FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikácia pre podporu výskumu biotopov rýb
Student:	Bc. Lukáš Kozlík
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cieľom diplomovej práce je vytvorenie aplikácie pre výskum rýb na prírodovedeckej fakulte Univerzity Karlovej.

1. Spracujte, vyčistite a zanalyzujte dáta riečnych tokov biotopu rýb.
2. Zozbierajte funčné a nefunkčné požiadavky na aplikáciu podľa užívateľa.
3. Vytvorte uživateľské prípady (Use cases) vrátane príslušných diagramov.
4. Navrhnite architektúru a design aplikácie.
5. Naimplementujte aplikáciu a otestujte ju s koncovým uživateľom.
6. Implementáciu ekonomicky vyhodnoťte a navrhnite ďalší rozvoj.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práca

Aplikácia pre podporu výskumu biotopov rýb

Bc. Lukáš Kozlík

Katedra softwarového inženýrství
Vedúci práce: Ing. Petra Pavlíčková, Ph.D.

6. mája 2019

Poďakovanie

Rád by som sa poďakoval vedúcej práce pani Ing. Petre Pavlíčkovej Ph.D. za vedenie práce a pravidelné konzultácie, vďaka ktorým mohla byť táto práca úspešne zhotovená. Taktiež by som rád vzdal vďaku za celé roky štúdia, počas ktorých bola mojím hlavným pedagógom tohto študijného oboru a priniesla nespočetné množstvo vedomostí a skúseností z praxe.

Druhé poďakovanie patrí Petre Horkej z Ústavu pre životné prostredie Karlovej Univerzity v Prahe. Vďaka zaujímavému zadaniu, ktorého výsledok bude využitý v praxi, bola táto práca pre mňa zábavná a motivujúca. Počas konzultácií s ňou som si vyskúšal nie len komunikáciu pri tvorbe zadania samotnej aplikácie, ale naučil sa aj mnoho informácií z oblasti rybích biotokov.

Na záver sa chcem poďakovať svojej rodine, priateľom a kolegom za podporu nie len počas tvorby diplomovej práce, ale i počas celého štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 6. mája 2019 .....

Odkaz na túto prácu

Kozlík, Lukáš. *Aplikácia pre podporu výskumu biotopov rýb*. Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Diplomová práca sa zaoberá analýzou, návrhom a implementáciou lokálnej webovej aplikácie pre podporu a spravovanie dát výskumu biotopu rýb. Aplikácia by mala slúžit pre tím z Ústavu pre životné prostredie Karlovej Univerzity. Jej cieľom je podporiť jednoduchšie spravovanie dát z výskumu. To obnáša vytvorenie jednotnej štruktúry dát, možnosť pristupovať k čiastočným dátam a export podskupín dát, ako ich predprípravа pre ďalšie analytické nástroje.

Kľúčové slová výskum biotopu rýb, lokálna webová aplikácia, spravovanie dát, návrh, analýza, implementácia, javascript, HTML, UML

Abstract

This master thesis deals with analysis, design and implementation of local web application for support and data management of fish research. The application is created for the team from Institute for Environmental Studies in Charles University. The Application goal is to support better data management from the fish research. It means to create uniform data structure, provide access for data subcategories and offer various export options. The process also includes data preparation for next usage by analytics tools.

Keywords fish habitat research, local web application, data management, design, analysis, implementation, javascript, HTML, UML

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Vedecká sféra pracuje s mnohými dátami. V posledných rokoch využíva na ich spracovanie moderné počítačové technológie, pre zjednodušenie výpočtov, pozorovania, ale aj samotného uskladnenia vedeckých dát. V minulosti boli všetky údaje zapisované ručne do zošitov, poznámkových blokov a ďalších rôznych nie príliš praktických úložísk. Pri pokuse o transformáciu takto uskladnených dát do jednotnej štruktúry sa výskumníci často stretávajú s rôznymi prekážkami.

Toto je prípad tímu pre výskum biotopu rýb pod vedením Petry Horkej z Ústavu pre životné prostredie Karlovej Univerzity v Prahe. Pri pokuse skompletizovania dát a ich predprípravu pre analytické a štatistické nástroje sa stretávajú s problémom dátovej nekonzistencie a náročného spôsobu úpravy dát.

V prípade, že chcú ďalej spracovávať určitú skupinu dát, musia si jednotlivé záznamy povyberať manuálne z historických excel súborov. Následne trávia čas ich úpravou a spájaním do presnej štruktúry, ktorú vyžadajú nástroje, pre ktoré dáta pripravujú.

Táto diplomová práca by mala priniesť riešenie jednotnej štruktúry dát, dynamické vytváranie podkategórii dát a ich samotné zobrazenie. To bude vykonané formou tabuľky, ale aj prostredníctvom grafu. Užitočná bude možnosť rôznych variantov exportu, podľa vstupu užívateľa. Aplikácia bude schopná dáta pridávať, editovať a mazať priamo z webového prostredia.

Motivácia

Hlavnou motiváciou tejto práce bolo vyskúšať si tvorbu aplikácie počas celého procesu vývoja softvéru. Počas štúdia sme v rôznych predmetoch riešili a vytvárali jednotlivé časti. Až táto práca mi dala možnosť nahliadnúť a reálne si vyskúšať proces návrhu a vývoja softvéru ako celku. Ďalším plusom bol fakt, že sa jednalo o skutočný problém v praxi, ktorý bolo nutné vyriešiť. Doteraz boli školské projekty ukážkové a po ich vypracovaní upadli do zabudnutia. Možnosť nasadiť a používať moju aplikáciu v praxi mi dala obrovskú mo-

tiváciu.

Druhá vec, vďaka ktorej som sa rozhodol pre túto tému, bola možnosť prvýkrát samostatne naprogramovať celú webovú aplikáciu ako celok a presne pochopiť ako jednotlivé štruktúry medzi sebou fungujú. Preto som prijal výzvu naučiť sa pre mňa úplne nový programovací jazyk javascript.

Ako bonus som bral skutočnosť, že sa dozviem viac o mne nie veľmi známej biologickej téme odlovu rýb. Pri výbere témy som sa tešil na preniknutie do tohto prostredia a zistenie nových poznatkov.

Štruktúra práce

Prácu som rozdelil na dve časti. Teoretickú, v ktorej popisujem jednotlivé definície a druhy analýzy pri tvorbe softvéru, a praktickú, kde riešim samotný návrh a implementáciu aplikácie.

Prvá, čisto teoretická časť, sa venuje témam procesu vývoja softvéru, funkčných a nefunkčných požiadaviek od zákazníka, prípady užitia, doménový model a ďalšie.

V praktickej časti analyzujem požiadavky zákazníka prostriedkami uvedenými v prvej časti. Následne tu zachytávam samotný návrh a implementáciu aplikácie. Nechýba ani sekcia venovaná jej inštalácii a testovaniu. Nachádza sa tu aj inštalačná príručka pre užívateľov. V závere hodnotím náklady na vývoj a ďalšiu tvorbu aplikácie. Snažím sa tiež načrtnúť jej ďalšie možné využitie a uplatnenie.

Kapitola 1

Cieľ práce

Cieľom práce je navrhnúť a implementovať plne funkčnú lokálnu webovú aplikáciu, ktorá bude schopná plnohodnotne riešiť potreby zákazníka ohľadom uskladnenia, štruktúry a prístupu k dátam. Zároveň je kladený dôraz na uchovanie a anonymitu dát. Samotnému návrhu a implementácii predchádza rozsiahla analýza, v ktorej si spolu so zákazníkom ujasňujeme jednotlivé prvky a funkcie aplikácie pre čo najefektívnejšie použitie a nasadenie v praxi.

Táto práca si dáva za cieľ poskytnúť teóriu k jednotlivým krokom analýzy a poskytnúť tak čitateľovi komplexnejší pohľad. Práca prináša i ekonomické zhodnotenie a ďalší možný rozvoj aplikácie.

Kapitola 2

Teória

Kapitola venovaná teórii metodiky vývoja softvéru. Z existujúcich a rozsiahlych častí a procesov vyberám a vysvetľujem tie, ktoré sú využité v rámci tejto diplomovej práce.

2.1 Proces vývoja softvéru

Základným cieľom softvérového inžinierstva je efektívne vytvárať softvérové produkty. Efektivity dosiahneme znalosťou softvérového procesu, ktorý sa zaoberá otázkou ako sa má správne postupovať, aké fázy a kroky vývoja sú podstatné. Dôraz sa kladie aj na pochopenie toho, čo je ich vstupom a výstupom. Pri tvorbe softvéru existuje niekoľko rolí. Keďže táto práca je samostatná a zastupujem takmer všetky z nich, nebudem sa zaoberať detailnejším rozborom rolí pri vývoji softvéru.

Fázy tvorby softvéru

- Zber požiadaviek
- Analýza
- Návrh
- Implementácia
- Testovanie

Podľa štatistík je percentuálne rozdelenie časového rozloženia nasledujúce: 2.1.

Analýza	40%
Návrh	40%
Implementácia	20%

Tabuľka 2.1: Časová náročnosť tvorby softvéru

2.2 Zber požiadaviek

Pre úspešné dokončenie nielen softvérových projektov je kľúčové správne pochopenie očakávaní zákazníka. Táto špecifikácia podrobne pojednáva o tom, čo bude výsledný systém riešiť, zvyčajne zoznamom jednotlivých požiadaviek. Nezaoberá sa však ešte návrhom a implementáciou systému.

Pri jednotlivých požiadavkách je dobré písať aj prioritu. Vieme sa podľa nej následne so zákazníkom dohodnúť, čo bude v prvej verzii systému, čo v ďalších, a čo vôbec. Prípadne vieme čo vypustiť, ak sa dostaneme do časového sklzu s dodaním systému [1].

Po obsahovej stránke sa požiadavky delia na [2]:

- **Funkčné:** Pojednávajú priamo o tom, čo má systém robiť.
- **Nefunkčné:** Sú to požiadavky, ktoré sa netýkajú toho, čo má systém robiť. Typicky hovoria o parametroch systému.

Dôležitou úlohou požiadaviek je spojiť a upresniť predstavy medzi zákazníkom a vývojom. Funkčným a nefunkčným požiadavkám sa venujem aj pri tvorbe mojej aplikácie 3.4, 3.5.

2.3 Analýza

Analýza je priamo napojená na zber požiadaviek a ich bližšiu špecifikáciu. Slúži pre ujasnenie a detailnejšie rozobratie požiadaviek. Vďaka nej často zistíme a odhalíme chýbajúce požiadavky. Pre analýzu využívame tvorbu modelov s využitím grafických prvkov. Tie uľahčujú zákazníkovi pochopiť modelovaný scenár, upraviť ho, či prípadne doplniť. K vytváraniu diagramov využíva UML, ktorého význam si rozoberieme v tejto sekcii [3].

Diagramov pre analýzu je nespočetné množstvo. Uvediem a rozoberiem len tie, ktoré využívam vo svojej práci.

2.3.1 UML

UML je jazyk využívaný pre špecifikáciu, vizualizáciu a konštrukciu prvkov softvérového systému. Svoje využitie nájde aj v dokumentácií. Je využitelný aj pre biznis modelovanie a modelovanie nesoftvérových systémov. Je v ňom možné namodelovať akýkoľvek typ aplikácie bežiacej na každom HW. Jeho

hlavnými prínosmi sú zrozumiteľnosť, rýchla a jednoduchá kontrola a znovupoužiteľnosť. UML diagramy nám dokážu uľahčiť odhadovanie [4].

UML rozdeľuje diagramy na 2 druhy [5]:

- **Diagram správania:** Diagram aktivít, prípadov užitia, sekvenčný a ďalšie.
- **Diagram štruktúry:** Diagram tried, komponent, nasadenia a balíčkov.

V tejto práci budem využívať hlavne diagram aktivít pre modelovanie procesov systému, diagram prípadov užitia a diagram nasadenia pre lepšie pochopenie prepojenia jednotlivých komponent systému.

2.3.2 Model procesov v aplikácii

Modelovanie procesov má svoj význam. Pomáha pochopeniu činnosti zákazníka, dokáže lepšie identifikovať problémové miesta a zlepšuje samotné procesy bez ohľadu na implementáciu. Vďaka nemu môžeme presnejšie špecifikovať požiadavky a docieliť tak lepšiu podporu procesov v navrhnutej aplikácii.

Pod pojmom proces rozumieme sadu usporiadaných aktivít, ktoré transformujú vstupy na výstupy a zachytávajú role zodpovedné za jednotlivé aktivity. Sledovanými informáciami sú dôležitosť procesu, početnosť prevedenia, časová a finančná náročnosť. Spôsoby zachytenia sú textové alebo prostredníctvom diagramu[6].

2.3.3 Diagram aktivít

Pre modelovanie procesov v mojej aplikácii využijem diagram aktivít. Ten je jeden z UML diagramov, ktoré popisujú správanie. Tento typ diagramu sa používa pre modelovanie procedurálnej logiky, procesov a zachytenie workflow [7]. Sú výhodné vďaka možnosti paralelizácie procesov.

Každý proces v diagrame aktivít je reprezentovaný sekvenciou jednotlivých krokov, ktoré sú v modeli zobrazené ako akcie. Pod pojmom akcia môžeme rozumieť činnosť, ktorá je aktívne vykonávaná vnútri aktivity. Aktivitou v tomto prípade rozumieme ako to, čo je modelované pomocou diagramu aktivít, teda proces.

Akcia môže byť vykonávaná človekom v určitej roli, alebo systémom. V diagrame aktivít sú jednotlivé systémy a role znázornené ako oblasti, čiary označené menom. Následne sa do každej oblasti kreslí akcia vykonaná systémom alebo človekom. Toto rozdelenie na človeka a systém využívam aj vo svojej práci. Pomáha mi lepšie pochopiť, ktoré akcie musí vykonať užívateľ, a ktoré vykonáva systém na pozadí.

2.3.4 Model prípadov užitia

Zachytáva ho diagram prípadov užitia. Ten zobrazuje vonkajší pohľad na systém. To znamená, že tento diagram popisuje systém tak, ako ho vidí aktér [8].

Diagram prípadov užitia obsahuje tieto časti:

- **Prípady užitia:** Každý prípad užitia charakterizuje určité použitie systému aktérom. Je označený menom a môže mať ďalšiu špecifikáciu.
- **Aktéri:** Reprezentujú kohokoľvek mimo systém, kto so systémom interaguje. Jediné čo aktéri môžu, je pridávať alebo dostávať informácie zo systému. Aktér vo väčšine prípadov nereprezentuje jednotlivca, ale celú rolu.
- **Ohraničenie:** Udáva hranice systému prípadne subsystému.
- **Relácie a vzťahy:** Definujú vzťahy medzi prípadmi užitia a samotnými aktérmi.

V tejto diplomovej práci využijem aj špeciálne vzťahy medzi prípadmi užitia a to „include“ a „extend“. V prvom prípade sa jedná o situáciu, kedy jeden prípad užitia môže obsahovať iný. Druhý prípad hovorí o rozširovaní jedného prípadu užitia druhým.

2.3.5 Doménový model

Poslednou časťou analýzy bude vytvorenie doménového modelu. Ten sa vytvára spolu s diagramom užitia. Jedná sa o formu diagramu tried. Triedy v doménovom modeli sú však značne zjednodušené, neobsahujú metódy a majú iba dôležité atribúty. Doménový model je teda akýsi náčrt základných entit systému a vzťahov medzi nimi.

Tento model sa skladá z 3 častí:

- **Entity:** Mapuje triedy, ktoré zastupujú objekty z problémovej oblasti bez implementačných detailov.
- **Atribúty:** Bližie špecifikujú typy v doméne. Neobsahujú detaily. Nedozvedáme sa ani dátové typy jednotlivých atribútov.
- **Vzťahy:** Zobrazujú násobnosti relácii medzi entitami.

Doménový model špecifikuje viacero druhov vzťahov medzi entitami. V tejto práci však budem využívať iba jednu základnú entitu, preto jednotlivé vzťahy nebudem ďalej rozoberať.

2.4 Návrh

Analýza priamo prechádza do návrhu. Otázky, čo sa má implementovať, sa menia na otázky, akým spôsobom implementovať. V tejto časti sa definuje konkrétna architektúra systému. Dôležité je aj navrhnúť správny spôsob uloženia dát a vytvorenie návrhového modelu tried. Je nutné zvoliť implementačný jazyk a použité frameworky.

2.4.1 Architektúra

V logickej časti obsahuje architektúra organizáciu softvérových tried do balíčkov. Tie potom usporadúva do vrstiev a podsystémov.

Vo fyzickej časti sa zaoberá rozložením komponent na výpočetné uzly. Fyzickú architektúru zobrazuje diagram nasadenia.

Existuje niekoľko druhou architektúry. Väčšinou sa rozlišujú vrstvy architektúry. Venovať sa budem iba dvojvrstvovej architektúre, ktorú vo svojej práci využívam. Definujem ju v ďalšej sekcii 2.4.3.

2.4.2 Diagram nasadenia

Diagram nasadenia ukazuje rozmiestnenie zdrojov(HW) a softvérové komponenty, procesy a objekty, ktoré na nich žijú. Využíva sa k špecifikácii fyzickej architektúry systému.

Základom diagramu sú uzly, ktoré sú vzájomne prepojené komunikačnými cestami. Môžu obsahovať artefakty, ktoré predstavujú fyzický výskyt softvéru.

2.4.3 Dvojvrstvová Architektúra

Architektúra klient-server označuje v informatike jeden z typov architektúry informačných systémov. Jedná sa o dvojvrstvovú architektúru. Klient obsahuje užívateľské rozhranie a aplikačnú logiku. Na serveri beží relačná databáza [9].

V praxi sa definuje tenký a tučný klient nasledovne:

- **Tenký klient:** Počítač alebo program, ktorý pri plnení svojej úlohy závisí na inom počítači, serveri. Konkrétne úlohy, ktoré môže server plniť, sa môžu líšiť. Rozsah úloh je od poskytovania dátového úložiska až k samotnému spracovaniu informácií namiesto klienta. Stará sa o zobrazovanie dát užívateľovi a transformáciu jeho požiadaviek na server.

- **Tučný klient:** Obvykle v sebe obsahuje ako prezentačnú, tak i aplikačnú vrstvu a pripája sa priamo k databázovému alebo inému serveru [10].

