nebo pravdivostní hodnoty. Neumožňuje tedy vložit soubor přímo do těla požadavku v originálním formátu. Existují způsoby, jak soubor převést na binární či Base64 formát a vložit ho do těla JSON objektu jako řetězec, ale soubory jsou v této podobě podstatně větší a ukládání do mezipa-

měti (*caching*) bývá u příchozích obrázků v tomto formátu problematické. Řešení je tedy implementováno následovně:

- Při posílání souborů na server se využívá typ těla požadavku `multipart/form-data` a daný soubor se do něj klasicky vloží v původním formátu.
- Získávání souborů ze serveru je vyřešeno skrz posílání cesty v kombinaci s názvem souboru – systém si u potřebných záznamů (kde náleží nějaký soubor, například uživatel má profilový obrázek) uchovává pouze cestu a název souboru, zatímco referenční soubor je na serveru uložen ve vlastním adresáři. Server neposílá v odpovědi celý soubor, ale jen zmiňovanou cestu k němu.

Ve frameworku Lumen (i Laravel) je známá chyba při posílání souborů v těle požadavku typu `multipart/form-data` pod metodou PATCH či PUT. Lumen soubory nedokáže v takovéto podobě detekovat, proto se zde využívá alternativní přístup, při kterém se požadavek pošle pod metodou POST a do těla požadavku se přidá atribut `_method` s hodnotou náležící názvu žádoucí metody (PUT, PATCH).

Pro autentizaci v bezstavové komunikaci je využíván takzvaný JWT[8] (JSON Web Token). Server tento token odesílá v těle odpovědi na úspěšně ověřený přihlašovací požadavek. Aplikace ověřeného žadatele musí token uchovat a pro každý požadavek na zabezpečené přístupové body ho vložit do autorizačního atributu se schématem nosiče (`Bearer` v `Authorization`). Serverová část tento token zpracuje a na základě jeho vyhodnocení odešlě příslušnou odpověď. K vytváření tokenů, jejich sestavení či samotné validaci je využita knihovna jwt-auth[9], která poskytuje velice jednoduché rozhraní a přehlednou dokumentaci pro práci s nimi. Používané JWT se skládají ze 3 částí – hlavička, tělo a podpis.

Hlavička obsahuje typ tokenu a šifrovací algoritmus (v tomto případě `HS256`). V těle tokenu jsou přítomny následující informace – tvůrce tokenu (URL serveru), čas vytvoření tokenu a jeho expirace. Dále je zde také aktivační čas tokenu (pokud je aktuální čas menší než aktivační, pak není token funkční), unikátní identifikátor tokenu, identifikátor uživatele (ověřeného žadatele) a v neposlední řadě hash hodnota třídy uživatelů. Podpis tokenu je vytvořen s pomocí šifrované hodnoty `JWT_SECRET` v konfiguračním souboru frameworku (soubor `.env`).

Využité nástroje

Při vývoji serverové části byly využívány tyto nástroje:

- **Vagrant**[10] – kompletně přenosné vývojové prostředí.
- **Homestead**[11] – předchystaný balíček od vývojářů frameworku Laravel pro jednoduchý vývoj na virtuálním Vagrant serveru bez potřeby instalace PHP či webového serveru.
- **Insomnia**[12] – pro efektivní vývoj a testování přístupových bodů API.
- **MongoDB Compass**[13] – grafické rozhraní pro správu lokálních a vzdálených databází v MongoDB.

[8] JSON Web Token https://jwt.io/introduction
[9] jwt-auth https://jwt-auth.readthedocs.io/
[10] Vagrant https://www.vagrantup.com
[11] Homestead https://laravel.com/docs/8.x/homestead
[12] Insomnia https://insomnia.rest
[13] MongoDB Compass https://www.mongodb.com/products/compass

6.2 Administrační webová aplikace

Administrační webová aplikace (rozhraní pro správu) je úzce spojena s vytvořenou API (je umístěna stejném serveru ve stejném adresáři). Framework Lumen disponuje základními možnostmi pro tvorbu frontendu a se svou svázaností s API se může jednat o relativně efektivní řešení, avšak po bližší inspekci potřebných funkcionalit a nevýhod, které tento přístup představuje, byla zvolena možnost klientského vykreslování (*client-side rendering*), primárně s pomocí jazyka Javascript spolu s takzvaným *runtime* prostředím Node.js. Ten nabízí v základu nástroj pro správu balíčku v jazyce Javascript zvaný Node Package Manager[14]. Lze s ním instalovat a spravovat balíčky za účelem rozšiřování aplikací či sdílení vlastního kódu ostatním uživatelům tohoto nástroje (je největším registrem s více než 1.3 miliony balíčky [12]).

Celá webová aplikace je založena na frameworku Vue.js. Ten je vytvořen v jazyce Javascript, využívá se primárně na tvorbu reaktivních aplikací, v posledních letech nabývá velmi vysoké popularity díky své malé datové velikosti, jednoduchosti používání, velké flexibilitě a například také díky své snadné integraci do existujících projektů [20].

Komunikace s API a správa získaných dat

Komunikace mezi serverovým API a aplikací je realizována s pomocí knihovny Axios[15]. Ta zde figuruje jako prostředník pro vytváření a zasílání asynchronních HTTP požadavků, založených na takzvaných slibech (`Promise`), využívá se ale také jako způsob zachycení a zpracování dat z HTTP odpovědi. Například přihlašovací požadavek a uložení přihlašovacího tokenu je implementováno následovně (Axios objekt je představován proměnnou `http`):

```
let jwt = (
    await http.post("/api/login", {
        email: auth.email,
        password: auth.password
    })
).data.token;
localStorage.setItem("jwt", jwt);
```

Tyto požadavky jsou klíčovým prvkem pro reálnou užitečnost aplikace, musí být navíc úzce propojeny se zmiňovaným frameworkem, aby je bylo možné napojit na tlačítka v uživatelském rozhraní. Bylo nutné taktéž vymyslet způsob, jak efektivně uchovávat data napříč celou aplikací a poskytnout komponentám možnosti pro práci s nimi, nehledně na podobu komponenty či její zanoření. Výchozí přístup pro správu dat je ve Vue.js většinou limitován na aktuální `scope` komponenty, a v případě použití takzvaných *props* v hluboce zanořených komponentách vzniká velmi chaotická a komplikovaná struktura. Nastává pak situace, kdy v případě, že je žádoucí získat aktuální hodnotu v trojnásobně zanořené komponentě, je nutné buď vytvořit vyvolávací řetězec nebo celou situaci obejít skrze globální proměnné.

Těmto problémům se staví knihovna Vuex[16], která představuje do aplikace návrhový vzor správy stavů (*state management pattern*). Každé entitě (zvíře, expozice, zařízení apod) je v aplikaci vytvořen vlastní Vuex modul, pojmenován podle entity, se kterou je svázán (například `store/modules/expositions.js` nebo `store/modules/species.js`).

