FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Front-end informačního systému pro podporu chovu šneků
Student:	Jakub Rigoci
Vedoucí:	Mgr. Petr Matyáš
Studijní program:	Informatika
Obor / specializace:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce zimního semestru 2022/2023

Pokyny pro vypracování

Tato bakalářská práce je součástí projektu, na němž spolupracuje více studentů. Jeho cílem je vytvořit informační systém pro chovatele šneků. Úkoly této bakalářské práce jsou následující:

- Proveďte rešerši informačních systémů pro podporu chovu zvířat.
- Vypracujte studii proveditelnosti tohoto informačního systému s důrazem na front-end:
  * ve spolupráci se spoluautorem navrhněte architekturu celé aplikace,
  * ve spolupráci se spoluautorem projektu analyzujte funkční a nefunkční požadavky na informační systém pro chovatele šneků,
  * analyzujte business požadavky,
  * analyzujte požadavky na uživatelské rozhraní,
  * navrhněte architekturu front-endu.
- Na základě provedené analýzy implementujte prototyp front-endové části aplikace.
- Veškeré softwarové výstupy důkladně otestujte.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 5. května 2021 v Praze.

Bakalárska práca

FRONT-END INFORMAČNÍHO SYSTÉMU PRO PODPORU CHOVU ŠNEKŮ

Jakub Rigoci

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedúci: Mgr. Petr Matyáš
7. februára 2022

Obsah

Zoznam obrázkov

Zoznam výpisov kódu

Chcel by som sa poďakovať hlavne vedúcemu tejto práce, pánovi Mgr. Matyášovi Petrovi za jeho pomoc, podporu a trpezlivitu pri tvorbe. Taktiež ďakujem mojej rodine a priateľom za psychologickú podporu.

Vyhlásenie

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 7. februára 2022 .....

Abstrakt

Táto práca sa zaoberá štúdiou uskutočniteľnosti a tvorbou front-endu informačného systému na podporu chovateľov slimákov. Tento systém sa zameriava na zjednodušenie uchovávania informácií spojených s chovom slimákov. Práca najprv analyzuje požiadavky, porovnáva súčasné technológie tvorby front-endových aplikácií, a následné daný front-end implementuje. Súčasťou je aj spomínaná štúdia uskutočniteľnosti, ktorá mapuje ekonomické a manažérske aspekty tvorby daného informačného systému.

Klíčová slova informačný systém, podpora chovateľov slimákov, front-end, štúdia uskutočniteľnosti, VueJs

Abstract

This work deals with feasibility study and creation of a front-end for information system for support of snail breeders. This information system focuses on simplification of information storage linked with snail breeding. The work first analyzes the requirements, compares the current technologies of creating front-end applications, and then implements the front-end. The mentioned feasibility study is also a part of the work, and it maps the economic and managerial aspects of the creation of the given information system.

Keywords information system, support for snail breeders, front-end, feasibility study, VueJs

Kapitola 1
Úvod

Chov zvierat je už od nepamäti súčasťou života ľudí, či už ide o chov za účelom potravy alebo chov domácich zvierat. Hoci, azda najpopulárnejšími domácimi zvieratami sú psy a mačky, netreba zabúdať ani na iné zvieratá, ako napríklad slimáky. Chov slimákov je síce určite menej populárny ako chov vyššie spomínaných psov a mačiek, má však aj tak dostatok vášnivých chovateľov. Tí si o slimákoch zapisujú rôzne údaje: od druhu jedincov, cez množstvo slimákov, až po dekorácie, s ktorými spomínané slimáky žijú. Mnoho z nich je však nútených si tieto údaje zapisovať do rôznych, nie vždy vyhovujúcich, textových aplikácií alebo rovno na papier.

Preto prišiel návrh vytvoriť informačný systém pre chovateľov slimákov, ktorý by uchovávanie informácií zjednotil a zjednodušil.

Nasledujúce sekcie sa budú zaoberať štúdiou uskutočniteľnosti systému ako aj analýzou, návrhom a implementáciou front-endu tohto informačného systému.

Táto práca nadväzuje na bakalársku prácu „Informační systém pro podporu chovu šneků“ [1] od autora Mikolasa Tesku

Cieľ práce

Cieľom tejto práce je analyzovať požiadavky na informačný systém pre podporu chovateľov slimákov, vytvoriť štúdiu uskutočniteľnosti na základe potrieb zadávateľa, navrhnúť architektúru daného systému, vytvoriť prototyp front-endu systému a daný front-end systému otestovať.

Na dosiahnutie cieľu práce je potrebné analyzovať požiadavky pre daný informačný systém. V rámci analýzy budú prediskutované funkčné a nefunkčné požiadavky, spolu s bussiness požiadavkami pre informačný systém.

Po zanalyzovaní požiadavok bude vytvorená štúdia uskutočniteľnosti, ktorá bližšie priblíži využiteľnosť systému v prostredí chovateľov slimákov, ako aj zmapuje nároky na jeho chod.

Taktiež bude na základe analýzy vytvorený návrh architektúry aplikácie, pričom táto práca sa bude sústrediť na jej front-end. Výsledkom práce je zároveň aj prototyp front-endu informačného systému, ktorý bude otestovaný autorom tejto práce.

Kapitola 2
Analýza

2.1 Špecifika chovu slimákov

Chov slimákov nepatrí k najrozšírenejším koníčkom. Aby sa ale aj neznalý čitateľ mohol lepšie orientovať v tejto práci, bolo pripravené malé zhrnutie dôležitých informácií o slimákoch. Slimáky sú hermafrodity, to znamená, že sú obojpohlavné živočíchy, preto k oplodneniu dochádza nezávisle od pohlavia slimáka. K splodeniu vajíčka sú potrební dvaja partneri, ale existuje aj tzv. „nepoškvrnené počatie“, pri stačí iba jeden slimák. Toto je však veľmi vzácne a pri tvorbe systému to neberieme do úvahy. Slimáky sú schopné aj tzv. „oneskoreného oplodnenia“, kedy slimák dokáže zadržať spermie na dlhší čas, pokojne aj rok. To je aj dôvodom prečo vo väčšine prípadov je nemožné zistiť a uchovávať otca ako rodiča slimáka. Takisto medzi druhové kríženie je medzi chovateľmi veľké tabu. Potomkovia slimákov z rovnakého rodu nie sú najlepším genetickým materiálom, a taktiež to má za následok vytrácanie pestrosti slimákov. Po oplodnení slimák znesie vajíčka. Tento akt sa nazýva „snúška“. Pri snúške má zmysel sledovať jej dátum a interval jej liahnutia. Podľa druhu slimáka môže snúška obsahovať rádovo desiatky až stovky vajíčok. Po vyliahnutí je na chovateľovi, ako sa s novými slimákmi vysporiada. Väčšinou si nechá veľké a výzorovo zaujímavé jedince a ostatné sa buď zamrazia, čo je považované ako najhumánnejší spôsob zbavenia sa slimákov, alebo môžu byť poskytnuté ako potrava iným živočíchom. Samotný slimák sa dožíva vyššie jednotky rokov a k životu potrebuje dve kľúčové veci. Vlhkosť a vápnik. Preto sa slimáky väčšinou chovajú v boxoch, kde sa dá pomerne ľahko zaručiť požadovaná vlhkosť. Aby im chovatelia pobyt v boxe trošku spríjemnili, pridávajú im na spodok lignocel – kokosovú drť ktorá slúži ako podstielka. Zároveň môžu do boxu pridávať rôzne dekorácie v podobe rastlín, samozrejme je dôležité aby spomínané rastliny zvládli nadmerné vlhko v boxoch. Slimáky sa najčastejšie kŕmia ovocím a zeleninou. Kŕmiť ich stačí obdeň, je však dôležité vybrať starú neskonzumovanú zeleninu a ovocie, pretože vo vlhkom prostredí boxu veľmi ľahko zhnije.

2.2 Prieskum systémov na podporu chovateľov zvierat

Keďže chov slimákov je veľmi špecifický a informačný systém zaoberajúci sa chovom slimákov sa nepodarilo nájsť, v tejto sekcii budú popísané už existujúce systémy na podporu chovateľov iných zvierat.

2.2.1 Cavalierclub

Cavalierclub [2] je dobrovoľné združenie pre chovateľov, majiteľov a priaznivcov psov plemena *Cavalier King Charles Spaniel*. Na stránke si uverejňujú aktuality, termíny domácich, ale aj medzinárodných akcií. Takisto sa môže človek dozvedieť informácie o plemene *Cavalier King Charles Spaniel* ako sú napríklad: charakteristika a história plemena, FCI štandard, ale aj rôzne choroby psov a spôsoby prevencie. Najzaujímavejšia časť je však databáza psov. V databáze sú uchovávané informácie o psoch.

Uchovávané sú:

- meno psa,
- rok narodenia,
- otec,
- matka,
- majiteľ,
- chovateľ,
- pohlavie,
- farba,
- číslo zápisu,
- chovnosť,
- súrodenci,
- potomci.

Informácie o rodičoch a potomkoch sú potom využité na vytvorenie rodokmeňa, ktorý môže siahať až po tri generácie. Zároveň sú uchovávané aj informácie o zdravotných vyšetreniach psa.

Výhody:

- prehľadná databáza psov,
- pravidelná aktualizácia udalostí a aktualít na stránke.

Nevýhody:

- zastaralé užívateľské rozhranie,
- zameranie iba na jedno plemeno psov.

2.2.2 Plemenársky informačný systém

Druhým systémom je Plemenársky informačný systém [3]. Ako už názov napovedá, ide o systém na podporu chovateľov dobytka. Systém sa delí na dve zóny. Voľná zóna a zabezpečená zóna. Bohužiaľ, pre prístup do zabezpečenej zóny je potrebné byť chovateľom dobytka a absolvovať zložitý registračný proces. Preto sa táto časť bude venovať hlavne voľnej zóne. Táto zóna ponúka verejne dostupné informácie ohľadom dobytka a chovateľov. Pri vstupe do voľnej zóny si užívateľ môže vybrať z jednotlivých podkategórií ako sú: kravy, býky, ošípané, ovce alebo kozy. Taktiež sú tu informácie o jednotlivých chovateľoch dobytka. Aplikácia obsahuje veľké množstvo rôznych údajov zobrazovaných hlavne vo forme tabuliek. Na vyhľadávanie a filtrovanie dát slúžia vyhľadávacie polia, ktoré sa nachádzajú v hornej časti väčšiny tabuliek. Dáta sa dajú triediť zostupne a vzostupne kliknutím na hlavičku v tabuľke. Nad tabuľkami sa nachádzajú ikony na ďalšiu prácu s údajmi. Dá sa tu vybrať mimo iné aj: export do CSV, export do PDF, stiahnutie dát, úprava stránkovania dát (všetky dáta budú zobrazené na jednej stránke nezávisle od ich počtu) alebo pripraviť dáta na tlač.

Výhody:

- rozsiahle možnosti exportu,
- rozsiahle možnosti filtrácie a triedenia tabuliek.

Nevýhody:

- niektoré prvky užívateľského rozhrania nesedia do celkového vzhľadu stránky.

2.2.3 Gespet: software for pet professional - Pet apps

Tretím systémom je aplikácia GesPet [4]. Aplikácia slúži pre chovateľov psov. Pri vstupe na stránku má užívateľ možnosť výberu spomedzi viacerých balíčkov s rôznymi možnosťami využitia. Nachádzajú sa tu balíčky pre psí obchod, psí hotel, balíček pre psích trénerov a mnoho ďalších. Aplikácia GesPet je platená, s cenami za balíček od 12.95 eur do 28.90 eur na mesiac. Pre prieskum aplikácie bola použitá demo verzia, ktorá je zadarmo. Hlavný náhľad sa mení v závislosti od vybratého balíčka, vo všeobecnosti sa tu ale nachádza okno s výberom dátumu, okno na pridávanie a editáciu dát súvisiacich s vybraným balíčkom, okno s úlohami, okno s upozorneniami a okno s udalosťami. Všetky okná sa dajú upravovať a zobrazovať v detailnom okne. V hornej lište sa nachádza menu pre rýchly prechod na tvorbu nových zápisov, zobrazenie podrobných štatistík, ale aj na prechod do iných balíčkov. Za zmienku stojí, že aplikácia v dobe skúšania bola pomalá (niekoľko sekúnd na zobrazenie okna). To má pravdepodobne na svedomí obrovská komplexnosť aplikácie.

Výhody:

- rozsiahle funkcionality,
- rozdelenie aplikácie na balíčky, užívateľ nemusí platiť za veci, ktoré nevyužije,
- možnosti zobrazenia podrobných štatistík.

Nevýhody:

- rýchlosť,
- užívateľské rozhranie môže pôsobiť zmätočne.

2.3 Funkčné a nefunkčné požiadavky

Pri tvorbe systému boli zadávateľom jasne definované funkčné a nefunkčné požiadavky na systém. Vymedzenie týchto požiadaviek prebehlo formou diskusie so zadávateľom.

2.3.1 Funkčné požiadavky

V nasledujúcej sekcií sú popísané požiadavky na funkcionality systému.

2.3.1.1 Registrácia užívateľov

Užívatelia sa budú schopní zaregistrovať do systému. Registrácia prebehne poskytnutím osobných údajov a následným aktivovaním administrátorom. Aktivácia bude časovo obmedzená pre komerčné účely a bude musieť byť predlžovaná administrátorom. Po registrácií a následnej aktivácií administrátorom bude mať užívateľ právo na používanie ostatných funkcionalít systému.

2.3.1.2 Administrácia užívateľov

Administrátor bude schopný spravovať užívateľské profily. V jeho kompetencií bude upravovať role registrovaných užívateľov a odoberať právo registrovaných užívateľov na používanie systému. Tento akt bude v aplikácií nazývaný deaktivácia.

2.3.1.3 Tvorba a editácia záznamov o slimákovi

Užívatelia si budú schopní vkladať záznamy o slimákoch do aplikácie a pridávať dodatočné informácie ako meno, komentár k slimákovi, farbu ulity, farbu slimáka, vzorec ulity, dátum vyliahnutia a pôvod slimáka. Následne budú užívatelia schopní tieto informácie editovať.

2.3.1.4 Pridanie, zobrazenie a editácia udalostí na časovej ose slimáka

Každý slimák bude mať časovú os, na ktorú sa budú môcť pridávať a následne editovať textové informácie zadané užívateľom. Typy týchto udalostí budú spravované administrátorom.

2.3.1.5 Pridávanie, zobrazenie a editácia záznamov o meraní slimáka

Každý slimák bude mať priradenú tabuľku s meraniami. Tieto merania bude možné pridávať a následne editovať užívateľom.

2.3.1.6 Pridávanie, zobrazenie a editácia obrázkov slimáka

Každý slimák bude mať priradenú galériu s obrázkami. Tieto obrázky bude možné pridávať, zobrazovať a mazať užívateľom.

2.3.1.7 Tvorba a editácia záznamov o snúške

Užívatelia si budú schopní vkladať záznamy o snúškách do aplikácie a pridávať dodatočné informácie ako komentár, veľkosť snúšky, dátum znesenia, začiatok periódy liahnutia, koniec periódy liahnutia, rodič a informácia o tom, či bola snúška ponechaná. Následne budú užívatelia schopní tieto informácie editovať.

2.3.1.8 Pridanie, zobrazenie a editácia udalostí na časovej ose snúšky

Každá snúška bude mať časovú os, na ktorú sa budú môcť pridávať a následne editovať textové informácie zadané užívateľom. Typy týchto udalostí budú spravované administrátorom.

2.3.1.9 Evidovanie chovateľských boxov

Užívatelia budú schopní evidovať svoje chovateľské boxy v systéme. Pri boxoch sa bude zaznamenávať komentár, meno, výška, šírka, hĺbka, a dátum zaobstarania boxu. Užívateľ bude môcť evidovať viacej takýchto boxov.

2.3.1.10 Pridanie, zobrazenie a editácia udalostí na časovej ose boxu

Každý box bude mať časovú os, na ktorú sa budú môcť pridávať a následne editovať textové informácie zadané užívateľom. Typy týchto udalostí budú spravované administrátorom.

2.3.1.11 Tvorba a následná editácia chovných skupin

Užívatelia budú schopní si v aplikácií tvoriť chovné skupiny slimákov. Ako chovnú skupinu berieme zoskupenie slimákov rovnakého druhu, ktorí zdieľajú spoločný box. Z toho vyplýva, že každá skupina musí mať určený špecifický box a každý jednotlivý slimák musí patriť do nejakej skupiny. Skupina môže byť tvorená jedným alebo viacerými slimákmi. V jednom boxe bude možné evidovať viacero skupín. Skupiny bude možné upravovať pridávaním a odoberaním slimákov. Užívateľ bude schopný upravovať informácie skupiny.

2.3.1.12 Taxonómia jedincov

Užívatelia budú schopní si uchovávať informácie o taxonómií jednotlivých slimákov.