2.4.4 Spôsob ukladania dát

Ďalšou dôležitou časťou je návrhový model tried. Vychádza z doménového jazyka a rozširuje ho o dátové typy atribútov, upresnenie relácii a pridáva nové softvérové triedy.

V súvislosti so správnym uložením dát si musíme zadefinovať databázový model. Ten popisuje uloženie dát v relačnej databáze, generuje zakladajúce SQL skripty a obsahuje dátové typy zvolenej aplikácie.

2.4.5 Implementačný jazyk

Existuje široká paleta implementačných jazykov a prostredí. Pri výbere vhodného programovacieho jazyka je nutné vziať do úvahy viacero kritérií:

- **Obecné:** Rýchlosť implementácie, pohodlie a možnosti vývojového prostredia, cena a licenčné podmienky.
- **Špecifické:** Aké vlastnosti by mali mať systémy a vlastnosti, ktoré sú ovlpyniteľné jazykom.

Z pohľadu zákazníka by sa malo jednať o stabilný a robustný softvér, ktorý bude schopný fungovať spoľahlivo a bez chýb. Taktiež by mal umožňovať rýchlu detekciu, analýzu a opravu problémov.

Z pohľadu vývojára je dôležitá dlhodobá udržateľnosť, čitateľnosť kódu, modularita, či automatické odhaľovanie chýb [11].

Pred začiatkom vytvárania tejto práce som taktiež zvažoval jednotlivé programovacie jazyky a snažil sa vybrať najvhodnejší. Hlavným kritériom bola aspoň čiastočná znalosť vybraného jazyka. Nechcel som sa učiť daný jazyk od základu, ale nadviazať na znalosti, ktoré mám. Zároveň to mal byť jazyk vhodný a odporúčaný na tvorbu webových stránok. Nakoniec som sa rozhodol pre využitie javascriptu. Jeho hlavné výhody a dôvody tohto rozhodnutia popisujem v samostatnej sekcii 5.2.

2.5 Testovanie

Naimplementovaním aplikácie sa proces vývoja softvéru nekončí. Je totiž potrebné zabezpečiť jeho kvalitu. To zahrňuje overovanie naplnenia požiadaviek zákazníka a kontrolu zhody so zadaním a špecifikáciou projektu. Je nutná aj kontrola výstupov vývojárov zameraná na dodržiavanie vopred definovaných štandardov a noriem. Testovanie sa často vykonáva za účelom hľadania chýb všetkých druhov: funkčných, logických, obsahových, aj systémových.

Všetky tieto kroky platia aj pre webové riešenia, a teda aj mnou vytvorenú aplikáciu. Špecifická je hlavne časť testovania, ktorá si vyžaduje prihliadať na skutočnosť, že sa jedná o testovanie webovej stránky a prispôsobiť tomu výber vhodných testov.

Samotné testovanie je systematický proces pozorovania správania sa systému v špecifických podmienkach, ktoré simulujú reálne prostredie. Dôležité je jednotlivé zistenia zaznamenať a vyhodnotiť. Hlavným cieľom testovania je teda vyhľadať maximálny možný počet chýb v čo najkratšom čase, na čo najnižšej úrovni vývoja riešenia [12].

Testovanie sa delí podľa viacerých kritérií:

Podľa spôsobu prevedenia je možné využiť **statické** testovanie, ktoré vôbec nevyžaduje spustenie programu. V mojej práci mi postačí testovanie **dynamické**, pri ktorom využijem spustiteľný kód.

Na základe znalosti vnútornej štruktúry je časté využitie **White Box** testovania. To vyžaduje znalosť zdrojového kódu a tým umožňuje lepšie identifikovať možné problémy. **Gray Box** testovanie ťaží zo znalosti použtých algoritmov bez informácií o danej implementačnej metóde. Táto práca využije **Black Box** testovanie, ktorým overím správanie sa aplikácie na jej rozhraní. Užívatelia, ktorí túto činnosť vykonajú, nebudú poznať vnútornú štruktúru kódu.

Aplikácia pre podporu výskumu rýb bude podľa spôsobu vyhodnocovania využívať manuálne testy užívateľov. V tomto prípade automatizácia testovacieho scenára nie je nutná.

Najzaujímavejším rozdelením je členenie testov podľa rozsahu [13]:

- **Jednotkové testy:** Testujú sa jednotlivé triedy samostatne. Tieto testy sú zvyčajne lokálne spustené na počítači vývojára. To bolo i v prípade tejto práce, kedy som využíval jednotkové testy na overenie funkčnosti jednotlivých častí kódu priamo pri ich tvorbe.

- **Testy komponent:** Testujú sa izolované komponenty. Od jednotkových testov ich oddeľuje skutočnosť, že testujú väčšie časti funkčností. Proces vývoja samotnej aplikácie obsahoval i tento druh testovania.

- **Integračné testy:** majú za úlohu overiť spoluprácu komponent. Simulujú reálne prostredia. Typickým príkladom testu je pripojenie k databáze. Integrácia s databázou pre uloženie dát bola jednou z hlavných častí tejto práce. Dôležité bolo pripraviť vhodné dáta a obnoviť stav databázy pred samotných testovaním.

- **Systémové testy:** Testujú aplikáciu ako celok na Black Box princípe. Využívajú sa scenáre modelu prípadov užitia a testovanie je častokrát aj na samotnom užívateľovi. Práve tomuto typu testu venujem celú kapitolu 6.

2.5.1 Akceptačné testy

V tejto fáze životného cyklu procesa vývoja softvéru je potrebné detailne otestovať funkčnosť celej aplikácie na základe požiadaviek zákazníka.

Súčasťou tejto fázy je aj zaškolovanie budúcich používateľov aplikácie. V mojom prípade pôjde o predstavenie aplikácie zadávateľke a jej najužšiemu tímu. Bude potrebné im podrobne ukázať ako navrhnutú aplikáciu správne použiť.

Akceptačné testovanie sa odlišuje v tom, že testujeme za prítomnosti zákazníka. Cieľom je overiť a dokázať zadávateľovi, že aplikácia funguje podľa požiadaviek, ktoré boli formulované a odsúhlasené.

Výsledkom akceptačného testovania má byť spokojnosť zadávateľa s preberanou aplikáciou.

Akceptačné testovanie môžeme rozdeliť do troch etáp:

- **Vlastné akceptačné testy:** Pre možnosť začať prevádzať akceptačné testy je potrebné mať pripravenú konfiguráciu systému, nainštalovanú aplikáciu a špecifikáciu akceptačných testov. Dôležitou súčasťou prípavy je zhotovenie testovacích dát, prípadne testovacích skriptov.

- **Zaškolovanie používateľov:** Je vhodné, aby akceptačné testovanie vykonával akceptačný tím spolu s budúcimi používateľmi aplikácie. Týmto spôsobom sa budúci užívatelia zoznámia s novým systémom. Zároveň budú mať možnosť priamo konzultovať niektoré kroky priamo s tvorcami systému, ktorí sa nachádzajú v testovacom tíme. Účastníci školenia musia byť evidovaní a musí byť vedená evidencia o obsahu školenia a účasti používateľov.

- **Odovzdávanie aplikácie zákazníkovi:** Po vykonaní akceptačných testov zadávateľ rozhodne, či nájdené chyby a nedostatky umožňujú formálne prebrať hotovú aplikáciu. Každá z chýb musí mať stanovenú prioritu riešenia a náročnosť na jej odstránenie. Tieto chyby sa uvedú v odovzdávacom a preberacol protokole, prípadne v prílohe. Ak pri odovzdávaní aplikácie zákazníkovi existujú nevyriešené chyby, súčasťou preberacieho protokolu sú aj podmienky dohodnuté pre ich odstránenie.

Vo fáze akceptačného testovania aplikácie vznikajú rôzne dokumenty:

- **Protokol o akceptačnom testovaní:** Dokument je vytváraný prevádzkovateľom systému v súlade s plánom akceptačného testovania. Dokument musí obsahovať informácie o priebehu a výsledkoch akceptačného testovania. Tento protokol musí obsahovať časti opisov prostredia a zdrojov, kde boli akceptačné testy realizované, opisom aktivít užívateľov a opisom jednotlivých testov spolu s ich hodnotením. Na záver je nutné dodať celkové zhodnotenie splnenia alebo nesplnenia akceptačných kritérií pre systém.

- **Protokol o zaškolení používateľov systému:** Dokument vypracúva riešiteľ systému a schvaľuje ho zadávateľ. Obsahuje zoznam použitých

materiálov pre školenie, hlavne užívateľské príručky. Opisuje rozsah a obsah školenia a zoznam účastníkov. Súčasťou je aj zhodnotenie výsledkov skúšobnej prevádzky užívateľmi.

- **Odovzdací a preberací protokol:** Jedná sa o dokument, ktorým sa odovzdáva vytvorená aplikácia alebo systém zákazníkovi. Obsahuje miesto, termín a osoby zúčastnené na preberacom konaní, predmet dodávky, zoznam magnetických a optických médií a informáciu o uložení programových produktov na nich. Sú v ňom uvedené identifikované chyby počas preberania a spôsob ich následného odstránenia.

Akceptačné testovanie začína v plánovanom alebo inak dohodnutom termíne. Predpokladom je úspešné ukončenie integračných testov [14].

2.5.2 Tester

Aplikáciu budem finálne testovať so zákazníkom. Pred tým potrebuje skupinu ľudí na prvotné otestovanie. Je dôležité zvoliť správne človeka na vykonanie testov aplikácie.

Je podstatné, aby mal vybraný tester čo najviac z výberu nasledujúcich vlastností [15]:

- **Konštruktívne myslenie:** Je veľkou výhodou, ak sa tester dokáže v krátkom čase zorientovať v aplikácii, s ktorou pracuje. Dôležité je pochopiť jej funkcie na čo využije svoju logiku.
- **Znalosť programovacích techník:** Určité druhy testovania si vyžadujú písanie automatických testov. Nevyhnutná vlastnosť dobrého testera je schopnosť algoritmizácie a ideálne aj vývojárske skúsenosti.
- **Zvedavosť:** Testovanie je o kladení otázok, zbieraní a vyhodnocovaní odpovedí. Bez investigatívneho prístupu by sa dobrý tester nezaobišiel.
- **Kreativita:** Kreatívni testeri sa líšia od ostatných tým, že skúšajú v záujme vlastnej zvedavosti nové testovacie postupy. Sú schopní vytvoriť takú kombináciu vstupov, ktorá by nemusela priniesť očakávané výsledky. Sú autormi mnohých nápadov a zlepšovacích návrhov.
- **Pozornosť:** Testerovi by nemal uniknúť rozdiel medzi špecifikáciou a realitou. Mal by byť vnímavý nielen na chyby, ale aj na ich príznaky.

Kapitola 3

Analýza

V tejto kapitole sa zaoberám analýzou pre vývoj výslednej webovej aplikácie. Na začiatku predstavím typ a charakter zákazníka. Následne sa zaoberám pôvodom a štruktúrou dát, ktoré sú najpodstatnejšie pri vytváraní konceptu a návrhu samotnej aplikácie. So zákazníkom si ujasňujem funkčné a nefunkčné požiadavky podľa jeho zadania. Okrem iného, táto kapitola obsahuje aj rozbor a konkrétne modely prípadov užitia.

3.1 Zákazník

Táto práca vzniká ako spolupráca ČVUT a Univerzity Karlovej v Prahe. Týka sa konkrétne oddelenia Ústavu pre životné prostredie.

3.1.1 Ústav pre životné prostredie

Ústav pre životné prostredie je celofakultným interdisciplinárnym pracoviskom Prírodovedeckej fakulty Univerzity Karlovej, ktorého náplňou je výuka a výskum v oblasti životného prostredia. Ústav pre životné prostredie poskytuje vysokoškolské vzdelanie bakalárskej, magisterskej a doktorantskej úrovne. Technické vybavenie laboratórií umožňuje výuku a výskum v mnohých oblastiach ochrany životného prostredia. Zameriava sa hlavne na ovzdušie, vodu, pôdu, ochranu prírody a aplikovanie geografických informačných systémov.

V súčasnej dobe ústav spolupracuje s množstvom univerzitných pracovísk, verejne dostupných ústavov a ústavom Akadémie vied ČR. Spoluprácu tvorí aj so zahraničnými inštitúciami predovšetkým prostredníctvom spoločných výskumných projektov a v rámci diplomových a dizertačných prác.

3.1.2 Petra Horká

Petra Horká pôsobí ako odborný asistent na Ústave pre životné prostredie Karlovej Univerzity. V rámci svojej práce sa zaoberá výukou (predmety Lim-

nológia, Moderné ichtyologické metódy, Environmental Issues - background examples and solutions, Environmentálne metódy a ďalšie), školením študentov a vedeckou prácou.

Zaoberá sa ekológiou rýb a spoločenstvom organizmov v tekutých vodách. Jej hlavné zameranie sú trofické vzťahy v spoločenstvách, chov rýb a vplyv antropogénnych faktorov na spoločenstvá rýb v riečnych tokoch. Časť jej výskumu prebieha v teréne a je ďalej spracovaná v laboratóriu (napríklad, stanovovanie trofickej úrovne pomocou stabilných izotopov C a N, alebo behaviorálne experimenty).

Celá diplomová práca je venovaná Petre Horkej a jej najužšiemu tímu. Preto po zvyšok práce bude pojem **Zákazník** označovať túto skupinu ľudí.

3.1.3 Využitie aplikácie pre podporu výskumu

V súvislosti s výskumom dochádza ku generovaniu dát, ktoré musia byť presne zaznamenané a roztriedené pre následné využitie v štatistickej analýze (program R, Canoco, SAS). Hlavne k terénnemu zberu dát sa neskôr pridávajú výsledky stanovenia fyzikálno-chemických parametrov, ktoré sú stanovované v laboratóriach a získané od ďalších spolupracujúcich výskumných inštitúcií.

Prvotný súbor dát, ktorý vstupuje do aplikácie, sú dáta z viac ako 30 rokov odlovu rýb v teréne a to hlavne riečnych tokov na území Česka a Slovenska. Dáta z terénnych odlovov sú obvykle prezentované ako zloženie spoločenstva rýb pod pojmom abundancia. Tá vyjadruje početnosť a biomasu rýb v špecifických lokalitách. Taktiež prináša doplňujúce údaje o dátumoch odlovu, konkrétnom mieste odlovu so špecifikáciou lokality, morfologickým stavom toku, vzdialenosti tokov od prameňa, prietok a mnohé ďalšie parametre. Štruktúrou pôvodných dát sa zaoberám v nasledujúcej sekcii 3.2.

Mnou navrhnutá a vytvorená aplikácia by mala slúžiť hlavne k:

- **Usporiadaniu dát**, ktoré vznikajú ako výsledky terénnych prác a laboratórnych meraní.
- **Výstupu dát vo formáte**, ktorý je požadovaný pre štatistické analýzy.
- **Doplneniu dát** výsledkov terénnych odlovov o výsledky stanovenia fyzikálno-chemických parametrov.

3.2 Zdrojové data

Výskum, pre ktorý je táto aplikácia vytvorená, pracuje s dátami odlovu rýb počas niekoľkých rokov. Obsahujú presné počty odchytených druhov rýb, lokalitu a zložky vody, v ktorých bol daný odlov vykonaný. Presnosť a dôležitosť dátam dodáva skutočnosť, že informácie o lokalite uvádzajú presný riečny kilometer, prietok vody, abundanciu, presný popis miesta odlovu a veľa ďalších informácií potrebných pre úspešný výskum.

Tok		po	pd	krc	nar	dn	
Váh	Jan Čaták	20ks	3	nic	2	28ks	8g/mg
Váh 2	Čat ja.	23	nula	3	4ks	7	3hk
Vltava		34	2	32	4	nula	H20- 40
Duna. úst	Ivanovič	33	nic	2	34		
Horný tok	Labe 5km	46v2007	33 v 2007	67	43	76ks	78/mg

Tabuľka 3.1: Príklad nekonzistentných dát

Tieto dáta vytvárali rybári, ktorí dokumentovali svoje dlhoročné pôsobenia v oblasti rybolovu a starostlivosti o riečne toky. Väčšina záznamov pochádza z obdobia, kedy IT technológie neexistovali. To má za následok absolútnu nekonzistenciu dát, ktorých úložisko môžeme charakterizovať nasledovne:

- Dáta zo zápisníkov rybárov, ktoré boli ručne prepísané do Microsoft Excel.
- Dáta zapísané priamo do Microsoft Excel bez legendy a štruktúry.

Príklad nekonzistentného formátu zobrazuje tabuľka 3.1. V skutočnosti zobrazuje informácie o názve a presnej lokalite toku, autorovi odlovu, počte odlovených kusov jednotlivých druhov a zložkách látok vo vode. Tieto dáta nie je jednoduché upraviť, kedže každý súbor obsahuje absolútne odlišnú štruktúru. Súborov obsahujúcich takto roztrúsené, neusporiadané dáta je niekoľko desiatok. Ako prvé je preto nevyhnutné nájsť spoločnú a jednotnú štruktúru pre dané dáta. Tá by mala byť schopná zoceliť historicky zozbierané dáta a zároveň byť predpokladom pre budúce dopĺňanie záznamov.