[14] NPM https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm
[15] Axios https://github.com/axios/axios
[16] Vuex https://vuex.vuejs.org

Každý modul obsahuje tyto objekty:

- **Stavy** (*state*) – aktuální hodnoty modulu. Mění se pomocí akcí a většinou představují uchovávaná data z HTTP odpovědí – například aktuálně upravovaná expozice či seznam všech událostí v zoo.
- **Gettery** (*getters*) – funkce pro získání aktuální či modifikované stavové hodnoty – například vyfiltrovaný seznam zvířat bez expozic či originální seznam zvířat.
- **Akce** (*actions*) – funkce pracující se stavy skrze následné mutace – ve většině případů jsou tyto funkce využity pro zmiňované HTTP požadavky skrze Axios, například změna popisku konkrétní expozice.
- **Mutace** (*mutations*) – funkce, které mění hodnoty stavů podle obrdžených parametrů.

Data, která se posílají v HTTP požadavcích, jsou předávána akcím skrz parametry, většinou v datovém typu `FormData` (bližší popis o skládání formulářových dat je v podsekci 6.2).

Přístup na zabezpečené URL

Pro přístup na jakoukoliv stránku administrační části webové aplikace musí být uživatel autentizován. Tato autentizace probíhá skrze zmiňovaný JWT token. Pro zabezpečení všech stránek a jejich jednoduchou správu je využita knihovna Vue Router[17]. Soubor `router.js` skrz tuto knihovnu registruje všechny stránky, které aplikace nabízí, ale hlavně nad nimi využívá metodu `beforeEach`, ve které dokáže zkontrolovat, jestli má uživatel získat k žádané URL přístup. U každé registrované stránky je přidán atribut odlišující veřejnou a zabezpečenou URL. V této metodě aplikace zkontroluje, jestli je žádáná URL veřejná, pokud ne, pak skrz Vuex akci `IsJWTActive` modulu `store/modules/auth.js` vyšle HTTP požadavek na validaci tokenu (i když je prázdný), a podle odpovědi přístup udělí nebo odepře.

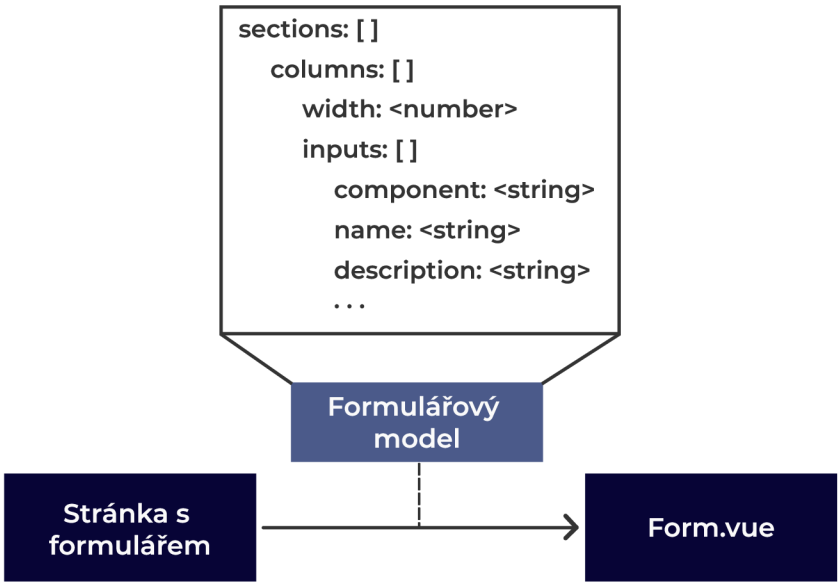
Formulářové vstupy a jejich automatické generování

Z grafických návrhů aplikace (celé jsou dostupné na přiloženém datovém nosiči) lze vidět, že formulářové vstupy na stránkách pro přidávání a úpravy se často opakují, ale pro každou entitu (zvířat, expozic apod) se mění jejich relativní uspořádání a rozměry. Proto bylo nutné vymyslet a implementovat způsob, skrz který půjde jednoduše přidávat, odebírat a měnit uspořádání těchto vstupů, s ohledem na znovupoužitelnost a rozšiřitelnost.

Pro každou stránku, která obsahuje formulář, je použita vytvořená komponenta `Form.vue` (obyčejná Vue.js instance). K této komponentě je přiřazený vždy jeden formulářový model (ze složky `models`) skrz který si komponenta automaticky generuje formulářové vstupy. Pro každou entitu je vytvořen tento speciální formulářový model, který definuje pozici, proporční velikost a typy vygenerovaných vstupů. Takovýto model obsahuje několik formulářových sekcí a každá sekce formuláře obsahuje vždy několik sloupců. Každý sloupec má definovánu proporční šířku vzhledem k ostatním sloupcům (například 1, 2, 1) a obsahuje formulářové vstupy. Tyto formulářové vstupy se přizpůsobují šířce sloupce. U každého formulářového vstupu je uveden jeho typ (podle názvu jeho Vue.js komponenty například

[17] Vue Router https://router.vuejs.org

text, audio), dále název atributu, pod kterým se má prezentovat ve finálním API požadavku. Taktéž obsahuje popisek pro uživatele a v některých případech dodatečné údaje jako například ikona pro výběr obrázku, nebo seznam možností v rozbalovací nabídce.



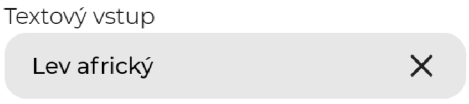
Obrázek 6.2: Zjednodušený diagram generování formulářů.

Komponenta `Form.vue` má zaregistrované všechny vytvořené vstupy, které se formuláři v rámci systému mohou vyskytnout (celkem je vytvořeno 16 vstupů) a s pomocí atributu `component` u každého vstupu ve formulářovém modelu dokáže rozpoznat, který vstup právě vygenerovat. S těmito informacemi úzce pracuje a generuje z nich žádoucí strukturu. Generování je docíleno skrze tři zanořené `foreach` cykly (sekce, sloupce a vstupy). Velmi zjednodušený pseudokód tohoto generování lze sepsat následovně:

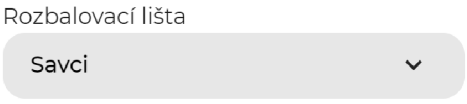
```
create div foreach section in sections and:
    create div foreach column in section.columns and:
        create input component foreach input in column.inputs
        close input component
    close div
close div
```

Celý formulář je sestaven jako rozšířená tabulka, ovšem s tím, že ve svém základu by vizuálně připomínala spíše obyčejný vertikální seznam (i přesto, že DOM je z procesu generování správně připraven). Sekce a sloupce jsou samozřejmě ale doplněné o CSS třídy, skrze které lze realizovat správné relativní pozicování a proporcionální šířku sloupců skrze CSS flexbox. Při změně stavu jakéhokoliv formulářového vstupu (uživatel něco zadá), se vyšle nová hodnota vstupu do metody v nadřažené komponentě `Form.vue` (`updateFormData()`) a pod přiřazeným atributem `name` uvedeným ve formulářovém modelu se přidá do seznamu finálních dat (typ `FormData`). Při dokončení (odeslání) formuláře se aktivuje metoda `submitForm`, která využije již zmiňované akce modulů k vyvolání HTTP požadavku se sesbíranými daty od uživatele. Stránka čeká po odeslání na odpověď a následně buď přiřadí chybové hlášky příslušným formulářovým vstupům, nebo úspěšně pokračuje. Drtivá většina formulářové validace je řešena na serverové části, která u každého atributu těla požadavku validuje jeho podobu a podle výsledku posílá chybovou hlášku směřující na chybný atri-

