VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÁ APLIKACE PRO VÝBĚR SEKVENAČNÍCH PRIMERŮ PRO AMPLIKONOVÉ SEKVENOVÁNÍ 16S RRNA

WEB APPLICATION FOR PRIMER SELECTION FOR 16S RRNA AMPLICON SEQUENCING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAN JURČA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. STANISLAV SMATANA**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání diplomové práce

Student:	**Jurča Jan, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Bioinformatika a biocomputing
Název:	**Webová aplikácia pre výber sekvenačných primerov pre amplikónové sekvenovanie 16S rRNA** **Web Application for Primer Selection for 16S rRNA Amplicon Sequencing**
Kategorie:	Bioinformatika

Zadání:

1. Naštudujte si problematiku určovania mikroorganizmov pomocou technológie 16s rRNA. Zamerajte sa na problematiku sekvenačných primerov a ich vlastností.
2. Preskúmajte existujúce nástroje na podporu výberu primerov, zhrnte ich výhody a nevýhody a na základe týchto zistení navrhnite nový nástroj.
3. Navrhnutú aplikáciu implementujte. Implementácia by mala byť užívateľsky prívetivá, umožňovať pokročilé filtrovanie, vizualizáciu dát a inteligentný výber primerovej kombinácie na základe obmedzení zadaných užívateľom.
4. Vyhodnoťte implementované riešenie, zhodnoťte naplnenie jednotlivých cieľov a navrhnite možné budúce zlepšenia.

Literatura:

- Na základe odporúčaní vedúceho

Při obhajobě semestrální části projektu je požadováno:

- Splnenie bodov 1 a 2 zadaní.

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

Vedoucí práce:	**Smatana Stanislav, Ing.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	18. května 2022
Datum schválení:	1. listopadu 2021

Abstrakt

Hlavním cílem této práce byl návrh a implementace webové aplikace, která je, na základě uživatelské volby, schopna ohodnotit primerové páry určené pro 16S rRNA amplikonové sekvenování a zjednodušit uživateli výběr primerového párů pro jeho specifické potřeby. Ohodnocení je realizováno na základě databáze primerových párů, která obsahuje data o specifitě a senzitivitě každého jednoho primerového páru. Součástí práce byla i výkonnostní optimalizace algortimu pro výpočet senzitivity a specificity a jeho integrace do aplikace. Díky čemuž aplikace umožňuje uživateli provést analýzu vlastního primerového páru a získat informace o pozici amplifikovaného regionu, senzitivitě a specificitě.

Abstract

Main goal of this thesis was an implementation of web application, that can evalutate and recommend primer pairs for 16S rRNA amplicon sequencing, according to users specific needs and by this simplify the procces of primer pair selection. Primer pair evaluation is based on database of primer pairs, which contains data about senzitivity and specifity of each primer pair. Part of the work was also the performance optimization of the primer pair analysis algorithm, that computes senzitivity and specifity data. This optimization helped in an integration of algorithm into the application, which means that users can submit their own primer pair sequences, run analysis and get informations about position of amplified region, sensitivity and specifity.

Klíčová slova

metagenomika, 16S rRNA, amplikonové sekvenování, PCR, primer, webová aplikace

Keywords

metagenomic, 16S rRNA, amplicon sequencing, PCR, primer, web application

Citace

JURČA, Jan. *Webová aplikace pro výběr sekvenačních primerů pro amplikonové sekvenování 16S rRNA*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Stanislav Smatana

Webová aplikace pro výběr sekvenačních primerů pro amplikonové sekvenování 16S rRNA

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Stanislava Smatany. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Jurča
16. května 2022

Poděkování

Chtěl bych velmi poděkovat mému vedoucímu práce Ing. Stanislavu Smatanovi, za jeho ochotu, pomoc, cenné rady a investovaný čas, který mi poskytl při odborném vedení této diplomové práce.

Výpočetní zdroje byly poskytnuty projektem e-Infrastruktura CZ (e-INFRA CZ LM2018140) podporovaným Ministerstvem školství, mládeže a tělovýchovy České republiky.

Obsah

Kapitola 1

Úvod

Jednou z možností studia populační diverzity bakteriálních společenství v rámci metagenomického studia, je klasifikace sekvencí získaných pomocí amplikonového sekvenování genu 16S rRNA. [73]

Amplikonové sekvenování je založeno na sekvenaci specifického regionu 16S rRNA genu, který je amplifikován pomocí reakce PCR [56]. Gen 16S rRNA je vysoce konzervovaný napříč bakteriální a archeální taxonomickou doménu [41]. Díky tomu je možné jediným primerovým párem, amplifikovat velkou podmnožinu všech druhů z těchto domén. Následně na základě variabilních regionů genu lze provést taxonomickou klasifikaci získaných sekvencí.

I přes vysokou konzervovanost genu, ale není možné jediným primerovým párem zaručit amplifikaci všech druhů. Skupina amplifikovaných druhů je primárně ovlivněna pozicí regionu, který je při PCR reakci zacílen. Množství amplifikovaných (zachycených) druhů, lze považovat za senzitivitu primerového páru. Zatímco specificita se odvíjí od množství rozlišitelných zachycených druhů. Pozice regionu, potažmo senzitivita a specificita, závisí na použitém primerovém páru. [54, 64]

Právě z toho důvodu je hlavním cílem této práce návrh a implementace webové aplikace, která uživateli na základě rozsáhlé databáze primerových párů zjednoduší proces výběru ideálního primerového páru. Aplikace musí být schopna, na základě uživatelského dotazu ohodnotit senzitivitu a specificitu všech primerových párů v databázi, provést vizualizaci výsledků a zjednodušit uživateli výběr takového primerového páru, který nejvíce vyhovuje jeho požadavkům.

Databáze aplikace by měla obsahovat podrobné informace o senzitivitě a specificitě tisíců primerových párů. Tyto informace jsou vypočítány pro každý primerový pár pomocí existujícího analytického algoritmu.

Cílem práce je mimo jiné i výkonnostní optimalizace a integrace analytického algoritmu přímo do webové aplikace. Což umožní uživatelům aplikace, provést analýzu jejich vlastních primerových sekvencí. Uživatelé díky tomu získají informace o pozici amplifikovaného regionu, senzitivitě a specificitě vloženého primerového páru. Zároveň uživatel bude moci tento nový primerový pár porovnat vůči ostatním párům v databázi.

Hlavní přínos této práce je především vytvoření veřejně dostupné webové aplikace, která bude obsahovat mechanizmy a algoritmy pro pokročilou, ale přesto uživatelsky přívětivou práci s primerovými páry a jejich sekvencemi.

Kapitola 2

Molekulární genetika a metagenomika

Molekulární genetika se jako věda zabývá strukturou genetické informace, genomu a popisem dějů na úrovni molekulární biologie [72]. Pro celistvé pochopení problematiky této práce, je nutné se seznámit se základními pojmy a principy biologie, molekulární genetiky a metagenomiky.

Prvním důležitým pojmem je **Taxonomie** [32]. V biologickém slova smyslu se taxonomie zabývá popisem, identifikací a klasifikací organismů. Cílem taxonomie je klasifikace všech známých biologických skupin (**taxonů**) do jednotné hierarchie biologických kategorií. Samotná taxonomická hierarchie sestává z několika úrovní z nichž hlavních sedm je: doména (domain), říše (kingdom), kmen (phylum), třída (class), řád (order), čeleď (family), rod (genus), druh (specie). Mezi těmito hlavnímy kategoriemi jsou ještě definovány mezistupňové kategorie jako například nadtřída, podřád, infrařád a podobně. Z informatického hlediska je taxonomická hierarchie stromová struktura, kde biologické kategorie pojmenovávají úrovně tohoto stromu a kde taxony jsou jednotlivými uzly stromu.

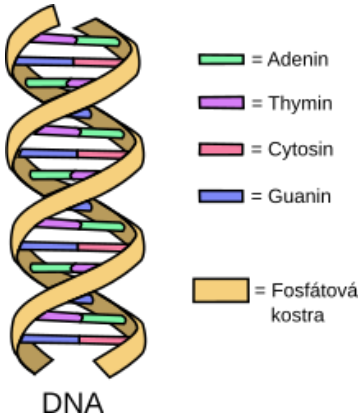
Každý organismus v taxonomické hierarchii lze identifikovat na základě genetické informace.

2.1 Genom jako nositel informace

Základním stavebním kamenem každého živého organismu je sada genetických informací, které určují vlastnosti, schopnosti a podobu buňky. Kompletní sada genetických informací dané buňky je souhrnně označena jako genom. U většiny organismů je genom tvořen dlouhými vlákny molekul DNA, respektive RNA u některých virů, do nichž je zakódována veškerá genetická informace daného organismu. Ve vláknech DNA, respektive RNA, jsou obsaženy veškeré kódující oblasti ve formě genů. [38]

Gen je základní jednotka genetické informace a tedy základní jednotka dědičnosti. Jedná se o sekvenci nukleotidů kódujících produkt. Tímto produktem je transkribovaná sekvence RNA, která může, ale nutně nemusí, kódovat protein. [38]

DNA, neboli deoxyribonukleová kyselina, je nukleová kyselina, do jejíž struktury je zakódovaná genetická informace. DNA je v buňce standardně ve formě dvoušroubovice vzájemně



Obrázek 2.1: Ilustrace struktury vlákna dna. Obrázek převzat ze serveru Wikimedia Commons.

komplementárních vláken. Vlákno DNA je složeno ze čtyř druhů nukleotidů (bází). Dvou pyrimidinových bází cytozinu (C) a thyminu (T) a dvou purinových bází adeninu (A) a guaninu (G) [36]. Ilustrace dvoušroubovice DNA je zobrazena na obrázku 2.1.

Jedno vlákno DNA je sekvencí tvořenou z jednotlivých bází. Dvě vlákna tvoří dvoušroubovici DNA, kde jednotlivé báze, jsou spojeny vodíkovou vazbou podle pravidel Watson–Crickova párování takovým způsobem, že adenin se váže na thymin a cytozin se váže na guanin. V důsledku tohoto párovaní jsou obě vlákna vzájemně komplementární, díky čemuž je dostačující znát sekvenci jednoho vlákna, abychom byli schopni dopočítat druhé.

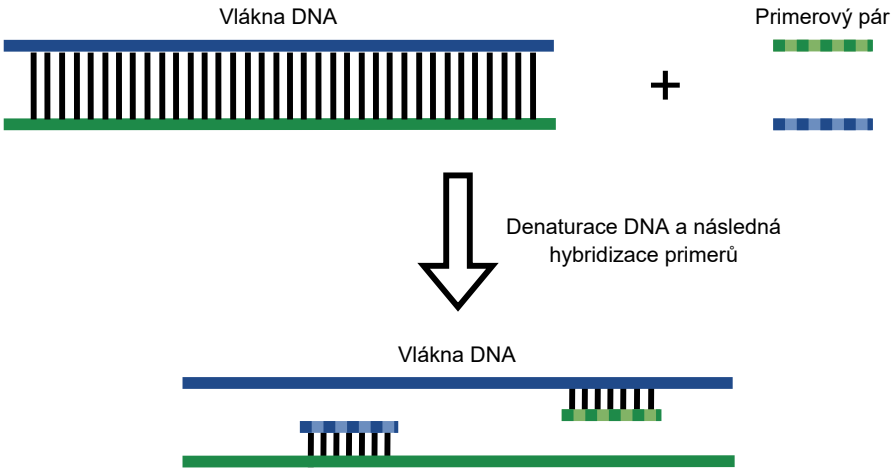
RNA, neboli ribonukleová kyselina, je nukleová kyselina, která v buňkách vystupuje jako produkt transkripce DNA, pouze u RNA virů RNA zastupuje funkci DNA a přímo tvoří genom daného viru. RNA se běžně vyskytuje v buněčné cytoplazmě. Stejně jako DNA je i RNA tvořeno sekvencí nukleotidů. Na rozdíl od DNA zde není zastoupen nukleotid thymin, ale je nahrazen uracilem (U), přičemž ostatní tři nukleotidy jsou v RNA sekvenci obsaženy beze změny. Na rozdíl od DNA se RNA ve většině případů vyskytuje pouze ve formě jednovláknové sekvence. [35]

Různé typy RNA jsou součástí mnoha různých procesů v buňce. Základním procesem jímž je konkrétně mRNA, tRNA a rRNA (messenger RNA, transfer RNA a ribosomal RNA) součástí, je syntéza proteinů. Krom toho ale RNA zastává i funkce, které nekódují žadný protein. Konkrétně se může jednat procesy regulující transkripci, replikaci chromozomů, degradaci proteinů a mnohé další. Souhrnně se tyto typy RNA označují jako ncRNA (non-coding RNA). [68]

Biologické procesy genomu

Nukleové kyseliny jsou základním držitelem informace živých organismů a umožňují jejich existenci a fungování. Interpretace informace uložené v DNA, potažmo RNA, je umožněno procesy, které realizují převod genetické informace z DNA na RNA (transkripce) a překlad RNA na protein (translace).

Transkripce [36] je přepis genetické informace z molekuly DNA na RNA. V naprosté většině případů se jedná o přepis jednoho genu, tedy jeho expresi. Transkripce genových



Obrázek 2.2: Schématická ilustrace hybridizace primerového páru na komplementární denaturovaná vlákna DNA.

sekvencí, je velmi komplexní proces regulován řadou transkripčních faktorů, které reagují na aktuální potřeby buňky a regulují expresi jednotlivých genů. Transkripce je vratný proces, což znamená, že chemickým působením enzymů lze provést reverzní transkripci z RNA na DNA.

Translace [36] je proces překladu sekvence mRNA (messenger RNA) do sekvence aminokyselin proteinu. Překlad mRNA do proteinů probíhá na ribozomech a jednotlivé aminokyseliny se zařazují podle pravidel genetického kódu, který představuje množinu pravidel definujících jaké kodony (trojice nukleotidů) odpovídají jakým aminokyselinám. Proces translace na rozdíl od transkripce není vratný a tedy není možné aminokyselinovou sekvenci převést zpět na RNA.

Možnosti přenosu informace v rámci DNA, RNA a proteinů je definováno v rámci centrálního dogmatu biologie. Toto dogma definuje, že přenos informace je možné realizovat těmito směry: z DNA na RNA (transkripce), z RNA na DNA (reverzní transkripce) a z RNA na protein (translace). Jiné způsoby přenosu informace nebyly zatím pozorovány. [36]

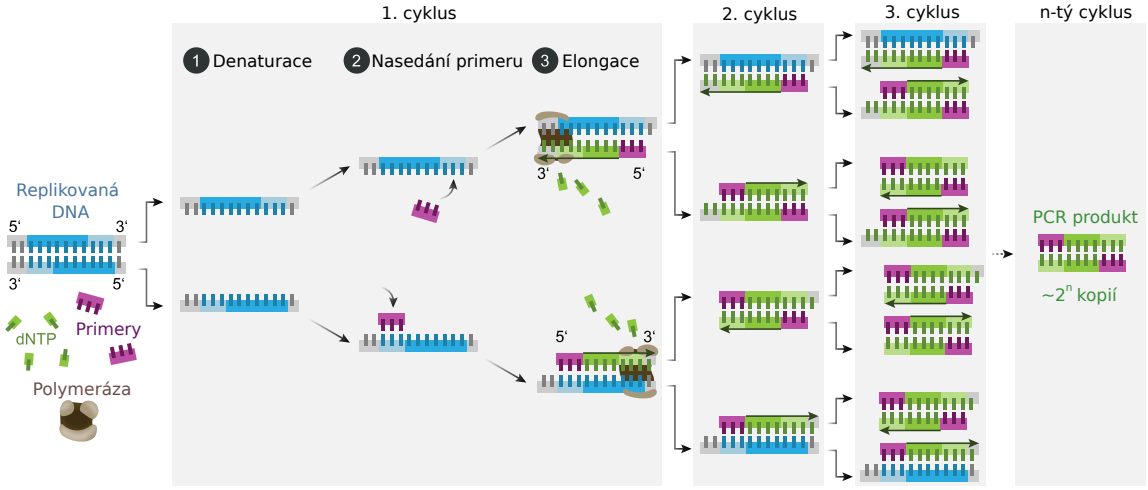
Polymerázová řetězová reakce (PCR)

Během studia DNA (a RNA) je dnes již běžnou praxí nepracovat pouze s jednou replikou daného vlákna, ale velkým množstvím identických replikovaných sekvencí. Pro replikaci vybraného segmentu DNA vlákna, lze využít polymerázové řetězové reakce, označované zkratkou PCR.

PCR [64, 65] je metoda jejíž historie sahá až do roku 1985, kdy ji doktor Kary B. Mullis a profesor Michael Smith prezentovali a později, v roce 1993, byly za svůj úspěch oceněni Nobelovou cenou [25].

PCR metoda umožňuje rychlou amplifikaci (replikaci) definovaného regionu DNA řetězce.

Základem PCR reakce je použití neporušeného úseku DNA, který je cílem replikace, neboli amplifikace. Amplifikovaný region je definován prostřednictvím použitých primerů. Primery



Obrázek 2.3: Průběh amplifikace pomocí PCR. Z obrázku si lze povšimnout, že v 1. cyklu PCR se nový řetězec syntetizuje až do konce vlákna (pokud je dostatek času) a nikoliv pouze do úrovně, kde se nachází druhý primer. Replikace pouze konkrétního definovaného regionu je provedena až ve druhém cyklu reakce, protože až ve druhém cyklu reakce jsou amplikony z 1. cyklu zkráceny na danou délku. Obrázek převzat ze serveru Wikimedia Commons.

jsou krátké, chemicky syntetizované oligonukleotidové sekvence. Jejich délka se pohybuje v řádu nízkých desítek nukleotidů. Konkrétní region DNA je definován pomocí dvou primerů, které se společně označují jako primerový pár.

PCR reakce využívá skutečnosti, že při denaturaci vlákna DNA dojde k rozdělení dvoušroubovice a při následném chladnutí dochází k hybridizaci primerů na příslušná místa v řetězci. Schématická ilustrace tohoto procesu je zobrazena na obrázku 2.2.

Průběh PCR lze rozdělit do tří hlavních kroků. V prvním kroku dochází k zahřívání vzorku DNA po dobu 10 až 30 sekund při teplotě 95 °C, což způsobí jeho denaturaci. V dalším kroku se teplota sníží na 50-65°C. Při této teplotě dochází k nasedání primerů na příslušná místa na řetězci. Po nasednutí primeru na DNA se na takto vzniklé dvouvláknové úseky váže DNA polymeráza. V posledním kroku je pomocí polymerázy syntetizováno vlákno DNA. Po provedení těchto tří kroků se počet replikovaných vláken zdvojnásobí. Opakováním kroků denaturace, navázání primerů na komplementární řetězce a následné syntézy vláken DNA vede k exponenciálnímu růstu množství replikovaného regionu DNA. Průběh PCR je vyobrazen na obrázku 2.3.

Celý proces PCR probíhá v připravené reakční směsi obsahující mimo jiné replikovanou DNA, primery a volné nukleotidy (dNTP). Výsledné replikované řetězce jsou označovány jako amplikony.

Pomocí PCR je možné replikovat pouze DNA řetězce. Replikace RNA je pomocí PCR možná pouze po převedení RNA vlákna na DNA, čehož lze dosáhnout působením enzymu reverzní transkriptázy.

Znak	Reprezentované báze	Znak	Reprezentované báze
A	{A}	**K**	{G, T}
C	{C}	**R**	{A,G}
G	{G}	**Y**	{C, T}
T	{T}	**B**	{C, G, T}
U	{U}	**D**	{A, G, T}
W	{A, T}	**H**	{A, C, T}
S	{C, G}	**V**	{A, C, G}
M	{A, C}	**N**	{A ,C, G, T}

Tabulka 2.1: Soupis rozšířené nomenklatury plně nespecifikovaných nukleotidů.

Zápis primerových párů

Primerový pár se zapisuje jako sekvence písmen reprezentujících specifické nukleotidy. Nejinak jako je tomu při zápisu jiných genomových sekvencí. Oproti standardní genomové sekvenci se v případě primerových párů, často používá nomenklatura zahrnující i částečně nespecifikované nukleotidy. [45]

V praxi to znamená, že standardní čtyři znaky (A,T,C,G) reprezentující DNA nukleotidy, jsou rozšířeny o další množinu znaků, které reprezentují hned několik možných nukleotidů. Příkladem může být znak *W*, který je zástupným znakem pro *A* nebo *T*.

Pokud je tedy primerová sekvence definovaná jako GGGCGG**W**GTG, pak taková sekvence reálně popisuje dvě plně definované sekvence a to konkrétně: GGGCGG**A**GTG a GGGCGG**T**GTG.

Z toho vyplývá, že jeden primerový pár často odpovídá množině reálných plně specifikovaných oligonukleotidových sekvencí.

2.2 Sekvenace genomu

Bioinformatika jako vědní obor propojující biologii a informatiku, potřebuje pro účely zpracování dat a realizaci výpočtů nad nimi, nejdříve data digitalizovat, respektive reprezentovat v počítači. Proces realizující čtení DNA (případně RNA) a jeho digitalizaci se nazývá sekvenování.

Sekvenování je obecný název shrnující různé metody, které zjišťují sekvenci nukleotidů určitého segmentu DNA(RNA). Počátek sekvenačních metod sahá do 70. let 20. století kdy anglický biochemik Frederick Sanger a jeho tým představily Sangerovu metodu sekvenování DNA. Krátce poté byla představena i druhá Maxam–Gilbertova sekvenační metoda. [34, 59]

Obě tyto metody jsou označovány jako metody první generace. Sangerova a Maxam-Gilbertova metoda jsou sice odlišné, ale přesto mají společný princip. Obě metody jsou založené na principu kdy se sekvenovaný řetězec DNA nareplikuje na podřetězce s náhodnou délkou, přičemž tyto podřetězce se dále sekvenují pomocí elektroforézy. Hlavní rozdíl obou metod je mechanizmus tvorby náhodně dlouhých řetězců.

Vývoj technologií, ale postupuje a v důsledku toho vznikly nové sekvenační metody 2. generace a aktuálně i metody 3. generace. [52, 66]

- 1. Generace
  - Sangerova a Maxam–Gilbertova metoda
  - Délka sekvencí 500-800bp (bázových párů)
- 2. Generace
  - Roche, Illumina, Ion Torrent Systems
  - Délka sekvencí se pohybuje kolem 300bp
- 3. Generace
  - Pacific Biosciences, Oxford Nanopore Technologies
  - Délka sekvencí v průměru 20kbp

Souhrnně se technologie 2. a 3. generace označují jako next-generation sequencing (NGS). NGS technologie se od první generace liší mimo jiné rychlostí zpracování, ale především výrazným cenovým rozdílem sekvenování genomu. Kde u první generace je cena sekvenace genomu 1 milion dolarů, tam se u druhé generace pohybuje cena v řádu nízkých tisíců dolarů a u třetí generace v řádu nízkých stovek dolarů. Dnes velmi rozšířenou metodou sekvenování je metoda používaná v sekvenátorech firmy Illumina[1].

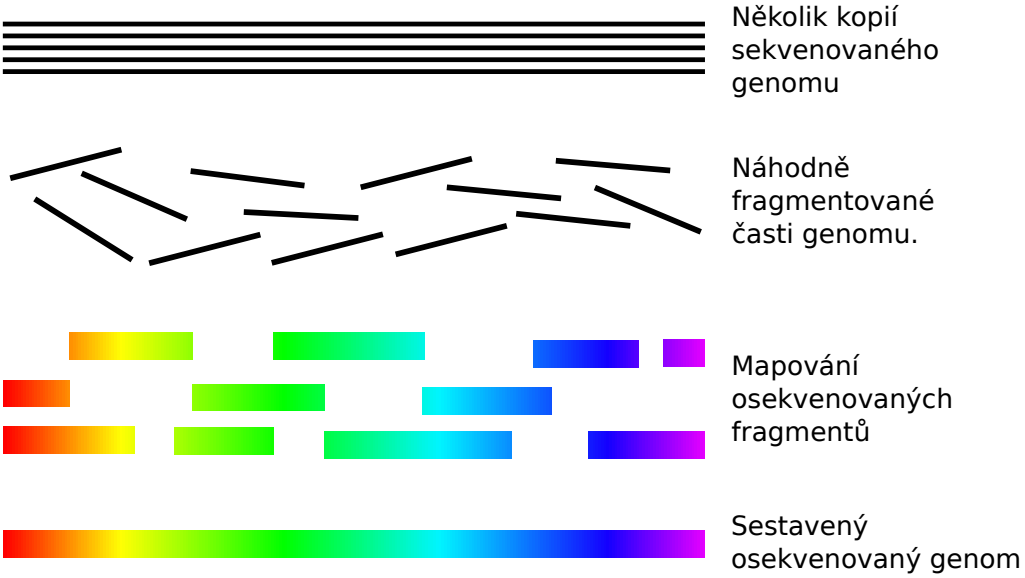
Illumina sekvenování [39, 19] je masivně paralelní sekvenační proces sestávající z několika fází. První fází je příprava vzorku. Sekvenovaný DNA řetězec se rozdělí na podřetězce dlouhé 150-300 bází (podle modelu sekvenátoru). Na konce připravených podřetězců jsou připojeny specifické krátké řetězce označovány jako adaptéry a sekvence vázající primer.

Druhou fází sekvenace je aplikace připravených denaturovaných DNA vzorků na speciální sekvenační destičku (flow cell). Flow cell je speciální destička, na níž jsou připravené krátké DNA řetězce, které jsou komplementární k aplikovaným adaptérům na sekvenovaných řetězcích. Aplikované vzorky se přichytí pomocí adaptérových sekvencí k destičce. Dále se pomocí PCR dané sekvence nareplikují do velkého počtu. Následně se DNA denaturuje a polovina nareplikovaných řetězců se odstraní tak, aby zbyla jen přímá DNA vlákna. Nakonec se přidají sekvenační primery, jenž se navážou na konce připravených řetězců.

Ve třetí fázi se realizuje samotné sekvenování. Na připravenou flow cell destičku jsou aplikované volné nukleotidy, upravené tak, aby byly fluorescentní. Zároveň tyto nukleotidy mají modifikovanou hydroxylovou skupinu, což způsobí pozastavení postupu polymerázy a efektivně tedy pozastavení syntézy řetězce. Na flow cell je přidána i polymeráza, která nasedá na primer a realizuje syntézu vlákna. Syntéza je pozastavena v důsledku připojení modifikovaného fluorescentního nukleotidu. V tuto chvíli je pořízen snímek mikroskopem, který zachytí fluorescenci daného nukleotidu. Podle barvy záření je identifikován typ nukleotidu. Jakmile jsou vzorky nasnímané, obnoví se hydroxylová skupina na modifikovaném nukleotidu, která umožní pokračování syntézy. Celý proces se opakuje do doby, dokud není nasyntetizované celé vlákno.

Celý tento proces je masivně paralelní a na jediné flow cell se takto sekvenuje obrovské množství řetězců. Výsledkem analýzy nasnímaných obrazů je tedy množina krátkých (150-300 znaků) textových řetězců, které reprezentují sekvenované řetězce.

[1] https://www.illumina.com/



Obrázek 2.4: Ilustrace procesu shotgun sekvenování. Obrázek inspirován ze stránek Okinawa institute of science and technology www.oist.jp

Aktivně se používají i další technologie jako: Ion Torrent technologie používaná v sekvenátorech firmy ThermoFisher[2], případně dnes již nevyvíjená technologie [28] 454 Pyrosekvenování a v neposlední řadě také technologie 3. generace jako nanopórové sekvenátory od Oxford Nanopore Technologies[3].

Všechny aktuální technologie jsou omezeny délkou osekvenovaných sekvencí. U dnes nejpoužívanějších technologií (1. a 2. generace) je tento limit v řádu stovek osekvenovaných bází. Celá genomová sekvence organismů se ale pohybuje v řádu milionů a více bázových párů. Konkrétně u člověka se jedná o přibližně 3200 Mbp [4], které tvoří celý genom. V důsledku tohoto faktu je jasné, že pro osekvenování celého genomu, je nutné sekvenovat po částech a následně tyto části spojovat do větších celků. Tato metoda se označuje shotgun sekvenování.

Shotgun sekvenování [67] je metoda sekvenování velkých částí (celého) genomu, při které se DNA náhodně rozstěpí na mnoho malých fragmentů, které se samostatně osekvenují. Celý proces štěpení a fragmentace je několikrát opakován, díky čemuž vznikne mnoho vzájemně se překrývajících sekvencí. V této fázi je osekvenován velký blok genomu ve formě velkého množství krátkých řetězců o kterých, ale není informace, kde se původně v genomu nacházejí a jak na sebe vzájemně navazují.