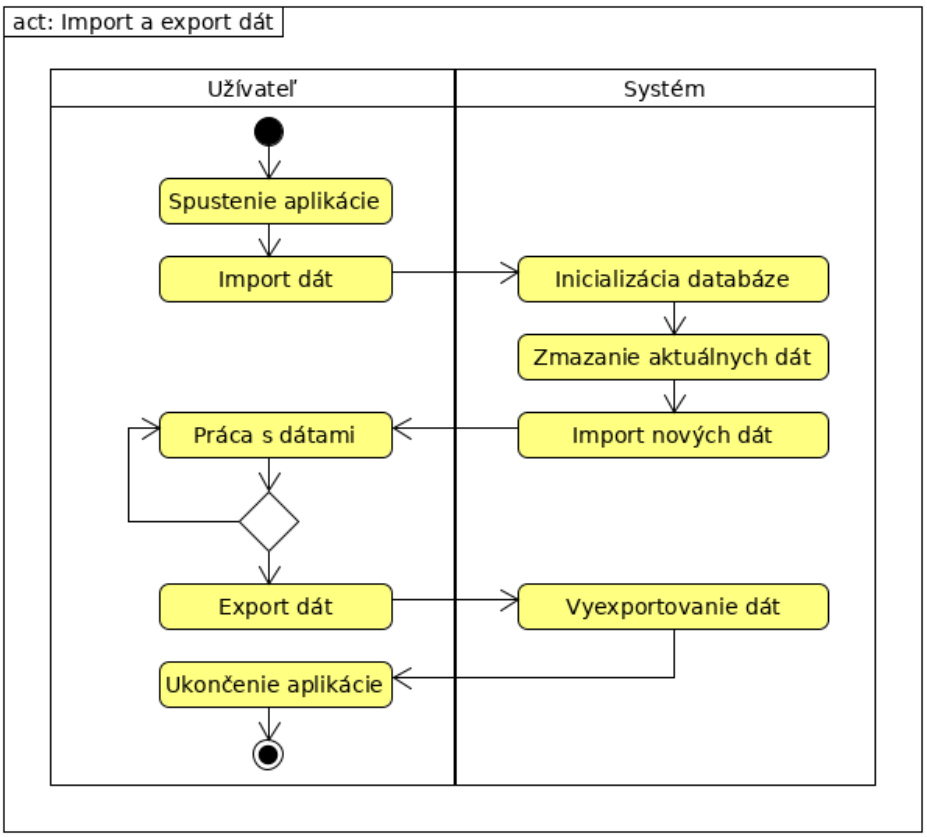
3.2.1 Hodnota Dát

Všetky dáta odlovov potrebné pre výskum sú jedinečné. Neexistuje oficiálne úložisko, kde by boli k dispozícii pre širokú verejnosť ani odborných nadšencov. Ich hodnota sa pohybuje v tisícoch eur a zaobchádzanie s nimi podlieha utajeniu. To je dôvod, prečo táto práca obsahuje iba ukážkové, s realitou sa nezhodujúce dáta. Ich cieľom je prezentovať aplikáciu pre daný výskum, no zároveň uchovať hodnotu a autenticitu záznamov.

3.3 Analýza procesov v aplikácii

Táto sekcia zachytáva hlavné procesy, ktoré podľa požiadavky zákazníka musí aplikácia spĺňať.

Procesy, ako aj funkčné a nefunkčné požiadavky v kapitolách 3.4 a 3.5 boli zozbierané počas viacerých osobných stretnutí so zákazníkom. Finálne



Obr. 3.1: Proces importu a exportu dát užívateľom.

a detailné požiadavky boli následne upresnené prostredníctvom emailovej komunikácie.

3.3.1 Import a Export dát

Základným procesom, ktorý musí aplikácia bezpodmienečne vykonávať, je nahranie dát do aplikácie. Zákazník z dôvodu utajenia dát trvá na uložení v lokálnom systéme, ktorý popisuje sekcia 5.1. Z tohto dôvodu je nevyhnutné dáta do systému importovať a po skončení práce exportovať. Túto rolu má na zadpovednosť užívateľ, ktorý momentálne aplikáciu využíva. Proces importu a exportu dát zachytáva diagram 3.1.

3.3.2 Pridanie, editácia a vymazanie dát

Druhým, pre zákazníka významným procesom pri používaní aplikácie je možnosť pridať, editovať a prípadné vymazať záznamy v reálnom čase priamo

z prostredia aplikácie.

Pri pridaní záznamu užívateľ odošle dáta do systému. Ten si ich následne uloží do databázy a pripraví ich na prípadnú požiadavku užívateľa o editáciu. V prípade žiadosti zmazania záznamu zo strany užívateľa, systém trvalo vymaže dáta zo svojej databázy. Proces pridania, editácie a zmazania dát zachytáva diagram 3.2. Užívateľ ma na výber 3 možnosti:

- **Pridať nový záznam:** Užívateľ počas tohto procesu zadá dáta do pripraveného webového formulára. Po potvrdení sa záznam uloží do databázy a bude mu priradené príslušné identifikačné číslo ID.

- **Editovať existujúci záznam:** V prípade žiadosti o úpravu, musí užívateľ zadať ID záznamu, ktorý chce upraviť. V prípade, že daný záznam neexistuje, má možnosť zadať iné ID. Pokiaľ záznam existuje, zobrazí sa užívateľovi prostredníctvom webového formulára, ktorý môže editovať. Potvrdením akcie sa vykoná následný update v databáze, kde zostanú uložené editované dáta. Pôvodné hodnoty sa týmto krokom vymažú.

- **Zmazať existujúci záznam:** Ak chce užívateľ záznam zmazať, musí si ho nájsť podľa ID rovnako, ako v prípade editácie. Rozdiel je v tom, že po potvrdení akcie bude záznam vymazaný z databázy.

3.3.3 Select dát a ich export

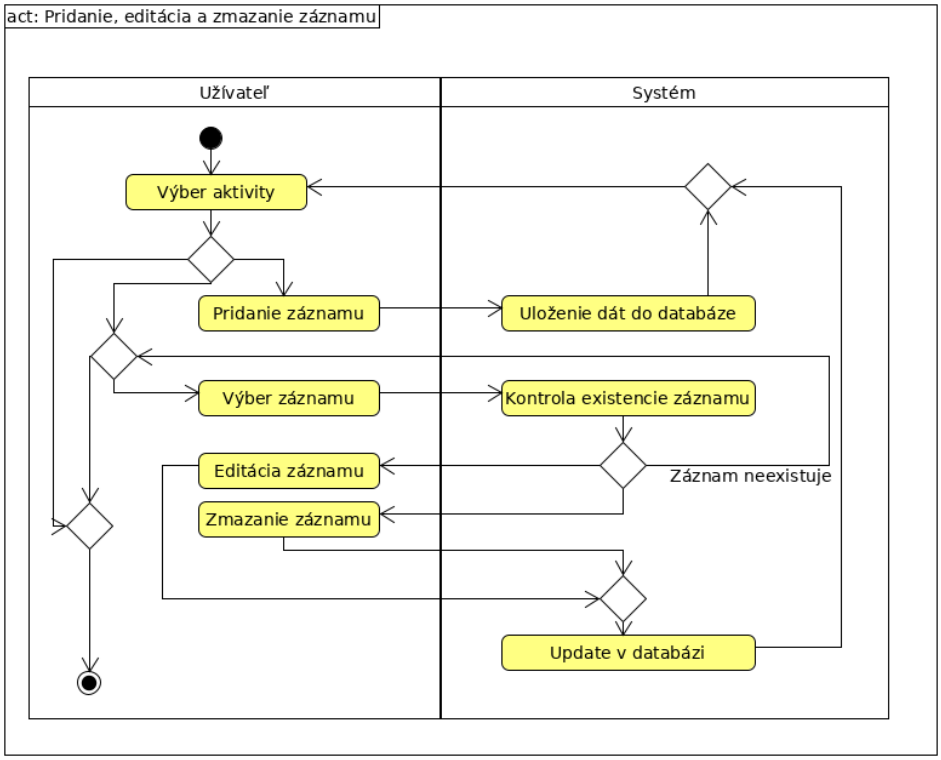
Proces zobrazenia konkrétnych dát na základe používateľa a následný export tejto podkategórie dát je jednou z hlavných požiadaviek zákazníka. Zároveň sa tak aplikácia odlišuje od štandardných nástrojov pre načítanie a prácu s dátami. Proces výberu špecifickej skupiny dát zobrazuje diagram 3.3. Užívateľ si pri výbere zvolí len tie stĺpce z databázy, ktorých hodnoty chce vidieť zobrazené v aplikácii, či prípadné následné vyexportovanie. Musí si vybrať minámalne jeden, aby sa mu dáta zobrazili. Následnou požiadavkou zákazníka na tento proces je dobrovoľné vyplnenie zložitejšej podmienky SQL pre výber konkrétnych dát.

Po zadaní validných podmienok a stĺpcov pre výber má užívateľ na výber 3 možnosti:

- **Exportovať zobrazené dáta:**Zadaním názvu súboru si užívateľ exportuje vybrané dáta do csv formátu.

- **Zadať nový select dát:**V prípade, že select je čisto informatívny a užívateľ nemá záujem o export, môže zadávať nové podmienky pre výber dát.

- **Ukončiť tento proces.**



Obr. 3.2: Proces pridania, editácie a zmazanie záznamu užívateľom.

3.3.4 Grafy

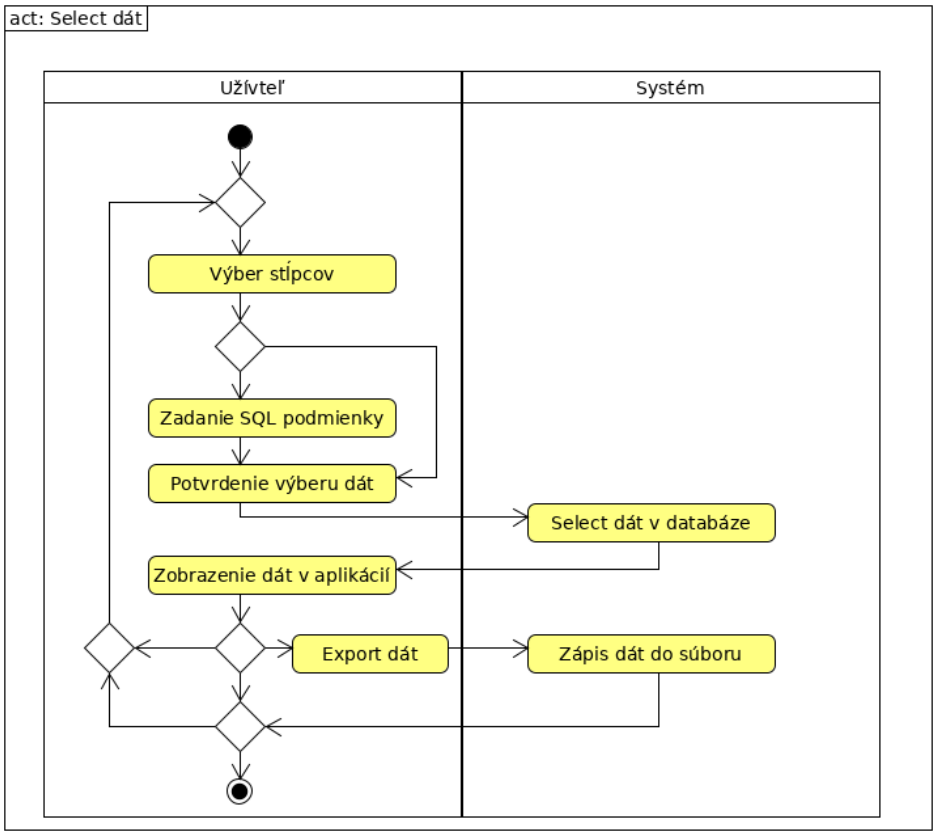
Doplnkovým procesom požiadavky zákazníka je zobrazenie špecifických grafov na základe parametru užívateľa. Ten zadá parametre prislúchajúce konkrétnemu typu grafu. Systém na základe nich vyberie dáta, vytvorí graf a zobrazí ho v aplikácii pre užívateľa. Pre triviálnosť tohto procesu sa tentokrát zaobídem bez diagramu.

3.4 Funkčné požiadavky

V tejto časti sa zamieriam na jednotlivé funkcie aplikácie. Tie vychádzajú a boli upresnené pomocou analýzy procesov 3.3.

3.4.1 Pridávanie dát

Prostredníctvom aplikácie bude mať užívateľ možnosť pridať nové dáta manuálne, vyplnením webového formulára. Pri vytvorení nového záznamu sa mu automaticky vygeneruje ID, ktoré bude reprezentovať jeho jedinečnú hodnotu



Obr. 3.3: Proces výberu a exportu čiastočných dát.

a bude následne použité pri vyhľadávaní záznamu za účelom editácie, zmazania alebo selectu.

3.4.2 Editácia dát

Užívateľ bude môcť priamo pristúpiť do databázy so zámerom zmeny jednotlivých záznamov. Táto zmena sa vykoná na základe ID daného záznamu s nasledujúcou reakciou:

- **ID Existuje:** V prvom prípade sa zobrazia hodnoty daného záznamu. Užívateľ má možnosť ich ľubovoľne meniť. Jedinou nemennou hodnotou je práve ID, ktoré nie je možné manuálne zmeniť z webovej aplikácie. Po ukončení zmeny dát ich užívateľ jedným kliknutím nahrá do databázy, čím prepíše aktuálne hodnoty daného záznamu.
- **ID Neexistuje:** V tomto prípade, kde ID nebolo nájdete a záznam teda neexistuje, je užívateľ o tomto stave informovaný a má možnosť zadať

novú hodnotu pre vyhľadanie záznamu.

3.4.3 Zmazanie dát

Pre odstránenie konkrétneho záznamu ho musí užívateľ vyhľadať pod jeho ID. Pokiaľ takýto záznam neexistuje, je užívateľ o tomto stave informovaný a má možnosť zadať nové vyhľadávanie.

V prípade existencie záznamu sa otvorí v stave pre editovanie, kde bude pre užívateľa možnosť jedných kliknutím záznam odstrániť.

Kedže aplikácia funguje ako import/export dát a pracuje s ich zálohou u zákazníka, je možné v aplikácii mazať záznamy iba jednotlivo po vyhľadaní a nie skupinovo. V prípade potreby zmazania väčšej skupiny záznamov si zákazník vyexportuje dáta do svojho prostredia mimo aplikácie, dáta si vlastnými nástrojmi vymaže a importuje zmenený dataset, s ktorým chce pokračovať v práci.

3.4.4 Import dát

Užívateľ si vždy po spustení aplikácie nahrá dáta, s ktorými chce pracovať. To neplatí v prípade, že dáta už v aplikácii má a chce pokračovať v práci s nimi.

Pri samotnom importe sa všetky aktuálne dáta vymažú a budú nahradené dátami novými. Ten funguje takým spôsobom, aby užívateľ mohol jednoducho nahrať vstupný súbor výberom z umiestnenia na lokálnom disku svojho zariadenia. Následným spôsobom svoj import potvrdil kliknutím na príslušné tlačítko.

3.4.5 Export dát

Export je súčasťou funkcie select popísanej nižšie 3.4.6. Užívateľ si po skončení práce svoje dáta exportuje do súboru na jeho lokálnom úložisku a využije ich pri importe počas ďalšieho využívania aplikácie s rovnakým datasetom. Exportuje využitím príslušného tlačidla, pričom názov exportovaného súboru obsahuje dve varianty:

- **Názov súboru užívateľ vyplnil:** V prípade zadania názvu súboru pre export bude po jeho potvrdení súbor pripravený na stiahnutie v tvare „názov-zadaný-užívateľom.csv“.

- **Názov súboru užívateľ nevyplnil:** Ak názov súboru pre export nebude zadaný, bude použitý predvolený názov. Stiahnutý súbor tak bude pomenovaný „export.csv“.

3.4.6 Select dát

Funkčnou požiadavkou zákazníka pre túto aplikáciu je aj pohľad na dáta priamo v aplikácii. Okrem celkového pohľadu na všetky záznamy je vyžadovaná funkcionalita zobrazenia vybraných dát v dvoch kategóriach:

- **Zobraziť dáta vybraných stĺpcov:** Možnosť definovať užívateľom stĺpce, ktoré chce vidieť v zobrazení dát a prípadnom exporte. Výber stĺpcov uskutočňuje klikaním na „check-box“ v zozname stĺpcov. Minimálny počet vybraných stĺpcov je 1, pričom maximum tvoria všetky stĺpce.
- **Zobraziť dáta vybraných záznamov:** Pre zobrazenie špecifickej skupiny záznamov musí mať užívateľ možnosť zadať pokročilejšiu podmienku s využitím SQL operací(AND,OR,=). Presná syntax bude uvedená v dokumentácii aplikácie. Využitie tejto podmienky závisí priamo od užívateľa a nie je tak povinná.

Predpokladom zákazníka je to, že užívateľ vo väčšine prípadov bude chcieť vybrať všetky stĺpce. Aby tak nemusel tráviť čas postupným vyklikávaním, je požiadavkou na funkčnosť vytvoriť možnosť jednorázovo zakliknúť a odkliknúť všetky stĺpce naraz.

Export definovaný vyššie 3.4.5 musí fungovať aj pre stiahnutie čiastkových dát na základe podmienok užívateľa.

3.4.7 Grafy

Aplikácia by mala dokázať vytvoriť vopred definové a implementované grafy. Fungovať by mali na princípe zadania hodnôt parametru od užívateľa a následného zobrazenia samotného grafu.

Zákazník ako jedinú presnú požiadavku na graf špecifikuje možnosť vytvoriť graf početnosti rýb za daný rok. Parameter rok je zadaný užívateľom. Spojí všetky záznamy s príslušným rokom, vykoná súčty jednotlivých druhov rýb a zobrazí ich v prehľadnom grafe. V prípade, že pre užívateľom zvolený rok žiadny záznam neexistuje, bude ho o tom informovať.

3.4.8 Upozornenia a oznámenia

Pri práci v aplikácii by sme mali užívateľa informovať o stavoch a aktivitách, ktoré neexistujú, nie sú správne nastavené a podobne. Tak isto užívateľ očakáva informáciu o tom, že niektoré procesy sa vykonali a ukončili správne. Aplikácia by preto mala zobraziť:

- Oznámenie o úspešnom pridaní záznamu do databázy.
- Upozornenie v prípade neexistujúceho záznamu pre prípad editácie a zmazania.

- Vyžiadanie potvrdenia zmazania záznamu, ako spôsob predídenia straty dát v dôsledku prekliknutia sa užívateľom.
- Oznámenie o úspešnom zmazaní záznamu v databáze.
- Upozornenie v prípade, že užívateľ nevybral žiadny záznam pri operácií select dát.
- Oznámenie alebo vizualizácia úspešného exportu súboru do lokálneho filesystému užívateľa.
- Upozornenie v prípade importu nevalidného súboru, že načítanie dát neprebehlo.
- Upozornenie pre užívateľa, že v prípade nahrania nových dát sa súčastný dataset vymaže.
- Oznámenie o úspešnom importe dát zo súboru.
- Upozornenie, že pre zadaný rok neexistuje žiaden záznam v databaze, v prípade pokusu o zobrazenie grafu výskytu rýb jednotlivých druhov za rok zadaný ako parameter od užívateľa.