2.3.1.13 Tvorba a následná editácia typov udalostí administrátorom

Administrátor bude schopný tvoriť typy udalostí pre box, snúšku, skupinu a slimákov. Tieto typy udalostí bude potom schopný editovať.

2.3.1.14 Tvorba a následná editácia taxonómií

Administrátor bude schopný tvoriť taxonómie, určovať ich taxony a tým tvoriť stromovú štruktúru taxonómií dostupných užívateľom.

2.3.2 Nefunkčné požiadavky

2.3.2.1 Webová služba

Systém bude dostupný na internete z väčšiny internetových prehliadačov.

2.3.2.2 Systém dostupný na mobilných zariadeniach

Systém bude dostupný aj na smartfónoch a tabletoch s pripojením na internet.

2.4 Bussiness požiadavky

Na rozdiel od funkčných a nefunkčných požiadaviek, úlohou bussiness požiadaviek je definovať obchodné potreby projektu a kritéria jeho úspechu. Po diskusií so zadávateľom boli vymedzené nasledujúce bussiness požiadavky popisujúce ideálny stav projektu.

2.4.1 Fungujúci informačný systém

Informačný systém pre podporu chovateľov slimákov je základným pilierom a odvíja sa od neho všetka prípadná komercializácia tohoto projektu. Aby bol informačný systém úspešný, potrebuje zahŕňať všetky potreby užívateľov na uchovávanie informácií o svojich slimákoch jednoducho a na jednom mieste.

2.4.2 Administrácia systému

Pre dlhodobú udržateľnosť projektu a zdieľaných informácií v ňom je potrebná intuitívna administrácia, vykonávaná prevažne administrátorom. Je potrebné udržiavať zdieľané informácie v systéme a taktiež aj informácie o užívateľoch.

2.4.3 Pokrytie nákladov na chod systému

Komercializácia systému by mala zaručiť pokrytie nákladov na chod a údržbu.

Kapitola 3

Štúdia uskutočniteľnosti

Štúdia uskutočniteľnosti obsahuje systematicky usporiadané informácie potrebné na celkové vyhodnotenie investičného projektu. Má za úlohu zhodnotiť rôzne alternatívy, posúdiť uskutočniteľnosť daného investičného projektu a poskytnúť všetky podklady pre investičné rozhodnutia. Je nástrojom na posúdenie návrhu projektu, najmä z ekonomického a technického hľadiska. Osnova štúdie bola prispôsobená potrebám projektu. [5]

3.1 Stručný popis

V tejto štúdii sa bude rozoberať návrh na vytvorenie informačného systému pre podporu chovateľov slimákov. Zadávateľom je Mgr. Petr Matyáš, ktorý je zároveň aj vedúcim tejto bakalárskej práce.

Projekt sa snaží vyplniť dieru na trhu chovateľov slimákov. Zadávateľ prejavil svoju nespokojnosť so súčasným stavom uchovávania informácií a rozhodol sa vytvoriť informačný systém, ktorý by tento stav zlepšil. Systém by mal slúžiť chovateľom na uchovávanie informácií spojených s chovom slimákov a zjednodušiť prístup k daným informáciám. Primárnym cieľom systému nie je profit pre zadávateľa, ale schopnosť pokryť výdavky na chod systému pomocou systému samotného.

3.2 Popis súčastného stavu

Následujúci popis bol vytvorený na základe diskusie so zadávateľom. Vzhľadom na to, že v súčastnosti neexistuje spoľahlivý informačný systém na podporu chovateľov slimákov, musia chovatelia využívať nie práve najoptimálnejšie spôsoby zaznamenávania a uchovávania informácií o svojich slimákoch. Najzaužívanejšie spôsoby v súčastnosti sú zapisovanie informácií na papier, elektronický textový editor, či počítačový program Microsoft Excel. To so sebou prináša mnoho nevýhod. Pri záznamoch na papieri majú chovatelia často problém so skladovaním, úpravami a rýchlym vyhľadávaním v papieroch. Program Microsoft Excel nie je koncipovaný na všetky informácie, ktoré sú pri chovaní slimákov zaznamenávané, a preto sú niektoré informácie uchovávané neprehľadne, alebo si ich chovatelia zaznamenávajú na viacerých miestach a vzniká chaos.

3.3 Zhrnutie požiadaviek

Ná základe požiadaviek popísaných v kapitole 2, bolo rozhodnuté vytvoriť informačný systém, ktorý daným požiadavkám vyhovuje.

3.4 Prieskum trhu

Na začiatku je dôležité určiť si cieľovú skupinu. Vzhľadom na špecifickosť projektu, hlavná cieľová skupina je už existujúca komunita chovateľov slimákov. Tá sa v súčastnosti združuje hlavne na sociálnych sieťach. V týchto skupinách chovatelia primárne zdieľajú fotky svojich slimákov a odovzdávajú si rady ohľadom chovu. Je dôležité podotknúť, že hlavne facebookové skupiny často slúžia aj na obchodovanie so slimákmi, to je však v rozpore s komunitnými pravidlami. [6]

Prieskum trhu bol rozdelený do dvoch častí. Prvá časť bola zameraná na analýzu konkurencie v česko-slovenskej komunite slimákov. Druhá časť sa sústredí na samotnú komunitu chovateľov slimákov.

3.4.1 Analýza konkurencie

V súčastnosti neexistuje priama konkurencia, ktorá by poskytovala funkcionalitu ako navrhovaný projekt.

3.4.2 Komunita

Ako už bolo spomínané, chovatelia slimákov sa združujú na sociálnych sieťach. Za zmienku stoja hlavne skupiny na sociálnej sieti Facebook, kde najväčšia skupina má okolo 1800 členov. V tejto skupine bol autorom uverejnený príspevok, v ktorom sa priamo pýtal členov na ich názor na navrhovaný projekt a prípadný členský poplatok. Z odpovedí vyplynulo, že o informačný systém záujem je, avšak na členský poplatok mali členovia rozdielne názory. Z početnosti skupiny a na základe ohlasov vyplýva, že systém má veľkú šancu prilákať chovateľov.

3.4.3 Odhad užívateľskej základne

Na základe prieskumu bol vytvorený aj základný odhad používateľskej základne. Ten sa pohybuje rádovo v desiatkach až stovkách užívateľov. Treba však vziať do úvahy aj pesimistickejšie a optimistickejšie odhady. Pri pesimistickom odhade sa počíta s veľkosťou základne rádovo v jednotkách až nižších desiatkach užívateľov. Pri optimistickejšom odhade je možné, že táto základňa sa rozšíri na nižšie tisícky užívateľov, ale v súčasnosti sa takýto odhad javí ako nepravdepodobný.

3.5 Situačná analýza

Situačná analýza je komplexná analýza, ktorá zachytáva vonkajšie a vnútorné faktory ovplyvňujúce súčasnú a budúcu situáciu. Najznámejším nástrojom na analyzovanie situácie je SWOT analýza [7]. SWOT analýza sa sústredí na silné stránky (Strenghts), slabé stránky (Weaknesses), príležitosti (Oportunities) a hrozby (Threats). Analýza silných a slabých stránok sa sústreďuje skôr na vnútorné prostredie projektu, zatiaľ čo príležitosti a hrozby sa sústredia skorej na vonkajšie prostredie projektu.

3.5.1 SWOT analýza

Silné stránky:

- V súčastnosti neexistuje produkt s podobným zameraním.
- Zadávateľ má skúsenosti v danom odbore.
- Systém výrazne zjednoduší súčasné procesy uchovávania informácií.
- Náklady na implementáciu projektu sú nulové.

Slabé stránky:

- Komunita cieľových užívateľov je pomerne malá.

Príležitosti:

- Systém poskytuje priestor na prípadné rozšírenia.
- Systém je možné propagovať na sociálnych sieťach.
- Existuje možnosť prípadného nárastu popularity chovu slimákov v budúcnosti.
- Systém má potenciál generovať finančný zisk.

Hrozby:

- V budúcnosti môže vzniknúť nová konkurencia.
- Užívatelia nemusia mať záujem o systém.
- Systém nemusí byť schopný pokryť finančné nároky na jeho chod.

3.6 Technické a technologické riešenie projektu

Podrobný popis architektúry systému a využitých technológií je popísaný v sekcii 4.

Vzhľadom na to, že sa jedná o webovú aplikáciu, treba spomenúť niekoľko dôležitých výhod a nevýhod. Veľká časť práce prebieha na strane serveru, takže odpadá nárok na výkon užívateľských zariadení. Dá sa povedať, že pokiaľ užívateľ vlastní zariadenia s internetovým prehliadačom a internetovým pripojením, nebude mať problém s prístupom a využívaním systému. Z toho vyplýva aj druhá výhoda. Keďže užívateľovi stačí na používaniu systému iba internetový prehliadač a zariadenie s prístupom na internet, môže využívať funkcionality systému odvšadiaľ. Ale takéto riešenie so sebou prináša aj jednu nevýhodu. Tou sú nároky na serverovú stranu. Vo všeobecnosti existujú nasledujúce možnosti: vytvorenie vlastnej infraštruktúry alebo využitie cloud hostingu.

Pred výberom vhodnej možnosti je potrebné vziať do úvahy veľkosť potenciálnej užívateľskej základne. Tá sa pri najoptimistickejších odhadoch pohybuje rádovo v nižších tisícoch. Zároveň, keďže sa nejedná o kritickú aplikáciu, nároky na rýchlosť nie sú až také vysoké.

3.6.1 Požiadavky na server

Server by mal byť zo začiatku schopný obslúžiť naraz aspoň 10 užívateľov a uchovávať v databáze údaje aspoň o 300 užívateľoch. Na splnenie týchto požiadaviek by mal server pozostávať aspoň z 3 GB pamäte RAM, 50 GB pamäte na SSD diskoch a aspoň dvoch virtuálnych procesorových jednotiek (vCPU). Zároveň bude dôležitou výhodou škálovateľnosť na základe potrieb zadávateľa.

3.6.2 Výber serveru

V prípade zadaného projektu sa ako najlepšia možnosť javí cloud hosting. Cloud hosting poskytuje najlepšiu škálovateľnosť a s tým spojenú aj cenu, pretože nevzniká riziko platenia za nevyužitý priestor.

Po zhodnotení dostupných možností bolo zvolené cloudové riešenie od spoločnosti DigitalOcean. Toto riešenie disponuje 4 GB pamäte RAM, 80 GB pamäte na SSD a dvomi virtuálnymi procesorovými jednotkami. Samozrejme, v prípade potreby je možné prejsť na niektorý z ďalších ponúkaných plánov.

3.7 Údržba systému

Údržbu systému bude mať na svedomí administrátor, ktorý ma v rámci systému práva na administráciu užívateľov a zdieľaných informácií. Vzhľadom na veľkosť projektu nie je v pláne využívať žiadny spôsob platenej externej údržby a podpory. Užívateľské otázky a problémy budú komunikované priamo s administrátorom a v prípade chýb v implementácií sa autori oboch častí systému, Mgr. Mikolas Teska a Jakub Rigoci, ponúkli tieto chyby vyriešiť. Avšak vzhľadom na to, že autori nebudú priamo administrátori, bude odozva na takéto chyby otázkou dní až týždňov. Keďže nejde o kritickú aplikáciu, rýchlosť odozvy nemá až takú vysokú prioritu. Tak isto plánovaný administrátor má dobré znalosti o problematike a mal by byť schopný vyriešiť väčšinu problémov.

3.8 Financovanie chodu systému

Z požiadaviek vyplýva, že systém by mal byť schopný si na seba zarobiť. To znamená, že financie potrebné na chod systému sa navrátia vo forme zisku. So zadávateľom bola diskutovaná hlavne možnosť mesačného predplatného.

Najprv treba určiť finančné nároky na tvorbu a chod systému. S ohľadom na to, že zdrojový kód oboch častí systému vznikol ako bakalárska práca autorov Mgr. Mikolasa Tesku a Jakuba Rigociho, nároky na tvorbu systému sú nulové. Nároky na chod systému sa dajú zredukovať na cenu hostingu, pretože údržbu bude poskytovať administrátor spolu s autormi bez nároku na odmenu.

Cena zvoleného serverového riešenia 3.6.2 je 440 Kč na mesiac. Pri väčšom alebo menšom záujme o aplikáciu existuje možnosť zmeny plánu, a s tým spojená aj zmena ceny. Vzhľadom na to ale, že cena sa mení priamo úmerne s počtom aktívnych užívateľov, je možné určiť jednotnú cenu predplatného, ktorá pokryje aj prechod na takéto rozdielne plány.

3.8.1 Mesačné predplatné

Pri odhade 20 aktívnych užívateľov na začiatku by cena predplatného musela byť 22 Kč/mes. Táto čiastka nie je nereálna, ale je veľmi pravdepodobné, že samotná existencia nutnosti predplatného

odláka značné percento potenciálnych užívateľov. Z tohto dôvodu, ak by sa rozhodlo pre mesačné predplatné, je odporúčané nastaviť cenu predplatného na 30 Kč/mes na poskytnutie istej rezervy v rozpočte pre nečakané udalosti. Pre užívateľov, ktorí sú ochotní platiť predplatné, samotná zmena výšky predplatného o 8 Kč/mes nebude predstavovať veľkú prekážku pri používaní systému.

3.9 Riziká

Pri tvorbe projektu je dôležité myslieť aj na potenciálne riziká. Každé riziko má určený dopad, závažnosť, plán na mitigáciu rizika a prípadný krízový plán.

3.9.1 Neprijatie systému užívateľmi

Aby bol systém úspešný, je potreba užívateľskej základne, ktorá bude systém využívať. Pri neprijatí systému užívateľmi táto užívateľská základňa existovať nebude.

Závažnosť: Kritická

Dopad: Systém nebude schopný na seba zarobiť

Plán na mitigáciu: Vývoj aplikácie na základe požiadaviek zadávateľa, dôraz na jednoduchosť užívateľského rozhrania

Krízový plán: Upraviť systém podla poznatkov užívateľov

3.9.2 Problémy pri vývoji systému

Vzhľadom na to, že systém bude rozdelený na back-end a front-end, a vyvíjaný dvomi autormi v rozličných časoch, vzniká riziko problémov pri vývoji.

Závažnosť: Stredná

Dopad: Oneskorenie dodania systému

Plán na mitigáciu: Pravidelná komunikácia medzi autormi, presné určenie požiadaviek na začiatku vývoja

Krízový plán: Úprava časti systému jedným z autorov

3.9.3 Skúsenosti

Autor tejto práce a vývojár časti systému nemá žiadne praktické skúsenosti s tvorbou danej časti. Preto vzniká riziko nesprávneho prístupu k implementácií a jej následnej nefunkčnosti alebo zbytočnej komplikovanosti.

Závažnosť: Stredná

Dopad: Nečitateľnosť zdrojového kódu, nedostatočná rýchlosť implementovaného riešenia až jeho nefunkčnosť

Plán na mitigáciu: Naštudovanie danej problematiky autorom, konzultovanie riešenia so skúseným vývojárom

Krízový plán: Konzultácia so skúseným vývojárom

3.10 Záverečné hodnotenie projektu

Vzhľadom na súčasnú neexistenciu konkurenčného riešenia, ktoré by sa zaoberalo podporou chovu slimákov rovnakým spôsobom ako navrhovaný projekt, exituje veľká šanca na úspech. Tú podtrhávajú aj ostatné pozitíva navrhovaného projektu. Finančne výhodná implementácia, nenáročná podpora systému a malé finančné nároky na jeho chod.

Samozrejme, riziko neuchytenia sa systému netreba brať na ľahkú váhu. Jeho dôvodom môže byť hlavne pomerne malá komunita chovateľov.

Po zvážení faktov má tento projekt potenciál byť jednou z dôležitých pomôcok pri chove slimákov, a preto je jeho tvorba a dlhodobá podpora odporúčaná.

Kapitola 4
Návrh

Pri webových aplikáciach môžeme systém rozdeliť na dve časti:

1. **Program na strane klienta:** Tento program beží v prehliadači užívateľa a reaguje na jeho vstupy. Tiež nazývaný front-end.
2. **Program na strane serveru:** Tento program beží na vzdialenom počítači nazývanom tiež server. Reaguje na požiadavky klienta, ktoré sú väčšinou zasielané pomocou protokolu HTTP (HyperText Transfer Protocol). [8]

4.1 Program na strane klienta

Program na strane klienta (front-end) bol vytvorený ako súčasť tejto bakalárskej práce. Front-end je časť, ktorá je priamo viditeľná užívateľom. Preto je tu kladený veľký dôraz na prehľadnosť, intuitívnosť a vzhľad. Súčasťou je aj celková grafická podoba, ktorá by mala užívateľa zaujať a podporovať intuitívnosť a prehľadnosť.