Díky překryvu sekvencí, je možné výsledné sekvence, které jsou v počítači uloženy jako sekvence znaků, vzájemně zarovnat a vypočítat celou sekvenci. Ilustrace tohoto procesu je vyobrazena na obrázku 2.4.

Výsledky sekvenace je potřeba pro učely dalšího zpracování reprezentovat a uložit v počítači. Pro tento účel vznikly formáty souborů jako FASTA nebo například FASTQ. V obou případech se jedná o textové soubory s definovanou strukturou.

[2] https://www.thermofisher.com/cz/en/home/brands/ion-torrent.html
[3] https://nanoporetech.com/
[4] https://www.ncbi.nlm.nih.gov/genome/browse!/overview/human

FASTA je textový formát určený k reprezentaci nukleotidových sekvencí a aminokyselin. Sekvence jsou reprezentovány pomocí sekvencí písmen, kde jedno písmeno odpovídá právě jednomu nukleotidu, respektive aminokyselině. Každé sekvenci ve FASTA souboru předchází její jednořádkový popis uvozený znakem ">". Na dalším řádku následně začíná daná sekvence. Sekvence se zapisuje do jednotlivých řádků, které by neměly přesáhnout délku 80 znaků. Příklad FASTA souboru lze nalézt na obrázku 2.5.

```
>Enterobacter--space--cloacae--13047
AAATTGAAGAGTTTGATCATGGCTCAGATTGAACGCTGGCGGCAGGCCTAACACATGCAAGTC
GCTTGCTCTCGGGTGACGAGTGGCGGACGGGTGAGTAATGTCTGGGAAACTGCCTGATGGAGG
CGGTAGCTAATACCGCATAATGTCGCAAGACCAAAGAGGGGGACCTTCGG
```

Obrázek 2.5: Příklad reprezentace sekvence prostřednictvím FASTA souboru.

FASTQ soubor, na rozdíl od FASTA souboru, dokáže navíc kromě samotné sekvence reprezentovat i kvalitativní informace o každém jednom znaku v sekvenci. Právě díky kvalitativnímu záznamu je často používán jako výstupní formát sekvenátorů. Struktura záznamu začíná řádkem s popisem sekvence uvozený znakem "@". Na dalším řádku je daná sekvence. Třetí řádek začíná znakem "+"a jménem sekvence. Čtvrtý řádek v řadě obsahuje kvalitativní informace o dané sekvenci ve formě ascii znaků. Příklad FASTQ souboru lze nalézt na obrázku 2.6. [42]

```
@Sequence_name
GCTTGCTCTCGGGTGACGAGTGGCGGACGGGTGAGTAATGTCTGGGAAACTGCCTGATGG
+Sequence_name
!''*((**21214>>>>CCCC65(*****)%%%$$$%%%%%%!!!***))()1112CC65
```

Obrázek 2.6: Příklad reprezentace sekvence a její kvality ve FASTQ souboru.

2.3 Metagenom a jeho studium

Metagenomika je soubor metod zabývající se studiem genetického materiálu získaného přímo z prostředí. Tento materiál obsahuje mnoho různých mikrobiálních organismů, které nelze, nebo jen velmi obtížně, oddělit do samostatných vzorků.

Studium metagenomu proniká do mnohých oblastí od studia prostředí, zpracování biopaliv až po studium metagenomu střev, úst, kůže lidí a zvířat a z toho vycházející diagnostiku různých onemocnění. Právě střevní metagenom se ukazuje jako možný indikátor různých onemocnění od rakoviny střev [53], přes psychologické nemoci jako deprese nebo bipolární porucha [55] až po Parkinsonovu chorobu [62]. Právě velký potenciál metagenomiky, vede k rozvoji různých metagenomických metod.

Do metagenomiky spadá studium funkčních vlastností, genetických sekvencí a populační diverzity celých mikrobiálních populací [75]. Tato práce spadá do oblasti metagenomiky,

která se zabývá studiem populační diverzity, tedy přítomností jednotlivých bakteriálních a archeálních druhů a jejich zastoupením.

Ke studiu metagenomické populační diverzity lze využít dva hlavní přístupy. Jednou možností je sekvenace všech genomových sekvencí v daném metagenomickém vzorku pomocí shotgun sekvenování. Soubor všech těchto genomových sekvencí se označuje jako metagenom a jeho sekvenace je anglicky označována jako **Whole metagenome sequencing** (WMS). Druhou možností je amplikonové sekvenování, kdy je sekvenována pouze určitá specifická část vysoce konzervovaného genu (u prokaryot se jedná o gen 16S rRNA).

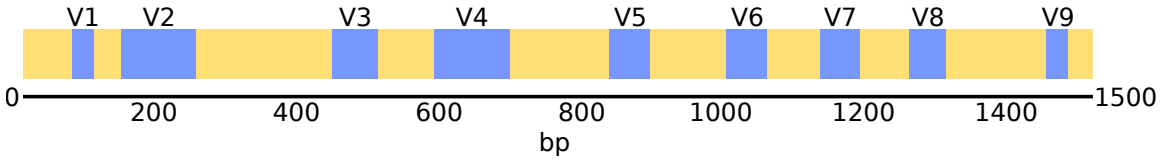
WMS [63] metoda spočívá v sekvenaci ideálně veškeré genetické informace obsažené ve vzorku pomocí shotgun sekvenování. Výstupem takto provedeného shotgun sekvenování je obrovské množství krátkých textových řetězců, reprezentujících úseky genomu veškerých možných organismů obsažených ve vstupním vzorku. Nasekvenovaná data se zpracovávají v počítači. Jednou možností zpracování je mapování získaných řetězců, na již známé genomové sekvence. V tomto případě se hledá ideální zarovnání získaného řetězce, vůči velkému množství genomových sekvencí v databázi. Díky namapování získaných řetězců jsme schopni identifikovat přítomnost daného druhu a díky četnosti daných sekvencí i jeho přibližné zastoupení. Tento postup je ovšem limitující na identifikaci druhů již obsažených v databázi. Druhou možností je de novo sestavování. V tomto případě se v řetězcích hledají překrývající se částí, které se na sebe zarovnávají, díky čemuž je možné vypočítat delší genomové bloky. V ideálním případě lze tímto způsobem zrekonstruovat celé genomy jednotlivých organismů. Reálně se ale daří zrekonstruovat pouze různě dlouhé bloky (označované jako contigy) genomu, u kterých lze dále identifikovat druh podle mapování na genomovou databázi, případně určit, že se jedná o nový druh v databázi neobsažený.

Popularita WMS roste společně se snižováním ceny sekvenování. Cena sekvenování je pro tuto metodu poměrně stěžejní, protože množství genetické informace obsažené ve vzorcích z prostředí je obrovské. Zároveň je WMS velmi výpočetně náročné, především při de novo sestavování se jedná o velmi komplexní proces.

Amplikonové sekvenování [73, 56] využívá skutečnosti, že v genomu organismů existují vysoce konzervované geny, tedy takové, které se vyskytují ve velmi podobné formě napříč velkou skupinou (ideálně všech) druhů. Jedním z takových genů, který se využívá při amplikonovém sekvenování prokaryotických organismů je gen 16S rRNA. Při amplikonovém sekvenování se ze vzorku z prostředí pomocí PCR amplifikuje specifický region 16S rRNA genu. Následnou sekvenací amplifikovaného vzorku, lze provést klasifikaci přítomných druhů a zjistit jejich poměrné zastoupení. Pro pochopení toho proč je to možné je potřeba pochopit strukturu a funkci genu 16S rRNA.

16S rRNA je gen kódující ribozomální RNA, která je potřebná při proteosyntéze u prokaryotických organismů [1]. Většinou je v genomu redundantně obsažen v několika kopiích (5-10) [40] . Celková délka genu se pohybuje kolem 1500bp a interní struktura genu je složena z konzervovaných a variabilních regionů, kde konzervované regiony jsou napříč druhy velmi podobné a variabilní se u jednotlivých druhů převážně liší. Standardně se v genu vyskytuje 9 variabilních regionů označených V1-V9, jejichž ilustrace je zobrazena v obrázku 2.7. [41]

Právě díky výskytu variabilních regionů je možné tento gen využít k fylogenetickému studiu organismů [74, 73], klasifikaci jednotlivých druhů, potažmo k určení druhové diverzity mikrobiomu. Přesto se ale nejedná o jednoznačný proces. Variabilní regiony nejsou všechny



Obrázek 2.7: Ilustrace variabilních (modré) a konzervovaných (žluté) regionů 16S rRNA genu.

rozdílné u každého druhu a proto výběr amplifikačních primerů, které jsou použity při PCR, není triviální. [41]

Amplikonové sekvenování využívá existence variabilních a konzervovaných regionů. Vzájemná variabilita regionů, ale není mezi všemi druhy stejná [41]. Zjednodušeným příkladem může být případ, kdy určitá třída bakterií se oproti jiné může nejvíce lišit v regionech V1-V3 zatímco regiony V4-V9 se budou mezi těmito třídami lišit pouze minimálně, nebo naopak se mezi dvěma druhy může nejvíce lišit region V7 a ostaní mohou být shodné. Z tohoto důvodu je výběr amplifikovaného regionu velmi důležitý a ovlivňuje kvalitu výsledku. To jaký region se bude amplifikovat záleží na sekvencích použitého primerového páru.

Proto je tvorba systému, schopného vybrat z databáze právě takový primerový pár, který bude maximálně vyhovovat podmínkám experimentu, tedy že zaručí co nejlepší specificitu vybraných druhů za zachování co nejvyšší senzitivity celého spektra organismů, cílem této práce.

Vlastnosti a výběr vhodného primerového páru jsou kruciálními faktory ovlivňující výsledky amplikonového sekvenování. Stěžejní vlastnosti, které je nutné brát v úvahu při výběru primerového páru jsou senzitivita a specificita experimentu.

Senzitivita určuje, množství druhů zachycené daným primerovým párem, tedy u kolika druhů bude amplifikovat cílený segment 16S rRNA genu.

Specificita určuje, kolik a jaké druhy je možné při použití daného primerového páru jednoznačně určit. Tedy kolik druhů nebude v rámci amplifikované sekvence kolidovat s jinými druhy.

Na počátku řešení této práce jsem dostal přístup k databázi (a skriptu, který je schopen ji generovat) obsahující data o několika tisících (celkem 5643) primerech. V databázi byly uloženy informace o primerech ve formě orientovaných grafů, které jsou více popsané v sekci 3.3. Grafy reprezentují, které taxony mezi sebou nerozlišitelné při použití určitého primerového páru. Zároveň obsahují i informace o tom, jaké taxony vybraný primerový pár zachytí. Z těchto grafů lze tedy vypočítat specificitu i senzitivitu primerů pro taxony, které jsou v databázi zahrnuty (celkem 24829 taxonů, přičemž 21493 na úrovni druhů). Z databáze lze na základě obsažených dat inferovat ideální primerový pár pro zvolené taxony. Algoritmus inference je uveden v sekci 3.5.

Ve většině případů nelze tvrdit, že ideálním primerovým párem, bude právě ten s nejvyšší globální senzitivitou i specificitou (pokud by tedy obě vlastnosti nebyly 100%). Zároveň nelze tvrdit, že k dostatečně přesnému výsledku je dostatečné použít "univerzální"primerový pár, který zachytí většinu druhů, protože určité primerové páry jsou schopny zachytit pouze určité taxony. Tato skutečnost je zachycena na obrázku 2.8, kde je vizualizována senzitivita a specificita vybraných primerových párů s vybranými kmeny prokariot. Na obrázku lze vidět

fakt, že sic určté primerové páry jsou schopny zachytit poměrně širokou škálu organismů, rozhodně nezachycují kompletní škálu. Takový primerový pár může být například P699D - 1492R (s), který umožňuje amplifikovat 16S rRNA region u většiny vizualizovaných kmenů, bude zcela nevhodný pro amplifikaci kmenu Abditibacteriota.

Z toho plyne, že výběr vhodného primerového páru tedy není jednoduchý a přímočarý problém. Při rozhodování je nutné vzít v potaz, které taxony jsou pro daný experiment důležité a které méně. Protože ideální primerový pár (takový, který by senzitivoval i specifikoval celé spektrum) při použití 16S sekvenování neexistuje, je nutné, při výběru počítat s kompromisními rozhodnutími.

Taxonomická klasifikace na základě 16S rRNA sekvencí

Samotnou sekvenací metagenomu nezískáme informace o přítomnosti konkrétních druhů. K tomu je nutné provést klasifikaci.

Klasifikace získaných sekvencí je proces, jehož vstupem je genomová sekvence a výstupem je taxonomická skupina ze které pochází včetně míry jistoty klasifikace. Výstupní taxonomická skupina není pouze na úrovni druhu. Častou skutečností je klasifikace na vyšší úrovni, například rodu. Důvodem k tomu je obtížnost realizace klasifikace. Jednotlivé druhy jednoho konkrétního rodu, mohou mít sekvence natolik podobné, že jejich přesná klasifikace by byla nemožná, nebo velmi obtížná. Z toho vyplývá, že chybovost klasifikace na úrovni druhu, je podstatně vyšší než na úrovni rodu.

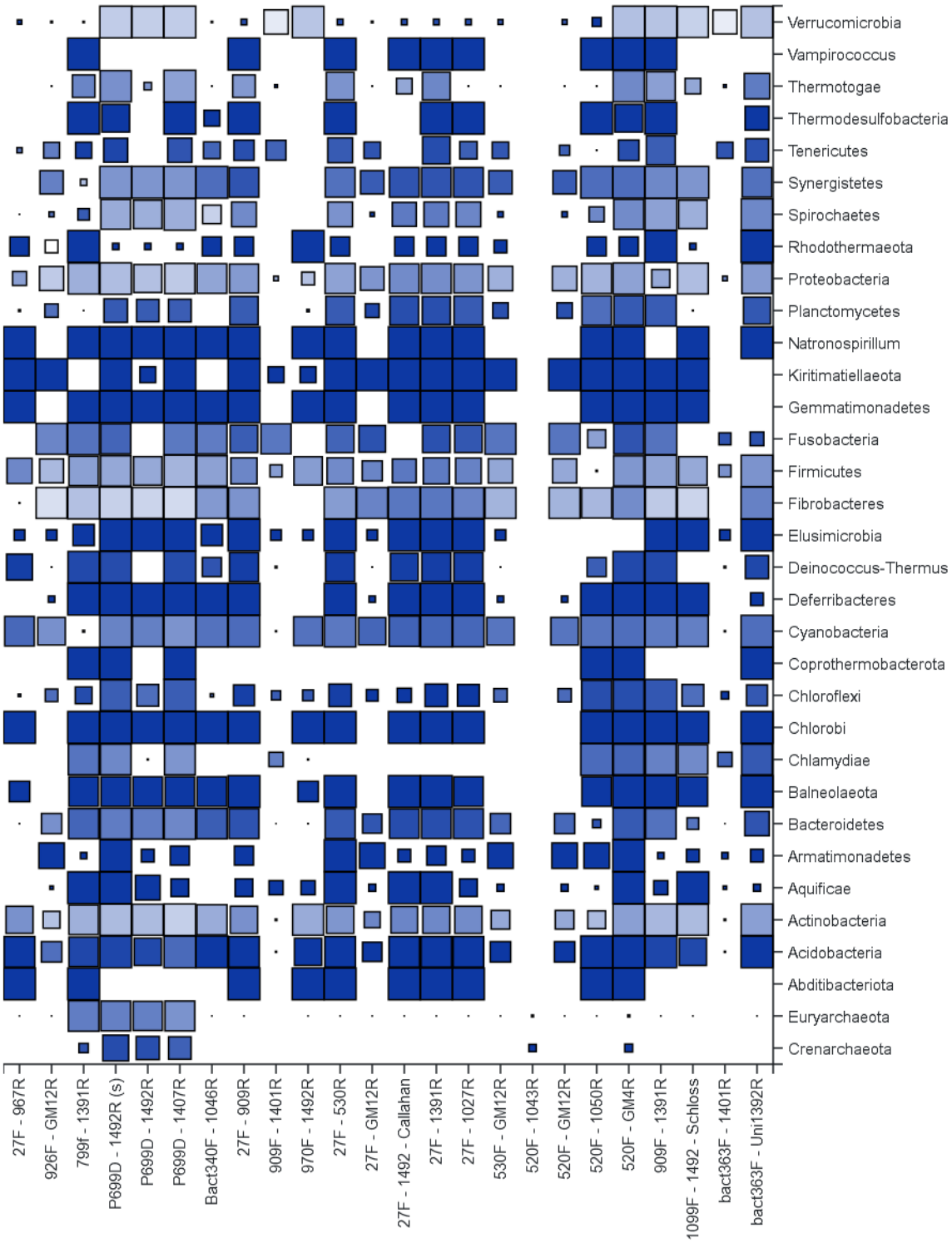
Existuje několik možných přístupů jak klasifikaci provést.

První možností je využití nástroje **BLASTn** [57]. Jedná se o nástroj schopný vyhledávat sekvence na základě podobnosti. Na základě vstupní sekvence jsou vyhledány nejpodobnější sekvence z databáze. Vyhledané sekvence jsou na základě podobnosti ohodnoceny a výsledky jsou zobrazeny uživateli. Nástroj je možné plně využít prostřednictvím webových stránek projektu BLAST, díky čemuž je základní použití nástroje snadné a především rychlé. Klasifikace s pomocí BLASTn je však pouze základní metodou a její výsledky jsou často používány spíše jako porovnání vůči komplexnějším metodám klasifikace. [37, 70]

Přesnějších výsledků klasifikace, lze dosáhnout použitím klasifikačního modelu, jako je například klasifikátor **RDP** [71].

Jedná o naivní bayesovský klasifikátor, který umožňuje klasifikaci sekvencí na základě 16S rRNA podsekvencí. Jedná se o velmi často používaný klasifikátor [44, 61]. Projekt RDP (Ribosomal database project) ovšem nesestává pouze z klasifikátoru, ale součástí projektu je i spravovaná databáze 16S rRNA sekvencí bakterií a archeí, dále doplněná o 28S rRNA sekvence hub [43]. Aktuálně je poslední verze databáze ze září roku 2016 a celkem obsahuje přes tři miliony anotovaných 16S rRNA sekvencí. Právě tyto sekvence jsou použity při trénování RDP klasifikátoru.

Alternativními, a podle jejich autorů přesnějšími, klasifikátory jsou **SINTAX** [47] nebo **SPINGO** [37]. Ani jeden z těchto klasifikátorů, není tak široce rozšířen jako klasifikátor RDP, přestože v obou případech jejich autoři uvádějí lepší výsledky klasifikace sekvencí.



Obrázek 2.8: Vizualizace specificity a senzitivity vybraných primerů pro vybrané kmeny bakterií. Velikost čtverce reprezentuje míru senzitivity (čím větší čtverec, tím větší senzitivita. Specificita je vyjádřena barvou čtverce (čím tmavší modrá, tím větší specificita). Z této vizualizace je jasně vidět důležitost výběru primerového páru, který odpovídá oblasti experimentu.

Kapitola 3

Návrh a architektura aplikace

Při tvorbě webové aplikace je třeba vzít do úvahy mnoho různých aspektů, požadavků a funkcionalit, které musí výsledná aplikace splňovat. Základním kamenem rozhodování je jaké technologie a přístupy při tvorbě aplikace použít. Velkou mírou tomuto rozhodování přispívá analýza požadavků aplikace. Jejím cílem je definovat požadovanou funkcionalitu a schopnosti výsledného systému, společně s cílovou skupinou a očekávaným použitím aplikace.

3.1 Analýza požadavků

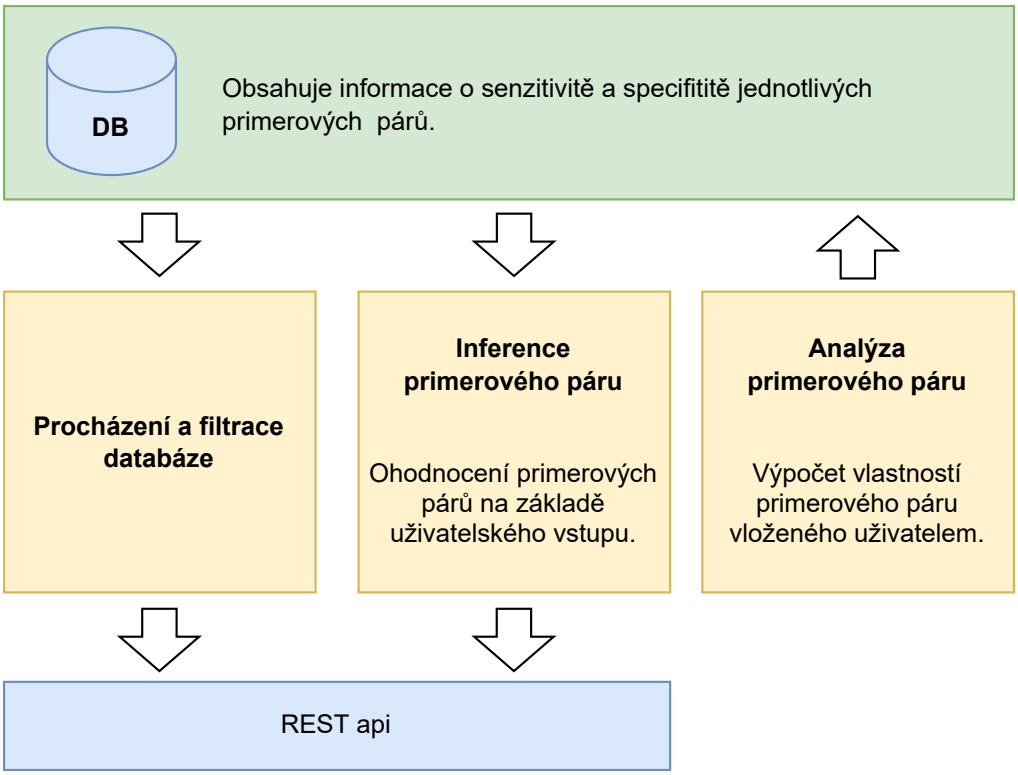
Hlavním cílem této práce, je tvorba webové aplikace, která bude obsahovat databázi primerových párů a jejich vlastností. Na základě těchto dat a uživatelského vstupu bude aplikace schopna sama vyhodnotit a doporučit uživateli primerové páry tak, aby maximálně vyhovovaly uživatelským požadavkům. Výsledky ohodnocení primerových párů bude aplikace zároveň uživateli pro přehlednost vizualizovat.

Zároveň bude aplikace schopna vypočítat vlastnosti uživatelem vloženého primerového páru.

Dále bude aplikace umožňovat procházení, filtraci a vizualizaci databáze primerových párů. V neposlední řadě bude aplikace poskytovat aplikační rozhraní pro možnost využití funkcionality inference i mimo webové uživatelské rozhraní. Základní schéma funkcí aplikace je vyobrazeno na obrázku 3.1.

Inference amplikonového primeru. Jedná se proces při němž se ohodnotí všechny primerové páry v databázi na základě zvolené množiny taxonů. Konkrétně se na základě taxonů v této zvolené množině vypočítá vybraná senzitivita a specificita každého primerového páru. Na základě vypočítaných vlastností lze primerové páry následně porovnávat řadit a vizualizovat.

S inferencí primerových párů úzce souvisí možnost získat informace o specicfikém primerovém páru. Mezi tyto informace spadá specificita, senzitivita, samotné primerové sekvence, poloha amplikovaného regionu 16S rRNA genu a především vizualizace nerozlišitelných taxonů. Tedy taxonů, které po osekvenování nebude možné mezi sebou klasifikovat.



Obrázek 3.1: Schéma základních funkcí aplikace. Vyobrazeny jsou tři hlavní funkce aplikace. Dvě, které pracují z daty obsaženými v databázi, a které je možné využít i s pomocí rozhraní REST: Inference a procházení a filtrace primerových párů. A funkce analýzy nového primerového páru.

Právě vizualizace nerozlišitelností je jednou ze stěžejních detailních informací, které bude aplikace umět uživateli předat. Vizualizace by měla být dostatečně přehledná a srozumitelná.

Díky ohodnocenému seznamu primerových párů a možnosti detailně prostudovat každý jednotlivý primerový pár a jeho vlastnosti, by měl být uživatel schopen zvolit takový primerový pár aby co nejlépe vyhovoval jeho potřebám.

Protože možných kombinací a sekvencí jednotlivých primerů je obrovské množství, je zcela jisté, že se objeví potřeba uživatele zjistit informace o jeho vlastní primerové sekvenci. Z toho důvodu je nutnou součástí aplikace i funkcionalita pro výpočet uživatelsky definovaného primerového páru.

Dalším požadavkem na systém, je tedy **možnost vložení primerového páru k analýze**. Uživateli bude umožněno spustit výpočet vlastností pro jeho vlastní primerový pár. Takový primerový pár se analyzuje stejným postupem jako všechny ostatní páry v databázi a uživatel díky tomu zjistí, který region genu bude daný pár amplifikovat, zdali vůbec nějaký, a samozřejmě také u jakých druhů bude region amplifikovat, a které druhy budou mezi sebou kolidovat. Zároveň bude moci tento nově vypočítaný pár porovnat vůči již stávajícím párům v databázi.

Výpočet vlastností primerového páru, může potenciálně trvat desítky minut, proto je nutné, aby sytém podporoval běh dlouhotrvajících úloh. V důsledku to znamená, že systém musí umět pracovat s frontou úloh a množinou výpočetních uzlů, kterým budou dané úlohy přiřazovány. Po ukončení výpočtu systém bude umožňovat upozornit uživatele prostřednictvím emailové zprávy.

V neposlední řadě bude systém umožňovat **procházení a parametrizované filtrování primerových párů** uložených v databázi.

Při tvorbě aplikace je také nutné vzít do úvahy cílovou uživatelskou skupinu. V tomto konkrétním případě cílovou skupinu představují převážně biologové, bioinformatici a jiní převážně vědečtí pracovníci. Z toho vyplývá, že se jedná o skupinu uživatelů, která je seznámena s problematikou metagenomiky, aspektů amplikonového sekvenování a problematiky senzitivity a specificity primerových párů.

Zároveň je při návrhu uživatelského rozhraní a celé aplikace nutné počítat z možností, že uživatel není seznámen s problematikou, principem a celkovým významem algoritmu inference. Z toho důvodu je nutné do rozhraní začlenit stručné nápovědy, které vysvětlí případné nejasnosti.

3.2 Existující řešení

Na internetu lze nalézt mnohé projekty a nástroje, které se zabývají obecnou problematikou 16S rRNA sekvencí. Široce používané projekty, které se zabývají extrakcí, anotací a správou databáze 16S rRNA sekvencí jsou projekty **SILVA** [49], nebo **RDP** [43]. Součástí projektu RDP je i klasifikátor sekvencí.

Kromě nich existují i nástroje, které se zaměřují specificky na problematiku primerových párů. Nástroje, které lze využít k výběru primerového páru jsou například **ProbeBase** [50] nebo **Primer BLAST** [76]. Oba nástroje se zaměřují na podobnou úlohu, jaká je řešena v rámci této práce, ale ani jeden z těchto nástrojů ovšem nedisponuje plnou funkcionalitou, kterou nabízí aplikace vyvinutá v rámci této práce.

Jednou z hlavních motivací tvorby těchto nástrojů a především této práce, je poskytnout výzkumníkům nástroj, díky němuž budou moci najít primerový pár, který je pro jejich experiment vhodný. **Nikoliv pár, který je obecně známý, případně často používaný**, a který pro jejich experiment ve své podstatě vůbec vhodný není.

ProbeBase je nástroj provozovaný Vídeňskou univerzitou. Jedná se o manuálně spravovanou databázi rRNA oligonukleotidů. Umožňuje procházet databázi primerů, vyhledávat primery na základě jména nebo sekvence. Dále lze nástroj použít k nalezení použitelných primerů pro jeden specifický taxon. Aplikace, ale postrádá možnost vyhledat a ohodnotit primery pro množinu několika taxonů, porovnání jednotlivých primerových párů. Dále není možné provést ani vizualizaci nebo filtraci nalezených párů. Také postrádá možnost vložení vlastního primeru, respektive tato možnost je dlouhodobě nedostupná (dle zachycených obrazů stránky na internetovém archivu[1], byla tato možnost dostupná těsně po spuštění této aplikace v roce 2015, ale už na archivované verzi z 29.3.2016 je funkce nedostupná a v následujících záznamech ani dnes už zpřístupněna nebyla).

[1] https://web.archive.org

Lze říci, že se jedná o databázi se základní funkcionalitou vyhledání a získání základních informací o primerových sekvencích.