3.5 Nefunkčné požiadavky

V tejto časti špecifikujem nefuknčné požiadavky, ktoré vychádzajú z aktuálnych potrieb zákazníka. Pri ich zostavovaní sme spolu so zákazníkom mysleli aj na budúci rozvoj a pri ich tvorbe počítali aj s plánmi v blízkej budúcnosti.

3.5.1 Webové rohranie

Aplikácia bude mať prehľadné a pre užívateľa jednoducho sa orientujúce webové GUI.

3.5.2 Podporovaný csv formát

Pre nahratie dát z a do aplikácie bude podporovaný výhradne formát CSV. Podľa aktuálnych formátov týchto súborov u zákazníka, ako aj budúcich plánov, sa za jediný platný oddeľovač považuje bodkočiarka „;“. Pre nahratie je striktne daný počet stĺpcov zdrojového súboru, ktorý bude aplikácia považovať za validný.

Čiastkové exporty po využití selectu dát a odstránenie niektorej časti nebudú akceptované pre import. Jediný spôsob, ako exportované dáta znova importovať, je využiť možnosť exportovať všetky stĺpce a zároveň nedefinovať žiadnu podmienku. Tým zaručíme, že dáta sa exportujú ako celok uložený v databáze a môžu byť následne znovu nahrané.

3.5.3 Lokálna aplikácia

Celá aplikácia pobeží na lokálnom systéme každého užívateľa a nebude prepojená s lokálnymi inštanciami aplikácie druhých užívateľov. Jej webové rozhranie bude fungovať na adrese localhost a porte 9080.

3.5.4 Autorizácia a autentifikácia

Vzhľadom na fakt, že ide o lokálnu aplikáciu, nie je vyžadované prihlásenie do aplikácie prostredníctvom hesiel. Taktiež nie je potrebné definovať viaceré role a užívateľ tak bude mať práva na všetky úkony rovnako ako každý ďalší užívateľ vo svojej inštancii aplikácie.

3.5.5 Úložisko dát

Pre každú lokálnu inštanciu aplikácie sa počas behu vytvorí pomocná databáza s funkciou datamartu na uskladnenie dát medzi importom a exportom. Táto databáza bude existovať ako databáza priamo v prehliadači (SQLite) a bude do nej možné pristupovať pomocou webových API (WebSQL). Každý užívateľ má vlastnú inštanciu databázy počas behu v jeho lokálnom prehliadači a žiadnym spôsobom nezdieľa vzácne dáta s ostatnými užívateľmi.

3.5.6 Dátové typy stĺpcov

Zákazník v tomto momente nedokáže presne stanoviť formát a typ jednotlivých stĺpcov. Z tohto dôvodu je nutné ponechať všetky stĺpce typu string.

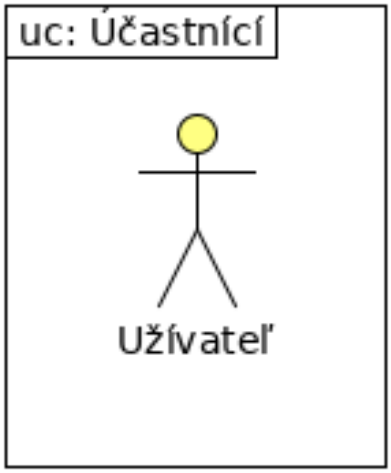
Ďalším dôvodom pre toto nastavenie je fakt, že zákazník chce v budúcnosti využiť aplikáciu aj pre iné dátové sady s odlišným názvami a dátovými typmi stĺpcov, ktoré pred importom správne namapuje na súčasné, bez riešenia konverzie medzi jednotlivými dátovými typmi.

3.6 Use cases (prípady užitia) - diagramy

V rámci špecifikácie, analyzovania a upresňovania funkčných požiadaviek som spolu so zákazníkom vytvoril prípady užitia. Práve use cases diagramy sa v praxi ukázali ako najpochopiteľnejšie pre zákazníka mimo nášho oboru pôsobenia. Vďaka nim sa mi podarilo presne určiť jednotlivé prípady užitia podľa predstáv a potrieb zákazníka.

3.6.1 Účastníci

Najčastejším bežným delením je rozdelenie účastníkov na adminov, ľudí ktorí spravujú oprávnenia, ovládajú niektoré skryté funkcie, a bežných užívateľov, ktorí systém využívajú.



Obr. 3.4: Use Cases - užívatelia.

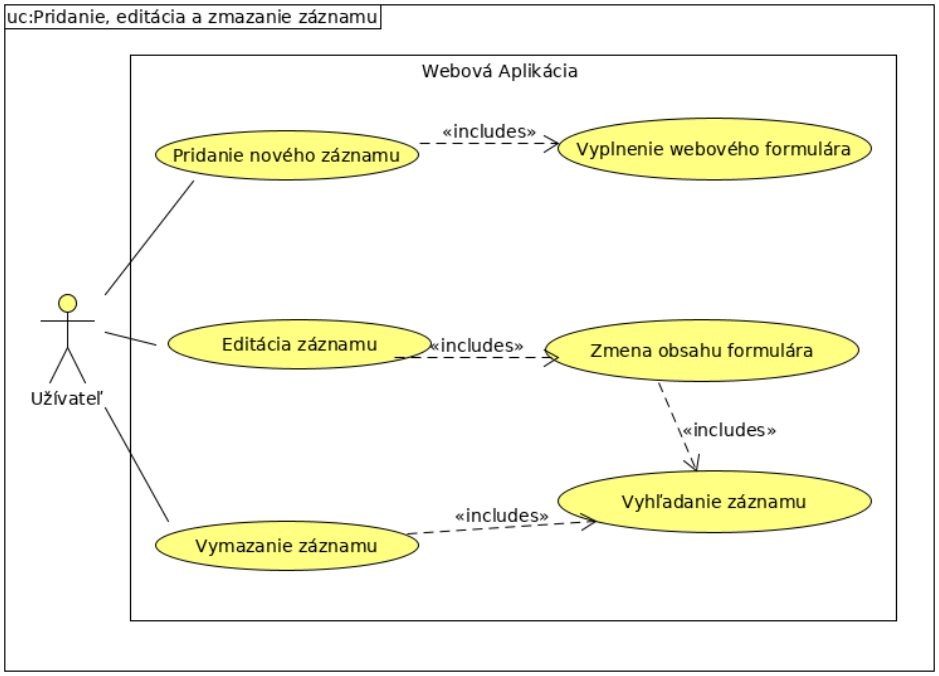
V prípade lokálnej aplikácie bez rozdelenia na role budeme pracovať s jedným typom účastníka, ktorý bude mať plné práva na všetky úkony. Pre tvorbu prípadov užitia si ho pomenujeme **Užívateľ** 3.4.

3.6.2 Zmena dát

V prvej skupine prípadov užitia na obrázku 3.5 sú zobrazené hlavné možnosti užívateľa v prípade, že jeho cieľom je zmeniť aktuálne dáta.

Pridanie nového záznamu
Užívateľ sa rozhodne pre pridanie nového záznamu do existujúceho datasetu. Aby mu to bolo umožnené, potrebuje vyplniť webový formulár obsahujúci predpripravené políčka pre všetky stĺpce. Po vyplnení klikne na „Add Record“ element a dáta sa zapíšu do databázy.

Editácia záznamu
V prípade, že chce užívateľ vykonať zmenu a upraviť jeden z už existujúcich záznamov v datasete, musí ako prvé daný záznam vyhľadať. Následne sa mu zobrazí webový formulár vybraného záznamu, ktorý má možnosť editovať a nahradiť tak pôvodné hodnoty novými. Kliknutím na element „Save Record“ sa editovaný záznam uloží do databázy.



Obr. 3.5: Use Cases - zmena dát.

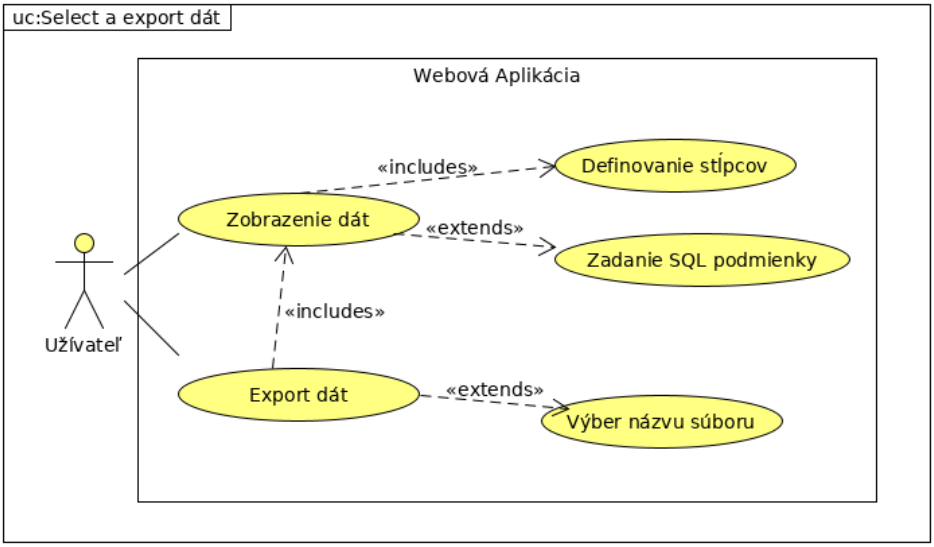
Zmazanie záznamu
Pre úplne zmazanie záznamu z databázy musí užívateľ záznam vyhľadať a následným kliknutím na „Delete Record“ element ho úplne odstráni.

3.6.3 Select a export dáť

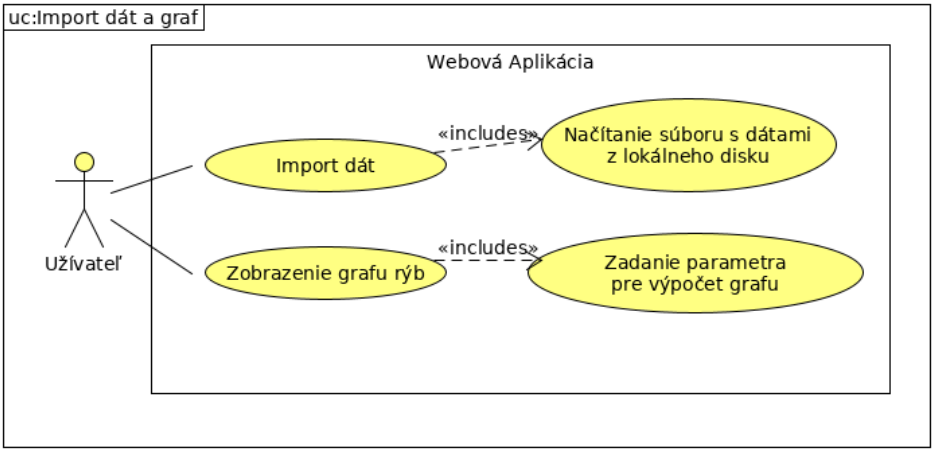
Prípady užitia, ktoré súvisia so zobrazením a následným exportom dát znázorňuje obrázok 3.6.

Zobrazenie dát
Užívateľ má možnosť nahliadnuť na dáta priamo v prostredí samotnej aplikácie. Jediné, čo k tomu potrebuje, je definovať stĺpce, ktorých dáta má záujem zobraziť. Ako voliteľná a nepovinná funkcia je tu pre neho možnosť zadať SQL podmienku, ktorá mu pomôže upresniť vyhľadávanie na úrovni konkrétnych záznamov.

Export zobrazených dát
Najžiadanejšou vlastnosťou aplikácie je práve export samotných dát. Pre túto možnosť musí užívateľ samotné dáta najskôr zobraziť pomocou predchádzajúceho scenára. Ako doplnková sa mu ponúka možnosť voľby vlastného názvu súboru pre export.



Obr. 3.6: Use Cases - Select a export dát.



Obr. 3.7: Use Cases - Import dát a graf.

3.6.4 Import dát a graf

Posledné dva prípady užitia, import dát a zobrazenie grafu, sú úplne nezávislé. Sú však tak malého rozsahu a jednoduchosti, že som ich spojil do jedného obrázku 3.7.

1	U	Úmysel pridať nový záznam
2	S	Zobrazenie prázdneho formulára
3	U	Vyplnenie formulára
4	U	Potvrdenie pridania záznamu
5	S	Uloženie záznamu do databázy
6	S	Informovanie o úspešnosti uloženia

Tabuľka 3.2: Scenár pridania záznamu.

Import dát
Užívateľ má možnost importovať celý dataset v prípade, že nechce začínať prácu s prázdnou databázou. Pre úspešne nahranie dát je potrebné načítať súbor obsahujúci dáta z lokálneho filesystému.

Graf
Užívateľ má možnosť zobraziť si vopred predimplementovaný graf založený na aktuálnych hodnotách dát v databáze. Jeho úlohou je definovať parameter príslušný k danému grafu, čím spôsobí dynamický obraz každého grafu.

3.7 Use cases (prípady užitia) - scenáre

Na základe diagramov prípadov užitia som vytvoril samotné scenáre. V scenároch je rola užívateľa reprezentovaná písmenom **U** a rolu systému značím písmenom **S**.

V tejto práci uvádzam len niektoré z nich. Snažil som sa vybrať tie najzaujímavejšie:

- **Scenár pridania záznamu** 3.2. Jednoduchý scenár pridania nového záznamu do databázy.
- **Scenár editácie záznamu** 3.3. Scenár popisuje kroky k úspešnému vyhľadaniu a zmene záznamu.
- **Scenár selectu a exportu dát** 3.4. Najzložitejší scenár, ktorý zachytáva výber stĺpcov, dát a následný export.

3.8 Doménový model

Posledným článkom analytickej časti je doménový model. Častokrát sa tu popisuje zložitá štruktúra medzi jednotlivými entitami a vzťahy medzi nimi. V prípade webovej aplikácie pre odlov rýb bude náš model tvoriť jediná doména.

Vzhľadom na pôvod a štruktúru dát pripomínajúcu zoznam nebolo možné

1	U	Úmysel zmeniť hodnotu záznamu
2	S	Zobrazenie dialógu pre zadanie ID
3	U	Zadanie ID záznamu
4	S	Vyhľadanie záznamu v databáze
5	S	Zobrazenie dát záznamu vo formulári
6	U	Zmena údajov formulára
7	U	Potvrdenie zmeny
8	S	Prepísanie hodnoty záznamu v databáze
9	S	Informovanie o úspešnom prevedení editácie

Tabuľka 3.3: Scenár editácie záznamu.

1	U	Úmysel výberu dát
2	S	Zobrazenie ponuky sĺpcov a možnosti zadať pokročilú podmienku
3	U	Výber stĺpcov
4	U	Zadanie pokročilej podmienky
5	U	Potvrdenie výberu dát
6	S	Zobrazenie vybraných záznamov
7	S	Zobrazenie možnosti exportu vybraných dát
8	S	Zobrazenie možnosti zadania názvu výstupného súboru
9	U	Zadanie názvu výstupného súboru
10	U	Potvrdenie exportu
11	S	Stiahnutie súboru na lokálny systém.

Tabuľka 3.4: Scenár selectu a exportu dát.

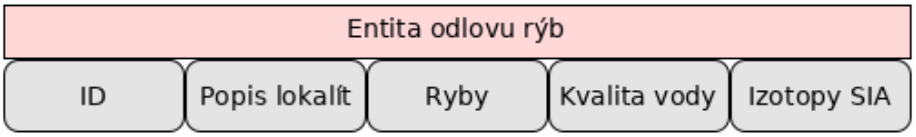
rozdeliť dáta do viacerých entit ako je zvykom pri štandardnom doménovom modeli. Po dôkladnej analýze a vyčistení dát z mnohých zdrojových súborov som sa so súhlasom zákazníka rozhodol použiť jedinú entitu obsahujúcu 133 atribútov. Pre našu problematiku je toto riešenie najprijateľnejšie, a preto som sa ho rozhodol implementovať touto cestou.

Napriek faktu, že atribúty sú uložené v jednej entite, môžeme ich pomyselne rozdeliť do 5 hlavných kategórii zobrazených na obrázku 3.8. Ich spojením dostanem jednu výslednú tabuľku, ktorá tvorí doménový model a zároveň je postačujúca pre potreby zákazníka.

Obrázok 3.9 zobrazuje atribúty v jednotlivých kategóriach. Ide o názvy stĺpcov v databáze. Tieto názvy sú aj súčasťou exportu. V aplikácii však vidí užívateľ kompletné názvy vrátane diakritiky, ktoré tak pre neho dávajú oveľa väčší zmysel.

Tabuľka 3.5 udáva, koľko obsahujú jednotlivé kategórie atribútov.

Z historických dát, ktoré máme k dispozícii je jasné, že pri prepise záznamov do novej štruktúry našej aplikácie nebudú všetky stĺpce vyplnené. Je to však dobrý predpoklad pre lepšiu správu dát v budúcnosti. Taktiež nové usporia-



Obr. 3.8: Doménový model - kategórie dát.

ID	Popis Lokalít	Ryby	Kvalita Vody	Izotopy SIA
1	33	57	37	4

Tabuľka 3.5: Počet atribútov v každej kategórií.

danie umožňuje v budúcnosti dáta dohľadať a doplniť.

3.8.1 ID

Prvá kategória obsahuje jediný stĺpec a to **ID**. Tento stĺpec nevidí užívateľ pri pridávaní ani editovaní záznamu. Je generovaný automaticky a zabezpečuje tak jedinečnosť záznamu. Môžeme ho preto s určitosťou považovať za **primárny kľúč** entity.

3.8.2 Popis lokalít

Táto kategória obsahuje všetky geografické informácie o každom odlove. Okrem predpripravenej štruktúry pre zachytenie štátu, kraja a presnej lokality, zachytáva aj presné čísla ako riečny kilometer, dĺžka toku, frekvencia toku a ďalšie.