Základom tvorby front-endu je trio jazykov: HTML (HyperText Markup Language), CSS (Cascading Style Sheets) a JavaScript. HTML definuje význam a štruktúru obsahu. CSS je jazyk používaný na popis prezentácie obsahu napísaného pomocou HTML. JavaScript je programovací jazyk, ktorý sa používa na pokročilejšiu úpravu chovania obsahu. V súčasnosti je populárne využitie rôznych JavaScriptových knižníc a frameworkov.

4.1.1 Voľba architektúry

Vo všeobecnosti existujú dva implementačné prístupy k tvorbe front-endovej časti webových aplikácií. Single Page Application (SPA) alebo Multi Page Application (MPA)[9].

4.1.1.1 Multi Page Application

Viac stránkové aplikácie fungujú tradičnejším spôsobom. Každá požiadavka užívateľa je odoslaná na server, ten ho spracuje a následne vygeneruje na základe požiadavky kód na zobrazenie stránky. Tento kód sa odošle späť na stranu klienta, kde je zobrazený. Z tohoto princípu vyplýva, že každá zmena na stránke musí prejsť cez server. V súčastnosti je ale tento prístup optimalizovaný použitím AJAX (Asynchronous JavaScript And XML). AJAX [10] je sada technológií, umožňujúcich

asynchrónnu aktualizáciu údajov na stránke. Vďaka AJAXu je možné aktualizovať iba určitú časť informácií bez potreby aktualizovania celej stránky.

4.1.1.2 Single Page Application

Pri jednostránkových aplikáciach tvorí celé užívateľské prostredie jedna stránka, ktorá sa mení na základe požiadaviek a akcií užívateľa. Server sa teda nestará o generovanie kódu na zobrazenie stránok, ale posiela klientovi iba dáta, o ktoré si požiada. Za generovanie je teda zodpovedný klient. Na generovanie je vo väčšine prípadov používaný už spomínaný JavaScript, alebo niektorý z rozsiahleho množstva frameworkov, ktoré sú na tomto jazyku založené. Tento prístup je podstatne rýchlejší, pretože zo servera sa odosielajú iba potrebné dáta, a taktiež zlepšuje užívateľský pôžitok, pretože všetky dáta sú získavané na pozadí. Na základe vyššie spomenutých výhod bol tento prístup zvolený aj pri tejto bakalárskej práci.

4.1.2 Framework a JavaScript

4.1.2.1 Webový framework

Webový framework je softwareová štruktúra, ktorá uľahčuje písanie, škálovanie a údržbu webových aplikácií. Poskytuje množstvo predripravených funkcionalít na zjednodušenie použitia prvkov dôležitých pri vývoji ako je routovanie (smerovanie), prepojenie s databázou, autentizáciou, bezpečnosťou a mnoho ďalších [11]. Zásadný rozdiel medzi frameworkom a knižnicou je, že zatiaľ čo autorský kód volá časti knižnice, framework volá autorský kód. Tento princíp sa nazýva inverzia kontroly (inversion of control) [12]. Webových frameworkov existuje mnoho, či už zameraných na front-end, alebo back-end. Frameworky určené pre front-end využívajú z veľkej časti už spomínaný JavaScript.

4.1.2.2 JavaScript

JavaScript [13] je vysokoúrovňový, dynamický, netypovaný, interpretovaný jazyk so širokou škálou využitia [14]. Používa sa či už na strane serveru,[1] na strane klienta alebo aj v desktopových aplikáciach[2]. Je to jeden z najrozšírenejších programovacích jazykov a interpret tohoto jazyka sa nachádza v prevažnej väčšine bežne používaných počítačov a smartfónov.

JavaScript je vytvorený na základe štandartu ECMAScript Language Specification [15] a ECMAScript Internationalization API specification [16].

4.1.2.3 Virtual DOM

Vzhľadom na to, že množstvo frameworkov a knižníc určených pre front-end používa koncept virtual DOMu, je potrebné tento koncept vysvetliť. DOM (Document object model) je dátová reprezentácia objektov, ktoré tvoria štruktúru a obsah dokumentu na webe. DOM je reprezentovaný ako stromová dátová štruktúra. Vďaka tomu sú zmeny a aktualizácie DOM rýchle. Po zmene sa však aktualizovaný prvok a jeho potomci musia znova vykresliť, aby sa aktualizovalo používateľské rozhranie aplikácie. Opätovné vykreslenie alebo prekreslenie používateľského rozhrania je to, čo ho spomaľuje.

Virtual DOM je pamäťová reprezentácia skutočného DOMu. Po pridaní nových prvkov do používateľského rozhrania sa vytvorí virtuálny DOM, ktorý je reprezentovaný ako strom. Každý prvok je uzlom v tomto strome. Ak sa stav ktoréhokoľvek z týchto prvkov zmení, vytvorí sa nový virtuálny

[1] `https://nodejs.org/en/`
[2] `https://www.electronjs.org/`

strom DOM. Tento strom sa potom porovná s predchádzajúcim virtuálnym stromom DOM. Na základe odlišností, virtuálny DOM vypočíta najlepšiu možnú metódu na vykonanie týchto zmien v skutočnom DOMe. To zaisťuje, že na skutočnom DOMe sa vykonáva minimum operácií. Tým sa znížia náklady na výkon aktualizácie skutočného modelu DOM [17].

4.1.3 Porovnanie JavaScriptových frameworkov a knižníc určených pre front-end

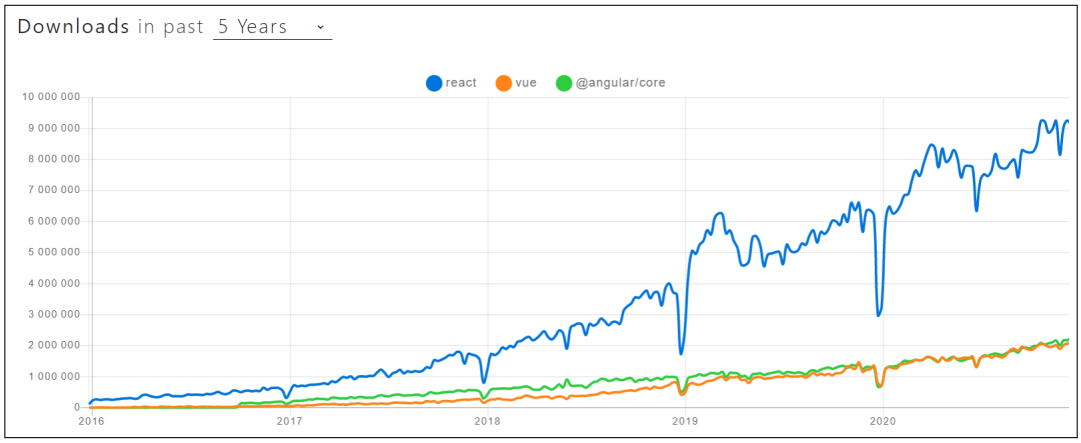
Na základe [18] je trojica v súčasnosti najpopulárnejších frameworkov a knižníc ReactJS, VueJs a AngularJS.

4.1.3.1 ReactJs

ReactJs [19] je bezpochyby v súčastnosti najpopulárnejšou technológiou na tvorbu užívateľských rozhraní 4.1. Bol vytvorený v roku 2013 spoločnosťou Meta. Je zároveň využívaný v ich produktoch ako sú Facebook, Instagram alebo WhatsApp.

Na rozdiel od nižšie opisovaných technológií, ReactJs nie je framework ale knižnica. Je založený na princípe komponentov, ktoré sú znovupoužitelné, majú svoju vnútornú logiku a uchovávajú si svoj stav. Tento stav je dôležitý pri zmenách. Pri zmene vnútorného stavu komponentu ReactJs tento komponent znovu vykreslí.

ReactJs je vhodný na tvorbu single-page aplikácií, keďže umožňuje meniť prvky viditeľné užívateľom bez potreby obnovenia stránky. Je taktiež ľahko integrovatelný z inými technológiami ako napríklad AngularJs. ReactJs využíva koncept Virtual DOM.



Obr. 4.1 Graf počtu stiahnutí pomocou npm. Modrá - ReactJs, Oranžová - VueJs, Zelená - AngularJs

4.1.3.2 AngularJs

AngularJs [20] bol vytvorený spoločnosťou Google v roku 2010, čo z neho robí najstarší z opisovaných frameworkov. V roku 2016 bol však vydaný Angular 2. Táto verzia a všetky následujúce tak upustili z koncovky „Js“ pôvodného názvu.

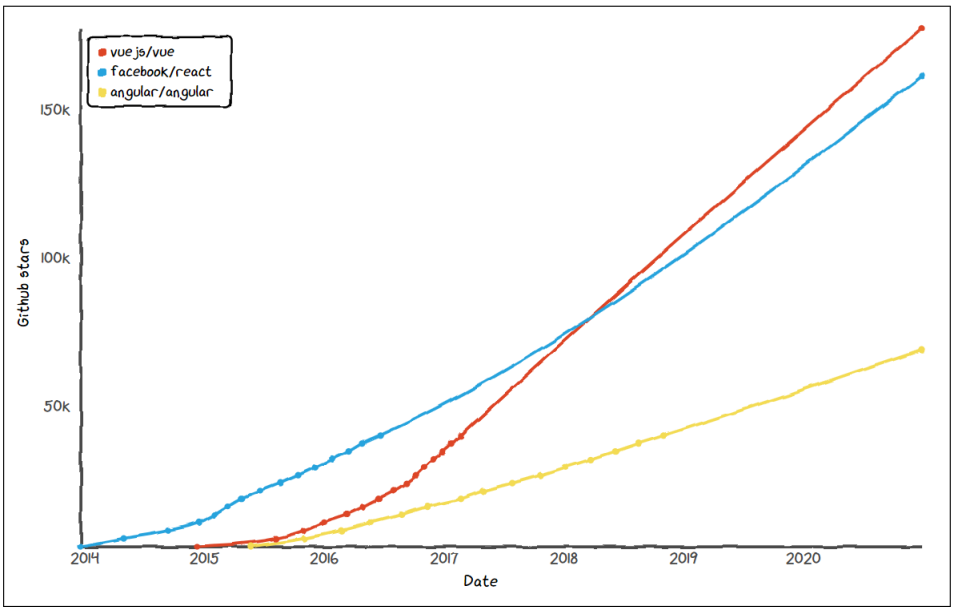
AngularJs je štrukturálny framework pre dynamické webové aplikácie. Používa takzvaný data binding (princíp dátovej väzby). Vďaka tomu dokáže automaticky aktualizovať užívateľské rozhranie

vždy, keď nastane nová zmena. Týmto podstatne zjednodušuje prácu a eliminuje veľké časti kódu, ktoré by museli byť napísané vývojárom. Umožňuje používanie takzvaných komponentov, znovupoužiteľných častí kódu. Vzhľadom na obrovskú komplexnosť tohoto frameworku, je najzložitejší na naučenie sa.

4.1.3.3 VueJs

VueJs [21] bol vytvorený bývalým pracovníkom spoločnosti Google Evanom You v roku 2014. V posledných rokoch sa tento framework teší nárastu popularity, a to aj napriek tomu, že nie je vytvorený veľkou spoločnosťou ako ostatné opisované frameworky.

VueJs má v súčastnosti najviac hviezd na GitHube v porovnaní s ostatnými frameworkmi 4.2. Aj keď táto metrika nie je veľmi objektívna [3], potvrdzuje jeho stúpajúcu popularitu.



Obr. 4.2 Graf počtu hviezd na GitHube. Modrá - ReactJs, Červená - VueJs, Žltá - AngularJs

Pri tvorbe sa autor Evan You snažil zobrať z každého front-endového frameworku to najlepšie a spojiť to do jedného celku. Ako ReactJs, VueJs používa komponenty a virtual DOM, od AngularJs si VueJs prevzalo myšlienku obojsmerného posielania dát medzi komponentami. VueJs má výborne spracovanú dokumentáciu, pevne stanovenú štruktúru komponentu, a tým pádom sa javí ako skvelá voľba pre začínajúcich vývojárov. VueJs má takisto širokú ponuku doplnkových knižníc ako Vuex alebo Vue Router. Jednotlivé komponenty používajú koncept „Single File Components“. To znamená, že celá štruktúra komponentu vrátane logiky a vzhľadu je definovaná v jedinom súbore s koncovkou *.vue*. Takýto súbor je rozdelený na tri časti:

1. **Template:** V tejto časti sa definuje štruktúra komponentu pomocou HTML obohateného o používanie premenných alebo pomocných *directives*. Tie dodávajú HTML elementom možnosti ako používať podmienky, cykly, alebo viazať do elementu hodnoty premennej.

[3] `https://twitter.com/sandofsky/status/1007727882095886336`

2. **Script:** V tejto časti sa definuje vnútorná logika komponentu. Každý komponent má dostupnú sadu metód, ktoré ho môžu upravovať v rôznych častiach jeho životného cyklu. Tieto metódy sa nazývajú *lifecycle hooks*.

3. **Style:** Tu sa definuje vzhľad komponentu pomocou CSS.

Po prevedenom porovnaní a následnou konzultáciou s vedúcim tejto práce, bol tento framework zvolený na implementáciu systému.

4.2 Program na strane serveru

Program na strane serveru (back-end) bol vytvorený spoluautorom Mgr. Mikolasom Teskou ako súčasť jeho bakalárskej práce „Informační systém pro podporu chovu šneků“ [1]. V následujúcom odstavci bude bližšie ukázaná architektúra programu na strane serveru.

4.2.1 Databáza

Databáza je organizovaná kolekcia štruktúrovaných informácií uložená elektronicky v počítačovom systéme. Slúži na uchovávanie dát aplikácie. Databáza je zvyčajne riadená systémom riadenia bázy údajov (DBMS). Spoločne sa dáta a DBMS spolu s aplikáciami, ktoré sú s nimi spojené, označujú ako databázový systém, často skrátený len na databázu [22].

Databázy môžeme rozdeliť na dve skupiny: NoSQL databázy a relačné databázy.

1. **NoSQL:** NoSQL databázy sú netabuľkové databázy a ukladajú údaje inak ako tradičné relačné databázy. Sú ľahko škálovateľné pri veľkom množstve informácií a pri veľkom nápore užívateľov. Preto sa tieto databázy často využívajú pri práci s veľkými dátami [23].

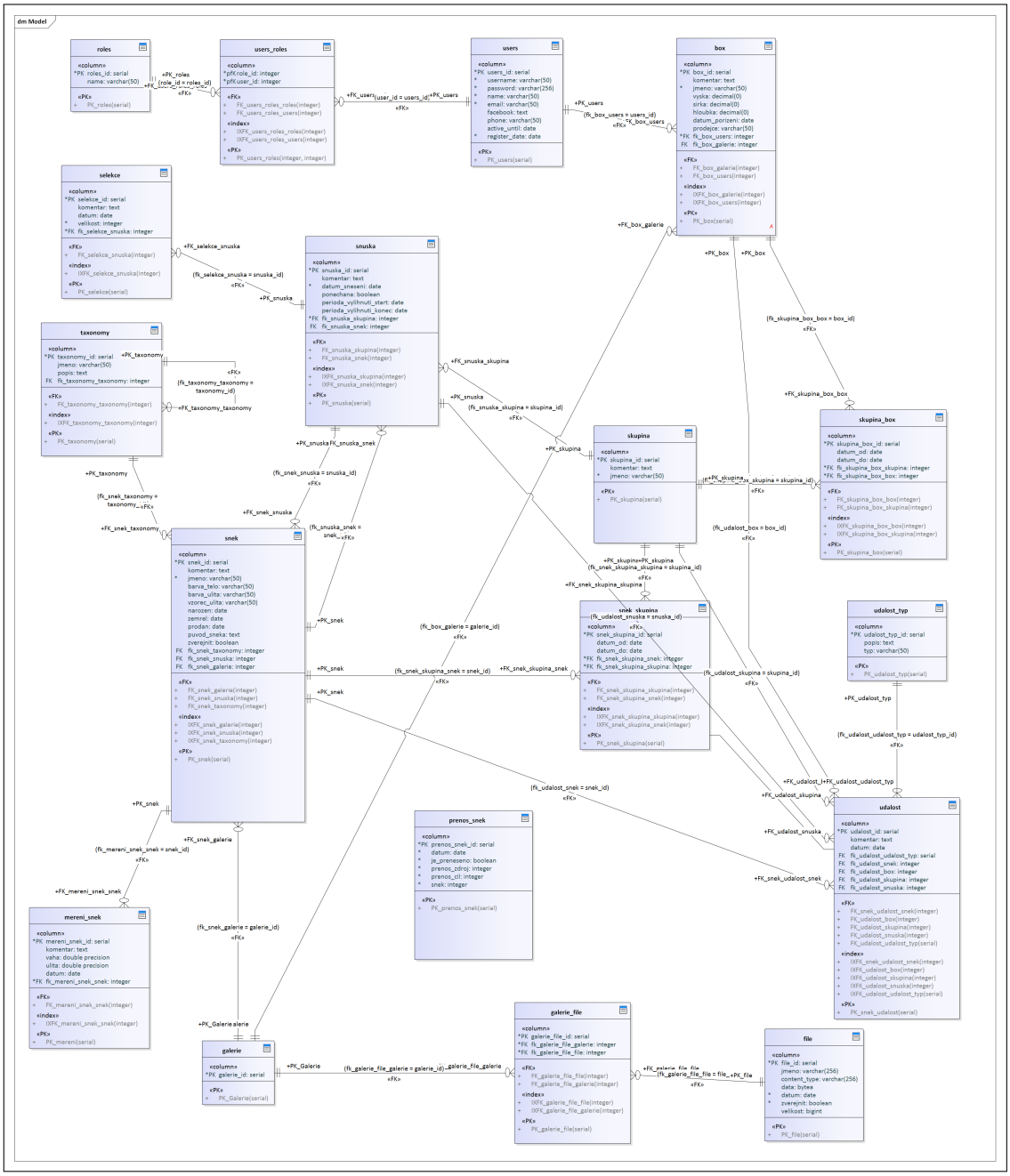
2. **Relačné databázy:** Relačná databáza je typ databázy, ktorá ukladá a poskytuje prístup k údajom, ktoré spolu súvisia. Relačné databázy sú založené na relačnom modeli, intuitívnom a priamom spôsobe reprezentácie údajov v tabuľkách. V relačnej databáze je každý riadok v tabuľke záznamom s jedinečným ID nazývaným kľúč. Stĺpce tabuľky obsahujú atribúty údajov a každý záznam má zvyčajne hodnotu pre každý atribút, čo uľahčuje stanovenie vzťahov medzi údajmi [24].