ProbeBase umožňuje získat více informací o vybrané oligonukleotidové sekvenci pomocí dvou zmíněných databází. Konkrétně je možné pro vybraný primer získat další informace za pomoci databází SILVA a RDP.

SILVA a RDP jsou spravované databáze ribozomálních rRNA genových sekvencí. Mimo jiné je zde možné najít ribozomalní rRNA sekvence velkého množství organismů, včetně podrobnějších informacích o daných druzích a případně i studiích, které je popisují.

Primer BLAST je nástroj, který se zabývá problematikou konstrukce primerového páru. Primer BLAST se nespecializuje na primerové páry pro amplikonové 16S rRNA sekvenování, ale umožňuje výpočet primerového páru pro specifickou sekvenci, která je cílem PCR amplifikace. Primer BLAST tedy dokáže zacílit vybraný primer tak, aby amplifikoval právě zadanou sekvenci a ideálně žádnou jinou. Tento přístup se, ale nehodí při 16S rRNA amplifikaci právě kvůli variabilním regionům, které jsou hlavním cílem následné sekvenace a navazující klasifikace.

3.3 Datový model

Veškerá funkcionalita aplikace závisí na informacích o jednotlivých primerových párech uložených v databázi. V databázi jsou uložena data o jednotlivých taxonech, jejich počtech, senzitivitě, specificitě a jejich vzájemné nerozlišitelnosti v souvislosti s jednotlivými primerovými páry.

Na počátku řešení práce jsem dostal přístup k databázi primerových párů a při návrhu datového modelu z této databáze sice vycházím, ale oproti původnímu návrhu jsem provedl poměrně mnoho změn, vylepšení a optimalizací.

Základní částí datového modelu je uložení taxonomické hierarchie. Ve své podstatě se jedná o stromovou strukturu, kde jednotlivé úrovně stromu představují taxonomické kategorie. Kompletní taxonomická hierarchie je sestavena z mnoha stupňů a mezistupňů (nadkmen, podkmen, infratřída...).

V této hierarchii ale v nezanedbatelném počtu případů nastává situace, kdy určitý organismus nemá definovanou určitou taxonomickou kategorii, ať už z jakéhokoliv důvodu. (Příkladem mohou být dosud nezařazené bakterie, jenž mají definovanou například pouze doménu a druh.) Z důvodu zjednodušení a normalizace taxonomického stromu, je tedy taxonomická hierarchie, pro účely aplikace, omezena pouze na 7 (respektive 8) hlavních kategorií. Ty jsou konkrétně: doména (domain), kmen (phylum), třída (class), řád (order), čeleď (family), rod (genus) a druh (specie). Osmou, ale z pohledu uživatele aplikace méně důležitou kategorií je *strain*, tedy specifická varianta určitého druhu.

Situaci, kdy druh nemá definovanou jednu nebo více taxonomických kategorií je z důvodu dalších aspektů aplikace nutné řešit. Tyto aspekty se týkají především analýzy a následné vizualizace vlastností primerového páru. Nedefinovaná místa jsem se rozhodl vyplnit označením kategorie z nižší úrovně. Tento mechanizmus je vyobrazen v tabulce 3.1.

Pro úplnost je potřeba uvést, že celá taxonomická hierarchie se s postupem času poměrně intenzivně mění a druhy, které dříve klasifikovány nebyly, dnes už být mohou. Případně

Druh	Rod	Čeleď	Řád	Třída	Kmen	Doména
druh-A	druh-A	čeleď-A	řád-A	třída-A	kmen-A	doména-A
druh-B	rod-B	rod-B	rod-B	třída-B	třída-B	doména-A

Tabulka 3.1: Mechanizmus vyplňování prázdných míst v taxonomické hierarchii. Oranžově vyznačená políčka byla vyplněna kategorií nižší třídy.

druhy, které patřily do jedné kategorie, mohou dnes patřit do jiné. Stejně tak mohou změnit pojmenování. Z toho vyplývá, že jednoduché vyčkávání na klasifikaci dnes nezařazených druhů, sice v budoucnu může vést plnému zařazení druhu do hierarchie, ale zároveň pravděpodobně přibydou další nezařazené druhy. V databázi je tedy uložen obraz z jednoho konkrétního stavu hierarchie.

Dalším úkolem datového modelu je uchování informací o primerových párech, označení jednotlivých primerů a jejich sekvencí. Stěžejní částí, a také majoritní co se množství dat týče, je především část zodpovědná za uložení informace o senzitivitě a specificitě jednotlivých primerových párů.

V rámci aplikace je senzitivita primerového páru definována jako množství druhů z celkového počtu, u nichž bude při reakci PCR amplifikován cílený region 16S rRNA genu. Konkrétně je senzitivita vyjádřena jako procento amplifikovaných druhů z celkového množství druhů.

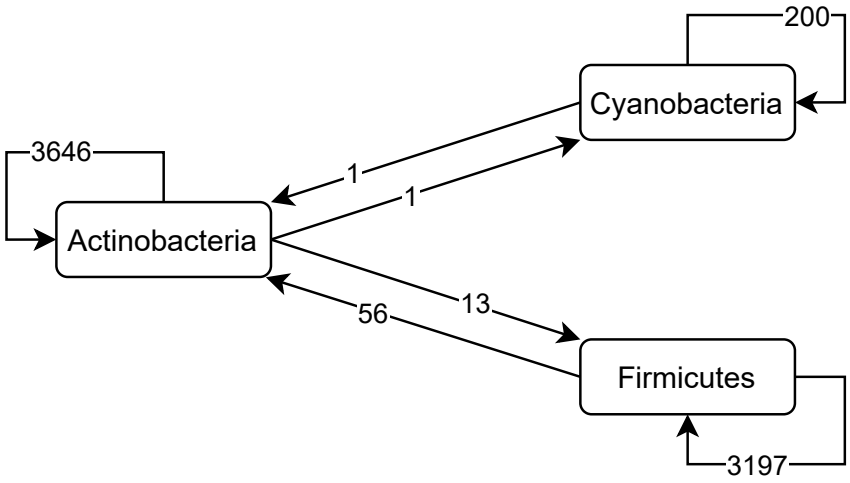
Na druhé straně celková specificita je definována jako množství rozlišitelých druhů z celkového počtu senzitivovaných druhů. Stejně jako senzitivita je i specificita vyjádřena procentuálně.

Celková senzitivita a specificita, jsou ale pouze základními a nikoliv dostačujícími informacemi. Pro celkovou funkcionalitu aplikace je třeba uchovávat informace o tom kolik druhů je zachycených a rozlišitelných v rámci všech taxonů a taxonomických kategorií v databázi. Je-li tedy v databázi čeleď A do které spadá N druhů, pak v datech musí být obsažena hodnota kolik z celkového počtu N druhů je daným primerovým párem zachyceno a kolik z těchto druhů bude nerozlišitelných vůči ostatním druhům v databázi. V databázi jsou v rámci jedné tabulky uloženy hodnoty množství amplifikovaných a nerozlišitelných druhů v závislosti na taxonu a na primerovém páru.

Na to přímo navazuje poslední významná část dat související se specificitou primerového páru. Jedná se o uložení konkrétních nerozlišitelností mezi taxony. Je-li v databázi taxon, do něhož spadají druhy, které jsou při použití specifického páru nerozlišitelné s jinými druhy, pak pro tento taxon jsou v databázi uloženy informace mezi kterými konkrétními taxony na stejné taxonomické úrovni dochází k nerozlišitelnosti a kolik druhů bude nerozlišitelných. Výsledná datová struktura odpovídá orientovanému grafu, kde ohodnocená hrana z jednoho uzlu (taxonu) do druhého vyjadřuje existenci a množství nerozlišitelných druhů. Fragment reálného grafu je vyobrazen na obrázku 3.2.

Popsaná část datového modelu je zachycena prostřednictvím ER diagramu na obrázku 3.3.

Možnosti rozšíření datového modelu. Zatím se veškeré informace uváděly na úrovni druhů, tedy kolik druhů je nerozlišitelných, kolik druhů je amplifikovaných a podobně. Původní databáze, ale i finální implementace, počítá i s takzvaným parametrem rozlišení, který definuje taxonomickou kategorii, pro níž jsou hodnoty vedeny. Ve finální implementaci



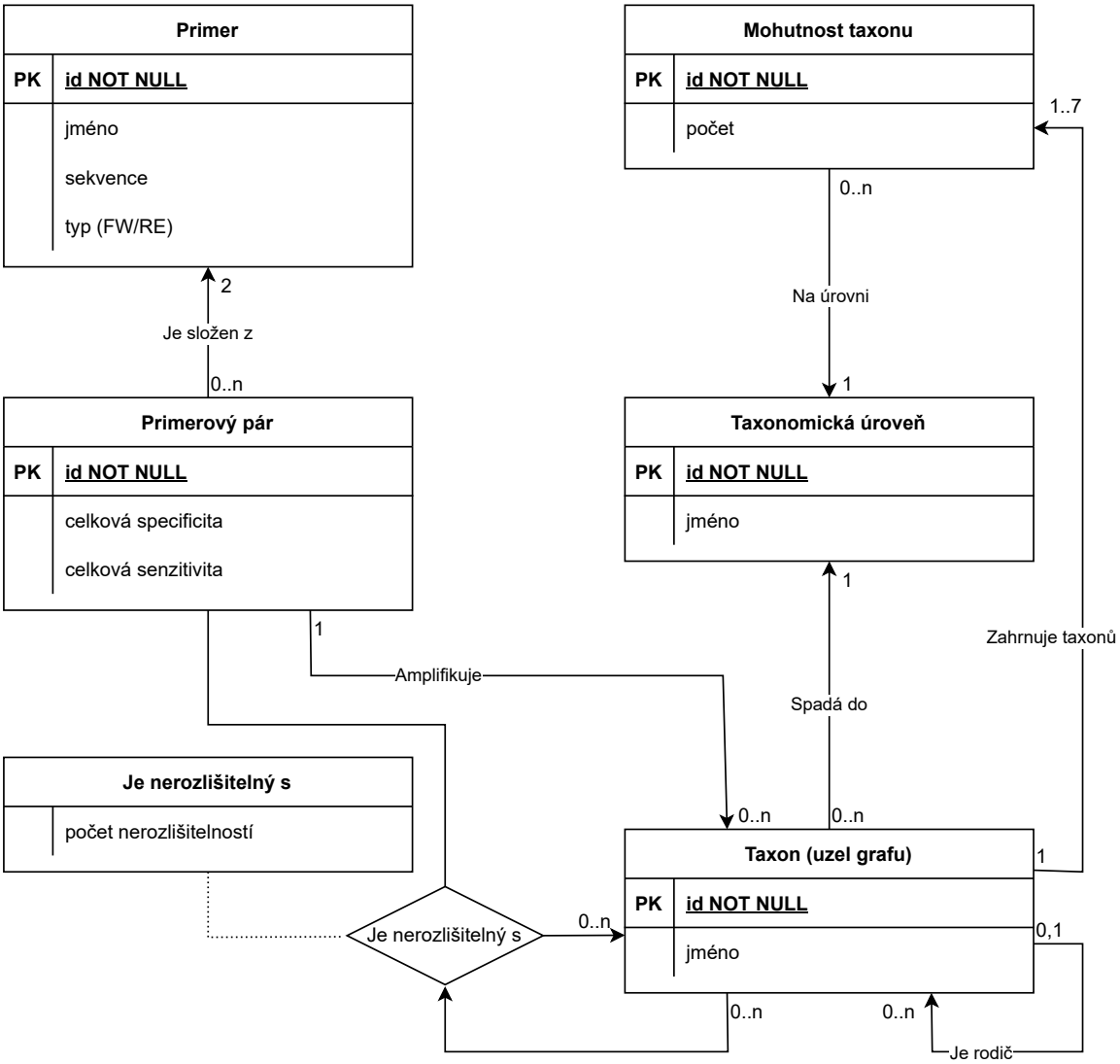
Obrázek 3.2: Fragment grafu reprezentující nerozlišitelnosti na úrovni druhů. V grafu jsou uzly představující taxony a hrany mezi uzly reprezentující nerozlišitelnosti sekvencí mezi těmito taxony. Konkrétní příklad z grafu: Celkem 13 druhů z kmene Actinobacteria je nerozlišitelných s nějakým druhem v kmenu Firmicutes. Zároveň 3646 druhů v rámci kmenu Actinobacteria bude nerozlišitelných s nějakým druhem ve stejném kmenu.

sice databáze a i implementace výpočtu informací o primerovém páru podporuje různé rozlišení, ale na úrovni aplikace se pracuje pouze s rozlišením na úrovni druhů.

Parametr rozlišení určuje taxonomickou kategorii na níž se počítá senzitivita, specificita a nerozlišitelnosti. Příklad lze ukázat na obrázku 3.2. Jedná se o graf s uzly na úrovni kmenu a rozlišením na úrovni druhu. (Všechny uzly v jednom grafu jsou vždy na stejné úrovni.) Při rozlišení na úrovni druhů, hodnoty hran odpovídají počtu nerozlišitelných druhů. Pokud by se jednalo o graf s jiným rozlišením, například čeleď, pak by hodnoty hran vyjadřovaly množství nerozlišitelností právě na úrovni čeledi. V tu chvíli by se hodnoty na hranách pohybovaly v podstatně menších číslech, protože čeledí je v celé taxonomii podstatně méně než druhů.

Důvodem proč by mohlo být vhodné operovat i s parametrem rozlišení je neproporční množství druhů v rámci například jednotlivých rodů. Konkrétně se lze podívat na situaci, kdy poměrně intenzivně zkoumaný rod Escherichia má v použité verzi taxonomie 104 identifikovaných druhů. Na rozdíl od toho například rod Rhodobaca má druh pouze jeden. A právě tento nepoměr může způsobovat výkyvy senzitivity a specificity. Protože je pravděpodobné že jeden primerový pár amplifikuje většinu druhů Escherichia, ale na druhou stranu je otázka jestli je bude možné navzájem rozlišit. Proto by v určitých situacích mohl parametr rozlišení pomoci.

Další motivace, pro budoucí integraci parametru rozlišení i do logiky aplikace, souvisí s problematikou klasifikace sekvencí. Velmi rozšířený RDP klasifikátor umožňuje klasifikaci sekvencí na úroveň rodu. Z čehož vyplývá, že analýza nerozlišitelností na úrovni druhu je sice přesnější co se týče druhové klasifikace, ale pro případy, kdy je potřeba klasifikovat na vyšší taxonomické úrovni, mohou být výsledky zkreslené, právě z důvodu nerovnoměrného množství identifikovaných druhů jednotlivých kmenů. Protože primerový pár může být schopen dobře senzitivovat a specifikovat kmen od ostatních, ale nemusí být schopen dobře specifi-



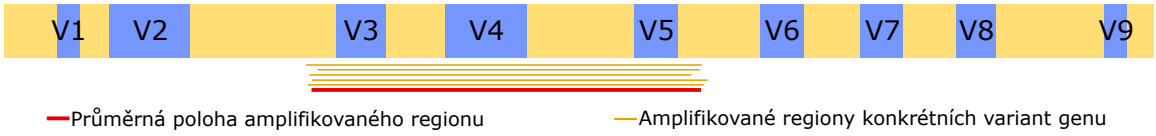
Obrázek 3.3: ER diagram datového modelu, který zachycuje strukturu dat a informací o primerových párech a jejich vlastnostech.

kovat jednotlivé druhy, což v případě kdy je identifikováno mnoho druhů v jednom kmeni, může znatelně snížit hodnotu senzitivity.

3.4 Algoritmus analýzy primerového páru

Cílem analýzy primerového páru je výpočet informací o senzitivitě, specificitě a poloze amplifikovaného regionu, na základě jednotlivých primerových sekvencí. Základní podoba algoritmu se skládá z několika částí:

1. Extrakce amplifikovaných sekvencí jednotlivých variantů pomocí nástroje V-ripper.
2. Identifikace nerozlišitelností mezi jednotlivými strainy. Nerozlišitelnost je definována jako identita amplifikované sekvence se sekvencí jiného strainu.



Obrázek 3.4: Mechanizmus výpočtu polohy amplifikovaného regionu specifikým primerovým párem.

3. Výpočet polohy amplifikovaného regionu genu.
4. Tvorba grafů nerozlišitelností a výpočet množství amplifikovaných a nerozlišitelných druhů (respektive taxonů na úrovni rozlišení) pro každý jednotlivý taxon.

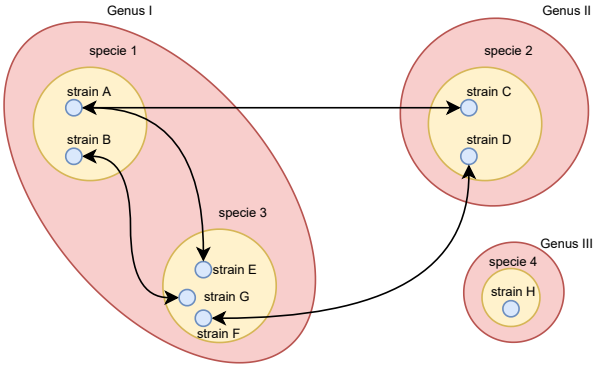
První bod, tedy výpočet amplifikovaných sekvencí, je ve své podstatě celý odveden v rámci nástroje V-ripper. Ten jako vstup dostane primerový pár a na jeho základě určí pro všechny předložené 16S rRNA sekvence amplifikovanou podsekvenci genu.

Druhý bod je časově nejnáročnější. Využívají se při něm vypočítané podsekvence z programu V-ripper. Při výpočtu nerozlišitelností všech amplifikovaných sekvencí je nutné provést vzájemné porovnání všech získaných podsekvencí. Vzájemným porovnáním se na základě identity sekvencí identifikují vzájemně nerozlišitelné taxony na úrovni *strain*. Jedná se o triviální identifikaci nerozlišitelnosti, která ale vylučuje existenci chyb vzniklých chybnou klasifikací pomocí klasifikátoru a lze ji považovat za určitou *ground-truth* klasifikaci.

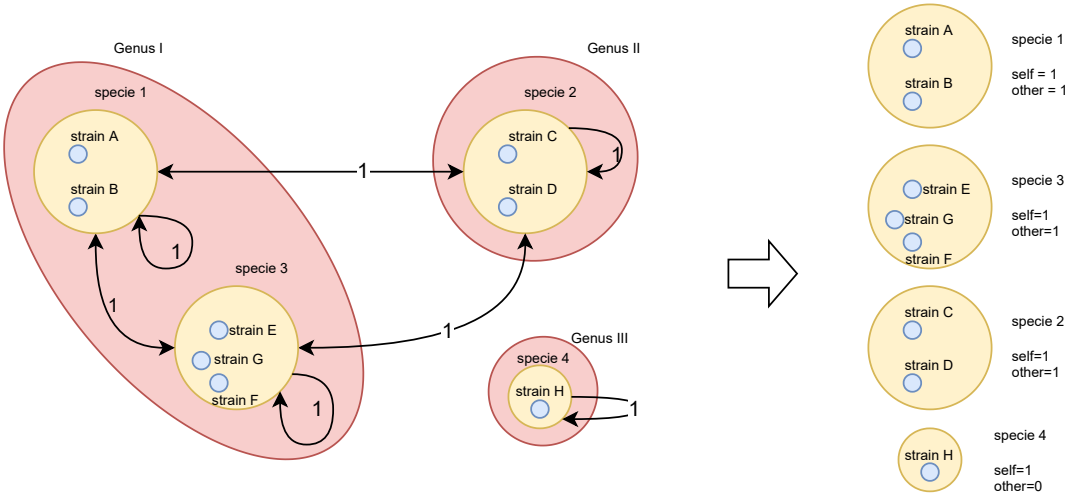
Mírně odlišným přístupem, by mohlo být využití RDP klasifikárotu a provedení klasifikace každé amplifikované sekvence. Na základě chybných klasifikací by se následně mohly identifikovat nerozlišitelné taxony. Díky tomu by se výsledky nerozlišitelností přiblížily výsledkům experimentů, které využívají klasifikátoru RDP. Pro potenciální budoucí rozšíření aplikace, by tato funkcionalita byla vhodným kandidátem k implementaci. Ovšem pouze jako doplněk ke stávajícím datům, nikoliv jako náhrada.

Třetím bodem je určení zacíleného regionu genu. Přestože je 16S rRNA gen poměrně konzervovaný, jeho délka není jednotná a v různých organismech se může mírně lišit. Z toho důvodu je délka genu určena jako průměr délek všech variant genu a při výpočtu je normalizovaná na délku 0 až 1. Zachycený region je vypočítán na základě průměrného umístění přes všechny amplifikované sekvence a poloha je promítnuta na polohu právě v intervalu 0 až 1. Proces výpočtu polohy amplifikovaného regionu je znázorněn na obrázku 3.4

Poslední, stěžejní a nejrozsáhlejší bod výpočtu je tvorba grafů nerozlišitelností jednotlivých taxonů a výpočet množství amplifikovaných a nerozlišitelných dceřiných taxonů pro každý jednotlivý taxon. Celý proces jsem znázornil postupně na několika obrázcích.



Obrázek 3.5: Zjednodušený ilustrační graf, který je získán po určení nerozlišitelností na úrovni strainů. Nerozlišitelnosti jsou vyobrazeny pomocí hran mezi uzly grafu.

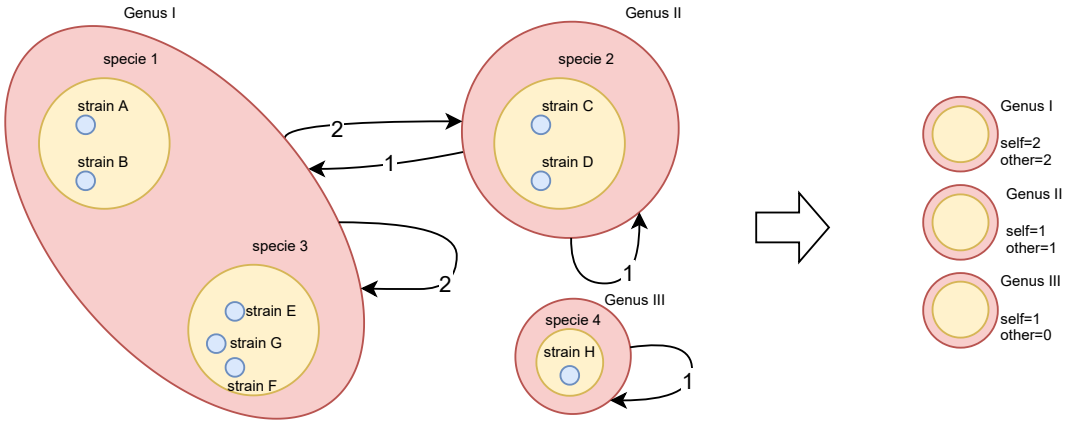


Obrázek 3.6: Graf agregovaný na úroveň a rozlišení druh. Vpravo jsou znázorněny zesumarizované informace on množství amplifikovaných a nerozlištelných druhů v rámci všech taxonů na úrovni agregace.

Výsledkem identifikace nerozlišitelností jednotlivých amplifikovaných sekvencí, jsou data, která lze interpretovat jako graf nerozlišitelností, kde uzly jsou jednotlivé *strainy* a hrany reprezentují nerozlišitelnosti. Podoba takového grafu je vybrazena na obrázku 3.5. V tomto obrázku je pro jednoduchost a přehlednost vybrazeno zařazení pouze do tří taxonomických tříd (strain, druh, rod). hrany mezi uzly znázorňují vzájemně nerozlišitelné strainy.

Tento počáteční graf je následně agregován na úroveň rozlišení druh. Výsledek procesu agregace je vyobrazen na obrázku 3.6. Mechanizmus výpočtu je takový: Jestliže existovala, alespoň jedna hrana mezi strainy druhu A a druhu B spoj tyto dva druhy hranou s hodnotou 1. Zároveň každému uzlu na úrovni druhu přidej hranu do sebe samotného.

Z grafu se následně vypočítají sumarizované informace o množství amplifikovaných taxonů na úrovni rozlišení (v této chvíli druh) tímto způsobem: Pro každý taxon se určí počet



Obrázek 3.7: Graf agregovaný na úroveň rodu (genus) a rozlišení druh. Vpravo jsou znázorněny zesumarizované informace on množství amplifikovaných a nerozlišitelných druhů v rámci všech taxonů na úrovni agregace.

dceřiných taxonů na úrovni rozlišení, které mají hranu sami do sebe. Protože všechny amplifikované taxony mají hranu sami do sebe, efektivně se jedná o jejich součet. Hodnota, která reprezentuje celkové množství je v databázi označena jako *self*.

Podobným způsobem se vypočítá i počet nerozlišitelností na úrovni rozlišení (v této chvíli druh): Pro každý taxon se určí počet dceřiných taxonů na úrovni rozlišení, které mají hranu do jiného taxonu. V tuto chvíli se nepočítá s počtem hran ani jejich hodnotou, počítá se pouze množství taxonů, které jsou z nějakým jiným nerozlišitelné. Hodnota, která reprezentuje celkové množství nerozlišitelných taxonů je v databázi označena jako *other*.

Tím se získají veškeré informace na dané taxonomické úrovni a provede se agregace grafu na vyšší úroveň. Výsledek takové agregace je vyobrazen na obrázku 3.7. Agregace na zvolenou agregovanou úroveň se provede, jako součet hodnot hran mezi taxony na úrovni rozlišení. Jestliže existují dva taxony na úrovni agregace, jejichž dceřiné taxony jsou mezi sebou spojeny hranou, pak tyto dva taxony spoj hranou jejíž hodnota se rovná součtu hodnot hran mezi dceřinými taxony.

Následně se stejným způsobem vypočítají hodnoty self a other a postupně se vypočítají grafy agregované na všech 7 taxonomických úrovních.

Výsledkem analýzy je tedy množina 7 grafů, kde každý graf obsahuje taxony a jejich nerozlišitelnosti na jedné taxonomické úrovni. Z těchto grafů nerozlišitelností jsou odstraněny uzly, které neobsahují žádné nerozlišitelnosti. Zároveň každý graf má k sobě přiřazené agregované informace o každém amplifikovaném taxonu na dané úrovni. Do těchto agregovaných informací spadají hodnoty *self* a *other*.

Optimalizovaná verze analytického algoritmu abstrahuje a zjednodušuje uvedený algoritmus na práci v rámci jedné tabulky. Popsaný algoritmus a práce nad popsanými grafy, lze realizovat jako nad tabulkou. Tato tabulka obsahuje v řádcích jednotlivé hrany (nerozlišitelnosti včetně rekurzivních hran v rámci uzlů) mezi taxony na úrovni druhu. Sloupce tabulky definují vstupní a výstupní taxon (uzel grafu) hrany, včetně celé jeho taxonomické

...	z uzlu			do uzlu		...
	Genus	Specie		Specie	Genus	
...	Genus I	specie 1	1	specie 1	Genus I	...
...	Genus I	specie 1	1	specie 2	Genus II	...
...	Genus I	specie 1	1	specie 3	Genus I	...
...	Genus II	specie 2	1	specie 2	Genus II	...
...	Genus II	specie 2	1	specie 1	Genus I	...
...	Genus II	specie 2	1	specie 3	Genus I	...
...	Genus I	specie 3	1	specie 3	Genus I	...
...	Genus I	specie 3	1	specie 1	Genus I	...
...	Genus I	specie 3	1	specie 2	Genus II	...
...	Genus III	specie 4	1	specie 4	Genus III	...

Tabulka 3.2: Ilustrační část tabulky, reprezentující orientovaný graf nerozlišitelností. Řádky odpovídají jednotlivým hranám a sloupce definují zdrojový a cílový uzel hrany s uvedenou hodnotou.

...	z uzlu		do uzlu	...
	Genus		Genus	
...	Genus I	2	Genus I	...
...	Genus I	2	Genus II	...
...	Genus II	1	Genus I	...
...	Genus III	1	Genus III	...

Tabulka 3.3: Tabulka ilustrující provedenou agregaci tabulky 3.2 na úroveň rodu (genus) s rozlišením druh.

hierarchie. Následné agregace a sumarizace jsou už pouze operace seskupení podle specifických sloupců tabulky. Uzly grafu jsou pouze implicitně definované v rámci jednotlivých řádků tabulky. Uzel je tedy deklarován jeho existencí v tabulce hran.

Je-li například cílem vypočítat graf nerozlišitelností agregovaný na úroveň rodu s rozlišením druh. Realizuje se seskupení tabulky dle rodových sloupců zdrojového a cílového uzlu, přičemž se při seskupení vypočítá suma unikátních druhů z nichž směřuje hrana do jiného druhu.