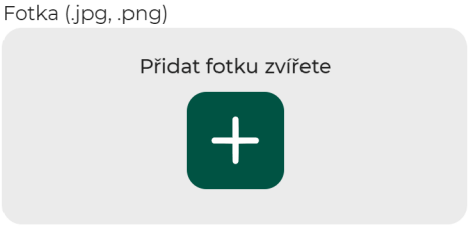
but. Chybová hláška se automaticky zobrazí u odpovídajícího formulářového vstupu (podle ekvivalence názvu atributu chybové hlášky a hodnoty atributu `name` formulářového vstupu v modelu). V aplikaci je vytvořeno mnoho různých vstupů, nicméně je vhodné ukázat jejich podobu. Na obrázcích níže jsou vyobrazeny vstupy pro text (6.3), rozbalovací seznam (6.4), vstup pro obrázek (6.5) a audio soubor (6.6). Dále je pro lepší představu vyobrazena výsledná formulářová stránka pro přidání zvířete s různými vstupy (obrázek 6.7).



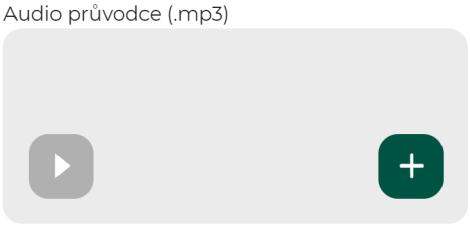
Obrázek 6.3: Textový vstup.



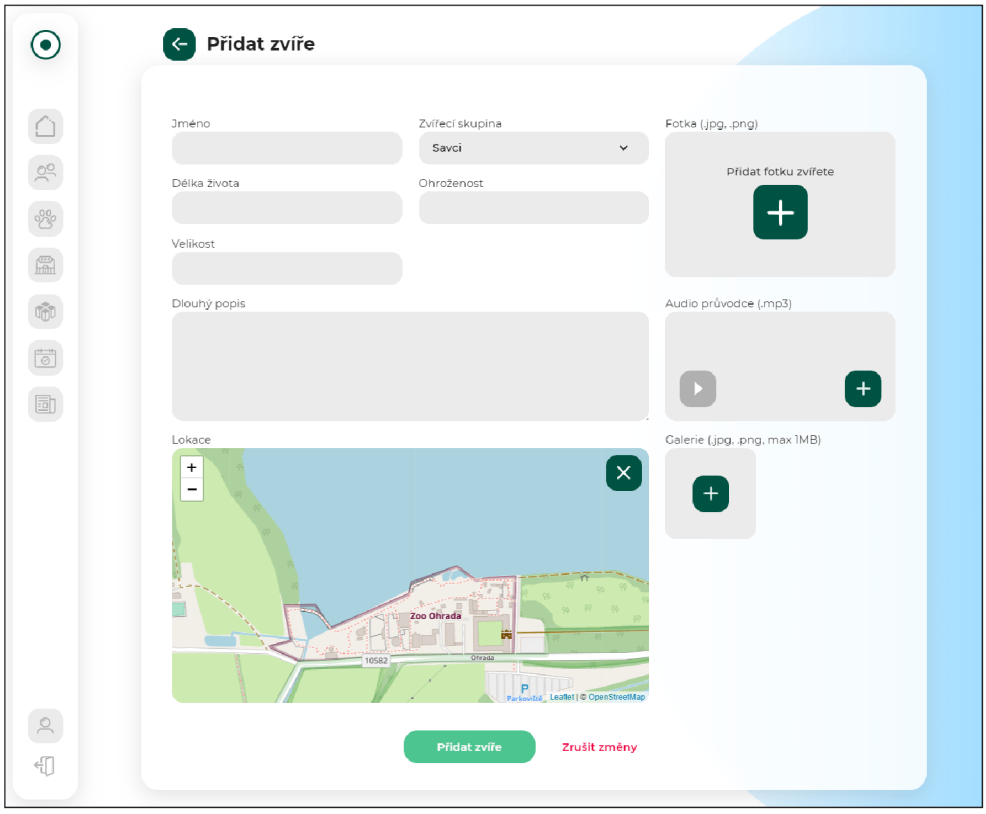
Obrázek 6.4: Rozbalovací lišta.



Obrázek 6.5: Vstup pro obrázek.



Obrázek 6.6: Vstup pro audio soubor.



Obrázek 6.7: Výsledná stránka přidání zvířete s využitím zmiňovaných vstupů.

Stylování

Pro stylování a sestavení struktury komponent byl využit preprocesor SASS, díky kterému lze jednodušeji tvořit znovupoužitelné CSS komponenty a přehledněji tyto styly zanořovat. Framework Vue.js sice v základu podporuje styly s atributem `scoped`, což zařídí absolutní nezávislost stylů, ale vzhledem k velikosti této aplikace a potřeby kompletně oddělit DOM od stylů, je zvolena alternativní možnost – centrální soubor `app.scss`, který importuje všechny potřebné `.scss` komponenty do sebe. Taktéž je zde využita metodologie BEM, což spolu s preprocessorem SASS vytvořilo velmi přehledné prostředí pro vývoj a budoucí nádstavby. Ke každé znovupoužitelné Vue.js komponentě je vytvořena svá vlastní `.scss` komponenta. V aplikaci není využito žádné další CSS rozšíření, neboť z grafického návrhu nabývá aplikace velmi specifického vzhledu a interaktivních vlastností, kterých by se skrz různé knihovny či řozšíření cílilo zbytečně složitě (například Bootstrap[18] většinou slouží spíše jako možnost rychlého prototypování).

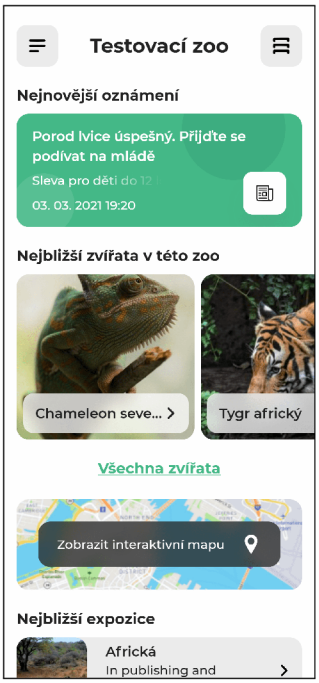
6.3 Mobilní aplikace

I přesto, že se zaměřuje na jiné skupiny uživatelů a jedná se o zcela jinou aplikaci, tak implementační struktura a přístup k mnoha situacím je stejný, jako při implementaci webové aplikace ze sekce 6.2. Důvod k volbě podobných (ne-li stejných) technologií a knihoven pro tvorbu mobilní aplikace vychází z faktu, že mobilní aplikace byla navržena jako hybridní, což znamená, že k její implementaci lze použít jakoukoliv webovou technologii. Další důvod k využití podobných technologií je ten, že během vývoje webové aplikace se použité technologie dostatečně ověřily, ukázaly svůj potenciál a poskytly kvalitní architekturu pro budoucí nádstavby.