Pre túto aplikáciu bol zvolený SQL systém PostgreSQL.[4] Tento systém bol zvolený pre jeho vysoký výkon, dostupnosť a pre jeho podporu v rámci licencie open source [25].

4.2.2 Dátový model

Dátový model je dôležitou súčasťou každej aplikácie, ktorá používa databázu na uchovávanie informácií. Používa sa na popísanie štruktúry a vzťahov dát v databáze. Dátový model pre túto aplikáciu bol vypracovaný spoluautorom Mgr, Mikolasom Teskou. Ilustrácia tohoto modelu sa nachádza na obrázku 4.3.

[4] `https://www.postgresql.org/`



Obr. 4.3 Dátový model

4.2.3 REST API

Komunikácia medzi front-endom a back-endom bude zabezpečovaná pomocou REST API. REST API, alebo tiež RESTful API je aplikačné programové rozhranie (API), ktoré vyhovuje obmedze-

niam architektonického štýlu REST a umožňuje interakciu s webovými službami RESTful. REST znamená reprezentačný prenos stavu a vytvoril ho počítačový vedec Roy Fielding [26]. REST API štandardne používa http metódy POST, PUT, GET a DELETE. Dáta sa prenášajú pomocou JSON [27] formátu.

4.2.4 Mapovanie metód REST API

Mapovanie je rozdelené do blokov podla funkcie. Všetky url majú spoločný začiatok /api/v1. Mapovanie bolo navrhnuté a zrealizované spoluautorom Mgr. Mikolasom Teskou. Po prvotnom návrhu boli pridané dve rozširujúce metódy. Tieto metódy boli pridané po odovzdaní práce Mgr. Mikolasa Tesku, a preto sa v jeho návrhu nenachádzajú. Ostatok mapovania ostal nezmenený a je prebratý z [1].

4.2.4.1 Rozširujúce metódy

GET	/{boxId}/skupina	Vráti skupiny, ktoré patria k danému boxu
GET	/skupina/{skupinaId}	Vráti slimákov, ktorí patria do danej skupiny

4.2.4.2 admin

URL pro zobrazení všech boxů, šneků, snůšek nebo skupin

GET	/admin/snek	Výpis všech šneků v aplikaci
GET	/admin/box	Výpis všech boxů v aplikaci
GET	/admin/skupina	Výpis všech skupin v aplikaci
GET	/admin/snuska	Výpis všech snůšek v aplikaci

4.2.4.3 auth

URL pro přihlášení a registraci do aplikace

POST	/auth/signup	Registrace uživatele do aplikace
POST	/auth/signin	Přihlášení uživatele do aplikace

4.2.4.4 snek

URL pro operace se šneky a jejich atributy

GET	/snek	Vrátí pole šneků
POST	/snek/{id}	Přidaní šneka do aplikace
GET	/snek/{id}	Vrátí objekt šnek
PUT	/snek/{id}	Upraví šneka
DELETE	/snek/{id}	Smaže šneka
GET	/snek/{snekId}/skupina	Vrátí skupinu ve které se šnek nachází
GET	/snek/{snekId}/snuska	Vrátí seznam snůšek, kterým je šnek matkou
POST	/snek/{snekId}/skupina/{skupinaId}	Změní skupinu šneka
GET	/snek/{snekId}/udalost	Vrátí pole událostí patřících ke šnekovi
GET	/snek/{snekId}/rodokmen	Vrátí rodokmen pouze matek pro vybraného šneka
GET	/snek/{snekId}/mereni	Vrátí pole měření patřících ke šnekovi
POST	/snek/{snekId}/image	Přidaní obrázků šneka do aplikace
POST	/snek/{snekId}/inactivate	Inaktivace šneka. Po níž nenáleží do žádné skupiny

4.2.4.5 box

URL pro operace s chovnými boxy

POST	/box	Přidání boxu do aplikace
GET	/box	Vrátí pole boxu patřících uživateli
GET	/box/{boxId}	Vrátí specificky box
PUT	/box/{boxId}	Upraví box
DELETE	/box/{boxId}	Smaže specifický box
GET	/box/skupina/{skupinaId}	Vrátí specificky box ve kterém je skupina
GET	/box/{boxId}/udalost	Vrátí pole událostí patřících k boxu
POST	/box/{boxId}/image	Přidání obrázku boxu do aplikace

4.2.4.6 skupina

URL pro operace se skupinami

GET	/skupina	Vrátí pole skupin příslušného uživatele
GET	/skupina/{id}	Vrátí skupinu
POST	/skupina/{id}	Přidaní skupiny do aplikace
PUT	/skupina/{id}	Upraví skupinu
DELETE	/skupina/{id}	Smaže specifickou skupinu
GET	/skupina/{skupinaId}/udalost	Vrátí pole událostí patřících ke skupině
POST	/skupina/{skupinaId}/inactivate	Inaktivuje skupinu vyřazením z boxu
POST	/skupina/{skupinaId}/box/{boxId}	Změna boxu u skupiny

4.2.4.7 snuska

URL pro operace se snůškami

GET	/snuska	Vrátí pole snůšek
GET	/snuska/{id}	Vrátí snůšku
POST	/snuska/{id}	Přidání snůšky do aplikace
PUT	/snuska/{id}	Upraví snůšku
DELETE	/snuska/{id}	Smaže specifickou snůšku
GET	/snuska/{snuskaId}/snek	Vrátí pole šneků kteří vzešli ze snůšky
GET	/snuska/{snuskaId}/udalost	Vrátí pole událostí patřících ke snůšce
GET	/snuska/{snuskaId}/selekce	Vrátí pole selekci patřících ke snůšce
POST	/snuska/{snuskaId}/selekce	Přidaní selekce ke snůšce v aplikaci
GET	/snuska/selekce/{id}	Vrátí selekci
PUT	/snuska/selekce/{id}	Upraví selekci
DELETE	/snuska/selekce/{id}	Smaže selekci

4.2.4.8 mereni

URL pro operace s měřeními šneků

GET	/mereni/{id}	Vrátí měření
POST	/mereni/{id}	Přidání měření šneka do aplikace
PUT	/mereni/{id}	Upraví měření
DELETE	/mereni/{id}	Smaže měření

4.2.4.9 prenos

URL pro operace s přenášením šneků mezi uživateli

POST	/prenos	Přidaní přenos záznamu do aplikace
GET	/prenos/nabidka	Vrátí seznam šneků které uživatel nabízí
GET	/prenos/prijem	Vrátí seznam šneků které je možné přijmout
GET	/prenos/{prenosId}	Vrátí přenos
PUT	/prenos/{prenosId}	Upraví přenos
DELETE	/prenos/{prenosId}	Smaže přenos
POST	/prenos/{prenosId}/skupina/{skupinaId}	Příjme přenos a šneka umístí do skupiny

4.2.4.10 udalost

URL pro operace s událostmi

POST	/udalost	Přidání události
GET	/udalost/{udalostId}	Vrátí událost
PUT	/udalost/{udalostId}	Upraví událost
DELETE	/udalost/{udalostId}	Smaže událost

4.2.4.11 udalost-typ

URL pro operace s typy událostí

GET	/udalost-typ	Vrátí seznam entit udalostTyp
GET	/udalost-typ/{udalostTypId}	Vrátí udalostTyp
POST	/udalost-typ/admin	Přidání udalostTyp
PUT	/udalost-typ/admin/{udalostTypId}	Upraví udalostTyp
DELETE	/udalost-typ/admin/{udalostTypId}	Smaže udalostTyp

4.2.4.12 taxonomy

URL pro operace s taxonomickým členěním

GET	/taxonomy	Vrátí pole taxonomy záznamu
GET	/taxonomy/{taxonomyId}	Vrátí taxonomy záznam
POST	/taxonomy/admin	Přidání taxonomy záznamu do aplikace
PUT	/taxonomy/admin/{taxonomyId}	Upraví taxonomy záznam
DELETE	/taxonomy/admin/{taxonomyId}	Smaže specificky záznam taxonomy

4.2.4.13 file

URL pro operace s upload soubory

GET	/file	Vrátí všechny soubory patřící uživateli
GET	/file/{fileId}	Vrátí soubor
DELETE	/file/{fileId}	Smaže soubor
GET	/file/galerie/{galerieId}	Vrátí soubory v příslušné galerii

4.2.4.14 users

URL pro operace s uživateli

GET	/users/{userId}	Vrátí záznam o uživateli
PUT	/users/{userId}	Upraví záznam o uživateli
GET	/users/{userId}/roles	Vrátí uživatelské role
POST	/users/{userId}/changePassword	Změní heslo specifického uživatele
GET	/users/admin	Vrátí pole záznamu o uživatelích
PUT	/users/admin/{id}/roles	Změní role pro specificky záznam o uživateli
PATCH	/users/admin/{id}/scramble	Přepíše slovem ”REMOVED”údaje uživatele
GET	/users/admin/{id}/substitute	Generuje token jako substituci za vybraného uživatele
PATCH	/users/admin/{id}/activeUntil/{date}	Nastaví datum do kdy je uživatel aktivní

4.2.4.15 export

URL pro export dat

4.3 Návrh užívateľského rozhrania

Pri tvorbe programu na strane klienta je dôležitý aj návrh užívateľského rozhrania. K tomu slúžia modely ako wireframe, mockup alebo prototyp aplikácie.

4.3.1 Wireframe

Wireframe alebo drôtený model je spôsob zobrazenia dizajnu. Je to grafické znázornenie aplikácie alebo webovej stránky obsahujúcej najdôležitejšie prvky a obsah [28]. Slúži na rýchle a lacné rozvrhnutie základných prvkov a požiadavkov v užívateľskom rozhraní. Wireframe neslúži ako grafický návrh aplikácie, neobsahuje obrázky a farba sa používa zriedkavo, väčšinou iba na odlíšenie jednotlivých prvkov.

4.3.2 Mockup

Mockup je v podstate wireframe so zameraním na vzhľad užívateľského rozhrania. Pri tvorbe mockupu sa používajú farby a typografia. Cieľom tohoto modelu je priblížiť finálnu vizuálnu podobu.

4.3.3 Prototyp

Prototyp je najbližšia reprezentácia finálneho produktu. Obsahuje potrebné grafické prvky a poskytuje možnosť interakcie. Jedná sa o ideálny spôsob predvedenia celej aplikácie, je ale aj z časového a finančného hľadiska najnáročnejší na tvorbu.

Pri tvorbe modelu užívateľského rozhrania bol použitý online nástroj Figma,[5] ktorý ponúka širokú škálu možností na prípravu modelu. Tento nástroj tiež podporuje spoluprácu viacerých ľudí na jednom projekte, čo bolo pre potreby tejto práce ideálne, pretože tvorba modelu prebiehala spoločne s vedúcim bakalárskej práce. Bol vytvorený wireframe najdôležitejších prvkov aplikácie. V nasledujúcich odstavcoch budú tieto prvky ukázané.

4.3.4 Návrh okien

Pri návrhu boli s vedúcim tejto bakalárskej práce diskutované aj požiadavky na užívateľské rozhranie. Na základe diskusie vysvitlo, že hlavná požiadavka bola hlavne na finálnu intuitívnosť. Preto mal autor pri návrhu užívateľského rozhrania voľnú ruku.

4.3.4.1 Menu

Menu slúži na orientáciu v aplikácií. Bude sa dať rozkliknúť pomocou tlačidla v hornej časti obrazovky. Položky menu budú rozdelené na dve skupiny. Užívateľské a administrátorské. K administrátorským položkám bude mať prístup iba užívateľ s priradenou rolou administrátora.

4.3.4.2 Návrh okna zoznam boxov

Toto okno uvidí užívateľ po prihlásení do aplikácie. Obsahuje zoznam jeho boxov, tlačidlo na pridanie boxu, hlavičku s možnosťou odhlásenia, rozklikávacím menu na orientáciu v aplikácií a pätičku s informáciami o počtoch.

4.3.4.3 Okno detail boxu

Toto okno uvidí užívateľ pri zobrazení detailu boxu. Okno obsahuje informácie o boxe, zoznam skupín a udalosti boxu. Po rozkliknutí skupiny sa zobrazí zoznam slimákov, ktorí k danej skupine patria. Keďže skupiny nemajú vlastné dedikované okno, v hlavičke záznamu skupiny budú tlačítka na jej úpravu. Po kliknutí na slimáka sa v rovnakom okne otvorí záznam daného slimáka.

4.3.4.4 Návrh okna záznam slimáka

Záznam slimáka bude obsahovať tabuľku s meraniami, údaje o slimákovi, galériu, kde bude možné zobrazovať fotky a udalosti patriace k danému slimákovi.

4.3.4.5 Návrh okna snúšiek

Okno snúšiek bude mať rovnakú štruktúru ako okno boxov.

[5] https://www.figma.com/

4.3.4.6 Návrh okna detailu snúšky

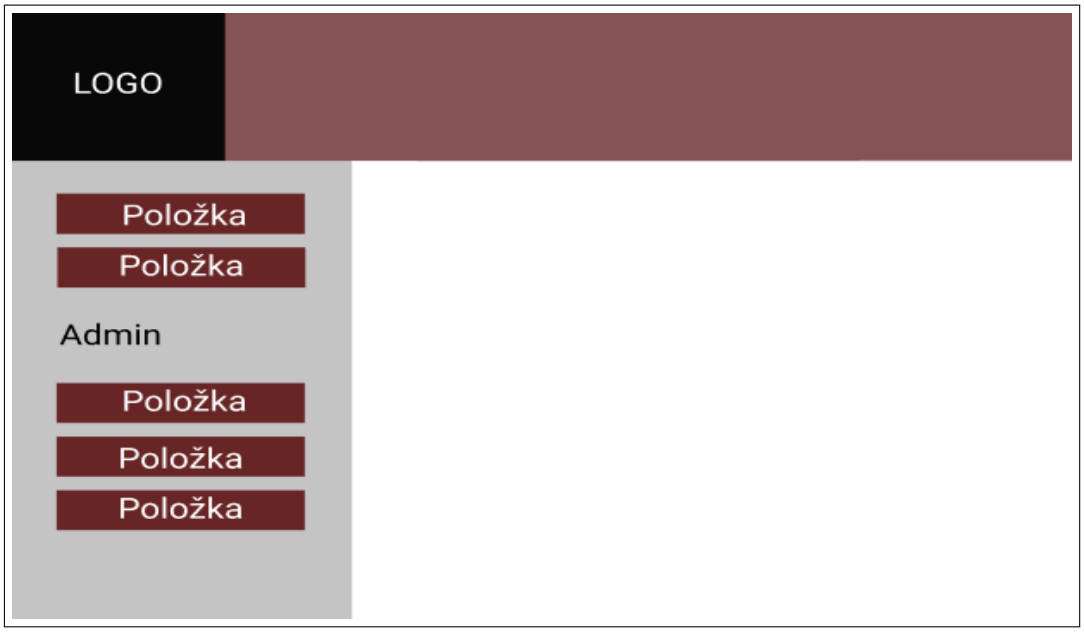
Okno detailu snúšky bude obsahovať informácie patriace k danej snúške a tlačidlá na úpravu a zmazanie.