3.8.3 Ryby

Môžme povedať, že obsah tejto kategórie tvorí zoznam druhov rýb. Aj z tohoto dôvodu má táto skupina najviac atribútov. V pôvodných dátach obsahoval každý súbor a zápis niektoré druhy rýb. Pre skompletizovanie údajov a ich prehľadnú štruktúru sme preto v úzkej spolupráci so zákazníkom vytvorili jeden zoznam všetkých dostupných druhov. Po vyčistení jednotlivých súborov tak budú počty rýb štruktúrovito a jednoznačne uložené v jednej entite databázy.

V aplikácii užívateľ uvidí celé latinské názvy atribútov pre jednotlivé druhy. Pre ich dĺžku a špeciálne znaky sa v databáze uvádzajú v skrátenom, na prvý pohľad menej napovedajúcom formáte.

3.8.4 Kvalita vody

Sekcia kvalita vody má na starosti zhromaždiť čo najviac informácii o chemickom zložení vody počas jednotlivých odlovov. Obsahuje pripravené atribúty pre najčastejšie sa vyskytujúce zložky vody. Zachytáva informácie napríklad o počte uhlíka, fosforu a ďalších látok, ktoré majú priamy dopad na rybiu populáciu daného úseku vodného toku.

V pôvodných dátach sú ku každému odlovu vyplnené len niektoré atribúty. Zákazník si však dokáže doplniť údaje o kvalite vody z podobných záznamov práve vďaka jednoznačnej štruktúre danej sekcie.

3.8.5 Izotopy SIA

Posledná kategória obsahuje len 4 atribúty. Jej cieľom je zachytiť a zaznamenať analýzu stabilných izotopov počas konkrétnych odlovov rýb.

ID	Popis lokalít	Ryby	Kvalita vody	Izotopy SIA
- ID	- nazev_toku - lokalita_toku - specifikace_lokality - stat - povodi - kraj - rok_odlovu - datum_odlovu - poznamka - autor - morfologie - mnm - spad - rad_toku - prolovena_delka - sirka_useku - plocha_useku - ricni_km - regulace - pramenmnm - ustimnm - povodivkm - delka_toku - prutok - speciesrichness - speciediversity - abundance - abundancenastometrov - abundance_hektar - frekvence - biomasa - biomananastometrov - biomasahektar	- po_Salmo - pd_Oncorhynchus - vr_Cottus - vrp_Cottus - mi_Lampetra - miu_Eudontomyzon - mr_Barbatula - hr_Gogio - hrk_Gobio - hrb_Gobio - li_Thymallus - si_Salvelinus - tl_Leuciscus - pr_Leuciscus - pl_Rutilus - js_Leuciscus - str_Phoxinus - strevlicka_Pseudorasbora - hlc_Protororhinus - pis_Misgurnus - parma_Barbus - ok_Perca - st_Esox - ka_Cyprinus - kar_Carassius - karst_Carassius - jez_Gymnocephalus - jezz_Gymnocephalus - sek_Cobitis - uh_Anguilla - mn_Lota - lin_Tinca - cv_Abramis - cs_Abramis - cp_Abramis - cm_Blicca - ou_Alburnus - cor_Alburnoides - per_Scardinius - pod_Vimba - ostr_Chondrostoma - kol_Gasterosteus - slu_Leucaspius - hor_Rhodeus - can_Stizostedion - cav_Stizostedion - drv_Zingel - bo_Aspius - ab_Ctenopharyngodon - tob_Hypophthalmichthys - sum_Silurus - smc_Ictalurus - hlav_Hucho - jesm_Acipenser - ostruch_Pelecus - sekc_Sabanejewia - jezdun_Gymnocephalaus	- kva_vod - tep_vod - ph_vod - priihlednost - sediment - roz_lat - neroz_lat - vodivost - o2_vt - chskmn - bsk - alkalita - acidita - vapnik - tvrdost - chloridy - chlorofil - uhlik - uhlik_org - uhlik_rozp - c_anor - uhlicitany - dus_cel - chskmg - nno2 - nno3 - no3mg - no2mg - nh4mg - tp_fosfor - ppo4 - femg - mnmg - hum_lat - pesticidy - pcb - pfos - ddt	- col_13c - col_15n - tc - tn

Obr. 3.9: Doménový model - atribúty.

Kapitola 4

Návrh Aplikácie

V tejto kapitole sa zameriam na samotný návrh frontendu aplikácie. Jedná sa o úplne novú aplikáciu, preto nebudem vychádzať z existujúcich návrhov a verzií. Pri tvorbe návrhu som sa snažil spojiť jednoduchosť, s akou má daný typ užívateľa využívať aplikáciu, s grafickou časťou, ktorú som v rámci mojej schopnosti dokázal vytvoriť. Návrh a samotná aplikácia preto neobsahuje graficky správne a čisté prvky ani efekty. Dôraz kladie na minimalizmus a zobrazenie nutného množstva funkcí a elementov, ktoré potrebuje užívateľ k tomu, aby dokázal aplikáciu plnohodnotne využívať.

4.1 Hlavná stránka

Stránka, ktorú užívateľ uvidí ako prvú pri štarte aplikácie. Úvodná, alebo hlavná stránka je prvým kontaktom užívateľa s prostredím. Je dôležité, aby svojim vzhľadom a obsahom poskytla informácie o celej aplikácii a zároveň dokázala užívateľa navigovať do ďalších častí aplikácie.

Kedže sa jedná o prvotnú verziu webových stránok, užívateľ tu nájde popis aplikácie a krátke zhrnutie pôvodu jej vzniku a účelu. Do budúcna sa samozrejme počíta s využitím hlavnej plochy na účely východzích nastavení jednotlivých elementov. Tie vzniknú ako odozvy od užívateľov po určitom čase používania.

Čo však užívateľ ocení a využije je menu ponuka, z ktorej sa priamo dostane do jednotlivých sekcií. Pre prvotnú verziu aplikácie je horný panel rovnaký pre všetky stránky s cieľom nemýliť užívateľov pri počiatočnom navyknutí na chod aplikácie.

4.2 Pridanie nového záznamu

Prvá naimplementovaná a užívateľmi na základe analýzy najočakávanejšia stránka pre pridanie nového záznamu do existujúcej databázy.

V súčasnom stave má databáza striktne stanovený počet a názvoslovie jednotlivých stĺpcov. To je dôvod, prečo sa užívateľom po otvorení úvodnej stránky zobrazí predpripravený formulár obsahujúci políčka všetkých stĺpcov aplikácie. Užívateľ bude mať možnosť ich vyplniť ako celok, ale validná akcia je aj vyplnenie len určitých údajov.

Pre zjednodušenie vyplňovania formulára viac ako 100 stĺpcov bude pre užívateľa možnosť preskakovať stlačením klavesy medzi jednotlivými stĺpcami. Urýchli a zefektívni to tak pridávanie nových záznamov.

Nápomocné môže byť aj tlačidlo pre vymazanie obsahu formulára na jedno kliknutie v prípade, ak zistí, že zadáva nesprávne hodnoty. Ušetrí to tak čas manuálneho vymazávania všetkých buniek.

Akciu pridania záznamu potvrdí užívateľ kliknutím na tlačidlo umiestnené vpravo na spodu formulára. O úspešnosti pridania záznamu do databázy ho informuje vyskakovacia hláška, ktorú musí potvrdiť.

Po úspešnom pridaní sa vizuál tejto stránky zmení na menší rozcestník, kde má užívateľ na výber pridanie ďalšieho záznamu, zmenu záznamu alebo náhľad na všetky dáta.

4.3 Editovanie a vymazanie záznamu

Túto podstránku navštívi užívateľ v prípade, že chce zmeniť hodnoty existujúceho záznamu, alebo tento záznam úplne vymazať. Z analýzy procesov vidíme, že je požiadavka záznam najskôr vyhľadať a až následne s ním vykonávať ďalšie operácie.

Z tohto dôvodu sa užívateľovi zjaví políčko, kde zadá ID hodnotu záznamu. Až v prípade zhody v databáze a teda existencie záznamu sa na stránke zobrazí formulár. Ten už pri zobrazení obsahuje aktuálne hodnoty z databázy.

Pre akciu editácie i zmazania bude mať užívateľ dve tlačidlá umiestnené na rovnakom mieste ako pri vytváraní záznamu. V prípade ich potvrdenia sa záznam vymaže respektíve obnoví v databáze.

4.4 Select a export záznamov

Z analýzy vyplýva, že výber stĺpcov je povinný. Z tohto dôvodu sa pri zobrazení stránky rovno ukáže zoznam stĺpcov, ktorý musí užívateľ vybrať. Označí tak stĺpce, o ktoré má záujem. Tento výber bude naimplementovaný pomocou check boxov, ktoré užívatelia žaškrtajú.

Očakávaným prípadom je záujem o všetky stĺpce. K dispozícii tak bude tlačidlo pre výber všetkých stĺpcov, čo ušetrí čas pri označovaní. Taktiež sa tu objaví aj opačné tlačidlo, ktoré zruší výber všetkých označených stĺpcov.

Pod týmto zoznamov bude k dispozícii miesto pre umiestnenie nepovinnej podmienky vyhľadávania. Hneď pri nej umiestnim tlačidlo pre potvrdenie a samotné zobrazenie dát. Tie sa zobrazia pod zoznamom stĺpcov a priestorom

pre podmienku. Užívateľ bude musiť zarolovať na stránke nižšie, kde nájde vizuálne zobrazenie vybraných dát.

Na úplnom spodku stránky nájde tlačidlo pre export dát a priestor na zvolenie vlastného názvu výstupného súboru. Exportová sekcia sa zjaví len v prípade, že vybrané dáta na základe požiadaviek užívateľa obsahujú aspoň jeden záznam.

4.5 Import dát

Táto stránka má veľký potenciál do budúcna. Môže obsahovať rôzne typy inportovacích šablón a funkcií. V prvotnej verzií aplikácie ale bude obsahovať iba jednoduché okno, v ktorom užívateľ nahrá zvolený súbor pre import. Ten bude úspešne dokončený v prípade správnej štruktúry súboru. O prípadnom úspechu, či neúspechu operácie importu bude užívateľ informovaný vyskakovacím oknom s príslušnou správou.

4.6 Graf

Do budúcna je možné implementovať viaceré grafy podľa požiadaviek zákazníka. Všetky by mali obsahovať približne rovnakú štruktúru. Prvotná verzia aplikácie preto bude obsahovať jeden graf, na základe ktorého bude možnosť implementovať ďalšie druhy.

V príklade grafu na počet rýb za jednotlivý rok sa užívateľovi zobrazí formulár so žiadosťou zadania hodnoty roku pre konkrétny výpočet. Potvrdením tohto výberu sa v spodnej časti obrazovky ukáže samotný graf. Ten bude typu bar. Vykreslenie grafu bude podmienené existujúcou skupinou záznamov pre daný rok. Zabránim tak vykresleniu prázdneho grafu, ktorý môže užívateľa zmýliť. O takomto konaní samozrejme informujem vyskakovacím oknom s príslušnou hláškou.

Kapitola 5

Implementácia

V tejto kapitole sa venujem jednotlivým technológiam použitým pri implementácii. Na začiatku detailne rozoberiem databázové úložisko a spôsob, ako k nemu pristupujem. Následne predstavujem jednotlivé implementačné nástroje a pridávam k nim aj krátky popis.

Nevynechávam ani popis architektúry, ktorý ukazuje, ako jednotlivé časti spolu súvisia. Pre správne fungovanie uvádzam aj obmedzenia pre daný systém.

5.1 Databáza

Pri výbere vhodného typu databázy som dbal na jednoduchý prístup a možnú lokálnu implementáciu bez zložitého nastavovania pre jednotlivé inštancie. Hľadal som takú možnosť, kde sa dáta môžu zapísať priamo do jednej tabuľky bez nutnosti zakladania zložitej schémy a vytvárania užívateľských rolí, prístupov a práv.

Použil som SQlite databázu počas procesu tvorby a debugovania aplikácie. Pre samotný beh už používam WebSQL lokálnu databázu, ktorá je súčasťou prehliadača každého užívateľa.

5.1.1 Databázový model

V mojom prípade je naprosto zhodný s doménovým modelom 3.8. Všetkým atribútom je priradený dátový typ string a sú spolu uložené v jednej tabuľke. Nepokladám preto za potrebné duplikovať túto informáciu v mojej práci. To hlavne z dôvodou toho, že nevyužívam žiadne vzťahy medzi entitou, ktoré by mohli byť rozoberané.

5.1.2 SQLite

Na rozdiel od známych databáz ako MySQL, PostgreSQL, Oracle, ktoré fungujú ako služba, je SQLite malá knižnica nástrojov, ktorú majú niektoré jazyky

zabudované v sebe. Databáza sa ukladá na disk ako súbor s príponou **.db**.

Výhodou SQLite je jednoduchá možnosť presunu objektu databázy. Stačí skopírovať zdrojovú aplikáciu so súborom s databázou a spustiť. Nie je potrebné riešiť akékoľvek zložité pripojenia a inštalácie. Neriešime ani prihlasovanie a nastavovanie rolí databázy. Je vhodná pre použitie na desktopových aplikáciach. Stáva sa tak ideálnou voľbou pre mnou vytvorenú aplikáciu [16].

5.1.3 DB Browser pre SQLite

Pri vytváraní databázy som potreboval kontrolovať správne načítanie a prácu s dátami. Na túto činnosť mi poslúžil DB Browser. Jedná sa o veľmi obľúbenú, jednoduchú aplikáciu pre prehliadania, úpravu a tvorbu SQlite databáz v príjemnom grafickom prostredí. Výhodou je fakt, že plne postačuje pre základné úpravy bez hlbšej znalosti SQL. Je zdarma dostupná z oficiálnych stránok výrobcu a inštaláciu zabezpečuje jednoduchý inštalátor [16].

5.1.4 Web SQL

Web SQL je webové API pre ukladanie dát v databázach, ktoré môžu byť dotazované pomocou variantu SQL. Oficálne ich podporuje niekoľko prehliadačov [17]. Ja som ich testoval v prehliadači „Google Chrome“.

Pri prístupe do databázy využívam tri operácie:

- **Open Database:** Definícia databázy. Ak konkrétna databáza neexistuje, API vytvorí novú. Zatvorenie spojenia s databázou v tomto prípade nie je nutné riešiť.

  ```
  var db = openDatabase('Fish_DB', '3.24.0', 'Fish Database',
                        2 * 1024 * 1024);
  ```

  Prvým argumentom je názov samotnej databázy. V mojom prípade je to „Fish_DB“. Nasleduje verzia databázy. Tá je vopred definovaná a otestovaná. Tretím argumentom je popis vytvorenej databázy. Na záver je nutné pridať predpokladanú veľkosť vzhľadom na počet užívateľských dát.

- **Transactions:** Teraz, keď som otvoril definovanú databázu, môžem vytvoriť transakcie. Sú dôležité a ich využitie je nevyhnutné. Transakcie mi umožňujú vrátiť sa späť. To znamená, že ak transakcia, ktorá by mohla obsahovať jeden alebo viac príkazov SQL, zlyhá (buď SQL alebo kód v transakcii), aktualizácie databázy sa nikdy nespájajú. To má za následnosť, že transakcia sa nikdy nestala.
  V transakcii sú aj spätné volania týkajúce sa chýb a úspechov, takže môžete spravovať chyby. Je dôležité pochopiť, že transakcie majú schopnosť vrátiť sa späť.

```
db.transaction(function (t) {
//samotná transakcia
 });
```

- **Execute SQL:**Ak mám definové transakcie, môžem vo vnútri volať ľubovolné množstvo SQL transakcí pre čítanie, vkladanie a ďalšie operácie s dátami v databáze.
  Ukážka kódu pre výber dát a následný výpis počtu riadkov v tabuľke „FISH“.

```
t.executeSql('SELECT * FROM FISH',[],function(tx,results){
     var rows = results.rows;
           var len = rows.length;
           var cislo;
           console.log(len);
   });
```

V kóde aplikácie využívam prevažne funkcie:

- **SELECT** pre získanie dát z databázy.
- **UPDATE** pre úpravu záznamu na základe aktivity užívateľa.
- **DELETE** v prípade potreby vymazania záznamu.

5.2 Použité technológie a nástroje

V tejto časti rozoberám a zdôvodňujem všetky technológie, ktoré využívam pri samotnej implementácii.

5.2.1 Javascript

JavaScript, je skriptovací programovací jazyk. Je používaný najmä pri tvorbe webových stránok. To bol jeden z hlavných dôvodov, prečo som si ho zvolil ako hlavný implementačný jazyk. Takisto som na neho získal kladné referencie a usúdil, že pre moju aplikáciu bude vhodný a plne postačujúci.

Pôvodne ho vyvíjal Brendan Eich zo spoločnosti Netscape Communications pod názvom LiveScript. Pred uvedením na verejnosť bol premenovaný na Javascript, najmä pre vtedajšiu popularitu jazyka Java. Aj na základe jeho názvu je rozšírený názor, že syntax Javascriptu sa podobá Jave. V skutočnosti bol jeho tvorca najviac inšpirovaný jazykom Self [18].

V súčasnosti je dostupné množstvo implementácii virtuálnych strojov jazyka Javascript, proprietárnych i s otvoreným zdrojovým kódom, pričom niektoré z týchto implementácii je možné používať ako knižnice do iných programov a mať tak podporu tohto jazyka v prakticky ľubovoľnej aplikácii. Spôsoby

interpretácie sa rôznia medzi implementáciami. Niektoré využívajú jednoduché spracovanie kódu do abstraktného syntaxného stromu, ktorý následne prechádzajú a interpretujú interpreterom priamo zdrojový kód.

5.2.2 HTML

HTML je štandardný značkovací jazyk na vytváranie webových stránok. Popisuje štruktúru webových stránok pomocou značenia. Jeho prvky sú stavebné bloky stránok a sú reprezentované tagmi. Značky označujú kúsky obsahu, ako napríklad nadpis, odsek, tabuľka. Prehliadače nezobrazujú značky, ale používajú ich na vykreslenie obsahu stránky [19].

HTML pokladám za základný prvok implementácie webovej stránky. Vo svojom kóde využívam samostatné HTML stránky pre jednotlivé operácie užívateľa na základe výberu z menu.