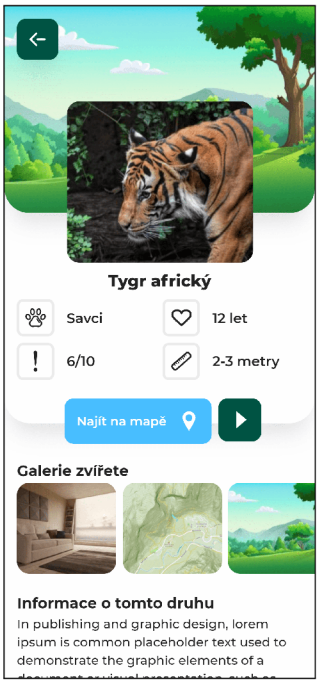
Aplikace proto tedy využívá stejný framework Vue.js jako základní stavební kámen pro implementaci. Komunikace s vytvořenou API (i přesto, že využívá pouze veřejné přístupové body) je realizována taktéž skrze knihovnu Axios, uchovávání stavů se strukturou pro funkce na posílání a zpracovávání API požadavků je implementována s použitím knihovny Vuex. Podobně jako u webové aplikace je zde taktéž využita knihovna Vue Router pro snadnou správu stránek a přechody mezi nimi. Stylování a struktura SASS souborů je zde taktéž stejná. Těmto aspektům se popis implementované mobilní aplikace opětovně nezabývá, nýbrž se věnuje ostatním využitým technologiím/knihovnám, specifickým problémům a jejich řešeními.

Průběžným testováním se odhalilo několik grafických detailů, které bylo nutné jemně poupravit, i přesto ale výsledný vzhled aplikace a funkčnost kvalitně odráží návrh. Na obrázcích níže (6.8 - 6.13) lze vidět reálné snímky obrazovky některých vybraných stránek v mobilní aplikaci.

[18] Bootstrap https://getbootstrap.com

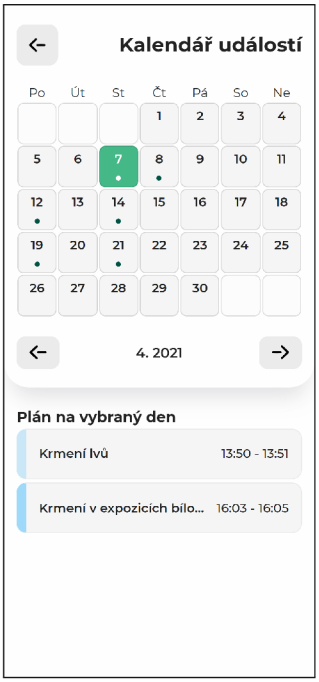


Obrázek 6.8: Domovská obrazovka vybrané zoo.



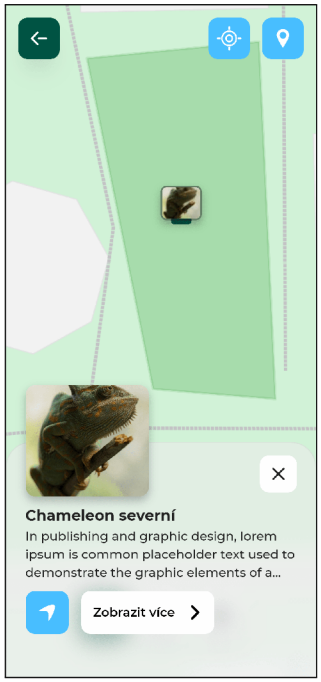
Obrázek 6.9: Stránka s vybraným zvířetem.

Obrázek 6.10: Seznam zvířat ve vybrané zoo.



Obrázek 6.11: Kalendář s událostmi vybrané zoo.

Obrázek 6.12: Interaktivní mapa.



Obrázek 6.13: Vyjíždecí panel zvířete.

Apache Cordova

Mobilní aplikace byla navržena jako hybridní, proto bylo nutné zvolit správnou technologii pro její realizaci. Apache Cordova je jeden z otevřených frameworků pro vývoj těchto hybridních, multiplatformních aplikací. Architektura Cordova se skládá z 3 primárních částí [4]:

- **WebView** – hlavní prostředí pro interní vyobrazení vytvořené aplikace (WebApp).
- **WebApp** – vytvořený zdrojový kód – celá Vue.js aplikace.
- **Pluginy** – doplňující balíčky, které poskytují rozhraní pro komunikace mezi frameworkem Cordova a nativními prvky zařízení (kamera, GPS apod). Tyto pluginy umožňují komunikovat s nativní API zařízení skrze Javascript objekty.

Úkony, které bylo potřeba realizovat v tomto frameworku, je pouhé upravení konfiguračního souboru (název, popisek, verze, ikona apod) a nainstalování cílených platforem příkazem `cordova add platform <platform>`. Během vývoje byla použita knihovna pro jednodušší vývoj Vue.js aplikace s frameworkem Cordova, konkrétně `vue-cli-plugin-cordova`[19]. Díky této knihovně je možné velmi jednoduše sestavit aplikaci v prohlížeči, na virtuálních zařízeních či na připojeném telefonu a jednoduše připravit prostředí pro automatické aktualizace náhledu při změnách ve zdrojovém kódu aplikace. Volba frameworku Cordova byla také kladně ovlivněna množstvím dostupných pluginů v případě potřeby budoucího rozšiřování aplikace.

Lokalizace a oprávnění

V celé aplikaci je využívána lokalita zařízení pro různé účely, od řazení zoo, zvířat a expozicí podle aktuální vzdálenosti, až po vyobrazení uživatele na interaktivní mapě s navigací. Tato lokalita je v aplikaci získávána skrz nativní objekt z WebView – `navigator`[20]. Díky tomuto objektu lze přistupovat k velkému množství informací o zařízení (například vlastnosti aktivního připojení, primárně používaný jazyk zařízení, různá metadata apod). Mezi dostupné atributy patří taktéž atribut `geolocation`, který sám od sebe neobsahuje žádná data, ale disponuje asynchronními metodami k jejich obdržení. Konkrétně se jedná o metody `getCurrentPosition`, která funguje jako jednorázový požadavek na získání aktuální polohy a `watchCurrentPosition`, jež automaticky navrací aktuální polohu při každém registrovaném pohybu. Obě metody vrací data o poloze ve formátu zeměpisných souřadnic (`latitude, longitude`). V aplikaci se primárně používá metoda `watchCurrentPosition`, která ja vyvolána ihned při spuštění aplikace v hlavní komponentě `App.vue`. K jednodušímu přístupu ostatních komponent k aktuální poloze jsou tyto údaje při každé aktualizaci polohy zasílány do modulu `location.js` z knihovny Vuex.