4.3.4.7 Pridávanie a úprava informácií

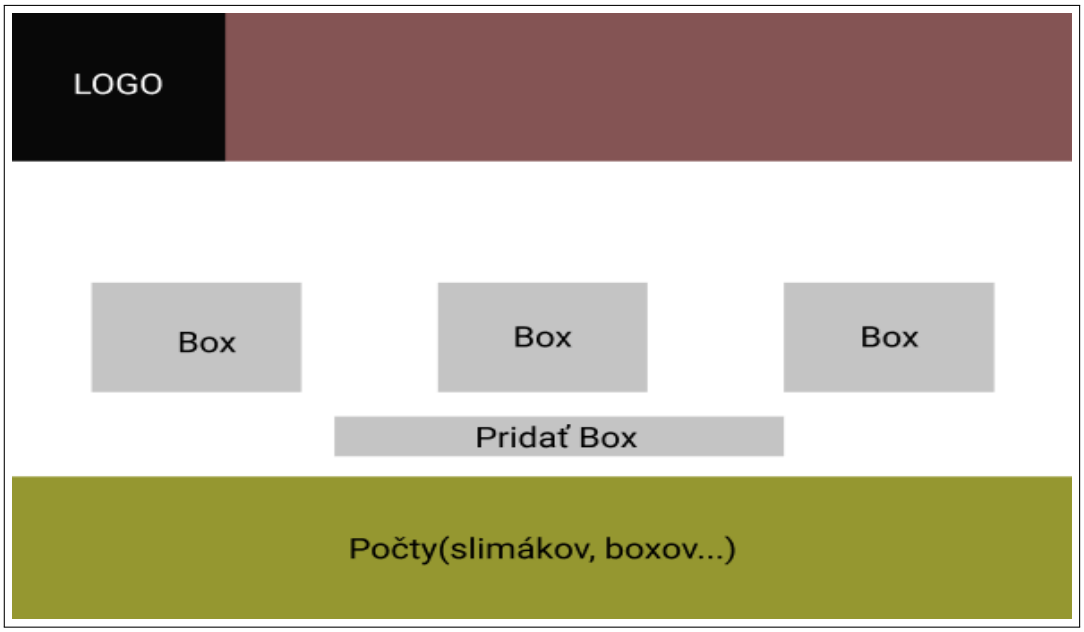
Pridávanie, mazanie a úprava drvivej väčšiny objektov v aplikácií bude zobrazované formou pop-upu. Tento pop-up bude obsahovať polia potrebné k pridaniu/vymazaniu objektu a tlačidla na uloženie alebo prípadne zrušenie pop-upu.

4.3.5 Ukážka návrhu okien

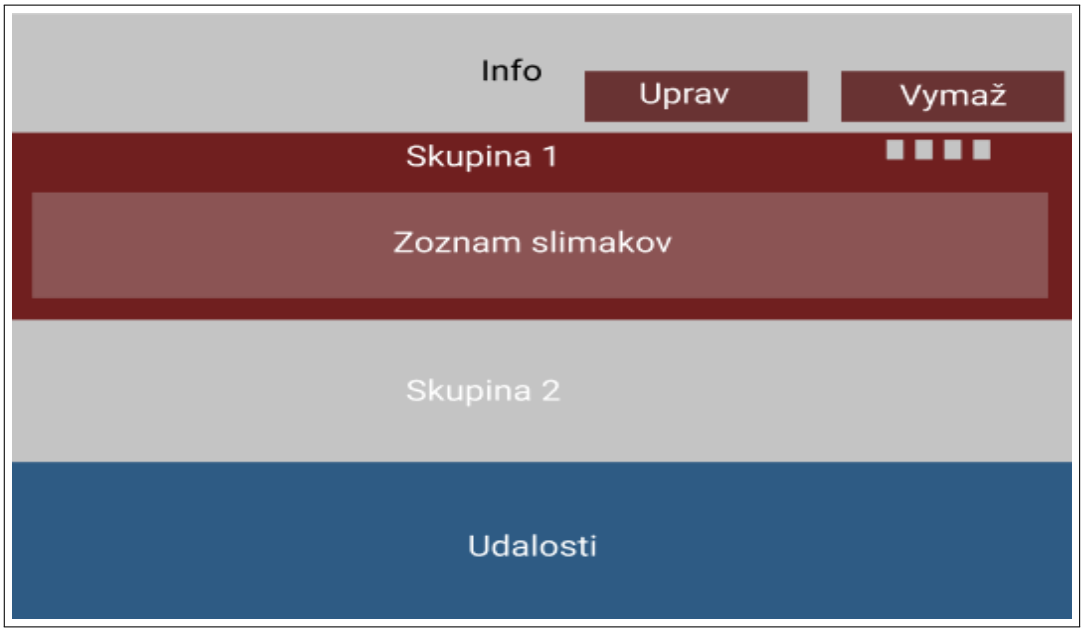
V tejto sekcií sú ukázané návrhy jednotlivých okien vytvorených vrámci návrhu užívateľského rozhrania.



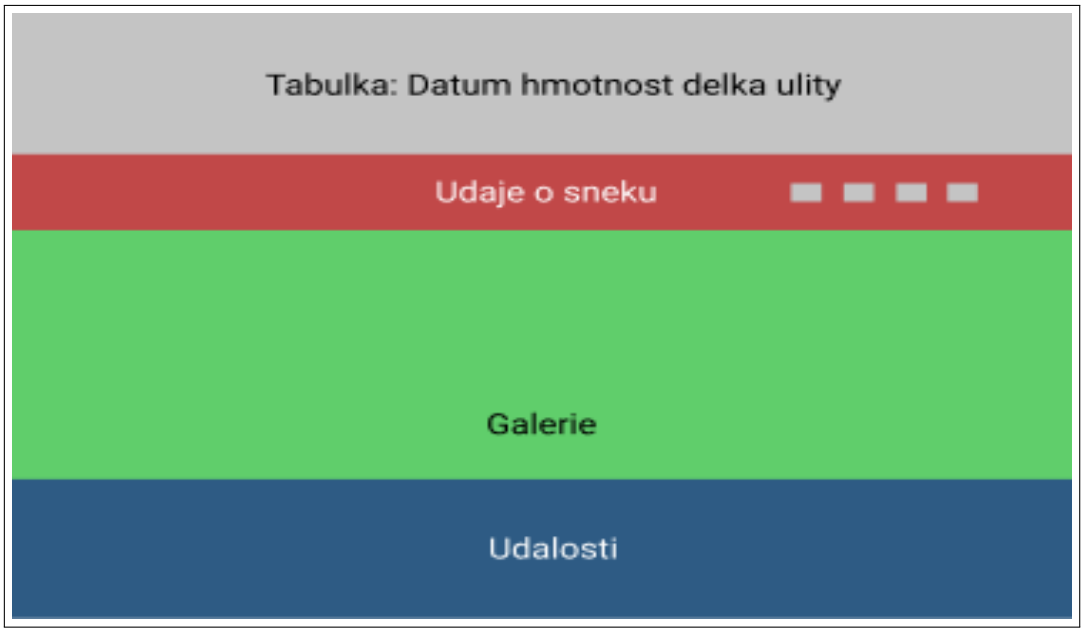
Obr. 4.4 Návrh rozpoloženia menu



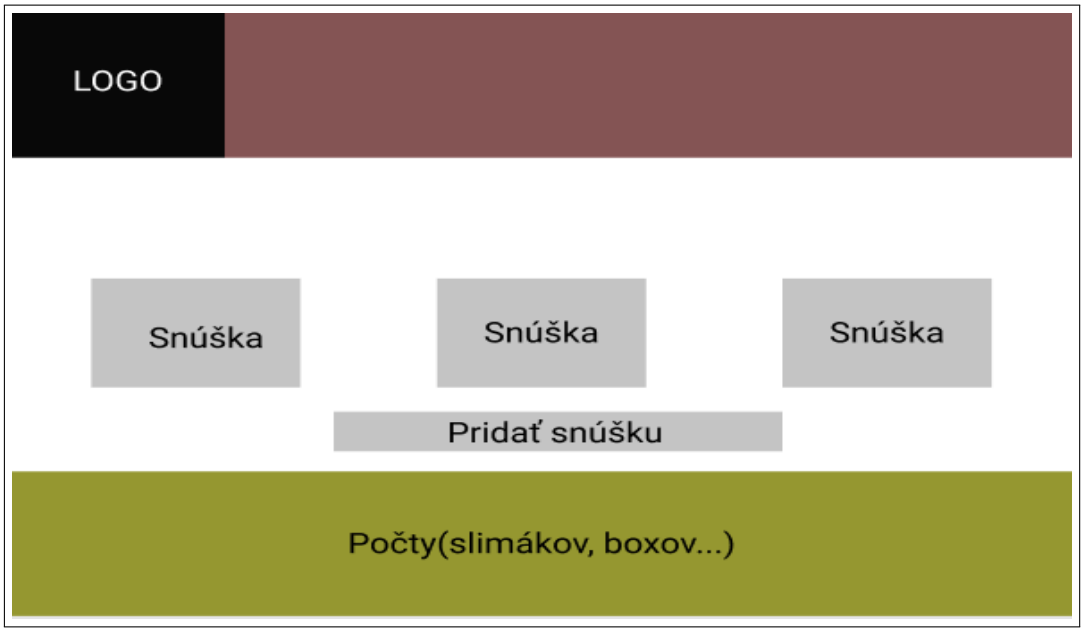
Obr. 4.5 Návrh okna zoznam boxov



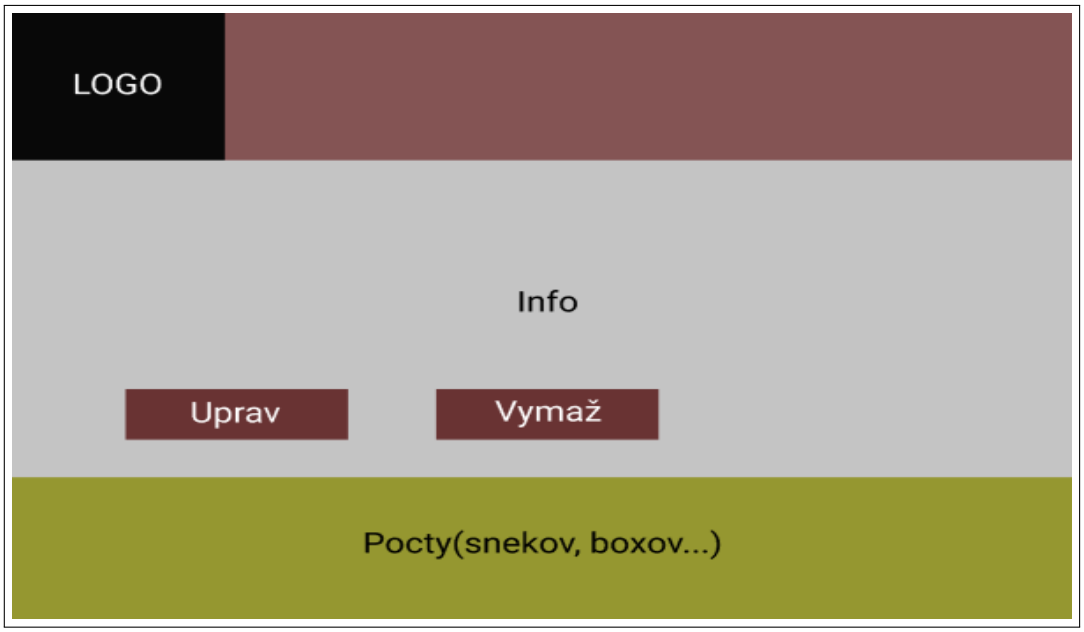
Obr. 4.6 Návrh okna detail boxu



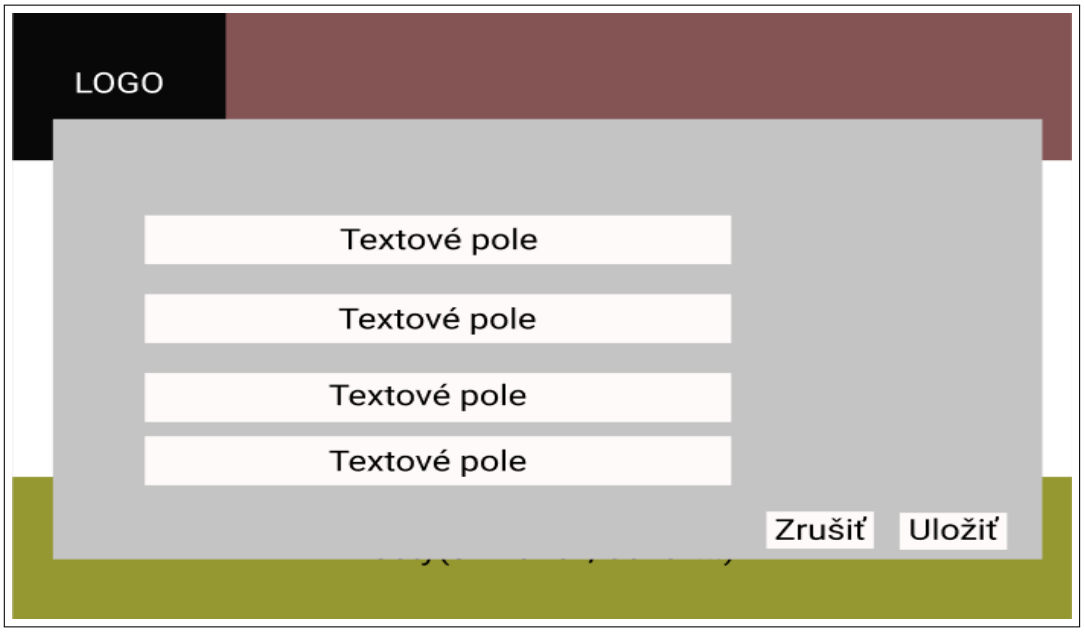
Obr. 4.7 Návrh okna záznamu slimáka



Obr. 4.8 Návrh okna snúšiek



Obr. 4.9 Návrh okna detailu snúšky



Obr. 4.10 Návrh okna pop-upu

Kapitola 5
Implementácia

Ako už bolo spomínané, na implementáciu bol využitý JavaScriptový framework VueJs. Na verzovanie bol použitý verzovací systém Git. Je to voľne dostupný distribuovaný systém na správu verzií, ktorý je navrhnutý tak, aby rýchlo a efektívne zvládal všetko od malých až po veľmi veľké projekty [29].

Popri VueJs boli v implementácií využité aj základné knižnice tohoto frameworku, a to knižnica Vuex a Vue Router. Taktiež bola vo veľkom využívaná knižnica Vuetify, ktorá využíva prvky Material Designu.

Vuex

Vuex [30] je knižnica na podporu state managementu. Slúži ako centralizované úložisko pre všetky komponenty v aplikácii s pravidlami, ktoré zaisťujú, že stav môže byť mutovaný iba predvídateľným spôsobom.

Vue Router

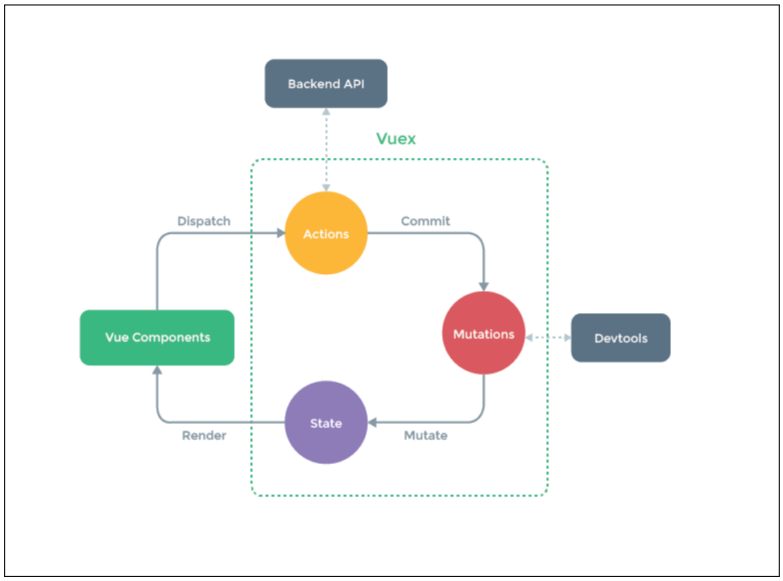
Vue Router [31] je knižnica na uľahčenie smerovania v aplikácií.

Vuetify

Vuetify [32] je knižnica na tvorbu užívateľského rozhrania. Obsahuje mnoho predom pripravených komponentov založených na špecifikácií Material Design [33].