Zo štandarných metód jazyka HTML využívam vo svojej práci metódu POST. Tá mi slúži pri potvrdení webového formulára. Prostredníctvom nej prechádza užívateľ na potvrdzovaciu stránku. Definujem ju v hlavičke daného formulára nasledovne:

```
<form id="adding_form" action="/add_confirm.html" method="POST">
```

5.2.3 CSS

Kaskádový štýl CSS je jazyk, ktorý popisuje a definuje štýl elementov v HTML. Určuje a definuje, ako budú jednotlivé elementy na výslednej stránke vyzerať [20]. Spolu s javascriptom a HTML tvoria základnú časť webu. Vďaka CSS môžem nastaviť elementom rozmer, farbu a určiť polohu na stránke, kde sa užívateľovi zobrazia.

Vo svojej práci využívam verejné dostupné šablóny, ktoré následne upravujem do finálnej potreby podľa potreby aplikácie. V kóde sú využité online odkazy, preto musí mať aplikácia pre správny chod prístup k internetu. Práve práca s CSS predstavovala najväčšiu výzvu pri tvorbe tejto práce z dôvodu najmenších znalostí danej časti. Preto aplikácia obsahuje len základné návrhy pre farbu, veľkosť elementov a rozmiestnenie. Lepší grafický vzhľad nie je hlavnou náplňou tejto práce.

5.2.4 Netbeans

Dôležitou súčasťou práce bol aj výber vývojového prostredia. Zvolil som open source Netbeans z dôvodu, že už som s ním mal drobné skúsenosti. Využíval som ho pre vývoj programov počas prvej polovice štúdia. Podstatná pre mňa bola možnosť prehľadného vytvárania jednotlivých častí aplikácie a podpora node.js, ako aj priama možnosť napojenia na verzovací nástroj GIT.

Pôvodne bolo Netbeans vytvorené ako vývojové prostredie pre javu. Dnes

má aj viaceré rozšírenia. Pri svojej práci využijem možnosť programovať v jazyku javascript priamo z tohto vývojového prostredia.

5.2.5 Node.js

Node.js je open source runtime prostredie, ktoré dokáže spúšťať javascript. Pri vývoji umožnuje skriptovanie a písanie príkazov na strane servera. To má za následok dynamickú tvorbu webu. V mojej práci teda Node.js reprezentuje stranu servera, kde sa stará o zobrazenie jednotlivých webových stránok na základe aktivít užívateľa.

Node.js využívam na programovanie všetkých funkcií na servrovej strane. Ako príklad uvedie metódu **POST**, ktorá ma na starosti potvrdenie webového formulára, odoslanie dát a zobrazenie nasledujúcej webovej stránky pri pridaní nového záznamu do databázy.

```
http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename =  q.pathname;
  //var filename = "." + q.pathname;
  fs.readFile("./app/project" + filename, function(err, data) {
  //fs.readFile("project/" + filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
  if(req.method === 'POST' && req.url === "/add_confirm.html"){
        var body = '';
        req.on('data', chunk => {
            body += chunk.toString();
        });
        // when complete POST data is received
        req.on('end', () => {
            // use parse() method
            body = querystring.parse(body);
        });
    }
}).listen(9080);
```

Server vytvorí spojenie na porte 9080. Následne čaká na ďalšie zmeny URL na základe aktivity užívateľa. V prípade, že sa užívateľ dožaduje potvrdiť formulár metódou POST, funkcia volá príslušné vnorené funkcie.

5.2.6 Git

Počas tvorby aplikácie som potreboval uchovávať jednotlivé verzie programu a prehľadávať medzi nimi. Taktiež bolo potrebné zabrániť zmazaniu a strate dát. Musel som preto vybrať vhodný verzovací systém, ktorý mi umožní rýchlo a jednoducho prístup a ukladanie verzií. Zároveň som požadoval priamu integráciu s vývojovým prostredím Netbeans. Rozhodol som sa pre použitie verzovacieho systému Git.

Jedná sa o bezplatný a open source distribuovaný systém riadenia verzií, ktorý je navrhnutý tak, aby zvládol všetko od malých až po veľmi veľké projekty s rýchlosťou a efektivitou [21].

5.2.7 JQuery

Samotný obsah webstránky som si vytvoril pomocou HTML. Servrovú časť mi zase zabezpečil javascript. Aby aplikácia bola dynamická a tieto dve časti medzi sebou interagovali, musel som využiť nástroj JQuery.

JQuery je ľahká cross-browser javascript knižnica, ktorá kladie dôraz na interakciu medzi javascriptom a HTML. Vo spojej práci ju využívam hlavne na objavenie a ukrytie HTML elementov na základe aktivity užívateľa.

Aby som knižnicu mohol vo svojej práci použiť, potreboval som ju stiahnuť alebo zahrnúť do kódu prostredníctvom odkazu. Vybral som si druhú možnosť a prostredníctvom skriptu javascriptu som zahrnul odkaz na JQuery v HTML súboroch, ktoré ju využívali. Aj z tohoto dôvodu je nutné pripojenie na internet pre správne používanie a chod aplikácie.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1
             /jquery.min.js"></script>
```

Príkladom základného využitia JQuery v aplikácii je zobrazenie a skrytie formulára pre editáciu dát na základe správneho ID.

```
function hide_form(){
     $('#two').hide();
}

function show_form(){
    $('#two').show();
}
```

Najzásadnejšie využitie JQuery bolo pri zobrazení dát. Vďaka odlišnému počtu zobrazených stĺpcov a počtu záznamov, bolo treba dynamicky modelovať tabuľku, ktorá dokáže v správnom formáte tieto dáta zobraziť. Dynamicky som vytváral HTML kód uložený v premennej, ktorý sa na záver zobrazil užívateľovi práva vďaka JQuery knižnici. Tento komplikovanejší algoritmus je rozdelený do viacerých funkcí, preto nemá zmysel ho tu uvádzať.

5.2.8 Nodemon

Aplikáciu vyvíjam v prostredí Netbeans. Užívatelia ale toto vývojové prostredie nepotrebojú a chcú si spustiť program jednoducho, ideálne dvojklikom na jeden súbor. Pre sprostredkovanie tejto aktivity využívam Nodemon, ktorý dokáže spúšťať aplikáciu mimo vývojového prostredia.

Nodemon je nástroj, ktorý pomáha vyvíjať aplikácie založené na node.js automatickým reštartovaním aplikácie uzlov, keď sú zistené zmeny súborov v adresári. Nevyžaduje žiadne ďalšie zmeny v kóde alebo spôsobe vývoja. Pred využitím ho musí užívateľ nainštalovať na svoj lokálny počítač. Všetky potrebné informácie nájde v inštalačnej príručke B.

5.3 Architektúra

Pre pochopenie spojenia jednotlivých komponent je nutné nahliadnuť na celkovú architektúru a fungovanie aplikácie. Pre zobrazenie vzťahov jednotlivých častí mi poslúžil obrázok 5.1.

Ako je vidieť, všetko sa odohráva v lokálnom prostredí užívateľa. Aplikácia nevyžaduje žiadne pripojenie k externým staniciam a serverom.

Najdôležitejšiu úlohu má komponenta node.js, ktorá spúšťa kód javascriptu. Užívateľ tak pri štarte aplikácie spúšťa vlastne túto komponentu.

Všetko podstatné zo strany užívateľa sa odohráva v prehliadači. Nachádza sa tam jeho lokálna databáza, do ktorej si ukladá dáta. Prehliadač so zdrojovým kódom komunikujú prostredníctvom HTTP portu 9080.

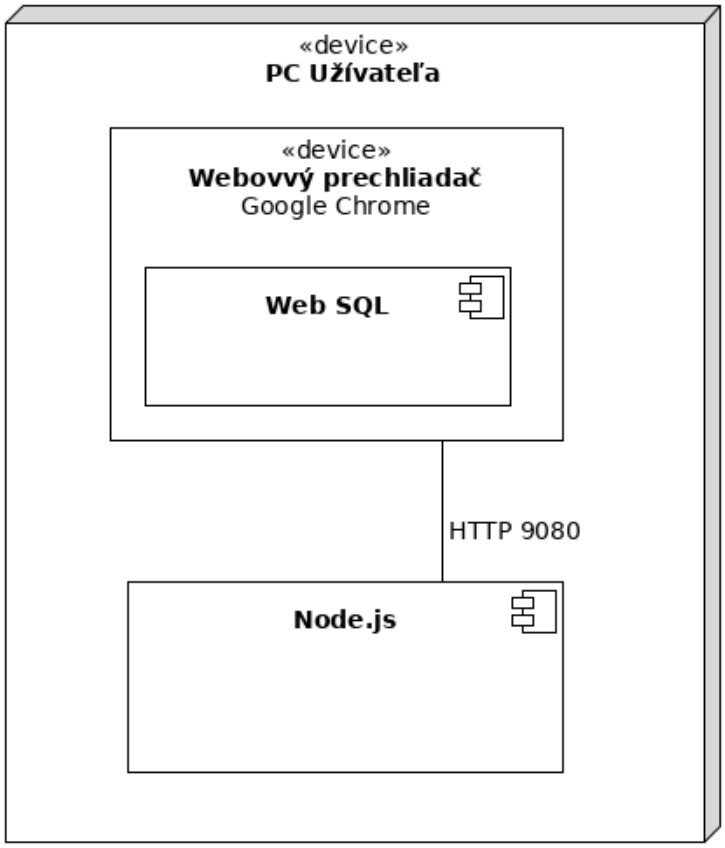
5.4 Obmedzenia

Ako každá aplikácia, aj tá moja so sebou prináša určité obmedzenia. Je to dané tým, že sa jedná o prototyp a niektoré časti neboli vyžadované, alebo na ich implementáciu a dlhodobejšie otestovanie nezostal čas.

Medzi hlavné obmedzenia patrí:

- **Inicializácia databázy:** Pred prvým spustením aplikácie na lokálnom zariadení je nutné inicializovať databázu. Nie je možné bez toho ihneď naimportovať súbor so vstupnými dátami. Inicializáciu vykoná užívateľ manuálnym pridaním jedného alebo viacerých záznamov do databázy. Hláška, že daný záznam bol úspešne pridaný je zároveň potvrdením úspešnej inicializácie databázy.

- **Uloženie dát v prehliadači:** Po skončení práce zostávajú dáta uložené v histórií prehliadača na lokálnom zariadení. Táto vlastnosť je zároveň požiadavkou klienta z dôvodu vyhnutia sa opakovanému nahrávaniu dát pri každom spustení aplikácie.
  Spomínaná vlastnosť ale môže byť nebezpečná. Toto obmedzenie preto



Obr. 5.1: Diagram nasadenia aplikácie.

upozorňuje na nutnosť manuálneho zmazania pamäte prehliadača pre trvalé zmazanie dát.

- **Pripojenie na internet:** Aplikácia využíva online odkazy pre použitie CSS šablón. Taktiež potrebuje internetové spojenie pre správne fungovanie JQuery príkazov. Do určitej mieri aplikácia funguje aj bez internetového pripojenia. Avšak nie je možné zaručiť jej plnú funkčnosť a správny grafický vzhľad.

- **Operačný systém Windows:** Z dôvodu spúšťacích skriptov pripravených pre Windows je oficiálne podporované spúšťanie aplikácie len na tejto platforme. Upravením skriptov je možné docieliť rozbehnutie aplikácie aj na ďalších platformách.

- **Prehliadač Google Chrome:** Práve z dôvodu využitia lokálnej API databázy priamo v prehliadači je nutné zaviesť toto obmedzenie. Oficiálne

je Web SQL podporovaná viacerými prehliadačmi. Plne otestovaná a overená funkčnosť je iba pre tento konkrétny typ.

- **Kontrola pridania záznamu:** Ďalšie z obmedzení, ktoré má za následok použitie Web SQL. Každé správne pridanie záznamu do databázy je potvrdené vyskakovacou hláškou v okne prehliadača. Prípadne je možnosť v časti SELECT skontrolovať správne pridanie záznamu výpisom všetkých dát. Niekedy sa stane, že z dôvodu nestabilného pripojenia, či iných dosiaľ neodladených vecí sa záznam nepridá. V tomto prípade sa nezobrazí potvrdzovacia hláška. Užívateľ tak musí opakovať pridanie záznamu.

- **Zobrazenie dát na spodku stránky:** Pri výbere dát z databázy sa samotné záznamy zobrazia pod elementom pokročilej podmienky výberu. Ak ich chce užívateľ vidieť, musí zarolovať nižšie na stránke. Táto vlastnosť bola objavená aj počas testovaní. Napokon sa zákazník rozhodol, že mu to takto plne vyhovuje a nepožadoval alternatívu.

- **Podmienka na select:** Ak chce užívateľ zobraziť dáta na základe podmienky, musí ju zadať do pripraveného políčka. Pre zobrazenie dát musí mať podmienka správny formát. Názov stĺpca musí byť presne zhodný s názvom v databáze. Hodnota parametru musí byť uprostred jednoduchých apostrofov. Podmienka podporuje logické operácie „AND“ a „OR“. Pre komplexnejšiu logiku sú vyžadované zátvorky „()“. Pri chybne zadanej podmienke sa žiadne dáta nezobrazia.
  Príklad použitia podmienky:

```
(id='5' and nazev_toku='Vltava') or (rok_odlovu='2019')
```

- **Import dát:** Nahranie dát je možné len zo súboru typu csv v správnom formáte. Tým sa rozumie formát, kde súbor obsahuje názvy všetkých stĺpcov oddelené bodkočiarkou. To isté platí aj o samotných dátach v ďalších riadkoch. Užívateľ bude upozornený na nevalidný súbor pri nahrávaní. Odporúčaná metóda je použiť výsledný súbor s exportom všetkých dát ako následný vstupný súbor pre import.

- **Hodnoty grafu:** Pre budúce možné používanie aplikácie pre iné stĺpce boli na žiadosť zákazníka definované stĺpce typu string. Graf zobrazenia počtu jedincov pracuje s číselnými hodnotami, ktoré prevádza z formátu string do formátu integer. Pokiaľ však užívateľ vyplní túto hodnotu nečíselným znakom, pre daný druh sa zobrazí nulová početnosť na grafe.

Kapitola 6

Testovanie

Na to, aby som mohol aplikáciu považovať za funkčnú a odovzdať ju zákazníkovi, bolo potrebné ju dôkladne otestovať na viacerých nezávislých užívateľoch. K tomu som potreboval určitú skupinu nezavislých užívateľov, ktorých som oboznámil s aplikáciou. Pre všetkých boli pripravené rovnaké testovacie scenáre.

6.1 Testovacie scenáre

Testovací scenár je v tomto prípade sled krokov, ktoré má za úlohu užívateľ vykonať k tomu, aby sme overili správnosť fungovania. Očakávam, že sa mi podarí odhaliť chyby, ktoré bude možné odstrániť pred finálnym dodaním aplikácie zákazníkovi.

6.1.1 Spustenie aplikácie

Prvý testovací scenár má za úlohu zistiť, či sú užívatelia schopní aplikáciu spustiť a nainštalovať podľa scenára v inštalačnej príručke B. Znakom úspešného dokončenia testu bolo zobrazenie domovskej obrazovky aplikácie. Týmto testom som odlaďoval inštalačné nedostatky a zjednodušoval proces samotnej inštalácie.

6.1.2 Pridanie záznamu

Scenár testuje schopnosť užívateľa pridať nový záznam do databázy vrátane vyplnenia formulára. Postup tohto testovacieho scenára zobrazuje tabuľka 6.1.

6.1.3 Editácia a zmazanie záznamu

Tento scenár je trochu obsiahlejší. Okrem zmeny a zmazania záznamu testuje aj pokus o vyhľadanie nevalidných záznamov. Jeho kroky popisuje tabuľka 6.2.

	Aktivita užívateľa	**Očakávaný stav aplikácie**
1	Prejsť na stránku pridania záznamu	Zobrazí sa prázdny formulár
2	Vyplniť prvých 5 stĺpcov	Vyplnených prvých 5 buniek
3	Vyčistiť hodnoty formulára príslušným tlačidlom	Prázdne bunky formulára
4	Zadávanie údajov za pomoci tabulátora	Vyplnené hodnoty formulára
5	Pridanie záznamu	Presunutie do potvrdzujúcej stránky

Tabuľka 6.1: TC - Pridanie záznamu.

	Aktivita užívateľa	**Očakávaný stav aplikácie**
1	Prejsť na stránku editácie	Zobrazí sa možnosť vyhľadať záznam
2	Zadať neexistujúce ID - 100	Upozornenie, že záznam neexistuje
3	Zadanie existujúceho ID - 1	Zobrazenie formulára s údajmi
4	Zmena hodnoty prvého stĺpca	Zmenený formulár
5	Uloženie zmeny	Informácia o vykonaní zmeny
6	Vyhľadanie rovnakáho záznamu	Zobrazenie zmenených údajov
7	Zmazanie daného záznamu	Overenie akcie zmazanie
8	Potvrdenie zmazania	Informácia o zmazaní
9	Vyhľadanie rovnakého záznamu	Upozornenie, že záznam neexistuje

Tabuľka 6.2: TC - Editácia a zmazanie záznamu.

6.1.4 Select a export dát

Najrozsiahlejší scenár testuje výber stĺpcov a podmienku pre select. Následne sa zaoberá správnym zobrazením dát a testuje export. Tabuľka 6.3 popisuje jeho podrobný priebeh.

6.1.5 Import dát

Krátky a jednoduchý testovací scenár. Za úlohu má overiť možnosť načítania dát zo strany užívateľa. Uvádza ho tabuľka 6.4.

6.1.6 Graf

Posledným scenárom je otestovanie funkcionality grafu pre vyjadrenie počtu jedincov daného druhu v roku zadaného ako parameter od užívateľa 6.5.

6.2 Výsledky testovania

Pre úspešné testovanie bolo potrebné vybrať vhodnú vzorku kandidátov. Keďže ide o špeciálnu aplikáciu na mieru pre zákazníka, bol aj on medzi tes-

	Aktivita užívateľa	**Očakávaný stav aplikácie**
1	Prejsť na stránku selectu	Zobrazí sa výber stĺpcov
2	Výber 3 náhodných stĺpcov	Zaškrtnuté 3 check-boxy stĺpcov
3	Zobrazenie vybraných dát	Dáta sa zobrazia užávateľovi
4	Použitie select all funkcie	Vyberú sa všetky stĺpce
5	Použitie unselect all	Odškrtnú sa všetky stĺpce
6	Výber 3 iných náhodných stĺpcov	Zaškrtnuté 3 check-boxy stĺpcov
7	Zadanie podmienky: autor='john'	Napísaná podmienka v aplikácii
8	Zobrazenie dát	Zobrazená podmnožina dát
9	Export dát	Dáta sa uložia na lokálnom disku
10	Zadanie mena súboru example.csv	Vyplnené meno súboru pre export
11	Export dát	Uloží sa súbor so zadaným názvom

Tabuľka 6.3: TC - Select a export dát.

	Aktivita užívateľa	**Očakávaný stav aplikácie**
1	Prejsť na stránku importu dát	Zobrazí sa možnosť nahrať súbor
2	Nahranie nevalidného súboru nic.png	Upozornenie, že sa dáta nenahrali
3	Nahranie validného súboru data.csv	Upozornenie, že existujúce dáta budú prepísané
4	Potvrdenie nahrania dát	Informácia, že dáta boli nahrané
5	Prejsť na stránku selectu	Zobrazí sa výber stĺpcov
6	Vybrať všetky stĺpce a potvrdiť	Zobrazia sa naimportované dáta

Tabuľka 6.4: TC - Import dát.

	Aktivita užívateľa	**Očakávaný stav aplikácie**
1	Prejsť na stránku grafu	Zobrazí sa políčko pre zadanie roku
2	Zadanie roku, ktorý nie je 2028 v databáze	Upozornenie, že žiaden záznam neexistuje
3	Zadanie existujúceho roku - 2002	Zobrazenie grafu

Tabuľka 6.5: TC - Graf.

termi spolu so svojim tímom. Ešte predtým som aplikáciu otestoval na svojich spolužiakoch, ktorí mi dali technické poznámky pre vylepšenie na základe testov.

Zatiaľ, čo výsledky testov od spolužiakov odhalili hlavne chybné časti kódu a samotnej implementácie, testy so zákazníkom mi poukázali na nejasné rozmiestnenie webových elementov a akcií. Po prvej sade testov som sa pokúsil chybné časti opraviť. Následne som s už menšou vzorkou testerov zopakoval dané testovacie scenáre.

Výsledky oboch druhov testov som spojil do jedného a ukázali mi nasledujúce nedostatky:

- **Nefunkčná podmienka:** Užívatelia dali najavo, že pokročilá podmienka pre výber podmnožiny dát nefunguje. Po jej zadaní im nezobrazovalo žiadané dáta. Ako sa ukázalo, podmienka funkčná bola. Chybná syntax podmienky spôsobila, že žiadne dáta neboli vybrané. Z tohto dôvodu som pridal znalosť syntaxe podmienky ako obmedzujúce kritérium pre používanie danej funkcie 5.4.

- **Nepresná „Select All“ funkcia:** Pri pokuse zaškrtnúť všetky stĺpce pri výbere dát sa neoznačili posledné tri. Chybu v kóde som našiel, opravil a otestoval.

- **Nefunkčný import dát:** Túto chybu sa podarilo odhaliť vďaka tomu, že niektorí testeri prehodili testovacie scenáre, a začali rovno importom dát. Ten, ako sa ukázalo, nie je možný bez predchádzajúcej inicializácie databázy. Toto odhalenie prinieslo nové obmedzujúce kritérium používania 5.4.

- **Chýbajúci graf:** Tretina užívateľov uviedla, že po zadaní roku sa im nezobrazuje daný graf. Ukázalo sa, že zadávali rok, ku ktorému nebol v databáze priradený žiaden záznam. Tento problém som vyriešil jednoducho. Pridal som hlášku, ktorá informuje užívateľa o neexistencií záznamov daného roku v databáze.

- **Vlastný názov exportného súboru:** Tu sa nejedná o chybu zistenú prvotnými testami. No vďaka nim prišla požiadavka od zákazníka na možnosť zadania vlastného názvu súboru pri exporte dát. Počas implementácie som sa rozhodol zanechať aj pôvodný variant vopred definovaného názvu v prípade, kedy k zmene nepríde.

- **Nefunkčné pridanie záznamu:** Užívatelia uviedli, že občas sa ich záznam pri zadávaní nepridá do databázy. Tento problém sa objavoval aj mne. Pokúšal som sa ho opraviť, no stále to nefunguje stopercentne. Ako preventívny manéver som pridal informačnú hlášku, ktorá potvrdí pridanie. Bez jej zobrazenia nie je nový záznam uložený. Túto občas

sa vyskytujúcu chybu a nutnosť zobrazenia potvrdzujúcej hlášky som pridal ako jedno z obmedzení používania aplikácie 5.4.

- **Prepis dát pri importe:** Pri nahraní nových dát sa aktuálne záznamy vymažú a nahrania novými. Jedná sa o želanú vlastnosť, o ktorej väčšina testerov nebola informovaná. Do budúcna som to vyriešil pridaním upozornenia na túto operáciu a nutnosťou potvrdiť premazanie dát.

Toto je zoznam najhlavnejších chýb a pripomienok vyplývajúcich z procesu testovania. Ten obsahoval ešte pár ďalších pripomienok, ktoré som dokázal jednoducho opraviť a nemusím ich tu teda rozoberať. Aj tu sa ukázalo, aký má testovanie obrovský význam napriek tomu, že sa pri vývoji softvéru často zanedbáva.

Kapitola 7

Ekonomické zhodnotenie a ďalší rozvoj aplikácie

Aplikácia bola navrhnutá, implementovaná a dôsledne otestovaná. Zostáva mi už len navrhnúť a odporučiť ďalší rozvoj aplikácie a zhodniť finančnú náročnosť.

7.1 Budúci rozvoj

Táto aplikácia je prvým prototypom nástroja pre uľahčenie spracovávania dát biotopu rýb. Je obrovské množstvo vecí, ktoré môžu byť dokončené, a ktorými môže byť aplikácia obohatená.

7.1.1 Grafy

Najväčší priestor pre vývoj je učite v oblasti zobrazenia grafov. Vo svojej aplikácii som vytvoril jeden ukážkový graf pre potreby zákazníka. Prvotný cieľ aplikácie bol totiž dáta predspracovávať pre následné použitie grafickými nástrojmi.

Vytvoril som však základ, do ktorého by nemal byť problém pridávať ďalšie množstvá grafov. Mohlo by to viesť až k momentu, kedy nebude dáta nutné posielať do grafických nástrojov, ale využívať priamo túto aplikáciu. K tomu je ale potrebná dôkladná analýza využívania dát a vhodné navrhnutie parametrov pre typ grafu a spôsob vyjadrenia hodnôt.

Ako prvý by mohol byť naimplementovaný graf alfa a beta diverzity, o ktorý už zákazník do budúcna prejavil záujem. Tento graf označil ako často používaný a teda odobril zmysel jeho pevnej implementácie v rámci aplikácie.

7.1.2 Dynamické názvy stĺpcov a dátových typov

Momentálne aplikácia obsahuje pevne definované stĺpce podľa požiadaviek zákazníka a k nim aj definový rovnaký dátový typ.

Všestrannejšie využitie aplikácie by mohlo priniesť vytváranie stĺpcov priamo z webového prostredia. To si vyžaduje implementáciu editoru na správu stĺpcov. Náročnejšia operácia, ktorá by dodala aplikácii širšiu využiteľnosť. Minimálne ďalšie živočíšne druhy, ale aj chemické a iné hodnoty by sa mohli prostredníctvom nej zaznamenávať a spracovávať.

7.1.3 Validácia hodnôt

Pre plnohodnotné fungovanie grafov a ďalších funkcií je v budúcnosti nevyhnutné zaviesť dátovú validáciu stĺpcov.

Tá poslúži ako prevencia proti zadávaniu nevalidných údajov. Predispozíciou je samozrejmosť možnosti určiť dátové typy.

Pri tvorbe validačných pravidiel je ale potrebná dôkladnejšia analýza. Pri tak veľkom počte potrebných vyplnení záznamov, ako obsahuje súčasná verzia aplikácie, môže byť pre užívateľa otravné nekonečné množstvo nutných úprav. Pre dokonalejšiu konzistenciu dát však bude nutné nájst rozumný kompromis.

7.1.4 Skupinové operácie

Ďalšie zlepšenie aplikácie by mohla priniesť implementácia skupinových operácii pri práci so záznamami.

Aktuálny proces mazania záznamov nie je príliš efektívny. Užívateľ musí vyhľadať záznam pod jeho ID, zobraziť ho a následne tak ho má možnosť zmazať. Vymazanie viacerých záznamov mu tak zaberie čas. Proces zmazania väčšieho množstva záznamov v mojej aplikácii vyžaduje vyexportovanie dát, zmazanie dát v súbore a následný import späť do aplikácie.

Práve implementácia skupinových operácii hromadného mazania záznamov užívateľov prispeje ku skvalitneniu práce. Podobné funkcie je možné vymyslieť aj pre hromadnú editáciu, či pridávanie záznamov.

7.1.5 Webový server

Najzásadnejšou zmenou a posunom vpred by bolo opustenie lokálnych inštancií aplikácie. Nasadenie sytému na webový server by uľahčilo užívateľom prístup z viacerých zariadení bez nutnosti kopírovania obsahu aplikácie.

Táto zmena si bude vyžadovať hlbší zásah do štruktúry a viaceré úkony:

- **Ochrana dát:** Tieto jedinečné údaje sú relatívne v bezpečí na lokálnej verzii. Pokiaľ však bude aplikácia bežať na webovom serveri, bude nutné riešiť otázku bezpečnosti.

- **Užívatelia a role:** Opustenie lokálnej verzie by znamenalo nutnosť vytvoriť prihlasovanie jednotlivých užívateľov a spravovanie rozličných rolí a prístupov. Výhodou by bolo možnosť zdielania dát medzi užívateľmi patriacimi do rovnakej skupiny.

7.1.6 Finančné zhodnotenie

Momentálne stav vytvárania a rozvoja aplikácie je finančne nenáročný. To má za následok aktuálnu kvalitu a stav aplikácie.

Vyžaduje si jedného človeka v roli analytika, ktorý zbiera požiadavky od zákazníka a vytvára celkový návrh. Ten istý človek musí zvládnuť samotné naprogramovanie aplikácie a ideálne aj jej otestovanie.

V prípade mnou odporúčaného rozvoja by náklady niekoľkonásobne narástli a využili by sa hlavne na:

- **Prevádzkovanie:** Bude nutné zabezpečiť webový hosting tak, aby vyhovoval a spĺňal požiadavky aplikácie.
- **Analytika:** Ktorý by na základe diskusií so zákazníkom vytváral logiku a architektúru zložitejších celkov.
- **Vývojárov:** Menšiu skupinu vývojového tímu, ktorý by mal nastarosť vývoj aplikácie na základe podkladov od analytika.
- **Testerov:** Skupinu ľudí najímanú jednorázovo pre otestovanie menších a väčších častí aplikácie pri nasadzovaní nových verzií a vylepšení.
- **Podporu:** Určitú mieru zákazníckej podpory, ktorú využijú užívatelia, ak nastanú problémy pri behu aplikácie, alebo nebudú schopní prihlásiť sa do svojho užívateľského účtu.

Možnosti napredovania sú viaceré. Od drobného rozvoja pre aktuálneho zákazníka, až po snahu urobiť aplikáciu komplexnou a rozšíriť ju tak medzi viacerých klientov. Všetko ale bude záležať od kvality jej spracovania a záujmu zo strany potencionálnych zákazníkov.

Záver

Cieľom práce bolo navrhnúť a vytvoriť aplikáciu pre pani Petru Horkú z Univerzity Karlovej v Prahe, oddelenia Ústavu pre životné prostredie. Tá so svojim tímom spracováva veľké množstvo vzácnych historických dát, ktoré boli roztrúsené bez jednotnej štruktúry po viacerých nejednotných úložiskách.

Zadanie z jej strany bolo vytvoriť pomocnú aplikáciu, ktorá bude schopná dáta načítať v jednotnej forme a prípraví ich pre následné použitie ďalším nástrojom. Po celý čas bol dôraz kladený hlavne na bezpečnosť záznamov.

Počas niekoľkých sedení sme si postupne ujasňovali výsledný formát aplikácie. Dominovali diskusie o funkciách, ktoré by mala aplikácia spĺňať. Dôležité bolo vytvoriť aplikáciu na mieru tak, aby sa dobre používala podľa ich potrieb. Implementačný jazyk a samotný spôsob implementácie bol na mojom výbere.

Po úspešnom zbere požiadaviek som sa pustil do samotnej analýzy. Vytvoril som doménový model, ktorý by bol schopný najlepšie zachytiť rozmanitú štruktúru dát. Následne mi diagramy aktivít pomohli vykresliť základné procesy budúcej aplikácie. Prípady užitia už len došpecifikovali možnosti užívateľa a dotvorili celkový náhľad na funkcie. Ukázalo sa, že pre ľudí mimo IT sú tieto diagramy najzrozumiteľnejšie a pomáhajú pochopeniu požiadaviek.

V návrhovej časti práce som sa zameral na vytvorenie logiky webovej časti tak, aby čo najviac vyhovovala samotnému užívateľovi. Snažil som sa odhadnúť správanie užívateľa z tohto prostredia podľa jeho potrieb.

Najdôležitejšia bola časť samotnej implementácie. Dôležité tiež bolo zvoliť vhodné metódy pre vytvorenie a používanie jednotlivých komponent. Práve počas implementácie som si prakticky vyskúšal, ako fungujú jednotlivé časti po spojení do funkčného celku. Zdokonalil som sa v tvorení CSS, ako aj tradičnom HTML. Vyskúšal som si implementáciu na dvojici klient server, kde som si mal možnosť vyskúšať samotný javascript.

Zabrať mi dala aj voľba databázy, ktorá by bola ľahko lokálne dostupná, bezpečná a jednoducho spravovateľná užívateľom. Nakoniec som zvolil pre mňa úplne novú metódu využitia webového API pre Web SQL databázu.

Pri tvorbe architektúry bolo potrebné, aby jednotlivé časti spolu komunikovali. Node.js, ktorý mal na starosti spúšťanie javascriptu, komunikuje s prehliadačom pomocou HTTP.

Po implementácii bolo nevyhnutné aplikáciu otestovať. Na to som využil skupinu okolo zákazníka, ale aj nezávislú skupinu spolužiakov. Už prvé testy na základe testovacích scenárov ukázali niektoré nedostatky. Tie, ktoré bolo možné odstrániť, som preprogramoval a následne znovu otestoval. Nedostatky väčších rozsahov som spojil do jedného celku a vytvoril obmedzenia aplikácie.

Dúfam, že tento úvodný prototyp aplikácie sa bude využívať v praxi čo najväčším počtom užívateľov. Zároveň si želám jeho ďalší rozvoj, pre ktorý som vytvoril zoznam funkcí a návrh rozvoja architektúry, aký by mohol tomuto budúcemu rozvoju pomôcť. Zároveň som zachytil, čo všetko to bude finačne obnášať. To môže pomôcť pri ďalšom rozhodovaní sa, ako prípadne aplikáciu rozvinúť.

Zároveň túto aplikáciu a samotnú diplomovú prácu vnímam ako jedno z prvých spojení dvoch diametrálne odlišných vysokých škôl. Rozšíril som si obzory vedomostí o špecifikácie a obory, ktorými sa zaujímajú v oblasti prírody. Do budúcna dúfam v rozšírenie spolupráce medzi našimi študentami a niektorou z univerzít, ako pomoc a podpora v oblasti informačných technológii.

Literatúra

[1] Šimko, A.: *Stručný úvod do špecifikácie požiadaviek*. 2019, [cit. 2019-04-12]. Dostupné z: http://dai.fmph.uniba.sk/~simko/teaching/srs/

[2] admin: *Nefunkčné systémové požiadavky: koncepty a príklady*. 2019, [cit. 2019-04-18]. Dostupné z: https://sk.flipperworld.org/pc/nefunkcne-systemove-poziadavky-koncepty-a-priklady

[3] CDI: *Systémová analýza*. 2019, [cit. 2019-04-17]. Dostupné z: http://www.cdi.cz/vyvoj-software-na-zakazku/sluzba/systemova-analyza/

[4] pavus: *UML - úvod*. 2005, [cit. 2019-04-12]. Dostupné z: http://mpavus.wz.cz/index.php

[5] Valášek, S.: *Prehľad UML diagramov*. 2018, [cit. 2019-04-17]. Dostupné z: https://valasek.wordpress.com/2012/05/02/prehlad-uml-diagramov/

[6] Mlejnek, I. J.: *Modelování obchodních procesu*. 2011, [cit. 2019-04-12].

[7] Rejnková, P.: *Diagram aktivit*. 2009, [cit. 2019-04-17]. Dostupné z: http://uml.czweb.org/diagram_aktivit.htm

[8] Rejnková, P.: *Diagram případů užití*. 2009, [cit. 2019-04-17]. Dostupné z: http://uml.czweb.org/pripad_uziti.htm

[9] Management Mania: *Architektúra klient-server*. 2016, [cit. 2019-04-12]. Dostupné z: https://managementmania.com/sk/architektura-klient-server

[10] Čermák, M.: *Vícevrstvá architektura: tenký, tlustý a chytrý klient*. Clever and Smart, 2012, [cit. 2019-04-14]. Dostupné z: https://www.cleverandsmart.cz/vicevrstva-architektura-tenky-tlusty-a-chytry-klient/

[11] Humaj, I. P.: *V čom je napísaná D2000-ka?* IPESOFT, 2017, [cit. 2019-04-16]. Dostupné z: https://www.ipesoft.com/sk/blog/vyber-programovacieho-jazyka-pre-realtime-systemy

[12] Soltészová, D.: *Manažment kvality a testovanie softvéru ako súčasť webovej integrácie*. Webová Integrace, 2012, [cit. 2019-04-16]. Dostupné z: http://www.web-integration.info/cs/blog/manazment-kvality-a-testovanie-softveru-ako-sucast-webovej-integracie/

[13] Mlejnek, I. J.: *Testování aplikací*. 2011, [cit. 2019-04-16].

[14] Zones SK: *Akceptačné testovanie IS*. 2014, [cit. 2019-04-17]. Dostupné z: https://www.zones.sk/studentske-prace/informatika/8634-akceptacne-testovanie-is/

[15] Soltészová, D.: *Vlastnosti dobrého testera*. Webová integrace, 2012, [cit. 2019-04-17]. Dostupné z: http://www.web-integration.info/cs/blog/vlastnosti-dobreho-testera/

[16] Martinek, M.: *Úvod do SQLite a příprava prostředí*. IT Network, 2019, [cit. 2019-04-21]. Dostupné z: https://www.itnetwork.cz/sqlite/sqlite-tutorial-uvod-a-priprava-prostredi

[17] Sharp, R.: *Introducing Web SQL Databases*. HTML5Doctor, 2010, [cit. 2019-04-20]. Dostupné z: http://html5doctor.com/introducing-web-sql-databases/

[18] Aston, B.: *A brief history of JavaScript*. Medium Corporation, 2015, [cit. 2019-04-20]. Dostupné z: https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17

[19] MDN web docs: *HTML: Hypertext Markup Language*. 2019, [cit. 2019-04-23]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[20] MDN web docs: *Introduction to CSS*. 2019, [cit. 2019-04-23]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS

[21] git: *Getting Started - Git Basics*. 2019, [cit. 2019-04-13]. Dostupné z: https://git-scm.com/book/en/v1/Getting-Started-Git-Basics

Dodatok **A**

Zoznam použitých skratiek

CSV Comma-Separated Values

GUI Graphical user interface

CSS Cascading Style Sheet

HD Hardware

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

ID Identifier

JSON JavaScript Object Notation

SAS Statistical Analysis System

SIA Stable Izotope Analysis

SQL Structure Query Language

UML Unified Modeling Language

URL Uniform Resource Locator

Dodatok B

Inštalačná príručka

Druhý dodatok diplomovej práce obsahuje pokyny pre každého, kto si chce aplikáciu lokálne nainštalovať. Zároveň tu nájde inštrukcie k prvému spusteniu a správnej inicializácii databázy.

B.1 Inštalácia aplikácie

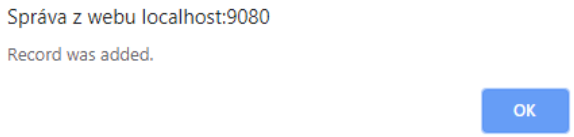
K tomu, aby mohol užívateľ začať plnohodnotne aplikáciu využívať, je nutné vykonať nasledujúcu sekvenciu krokov:

1. **Rozbalenie zip archívu:** Táto možnosť platí pre tím zákazníka, ktorý obdrží aplikáciu vo formáte zip. Je potrebné súbor **fish_application.zip** rozbaliť lokálne vo svojom poočítači.
   Pre potreby kontroly tejto práce je nutné si z priloženého CD prekopírovať zložku **fish_application** do svojho počítača.

2. **Inštalácia nodemonu a sqlite3 modulu:** Inštalácia nodemonu sa vykoná automaticky spustením jedného skriptu. Ten zároveň obsahuje aj príkaz k inštalácii sqlite3 modulu. Táto inštalácia sa vykoná spustením skriptu **fish_application/install/install_nodemon.bat**. Otvoria sa dve windows konzoly, v ktorých inštalácia prebehne. Po jej skončení musíme konzoly manuálne zatvoriť.

3. **Inštalácia node.js:** Spustením msi súboru inštalačného programu **fish_application/install/node-v10.15.3-x64.msi** sa spustí sprievodca inštaláciou, ktorú je potrebné úspešne dokončiť.

4. **Spustenie a ukončenie aplikácie:** Spôsob spustenia aplikácie je veľmi jednoduchý. Stači spustiť skript **fish_application/start.bat** a otvorí sa konzola, ktorú je nutná nechať bežať. Následne sa zadaním URL s adresou aplikácie **http://localhost:9080/main_page.html** v prehliadači zobrazí domovská stránka C.1.



Obr. B.1: Potvrdenie pridania záznamu.

Pre ukončenie aplikácie je nutné spustiť **fish_application/stop.bat** skript, alebo jednoducho zatvoriť otvorenú konzolu.

B.2 Prvotná inicializácia databázy

Pred prvým spustením aplikácie je nutné databázu inicializovať. To sa vykoná nasledujúcim spôsobom:

1. Presuniem sa na stránku pridania záznamu.
2. Zadám náhodné hodnoty záznamu.
3. Pri pridaní záznamu očakávam hlášku B.1. Tá znamená úspešné pridanie záznamu. Pokiaľ sa nezobrazí, skúšam pridávať záznamy znova, prípadne reštartujem aplikáciu.

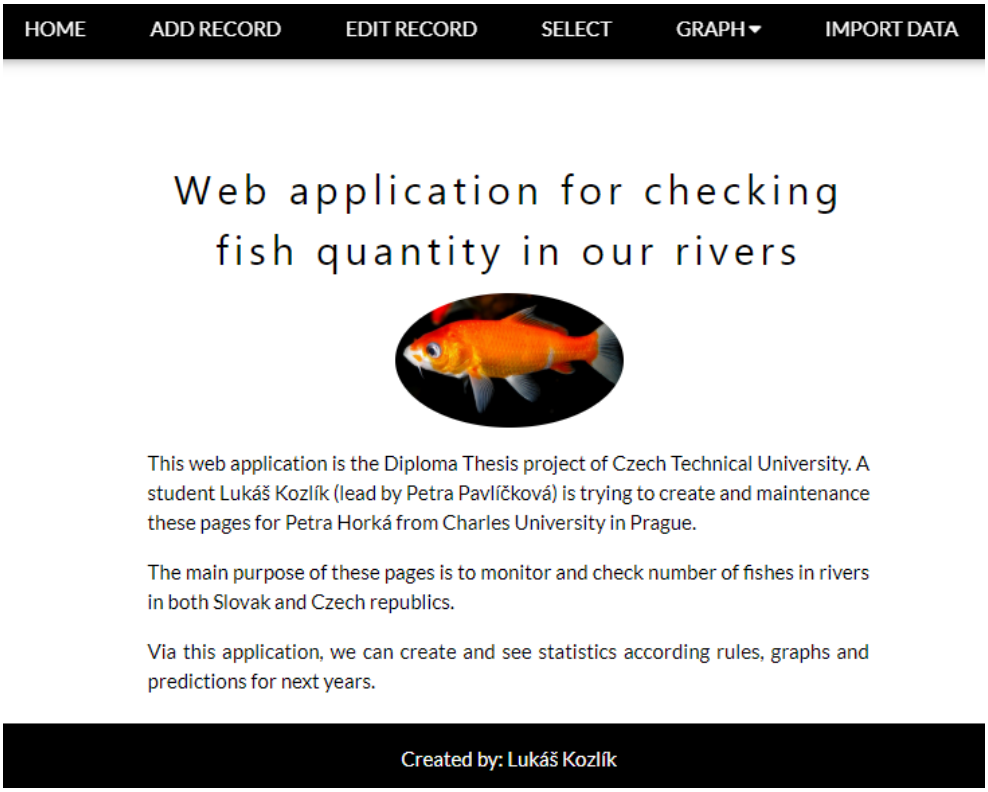
Dodatok C

Ukážky aplikácie

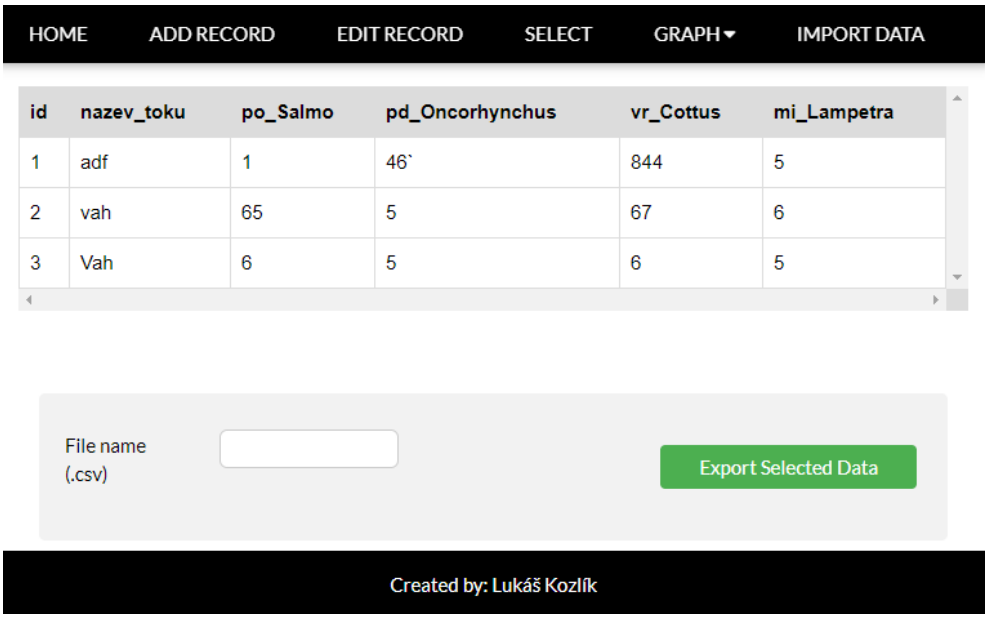
Táto príloha obsahuje ukážky z hlavnej aplikácie. Slúži pre predstavu čitateľa práce o aplikácii bez toho, aby si ju nutne musel spustiť.

Na nasledujúcich stránkach prinášam ukážky, ktoré zobrazujú:

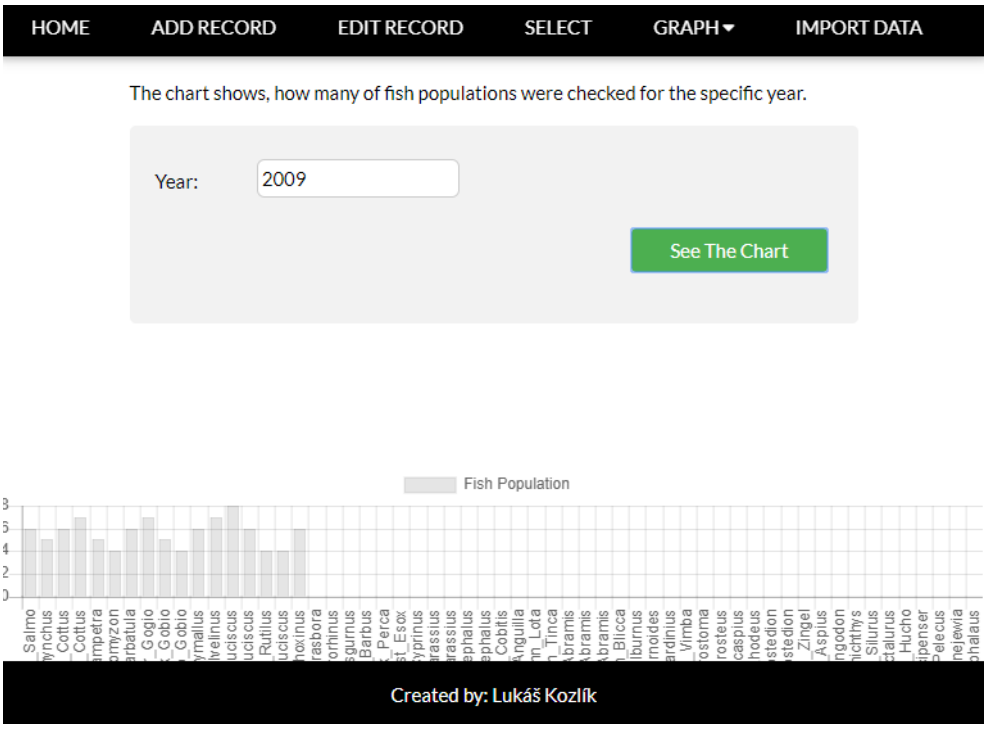
- **Hlavnú stránku** C.1. Tak ako ju vidí užívateľ pri prvotnom spustení aplikácie.
- **Zobrazenie dát** C.2. Názorné zobrazenie podmnožiny dát na základe podmienky zadanej užívateľom.
- **Graf výskytu jedincov** C.3. Príklad zobrazenia grafu počtu jedinecov pre všetky druhy rýb. Základným parametrov je rok odlovu.
- **Vyhľadanie záznamu** C.4. Grafické znázornenie prípadu, kedy užívateľ na základe ID vyberá záznam pre prípadnú editáciu a zmazanie.
- **Vyplnenie formuláru** C.5. Pohľad na webový formulár pri možnosti pridania nového záznamu.
- **Pokročilá podmienka** C.6. Zobrazenie použitia pokročilej podmienky v kombinácii s výberom stĺpcov z kategórie izotopy.



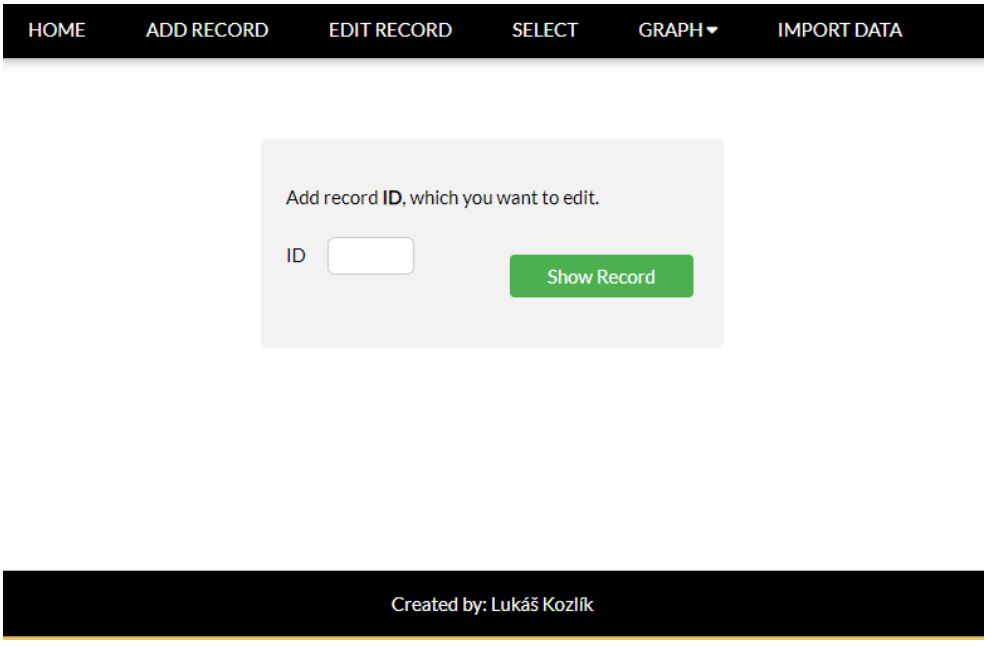
Obr. C.1: Hlavná obrazovka.



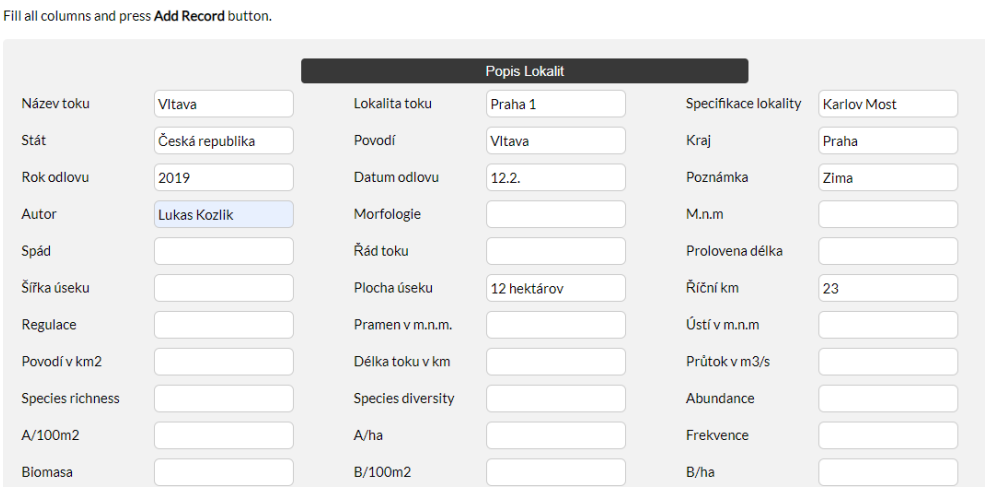
Obr. C.2: Zobrazenie dát.



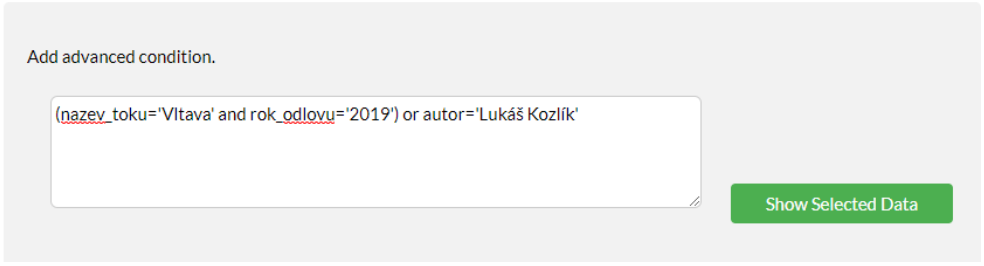
Obr. C.3: Graf počtu jedincov.



Obr. C.4: Vyhľadanie záznamu.



Obr. C.5: Vyplnenie formulára.



Obr. C.6: Pokročilá podmienka.

Dodatok D

Obsah priloženého CD

- `readme.txt` ..... Stručný popis obsahu CD
- `fish_application` .....
  - `install` .....
    - `install_nodemon.bat` ..... Inštalačný skript nodemonu
    - `node-v10.15.3-x64.bat` ..... Inštalačný program pre node.js
  - `start.bat` ..... Spúšťací súbor aplikácie
  - `app` .....
    - `project` ..... Zdrojové kódy aplikácie
  - `doc.html` ..... Dokumentácia zdrojového kódu
- `text` ..... Adresát s textom práce
  - `thesis.pdf` ..... Text práce vo formáte PDF
  - `thesis` ..... Zdrojová forma práce vo formáte LaTeX