Z bezpečnostních důvodů musí být přístup k lokalizačním údajům povolen uživatelem, proto se při zapnutí aplikace automaticky generuje požadavek na povolení přístupu k geolokaci zařízení. V případě, že uživatel nepodá svolení, pak aplikace ztrácí určité funkcionality, jako například zmiňované řazení dle aktuální vzdálenosti či interaktivní mapu (k interaktivní mapě má mít uživatel přístup pouze tehdy, pokud je blízko vybrané zoo). V reálném provozu bude interaktivní mapa v tomto případě automaticky deaktivována, ale pro účely demonstrace funkcionalit je k datu odevzdání této práce prozatím zapnuta.

[19] vue-cli-plugin-cordova https://github.com/m0dch3n/vue-cli-plugin-cordova
[20] Navigator https://developer.mozilla.org/en-US/docs/Web/API/Navigator

Využité technologie a knihovny pro práci s mapou

Interaktivní mapa s navigací je jednou z primárních funkcionalit celé aplikace. Po důkladném rozboru dostupných technologií byla zvolena knihovna Leaflet, spolu s mapovými dlaždicemi a navigačním API od technologie Mapbox. Google Maps i Mapbox nabízí svůj vlastní způsob, jak integrovat mapy do aplikací (například Mapbox GL JS[21]) a je žádoucí, aby řešení bylo implementováno jednotným způsobem, ale vzhledem k velmi jednoduchému procesu výměny zdroje mapových dlaždic a navigačních API s knihovnou Leaflet, je vhodné se při implementaci neomezovat přímo na jednu technologii a uzavřený ekosystém. Zjednodušeně řečeno, Leaflet je zvolen kvůli potencionální potřebě pro jednoduchou výměnu zdroje mapových dlaždic nebo navigační API (pro Mapbox jsou do určitého počtu požadavků zdarma, poté začínají být zpoplatněny), spolu s jednoznačnou výhodou velkého množství dodatečných knihoven a bohaté komunity. Styly mapových dlaždic jsou pro tuto aplikaci vytvořeny s pomocí aplikace Mapbox Studio, ve které lze efektivně navolit podobu všech zobrazovaných informací (cesty, budovy apod) a velmi jednoduše je poté využít v knihovně Leaflet (skrze obyčejné URL s API klíčem).

Pro navigaci uvnitř vytvořené mapy je nutné ke knihovně Leaflet přidat rozšíření Leaflet Routing Machine[22] (LRM), které umožní využívat navigační funkce s pomocí vzdálených serverů. Ke konfiguraci navigace je přistupováno skrze LRM objekt `Routing` spolu s proprietárními atributy podle zvoleného serveru. Knihovna LRM ve svém základu nabízí možnosti navigace skrz tyto navigační servery: Open Source Routing Machine[23] (OSMR), Mapbox Directions API[24], GraphHopper[25], Mapzen Valhalla[26], TomTom Online Routing API[27] a Esri[28]. Každý z uvedených serverů má svoji proprietární metodu v objektu `Routing`, spolu s individuálními parametry API klíče a konfiguračního objektu. Zde je konkrétně využíván Mapbox Directions API, neboť při výběru již byl investován čas do průzkumu technologie Mapbox a splňuje primární požadavek – pěší navigace, což například server OSMR nepodporuje.

Vytváření mapy a docílení interaktivního chování se v Leaflet realizuje skrze čistý Javascript kód. Aplikace staví ale na frameworku Vue.js, tudíž je mnohem jednodušší a přehlednější využít knihovnu, která dokáže Leaflet integrovat skrz Vue.js komponenty. Přesně tomuto problému se staví knihovna Vue Leaflet[29] (někdy označována jako vue2leaflet). Díky ní lze pracovat s mapou jako s klasickými Vue.js komponentami a často není potřeba psát přebytečný kód mimo tyto komponenty či jejich `props`. Pro bližší představu reálného využití této knihovny lze nastínit jednoduchý způsob inicializace mapy s mapovými dlaždicemi:

```
<l-map :bounds="zoo.location">
    <l-tile-layer :url="tiles"></l-tile-layer>
</l-map>
```

[21] Mapbox GL JS https://www.mapbox.com/mapbox-gljs
[22] Leaflet Routing Machine https://www.liedman.net/leaflet-routing-machine
[23] Open Source Routing Machine http://project-osrm.org
[24] Mapbox Directions API https://docs.mapbox.com/help/glossary/directions-api
[25] GraphHopper https://www.graphhopper.com
[26] Mapzen Valhalla https://www.mapzen.com/blog/valhalla-intro
[27] TomTom Online Routing API https://developer.tomtom.com/routing-api
[28] Esri https://github.com/jgravois/lrm-esri
[29] Vue Leaflet https://vue2-leaflet.netlify.app

Převzaté algoritmy

Ve zdrojovém kódu aplikace se vyskytují pasáže kódu, které byly převzaté nebo vysoce inspirované cizími řešeními. Konkrétně kód umožňující pracovat s navigací jako s Vue.js komponentou `LRoutingMachine.vue`[30]. Tato komponenta je převzatá z repozitáře na serveru Github od uživatele Giordanna[31]. Autor souhlasil s využitím jeho kódu v této aplikaci. Využitá komponenta byla následně rozšířena vlastním kódem o několik dodatečných atributů (například `fitSelectedRoutes` a `router`).

Druhým zdrojovým kódem, který je převzat z cizího zdroje, je algoritmus pro výpočet hrubé vzdálenosti mezi dvěma zeměpisnými souřadnicemi ve funkci `assignDistances` služby `services/distance.js`. Jedná se o ukázkový kód z webu Geodatasource[32].

6.4 Rozšiřitelnost a budoucí nádstavby

Jak již bylo několikrát zmíněno, implementace webové i mobilní aplikace je realizována tak, aby nebyl problém v budoucnu kdykoliv přidat další stránky, funkcionality, či upravit vzhled nebo chování jednotlivých komponent.

Díky automatickému generování formulářových prvků není problém v případě potřeby kamkoliv přidat další vstup (například výskyt zvířete skrz textový vstup, určení potravy apod). Ve webové aplikaci je zkušebně využita knihovna Vue I18n[33], která umožňuje zpřístupnit statické rozhraní více národnostem skrze jazykové mutace. Podobně by se tato knihovna dala nasadit i na mobilní aplikaci. Databáze na tyto jazykové mutace nemá připravené kolekce, ale její způsob implementace nijak nezabraňuje tomuto rozšíření (například podle článku [8]). V případě, že nastane potřeba přidat do aplikace určitou formu gamifikace pro nalákání více návštěvníků, jedná se o pouhou manipulaci s Vue.js stránkami, komponenty a jejich styly.