5.1 Práca s REST API

Pre lepšiu prehľadnosť, rozšíriteľnosť a prácu s REST API boli jednotlivé metódy implementované ako prvky akcií v knižnici Vuex. Získané dáta sú následne uložené v globálnom stave aplikácie. Vďaka tomuto prístupu môžu komponenty zdieľať informácie bez potreby posielania si dát medzi sebou, alebo znovu volaním metódy REST API. Tento globálny stav je zároveň reaktívny, takže zmena dát vykonaná v jednom komponente sa prejaví aj v rovnakých dátach iných komponentov. Podrobná implementácia využívaných metód REST API sa nachádza v súbore *actions.js*. Útržok kódu 5.3 ukazuje implementáciu jednej z týchto metód.



Obr. 5.1 Popis fungovania state managementu knižnice Vuex[30]

Výpis kódu 5.1 Ukážka implemntácie metódy getSnails

```
export const getSnails = ({commit}) => {
  return new Promise((resolve, reject) => {
    axios.get("snek").then(response =>{
      commit("GET_SNAILS", response.data.map(d => d.snek))
      resolve(response)
    }).catch(e => {
      error({commit}, e.response.data.message)
      reject(e)
    })
  })
}
```

Pre zjednodušenie práce s HTTP požiadavkami bola využitá knižnica Axios [34].

5.2 Smerovanie

Ako už bolo spomínané, na smerovanie bola využitá knižnica Vue Router. Pri smerovaní bolo dôležité oddeliť časti aplikácie určené novým užívateľom, aktivovaným užívateľom a administrátorom. K tomu pomohol takzvaný *Navigation Guard* určený pri registrovaní smerovania. Práva na prístup k oknám boli teda vďaka funkciám v 5.2 rozdelené na 3 časti.

Výpis kódu 5.2 Implementácia Navigation Guards

```
const ifNotAuthenticated = (to, from, next) => {
  if (!store.getters.isLoggedIn) {
```

```
    next()
    return
  }
  next("/")
}

const ifAuthenticated = (to, from, next) => {
  if (store.getters.isLoggedIn) {
    next()
    return
  }
  next("/login")
}

const ifAdmin = (to, from, next) => {
  if (store.getters.isLoggedIn && store.state.user.userRoles && store.state.
      user.userRoles.includes("ADMIN")) {
    next()
    return
  }
  next("/home")
}
```

Výpis kódu 5.3 Príklad registrovania smerovania:

```
const routes = [
  {
    path: "/users",
    name: "Users",
    component: Users,
    beforeEnter: ifAdmin,
  }
]

const router = new VueRouter({
  mode: "history",
  base: process.env.BASE_URL,
  routes
})
```

5.3 Autentizácia

Registrácia a prihlásenie sú verejne dostupné pomocou formulárov. Po prihlásení sa aktivovaný užívateľ dostane na okno zoznamu boxov, možnosť na odhlásenie sa nachádza v podobe tlačidla *logout* v hlavičke stránky. Informácie na autentizáciu užívateľa sú uložené v Local Storage, takže užívateľ zostane prihlásený aj po znovuotvorení stránky. Local Storage je lokálne úložisko dát priamo v prehliadači dostupné pomocou JavaScriptu. Po odhlásení sa tieto informácie z Local Storage vymažú.

5.4 Validácia

Pri pridávaní a úprave informácií na stránke je potrebná validácia užívateľského vstupu. V aplikácií je využitá vstavaná validácia knižnice Vuetify. Validácia prebieha pomocou overovania pravidiel a je prítomná v prevažnej časti formulárov. V časti kódu nižšie sa nachádza príklad pravidiel použitých pri validácií užívateľského vstupu a v 5.5 sa nachádza príklad využitia týchto pravidiel v poli formulára.

Výpis kódu 5.4 Príklad validačných pravidiel:

```
const emailRegex = /(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_
    `{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01
    -\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0
    -9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)
    \.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01
    -\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])
    +)\])/

export const emailRules = [
    v => !!v || 'E-mail je povinný',
    v => emailRegex.test(v) || 'E-mail musí být validní'
]

export const passwordRules = [
    v => !!v || 'Heslo je povinný',
    v => /^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#\$%\^&\*,:;])(?=.{8,})/.
        test(v) ||
     'Heslo musí obsahovat minimálne 8 znakú, velké písmeno, číslo a specialní
         znak'
]

export const intRules = [
    v => Number.isInteger(Number(v)) || 'Hodnota musí být celé číslo'
]
```

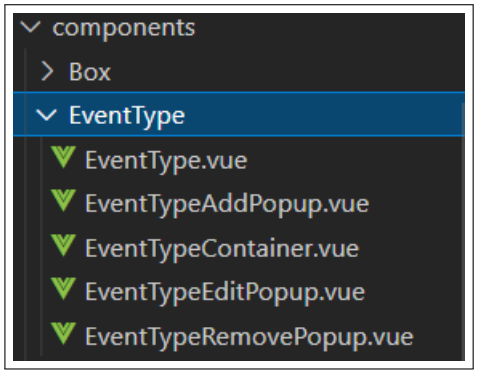
Výpis kódu 5.5 Príklad použitia validačných pravidiel:

```
<v-text-field color="secondary" :rules="textRules" v-model="eventType.popis"
    label="Popis*" required></v-text-field>
```

5.5 Štruktúra komponent aplikácie

Hlavné komponenty, presnejšie komponenty patriace k boxom, skupinám, slimákom, snúškam, taxonómiám a typom udalostí, zachovávajú podobnú štruktúru. Základný komponent, ktorý určuje vzhľad a logiku jedného objektu, je použitý v komponente, ktorý sa stará o zobrazenie celého zoznamu týchto objektov. Tento komponent je potom použitý v okne, ktoré je zobrazené koncovému užívateľovi. Zároveň sú pre každý takýto hlavný komponent implementované komponenty zobrazu-

júce pop-upy umožňujúce vymazanie, editáciu alebo tvorbu nového záznamu. Na výpise kódu nižšie je ukázaná implementácia komponentu zobrazujúceho zoznam objektov, presnejšie typov udalostí.



Obr. 5.2 Ukážka rozdelenia typov udalostí na jednotlivé komponenty

Výpis kódu 5.6 Implementácia komponenty EventTypeContainer

```
<template lang="">
<v-container fluid>
    <v-row allign="center">
        <v-col v-for="eventType in eventTypes" :key="eventType.udalostTypId" :
            cols='12' :md='6' :lg="4">
            <div>
                <EventType :eventTypeId="eventType.udalostTypId"></EventType>
            </div>
        </v-col>

        <v-col>
            <v-tooltip bottom>
                <template v-slot:activator="{ on, attrs }">
                    <div v-bind="attrs" v-on="on">
                        <EventTypeAddPopup></EventTypeAddPopup>
                    </div>
                </template>
                <span>řPidat typ události</span>
            </v-tooltip>
        </v-col>

    </v-row>
</v-container>
</template>

<script>
import EventTypeAddPopup from '@/components/EventType/EventTypeAddPopup.vue'
import EventType from "@/components/EventType/EventType.vue"
export default {
    props: {
        eventType: String
```

```
    },
    components: {
        EventTypeAddPopup,
        EventType
    },
    computed: {
        eventTypes() {
            return this.$store.state.eventTypes.filter(e => e.typ === this.
                eventType);
        },
    },
    created() {
        this.$store.dispatch("getEventTypes");
    },
};
</script>

<style lang="">
</style>
```

5.6 Dátové tabuľky

Zobrazenie meraní a užívateľov je implementované pomocou dátových tabuliek knižnice Vuetify. Tieto tabuľky umožňujú hľadanie a zoraďovanie záznamov na základe jednotlivých stĺpcov. Taktiež takáto tabuľka umožňuje vysokú prispôsobiteľnosť, ktorá bola využitá napríklad pri zobrazovaní ikon slúžiacich na úpravu alebo vymazanie záznamu. V útržku kódu je možnosť vidieť implementáciu tabuľky meraní.

Výpis kódu 5.7 Príklad implementácie dátovej tabuľky

```
    <v-card>
        <v-card-title>
            <v-text-field color="secondary" v-model="search" append-icon="mdi-
                magnify" label="Hledat" single-line hide-details></v-text-
                field>
            <v-spacer></v-spacer>
            <MeasuresAddPopup class="ml-4" :snailId="snailId"></
                MeasuresAddPopup>
        </v-card-title>
        <v-data-table :headers="headers" :search="search" :items="mereni"
            class="elevation-1">
            <template v-slot:item.actions="{ item }">
                <MeasuresDeletePopup :measureId="item.mereniSnekId"></
                    MeasuresDeletePopup>
                <MeasuresEditPopup :measureProp="Object.assign({}, item)"></
                    MeasuresEditPopup>
            </template>
        </v-data-table>
    </v-card>
</div>
```

5.7 Popis finálnych častí aplikácie

Jednotlivé časti aplikácie sú rôzne dostupné v závislosti od rolí užívateľa. Pre neprihlásených a neaktivovaných užívateľov sú dostupné iba okná na registráciu a prihlásenie. Aktivovaní užívatelia majú prístup k oknám boxov a snúšiek. Samotné pristupovanie k skupinám, slimákom a funkcionalitám súvisiacim s nimi sú dostupné v rámci okna boxu. Aktivovaní užívatelia s pridanou rolou *ADMIN* majú prístup okrem všetkých okien, ktoré majú aj bežní užívatelia, aj k oknám taxonómií, typom udalostí a samozrejme k oknu na správu užívateľov.

5.7.1 Boxy, skupiny, slimáky

Táto časť aplikácie je dostupná všetkým aktivovaným užívateľom. Obsahuje zoznam boxov s možnosťami na zobrazenie detailu boxu, vymazanie boxu a taktiež tlačidlo na pridanie nového boxu. Po kliknutí na zobrazenie detailu boxu sa aplikácia presmeruje na okno detailu boxu.

Tu sa, ako už názov napovedá, nachádzajú detailné informácie boxu, ale taktiež aj zoznam udalostí patriacich k danému boxu a zoznam slimákov/skupín. Tento zoznam sa upravuje na základe počtu skupín v boxe. V prípade, že box obsahuje iba jednu skupinu, väčšinou je to základná skupina, ktorá sa vytvorí spolu s boxom, zobrazí sa rovno zoznam slimákov. V prípade ale, že box obsahuje viac ako jednu skupinu, zobrazí sa interaktívny zoznam týchto skupín, kde každá skupina po rozkliknutí obsahuje zoznam slimákov patriacich do danej skupiny. Taktiež hlavička skupiny obsahuje rozklikávacie menu s možnosťami na vymazanie, úpravu alebo zobrazenie snúšiek patriacich k danej skupine.

Samotná karta slimáka ukazuje jeho základné informácie s tlačidlami na ich úpravu, alebo celkové odstránenie slimáka z aplikácie. Taktiež je tu tlačidlo na zmenu skupiny slimáka. Nachádza sa tu aj tabuľka meraní, zoznam udalostí patriacich k slimákovi a galéria, kde je možné pridávať a odoberať fotky slimáka.

5.7.2 Snúšky

V tejto časti sa nachádza zoznam všetkých snúšok. Tento zoznam je možné filtrovať a zoraďovať. Každá snúška v zozname má tlačidlo na vymazanie a zobrazenie detailov. Po rozkliknutí tlačidla na zobrazenie detailov sa aplikácia presmeruje na okno detailu snúšky. Toto okno poskytuje detailné informácie o snúške a taktiež zoznam udalostí patriacich k nej. Nachádzajú sa tu aj tlačidlá na upravenie alebo vymazanie snúšky.

5.7.3 Typy udalostí

Toto okno je dostupné iba užívateľom s rolou administrátora. Nachádza sa tu zoznam na výber typu udalostí. Tie môžu byť: udalosti boxu, udalosti snúšky, udalosti slimáka alebo udalosti skupiny. Po vybraní požadovaného typu udalosti sa zobrazí ich zoznam. Každá udalosť obsahuje tlačidlá na úpravu alebo vymazanie. Taktiež sa tu nachádza tlačidlo na pridanie novej udalosti.

5.7.4 Taxonómia

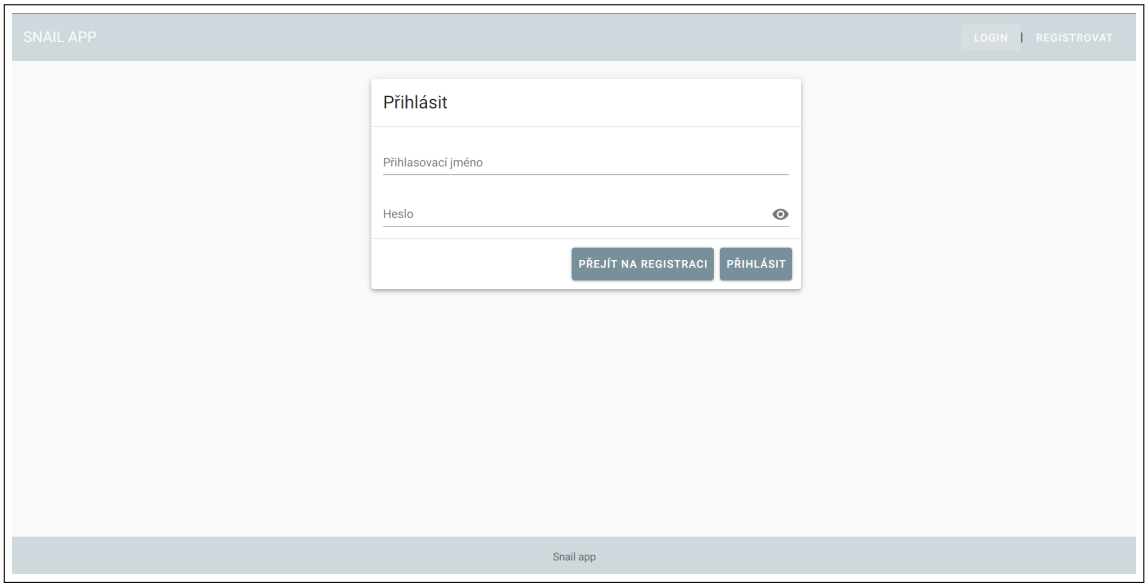
Podobne ako okno typov udalostí aj toto okno je prístupné iba užívateľom s rolou administrátora. Nachádza sa tu zoznam taxónov. Každý taxón obsahuje tlačidlo na úpravu alebo vymazanie. Nachádza sa tu aj tlačidlo na pridanie taxónu.

5.7.5 Užívatelia

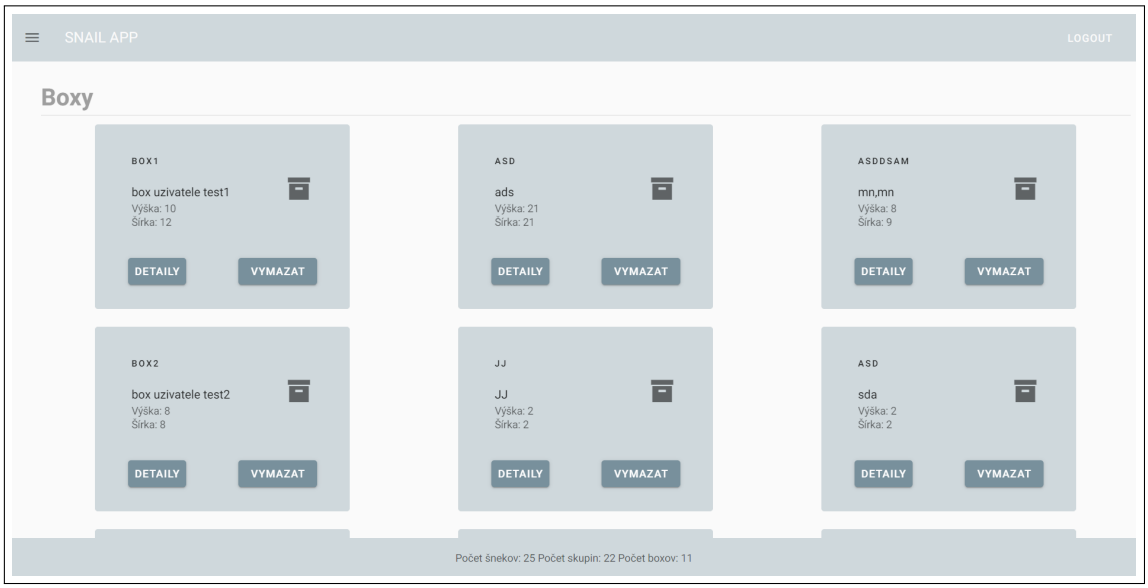
Okno užívateľov je tiež prístupné iba užívateľom s rolou administrátora. Nachádza sa tu tabuľka užívateľov. Tabuľka umožňuje zoraďovanie a vyhľadávanie užívateľov podla jednotlivých informácií, ako aj filtrovanie iba neaktívnych užívateľov. Pri každom užívateľovi sú zobrazené jeho detailné informácie a tlačidlo na úpravu užívateľa. V rámci úprav je k dispozícii možnosť aktivácie, deaktivácie, alebo zmeny role užívateľa.