Podobným mechanizmem získáme i hodnoty *self* a *other*. V tomto případě probíhá seskupení pouze podle sloupce rodu zdrojového uzlu a realizuje se určení počtu uzlů na úrovni druhu (rozlišení), které mají hrany samy do sebe, respektive do jiného uzlu.

Popsaný mechanizmus agregací grafů nad tabulkou je ústřední částí optimalizace celého výpočtu. Pro zjednodušení práce nad touto tabulkou jsem využil knihovny Pandas [69, 60], která umožňuje snadnou a především velmi rychlou a paměťově optimalizovanou práci s datovými rámci v jazyce Python. Výpočet agregovaného grafu je pak díky této knihově otázkou zlomku vteřiny.

3.5 Inference primerového páru

Stěžejní částí funkcionality celé aplikace je právě algoritmus ohodnocení veškerých primerových párů v celé databázi na základě uživatelského dotazu. Celý proces ohodnocení a speciálního seřazení primerových párů je v aplikaci pojmenován **inference**. Částečně byla tato problematika již nastíněna v sekci 3.1.

Idea inference je založena skutečnosti, že různých kombinací primerových sekvencí je nepřeberné množství. Každý takový pár má jiné vlastnosti a cílí jinou množinu organismů. Pokud by tedy při zkoumání zastoupení určitého bakteriálního kmene ve vzorku, byl použit primerový pár, který jej neamplifikuje, případně neumožňuje jeho rozlišení s jinými kmeny, výsledky by byly značně nepřesné. Výběrem správného primerového páru je ale možné přesnost podobných zkoumání maximalizovat, což je hlavní motivací implementace inference.

Stručně popsaný průběh inference, začíná uživatelskou volbou cílených taxonů. Na základě zvolených taxonů, kterých může být libovolné množství z libovolných vrstev taxonomické hierarchie, se realizuje ohodnocení vybrané senzitivity a specificity každého primerového páru. Na základě vybrané senzitivity a specificity se vypočítají Pareto množiny, do nichž jednotlivé páry spadají. Nakonec se primerové páry seřadí podle několika klíčů v tomto pořadí: počet Pareto dominancí (vzestupně), vybraná senzitivita (sestupně), vybraná specificita (sestupně), celková senzitivita (sestupně) a celková specificita (sestupně). Tímto způsobem seřazené pole primerových párů upřednostňuje Pareto optimalní řešení nad ostatními méně optimálními. Zároveň upřednostní vybranou senzitivitu nad vybranou specificitou. Poslední dva parametry řazení, celková senzitivita a specificita, už výsledné pořadí změní spíše minimálně.

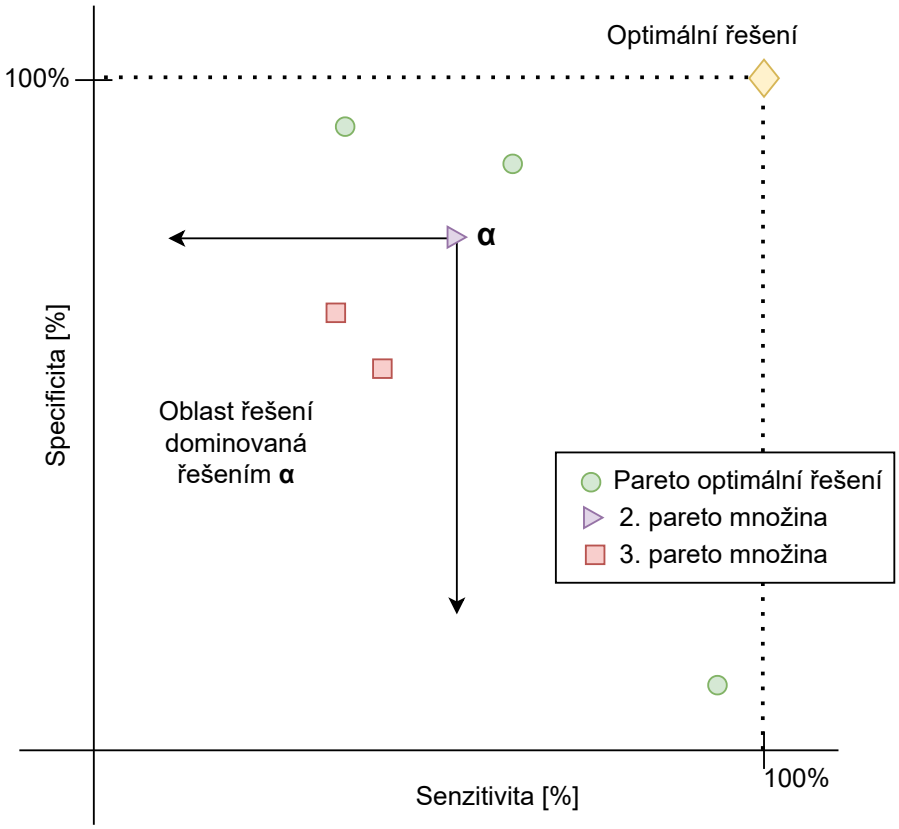
Využití a princip Paretova optima [51]. Ve své podstatě, je výběr primerového páru pouze multikriteriální rozhodování na základě několika parametrů. Vybrané dva hlavní parametry, podle kterých se ve výsledku rozhoduje je senzitivita a specificita primerových párů pro uživatelem zvolené taxony. Protože neexistuje primerový pár, který by měl oba parametry ideální, je nutné se rozhodovat na základě kompromisu obou kritérií.

Častým problémem, který nastává při multikriteriální optimalizaci parametrů, je vzájemný konflikt rozhodovacích parametrů. To znamená, že se zlepšující se hodnotou jednoho parametru, se obvykle ostatní zhoršují. Stejný případ nastává i při výběru primerového páru. Vysoká senzitivita, je většinou spojena s nižší specificitou a naopak. Pro zjednodušení a systematizaci procesu se používá mechanizmu Paretova optimálního řešení, se kterým úzce souvisí pojmy **Paretovi dominance** a **Paretovi množiny**.

V případě inference se za možná řešení považují jednotlivé primerové páry. Pro každý primerový pár, lze určit počet ostatních primerových párů, kterými je na základě zvolených parametrů (senzitivity a specificity) dominován. Dominance jednoho řešení druhým, je situace, kdy dominované řešení má všechny (v tomto případě obě) kritéria horší než dominující řešení.

Podle počtu dominujících řešení, lze jednotlivá řešení shlukovat do Paretovi množiny. Jedná se o množinu řešení, kde jsou všechna řešení dominována stejným množstvím jiných řešení.

Vizualizace pojmů souvisejících s Paretovou optimalizací je zobrazena na obrázku 3.8.



Obrázek 3.8: Vizualizace pojmů souvisejících s Paretovou optimalizací.

Triviální algoritmus výpočtu senzitivity a specificity. Díky reprezentaci taxonomické hierarchie v databázi, je možné pro všechny zvolené taxony určit množinu druhů, které zahrnují. Jakmile uživatel zvolí množinu taxonů, z databáze se získá množina všech taxonů na úrovni druhu, které do nich spadají. Následně se pro každý primerový pár v databázi vypočítá průnik kompletní množiny všech druhů a množiny senzitivovaných druhů. Výsledkem takového průniku je množina druhů, které jsou předmětem uživatelského dotazu a zároveň jsou daným párem senzitivovány. Jestliže množinu všech uživatelem vybraných druhů označíme A a množinu všech amplifikovaných druhů z množiny A označíme B, pak výpočet senzitivity jednoho primerového páru pro zvolené taxony je

$$\texttt{senzitivita [\%]} = (|B|/|A|) * 100.$$

Reálná implementace využívá hodnoty *self*, která je na úrovni druhu vždy rovna 1, pokud je druh senzitivován. Sumou hodnoty self, přes zvolené taxony, získáme v tomto případě hodnotu ekvivalentní jejich počtu. Výsledná podoba rovnice ekvivalentní k implementaci pak je

$$\texttt{senzitivita [\%]} = (\sum_{b}^{B} b_{self}/|A|) * 100.$$

Využití hodnoty *self* při implementaci souvisí především s unifikací s výpočtem specificity. Při výpočtu specificity, je potřeba zjistit. kolik druhů z množiny vybraných amplifikovaných druhů, dříve označené jako B, je nerozlišitelných. Nerozlišitelnost taxonu je reprezentována hodnotou *other*, která na úrovni druhu nabývá hodnot 0 (rozlišitelný) a 1 (nerozlišitelný). Pro výpočet specificity je implementována rovnice:

$$\texttt{specificita [\%]} = (1 - \sum_{b}^{B} b_{other} / \sum_{b}^{B} b_{self}) * 100.$$

Z popisu této triviální verze algoritmu, je zřejmé, že se nejedná o optimální verzi algoritmu hned z několika důvodů. Prvním důvodem, je velká závislost na vybraných taxonech v ohledu množství dat, se kterými se při výpočtu pracuje. Extrémním případem muže být volba taxonu na úrovni domény, například domény Bakterie. Doména Bakterie obsahuje v databázi 18174 druhů a celkový počet primerových párů je 5661. Pro výpočet je nutné získat hodnotu z databáze pro každý jednotlivý druh. Z toho vychází že pro každý primerový pár by se realizovala operace s řádově tisíci záznamy v databázi a vzhledem k počtu primerových párů by se celkový počet zpracovávaných záznamů vyšplhal na miliony záznamů. Tato zkutečnost znamená velký nápor na databázový server a pomalý průběh výpočtu vybrané senzitivity, respektive specificity.

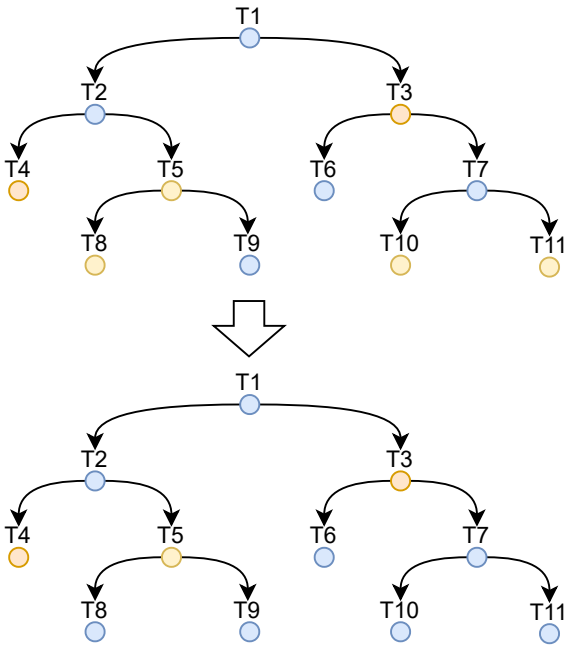
Přesto tato triviální varianta v průběhu vývoje plnila funkci kontroly výsledků optimalizované verze výpočtu.

Optimalizovaný algoritmus výpočtu senzitivity a specificity využívá dat uložených v databázi takovým způsobem aby se maximálně omezilo množství dat s nimiž se pracuje. Je toho docíleno plným využitím taxonomické hierarchie uložené v databázi v kombinaci s existencí agregovaných hodnot *self* a *other* pro každý taxon a každý primerový pár v databázi.

Jako první příklad uvedu stejnou situaci jako u triviální implementace. Uživatel vybere jako cílený taxon doménu Bakterie. Tento taxon má pro každý primerový pár uložené hodnoty *self* i *other*, v rámci jednoho řádku databáze. Díky tomu je možné vypočítat vybranou senzitivitu a specificitu, pomocí selekce 5661 (celkový počet párů) řádků z jedné tabulky. Celkový počet druhů zahrnutých v rámci jednoho taxonu je uložen v další tabulce, z níž by proběhla selekce pouze jednoho řádku (zjištění počtu druhů v doméně Bakterie). A pokud se nebudou započítávat podpůrné dotazy, související s uživatelským rozhraním a jinými maličkostmi, celkový počet dotázaných řádků z databáze je 5662, což je oproti řádově milionům u triviální implementace značné zlepšení.

Jako druhý příklad uvedu optimalizovaný výpočet pro dva zvolené taxony jejichž, množiny druhů jsou vzájemně disjunktní. Celý výpočet probíhá obdobným způsobem, s jediným rozdílem že hodnoty *self* obou taxonů se pro zjištění celkového počtu amplifikovaných druhů sečtou, obdobně i hodnoty *other*. Z databáze se načítají hodnoty pro dva taxony a tedy místo 5625 řádků se jich načte 11250. Je zde tedy lineární závislost množství dat získávaných z databáze v závislosti na počtu zvolených taxonů.

Komplikace a netriviálnost algoritmu se objeví v situaci, kdy uživatel zvolí dva, nebo více taxonů, z nichž alespoň některé nemají disjunktní množinu druhů. Jeden z taxonů, v rámci hierarchie, spadá pod druhý. V tomto případě není možné pouze jednoduše sčítat hodnoty jednotlivých taxonů, protože by jsme určité druhy započítávaly duplicitně.



Obrázek 3.9: Ilustrace eliminace taxonů, které už jsou pokryty jiným taxonem z vyšší úrovně. Žluté uzly představují taxony zvolené uživatelem a modré představují zbytek hierarchie. Lze si povšimnout že po eliminaci zůstaly zvoleny pouze vzájemně disjunktní taxony.

Finální podoba algoritmu nejdříve identifikuje takové taxony (z dotázaných taxonů), které jsou pokryty nějakým jiným taxonem. Takové taxony je potom možné při navazujícím výpočtu ignorovat, protože jsou již zahrnuty v taxonu vyšší úrovně. Příklad eliminace pokrytých taxonů je vyobrazen na obrázku 3.9. Následně pokračuje výpočet již popsaným způsobem. Tedy vypočítá se suma hodnot *self* a suma hodnot *other* všech zvolených taxonů a na základě množství druhů, které zvolené taxony zahrnují, se vypočítá vybraná senzitivita a specificita.

Vypočítané hodnoty se následně seřadí a zobrazí uživateli, který může výsledky dále podrobněji procházet a filtrovat.

Kapitola 4

Implementace a výkonnostní vyhodnocení

Implementace celé aplikace se skládá z několika částí. Od nejviditelnější části aplikace, kterou tvoří webové uživatelské rozhraní, přes logiku inference primerového páru, aplikační rozhraní REST, až po implementaci a optimalizaci algoritmu analyzujícího vlastnosti primerového páru.

4.1 Vybrané technologie

Jedna z prvotních otázek, kterou je potřeba položit před tvorbou nejen webové aplikace, je především volba programovacího jazyka, potažmo knihoven a jiných nástrojů, které budou použity. Volba kvalitních a rozšířených nástrojů a knihoven, je rozhodující nejen pro aktuální implementaci, ale i pro budoucí udržitelnost a případnou rozšiřitelnost aplikace.

Výběr a celková architektura aplikace vychází z analýzy požadavků shrnutých v sekci 3.1.

Hlavní motivací pro tvorbu právě webové aplikace je aspekt jednoduchosti použití aplikace a téměř naprostá nezávislost na uživatelově systému a platformě. Jediné co je potřeba pro běh aplikace, je webový prohlížeč a přístup k internetu. Zároveň vydání nástroje v podobě webové aplikace kopíruje trend i jiných, nejen bioinformatických, aplikací, které už několik let získávají na velké popularitě.

Právě masivní rozšíření webových aplikací samozřejmě souvisí i s intenzivním vývojem knihoven a nástrojů zjednodušujících a urychlujících jejich vývoj. Konkrétně lze jmenovat sady knihoven (frameworky) pro jazyk PHP jako jsou CakePHP nebo Laravel, případně pro jazyk Ruby lze zmínit například Ruby on Rails a v neposlední řadě i Flask a Django pro jazyk Python.

Po zhodnocení několika faktorů, jako je použitý programovací jazyk, aktivita vývoje projektu a mimo jiné i popularita s čímž přímo souvisí velikost komunity, jsem jako programovací jazyk zvolil Python 3 doprovázený frameworkem Django [46].

Framework Django je velmi rozšířený a aktivně vyvíjený opensource framework pro tvorbu webových aplikací s širokou komunitou a nesčetným množstvím různých doplňujících knihoven. Umožňuje vývoj backendové, ale i frontendové části webové aplikace. V rámci

samotného Djanga je implementováno i objektově-relační mapování (ORM) zjednodušující práci s relačními databázemi (na rozdíl od frameworku Flask, jenž se často používá v kombinaci s knihovnou SQLAlchemy, která ORM implementuje). Dále Django nabízí mnoho dalších nástrojů a přidružených knihoven třetích stran, jako jsou: odlehčený webový server určený pro vývoj, implementované knihovny pro webovou cache, REST api a mnoho dalších z velké části komunitně vyvíjených knihoven.

Protože aplikace je napsaná v jazyce Python, není možné ji přímo provozovat prostřednictvím standardního HTTP serveru, jako je například Apache [2]. Pro provoz Django aplikace je nutné využít specializovaného serveru. Konkrétně jsem se rozhodl pro server Gunicorn [18].

Django pro tvorbu frontendové části aplikace využívá šablon, jejichž syntaxe z velké části vychází z šablonovacího systému **Jinja** [20]. Django samo o sobě, ale neobsahuje žádné knihovny přímo určené pro tvorbu webového uživatelského rozhraní. Pro tento účel jsem se rozhodl využít framework Bootstrap [4].

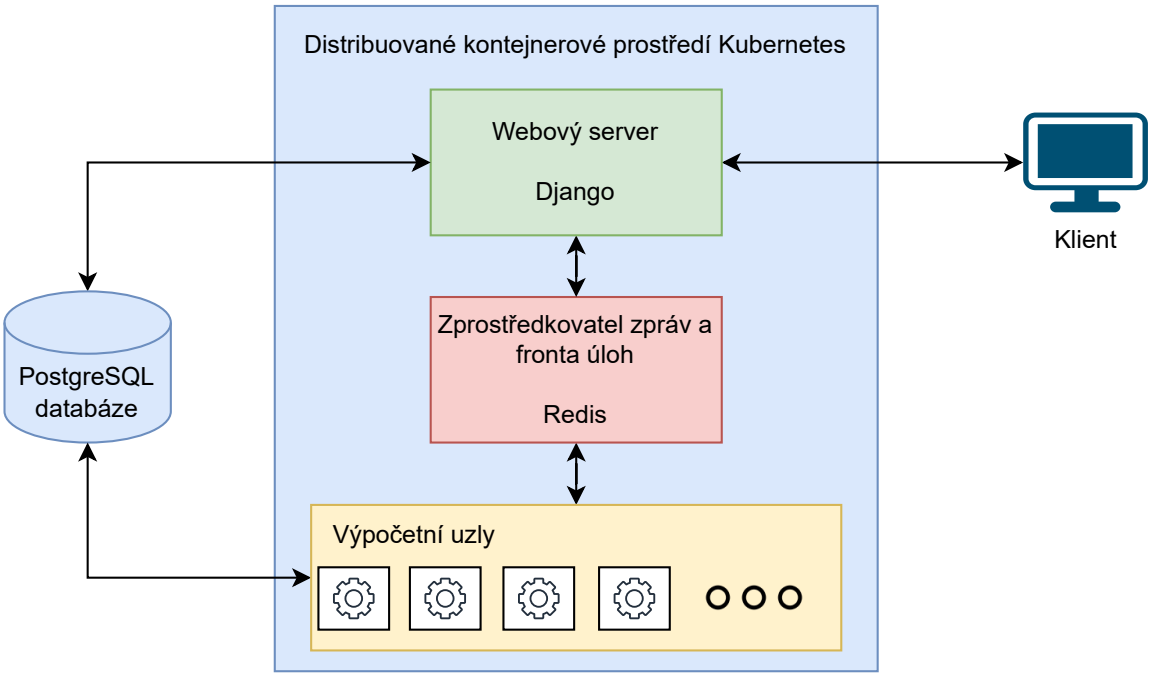
Bootstrap, je opensource framework zaměřený na tvorbu webových uživatelských rozhraní. Ve své podstatě se jedná o sadu CSS a Javascriptových knihoven, které implementují sadu připravených komponent a jiných funkcionalit. Tyto komponenty je možné jednoduše použít a zaintegrovat do stránky. Již naimplementované komponenty lze tedy použít pro tvorbu formulářů, tabulek, tlačítek a mnoha jiných interaktivních prvků, které jsou zároveň navrženy takovým způsobem aby společně tvořily ucelený vizuální dojem. Konkrétně jsem využil aktuálně nejnovější pátou verzi.

Aplikace ke svému běhu potřebuje uchovávat velké množství strukturovaných dat. Proto je relační databáze další důležitou komponentou systému. Zde jsem se rozhodoval ze dvou možných alternativ: **PostgreSQL** [26] a MySQL [24] (respektive MariaDB [23]).

Na počátku řešení projektu jsem dostal přístup k databázi primerových párů. Jednalo se databázový server MySQL a proto prvotní idea byla, pokračovat v řešení práce právě s databází MySQL. Po několika týdnech práce s MySQL databází, jsem ale pozoroval velké propady výkonu při určitých operacích, které zahrnovaly práci s velkým množstvím dat. Jednalo se o situace, kdy se měla agregovat data z velké tabulky (+-120 GB), přičemž databáze byla uložena na poměrně pomalém rotačním disku. Takové dotazy trvaly i desítky minut. Z toho důvodu jsem se rozhodl vyzkoušet jestli PostgreSQL databáze nebude schopna lépe pracovat na pomalém disku s takto velkými daty. Výsledek této změny bylo poměrně výrazné zrychlení a operace, které na MySQL trvaly desítky minut na PostgreSQL, trvaly maximálně jednotky minut. Z toho důvodu jsem se rozhodl u **PostgreSQL** serveru zůstat.

Z analýzy požadavků dále vychází potřeba integrovat frontu úloh, která je potřeba pro uskutečňování analýzy uživatelských primerových párů. Pro tento účel jsem se rozhodoval ze dvou knihoven. První knihovnou je Django-Q [12]. Jedná se o knihovnu implementující frontu úloh, přičemž celá knihovna je úzce provázaná z frameworkem Django. Z toho důvodu se jednalo o moji první volbu. Po delším testování jsem se, ale pro jisté nestability a již téměř rok neaktualizovaný kód, rozhodl přejít na frontu úloh implementovanou knihovnou **Celery** [5].

Knihovna Celery je implementovaná v jazyce Python a je ji možné poměrně jednoduše využít v kombinaci s frameworkem Django. Pro svou funkci potřebuje Celery databázi, která bude schopna udržovat samotnou frontu úloh a zároveň komunikovat s jednotlivými



Obrázek 4.1: Schéma navrženého systému. Většinová část aplikace (aplikační webový server, Redis a výpočetní uzly) je připravena na nasazení v prostředí typu Kubernetes. Mimo PostgreSQL aplikační databáze, která aktuálně počítá s nasazením na separátním dedikovaném stroji.

výpočetními uzly. Pro tento účel jsem zvolil databázový server Redis [27]. Mechanizmus implementovaný v Celery je tedy takový: Nejdříve je v Django aplikaci vytvořen požadavek na spuštění úlohy. Tento požadavek se zpracuje a vyšle na Redis databázi. V ten moment se úloha zařadí do fronty a jakmile je volný některý výpočetní uzel, pošlou se mu informace o úloze a uzel ji začne vykonávat. Jakmile je úloha dokončena, její výsledek se zapíše zpět do databáze.

Řešení fronty úloh pomocí knihovny Celery je dobře horizontálně škálovatelné a umožňuje zapojení velkého množství výpočetních uzlů. Předpoklad je ale takový, že při produkčním nasazení aplikace budou potřeba maximálně jednotky výpočetních uzlů. I přesto je vhodné mít možnost systém horizontálně škálovat.

Nejen z důvodu snadného škálování, ale i z důvodu standardizace nasazení aplikace do produkce, jsem se rozhodl aplikaci připravit na běh v distribuovaném prostředí typu **Kubernetes**[1]. Právě prostředí Kubernetes mimo jiné umožní: standardizované nasazení aplikace do provozu v clusteru, škálování výpočetních uzlů dle vytížení, jejich podrobné monitorování, snadnou správu, nezávislost na hardwaru a v ideálním případě i vysokou dostupnost aplikace.

Schéma navrženého systému je vyobrazeno na obrázku 4.1.

[1] https://kubernetes.io/

	Rychlost [MB/s]					
Operace	HDD instance			SSD instance		
	min	avg	max	min	avg	max
randread	4.6	7.9	11.8	350.6	462.9	542.8
read	8.8	19.6	25.6	470.1	578.6	650.2

Tabulka 4.1: Výsledky měření rychlosti disku na obou vývojových instancích. Z výsledků je zřejmé, že v oblasti rychlosti čtení dat se HDD instance provozovaná v Metacentru nemůže rovnat s výkonem SSD instance. Data jsou průměrem 5 běhů FIO benchmarku s parametry: size=10GB, direct=1, bs=4k, ioengine=libaio, iodepth=256, runtime=30, numjobs=1, time$_b$*ased*.

4.2 Impementace a vyhodnocení inference

Algoritmus inference a důležité aspekty jsou uvedeny v sekci 3.5. Implementace inference v aplikaci, odpovídá popsané optimalizované verzi algoritmu.

Výkon a rychlost algoritmu inference z největší části stojí na rychlosti databáze. Samotný proces získání dat o primerových párech z databáze, je zcela majoritní částí výpočetního času. Tato skutečnost vedla i ke vzniku popsané optimalizované verze algoritmu.

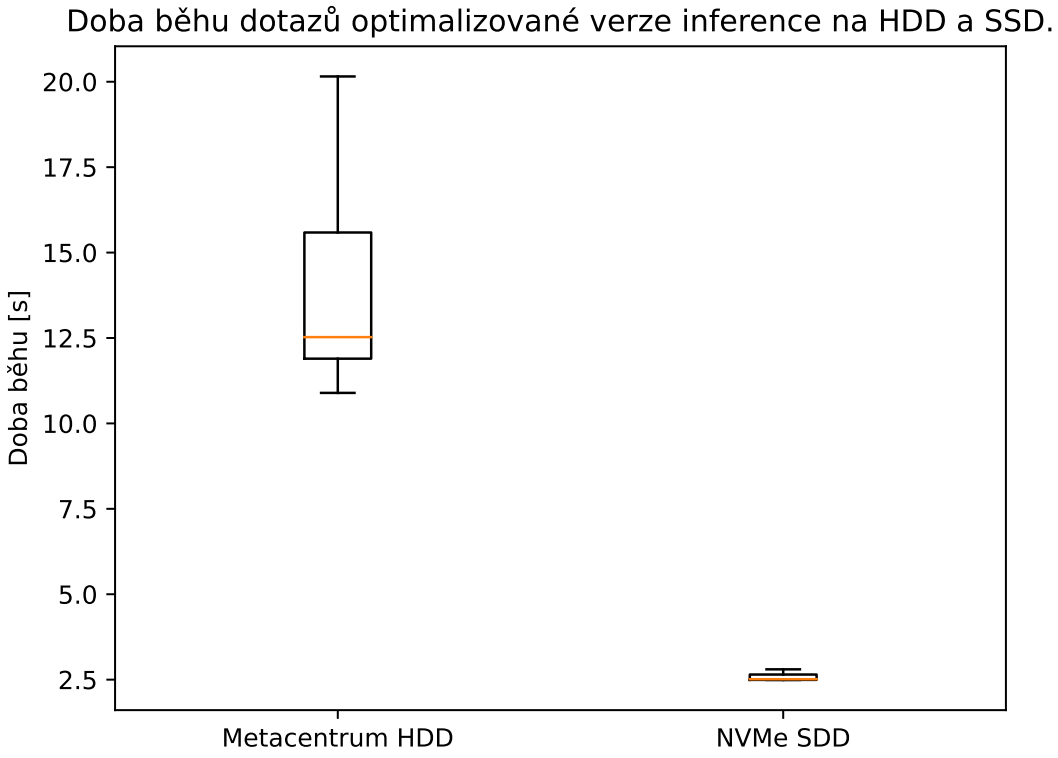
Koncepce aplikace zároveň počítá se schopností provedení inference v řádu nízkých jednotek, maximálně nízkých desítek, vteřin. Tento fakt ještě zvyšuje nároky na rychlost databáze.

Při vývoji aplikace jsem pracoval se dvěma instancemi databáze. První jsem provozoval ve virtuálním stroji v rámci infrastruktury *cloud.muni.cz*. Tato instance má uloženou databázi na rotačním disku. Druhou instanci jsem provozoval na lokálním počítači, kde databáze byla uložena na NVMe SSD disku. Z podstaty technologií se obě instance, velmi liší v rychlosti práce s diskovými operaci, jako čtení a zápis.

Právě rychlost čtení z disku je pro algoritmus inference stěžejní. V rámci jeho výpočtu je nutné velmi rychle získat množství dat z tabulek, které celkem obsahují řádově desítky gigabytů dat. Z toho samozřejmě vyplývá fakt, že lokální instance z SSD diskem umožňuje násobně rychlejší výpočet inference. Pro srovnání jsou v tabulce 4.1 uvedeny naměřené rychlosti IO operací na obou instancích. Hodnoty jsou průměrem 5 měření, přičemž měření jsem provedl pomocí široce používaného benchmarku FIO [14].

Rychlost získání dat z databáze odpovídá trendu z tabulky 4.1. Provedl jsem měření jednoho konkrétního případu inference tří kmenů. Konkrétně se jednalo o kmeny Actinobacteria, Firmicutes, Proteobacteria. Hlavním předmětem měření byla pravě rychlost čtení z databáze. Prakticky tedy měření zahrnovalo posloupnost dotazů na databázi, přičemž jsem měřil dobu provedení všech dotazů. Posloupnost dotazů odpovídá posloupnosti, která je provedena při inferenci tří zmíněných kmenů. Celkem jsem provedl 10 měření pro každou databázovou instanci (SSD a HDD). Mezi každým měřením jsem ukončil databázový server, vymazal systémové mezipaměti a znovu nastartoval server, pomocí příkazů:

```
$> systemctl stop postgresql
$> sync; echo 3 > /proc/sys/vm/drop_caches
$> systemctl start postgresql
```



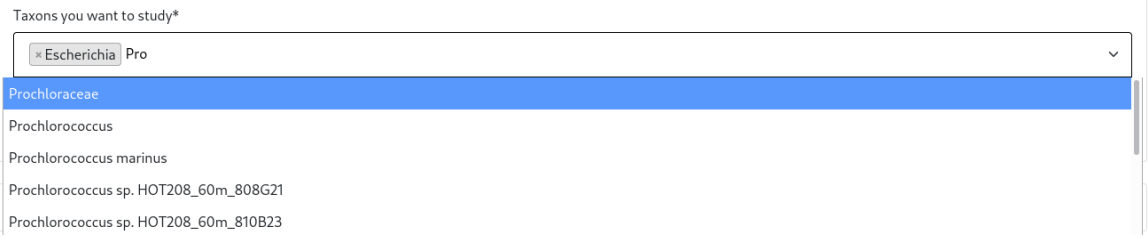
Obrázek 4.2: Výsledky měření času potřebného k vyhodnocení posloupnosti databázových dotazů odpovídajících optimalizované verzi inference pro kmeny Actinobacteria, Firmicutes, Proteobacteria.

Výsledky měření rychlosti dotazů, které odpovídají optimalizované verzi algoritmu inference, jsou vyobrazeny na obrázku 4.2. Z tohoto obrázku je patrný výrazný rozdíl mezi HDD instancí a SSD instancí. Kde databáze HDD instance obsluhuje jednu inferenci přibližně 12 až 15 vteřin, SSD instance dokáže stejnou posloupnost dotazů vyhodnotit přibližně za 2 vteřiny. Zároveň je SSD instance podstatně stabilnější, pokud se jedná o dobu zpracování. Na rozdíl od HDD instance, kde je rozdíl maxima a minima téměř 10 vteřin.

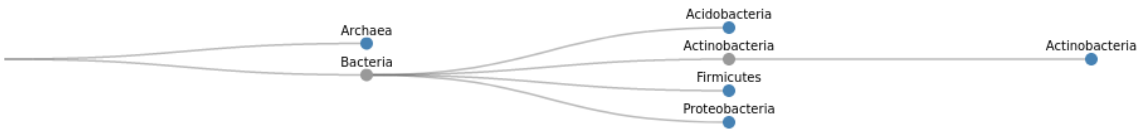
Pro srovnání jsem provedl i měření dotazů, které odpovídají triviální verzi inference. Doba vyhodnocení těchto dotazů na SSD instanci se pohybovala mezi **6-7 minutami**.

Z výsledků lze usoudit, že pro produkční nasazení je velmi vhodné zvolit rychlé, ideálně SSD, úložiště. Výsledky dále ukazují, že databáze a potažmo celá funkcionalita inference bude s rychlostí odpovídající HDD instanci, schopna obsloužit maximálně jednotky uživatelů za minutu. V případě SSD instance se jedná o desítky uživatelů za minutu.

Vybraný databázový server PostgreSQL podporuje možnost vytvoření replikovaných instancí a tedy možnost rozložení požadavků na několik různých strojů. Díky tomu, by bylo v případě potřeby možné zvýšit kapacitu aplikace.



Obrázek 4.3: Snímek textového pole se schopností automatického doplňování. V aplikaci je použit mimo jiné pro výběr taxonů určených k inferenci.



Obrázek 4.4: Podoba vykresleného taxonomického stromu na základě zvolených taxonů pro inferenci. Modře jsou zvýrazněny body taxonů které jsou vybrány a šedě ty, které ne.

4.3 Uživatelské rozhraní

Implementace frontendové části aplikace a uživatelského rozhraní, je provedena s pomocí použitého frameworku Django. Jedná se tedy o implementaci v rámci jednotlivých HTML šablon, které jsou pro uživatele zpracovávány na serverové části aplikace.

Funkcionalita většiny stránek je doplněna o logiku implementovanou v jazyce Javascript. Konkrétně je s pomocí Javacriptu implementována veškerá komunikace se serverovou částí od uživatele (dotazy na api), vykreslování a interakce s veškerými grafy a v neposlední řadě i další interaktivní prvky stránek.

Pro ulehčení tvorby uživatelského rozhraní jsem využil knihoven: Bootstrap [4] (sada komponent a nástrojů na tvorbu uživatelského rozhraní), jQuery (knihovna pro jednoduší práci v rámci javascriptu), Axios [3] (knihovna realizující HTTP dotazy), D3.js [7] (knihovna určená pro jednoduchou manipulaci s HTML,CSS a SVG za účelem tvorby grafů a vizualizace dat), DataTables [8] (knihovna implementující datové tabulky). Zdrojem mnohých navigačních a jiných obrázků použitých v aplikaci je služba Font Awesome [15].

Celkový vizuální dojem z aplikace je do velké míry ovlivněn právě využitou knihovnou Bootstrap. Tato knihovna implementuje specifický vizuální styl, a protože se jedná o široce využívanou knihovnu je zřejmé, že vizuální dojem aplikace nebude působit vyloženě unikátně, ale bude spíše zapadat do vizuálního stylu dnešního internetu. Především se jedná o podobu tlačítek, formulářů a navigačních lišt.

Aplikace a její rozhraní, sestává ze tří hlavních částí: inference, prohlížeč databáze primerových párů a analýza primerového páru. Celá aplikace je uvedena hlavní stranou, která slouží jako informační rozcestník na zmíněné hlavní části, ale i na stránku s krátkou dokumentací, popisem pojmů a rychlými příklady využití aplikačního rozhraní.

Hlavní stránka, její obsah a vizuální podoba umožní uživateli se rychle zorientovat a pokračovat na tu část aplikace, kterou zrovna potřebuje. Její podoba je zachycena v příloze B na obrázku B.1.

Inference by pro uživatele měla být otázkou snadného vyplnění jednoho pole taxonů a kliku na jedno tlačítko. Z toho důvodu je stránka s inferencí z počátku velmi minimalistická. Obsahuje pouze jedno políčko pro taxony a jedno tlačítko.

Políčko pro taxony je vybaveno schopností automatického doplňování. Jakmile uživatel začne do pole psát, přes aplikační rozhraní se stránka začne dotazovat serveru na možné taxony, jejichž jméno začíná na napsaný řetězec. Uživateli je následně nabídnut seznam možných taxonů, ze kterých si může vybrat anebo pokračovat ve psaní. Seznam nabízených taxonů je aktualizován s každou další změnou napsaného řetězce. Konkrétní podoba tohoto pole je zobrazena na obrázku 4.3. Do jednoho pole je tímto způsobem možné zadat libovolné množství taxonů, které je možné kliknutím na křížek odebrat. Ke snadnější implementaci tohoto pole jsem využil knihovny Django autocomplete light [10].

Ve chvíli kdy uživatel zadá do pole nový taxon, vyšle se dotaz na server s informací o zadaném taxonu. Server vrátí informaci o poloze taxonu v taxonomické hierarchii, která se následně vykreslí a zvýrazní vybrané taxony ve formě taxonomického stromu pod vyplňovaným formulářem. Podoba takového vykresleného stromu je zachycena na obrázku 4.4.

Jakmile je inference tlačítkem spuštěna, vyšle se dotaz na server s informací o daných taxonech. Server pomocí algoritmu popsaném v sekci 3.5 ohodnotí všechny primerové páry z databáze, vypočítá tedy pro každý pár vybranou senzitivitu a specificitu a pošle výsledek zpět. Jakmile stránka obdrží výsledek, vykreslí statistiky výsledků, přehledové grafy a souhrnnou tabulku.

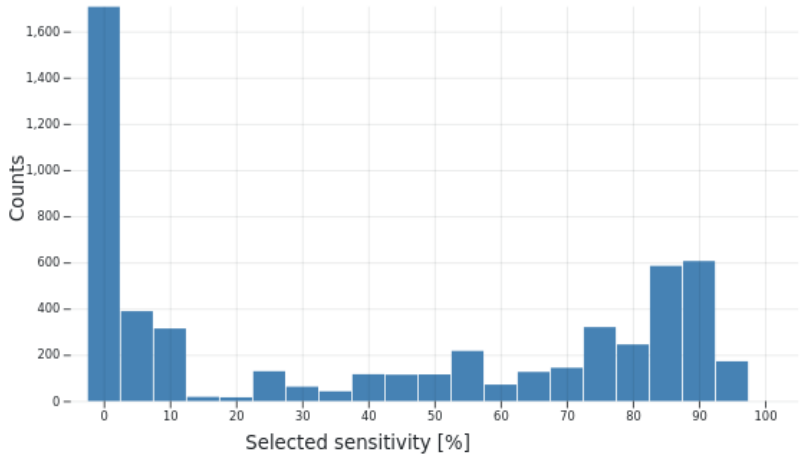
Celkem jsou vykresleny 4 přehledové grafy. První dva grafy ve formě histogramu zobrazují počet primerových párů v závislosti na vybrané senzitivitě, respektive specificitě. Jedná se čistě o přehledové grafy, které neumožňují jakoukoliv filtraci, či jinou interakci s výsledky. Jejich úkolem je rychle předat informaci o rozložení zmíněných dvou veličin. Konkrétní snímek jednoho z histogramů je na obrázku 4.5

Druhé dva grafy, které jsou z výsledků vykresleny, jsou bodové grafy závislosti vybrané senzitivity na vybrané specificitě v prvním grafu, respektive závislosti celkové senzitivity na celkové senzitivitě v grafu druhém. Tyto dva grafy umožňují uživateli vybrat specifikou oblast grafu a tím filtrovat zobrazené primerové páry v přehledové tabulce. Lze filtrovat podle vybraných i celkových vlastností zároveň. V obou grafech jsou primerové páry reprezentovány modrými body, respektive zlatými body, pokud se jedná o primerový pár z pareto optimální množiny. Podoba grafů, včetně podoby mechanizmu výběru oblasti grafu je vyobrazena na obrázku 4.6.

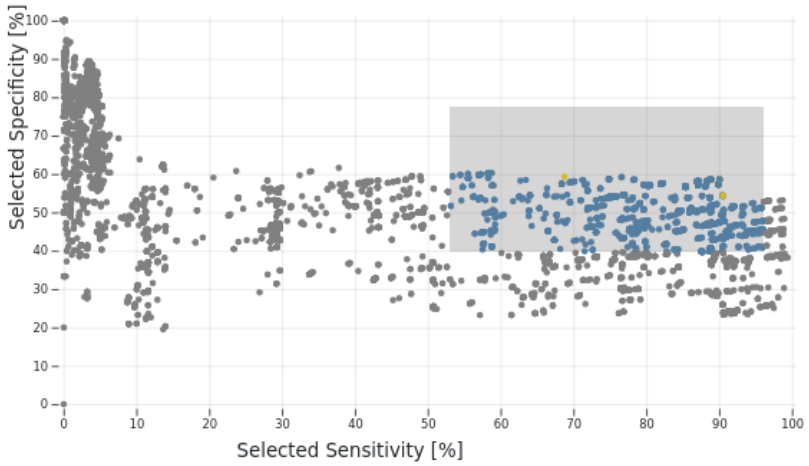
Poslední částí, výsledků inference, je přehledová tabulka všech (případně pouze vyfiltrovaných) seřazených primerových párů. V tabulce jsou v jednotlivých sloupcích uvedeny hodnoty vybrané a celkové senzitivity a specificity a počet Pareto dominujících primerových párů. Řádky v tabulce jsou ve výchozím stavu multikriteriálně seřazeny takovým způsobem jak je popsáno v sekci 3.5. Uživatel může upřednostnit jiné řazení a pomocí kliku na nadpisy jednotlivých sloupců seřadit výsledky jiným způsobem.

Všechny důležité informace o funkcionalitě inference a celé aplikace jsou uvedeny buď v dokumentaci, anebo v informačních boxech strategicky rozmístěných na souvisejících místech jednotlivých stránek.

Kompletní snímek stránky s výsledky inference je zobrazen v příloze B na obrázku B.2.



Obrázek 4.5: Snímek histogramu vybrané senzitivity z výsledku inference.

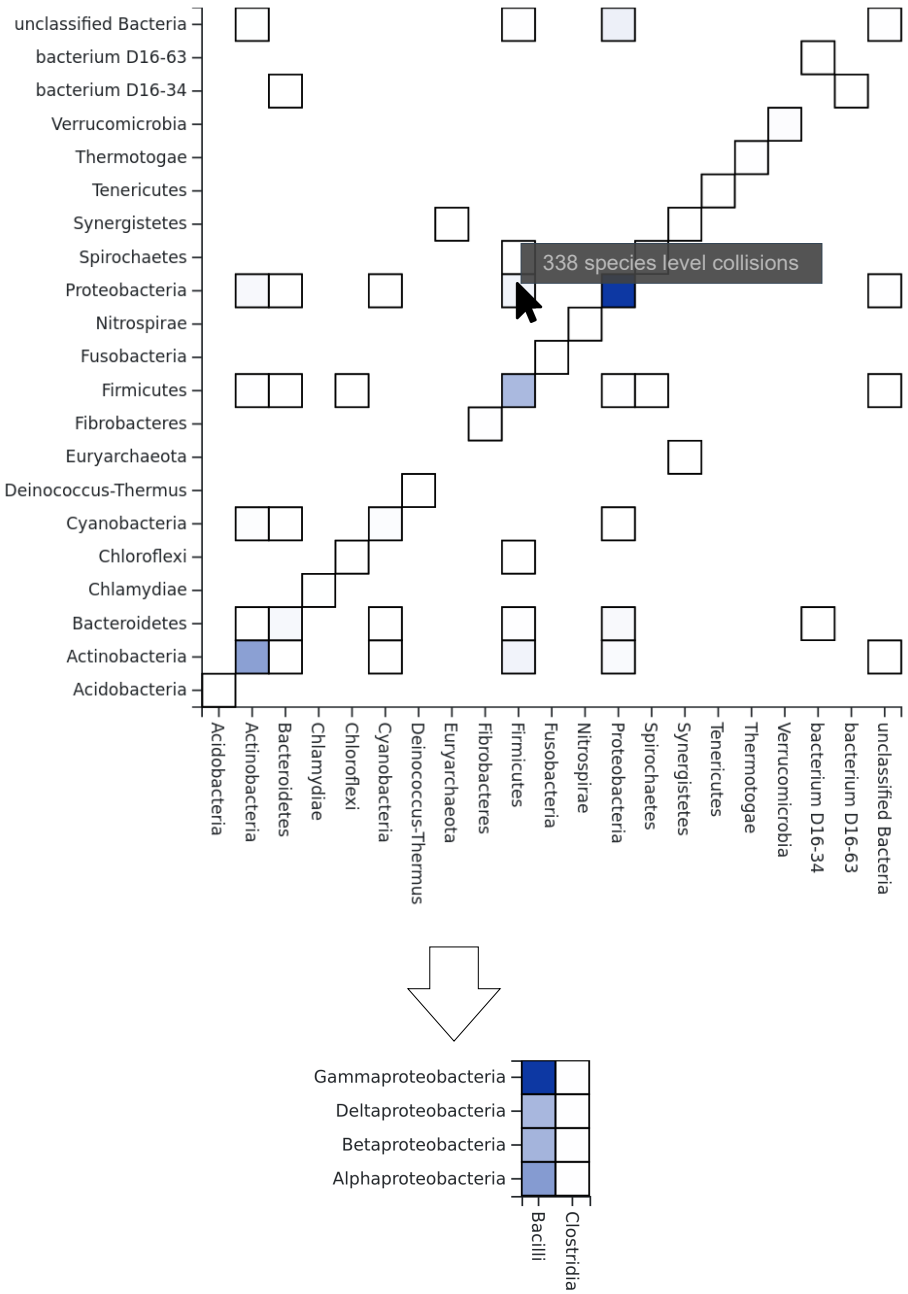


Obrázek 4.6: Snímek bodového grafu závislosti vybrané specificity na senzitivitě primerových párů. Vybraná oblast reprezentovaná šedým obdelníkem realizuje filtraci párů podle vybraných hodnot daných veličin. Ve zvolené oblasti si lze všimnout několika zlatě zbarvených bodů. Jedná se primerové páry z Pareto optimální množiny.

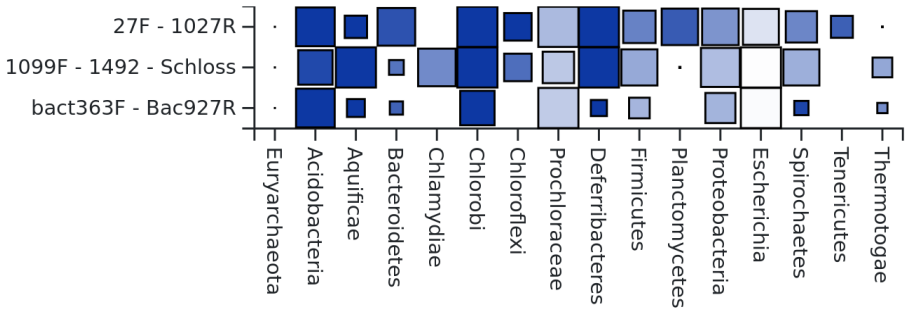
Přehledová tabulka, mimo jiné umožňuje přistoupit na stránku s **detailními informacemi o primerovém páru**. Zde jsou uvedeny sekvence jednotlivých primerů, grafické zobrazení amplifikovaného regionu 16S rRNA genu a především je zde možné zjistit konkrétní nerozlišitelnosti a porovnat primerový pár vůči jiným párům.

Prohlížení nerozlišitelností je implementováno pomocí interaktivní *teplotní mapy*, kde jsou v řádcích a sloupcích taxony a prostřednictvím barev uvnitř matice je vizualizováno množství druhů z taxonu na řádku, které jsou nerozlišitelné s nějakým druhem z taxonu ve sloupci.

V počátečním stavu, jsou zobrazeny všechny taxony na úrovni kmene, mezi kterými jsou nerozlišitelnosti. Uživatel se může postupně proklikávat hlouběji do hierarchie, až na úroveň jednotlivých druhů. Tento proces je ilustrován na obrázku 4.7, kde je nejdříve zobrazen počáteční stav na úrovni kmene a po kliknutí na řádek Proteobacteria a sloupec Firmicutes,



Obrázek 4.7: Ilustrace procesu vizualizace nerozlišitelností mezi jednotlivými taxony. Interaktivita je zde naznačena kurzorem, který by kliknutím způsobil překreslení původní mapy výše na mapu pod ní, která zobrazuje nerozlišitelnosti zvolených taxonů na nižší taxonomické úrovni.



Obrázek 4.8: Vizualizace specificity a senzitivity vybraných primerových párů pro vybrané taxony. Senzitivita je vyjádřena velikostí čtverce v příslušném řádku a sloupci a specificita je následně vyjádřena barvou daného čtverce. Čím větší plocha čtverce tím vyšší senzitivita a podobně čím tmavější modrá tím vyšší specificita. Z této konkrétní vizualizace je jasně patrný vliv výběru správného primerového páru na kvalitu záchytu amplifikace.

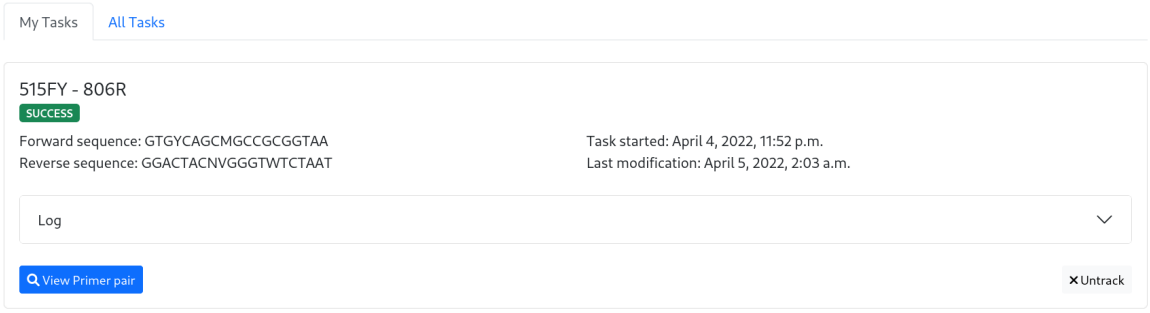
Obrázek 4.9: Snímek vizualizace polohy amplifikovaného regionu v rámci 16S rRNA genu.

je mapa překreslena takovým způsobem, že na řádcích a sloupcích jsou taxony o jednu taxonomickou úroveň níže, které spadají do vybraných taxonů a zároveň obsahují mezi sebou nerozlišitelné druhy.

Ve spodní části stránky detailního přehledu primerového páru, je možnost porovnání několika různých párů na několika různých taxonech. Uživatel si pomocí formuláře může zvolit konkrétní páry a konkrétní taxony, pro které se následně vyzualizuje speciální mapa zachycující specificitu a senzitivitu zvolených párů na zvolených taxonech. Konkrétní podoba takové mapy je zachycena na obrázku 4.8.

Poslední vizualizací, která se na stránce nachází, je vizualizace pozice amplifikovaného regionu. Tato vizualizace je umístěna na samém vrcholu stránky a lze z ní rychlým pohledem zjistit, které varibilní regiony jsou zacíleny a které nikoliv. Konkrétní podoba této vizualizace je zobrazena obrázku 4.9. Kompletní snímek stránky s detailem primerového páru je k nalezení v příloze B na obrázku B.3.

Pro účely jednoduchého procházení a filtrování databáze párů, je pro uživatele připravena stránka s **prohlížečem primerových párů**. Tato stránka poskytuje základní možnosti filtrace databáze. Konkrétně je možné filtrovat konkrétní primery, případně filtrovat podle regulárních výrazů primerových sekvencí. Pro přehled a jako další možnost filtrace je na této stránce bodový graf závislosti specificity na senzitivitě s obdobnými schopnostmi filtrace pomocí výběru plochy, jako v případě stránky s inferencí. Stránka je zakončena přehledovou tabulkou všech, respektive všech vyfiltrovaných, primerových párů v databázi.



Obrázek 4.10: Snímek seznamu uživatelem sledovaných úloh. Uživatel si kromě stavu úlohy, může zobrazit stručný záznam z běhu úlohy a případně přejít na stránku s detailními informacemi o daném primerovém páru.

Kompletní snímek stránky s prohlížečem databáze primerových párů je k nalezení v příloze B na obrázku B.4.

Velmi důležitou funkcí aplikace je **analýza uživatelského primerového páru**, jejíž algoritmus je podrobně popsán v sekci 3.4. Uživatel může pomocí formuláře, kde vyplní jména a sekvence obou primerů spustit výpočet analýzy primerového páru. Jako doplňující informaci může zadat emailovou adresu, na kterou mu budou zasílány notifikace o stavu výpočtu.

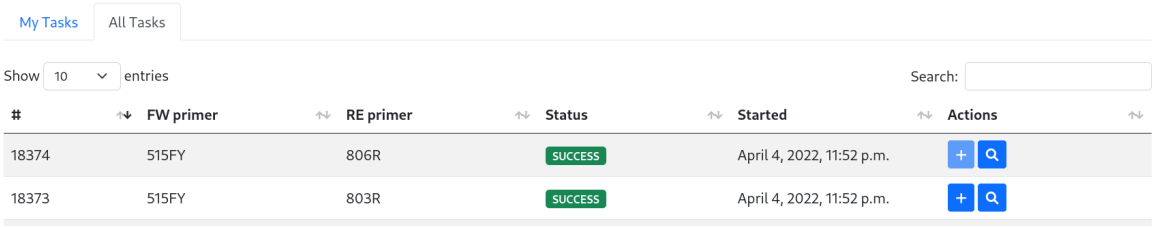
Aplikace neumožňuje registraci ani přihlašování uživatelů, s výjimkou správcovského uživatele, který má přístup k Django administraci. Z toho důvodu je veškerá identifikace uživatelů realizována pomocí mechanizmu uživatelské relace, která se udržuje v rámci cookie hostitelského prohlížeče.

Mechanizmus a implementace správy relací a s tím související správy uživatelských cookie je plně integrována v použitém frameworku Django. Jednou z hlavních otázek práce s relacemi, je způsob uchování souvisejících dat. Příkladem může být právě seznam úloh každého uživatele, který je nutné na serveru uchovat a umožnit jeho případnou manipulaci. Django implementuje několik různých řešení uchování těchto dat [11], přičemž tři hlavní jsou:.

Uložení dat do dočasných souborů na straně souborového systému serveru. Jedná se o poměrně triviální možnost, která nevyžaduje žádnou velkou konfiguraci, snad kromě nastavení práv k cílové složce. Zároveň toto řešení, ale zahrnuje jednu velkou nevýhodu, a to že taková paměť je exkluzivní pouze pro jednu repliku daného serveru. Což v případě provozu několika replik serveru v systému Kubernetes, které se mohou vzájemně střídat a vyvažovat vzájemnou zátěž, je nevhodné.

Druhou možností je uchování všech dat na straně uživatele protřednictvím samotných cookies. Integrita takových cookies, je na straně serveru udržována pomocí kryptografického podpisu za využití neveřejného klíče, který je v nastavení serveru. Přestože se jedná o poměrně pohodlné řešení, v celkovém pohledu naneštěstí zvyšuje bezpečnostní rizika, která by byla spojena s případným únikem zmíněného neveřejného klíče.

Třetí a zároveň v aplikaci použitý přístup, je udržování dat v databázi aplikace. I toto řešení je v rámci frameworku připraveno ke snadnému použití. Velká výhoda tohoto řešení je, že udržuje jednotnost dat napříč několika replikami serveru. Nevýhodou ukládání v databázi, je postupný nárůst a zastarávání dat v databázi. Data starých a nepoužívaných relací se naneštěstí automaticky nemažou a postupně se přidávají nová. Z tohoto důvodu je v případě



Obrázek 4.11: Snímek seznamu všech úloh, které v aplikaci existují. Uživatel si může jednotlivé úlohy přidat do sledovaného seznamu.

velkého nárůstu množství dat, nutné manuálně iniciovat smazání starých dat. Množství dat se v aktuální verzi aplikace drží pouze na několika málo hodnotách na jednoho uživatele a to pouze u uživatelů, kteří využívají výpočetní funkci aplikace, nikoliv u uživatelů, kteří pouze používají inferenci, a proto by tato vlastnost neměla působit větší problémy.

Jakmile tedy uživatel spustí výpočet analýzy primerového páru, úloha se zařadí do databáze, kde se udržuje její stav a její metadata. Zároveň, je identifikační číslo úlohy přiřazeno k danému uživateli, právě na základě identifikátoru relace daného uživatele. Konkrétně se identifikační číslo zařadí do seznamu úloh, které jsou v dané relaci sledovány.

Uživatel si následně **sledované úlohy**, může prohlédnout na **stránce úloh**. Seznam všech úloh je veřejný pro všechny uživatele a každý si může prohlédnout aktuální stav běžících, čekajících a již dokončených úloh. Snímek tohoto seznamu je k nalezení na obrázku 4.11. Jednotlivé úlohy si lze přidávat do sledovaných úloh, případně je ze seznamu sledovaných odebírat. Snímek a tedy i podoba seznamu sledovaných položek je zachycena na obrázku 4.10.

Jedním z důvodů otevřenosti kompletního seznamu úloh je i možnost prohlížet si primerové páry, které zatím nejsou zařazeny v databázi. Přestože by se mohlo zdát ideální, rovnou zařadit nově vypočítaný primerový pár do celkového seznamu primerových párů a používat ho při inferenci. Při řešení této práce padlo rozhodnutí toto automatické zařazování nečinit. Důvodem je především garance správnosti a použitelnosti jednotlivých primerových párů. Při automatické integraci vypočítaných párů by mohlo docházet k situacím, kdy uživatel zadá pár, který by mohl být například obtížně syntetizovatelný, nebo by jednotlivé primery mohly mít tak rozdílnou teplotu tavení, že by jejich společné využití nebylo možné. Z toho důvodu jsou uživatelské páry v databázi označeny specifickým příznakem, aby při inferenci nebyly využity.

Kompletní podoby stránek s oběma seznamy úloh jsou k nalezení v příloze B na obrázcích B.5 a B.6.

Poslední výraznější stránkou aplikace je **krátká dokumentace**. Zde jsou stručně uvedeny a vysvětleny informace k procesu inference a především je zde uvedeno množství příkladů toho, jak využít aplikační rozhraní REST k realizaci inference mimo uživatelské rozhraní, přímo s pomocí jazyka Python.

Celkový ráz webové aplikace je silně ovlivněn použitou barevnou paletou a vizuální stylizací. Celá aplikace je stylizována do světlého motivu doplněného minimalistickými ohraničeními jednotlivých segmentů stránky, jako jsou grafy, tabulky nebo formuláře. Nejvýraznějšími vizuálními a funkčními prvky jsou tlačítka. Ta jsou díky Bootstrap stylizaci poměrně dost barevně saturovaná a díky tomu těžko přehlédnutelná.

Obrázek 4.12: Logo aplikace.

Barevná paleta stránky se pak skládá ze 3 barev: ocelová modř zlatá a šedá.

Vizuálně výrazná je především hlavní strana aplikace, která by měla sloužit jako dveře pro uživatele, ale i jako charakteristický a na první pohled rozpoznatelný bod. Tomuto účelu by měl napomáhat zvolený obrázek na panelu hlavní stránky. Tento obrázek je jakousi abstraktní ilustrací výsledku sekvenování pomocí elektroforézy.

Posledním prvkem, který identifikuje a tvoří značku aplikace, je její jméno, potažmo logo. Aktuální pracovní jméno aplikace je PrimerIO. Jméno vychází z hlavní domény aplikace a to prací s primerovými páry a zároveň možností aktivního používání a spouštění úloh přímo v rámci aplikace.

Logo aplikace je navrženo tak, aby vizuálně zapadalo do použité barevné palety a minimalistického vizuálu. Logo je zobrazeno na obrázku 4.12. Motiv dvoušroubovice koresponduje s tématikou aplikace. Výhodou tohoto návrhu loga je jeho poměrně vysoká univerzálnost, co se týče pojmenování aplikace, protože ve stejném pojetí si lze představit téměř libovolné pojmenování.

4.4 Realizace analýzy primerových párů

Na počátku řešení práce jsem dostal k dispozici implementovaný nástroj, schopný vypočítat kompletní informace o primerovém páru, které jsou rozebrány v sekci o datovém modelu 3.3. Jednalo se nástroj napsaný ve funkcionálním jazyce Clojure, jehož vstupem je primerový pár, vyextrahované 16S rRNA sekvence a taxonomická hierarchie a výstupem jsou vypočítané orientované grafy nerozlišitelností a informace potřebné pro výpočet senzitivity a specificity. Pro získání amplifikovaných regionů z extrahovaných 16S rRNA sekvencí využívá V-ripper skript z projektu SPINGO [37]. Výpočet jednoho primerového páru pomocí tohoto nástroje trvá, v závislosti na konkrétním páru, řádově hodiny (až 8 hodin).

Z důvodu optimalizace, zlepšení přehlednosti kódu a lepší integrace do vyvíjené aplikace jsem se rozhodl tento nástroj nepoužít. Po prozkoumání celkové funkcionality a principu výpočtu, jsem navrhl řadu změn, optimalizací a vylepšení výpočetního algoritmu, jehož finální podoba je podrobně popsána v sekci 3.4.

Funkcionalita nástroje je tedy taková: Na základě primerového páru a vyextrahovaných 16S rRNA sekvencí organismů, je nutné vypočítat, které druhy budou daným párem amplifikovány, a které variabilní regiony budou daným párem zacíleny. Dále je nutné určit nerozlišitelnosti a vytvořit jednotlivé orientované grafy nerozlišitelností.

Druhotnými funkcemi jsou funkce na inicializaci databáze, jako výpočet množství dceřiných taxonů každého jednotlivého taxonu a konstrukce taxonomického stromu v databázi. Tyto funkce jsou, ale pouze jednorázové a použijí se pouze při počáteční inicializaci databáze.

Stejně jako původní Clojure skript i já využívám skript V-ripper z projektu SPINGO. V-ripper dokáže na základě přímé a reverzní primerové sekvence vypočítat, jaký (případně žádný) region 16S rRNA genu se v dané sekvenci amplifikuje.

Vyextrahované 16S rRNA sekvence jsem dostal při řešení projektu. Celkový počet vyextrahovaných sekvencí je 88646. Přičemž celkový počet strainů je 39628. Sekvencí je více než strainů, protože každý jednotlivý strain může obsahovat několik různých variant 16S rRNA sekvence.

Srovnání výkonosti původní a nové implementace. Výkonnostní aspekt byl jedním z hlavních priorit reimplementace celého výpočtu primerového páru. Původní implementace v závislosti na konkrétním primerovém páru běžela řádově hodiny. Velkou časovou zátěží výpočtu byla agregace grafů, ale především následné ukládání dat do výstupního souboru. Právě na tyto dvě části jsem se v nové implementaci zaměřil a optimalizoval je.

Optimalizaci agregace grafů jsem popsal v sekci 3.4 a je jí docíleno především díky použití mechanizmu práce nad datovým rámcem a použitím knihovny Pandas.

Výstup původní implementace se ukládal ve formě databáze, buď do běžící MySQL instance, nebo do sqlite souboru. Pro potřeby integrace do aplikace byl právě výstup do souboru sqlite databáze vhodnější. Po bližším zkoumání, jsem ale přišel na to, že data jsou do databáze ukládána po jednotlivých řádcích. Což se v případě primerového páru, jehož jeden agregovaný graf může obsahovat i 100 tisíc hran, ukázalo jako značně neefektivní.

Aktuální implementace umožňuje ukládat data ve formátu JSON souboru. Navíc díky skutečnosti, že je nová implementace napsaná v jazyce Python, bylo možné celou implementaci zařadit do aplikace jako samostatný modul a jakýkoliv výstup do souboru tedy není nutný. Výsledná data analýzy se vkládají do databáze přímo v rámci kódu aplikace po velkých blocích, obsahujících 10000 řádků.

Nezanedbatelným aspektem programu, je také jeho spotřeba paměti. V dnešní době už je sice spotřeba paměti podstatně menší problém než v minulosti. Přesto jedná-li se o aplikaci běžící v clusteru, kde každé další procesorové jádro a každý další gigabyte paměti může znamenat vyšší finanční náklady, je vhodné i spotřebu paměti minimalizovat.

Původní implementace v závislosti na primerovém párů alokovala řádově jednotky gigabytů paměti (cca do 10 GB). Což na první pohled nemusí působit jako nadstandardně vysoká hodnota. Komplikace nastává až v situaci, kdy je potřeba provozovat jednotky, případně desítky, výpočetních uzlů. V takovém případě, kdy jediný uzel alokuje 10GB paměti, se celková alokovaná paměť dostává do desítek až stovek gigabajtů.

Nová implementace byla optimalizovaná i v oblasti spotřeby paměti. Nízká paměťová náročnost přímo vychází z filozofie optimalizované verze algoritmu. Výsledkem je, že spotřeba paměti se u naprosté většiny primerových párů pohybuje do 500 MB.

Při testování a vyhodnocení nové implementace, jsem provedl nespočet experimentálních výpočtů s různými primerovými páry a tedy i s různým množstevním záchytem amplifikace a různým množstvím nerozlišitelností. Vybrané příklady jsem shrnul do tabulky 4.2. V tabulce je uvedeno konkrétní srovnání časové a pamětové náročnosti výpočtu původní a nové implementace analýzy primerového páru. Oproti původní implementaci je vidět zřejmé zlepšení doby potřebné pro výpočet i celkové snížení spotřeby paměti.

	Původní / Nová implementace	
Primerový pár	Čas výpočtu	Potřebná paměť
530F - 1391R	4h47m/2m47s	6.2GB/403MB
P609D - 1391R	5h4m/2m24s	5.4GB/402MB
1099F - 1391R	6h7m/2m38s	4.8GB/387MB
515F - 806R	5h34m/2m1s	5.7GB/402MB

Tabulka 4.2: Srovnání výkonnosti původní a nové implementace analýzy primerového páru. Měření času bylo provedeno pouze jednou pro každý pár, protože rozdíl je tak významný, že na jednotkách sekund, nebo minut, nezáleží.

Implementace výpočetních uzlů

Protože se, i po optimalizaci, doba výpočtu pohybuje v řádu jednotek minut, je potřeba v rámci aplikace udržovat množinu výpočetních uzlů, na nichž bude výpočet realizován. Nutnost existence výpočetních uzlů, respektive nemožnost využití výpočetní síly webového serveru, vychází z filozofie funkce webového serveru. Webový server aplikace je sice technicky připraven na to aby spustil příkladně 5 minut dlouhou úlohu, v rámci svého vlastního procesu. Obecně se ale nejedná o správný přístup. V případě, kdy by uživatel spustil takovou dlouhotrvající úlohu, dostal by odpověď od serveru až po jejím dokončení a po celou dobu výpočtu by daná replika webového serveru byla nedostupná.

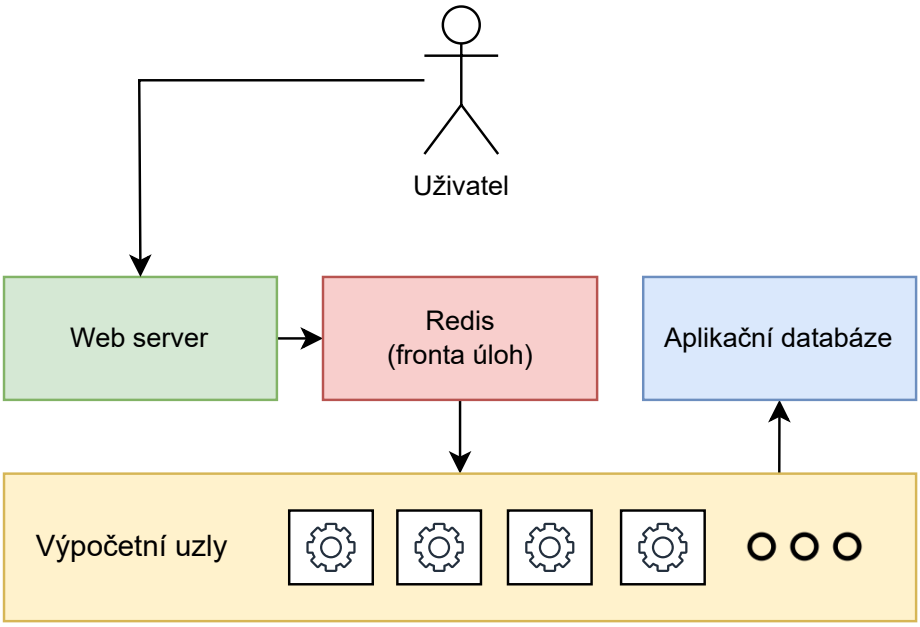
Triviální, ale ani zdaleka ideální řešení, by bylo spouštění nových výpočetních procesů pro výpočet na stejném stroji, kde je provozován webový server. Uživatel by v takovém případě poslal požadavek na server, server by vytvořil nový proces, který by začal danou úlohu zpracovávat samostatně a server by vzápětí vrátil uživateli zprávu o úspěšném spuštění úlohy. Přestože by toto řešení do určité míry fungovalo, obsahuje hned několik nedostatků. Hlavním nedostatkem je fakt, že výpočet úlohy je spuštěn na stejném uzlu jako webový server. To může potenciálně vést k situaci, kdy výpočetní uzel nebude mít dostatečnou výpočetní sílu na výpočet úlohy a provoz serveru zároveň, což povede ke zpomalení nejen výpočtu, ale i ke snížení responzivity webového serveru a tedy i celé aplikace. Tento problém se ještě zvýrazní, pokud v jednu chvíli bude spouštět úlohu více uživatelů najednou. Ideálně by výpočetní úlohy měly být spuštěny na uzlech k tomu dedikovaných a jejich počet, na jednotlivých uzlech, by měl být řízen a korigován.

Proto se pro tyto účely standardně využívá fronta úloh a množina dedikovaných výpočetních uzlů, které úlohy z fronty zpracovávají.

S účelem zjednodušení implementace, respektive dnes už spíše pouhé integrace, mechanizmu fronty úloh do vlastní implementace, vzniklo hned několik opensource projektů. Pro jazyk Python lze jmenovat projekty: Redis queue (RQ) [29], Django Q [12] a především velmi rozšířené Celery [5].

Do aplikace jsem se rozhodl integrovat právě knihovnu Celery, především z důvodů stability, velikosti komunity, aktuálnosti a živosti projektu.

Princip činnosti knihovny Celery se zakládá na několika komponentách. Schéma jednotlivých komponent, které celý mechanizmus využívá, je zobrazen na obrázku 4.13.



Obrázek 4.13: Schéma jednotlivých komponent systému, které zabezpečují běh výpočetních úloh. Na počátku celého procesu je uživatel iniciující výpočet přes požadavek na webový server. Dále je úloha zařazena do fronty, následně zpracována výpočetním uzlem a výsledky výpočtu jsou nakonec uloženy v aplikační databázi.

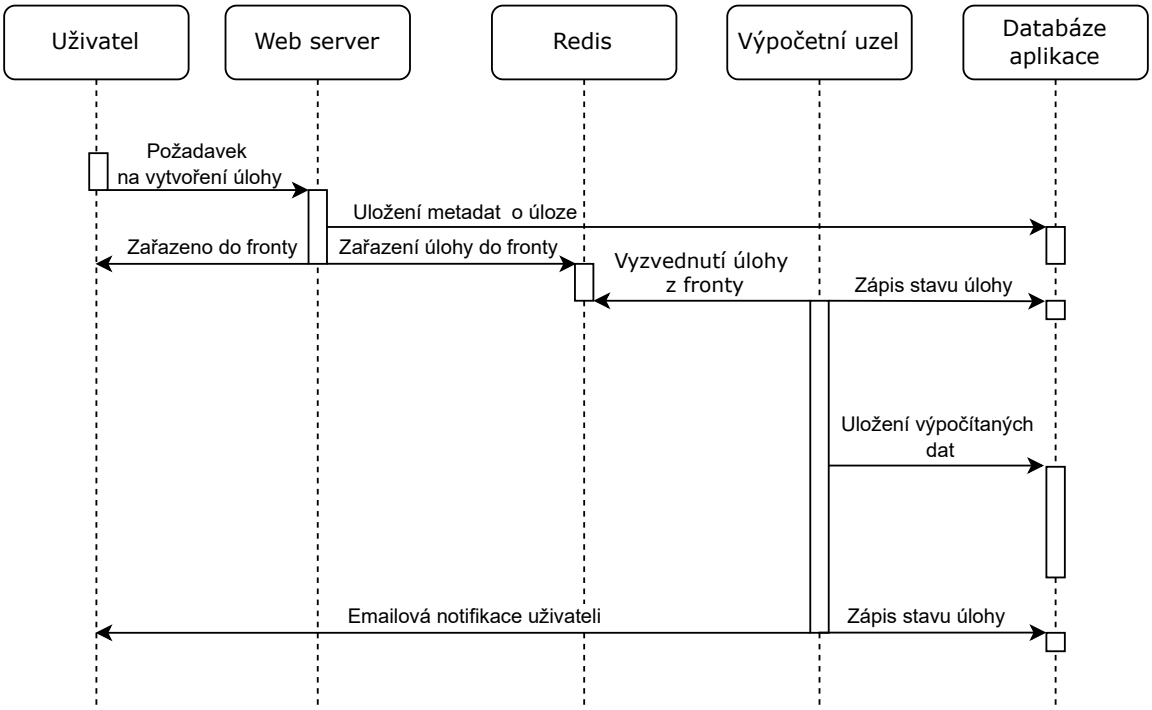
Na začátku celého mechanizmu stojí webový server, respektive uživatel. Uživatel vyšle požadavek na webový server, ten ho zpracuje a na jeho základě zařadí výpočetní úlohu do fronty.

Fronta úloh je uložena v rámci samostatně běžícího databázového serveru Redis. Redis zde slouží především jako prvek, který udržuje frontu úloh a realizuje rozesílání zpráv mezi jednotlivými komponentami celého mechanizmu. Celery umožňuje v rámci jednoho Redis serveru udržovat téměř libovolné množství různě pojmenovaných front, které můžou být obsluhovány různými množinami uzlů a plnit tedy různě prioritní úlohy. V rámci aplikace je využita pouze jedna fronta určená právě pro úlohy analýzy primerových párů.

Poslední hlavní komponentou mechanizmu jsou výpočetní uzly. Ve své podstatě se jedná o procesy, které buď čekají na úlohu, anebo úlohu zpracovávají. Výpočetní uzly v rámci svého běhu komunikují s databází aplikace, do níž ukládají výsledky analýzy. Výpočetních uzlů je možné provozovat téměř neomezené množství. Respektive takové množství, jaké zvládnou ostatní prvky aplikace, především právě Redis a samotná databáze aplikace.

Sekvenční diagram průběhu vytvoření úlohy jejího zpracování a příslušné komunikace mezi jednotlivými komponentami je vyobrazen na obrázku 4.14.

Pokud by bylo nutné provozovat velké množství výpočetních uzlů (stovky až tisíce), bylo by z velkou pravděpodobností nutné řešit škálovatelnost aplikační databáze, která je v aktuální době nejužším hrdlem celého systému. Ovšem pravděpodobnost potřeby, tak velkého množství uzlů je zcela mizivá, protože by se jednalo o situaci, kdy by v aplikaci byly spouštěny tisíce úloh každou hodinu. Což rozhodně není očekávaný scénář vytížení aplikace.



Obrázek 4.14: Sekvenční diagram zachycující sekvenci jednotlivých úkonů při analýze jednoho primerového páru.

Očekávaný scénář využívání analytické funkce aplikace předpokládá maximálně s jednotky úloh týdně. I přes tento poměrně malý předpoklad je aplikace v aktuální situaci i na HDD instanci schopna obsluhovat nízké stovky úloh za hodinu.

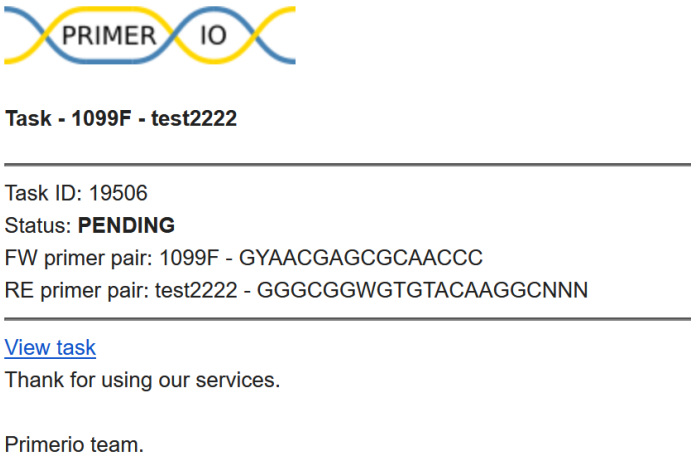
Pro ilustraci výkonu. Celá aktuální databáze, obsahující 5661 primerových párů, byla s využitím 20 výpočetních uzlů, vytvořena během necelých dvou dnů přímo pomocí výpočetní funkce aplikace. Databáze, která byla při tomto vytváření použita, běžela na HDD instanci rozebrané podrobněji v sekci 4.2. Tato skutečnost ukazuje připravenost aplikace na řádově tisíce úloh denně.

Pokud uživatel zadal svoji emailovou adresu, pak je o celém průběhu výpočtu informován prostřednictvím emailových notifikací. V okamžiku, kdy je úloha zadána do systému, uživatel dostane potvrzující notifikaci, ve které je uveden odkaz pro sledování úlohy prostřednictvím uživatelského rozhraní. Další notifikace jsou následně zasílány při spuštění úlohy a při jejím úspěšném či neúspěšném dokončení.

Podoba emailové notifikace je zachycena na obrázku 4.15.

4.5 Aplikační rozhraní REST

Aplikační rozhraní umožňuje standardizovanou komunikaci mezi serverovou částí aplikace a klientem. Klientem může být uživatelské rozhraní aplikace, případně uživatelský skript.



Obrázek 4.15: Obsah emailové notifikace o stavu zadané úlohy.

V dnešní době existuje hned několik možností, standardů a architektur, aplikačních rozhraní, které lze volně použít. Namátkou lze jmenovat poměrně populární GraphQL [17], nebo velmi populární rozhraní REST.

Právě architektura rozhraní REST je do aplikace integrována.

REST [48, 58], neboli Representational State Transfer, je architektura rozhraní, popsaná v roce 2000 Royem Fieldingem . Jedná se o bezstavové, datově orientované rozhraní, které využívá k přenosu doménově specifických dat primárně (je možné využít REST architekturu i bez HTTP) protokol HTTP. Popisuje definici a adresaci dat, neboli zdrojů a definuje čtyři základní metody pro přístup k nim.

Jedná se tedy o architekturu klient a server, přičemž není udržován stav komunikace. Z toho plyne, že každý jeden požadavek na rozhraní REST, popisuje kompletně požadavek na server, protože bezstavovost architektury neumožňuje vyslání jakéhokoliv přímo navazujícího požadavku.

Bezstavovost rozhraní, umožňuje velmi snadné nasazení v distribuovaném prostředí, protože veškerý kontext, je vždy obsažen přímo v rámci jednoho požadavku na server a tedy střídání replik serverů nečiní komunikaci přes REST žádný problém.

Definovány jsou čtyři základní metody pro přístup k datům.

1. **GET** - čtení existujících dat.
2. **PUT** - Zápis a modifikace
3. **POST** - Vytvoření nové položky
4. **DELETE** - Vymazání záznamu

Adresace dat je realizována pomocí strukturované url adresy. Obecně lze adresovat konkrétní položky, případně celé datové kolekce.

Příkladem adresace datové kolekce, v rámci GET dotazu, obsahující primerové páry, může být:

```
{
    "count": 5624,
    "next": "http://primerio.eu/api/primer_pair/?page=2",
    "previous": null,
    "results": [
        {
            "id": 33859,
            "name": "bact363F - Uni1392R",
            "fw_primer": 1179,
            "re_primer": 1272,
            "sensitivity": 91.7,
            "specificity": 54.3
        },
        ...
    ]
}
```


Obrázek 4.16: Příklad dat vrácených GET dotazem na kolekci primerových párů prostřednictvím rozhraní REST. Výsledkem je datová struktura, která představuje jednu stránku položek kolekce. Získaná struktura obsahuje metadata o celkovém množství dat v kolekci, adresu odkazu na předchozí a následující stránku, pokud existuje a samozřejmě pole dotázaných dat.

`GET http://primerio.eu/api/primer_pair/`

Výsledkem takového dotazu by byla kolekce všech uložených primerových párů. Doplňujícím mechanizmem, který se využívá při zacházení s kolekcemi je stránkování výsledků, po definovaném počtu výsledků na jednu stránku.

Jako formát pro přenos dat přes rozhraní REST je velmi často používán formát JSON. Data získaná pomocí zmíněného GET dotazu, by tedy mohla mít strukturu jako na obrázku 4.16.

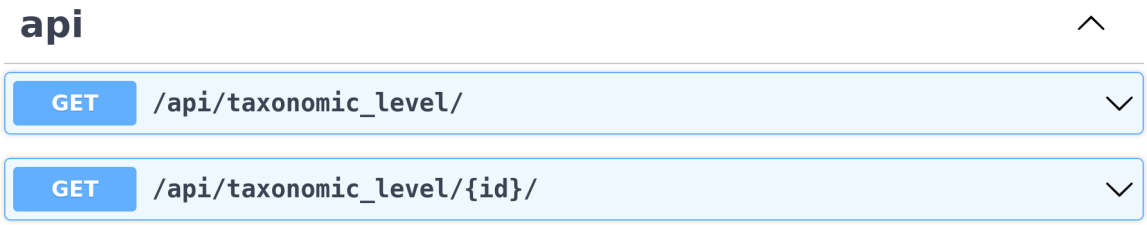
V případě potřeby filtrace dotazované kolekce, je dobrým zvykem parametry filtrace umístit do proměnných url dotazu: `GET http://primerio.eu/api/primer_pair/?id=33859`.

Adresace jedné konkrétní datové položky je standardně provedena pomocí rozšíření adresy o identifikační číslo dané položky: `GET http://primerio.eu/api/primer_pair/33859`

V rámci implementované aplikace je rozhraní REST využito, především jako nástroj implementující prvky využité uživatelským rozhraním. Z uživatelského rozhraní jsou prostřednictvím REST posílány dotazy realizující inferenci, interaktivní našeptávání taxonů a primerů a v neposlední řadě jsou jeho prostřednictvím získávány data o primerových párech v prohlížeci primerových párů.

Druhotnou funkcí implementovaného rozhraní, je možnost přímého využití uživatelem. V případech, kdy by uživatel chtěl určitá data aplikace zpracovávat prostřednictvím svých vlastních skriptů, může využít aplikační rozhraní a přistupovat k datům například pomocí jazyka Python.

Právě příklady v jazyce Python jsou uvedeny v aplikaci na stránce s dokumentací. Konkrétně jsou zde uvedeny příklady, jak získat taxonomickou hierarchii, jak získat ohodnocené



Obrázek 4.17: Snímek z nástroje Swagger, který uživateli umožní si prohlédnout a vyzkoušet funkcionalitu rozhraní REST.

primerové páry z procesu inference a v neposlední řadě je přes rozhraní vystavena databáze primerových párů, včetně informací o nerozlišitelných a senzitivovaných taxonech.

Implementace rozhraní REST je v aplikaci realizována s využitím knihovny Django REST framework (DRF) [9]. Ta implementuje integraci REST komunikace přímo do frameworku Django a usnadňuje propojení rozhraní s datovým modelem aplikace. Dále knihovna zajišťuje správnou serializaci dat a zpracování uživatelských dotazů. Dnes je právě tato knihovna určitým standardem pro implementaci REST rozhraní s frameworkem Django.

Knihovna DRF umožňuje vygenerování schématu kompletní funkcionality rozhraní ve formátu OpenAPI. Jedná se o standardizovaný formát, který popisuje funkcionalitu RESTful rozhraní. Výhodou tohoto formátu je právě jeho standardní forma, díky níž vznikly nástroje, které dokáží, uživatelsky přívětivě, zobrazit jednotlivé funkce rozhraní.

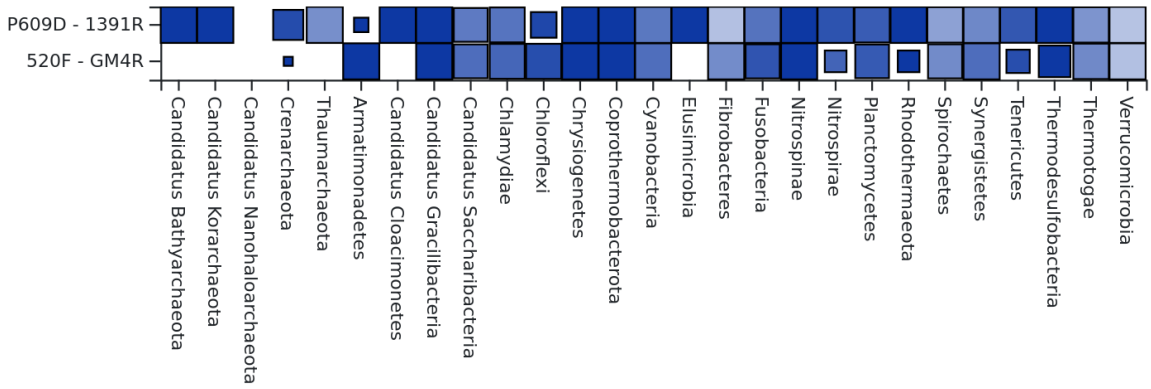
Jedním takovým nástrojem je Swagger [31]. Jedná se o nástroj, který dokáže zpracovat soubor v OpenAPI formátu a na jeho základě zobrazit uživateli možnosti a funkcionalitu, kterou dané rozhraní poskytuje. Zároveň uživateli umožňuje si jednotlivé funkce přímo v prohlížeči vyzkoušet a prohlédnout výsledky daného dotazu. Konkrétní podoba nástroje Swagger, je zobrazena na obrázku 4.17.

V kontextu aplikace, je z pohledu uživatele samotné rozhraní spíše doplněk funkcionality, nikoliv kompletní přenesení funkcí aplikace.

4.6 Možnosti budoucího rozšíření

První možnost vylepšení aplikace, byla nastíněna už v kapitole 3.3. Konkrétně se jedná o volbu parametru rozlišení, na kterém chce uživatel vyhodnocovat primerové páry. Aktuální verze aplikace, respektive uživatelské rozhraní aplikace, dokáže pracovat pouze na úrovni rozlišení druhu. Tedy veškeré ohodnocení senzitivity a specificity primerových párů je vypočítáváno na této úrovni bez možnosti volby. Hlavní část implementace tohoto rozšíření by byla na úrovni uživatelského rozhraní, protože databáze a algoritmus analyzující primerové páry už s funkcionalitou rozlišení počítají a jsou na ni připraveni.

Druhé možné rozšíření aplikace se týká rozšíření algoritmu inference. Aktuální princip inference, spočívá v práci s každým primerovým párem samostatně. Konkrétně se počítá s tím, že cílem je vybrat jeden jediný primerový pár. K tomu je uzpůsobeno i uživatelské rozhraní, které je primárně zaměřeno na vizualizaci primerových párů, jako samostatných jednotek.



Obrázek 4.18: Dvojice primerových párů, kde primerový pár P609D - 1391R neumožňuje záchyt většiny bakterií z kmenu Armatimonadetes, ale při kombinaci s primerovým páre 520F - GM4R, by ve výsledku daný kmen byl zachycen plně.

Na celkovou inferenci se lze podívat ze širšího pohledu a uzpůsobit algoritmus k vyhledávání skupin primerových párů, které je možné využít společně a tím docílit maximální senzitivity, za zachovaní maximální specificity.

Výsledkem inference by v takovém případě nebyly ohodnocené primerové páry samostatně. Naopak, výsledkem by byly ohodnocené n-tice primerových párů, které by dohromady umožňovaly maximální pokrytí zvoleného, ale i celého, taxonomického spektra.

Triviální ilustrační příklad z databáze je zobrazen na obrázku 4.18. Na obrázku je dvojice primerových párů, kde jeden dokáže zachytit pouze minimum druhů bakteriálního kmene Armatimonadetes. Pokud by se použil v kombinaci druhým primerovým párem, který umožňuje záchyt celého kmene Armatimonadetes, mohl by celkový výsledek sekvenace poskytnout přesnější data k následné klasifikaci.

Mimo uvedené větší možnosti rozšíření, lze aplikaci rozvíjet i jen malými inkrementálními vylepšeními. Z nichž některá jsou:

- Provázání taxonomické hierarchie v aplikaci s NCBI hierarchií prostřednictvím NCBI taxonomického identifikačního čísla.
- Implementace algoritmu pro výpočet teploty tavení primerů.
- Provázaní primerových sekvencí s databází SILVA, případně RDP. Umožnit uživatelům jednoduše prozkoumat primery pomocí těchto nástrojů.
- Doplnění dalších informací o primerech. Například: Uvedení případných studií z nichž pochází.

Kvalita aplikace přímo souvisí s kvalitou dat, se kterými pracuje. V sekci 3.3 bylo nastíněno, že data v databázi, s novými objevenými druhy a novými variantními sekvencemi, postupem času zastarávají, respektive jsou objevovány nové. Z toho důvodu je jasné, že pokud by aplikace měla být používána v řádu mnoha let, budou muset být data aktualizována, aby odpovídala aktuálním poznatkům. Proto by budoucí motivací měla být tvorba, minimálně semi-automatizovaného procesu pro získání aktuální taxonomické hierarchie a genových sekvencí, z nichž by mohla být databáze v pravidelných intervalech (např. jednou ročně)

aktualizována. Díky tomu by kvalita aplikace a její smysl, byl udržitelný po dobu mnoha let.

V neposlední řadě, by se mohl rozšířit celý obor aplikace. Aktuálně je aplikace zaměřena na 16S rRNA amplikonové sekvenování baterií a archeí. Nicméně, aplikace by se mohla rozšířit o podporu 18S rRNA sekvenování a fylogenetickou analýzu určitých eukariotických organismů, případně o primery 28S rRNA používané ve fylogenetice hub.

Mimo zmíněné možnosti, lze bezpochyby nalézt řadu dalších potenciálních vylepšení od funkcionality uživatelského rozhraní, přes logiku práce s primerovými páry, až po zcela novou zatím nepokrytou funkcionalitu.

Kapitola 5

Produkční nasazení aplikace

V kapitole 4 jsou podrobně rozepsány jednotlivé komponenty systému, které tvoří celou aplikaci. Ve zkratce jde o webový server, PostgreSQL a Redis databáze a výpočetní uzly. Všechny tyto komponenty je nutné někde provozovat. Z důvodů popsaných v sekci 4.1, jsem zvolil nasazení většiny částí aplikace v distribuovaném prostředí typu Kubernetes.

Kubernetes [21] je open-source systém určený pro automatizovanou správu kontejnerizovaných aplikací. Poskytuje funkcionalitu horizontálního škálování, standardizaci produkčního nasazení aplikací a mimo jiné zjednodušuje provoz vysoce dostupných aplikací.

Samotný koncept systému Kuberenetes se skládá z několika prvků, které dohromady umožňují běh aplikací, jejích síťovou komunikaci a správu úložiště. Mezi hlavní prvky systému mimo jiné patří:

- Výpočetní uzly: Konkrétní stroje, které jsou zapojeny do Kubernetes clusteru a na kterých jsou provozovány jednotlivé kontejnery.
- Pody: Konkrétní kontejnerové instance, které běží na jednotlivých uzlech a poskytují služby. Specifikace homogenní množiny podů, je standardně popsána tzv. *deploymentem*.
- Služby: Množiny podů, které společně obsluhují definovanou síťovou službu. (Např.: webový server)

Dalšími prvky systému, které umožňují běh, automatizovanou správu a řízení aplikací jsou mimo jiné: kontroléry replik, správa úložiště, sady konfigurace a další.

Na první pohled je z množství jednotlivých komponent, zřejmá celková robustnost a komplexnost systému Kubernetes, která ale umožňuje maximálně univerzální použití pro provoz rozličných aplikací.

Alternativou k systému Kubernetes je například systém OpenShift. Přičemž aplikace, která je připravená pro běh v Kubernetes, bude zcela jistě schopna provozu i v systému OpenShift, protože rozdíl funkcionality obou systémů je minimální. Konfigurační soubory obou systémů jsou dokonce natolik podobné, že pro migraci z jednoho systému do druhého, je potřeba zcela minimálních změn.

```
FROM python:3.10 # Rodičovský obraz, z~něhož se odvozuje.
RUN pip install pipenv # Doinstatalce potřebných balíčků.
COPY primerio /home/primerio # Vložení kódu aplikace.
COPY Pipfile /home/
RUN cd /home/ && pipenv lock --requirements > requirements.txt
RUN pip install -r /home/requirements.txt # Instalace Python závislostí.
EXPOSE 8080 # Vystavení portu serveru
RUN chmod +x /home/primerio/run.sh
ENV RELEASE=True # Nastavení proměnných prostředí.
ENV REDIS_HOST=127.0.0.1
WORKDIR /home/primerio # Nastavení výchozí pracovní složky.
CMD ["./run.sh"] # Definice skriptu, který se spustí při výchozím spuštění
    kontejneru.
```

Obrázek 5.1: Docker šablona popisující tvorbu obrazu kontejneru webového serveru aplikace.

5.1 Postup přípravy nasazení aplikace v systému Kubernetes

Poměrně vysoká univerzálnost systému Kubernetes s sebou nese i určitou míru komplexity přípravy aplikace pro úspěšné nasazení. Z toho důvodu je nejdříve nutné posoudit, jestli se vůbec vyplatí danou aplikaci tímto způsobem provozovat a jestli to nepřinese více nevýhod než výhod.

Obecně lze říci, že v případě webových aplikací je nasazení v Kubernetes výhodné. V případě této práce je to velmi výhodné i z důvodu provozu výpočetních uzlů.

Prvním krokem přípravy aplikace je její **kontejnerizace**. Standardně se využívá kontejnerizace pomocí nástroje **Docker** [13].

Docker umožňuje tvorbu kontejnerových obrazů na základě šablon, tzv. Dockerfile. Tato šablona mimo jiné popisuje, jaké soubory a programy budou v kontejneru dostupné, jaké síťové porty má kontejner přístupné a jaký program se spustí při výchozím nastartování kontejneru. Standardně se šablony kontejnerů vytváří odvozením od jiných šablon, respektive obrazů, které lze nalézt například na oficiálních Docker registrech Docker Hubu[1] .

Konkrétní postup tvorby šablony na příkladu kontejnerizace webového serveru aplikace:

Prvním krokem je volba obrazu z něhož se bude vycházet. Hlavním parametrem výběru obrazu je podpora jazyka Python. Na Docker Hubu je dostupný oficiální obraz kontejneru, který má integrovanou specifickou verzi (kterou lze zvolit) jazyka Python. Tento obraz použiji jako rodičovský obraz. Následně doinstaluji závislosti aplikace, zkopíruji do kontejneru samotný kód aplikace, definuji síťové porty, proměnné prostředí a spouštěcí skript.

Výsledná šablona je uvedena na obrázku 5.1.

Na základě vytvořené šablony je možné sestavit obraz kontejneru, který je možné spustit prakticky kdekoliv, bez jakýchkoliv problémů se závislostmi. Podobným způsobem je vytvořena i šablona pro výpočetní uzly a šablona pro hostování statických souborů aplikace.

[1] https://hub.docker.com/

Do statických souborů aplikace spadají: obrázky, javascript soubory, css soubory a případné další statické soubory, které se ve výsledné stránce načítají.

Díky docker registrům (Docker hub...) je možné jednoduše v kontejneru spustit i jiné programy, protože autoři různých projektů přímo vydávají svůj software i prostřednictvím Docker obrazů. Například databáze Redis, obsahuje plně použitelný a konfigurovatelný obraz přímo v registru. Díky tomu se instalace Redisu redukuje na pouhé spuštění kontejneru a nastavení proměnných prostředí.

Jakmile jsou všechny části aplikace , které mají být provozovány v rámci Kubernetes kontejnerizovány, lze přejít k dalšímu kroku. Tvorba šablony pro Kubernetes.

Kompletní strukturu, služby a konfiguraci aplikace, lze popsat prostřednictvím yaml souboru. V něm je popsáno jaké pody, služby, úložiště, konfigurace a další prvky systému aplikace používá. Standardně se pro každou část aplikace (skupinu podů) definuje tzv. deployment. V rámci deploymentu může být definováno: jaké pody se mají vytvořit, jakým způsobem, kolik jich má být, jaké služby poskytují, jaká potřebují úložiště, jaké proměnné prostředí se mají nastavit a další. Dále je možné definovat skripty pro kontrolu dostupnosti jednotlivých podů, s tím související politiku restartování podů při nedostupnosti a množství dalších parametrů, které jsou popsané v dokumentaci.

Vytvořená šablona ve výsledku přesně popisuje, co vše je potřeba vytvořit, spustit a propojit aby aplikace mohla fungovat.

Přes mnoho výhod sebou Kubernetes nese i určité aspekty, se kterými je nutné počítat. Kromě komplexnosti a strmé učící křivky, se jedná i o aspekty související přímo s filozofií celého systém. Například je nutné počítat s možnými tranzicemi podů z uzlu na uzel a tedy neočekávaným přerušením činnosti daného podu.

Jedna z velkých otázek při nasazování systému, byla ohledně provozu databázového serveru. Konkrétně, jestli i databázi aplikace provozovat v rámci Kubernetes, anebo ji provozovat na separátním stroji.

Aktuální stav aplikace je takový, že aplikační PostgreSQL databáze, je jedinou částí aplikace, která je provozována mimo Kubernetes. Běží na separátním virtuálním stroji. Důvodem toho je, především snazší administrace databáze a vyšší míra kontroly nad celým prostředím, kde databáze běží. Protože, jak bylo popsáno v kapitole 4, aplikační databáze je poměrně rozsáhlá (aktuálně kolem 150GB) a jsou na ni kladeny velké výkonnostní nároky.

Přesto do budoucna, by bylo vhodné provést experimenty s výkoností databáze v prostředí Kubernetes. Pokud by se výkonnostně pohybovala minimálně na stejné úrovni, jako na separátním stroji, pak by se mohla přesunout do Kubernetes permanentně. Nejvhodnějším způsobem nasazení v Kubernetesu, by pravděpodobně bylo využití tzv. operátoru. Operátor je určitým softwarovým doplňkem systému Kubernetes, který například může zjednodušovat správu a provoz distribuované databáze. Pro PostgreSQL jsou dostupné například operátory: Stolon [30] nebo Zalando Postgres Operator [33].

Do budoucna, by přesun databáze do Kubernetesu mohl zjednodušit proces horizontálního škálování databáze, který PostgreSQL sice podporuje, ale rozhodně se nejedná o triviální proces. Díky tomu, by se potenciálně mohlo zvýšit množství uživatelů, které je aplikace schopna obsloužit.

Kompletní šablona definující veškeré prvky aplikace, je součástí odevzdaných zdrojových kódů v adresáři *deploy*.

5.2 CI proces

Standardem vývoje téměř jakéhokoliv software, je využití verzovacího systému. Dnes se většinou využívá nástroje Git. Ten je podporován řadou online služeb, které umožňují hostování Git repozitáře na jejich serverech. Mimo samotné hostování souborů projektu, je dnes už standardem poskytování CI (Continuos integration) služeb.

CI je proces automatizace začleňovaní změn do nových verzí aplikace. CI služby umožňují mimo jiné automatizaci kompilace softwaru, následné automatické spuštění testů a sestavení softwarového balíčku, nebo jako v případě této aplikace sestavení Docker obrazu.

Pro hostování repozitáře aplikace, jsem se rozhodl využít službu GitLab [16]. GitLab integruje CI podporu přímo, bez nutnosti používat jakoukoliv službu třetí strany.

Celý CI proces aplikace sestává z několika kroků. Jakmile jsou provedeny změny v hlavní větvi repozitáře, v rámci GitLab CI se automaticky spustí úlohy pro sestavení jednotlivých kontejnerových obrazů. Konkrétně se sestavuje obraz webového serveru aplikace, obraz výpočetního uzlu a obraz webového serveru se statickými soubory aplikace. Všechny tři úlohy běží paralelně. Jakmile jsou obrazy sestaveny, uloží se do registru, který je asociován k danému repozitáři. V tuto chvíli jsou obrazy připravené k nasazení v Kubernetes.

Samotné nasazení je možné také automatizovat. Já jsem se ale pro tuto možnost nerozhodl, a protože se reálně jedná o stisk jednoho tlačítka v ovládacím rozhraní clusteru, ponechal jsem nasazení nové verze manuální. Díky tomu je nasazená verze více pod kontrolou a nebezpečí toho, že dojde k nasazení verze, která k nasazení určena nebyla, je minimalizováno.

5.3 Doplňující detaily

Několikrát byla v textu zmíněna informace o statických souborech aplikace. Většinově jde o soubory, které se načítají až z klientské strany po získání HTML stránky z aplikačního serveru (Django aplikace). Jde o CSS a Javascript soubory, obrázky a další potřebné soubory. Obecně není doporučováno tento typ souborů poskytovat přímo pomocí aplikačního serveru. Protože ten není přímo stavěn na tento typ využití a tudíž by celková výkonnost aplikace mohla zbytečně utrpět.

Místo toho Django framework dokáže veškeré statické soubory aplikace vyextrahovat tak, aby mohly být hostovány na jiném standardním HTTP serveru.

To znamená, že veškeré statické soubory jsou umístěny na subdoméně. Aplikace je aktuálně provozována pod doménou *primerio.eu*. Pod touto doménou se uživatel připojí k jedné z replik aplikačního webového serveru. Statická data jsou provozována na doméně *static.primerio.eu*. Pod touto doménou se tedy uživatel připojí k serveru Apache, kde jsou hostovány veškeré statické soubory.

200	GET	cdn.jsdelivr.net	axios.min.js	script	js	v mezipaměti	0 B
200	GET	static.primerio.eu	logo.png	img	png	v mezipaměti	8,13 KB
200	GET	static.primerio.eu	main.png	img	png	v mezipaměti	76,13 KB

Obrázek 5.2: Statické soubory jsou získávány ze subdomény aplikace.

Jediné co je, pro správnou funkci aplikace, nutné nastavit, je právě zmíněná subdoména v konfiguraci Django aplikace. Díky tomu se veškerý provoz statických souborů přesune na jiný server, což je ilustrováno na obrázku 5.2. (Případně se načítají z mezipaměti prohlížeče.)

Dalším standardem dnešní doby, je provoz webové stránky prostřednictvím šifrovaného protokolu **HTTPS**. Použitý aplikační server Gunicorn podporuje pouze provoz s protokolem HTTP. Díky provozu aplikace v Kubernetes to ovšem není žádný problém. Protože Kubernetes veškerou doménovou komunikaci zastřešuje ještě jednou vrstvou webového serveru, který slouží jako proxy mezi uživatelem a aplikací. HTTPS je tedy nutné řešit právě na tomto zastřešujícím serveru. Pro zjednodušení nasazení HTTPS v clusteru vznikl projekt Cert Manager [6], který se jako rozšíření provozuje na clusteru a poskytuje pro všechny uživatele možnost správy HTTPS, mimo jiné prostřednictvím Let's Encrypt [22] certifikátů.

Díky tomu, že na clusteru Metacentra je Cert Manager dostupný, spuštění HTTPS je otázkou doplnění konfigurace doménové služby o potřebné informace.

Ve výsledku je aplikace přístupná prostřednictvím odkazu: https://primerio.eu/.

Kapitola 6

Závěr

Hlavním cílem práce byl návrh a implementace webové aplikace, která uživatelům usnadní výběr primerového páru určeného pro 16S rRNA amplikonové sekvenování a umožní maximalizovat úspěšnost následné klasifikace sekvencí.

Na počátku řešení práce, jsem dostal přístup k databázi analyzovaných primerových párů. Při tvorbě aplikace jsem z poskytnuté databáze sice vycházel, ale ve finální aplikaci jsem ji nepoužil, přesto výrazně ovlivnila celý vývoj.

Hlavní funkcionalitou aplikace je algoritmus ohodnocení a následného doporučení primerových párů, podle specifického zadání uživatele. Tento algoritmus je v aplikaci pojmenován *inference* a dokáže ohodnotit senzitivitu a specificitu primerových párů, pro uživatelem vybrané taxonomické skupiny. Při inferenci se během nízkých jednotek vteřin ohodnotí všechny primerové páry v databázi, což je aktuálně celkem 5661 primerových párů.

Ohodnocené primerové páry jsou následně podle několika kritérií seřazeny a výsledky jsou zobrazeny uživateli. Výsledky u každého páru zahrnují kromě vypočítaných hodnot senzitivity a specificity i množství pareto dominantních primerových párů. Výsledky inference jsou zároveň vizualizovány v několika interaktivních grafech, které umožňují i filtraci výsledků.

Pro každý primerový pár lze zobrazit stránku s detailními informacemi. Uživatel zde zjistí pozici amplifikovaného regionu, primerové sekvence a s využitím interaktivní mapy nerozlišitelností, může zjistit, které všechny druhy,při použití daného páru, nebude možné plně rozlišit.

Samozřejmostí je i přítomnost jednoduchého prohlížeče primerových párů, kde uživatel může procházet kompletní databázi a vyhledávat páry podle základních parametrů.

Další důležitou součástí aplikace je možnost analýzy vlastního primerového páru. Uživatel tedy může prostřednictvím formuláře v aplikaci, spustit výpočet analýzy nové primerové sekvence. K realizaci analýzy nového páru bylo potřeba integrovat analytický nástroj přímo do aplikace.

První iteraci analytického nástroje jsem dostal k dispozici na počátku řešení práce. Jednalo se o nástroj implementovaný ve funkcionálním jazyce Clojure.

V průběhu řešení jsem se rozhodl tento nástroj z důvodu špatné výkonosti kompletně nahradit. Funkcionalitu nástroje jsem zcela reimplementoval a optimalizoval v jazyce Python. Díky optimalizaci jsem dosáhl velmi výrazného výkonnostního zlepšení výpočtu.

Výpočet, který stará implementace realizovala jednotky hodin (cca 5-6 hodin), nový nástroj vypočítá se stejným výpočetním výkonem během jednotek minut (cca 2 minuty). Méně výrazný, ale přesto znatelný je i trend ve spotřebě paměti. Kde se spotřeba paměti staré implementace pohybovala kolem 5 až 6 GB, se nová implementace pohybuje kolem 400 MB. Díky výrazné optimalizaci se celková integrace analytické funkce do aplikace značně zjednodušila.

Spouštění uživatelských analytických úloh si vyžádalo rozšíření aplikace o množinu výpočetních uzlů a frontu úloh. Nejen z tohoto důvodu jsem se rozhodl aplikaci provozovat v distribuovaném prostředí typu Kubernetes. Což ve výsledku zjednodušilo správu nejen výpočetních uzlů, ale i celého zbytku aplikace, včetně možnosti horizontálního škálování určitých částí aplikace.

Aktuálně je aplikace dostupná prostřednictvím domény https://primerio.eu/ a umožňuje jakémukoliv uživateli internetu, využít implementované mechanizmy a algoritmy realizující pokročilou práci s primerovými páry a jejich sekvencemi.

Literatura

[1] *16S rRNA, One of the Most Important rRNAs* [online]. [cit. 2022-01-08]. Dostupné z: https://www.cd-genomics.com/blog/16s-rrna-one-of-the-most-important-rrnas/.

[2] *Apache HTTP Server*. Dostupné z: https://httpd.apache.org/.

[3] *Axios*. Dostupné z: https://axios-http.com/.

[4] *Bootstrap*. Dostupné z: https://getbootstrap.com.

[5] *Celery*. Dostupné z: https://docs.celeryq.dev/en/stable/.

[6] *Cert manager*. Dostupné z: https://cert-manager.io/.

[7] *D3.js*. Dostupné z: https://d3js.org/.

[8] *DataTables*. Dostupné z: https://datatables.net/.

[9] *Djagno REST framework*. Dostupné z: https://www.django-rest-framework.org/.

[10] *Django Autocomplete Light*. Dostupné z: https://github.com/yourlabs/django-autocomplete-light.

[11] *Django documentation - How to use sessions* [online]. [cit. 2022-4-28]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/http/sessions/.

[12] *Django Q*. Dostupné z: https://github.com/Koed00/django-q.

[13] *Docker*. Dostupné z: https://www.docker.com/.

[14] *Flexible I/O Tester*. Dostupné z: https://github.com/axboe/fio.

[15] *Font Awesome*. Dostupné z: https://fontawesome.com/.

[16] *GitLab*. Dostupné z: https://gitlab.com/.

[17] *GraphQL*. Dostupné z: https://graphql.org/.

[18] *Gunicorn*. Dostupné z: https://gunicorn.org/.

[19] *An introduction to Next-Generation Sequencing Technology* [online]. [cit. 2022-01-08]. Dostupné z: https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf.

[20] *Jinja - A very fast and expressive template engine*. Dostupné z: https://github.com/pallets/jinja.

[21] *Kubernetes*. Dostupné z: https://kubernetes.io/.

[22] *Let's Encrypt*. Dostupné z: https://letsencrypt.org/.

[23] *MariaDB*. Dostupné z: https://mariadb.org/.

[24] *MySQL*. Dostupné z: https://www.mysql.com/.

[25] *The Nobel Prize in Chemistry 1993* [online]. [cit. 2022-4-28]. Dostupné z: https://www.nobelprize.org/prizes/chemistry/1993/summary/.

[26] *PostgreSQL*. Dostupné z: https://www.postgresql.org/.

[27] *Redis*. Dostupné z: https://redis.io/.

[28] *Roche to close 454 Life Sciences as it reduces gene sequencing focus* [online]. [cit. 2022-01-08]. Dostupné z: https://www.fiercebiotech.com/medical-devices/roche-to-close-454-life-sciences-as-it-reduces-gene-sequencing-focus.

[29] *RQ - Redis Queue*. Dostupné z: https://python-rq.org/.

[30] *Stolon*. Dostupné z: https://github.com/sorintlab/stolon.

[31] *Swagger*. Dostupné z: https://swagger.io/.

[32] *Taxonomy* [online]. [cit. 2022-01-08]. Dostupné z: https://www.biologyonline.com/dictionary/taxonomy.

[33] *Zalando Postgres Operator*. Dostupné z: https://github.com/zalando/postgres-operator.

[34] A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*. 1975, sv. 94, č. 3, s. 441–448. ISSN 0022-2836.

[35] AL., A. B. et. *Molecular Biology of the Cell*. 4. vyd. Garland Publishing Inc., 1983. ISBN 0-8153-3218-1.

[36] ALBERTS B. BRAY D. KAREN H. JOHNSON A. LEWIS J. RAFF, M. R. K. W. P. *Essential cell biology*. 4. vyd. Garland Science, 2014. ISBN 978-0-8153-4454-4.

[37] ALLARD, G., RYAN, F. J., JEFFERY, I. B. a CLAESSON, M. J. SPINGO: a rapid species-classifier for microbial amplicon sequences. *BMC Bioinformatics*. Oct 2015, sv. 16, č. 1, s. 324. ISSN 1471-2105.

[38] BROSIUS, J. The Fragmented Gene. *Annals of the New York Academy of Sciences*. 2009, sv. 1178, č. 1, s. 186–193.

[39] CANARD, B. a SARFATI, R. S. DNA polymerase fluorescent substrates with reversible 3-tags. *Gene*. 1994, sv. 148, č. 1, s. 1–6. ISSN 0378-1119.

[40] Case, R. J., Boucher, Y., Dahllöf, I., Holmström, C., Doolittle, W. F. et al. Use of 16S rRNA and rpoB genes as molecular markers for microbial ecology studies. *Appl Environ Microbiol*. říjen 2006, sv. 73, č. 1, s. 278–288.

[41] Chakravorty, S., Helb, D., Burday, M., Connell, N. a Alland, D. A detailed analysis of 16S ribosomal RNA gene segments for the diagnosis of pathogenic bacteria. *J Microbiol Methods*. únor 2007, sv. 69, č. 2, s. 330–339.

[42] Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L. a Rice, P. M. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic acids research*. 2009/12/16. Oxford University Press. Apr 2010, sv. 38, č. 6, s. 1767–1771.

[43] Cole, J. R., Wang, Q., Fish, J. A., Chai, B., McGarrell, D. M. et al. Ribosomal Database Project: data and tools for high throughput rRNA analysis. *Nucleic Acids Research*. Listopad 2013, sv. 42, D1, s. D633–D642. ISSN 0305-1048.

[44] Conlan, S., Kong, H. H. a Segre, J. A. Species-level analysis of DNA sequence data from the NIH Human Microbiome Project. *PLoS One*. říjen 2012, sv. 7, č. 10.

[45] Cornish Bowden, A. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic acids research*. May 1985, sv. 13, č. 9, s. 3021–3030.

[46] Django Software Foundation. *Django*. Dostupné z: https://djangoproject.com.

[47] Edgar, R. C. SINTAX: a simple non-Bayesian taxonomy classifier for 16S and ITS sequences. Cold Spring Harbor Laboratory. 2016.

[48] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. Disertační práce. UNIVERSITY OF CALIFORNIA, IRVINE. Chapter 5. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[49] Glöckner, F. O., Yilmaz, P., Quast, C., Gerken, J., Beccati, A. et al. 25 years of serving the community with ribosomal RNA gene reference databases and tools. *J Biotechnol*. Netherlands: [b.n.]. červen 2017, sv. 261, s. 169–176.

[50] Greuter, D., Loy, A., Horn, M. a Rattei, T. probeBase—an online resource for rRNA-targeted oligonucleotide probes and primers: new features 2016. *Nucleic Acids Research*. Listopad 2015, sv. 44, D1, s. D586–D589. ISSN 0305-1048.

[51] Grierson, D. E. Pareto multi-criteria decision making. *Advanced Engineering Informatics*. 2008, sv. 22, č. 3, s. 371–384. ISSN 1474-0346. Collaborative Design and Manufacturing.

[52] Heather, J. M. a Chain, B. The sequence of sequencers: The history of sequencing DNA. *Genomics*. 2016, sv. 107, č. 1, s. 1–8. DOI: https://doi.org/10.1016/j.ygeno.2015.11.003. ISSN 0888-7543.

[53] Kim, D. J., Yang, J., Seo, H., Lee, W. H., Ho Lee, D. et al. Colorectal cancer diagnostic model utilizing metagenomic and metabolomic data of stool microbial extracellular vesicles. *Scientific Reports*. Feb 2020, sv. 10, č. 1, s. 2860. ISSN 2045-2322.

[54] Klindworth, A., Pruesse, E., Schweer, T., Peplies, J., Quast, C. et al. Evaluation of general 16S ribosomal RNA gene PCR primers for classical and next-generation sequencing-based diversity studies. *Nucleic Acids Research*. Srpen 2012, sv. 41, č. 1, s. e1–e1. ISSN 0305-1048.

[55] Lai, J., Li, A., Jiang, J., Yuan, X., Zhang, P. et al. Metagenomic analysis reveals gut bacterial signatures for diagnosis and treatment outcome prediction in bipolar depression. *Psychiatry Research*. 2022, sv. 307, s. 114326. ISSN 0165-1781.

[56] Liu, Y.-X., Qin, Y., Chen, T., Lu, M., Qian, X. et al. A practical guide to amplicon and metagenomic analysis of microbiome data. *Protein & Cell*. May 2021, sv. 12, č. 5, s. 315–330. ISSN 1674-8018.

[57] Madden, T. The BLAST sequence analysis tool. *The NCBI Handbook*. Leden 2002.

[58] Masse, M. *REST API Design Rulebook*. Sebastopol: O'Reilly Media, 2011. ISBN 978-1-449-31050-9.

[59] Maxam, A. M. a Gilbert, W. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences*. National Academy of Sciences. 1977, sv. 74, č. 2, s. 560–564. DOI: 10.1073/pnas.74.2.560. ISSN 0027-8424.

[60] McKinney Wes. Data Structures for Statistical Computing in Python. In: Walt Stéfan van der a Millman Jarrod, ed. *Proceedings of the 9th Python in Science Conference*. 2010, s. 56 – 61.

[61] Nakayama, J., Jiang, J., Watanabe, K., Chen, K., Ninxin, H. et al. Up to Species-level Community Analysis of Human Gut Microbiota by 16S rRNA Amplicon Pyrosequencing.

[62] Qian, Y., Yang, X., Xu, S., Huang, P., Li, B. et al. Gut metagenomics-derived genes as potential biomarkers of Parkinson's disease. *Brain*. Srpen 2020, sv. 143, č. 8, s. 2474–2489. ISSN 0006-8950.

[63] Quince, C., Walker, A. W., Simpson, J. T., Loman, N. J. a Segata, N. Shotgun metagenomics, from sampling to analysis. *Nature Biotechnology*. Sep 2017, sv. 35, č. 9, s. 833–844. ISSN 1546-1696.

[64] Saiki, R. K., Scharf, S., Faloona, F., Mullis, K. B., Horn, G. T. et al. Enzymatic amplification of beta-globin genomic sequences and restriction site analysis for diagnosis of sickle cell anemia. *Science*. prosinec 1985, sv. 230, č. 4732, s. 1350–1354.

[65] Schleif, R. *Genetics and Molecular Biology*. 2. vyd. The Johns Hopkins University Press, 1993. ISBN 0-8018-4673-0.

[66] Slatko, B. E., Gardner, A. F. a Ausubel, F. M. Overview of Next-Generation Sequencing Technologies. *Curr Protoc Mol Biol*. duben 2018, sv. 122, č. 1.

[67] Staden, R. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Res*. červen 1979, sv. 6, č. 7, s. 2601–2610.

[68] Storz, G. An Expanding Universe of Noncoding RNAs. *Science*. 2002, sv. 296, č. 5571, s. 1260–1263.

[69] The pandas development team. *Pandas-dev/pandas: Pandas*. Zenodo, únor 2020.

[70] Tuzhikov, A., Panchin, A. a Shestopalov, V. I. TUIT, a BLAST-based tool for taxonomic classification of nucleotide sequences. *Biotechniques*. únor 2014, sv. 56, č. 2, s. 78–84.

[71] Wang, Q., Garrity, G. M., Tiedje, J. M. a Cole, J. R. Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl Environ Microbiol*. červen 2007, sv. 73, č. 16, s. 5261–5267.

[72] Waters, K. Molecular Genetics. In: Zalta, E. N., ed. *The Stanford Encyclopedia of Philosophy* [https://plato.stanford.edu/archives/fall2013/entries/molecular-genetics/]. Fall 2013. Metaphysics Research Lab, Stanford University, 2013.

[73] Weisburg, W. G., Barns, S. M., Pelletier, D. A. a Lane, D. J. 16S ribosomal DNA amplification for phylogenetic study. *J Bacteriol*. leden 1991, sv. 173, č. 2, s. 697–703.

[74] Woese, C. R. a Fox, G. E. Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc Natl Acad Sci U S A*. listopad 1977, sv. 74, č. 11, s. 5088–5090.

[75] Wooley, J. C. a Ye, Y. Metagenomics: Facts and Artifacts, and Computational Challenges*. *J Comput Sci Technol*. leden 2009, sv. 25, č. 1, s. 71–81.

[76] Ye, J., Coulouris, G., Zaretskaya, I., Cutcutache, I., Rozen, S. et al. Primer-BLAST: A tool to design target-specific primers for polymerase chain reaction. *BMC Bioinformatics*. Jun 2012, sv. 13, č. 1, s. 134.

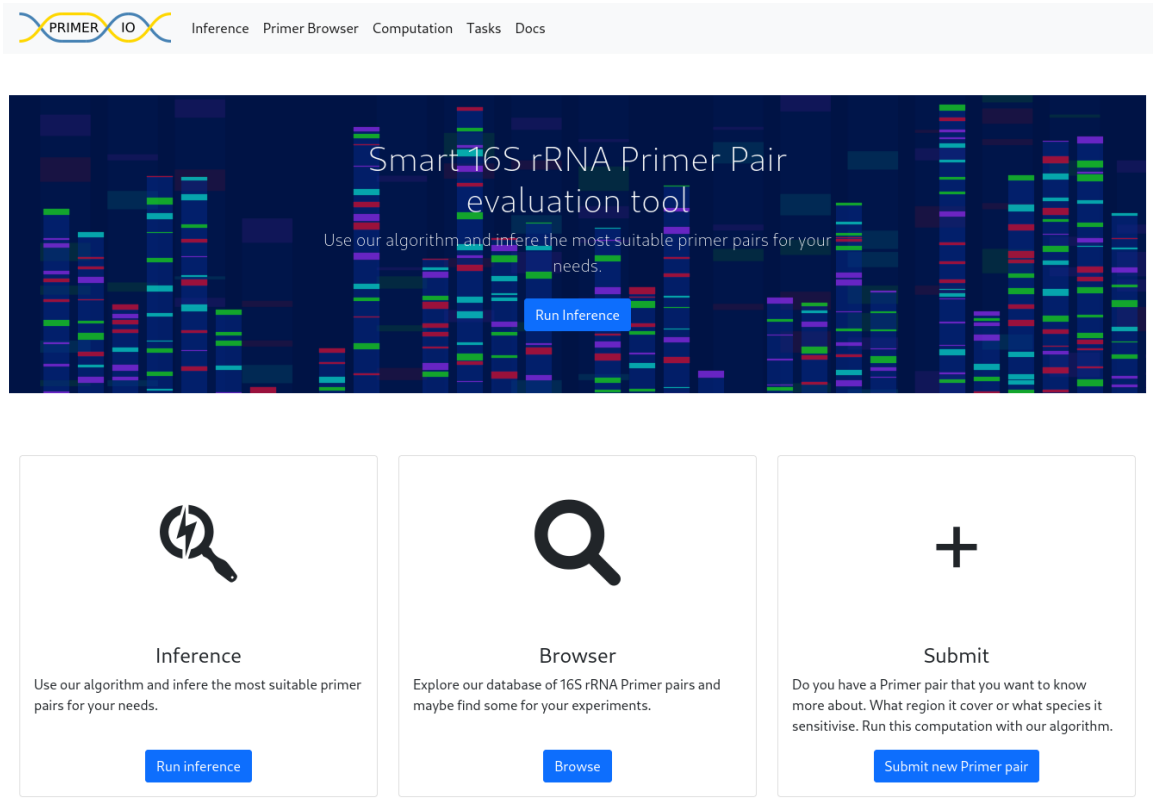
Příloha A

Obsah přiloženého paměťového média

deploy	Složka s Kubernetes manifestem
Dockerfile.staticfiles	Docker šablona obrazu se statickými soubory
Dockerfile.web	Docker šablona obrazu aplikačního webového serveru
Dockerfile.worker	Docker šablona obrazu výpočetního uzlu
misc	Podpůrné skripty
Pipfile	Python závislosti
primerio	Zdrojové kódy aplikace

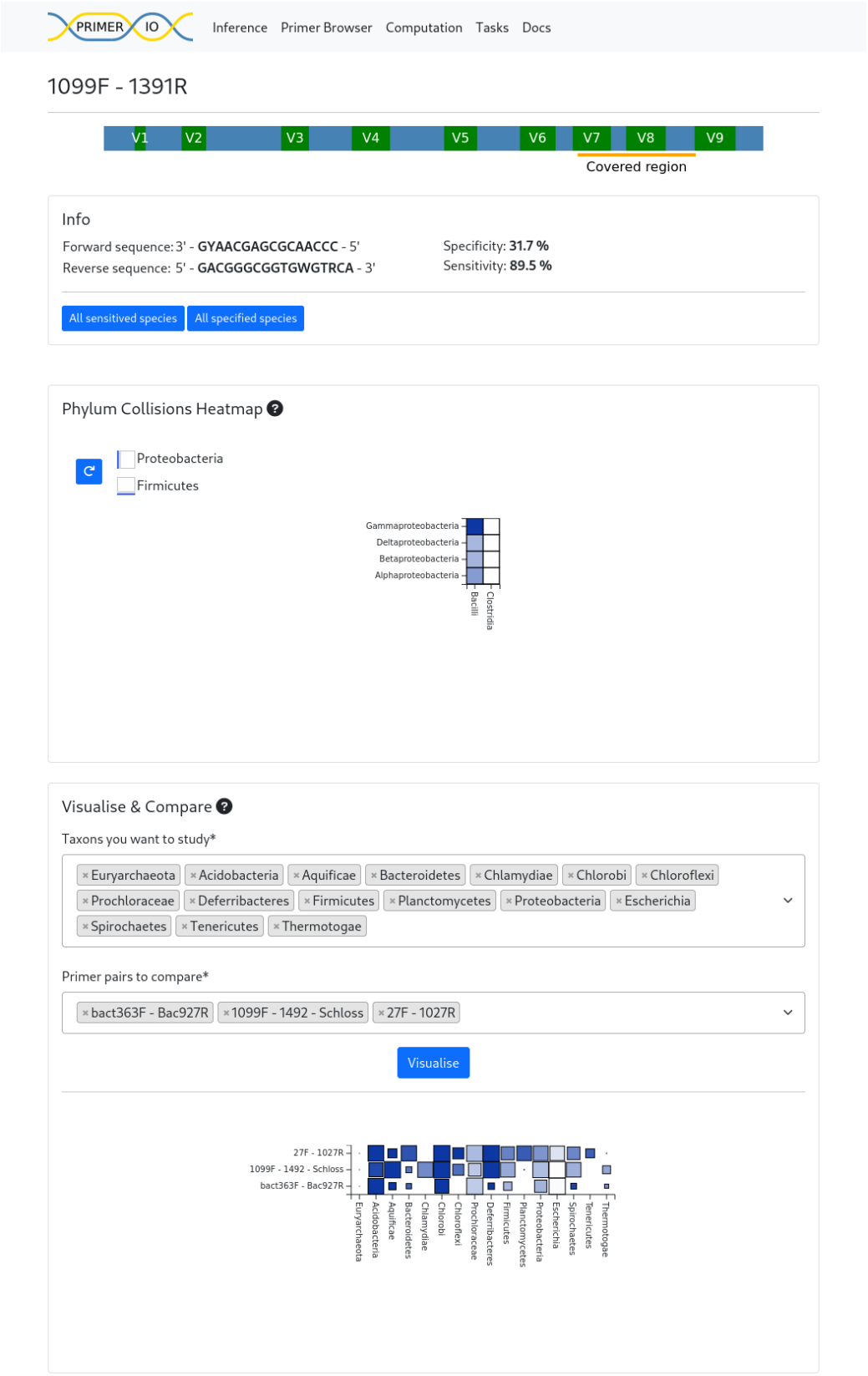
Příloha B

Přehled stránek uživatelského rozhraní

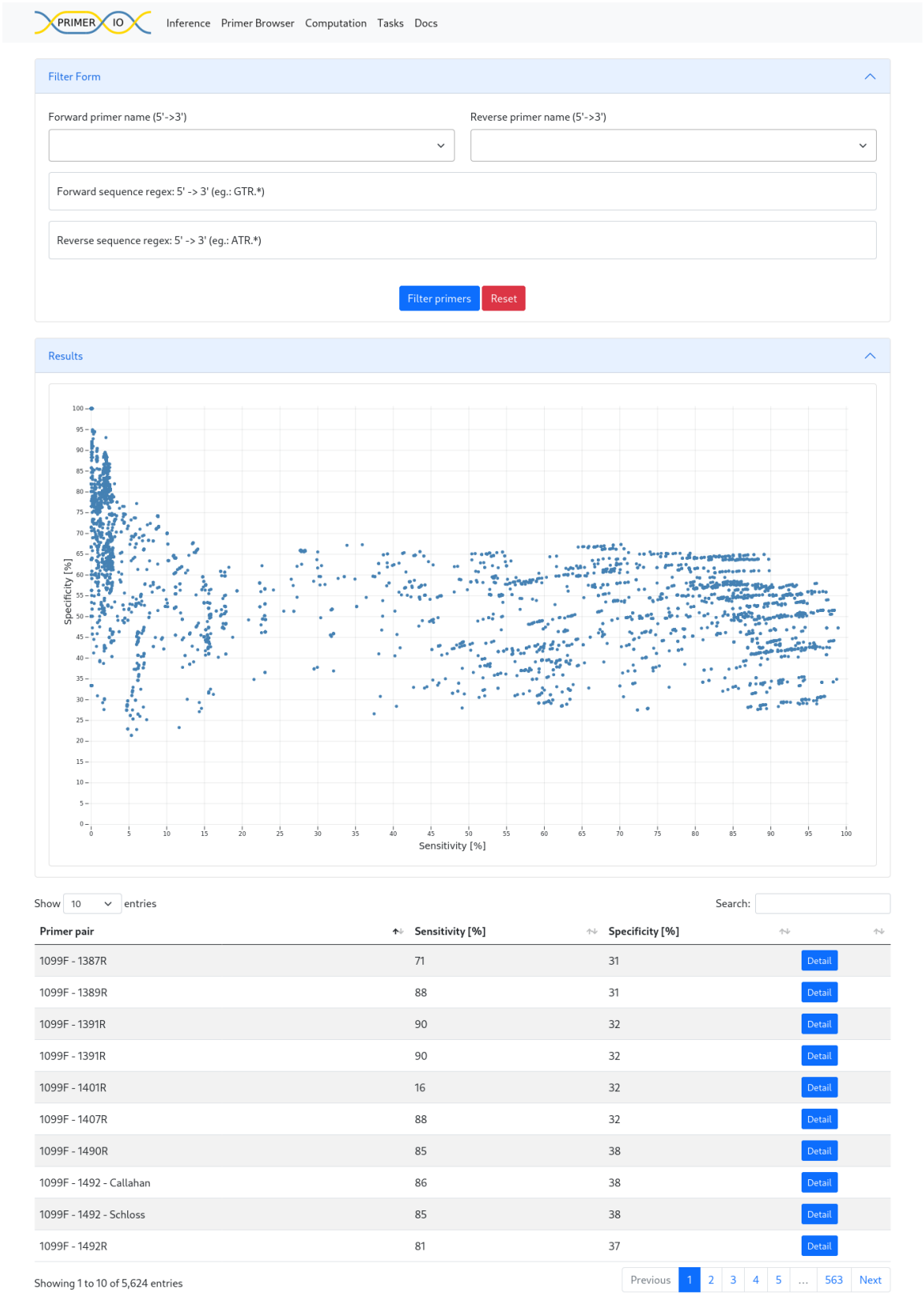


Obrázek B.1: Snímek hlavní strany aplikace.

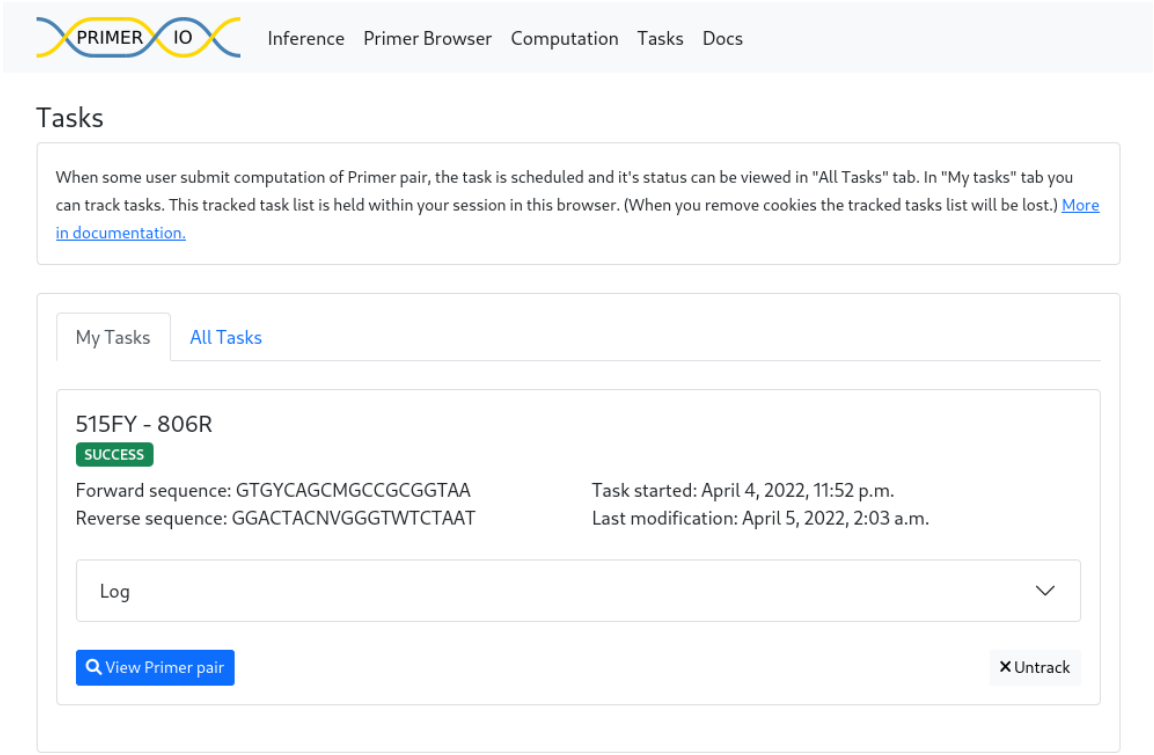
Obrázek B.2: Snímek výsledků inference.



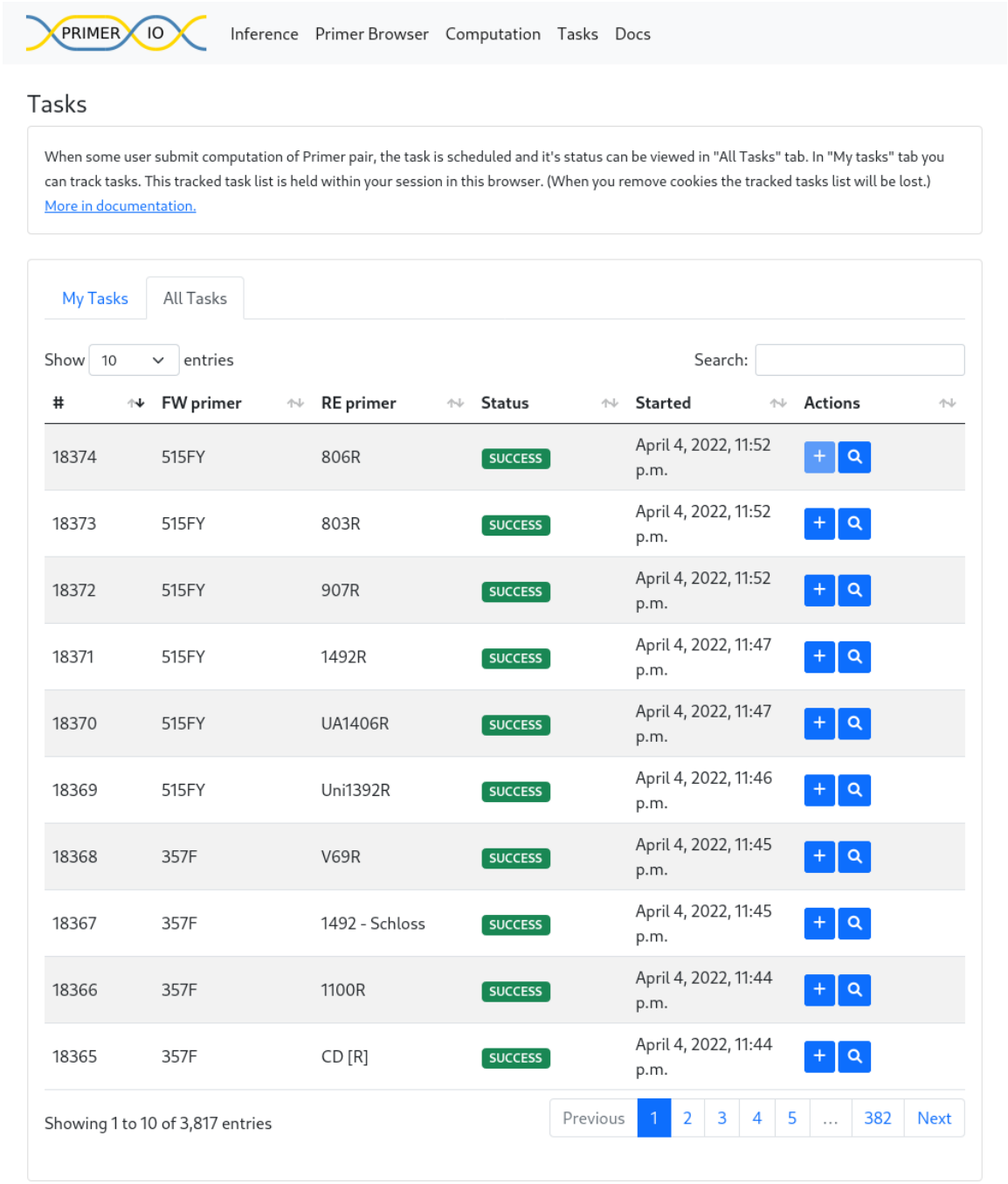
Obrázek B.3: Snímek stránky s detailem primerového páru.



Obrázek B.4: Snímek stránky prohlížeče primerových párů.



Obrázek B.5: Snímek stránky se seznamem uživatelem sledovaných úloh.



Obrázek B.6: Snímek stránky se seznamem všech úloh.

Příloha C

Aspekty nasazení aplikace

Ukázky Kubernetes konfigurací

V kapitole 5 je stručně popsán princip systému Kubernetes a jeho konfiguračních souborů. Z důvodu jejich velikosti jsem se rozhodl vložit některé ukázky zde do přílohy. Kompletní manifest je obsažen v odevzdaných souborech.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: primerio-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    kubernetes.io/tls-acme: "true"
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  tls:
    - hosts:
        - "primerio.eu"
      secretName: primerio-eu
  rules:
  - host: "primerio.eu"
    http:
      paths:
      - backend:
          service:
            name: primerio-service
            port:
              number: 80
        pathType: ImplementationSpecific
```

Obrázek C.1: Ukázka konfigurace domény a https.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: primerio-worker
spec:
  replicas: 2
  selector:
    matchLabels:
      app: primerio-worker
  template:
    metadata:
      labels:
        app: primerio-worker
    spec:
      containers:
      - name: primerio-worker-pod
        env:
          - name: REDIS_HOST
            value: redis-service
          - name: EMAIL_HOST
            value: "host"
          - name: EMAIL_HOST_USER
            value: "user"
          - name: EMAIL_HOST_PASSWORD
            value: "pass"
          - name: EMAIL_USE_TLS
            value: "true"
          - name: DB_HOST
            value: "thesis.tensorcloud.cz"
        resources:
          requests:
            memory: "800Mi"
          limits:
            cpu: "1"
            memory: "1500Mi"
        image: registry.gitlab.com/kohoutovice/diplomka:worker
        imagePullPolicy: Always
        securityContext:
          allowPrivilegeEscalation: false
          capabilities: {}
          privileged: false
          readOnlyRootFilesystem: false
          runAsUser: 1000
      imagePullSecrets:
        - name: gitlab-pull-secret
```

Obrázek C.2: Ukázka konfigurace deploymentu pro výpočetní uzly aplikace.

Příloha D

Příklady použití aplikace

Aktuálně je systém nasazen v rámci serverů *cloud.muni.cz*, které je provozováno jako *best effort* řešení a přestože většinu času běží spolehlivě. Během vývoje jsem se setkal s mnohými výpadky, případně velmi intenzivními výkonnostními propady.

Inference

Příkladem využití inference může být situace kdy jsou cílem studia dva bakteriální kmeny: Proteobacteria a Firmicutes.

1. Nejdříve otevřeme stránku aplikace. https://primerio.eu/
2. Na hlavní stránce klikneme na Inference. (Buď v boxíku, nebo na liště.)
3. Do políčka pro taxony napíšeme postupně zvolené kmeny a vybereme ze seznamu.
4. Klikneme na tlačítko *Run inference*.
5. Stránka se znovu načte a na server se vyšle požadavek na inferenci. Chvíli počkáme, protože aktuálně je databáze provozována na instanci s HDD úložištěm a rychlost může být lehce omezena.
6. Jakmile je inference provedena, na stránce se vykreslí několik grafů a přehledová tabulka ohodnocených primerů.
7. Teď je možné v obou bodových grafech vybrat oblast hodnot senzitivity a specificity, které vyhovují nejvíce a následně v seřazené tabulce primerových párů zvolit vybraný pár. Tabulka je seřazena, dle všech parametrů, prioritně zleva do prava. Řazení si lze upravit kliknutím na hlavičku sloupce. Případně řazení podle několika sloupců je možné klikem na hlavičku sloupce za současného držení klávesy shift.

Detail primerového páru

Ve výsledcích inference, nebo v prohlížeči primerových párů, je možné přistoupit k detailu primerového páru. Zde je možné zjistit pozici amplifikovaného regionu, sekvence primerů

a nerozlišitelnosti pomocí interaktivní mapy. V této mapě je možné se kliknutím na položku zanořit do taxonomické hierarchie a postupně až na úroveň druhu zjistit jednotlivé nerozlišitelnosti.

Spuštění analýzy primerového páru

Kliknutím na možnost *Computation* se uživatel dostane na formulář spuštění výpočtu. Pro účely otestování aplikace, je možné postupovat takto:

1. Z prohlížeče primerových párů si vybereme libovolný pár a z náhledu detailu zkopírujeme obě sekvence do formuláře pro výpočet.
2. Vyplníme jména primerů libovolně.
3. K jednomu (nebo oběma) primerům připíšeme na konec sekvence znak "N"reprezentující jakýkoliv nukleotid. Díky tomu se funkce primeru nezmění, ale v databázi ještě nebude takový primer pravděpodobně existovat.
4. Vyplníme emailovou adresu a spustíme výpočet.
5. Pokud jsme změnili pouze jednu sekvenci, pak aplikace automaticky identifikuje tu nezměněnou a přiřadí jí jméno, které už v databázi je.
6. Na email by měla dojít zpráva o zařazení úlohy.
7. Výpočet by se měl záhy spustit. Tento postup je možné sledovat na stránce *Tasks*.
8. Výpočet by podle zvoleného páru a vytížení databáze, měl být dokončen během několika jednotek až desítek minut.
9. V průběhu výpočtu, se jako první vypočítá amplifikovaný region. Pokud tedy klikneme z přehledu úloh na zobrazení primerového páru, můžeme zjistit pozici regionu ještě před úplným dokončení výpočtu.
10. Jakmile je výpočet dokončen, je možné si prohlédnout veškeré vypočítané informace na stránce detailu.

Pokud by primerový pár neamplifikoval žádný úsek 16S rRNA genu, úloha skončí s chybou a v logu úlohy se tato skutečnost zapíše.