[30] LRoutingMachine.vue https://github.com/giordanna/vue2-leaflet-routing-machine/blob/master/src/components/LRoutingMachine.vue

[31] Github uživatel Giordanna https://github.com/giordanna

[32] Geodatasource kód https://www.geodatasource.com/developers/javascript

[33] Vue I18n https://kazupon.github.io/vue-i18n

Kapitola 7

Testování a plán do budoucna

Tato kapitola se zaměřuje na testování vytvořeného systému a mobilní aplikace, spolu s budoucím plánem pro celou práci. První sekce 7.1 je zaměřena na testování API a webové aplikace, druhá 7.2 na mobilní aplikaci. Třetí sekce 7.3 se věnuje budoucímu plánu s celým systémem a plánovanému rozšiřování.

7.1 Testování API a webové aplikace

Vytvořené API bylo průběžně testováno nástrojem Insomnia, jak na lokálním, tak na reálném serveru. Během implementace byla vytvořená API spolu s webovou administrační aplikací vložena na virtuální privátní server (VPS) hostován na platformě Digital Ocean. Na ten je napojena koupená doména ourzoo.eu[1]. Tato kombinace ve finálním stavu reprezentuje konečný systém v reálném provozu. V rámci testování bylo v systému vytvořeno několik zkušebních zoo, na kterých se toto testování provádělo. Do uživatelského testování administrační aplikace bylo vhodné zapojit konkrétní zoo, v tomto případě byla kontaktována Zoo Brno. Té byl vytvořen uživatelský účet a odeslán email s pokyny, otázkami a vysvětlením účelu tohoto testování. Tato zoo bohužel na testování neodpověděla, proto bylo nutné toto testování realizovat jiným způsobem.

Aplikace byla vystavena uživatelskému testování lidmi, kteří svými technickými zkušenostmi vhodně reprezentují zaměstnance zoo. Kvalitativní zpětnou vazbu od zoo o chybějících částech či nalezených nedostatcích sice plně neposkytnou, ale mohou alespoň ověřit uživatelskou přívětivost a sdělit celkový dojem ze systému a webového rozhraní. Těmto lidem (celkem 4) byla aplikace krátce představena, byl jim sdělen účel tohoto testování a poté jim byl předán seznam úkolů, díky kterým bylo možné odpozorovat takzvaný *user-flow* a míru přehlednosti pro člověka neseznámeného s rozhraním:

- **První úkol** – vytvořit dva zvířecí druhy, s tím, že nebylo striktně stanovené, jaké formulářové vstupy vyplnit. Ve většině případů se uživatelům podařilo tyto zvířecí druhy vytvořit, ve třech ze čtyř případů nenastal vůbec žádný problém, nicméně u posledního uživatele se ukázalo, že většina chybových hlášek, které se přiřazují k chybným vstupům, jsou v anglickém jazyce, což uživateli znemožnilo zvíře vytvořit, neboť není dostatečně zdatný v anglickém jazyce a nerozumněl chybové zprávě (uživatel nevyplnil povinný vstup – jméno zvířecího druhu). Tyto chybové zprávy jsou generovány automaticky frameworkem Lumen, je tudíž žádoucí přidat automatický překlad validace nebo manuálně přidat chybové zprávy.

[1] Webová aplikace https://ourzoo.eu

- **Druhý úkol** – vytvořit expozici, přiřadit do ní zvířecí druh z prvního úkolu, zvolit lokaci této expozice na mapě a přidat k ní krátkou audio nahrávku. Jediná část, která uživatele prvotně lehce zmátla, je určení lokace na mapě. Dva ze čtyř uživatelů zmínili, že jim nebylo zprvu jasné, že pro určení lokace se využívá klikání na mapu (klikáním na mapu se generují takzvané *markery*, skrze které jde „nakreslit“ polygon pro ohraničení lokace).

- **Třetí úkol** – třetím úkolem bylo vytvoření události, která se opakuje každou středu a pátek od 15:00 do 19:00, začíná 25. dubna 2021 a končí 19. července 2021. Tímto úkolem uživatelé prošli bez sebemenšího problému, ale objevila se zajímavá připomínka. V případě, že člověk chce vytvořit novou událost, musí přejít do stránky událostí s kalendářem, kliknout na tlačíko přidat událost a manuálně vyplnit datum události. Mnohem lepší řešení by bylo mít možnost kliknout na vybraný den v kalendáři, čímž by se automaticky přešlo do stránky na vytváření události, s tím, že datum události by byl automaticky vyplněn podle dne, na který bylo v kalendáři kliknuto.

- **Čtvrtý úkol** – poslední úkol spočíval ve vymazání všeho, co se během předešlých tří úkolů vytvořilo. Úkol bez problému splnili všichni uživatelé.

Toto testování poskytlo velmi kvalitní zpětnou vazbu pro odhalení nedostatků, se kterými by se vývojář nesetkal, nebo by si je neuvědomil.

7.2 Testování mobilní aplikace

Mobilní aplikace je v aktuálním stavu dostupná na platformě Android skrz distribuční nástroj Google Play[2]. Na platformu iOS je zdrojový kód aplikace připraven, ale pro její reálné nasazení je nutné ročně hradit poplatek 99 dolarů a bez zařízení s operačním systémem macOS nelze aplikaci přeložit do potřebné podoby.

Nehledně na cílovou platformu bylo díky dostupnosti na Google Play možné realizovat dvě fáze uživatelského testování, díky kterému bylo možné poznat názor jiné osoby a získat cennou zpětnou vazbu. Toto testování probíhalo skrze online dotazník (dotazník je dostupný na přiloženém datovém nosiči v souboru `questionnaire_mobile_testing.pdf`), který obsahoval otázky k zjištění názoru návštěvníků na aplikaci jako takovou. Dotazovaní jsou s aplikací velmi spokojeni jak po grafické stránce (v mezích 1-5 je průměr 4,6), tak po stránce přehlednosti (průměr 4,3). Dotazovaní mohli uvést, kterou funkcionalitu by uvítali v aplikaci jako další. Možnost stát se adoptivním rodičem a přidání určité formy gamifikace byly nejpopulárnějšími možnostmi. Skrze tento dotazník bylo možné taktéž zjistit nedostatky aplikace či problémy, se kterými se dotazovaní potýkali. V rámci výkonu a plynulosti tato aplikace získala průměrné hodnocení 3,6. Mezi hlavní nedostatky podle dotazovaných patřily:

- tlačítka s ikonami jsou příliš malé, a často nedostatečně viditelné,
- na stránce s interaktivní mapou není ihned jasné, k čemu slouží modré tlačítko v pravém horním rohu,
- na starších zařízeních se aplikace seká, někdy i na novějších zařízeních v menu,
- při pohybu se zařízením během spuštěné navigace se mapa neustále oddaluje.