5.8 Ukážky finálnej aplikácie

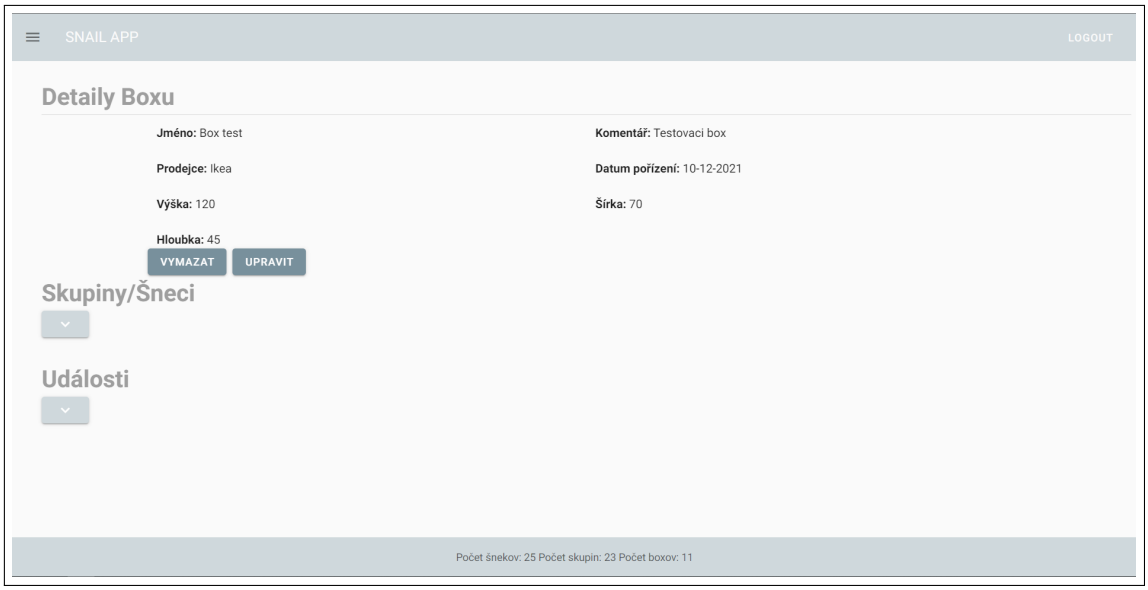
Na následujúcich obrázkoch je možné vidieť ukážky implementácie informačného systému pre podporu chovu slimákov.



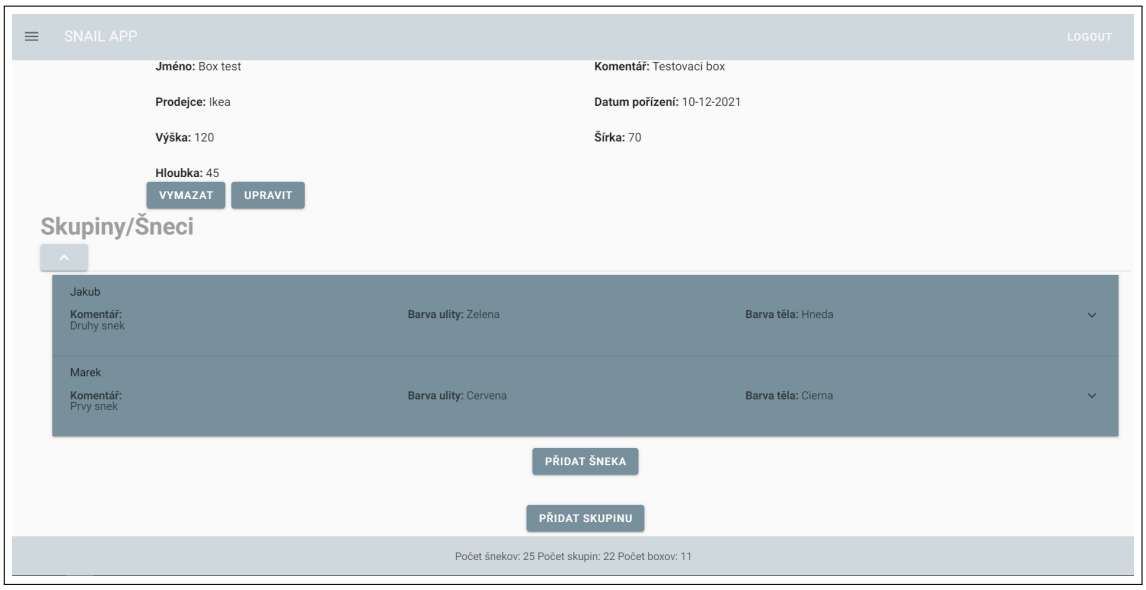
Obr. 5.3 Prihlasovacie okno



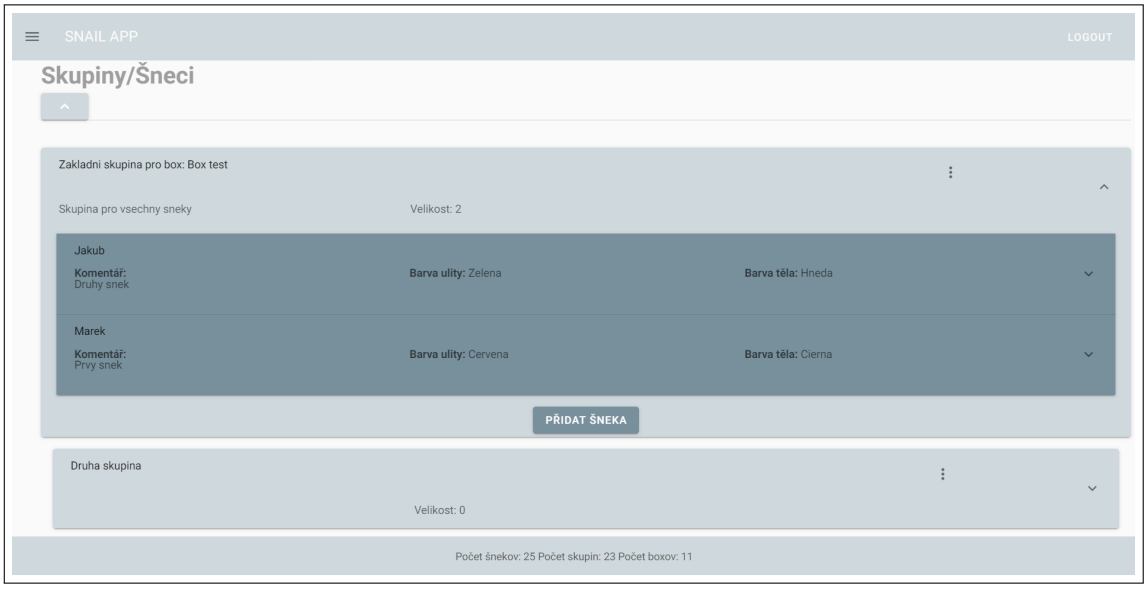
Obr. 5.4 Okno zoznamu boxov zobrazené po prihlásení



Obr. 5.5 Okno detailu boxu



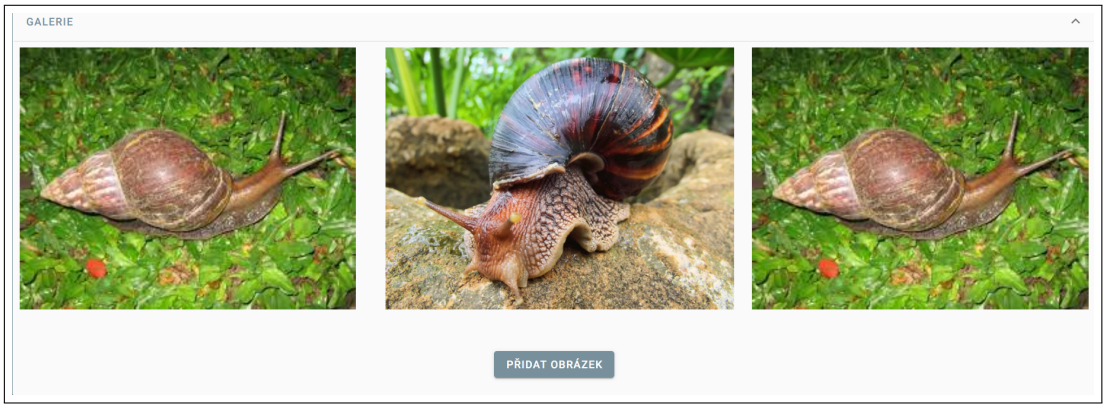
Obr. 5.6 Okno detailu boxu obsahujúceho iba jednu skupinu



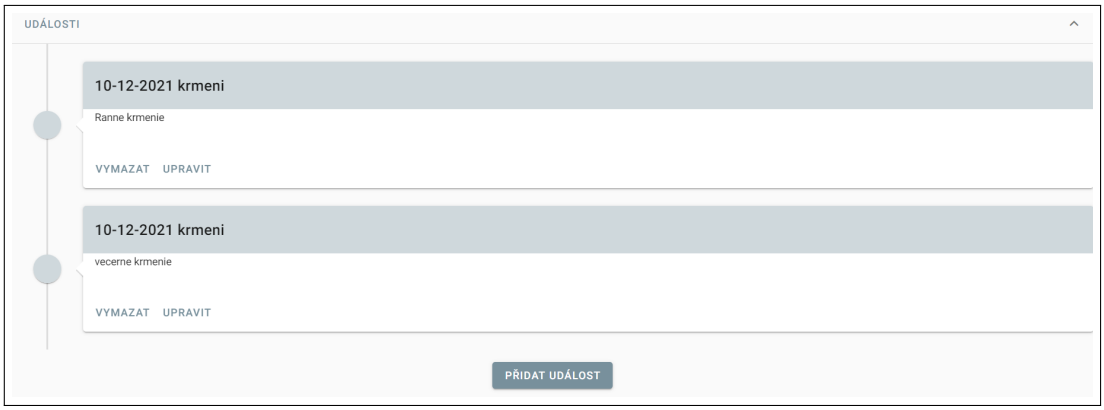
Obr. 5.7 Okno detailu boxu obsahujúceho viac ako jednu skupinu

Obr. 5.8 Záznam slimáka

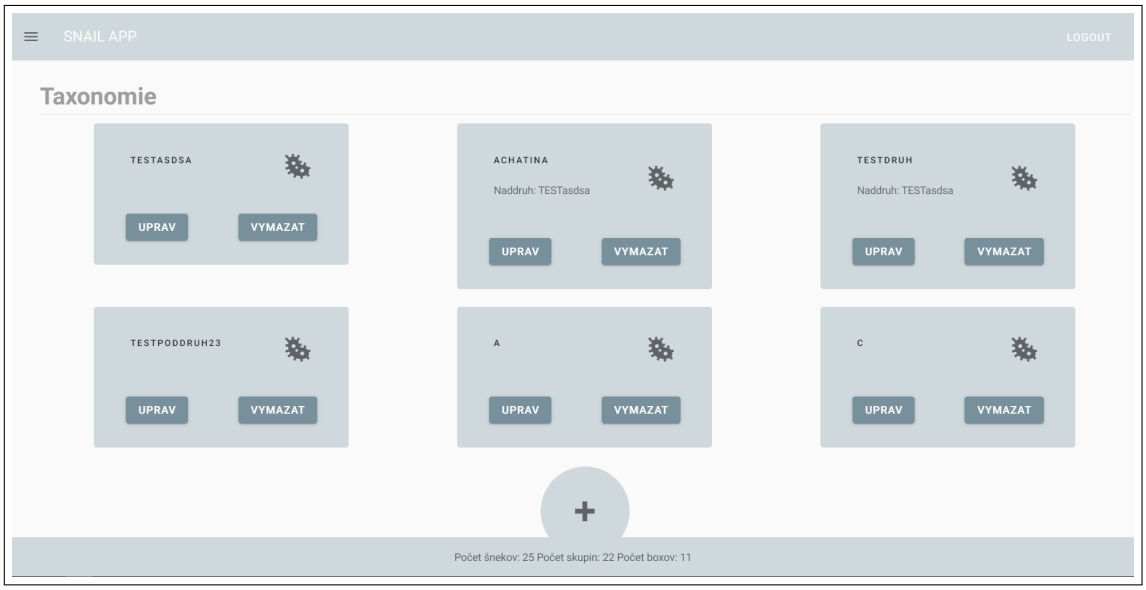
Obr. 5.9 Tabulka meraní slimáka



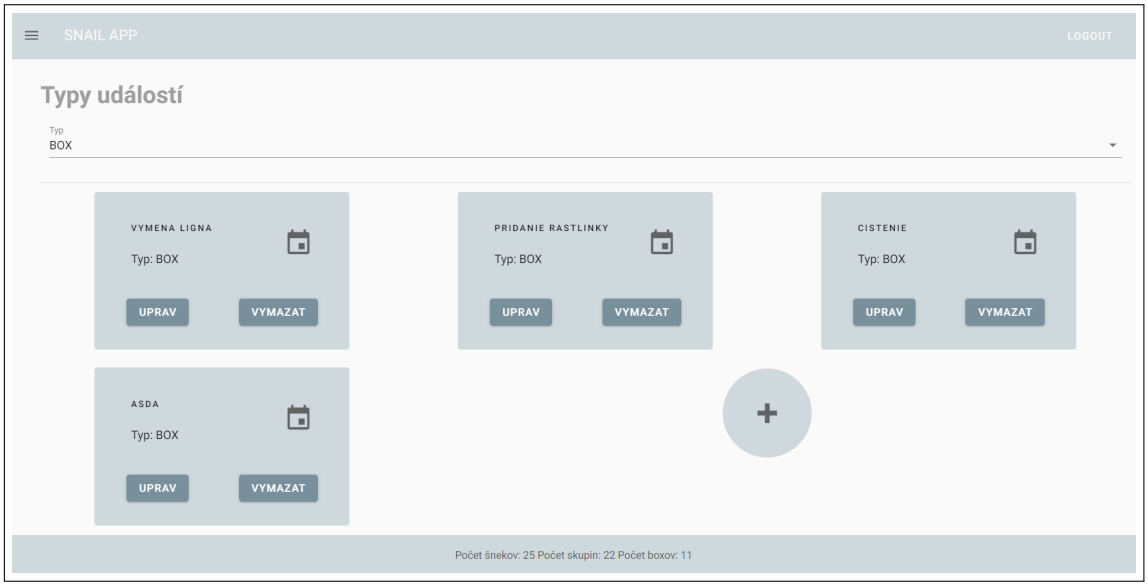
Obr. 5.10 Galéria slimáka



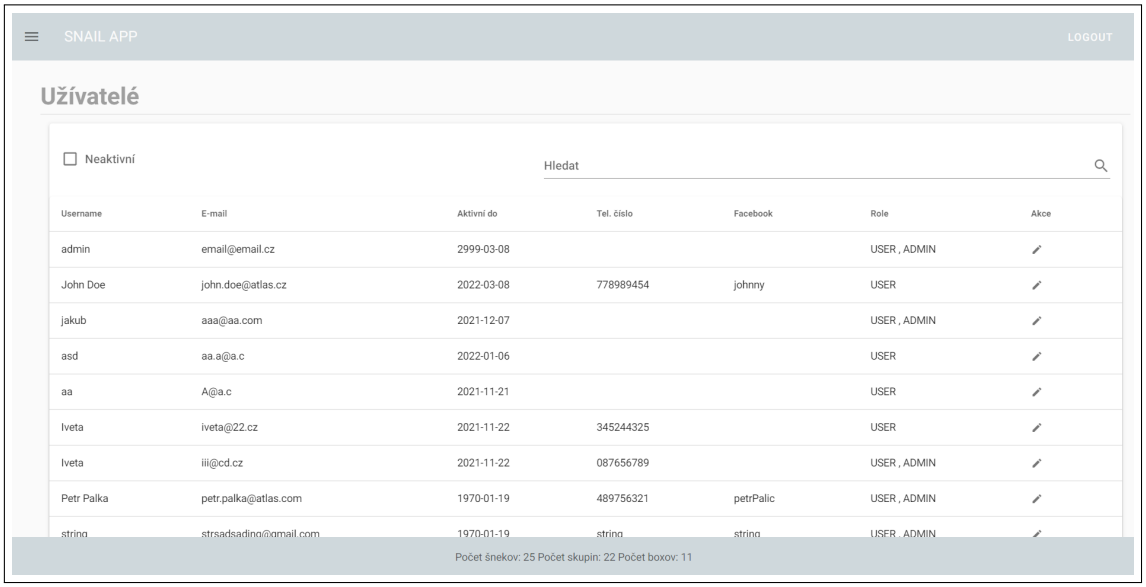
Obr. 5.11 Zoznam udalostí



Obr. 5.12 Okno zoznamu taxonómií



Obr. 5.13 Okno zoznamu typov udalostí



Obr. 5.14 Okno zoznamu užívateľov

Kapitola 6
Testovanie

Pre otestovanie funkčnosti aplikácie bola využitá metóda užívateľského testovania. Táto metóda bola vybraná na základe článku [35]. Pri užívateľskom testovaní sa sleduje chovanie užívateľov pri priechode aplikáciou. Táto metóda sa najčastejšie používa pre otestovanie použiteľnosti aplikácie, a môže odhaliť nové chyby ktoré neboli zachytené pri samotnom vývoji.