[2]Odkaz na Google Play aplikaci https://play.google.com/store/apps/details?id=com.ourzooapp

Problémy číslo 1, 2, a 4 byly ihned opraveny. Problém záseků částečně přetrvává, neboť se bohužel jedná o častý nedostatek starších zařízení a relativně velké náročnosti na grafický čip skrz WebView. Zároveň ale byla opravena náročná animace při otevírání menu, která způsobovala záseky i na novějších zařízeních. Mimo tyto poznatky se vyskytovaly poznámky k různým detailům, například že odsazení stránky ze spodní strany by mohlo být větší, nebo že dlouhé názvy událostí jsou někdy useknuté. Součástí dotazníku byla taktéž otázka, jestli by aplikaci doporučili ostatním návštěvníkům zoo. Mezi 13 respondenty převažovala kladná odpověď s **84,6** procenty, lze tedy s určitou jistotou uvést, že aplikace je užitečná a plní stanovené cíle.

Uživatelé byli v druhé fázi (při osobním testování) vystaveni několika úkolům, díky kterým bylo možné odhalit míru jednoduchosti používání. Úkoly byly následující:

- najděte klokana v Zoo Brno na mapě a zjistěte jeho ohroženost,
- spusťte audio nahrávku chameleona v Zoo Hluboká,
- zjistěte, jaké události jsou v plánu na zítřejší den v Zoo Olomouc a co zajímavého se odehrálo minulý týden.

Všem účastníkům testování se bez jakéhokoliv záseku úspěšně podařilo splnit každý úkol, což do určité míry ověřuje žádoucí přehlednost a jednoduchost použití i pro nové uživatele.

7.3 Plán do budoucna

Problémy, které bude muset systém konfrontovat, se týkají převážně důvěryhodnosti a záruky budoucí správy, neboť větší zoo, které mají svůj vlastní systém, nemají z počátku důvod v systému informace plnit (z důvodu časové náročnosti plnění nebo problému duplicity). Je v plánu tento systém z počátku soustředit na menší zoologické zahrady, které nemají finanční částku pro realizaci mobilní aplikace či informačního systému. S těmito zoo se povede úzká, individuální spolupráce (někdy i skrz zpoplatněnou formu), díky které bude možné opravit chyby a přidat do systému co nejvíce zajímavých komponent. Vedle této spolupráce je v plánu do systému postupně vkládat i záznamy pro zoo, které si o účast v systému nezažádali, jelikož je hlavní, aby aplikace byla užívaná co nejvíce návštěvníky, nehledně na vybranou zoo. V další fázi, ve které bude systém obsahovat dostatečné množství zoologických zahrad, a mobilní aplikace bude veřejně známou možností kapesního průvodce, je vhodné začít systém určitým způsobem monetizovat, což zajistí jistotu budoucí správy celého systému a rozvoje, spolu s důvěryhodností, které větší zoo potřebují.

Ještě před vylepšováním systému je nutné opravit nalezené nedostatky a pokusit se zaměřit na přístupnost pro zaměstnance zoo (například problém s anglickým jazykem chybových zpráv apod). Po opravení nalezených chyb je jako první vylepšení celého systému s aplikací aktuálně přidání možnosti pro návštěvníky stát se adoptivním rodičem, či přispět určitou finanční částkou. Tato komponenta by sloužila nejen jako motivace pro zoo k účasti na systému, ale také by ji šlo po určité domluvě se zoo využít jako finanční zdroj na budoucí rozvoj systému (například určitá fixní částka či procento ze všech příspěvků skrz vytvořenou mobilní aplikaci). Tato možnost nepřinese zahradám ani návštěvníkům žádné nevýhody. Zahrady budou získávat více financí skrz příspěvky z mobilní aplikace, návštěvníci nebudou nijak nuceni do jakékoliv platby, a systém se bude moci dále rozvíjet a bude do určité míry soběstačný (například cena serveru, doména, poplatky za distribuční platformy apod).

Kapitola 8

Závěr

Cílem této práce bylo vytvořit sjednocující informační systém pro zoo, spolu s moderní mobilní aplikací pro návštěvníky. K návrhu řešení bylo v první fázi nutné zpracovat problematiku využívaných možností k prezentaci informací návštěvníkům zoo. V rámci teoretické části práce bylo důležité taktéž prostudovat principy tvorby informačního systému, návrhu architektury, komunikace se serverem a API, uživatelských rozhraní a možností realizace klientských aplikací. Vedle tohoto průzkumu byly taktéž prostudovány možnosti pro tvorbu interaktivních map v aplikacích. K detailnímu ověření aktuálního stavu v zoo a zjištění konkrétních potřeb bylo nutné zanalyzovat názory samotných zoo a jejich návštěvníků.

Na základě těchto průzkumů, konzultací a dotazníků byl vytvořen návrh celého systému (architektura, webová aplikace pro správu a mobilní aplikace pro návštěvníky). Tento návrh sestával z určení množiny funkctionalit každé části systému a následného vytvoření propracovaných grafických návrhů. Během tohoto navrhování již bylo taktéž možné začít uvažovat nad technologiemi, které jsou využity při implementaci.

Implementaci bylo nutné rozdělit do několika částí. Serverová část je založena na frameworku Lumen, nabývá podoby plně bezstavové RESTful API. Veškerá funkcionalita administrační aplikace je založena na komunikaci s touto API. Webová aplikace využívá všechny přístupové body API a je založena na frameworku Vue.js. Mobilní aplikace tuto API využívá čistě jako zdroj dat. Je implementována jako hybridní ve frameworku Apache Cordova, v kombinaci s frameworkem Vue.js. Práce s interaktivní mapou je realizována s pomocí knihovny Leaflet v kombinaci s technologiemi Mapbox.

V průběhu práce byla mírně zaměňována vizuální podoba a funkčnost určitých komponent podle zpětné vazby. Implementovaná řešení byla testována lidmi reprezentující cílovou skupinu dané části, čímž bylo možné ověřit míru použitelnosti a zjistit uživatelský názor. Výsledné řešení kvalitně reflektuje uživatelské požadavky a dosahuje cílů stanovených na začátku této práce.

Při vývoji byl objeven nečekaně velký potenciál pro budoucí rozvoj vytvořeného systému. Celý systém nabírá zajímavého rozsahu a s dobrou strategií a bussiness plánem by se z něj mohl vytvořit reálný startup, který by se zpočátku zaměřoval na požadavky menších zoo a chovatelů, kteří nemají dostupnou finanční částku na mobilní aplikaci nebo kteří zatím nemají žádný informační systém. Během této cesty bude nutné ještě konfrontovat velké množství problémů (hlavně co se důvěryhodnosti vzhledem k větším zoo týče), nicméně s implementovaným řešením, které nabízí velké množství rozšiřitelnosti, se jedná o správný krok pro budoucí nádstavby.

Literatura

[1] Ater, T. *Building progressive web apps: bringing the power of native to the browser.* 1. vyd. Sebastopol, CA: O'Reilly Media, 2017. ISBN 9781491961629.

[2] Babich, N. *Sketch, Wireframe, Mockup, and Prototype: Why, When and How* [online]. 2020 [cit. 2021-01-21]. Dostupné z: https://uxplanet.org/sketch-wireframe-mockup-and-prototype-why-when-and-how-29a25b3157c4.

[3] Feoktistov, I. *Choosing a Map API: Mapbox vs OpenStreetMap vs Google Maps* [online]. 2021 [cit. 2021-04-21]. Dostupné z: https://relevant.software/blog/choosing-a-map-amapbox-google-maps-openstreetmap.

[4] Foundation, T. A. S. *Architectural overview of Cordova platform - Apache Cordova* [online]. Únor 2021 [cit. 2021-04-19]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview.

[5] Fowler, M. *Destilované UML.* 2. vyd. Praha: Grada, 2009. ISBN 9788024720623.

[6] Hague, P. *Market research in practice: how to get greater insight from your market.* 2. vyd. London: Kogan Page, 2013. ISBN 0749468645.

[7] Hammink, J. *The Types of Modern Databases | Alooma* [online]. Březen 2018 [cit. 2021-01-31]. Dostupné z: https://www.alooma.com/blog/types-of-modern-databases.

[8] Herten, F. V. der. *How To Add Multilingual Support to Eloquent | Laravel News* [online]. Září 2015 [cit. 2021-04-21]. Dostupné z: https://laravel-news.com/how-to-add-multilingual-support-to-eloquent.

[9] Logic, S. *What is CRUD? Explaining CRUD Operations | Sumo Logic* [online]. 2021 [cit. 2021-03-20]. Dostupné z: https://www.sumologic.com/glossary/crud.

[10] MDN. *Server-side web frameworks - Learn web development | MDN* [online]. Leden 2021 [cit. 2021-01-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks.

[11] Nielsen, J. *Usability 101: Introduction to Usability* [online]. Leden 2012 [cit. 2021-01-20]. Dostupné z: https://www.nngroup.com/articles/usability-101-introduction-to-usability/.

[12] Prakash, A. *With npm, Microsoft Now Owns the Largest Software Registry in the World* [online]. Březen 2020 [cit. 2021-04-14]. Dostupné z: https://itsfoss.com/microsoft-npm-acquisition.

[13] Preece, J. *Interaction design : beyond human-computer interaction.* 4. vyd. Chichester, West Sussex: John Wiley & Sons Ltd, 2015. ISBN 1119020751.

[14] Rakestraw, T. L., Eunni, R. V. a Kasuganti, R. R. The mobile apps industry: A case study. *Journal of Business Cases and Applications.* 1. vyd. Citeseer. 2013, sv. 9.

[15] Reichel, J. *Kapitoly metodologie sociálních výzkumů.* 1. vyd. Praha: Grada, 2009. ISBN 9788024730066.

[16] Anthony. *Why Rounded Corners Are Easier on the Eyes* [online]. 2011 [cit. 2021-01-30]. Dostupné z: https://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes/.

[17] Sklenář, V. *Data, informace, znalosti a Internet.* 1. vyd. Praha: C.H. Beck, 2001. ISBN 80-7179-409-0.

[18] Suchá, L. Z., Kocurek, J., Ondrášková, M. a Kalíšek, P. *100 metod* [online]. 2016 [cit. 2020-12-18]. Dostupné z: https://100metod.cz/tagged/v%C3%BDzkum.

[19] Tuch, A. N., Presslaber, E. E., Stöcklin, M., Opwis, K. a Bargas Avila, J. A. The role of visual complexity and prototypicality regarding first impression of websites: Working towards understanding aesthetic judgments. *International Journal of Human-Computer Studies.* Listopad 2012, č. 11. ISSN 1071-5819. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1071581912001127.

[20] Vivek. *7 advantages of using Vue.JS | The Progressive Framework* [online]. Srpen 2018 [cit. 2021-04-16]. Dostupné z: https://www.inkoop.io/blog/7-advantages-of-using-vue-js.

[21] W3techs. *Usage Statistics and Market Share of PHP for Websites, February 2021* [online]. 2021 [cit. 2021-02-15]. Dostupné z: https://w3techs.com/technologies/details/pl-php.

[22] Šmíd, V. *Specifikace informačního systému* [online]. Duben 2002 [cit. 2021-01-20]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsysspec.htm.

Příloha A

Přístupové body API

Metoda	URL
POST	/api/login
POST	/api/password/email
POST	/api/password/reset
GET	/api/check-token
GET	/api/roles

Tabulka A.1: Pomocné API body.

Metoda	URL
GET	/api/zoos
GET	/api/zoos/{zooId}
POST	/api/zoos
PATCH	/api/zoos/{zooId}
POST	/api/zoos/{zooId}/delete

Tabulka A.2: Správa zoologických zahrad.

Metoda	URL
GET	/api/users/{userId}
GET	/api/zoos/{zooId}/users
PATCH	/api/users/{userId}
POST	/api/zoos/{zooId}/users
POST	/api/users/{userId}/delete

Tabulka A.3: Správa uživatelů.

Metoda	URL
GET	/api/zoos/{zooId}/galleries
POST	/api/zoos/{zooId}/galleries
POST	/api/zoos/{zooId}/galleries/{galleryId}/delete
POST	/api/zoos/{zooId}/galleries/{galleryId}/{image}/delete
POST	/api/zoos/{zooId}/galleries/{galleryId}/delete

Tabulka A.4: Správa galerií.

Metoda	URL
GET	/api/species/types
GET	/api/zoos/{zooId}/species
GET	/api/species/{speciesId}
PATCH	/api/zoos/{zooId}/species/{speciesId}
POST	/api/zoos/{zooId}/species
POST	/api/zoos/{zooId}/species/{speciesId}/delete
PATCH	/api/zoos/{zooId}/species/{speciesId}/unbindexposition

Tabulka A.5: Správa zvířecích druhů.

Metoda	URL
GET	/api/expositions/{expositionId}
GET	/api/zoos/{zooId}/expositions
PATCH	/api/zoos/{zooId}/expositions/{expositionId}
POST	/api/zoos/{zooId}/expositions
POST	/api/zoos/{zooId}/expositions/{expositionId}/delete

Tabulka A.6: Správa expozic.

Metoda	URL
GET	/api/announcements/{announcementId}
GET	/api/zoos/{zooId}/announcements/latest
GET	/api/zoos/{zooId}/announcements
PATCH	/api/zoos/{zooId}/announcements/{announcementId}
POST	/api/zoos/{zooId}/announcements
POST	/api/zoos/{zooId}/announcements/{announcementId}/delete

Tabulka A.7: Správa oznámení.

Metoda	URL
GET	/api/facilities/types
GET	/api/facilities/{facilityId}
GET	/api/zoos/{zooId}/facilities
PATCH	/api/zoos/{zooId}/facilities/{facilityId}
POST	/api/zoos/{zooId}/facilities
POST	/api/zoos/{zooId}/facilities/{facilityId}/delete

Tabulka A.8: Správa zařízení.

Metoda	URL
GET	/api/events/{eventId}
GET	/api/zoos/{zooId}/events
PATCH	/api/zoos/{zooId}/events/{eventId}
POST	/api/zoos/{zooId}/events
POST	/api/zoos/{zooId}/events/{eventId}/delete

Tabulka A.9: Správa událostí.