Pre otestovanie aplikácie bolo vybraných na základe [36] 5 užívateľov. Pri výbere bola snaha o čo najpestrejšie rozdelenie užívateľov, či už na základe veku, počítačovej gramotnosti, ale aj náhľadu do sveta chovateľov slimákov. Vybraný boli nasledujúci testeri:

1. Muž, 23 rokov, absolvent FIT ČVUT, základné znalosti o chove slimákov.
2. Žena, 22 rokov, pokročilé počítačové znalosti, základné znalosti o chove slimákov.
3. Muž, 53 rokov, bežné počítačové znalosti, základné znalosti o chove slimákov.
4. Žena, 52 rokov, bežné počítačové znalosti, žiadne znalosti o chove slimákov.
5. Žena, 23 rokov, absolventka FIT ČVUT, žiadne znalosti o chove slimákov.

V aplikácií boli predpripravené typy udalostí, taxonómie a administrátorský účet. Pre testovacie účely, mali 3 z vybraných testerov pridelené administrátorské práva.

6.1 Testovacie scenáre

Pre testerov boli pripravené nasledujúce testovacie scenáre:

1. **Autorizácia** Zaregistrujte sa do aplikácie, počkajte na aktiváciu administrátorom a následne sa prihláste do aplikácie. Po prihlásení sa odhláste a pokúste prihlásiť znova.
2. **Boxy** Pridajte si nový box do aplikácie, po pridaní prejdite na detaily boxu a upravte jeho vlastnosti. Pridajte udalosť. Následne upravte vlastnosti tejto udalosti a udalosť zmažte.
3. **Skupiny** V okne detaily boxu pridajte novú skupinu do aplikácie, po pridaní upravte jej vlastnosti, prejdite na snúšky danej skupiny a vráťte sa naspäť.

4. **Slimáky** Pridajte nového slimáka do aplikácie, pridajte mu meranie a udalosti, následne upravte vlastnosti daných prvkov a zmažte ich. Prejdite na galériu a pridajte nový obrázok. Pridaný obrázok v galérií zmažte.
5. **Snúšky** Prejdite v navigačnom menu na okno snúšky. Pridajte 3 nové snúšky. Po pridaní sa pokúste snúšky zoradiť a prefiltrovať. Vyberte si jednu snúšku, ktorú upravíte, a jednu snúšku zmažte.

Pre testerov s administrátorskými právami boli pridané nasledujúce testovacie scenáre navyše:

1. **Typy udalostí** Prejdite v navigačnom menu na okno události. Pridajte udalosť ľubovoľného typu. Po pridaní si vyberte v rozbaľovacom zozname rovnaký typ, ako ste práve pridali, a presvedčte sa, že vami vytvorený záznam je viditeľný. Upravte typ udalosti a následne ho zmažte.
2. **Taxonómia** Prejdite v navigačnom menu na okno taxonomie. Vytvorte aspoň 2 taxóny kde jeden bude nadranený druhému. Vyberte si jeden taxón, ktorý upravíte, a jeden zmažte.
3. **Užívatelia** Prejdite v navigačnom menu na okno užívatelé. Zobrazte si neaktívnych užívateľov a jedného takého užívateľa aktivujte. Vyberte si užívateľa a upravte mu role na administrátorské. Deaktivujte užívateľa.

Pred samotným užívateľským testovaním absolvoval priechod aplikáciou podla testovacieho scenáru aj samotný autor. Tento priechod bol z dôvodu uistenia sa, že všetky dôležité časti aplikácie fungujú správne, a že testovacie scenáre pokrývajú všetky dôležité funkcionality aplikácie.

6.2 Priebeh testovania

Testeri dostali možnosť vybrať si internetový prehliadač, v ktorom si aplikáciu prešli. Pre otestovanie responzivity aplikácie, absolvoval každý tester priechod aplikáciou aj pri simulovaní rozlíšenia rôznych mobilných telefónov a tabletov v internetovom prehliadači.

Dohromady teda prebehlo 10 priechodov aplikáciou, z toho 3 priechody pri simulovaní rozlíšenia mobilného telefónu a 2 pri simulovaní rozlíšenia tabletu. Najvyberanejší prehliadač bol Google Chrome, ktorý si vybrali až 3 testeri. Jeden tester absolvoval priechod v prehliadači Microsoft Edge, a jeden v prehliadači Mozilla Firefox. Pri testovaní sa kládol dôraz ako na intuitívnosť aplikácie, tak na odhalenie chýb implementácie a rozdielov v rozličných prehliadačoch.

6.3 Výsledky testovania

Pri testovaní bolo odhalené množstvo chýb a problémov, či už podotknutých samotnými testermi alebo vypozorovaných autorom pri prechode testerov aplikáciou. Neboli však odhalené žiadne rozdiely medzi prehliadačmi a takisto ani chyby, ktoré by boli špecifické iba pre jeden prehliadač.

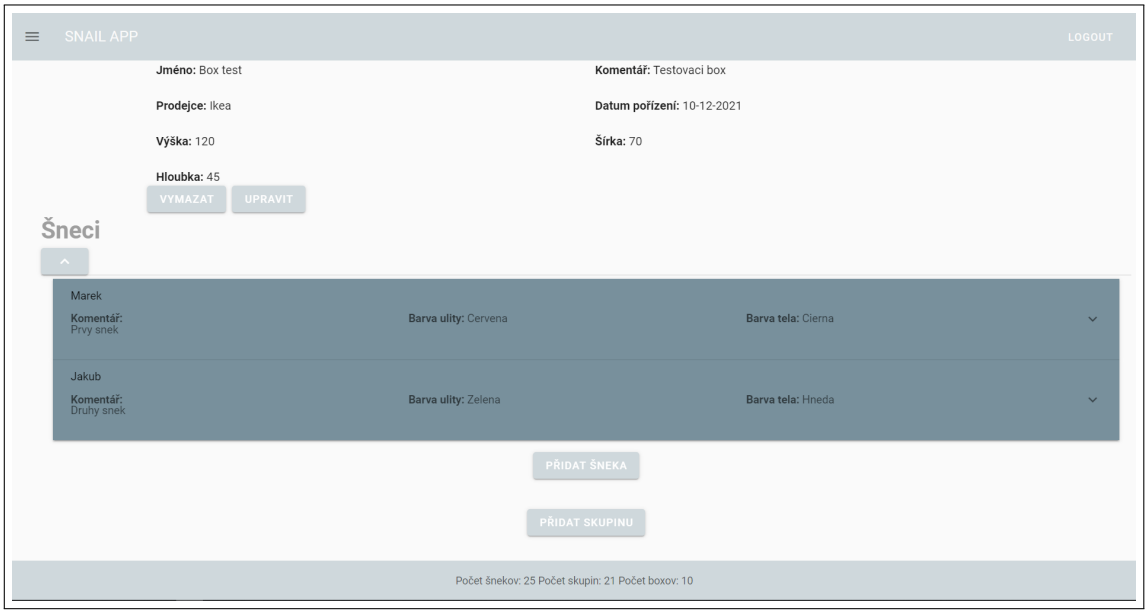
Medzi chyby odhalené pri testovaní patrí:

- chýbajúce ukazovatele povinných polí pri registrácií užívateľa,
- mätúci popis tlačidiel pri registrácií a prihlásení,
- odsadenie tlačidiel v udalostiach,

- miznutie vyberača dátumu po kliknutí,
- nesprávne vyberanie typov udalostí pri tvorbe novej udalosti,
- neintuitívne tlačidlo na pridanie nového slimáka,
- pretekanie textu pri skupinách a slimákoch na mobilných zariadeniach,
- prijímanie nulovej veľkosti pri tvorbe snúšky,
- farba písma v hlavičke skupiny,
- problém nájsť skupiny a slimákov v okne detailu boxu,
- farebná schéma tlačidiel evokuje pocit, že sa nedajú kliknúť.

Testeri oceňovali hlavne intuitívnosť riešenia, ako aj vzhľad a použité UI prvky. Najčastejšie spomínaným problémom bola farebná schéma tlačidiel. Tá evokovala pocit, že sa nedajú zakliknúť. Túto výčitku mali 4 z 5 testerov. Preto bola farba väčšiny tlačidiel v aplikácií zmenená na tmavší odtieň. Takisto mali testeri problém zorientovať sa v okne detailu boxu, a to hlavne pri pridávaní skupiny a slimáka. Preto bol názov tejto sekcie lepšie pomenovaný. Ostatok zistených problémov boli prehliadnuté chyby pri implementácií, a väčšinou vyžadovali iba jednoduchú úpravu, aby fungovali správne.

Bohužiaľ kvôli reštrikciám spojenými s pandémiou koronavírusu, nemohlo testovanie prebehnúť naraz ale prebiehalo postupne počas troch týždňov. Preto boli odhalené chyby implementácie postupne odstraňované medzi jednotlivými testovaniami. Problémy, ktoré sa týkali intuitívnosti a vzhľadu aplikácie, však boli ponechané do konca testovania.



Obr. 6.1 Pôvodná farebná schéma tlačidiel a pôvodný popis sekcie slimákov a skupín

Obr. 6.2 Nová farebná schéma tlačidiel a nový popis sekcie slimákov a skupín

Kapitola 7
Záver

Cieľom tejto práce bolo vytvorenie front-endu informačného systému pre podporu chovateľov slimákov. Tento systém by mal slúžiť na uľahčenie uchovávania informácií spojených s chovom. Dôležité bolo zozbierať a analyzovať požiadavky na systém, a na ich základe vytvoriť štúdiu uskutočniteľnosti, ktorú sa v tejto práci podarilo spísať. Práca tiež porovnala jednotlivé technológie používané pri tvorbe front-endu, vybrala na základe analýzy najlepšiu možnosť a následne túto technológiu použila v implementácií.

Front-end informačného systému bol v rámci tejto práce vytvorený a úspešne prepojený pomocou REST API s back-endovou časťou vytvorenou autorom Mgr. Mikolasom Teskou. Výsledkom je informačný systém dostupný z webu, ktorý spĺňa všetky predom definované požiadavky.

Zároveň bolo front-endové riešenie otestované pomocou užívateľského testovania, ktoré úspešne overilo jeho správnosť.

Implementácia takisto poskytuje priestor na prípadné rozšírenia informačného systému v budúcnosti.

Bibliografia

1. TESKA, Mikolas. *INFORMAČNÍ SYSTÉM PRO PODPORU CHOVU ŠNEKŮ* [online] [cit. 2021-10-03]. Dostupné z : https://dspace.cvut.cz/handle/10467/95449.
2. CRNET S.R.O. *cavalierclub.cz* [online] [cit. 2021-10-08]. Dostupné z : https://www.cavalierclub.cz/.
3. PSSR Š.P. *Plemenársky informačný systém* [online] [cit. 2021-10-08]. Dostupné z : https://www.plis.sk/.
4. GESPET SOFTWARE. *Gespet: software for pet professional - Pet apps* [online] [cit. 2021-10-08]. Dostupné z : https://www.gespet.com/.
5. THE INVESTOPEDIA TEAM. *Feasibility Study* [online]. Investopedia, aug 2021 [cit. 2021-11-26]. Dostupné z : https://www.investopedia.com/terms/f/feasibility-study.asp.
6. META. *Podmienky a zásady - ZAKÁZANÝ OBSAH - Zvieratá a živočíšne produkty* [online] [cit. 2021-10-18]. Dostupné z : https://www.facebook.com/policies_center/commerce/animals.
7. NÁPLAVA, Ing. Pavel. *BI-TIS PROCESNÍ POHLED, ZPŮSOBY IMPLEMENTACE* [online]. 2019 [cit. 2021-11-26]. Dostupné z : https://moodle-vyuka.cvut.cz/pluginfile.php/428549/mod_page/content/17/Prednaska04.pdf.
8. MDN. *HTTP* [online] [cit. 2021-11-18]. Dostupné z : https://developer.mozilla.org/en-US/docs/Web/HTTP.
9. NEOTERIC. *Single-page application vs. multiple-page application* [online] [cit. 2021-11-21]. Dostupné z : https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58.
10. W3. *AJAX Introduction* [online] [cit. 2021-11-21]. Dostupné z : https://www.w3schools.com/js/js_ajax_intro.asp.
11. MDN. *Server-side web frameworks* [online] [cit. 2021-11-21]. Dostupné z : https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks.
12. SINGH, Vijay. *What is a Framework? [Definition] Types of Frameworks* [online] [cit. 2021-11-21]. Dostupné z : https://hackr.io/blog/what-is-frameworks.
13. MDN. *JavaScript* [online] [cit. 2021-11-21]. Dostupné z : https://developer.mozilla.org/en-US/docs/Web/JavaScript.

14. *What are the uses of JavaScript?* [Online] [cit. 2021-11-21]. Dostupné z : https://www.javatpoint.com/what-are-the-uses-of-javascript.
15. ECMA INTERNATIONAL. *ECMAScript® 2022 Language Specification* [online] [cit. 2021-11-21]. Dostupné z : https://tc39.es/ecma262/.
16. ECMA INTERNATIONAL. *ECMAScript® 2022 Internationalization API Specification* [online] [cit. 2021-11-21]. Dostupné z : https://tc39.es/ecma402/.
17. RAVICHANDRAN, Adhithi. *React Virtual DOM Explained in Simple English* [online] [cit. 2021-11-21]. Dostupné z : https://programmingwithmosh.com/react/react-virtual-dom-explained/.
18. KROTOFF, Tanguy. *Front-end frameworks popularity (React, Vue, Angular and Svelte)* [online] [cit. 2021-11-23]. Dostupné z : https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190.
19. *React A JavaScript library for building user interfaces* [online] [cit. 2021-11-23]. Dostupné z : https://reactjs.org/.
20. *AngularJS — Superheroic JavaScript MVW Framework* [online] [cit. 2021-11-23]. Dostupné z : https://angularjs.org/.
21. *Vue.js* [online] [cit. 2021-11-23]. Dostupné z : https://vuejs.org/.
22. ORACLE. *What Is a Database?* [Online] [cit. 2021-11-13]. Dostupné z : https://www.oracle.com/database/what-is-database/.
23. MONGODB. *What is NoSQL?* [Online] [cit. 2021-11-13]. Dostupné z : https://www.mongodb.com/nosql-explained.
24. ORACLE. *What is a Relational Database (RDBMS)?* [Online] [cit. 2021-11-13]. Dostupné z : https://www.oracle.com/database/what-is-a-relational-database/.
25. OPEN SOURCE INITIATIVE. *Licenses & Standards* [online] [cit. 2021-11-14]. Dostupné z : https://opensource.org/licenses.
26. RED HAT. *What is a REST API?* [Online] [cit. 2021-11-14]. Dostupné z : https://www.redhat.com/en/topics/api/what-is-a-rest-api.
27. JSON. *Introducing JSON* [online] [cit. 2021-11-13]. Dostupné z : https://www.json.org/json-en.html.
28. MKRTCHYAN, Rafayel. *Wireframe, Mockup, Prototype: What is What?* [Online] [cit. 2021-12-01]. Dostupné z : https://uxplanet.org/wireframe-mockup-prototype-what-is-what-8cf2966e5a8b.
29. *Git* [online] [cit. 2021-12-01]. Dostupné z : https://git-scm.com/.
30. *What is Vuex?* [Online] [cit. 2021-12-01]. Dostupné z : https://vuex.vuejs.org/.
31. *Vue Router* [online] [cit. 2021-12-01]. Dostupné z : https://router.vuejs.org/.
32. *Vuetify* [online] [cit. 2021-12-01]. Dostupné z : https://vuetifyjs.com/en/.
33. *Material Design* [online] [cit. 2021-12-01]. Dostupné z : https://material.io/.
34. *Axios* [online] [cit. 2021-12-01]. Dostupné z : https://github.com/axios/axios.
35. WEI-SIONGTAN; LIU, Dahai; BISHU, Ram. Web evaluation: Heuristic evaluation vs. user testing [online]. 2009, roč. 39. ISSN 0169-8141. Dostupné z DOI: https://doi.org/10.1016/j.ergon.2008.02.012.
36. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users* [online] [cit. 2021-12-01]. Dostupné z : https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `front-end` ..... zdrojové kódy implementácie front-endu
  - `back-end` ..... zdrojové kódy upravenej implementácie back-endu
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF