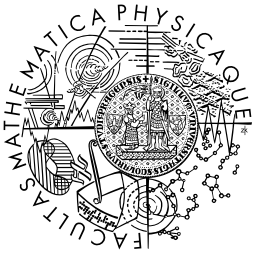


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Veronika Scheuerová

Webový plugin pro kombinovanou sekvenčně strukturní analýzu proteinů

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. David Hoksza, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Ráda bych poděkovala vedoucímu práce RNDr. Davidovi Hokszovi, Ph.D za cenné rady a pomoc v průběhu psaní práce a za čas, který věnoval konzultacím.

Název práce: Webový plugin pro kombinovanou sekvenčně strukturní analýzu proteinů

Autor: Veronika Scheuerová

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. David Hoksza, Ph.D., Katedra softwarového inženýrství

Abstrakt: Tato práce se zabývá modernizací webového pluginu MolArt, který slouží ke znázornění závislosti mezi anotovanou proteinovou sekvencí a experimentální nebo predikovanou 3D strukturou proteinu. Nový MolArt (MolArt 2.0) se skládá ze dvou komponent. První zajišťuje vizualizaci proteinové sekvence pomocí knihovny Nightingale. Druhá má na starosti vykreslení 3D struktury proteinu, k tomu je využita knihovna Mol*. Propojení těchto komponent spočívá v grafickém zvýraznění (i vícenásobném) odpovídajících si skupin aminokyselin v obou částech, což umožňuje snazší generování biologických hypotéz. Data jsou načítána z několika externích databází nebo je možné využít data poskytnutá uživatelem. MolArt 2.0 je implementován v jazyce TypeScript.

Klíčová slova: bioinformatika protein web

Title: Web-based plugin for combined sequence and structure analysis of proteins

Author: Veronika Scheuerová

Department: Department of Software Engineering

Supervisor: doc. RNDr. David Hoksza, Ph.D., Department of Software Engineering

Abstract: This work focuses on the modernization of the web plugin MolArt, which is used to illustrate the relationship between the annotated protein sequence and the experimental or predicted 3D structure of the protein. The new MolArt (MolArt 2.0) consists of two components. The first one provides visualization of the protein sequence using the Nightingale library. The second one is responsible for rendering the 3D structure of the protein using the Mol* library. We connect these two components by using graphical highlighting (even multi-highlighting) of corresponding groups of amino acids in both parts, which makes it easier to generate biological hypotheses. The data is read from several external databases or it is possible to use the data provided by the user. MolArt 2.0 is implemented in TypeScript.

Keywords: bioinformatics protein web

Obsah

Úvod **3**

1 Úvod do problematiky **4**
1.1 Úvod do biologie proteinových struktur 4
1.1.1 Aminokyseliny 4
1.1.2 Proteiny 4
1.1.3 Struktura proteinů 5
1.2 Používané veřejně dostupné datové zdroje 5
1.2.1 Datové formáty 5
1.2.2 Databáze 10
1.2.3 Data v MolArtu 2.0 11
1.3 Příbuzné projekty 14
1.3.1 ProtVista 14
1.3.2 Nightingale 14
1.3.3 LiteMol 14
1.3.4 Mol* 16
1.3.5 protvista-uniprot 16
1.3.6 ProtVista PDB 16
1.3.7 MolArt 18

2 Programátorská dokumentace **19**
2.1 uniprot-nightingale 19
2.1.1 Vstupní data 19
2.1.2 Objektový model 19
2.1.3 Integrace s Nightingale 31
2.2 MolstarPlugin 34
2.3 MolArt 36
2.3.1 StructureViewer 36
2.3.2 Veřejné rozhraní 37
2.4 Kompilace a debugování 39
2.5 Unit testy 39

3 Uživatelská dokumentace **40**
3.1 Přepínání struktur 40
3.2 Zvýrazňování v uniprot-nightingale 40
3.3 Zvýrazňování v Mol* 41
3.4 Filtrování mutací 41
3.5 Popis anotace 42
3.6 Přibližování 42

Závěr **43**

Seznam použité literatury **44**

Seznam obrázků **46**

Seznam použitých zkratek **47**

A Přílohy **48**
A.1 Zdrojové kódy 48
A.2 Typy popisující uživatelské anotace 48
A.3 Datový typ popisující uživatelské pokrytí sekvence a mapování . . 52
A.4 Git repozitáře 53
A.5 Npm 53

Úvod

Veřejně dostupné databáze poskytují mnoho proteinových sekvencí spolu s popisem výskytů zajímavých oblastí v dané sekvenci (anotací). Počet známých proteinových sekvencí velmi rychle roste, například v databázi UniProtKB (UniProt Knowledgebase) bylo v červnu roku 2021 přibližně 219 milionů automaticky anotovaných sekvencí, přičemž v dubnu jich bylo asi o 14 milionů méně (EBI, 2021). Proteinové sekvence procesem skládání proteinů zaujímají polohu ve 3D prostoru a tím tvoří 3D strukturu.

Pomocí různých experimentálních metod lze odhadnout tuto 3D strukturu. Každá takto vyzkoumaná 3D struktura má vlastní záznam v databázi i přesto, že vychází ze stejné proteinové sekvence. U větších proteinů se většinou zkoumá pouze část struktury, protože může být velmi složité zjistit experimentálními metodami celou strukturu najednou. Opačně se může stát, že některé 3D struktury v databázi mají přiřazenu více než jednu proteinovou sekvenci, protože tyto dvě sekvence tvoří komplex.

Existuje několik nástrojů, které dokáží vizualizovat 3D strukturu proteinu a také existují nástroje, které zobrazující sekvenci s anotacemi. MolArt je webový plugin, který vizualizuje proteinové sekvence společně s anotacemi a seznamem příslušných 3D struktur. Dále umožňuje zobrazit vedle sebe tuto proteinovou sekvenci a jakoukoliv z příslušných 3D struktur a zvýraznit anotace nebo vybraná místa synchronizovaně na obou částech.

Cílem naši práce je vytvořit novou verzi MolArtu, splňující následující požadavky:

- Zachová všechny důležité funkce původního MolArtu.
- Budou použity novější verze obou vizualizačních komponent.
- Rozhraní pluginu bude navrženo tak, aby bylo snadné nahradit 3D vizualizační komponentu.
- Kód bude psaný v jazyce TypeScript, což zajistí lepší udržitelnost a zjednoduší následný vývoj.
- Poskytne rozšíření možnosti zvýrazňování o vícenásobné zvýraznění.

Struktura práce

V první kapitole vysvětlujeme méně známé pojmy a uvádíme tuto práci do kontextu v rámci příbuzných projektů. Druhá kapitola obsahuje programátorskou dokumentaci, která popisuje strukturu projektu a objevené problémy a jejich řešení. Ve třetí kapitole je popsán návod k používání aplikace. Závěr obsahuje shrnutí práce a zamyšlení nad možnostmi rozšíření do budoucna.

1. Úvod do problematiky

V této kapitole seznámíme čtenáře se základními biologickými pojmy, ukážeme s jakými datovými zdroji budeme pracovat a představíme související projekty.

1.1 Úvod do biologie proteinových struktur

V následující sekci vysvětlíme několik biologických faktů, které jsou stěžejní pro porozumění textu a významu práce.

1.1.1 Aminokyseliny

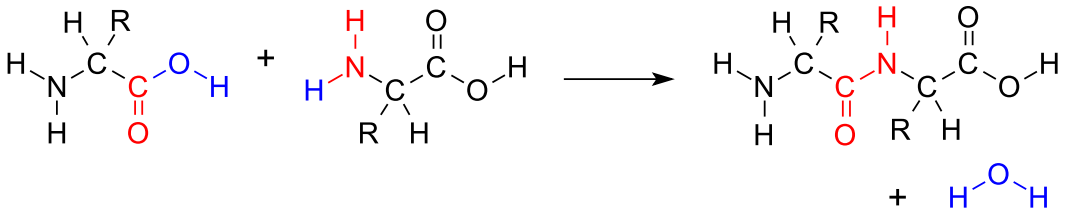
Aminokyseliny jsou obecně molekuly obsahující alespoň jednu karboxylovou ($-COOH$) a alespoň jednu aminovou ($-NH_2$) funkční skupinu[1]. Aminokyseliny jsou základním stavebním prvkem všech proteinů. Je známo kolem 500 různých aminokyselin, ale pouze 23 se jich podílí na vzniku bílkovin - tyto aminokyseliny se nazývají proteinogenní. Pod pojmem aminokyseliny rozumíme v rámci této práce právě proteinogenní podmnožinu všech aminokyselin.

Chemickou reakcí karboxylové a aminové skupiny vzniká **peptidická vazba** ($-CO-NH-$) a odštěpuje se molekula vody (H_2O). Zbytky aminokyselin po odštěpení molekuly vody se nazývají **rezidua**. Při této reakci se aminokyseliny pojí do peptidových řetězců, které dále tvoří bílkoviny. Ilustraci vzniku peptidické vazby představuje Obrázek 1.1.

1.1.2 Proteiny

Proteiny (MEFANET, 2019b) neboli bílkoviny jsou makromolekuly složené z řetězců aminokyselin propojených peptidickou vazbou. V organismu zastávají mnoho důležitých funkcí (stavební, transportní, obrannou, regulační a zajišťující pohyb). Význam proteinu je určen zejména jeho 3D strukturou.

[1] Funkční skupina je skupina atomů v molekule, která určuje charakteristické chemické reakce molekuly. (Wikipedia, 2021c)



Obrázek 1.1: Schéma vzniku peptidické vazby reakcí dvou aminokyselin. Červeně jsou značeny atomy tvořící peptidovou vazbu. Odštěpená molekula vody je zvýrazněna modře. Obrázek byl převzat z Wikipedia (2021d).

1.1.3 Struktura proteinů

Struktura bílkoviny (MEFANET, 2018) je určena uspořádáním aminokyselin v řetězci. Reprezentaci proteinu v prostoru zkoumáme na čtyřech níže popsaných úrovních. Názornou ukázku těchto úrovní představuje Obrázek 1.2.

Primární struktura

Primární struktura je dána pomocí DNA a představuje pořadí aminokyselin v bílkovině. Lze ji reprezentovat jako řetězec nad abecedou 23 znaků, kde každý značí jednu aminokyselinu. Zápis obvykle začíná aminokyselinou s volnou aminoskupinou a končí aminokyselinou s volnou karboxylovou skupinou.

Sekundární struktura

Sekundární struktura znázorňuje lokální prostorové uspořádání aminokyselin. Je podmíněna vznikem vodíkových můstků[2] na peptidické vazbě. Mezi nejčastější sekundární struktury patří alfa-šroubovice (alpha helix) a beta-skládaný list (beta sheet).

Terciární struktura

Terciární struktura určuje prostorovou reprezentaci celé molekuly proteinu. Tuto reprezentaci si můžeme představit jako uspořádanou množinu atomů v proteinu, kde každý atom je definovaný 3D souřadnicí a několika dalšími vlastnostmi (typ atomu, index aminokyseliny...).

Kvartérní struktura

Některé bílkoviny se skládají z více různých (heteromery) nebo stejných (homomery) terciárních struktur. Jejich vzájemné uspořádání popisuje kvartérní struktura.

1.2 Používané veřejně dostupné datové zdroje

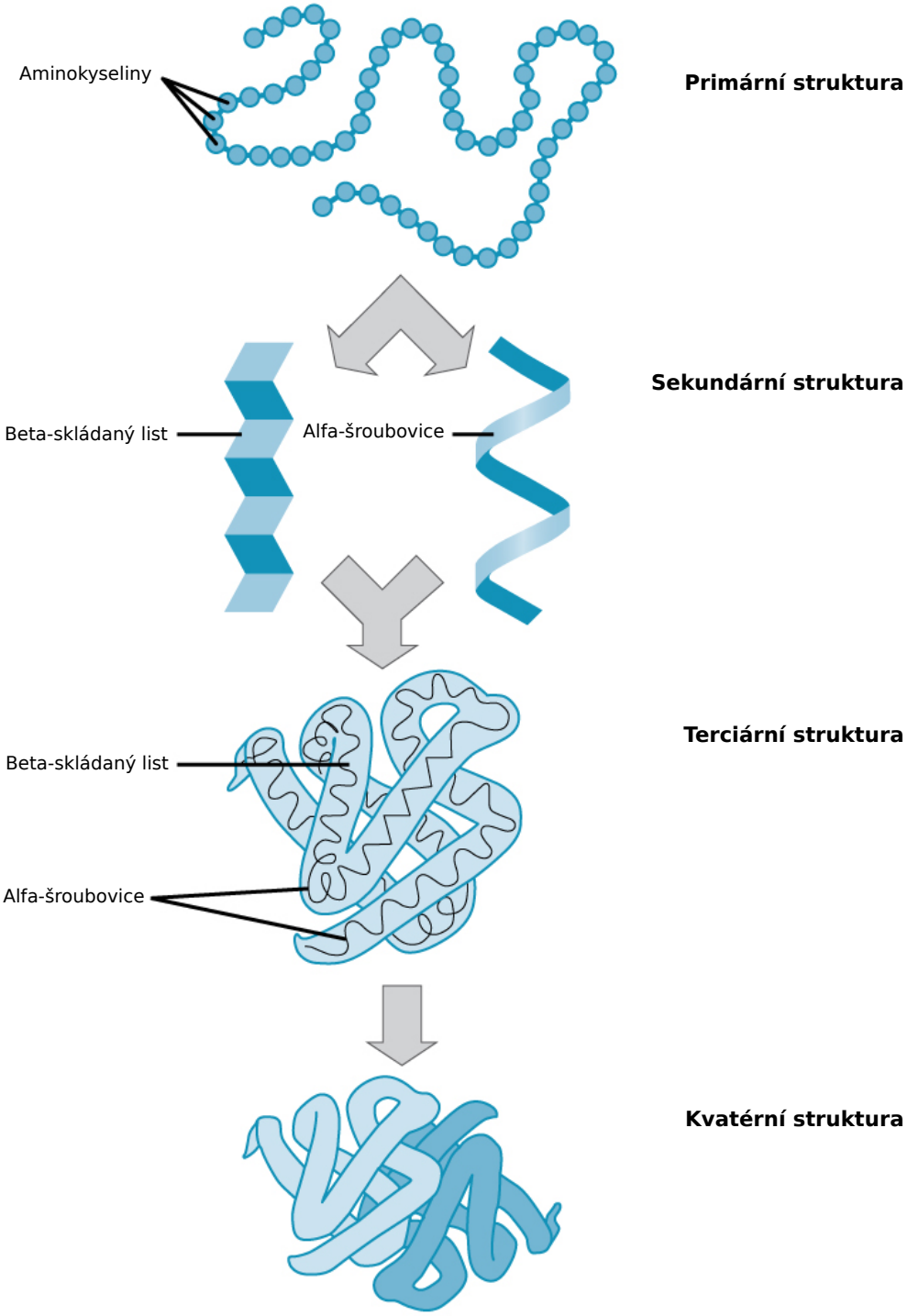
MolArt 2.0 načítá data z veřejně dostupných datový zdrojů. Tyto zdroje poskytují informace o sekvenci aminokyselin s anotacemi, o struktuře proteinu a mapování mezi nimi.

V následujících dvou sekcích popíšeme datové formáty a databáze, které MolArt 2.0 využívá. Ve třetí sekci vysvětlíme, jak MolArt 2.0 pracuje se získanými daty.

1.2.1 Datové formáty

Většina dotazovaných databází poskytuje data v univerzálním formátu JSON (JavaScript Object Notation), ale vyskytují se i účelově specifické bioinformatické formáty, které budou popsány níže.

[2]Vodíkový můstek tvoří vodík se silně elektronegativním atomem a atom s volným elektronegativním párem (Bartovská a Šišková, 2005).



Obrázek 1.2: Úrovně struktur proteinů. Obrázek byl převzat z Commons (2007) a byl upraven.

FASTA

FASTA (Wikipedia, 2021b) je textový formát používaný v bioinformatice zejména pro reprezentaci nukleotidových bází[3] nebo aminokyselinových sekvencí. Na prvním řádku souboru ve formátu FASTA je hlavička začínající znakem >, za kterým následuje unikátní identifikátor nebo název sekvence a další nepovinné údaje. Na ostatních řádcích je zapsána sekvence pomocí jednopísmenných kódů aminokyselin či nukleotidů. Délka řádku není striktně definovaná, ale doporučuje se, aby nepřekračovala 80 znaků. Ukázku dokumentu v tomto formátu představuje Kód 1.1.

```
1 >sp|P37840|SYUA_HUMAN Alpha-synuclein OS=Homo sapiens OX=9606 GN=
     ↪SNCA PE=1 SV=1
2 MDVFMKGLSKAKEGVVAAAEKTKQGVAEAAGKTKEGVLYVGSKTKEGVVHGVATVAEKTK
3 EQVTNVGGAVVTGVTAVAQKTVEGAGSIAAATGFVKKDQLGKNEEGAPQEGILEDMPVDP
4 DNEAYEMPSEEGYQDYEPEA
```

Kód 1.1: Příklad proteinové sekvence ve formátu FASTA načtené z databáze UniProt.

PDB

PDB[4] textový formát je používán k reprezentaci 3D struktury proteinu. Obsahuje souřadnice atomů a další informace o experimentu či sekundární struktuře. V tomto formátu však není možné zobrazit velké struktury, proto byl nahrazen formátem PDBx/mmCIF. Příklad souboru ve formátu PDB ukazuje Kód 1.2.

```
HEADER    HORMONE                                 17-OCT-77   1GCN
TITLE     X-RAY ANALYSIS OF GLUCAGON AND ITS RELATIONSHIP TO
     ↪RECEPTOR
TITLE    2 BINDING
⋮
DBREF  1GCN A    1    29  UNP    P01274   GLUC_PIG        33
     ↪61
SEQRES   1 A   29  HIS SER GLN GLY THR PHE THR SER ASP TYR SER LYS
     ↪ TYR
SEQRES   2 A   29  LEU ASP SER ARG ARG ALA GLN ASP PHE VAL GLN TRP
     ↪ LEU
SEQRES   3 A   29  MET ASN THR
HELIX    1   A PHE A    6  LEU A   26  1
     ↪                                  21
CRYST1   47.100   47.100   47.100  90.00  90.00  90.00 P 21 3
     ↪       12
ORIGX1      0.021231  0.000000  0.000000        0.00000
ORIGX2      0.000000  0.021231  0.000000        0.00000
ORIGX3      0.000000  0.000000  0.021231        0.00000
```

[3]Jedná se o dusíkaté báze obsažené v DNA a RNA, konkrétně Adenin, Guanin, Cytosin, Thymin a Uracil (MEFANET, 2019a).

[4]Dokumentace se nachází na `https://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html`.

```
SCALE1      0.021231  0.000000  0.000000        0.00000
SCALE2      0.000000  0.021231  0.000000        0.00000
SCALE3      0.000000  0.000000  0.021231        0.00000
ATOM      1  N   HIS A   1      49.668  24.248  10.436  1.00 25.00
   ↪           N
ATOM      2  CA  HIS A   1      50.197  25.578  10.784  1.00 16.00
   ↪           C
ATOM      3  C   HIS A   1      49.169  26.701  10.917  1.00 16.00
   ↪           C
ATOM      4  O   HIS A   1      48.241  26.524  11.749  1.00 16.00
   ↪           O
ATOM      5  CB  HIS A   1      51.312  26.048   9.843  1.00 16.00
   ↪           C
ATOM      6  CG  HIS A   1      50.958  26.068   8.340  1.00 16.00
   ↪           C
⋮
```

Kód 1.2: Částečná ukázka popisu struktury *1gcn* ve formátu PDB.

PDBx/mmCIF

PDBx/mmCIF[5] je rozšíření formátu mmCIF (v této práci budeme formát PDBx/mmCIF označovat jako mmCIF). Tento formát je určený k reprezentaci 3D struktury makromolekulárních dat. Nemá žádné omezení na velikost zobrazované molekuly (na rozdíl od formátu PDB) a od roku 2014 je používán jako standardní formát pro databázi PDB (Protein Data Bank) [6]. Data jsou reprezentována jako názvy atributů a jejich hodnoty (viz Kód 1.3).

```
data_1GCN
#
_entry.id   1GCN
#
_audit_conform.dict_name       mmcif_pdbx.dic
_audit_conform.dict_version    5.279
⋮
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
```

[5]Dokumentace se nachází na `https://mmcif.wwpdb.org/pdbx-mmcif-home-page.html`.
[6]PDB je databáze všech známých proteinových struktur.

```
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.pdbx_formal_charge
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM 1   N N   . HIS A 1 1  ? 49.668 24.248 10.436 1.00 25.00 ? 1
   ↪ HIS A N   1
ATOM 2   C CA  . HIS A 1 1  ? 50.197 25.578 10.784 1.00 16.00 ? 1
   ↪ HIS A CA  1
ATOM 3   C C   . HIS A 1 1  ? 49.169 26.701 10.917 1.00 16.00 ? 1
   ↪ HIS A C   1
ATOM 4   O O   . HIS A 1 1  ? 48.241 26.524 11.749 1.00 16.00 ? 1
   ↪ HIS A O   1
ATOM 5   C CB  . HIS A 1 1  ? 51.312 26.048 9.843  1.00 16.00 ? 1
   ↪ HIS A CB  1
ATOM 6   C CG  . HIS A 1 1  ? 50.958 26.068 8.340  1.00 16.00 ? 1
   ↪ HIS A CG  1
ATOM 7   N ND1 . HIS A 1 1  ? 49.636 26.144 7.860  1.00 16.00 ? 1
   ↪ HIS A ND1 1
⋮
```

Kód 1.3: Částečná ukázka popisu struktury *1gcn* ve formátu mmCif.

JSON

JSON je datový formát určený k reprezentaci a přenosu strukturovaných objektů. Vychází z objektové reprezentace v jazyce JavaScript, ale na programovacím jazyce je zcela nezávislý. Skládá se z párů atribut – hodnota, kde hodnota může představovat různé datové typy, například boolean, string, pole nebo další objekt (viz Kód 1.4).

```
{"P01274":[
        {
                "end":29,
                "chain_id":"A",
                "pdb_id":"1gcn",
                "start":1,
                "unp_end":81,
                "unp_start":53
        }
]}
```

Kód 1.4: Ukázka formátu JSON.

1.2.2 Databáze

V následující sekci popíšeme hlavní datové zdroje, které MolArt 2.0 využívá k získání informací o sekvenci a 3D struktuře proteinu. Všechny níže popsané databáze jsou veřejně přístupné.

UniProt

UniProt (The UniProt Consortium, 2020) je databáze poskytující informace o všech známých proteinových sekvencích s anotacemi. V roce 2021 obsahovala přes 219 000 000 automaticky anotovaných a přes 565 000 ručně anotovaných záznamů. Je vedena konsorciem UniProt, jehož členové jsou EBI (European Bioinformatics Institute), SIB (Swiss Institute of Bioinformatics) a PIR (Protein Information Resource). UniProt se skládá ze tří níže popsaných databází.

- **UniProtKB**[7] je hlavním zdrojem sekvencí a anotací. Podle zdroje dat je rozdělena na dvě části – UniProtKB/Swiss-Prot a UniProtKB/TrEMBL. Swiss-Prot poskytuje přezkoumané a ručně anotované záznamy. TrEMBL obsahuje data nekontrolovaná a automaticky anotovaná. Záznamům je přiřazen unikátní identifikátor (accession number). Příkladem takového identifikátoru mohou být *A2BC19* nebo *P12345*.

  Pro přístup k anotacím využíváme **Proteins REST API**[8], které je poskytuje ve formátu JSON. Sekvence získáváme pomocí **UniProt website REST API**[9] ve formátu FASTA (popsán v sekci 1.2.1) s hlavičkou[10], která obsahuje název databáze, unikátní identifikátor, název záznamu, název proteinu, název organismu, identifikátor organismu, název genu (nepovinně), míru věrohodnosti existence proteinu a verzi sekvence.
- **UniParc (UniProt Archive)**[11] je rozsáhlá databáze proteinových sekvencí bez anotací načtených z různých zdrojů, které ale mohou obsahovat duplicitní záznamy. V UniParc se ukládá každá sekvence pouze jednou a přiřazuje se ji unikátní identifikátor (**UPI**).
- **UniRef (UniProt Reference Clusters)**[12] snižuje počet sekvencí tím, že seskupuje příbuzné sekvence z UniProtKB a UniParc do jednoho záznamu.

PDB

PDB (Berman a kol., 2000) je databáze obsahující 3D struktury makromolekul, například proteinů. Je vedena organizací wwPDB (Worldwide PDB) (Berman a kol., 2003), jejíž zakladatelé jsou následující instituce: PDBe (PDB in Europe), RCSB (Research Collaboratory for Structural Bioinformatics) a PDBj (PDB Japan). V roce 2021 obsahovala databáze přibližně 180 000 struktur, kde každé je přiřazený unikátní čtyřmístný identifikátor (**PDB ID**), například *5uig* nebo *3q26*.

[7] Detailní popis je dostupný na `https://www.uniprot.org/help/uniprotkb`.
[8] Dokumentace se nachází na `https://www.uniprot.org/help/api`.
[9] Dokumentace se nachází na `https://www.ebi.ac.uk/proteins/api/doc/`.
[10] Detailní popis je dostupný na `https://www.uniprot.org/help/fasta-headers`.
[11] Detailní popis je dostupný na `https://www.uniprot.org/help/uniparc`.
[12] Detailní popis je dostupný na `https://www.uniprot.org/help/uniref`.

K popisu struktury proteinu ve formátu mmCIF (popsán v sekci 1.2.1) lze přistoupit pomocí **FTP**[13].

PDBe-KB (PDBe - Knowledge Base)

PDBe-KB (PDBe-KB consortium, 2019) je databáze zajišťovaná organizací PDBe. Obsahuje agregovaná data z různých datových zdrojů poskytujících anotace ke strukturám proteinů z databáze PDB. Tyto data sdružuje do několika skupin (PDB, Compounds, EMDB, SIFTS, Nucleic mappings, Pisa, Validation, Topology a Search), z nichž v MolArtu 2.0 využíváme SIFTS a PDB. SIFTS (Structure Integration with Function, Taxonomy and Sequence) (Dana a kol., 2018) popisuje mapování mezi sekvencí a strukturou proteinu na úrovni reziduí. Z PDB získáváme strukturální anotace pro záznamy v PDB databázi.

Pro přístup k databázi využíváme **PDBe REST API**[14] poskytující data ve formátu JSON.

SMR (SWISS-MODEL Repository)

SMR (Bienert a kol., 2016) je databáze modelů 3D struktur proteinů získaných pomocí homologického modelování. Tato technika k dané sekvenci predikuje model struktury na základě příbuznosti s proteinem, u kterého již známe strukturu i sekvenci. Vzorový protein se nazývá templát. Databáze je pravidelně aktualizována a obsahuje záznamy pro všechny proteinové sekvence modelových organismů (například člověk, myš domácí, ...) poskytovaných UniProtKB databází.

Databáze je přístupná přes **SMR REST API**[15], pomocí kterého stahujeme data ve formátu PDB (popsán v sekci 1.2.1).

1.2.3 Data v MolArtu 2.0

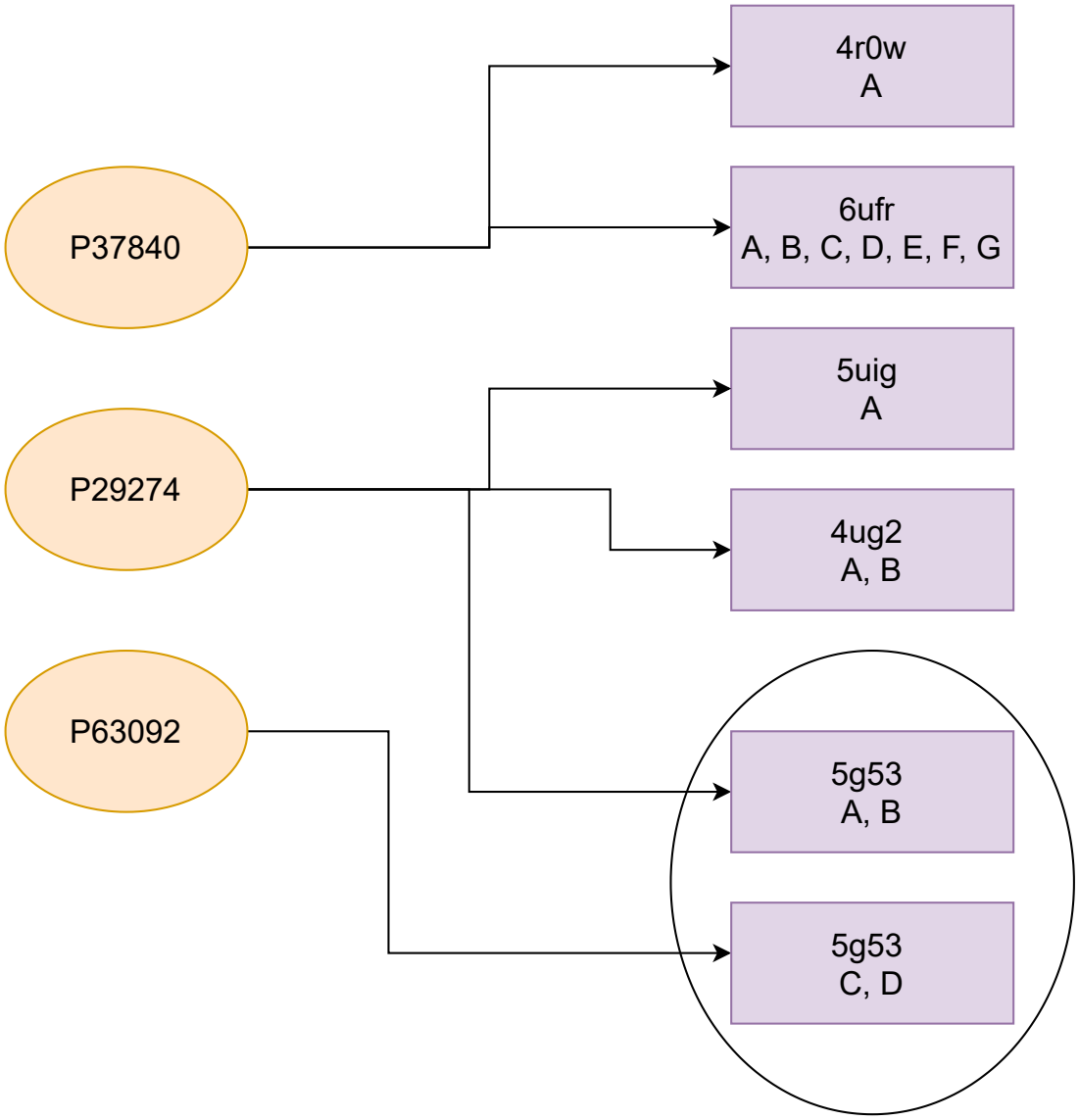
V následujících sekcích popíšeme, jak MolArt 2.0 získává potřebná data a jakým způsobem jsou vizuálně reprezentována.

Získávání dat

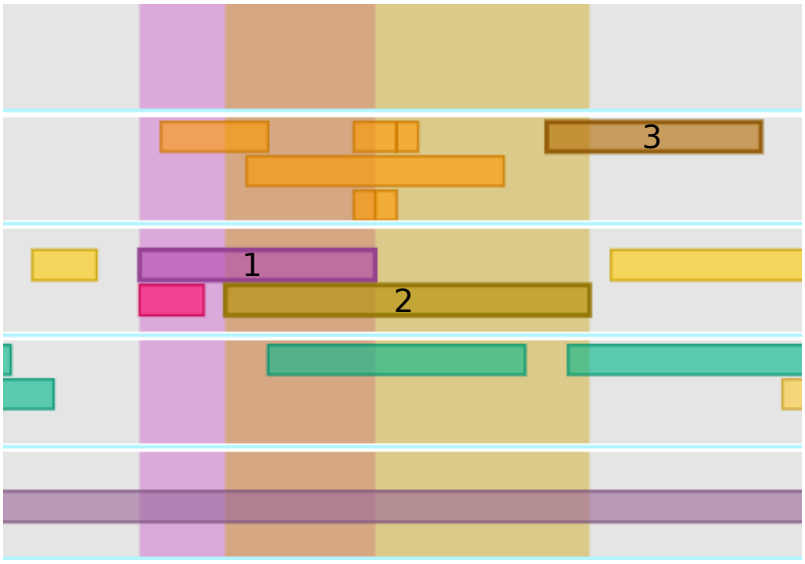
Řetězec aminokyselin je získán z databáze UniProtKB, dle zadaného identifikátoru. Poté pomocí stejného identifikátoru dohledáme v databázích PDBe-KB, respektive SMR informace o experimentálních, respektive predikovaných strukturách vztahujících se k dané sekvenci. Tuto operaci nazýváme **mapování první úrovně**. Příklad lze vidět na Obrázku 1.3. Pojem **mapování druhé úrovně** představuje mapování na úrovni reziduí, to znamená, že přesně definuje, které části sekvence z databáze UniProtKB se zobrazují na které části struktury. V této práci budeme pro oba výše definované pojmy používat zkrácený výraz **mapování**, půjde-li zamýšlený význam rozpoznat z kontextu.

Data o experimentálních strukturách získaných z databáze PDBe-KB představují především PDB ID daných struktur, mapování druhé úrovně a rozsahy

[13]Dokumentace se nachází na `https://www.wwpdb.org/ftp/pdb-ftp-sites`.
[14]Dokumentace se nachází na `https://www.ebi.ac.uk/pdbe/pdbe-rest-api`.
[15]Dokumentace se nachází na `https://swissmodel.expasy.org/docs/smr_openapi`.



Obrázek 1.3: Ukázka mapování první úrovně. Oranžové ovály představují UniProt sekvenci značenou identifikátorem. Fialové obdélníky reprezentují PDB strukturu - na prvním řádku je PDB ID kvartérní struktury, na druhém ID terciární struktury. Lze si všimnout, že některé různé sekvence jsou obsaženy ve stejné kvartérní struktuře a v rozdílné terciární struktuře. Toto je ilustrováno na sekvencích *P29274* a *P63092* a struktuře *5g53*.



Obrázek 1.4: Ukázka zvýrazněných a vybraných anotací. Anotace značené čísly **1** a **2** jsou zvýrazněné (tedy i vybrané) a anotace značená číslem **3** je vybraná.

pozorovaných segmentů[16]. Informace o predikovaných strukturách poskytovaných SMR databází obsahují identifikátor templátu, mapování druhé úrovně a odkaz na soubor s 3D souřadnicemi struktury.

Dalším druhem zobrazovaných dat jsou anotace. Ty popisují zajímavé oblasti v proteinové sekvenci. Jedná se o vazebná místa[17], sekundární struktury, aktivní místa[18], varianty (mutace) a další charakteristiky[19]. Anotace načítáme z databáze UniProtKB.

Vizualizace dat

Anotace se vizualizují nejčastěji jako obdélníky na řádcích v místě, kde pokrývají sekvenci. Možné jsou ale i jiné tvary. Nejvíce atypické zobrazení, které se skládá dokonce ze dvou komponent, mají varianty. První komponenta představuje pomocí grafu počet variant dané aminokyseliny. V grafu jsou barevně rozlišeny mutace způsobující nemoc (červený graf) a ostatní mutace (šedý graf). Druhá komponenta obsahuje informaci o alternativní aminokyselině příslušné mutace. Dále je přítomen filtr umožňující filtrovat data v obou komponentách podle zdroje dat nebo důsledků mutace.

Každý řádek, který obsahuje pokrytí experimentální struktury, predikované struktury nebo jeden typ anotace, patří do určité **kategorie**. Kategorie se ve výchozím stavu zobrazuje jako agregovaný pohled na všechny její položky. Příklady kategorií mohou být: Experimentální struktury (*Experimental structures*), Predi-

[16]Části molekuly proteinu, které se pohybují nejsou obvykle v experimentech pozorovány, a proto nejsou v záznamu v databázi PDB přítomny jejich 3D souřadnice. Tyto segmenty Mol* zobrazuje přerušovanou čarou mezi předchozím a následujícím pozorovaným segmentem.

[17]Vazebné místo označuje část proteinu, kam se váží jiné molekuly (tzv. ligandy). Vznik této vazby často podmíní změnu funkce proteinu. (Wikipedia, 2021a)

[18]Aktivní místo představuje pozici, kde probíhá chemická reakce.

[19]Detailní popis anotací se nachází zde `https://www.uniprot.org/help/sequence_annotation`.

kované stuktury (*Predicted structures*), Mutace (*Variation*), Vlastnosti struktury (*Structural features*)...

Anotace, na které uživatel klikne myší, nazýváme **vybrané** anotace a vyznačujeme je pomocí širšího okraje a tmavšího vybarvení. Vybere-li uživatel anotaci s držením klávesy Shift, nazýváme ji **zvýrazněná** anotace (je zároveň i vybraná). Zvýrazněním anotace se vybarví všechny řádky v místě rozsahu zvýrazněné anotace. Na toto zvýraznění je použita stejná barva jako má zvýrazněná anotace. V případě překryvu zvýraznění se tyto barvy smíchají. Tento princip popisuje Obrázek 1.4.

MolArt 2.0 dokáže zobrazit experimentální i predikovanou kvartérní strukturu a propojuje ji se sekvencí pomocí barevného zvýraznění částí struktury, kde se vyskytují vybrané anotace. Nepozorované segmenty experimentálních struktur se vykreslují šedou barvou a pozorované modrou barvou.

Všechny tyto koncepty lze vidět na Obrázku 1.5.

1.3 Příbuzné projekty

V této sekci představíme několik projektů souvisejících s naší prací. Jedná se převážně o projekty zaměřené k podobnému účelu jako MolArt 2.0 a o projekty, které MolArt 2.0 využívá, nebo z nich vychází.

1.3.1 ProtVista

ProtVista (Watkins a kol., 2017) je nástroj napsaný v jazyce JavaScript určený k vizualizaci sekvence aminokyselin a jejich anotací. Nyní je již nahrazen knihovnou Nightingale a neprobíhá na něm žádný další vývoj. Obrázek 1.6 představuje ukázku tohoto nástroje.

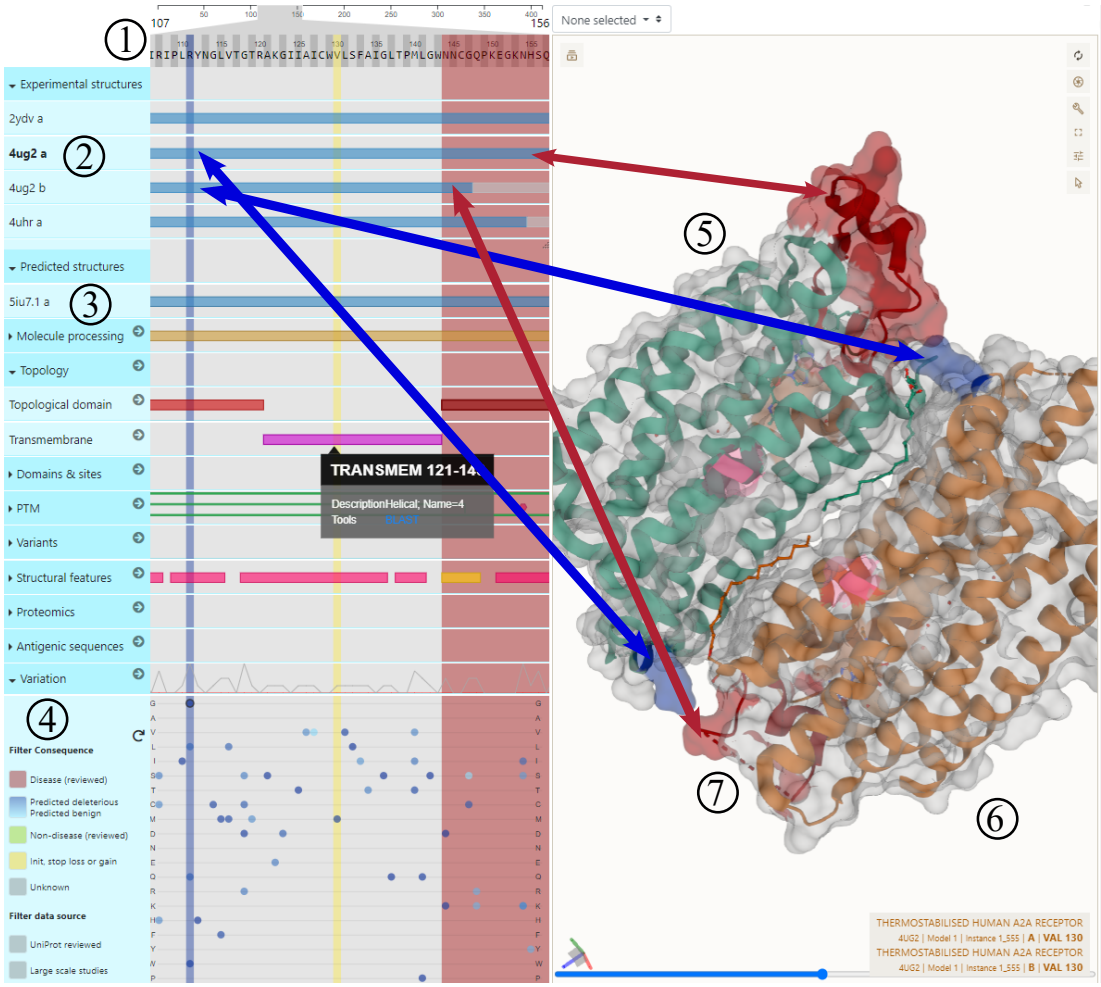
Upravená verze této komponenty byla použita v původním MolArtu.

1.3.2 Nightingale

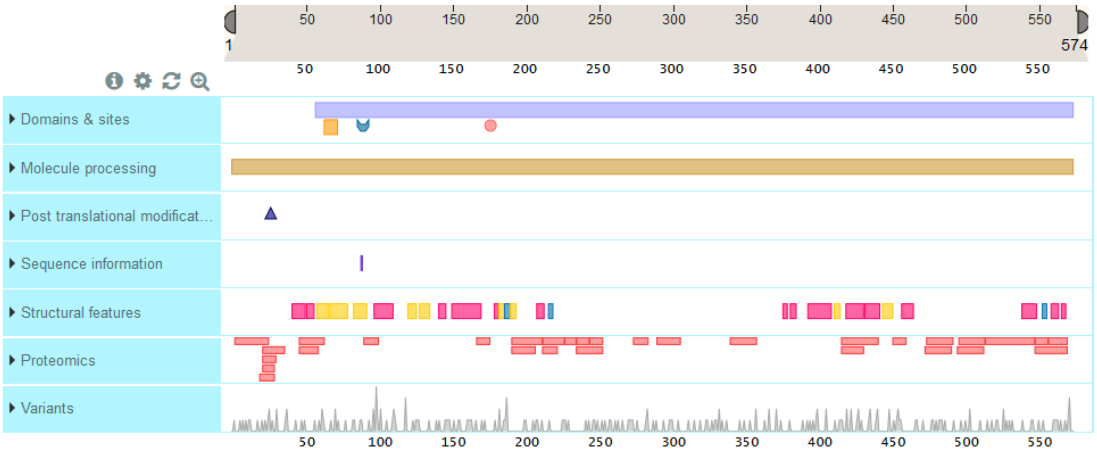
Nightingale (EBI) je knihovna poskytující několik komponent k vizualizaci dat. Na rozdíl od ProtVisty je modulárnější a díky tomu poskytuje větší flexibilitu v možnosti využívání jednotlivých komponent. Nightingale je implementován převážně v JavaScriptu, ale obsahuje také části programované v jazyce TypeScript. V našem projektu využíváme k zobrazení sekvence několik modulů z této knihovny.

1.3.3 LiteMol

LiteMol (Sehnal, 2019) je typescriptový webový plugin určený k vizualizaci makromolekulárních strukturálních dat. Stejně jako ProtVista byl použit v původním MolArtu. V současné době je nahrazen pluginem Mol* a není dále udržován.



Obrázek 1.5: Na obrázku je ukázka nástroje MolArt 2.0. Na levé části se nachází komponenta vizualizující sekvenci s anotacemi, pravá představuje kvartérní strukturu. Čísly **1** až **6** jsou označeny nejdůležitější části MolArtu 2.0. Číslo **1** ukazuje na sekvenci aminokyselin. Číslo **2** představuje vybranou experimentální strukturu, která je zobrazena v pravé části. Všimněme si, že pro tuto kvartérní strukturu (*4ug2*) existuje k dané sekvenci mapování mezi dvěma terciárními strukturami (*a* a *b*). Číslo **3** značí příslušnou predikovanou strukturu a identifikátor templátu, podle kterého byla vytvořena. Predikovaných i experimentálních struktur může být různý počet (i žádné). Číslo **4** označuje kategorii zobrazující mutace. Čísla **5** a **6** představují terciární struktury příslušné k vybranému PDB ID. Zelená část vizualizuje terciární strukturu s ID *a* a oranžová s ID *b*. Číslo **7** značí část struktury, která nebyla v experimentu pozorována. Barevné šipky ilustrují zvýraznění vybraných anotací (zvýrazňují se v obou terciárních strukturách).



Obrázek 1.6: Ukázka nástroje ProtVista. Obrázek byl převzat z ProtVista (EMBL-EBI).

1.3.4 Mol*

Mol* (Sehnal a kol., 2021) je plugin pro zobrazování 3D struktur makromolekul používaný v MolArtu 2.0. Velká nevýhoda práce s touto knihovnou byla, že v době psaní projektu nebyla dokončená její dokumentace. Tento plugin je implementován v jazyce TypeScript.

1.3.5 protvista-uniprot

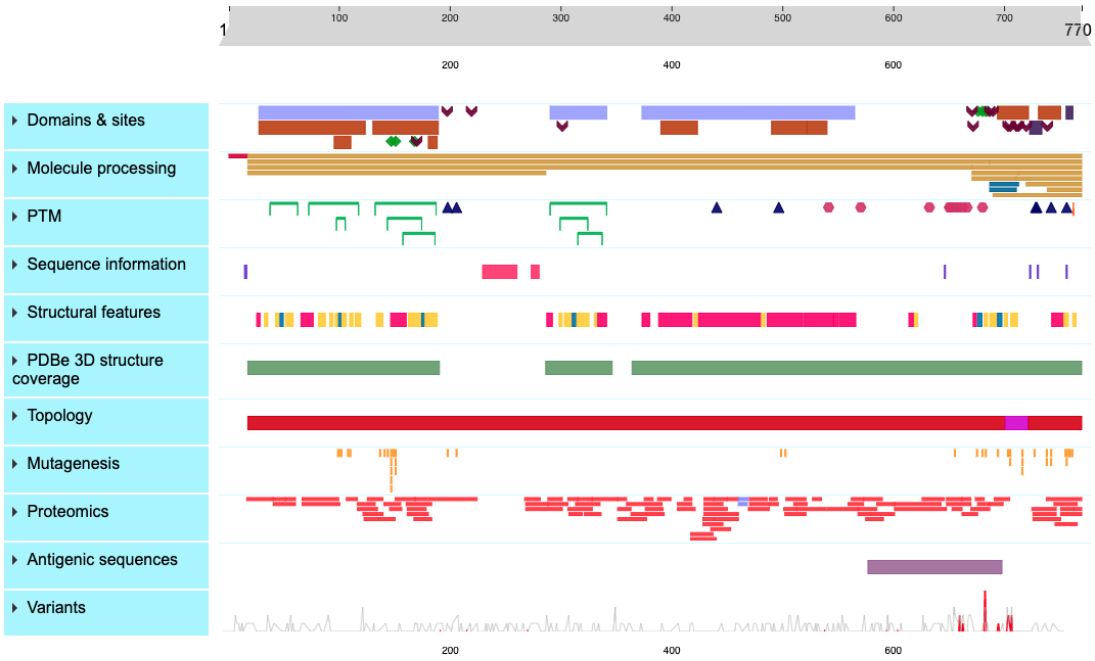
Webová komponenta protvista-uniprot (Watkins) používá knihovnu Nightingale k vizualizaci 3D struktury proteinu (pomocí Mol*) a k vizualizaci informací o proteinové sekvenci. Oproti MolArtu 2.0 však neposkytuje mapování mezi sekvencí a 3D strukturou nebo možnost načtení uživatelských dat. Protvista-uniprot je implementován v jazyce TypeScript. Ukázku protvista-uniprot představuje Obrázek 1.7.

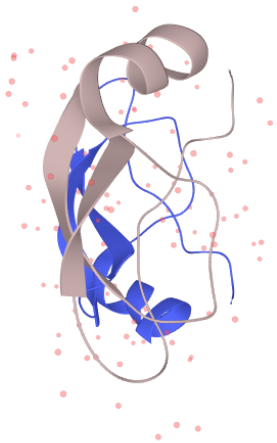
1.3.6 ProtVista PDB

ProtVista PDB (Deshpande) je javascriptový projekt organizace PDBe. Pro vizualizaci sekvence a anotací používá komponenty z knihovny Nightingale. Tento nástroj má následující nedostatky.

- Ke každé kvartérní struktuře zobrazuje pouze jednu terciární strukturu.
- Zvýrazňování anotací není správně zarovnané v případě, že se zobrazuje scrollbar.
- Neumí vícenásobné zvýraznění anotací.
- Špatné vykreslování a vybarvení variant.
- Filtrování variant se neprojevuje v grafu představujícím počet mutací.

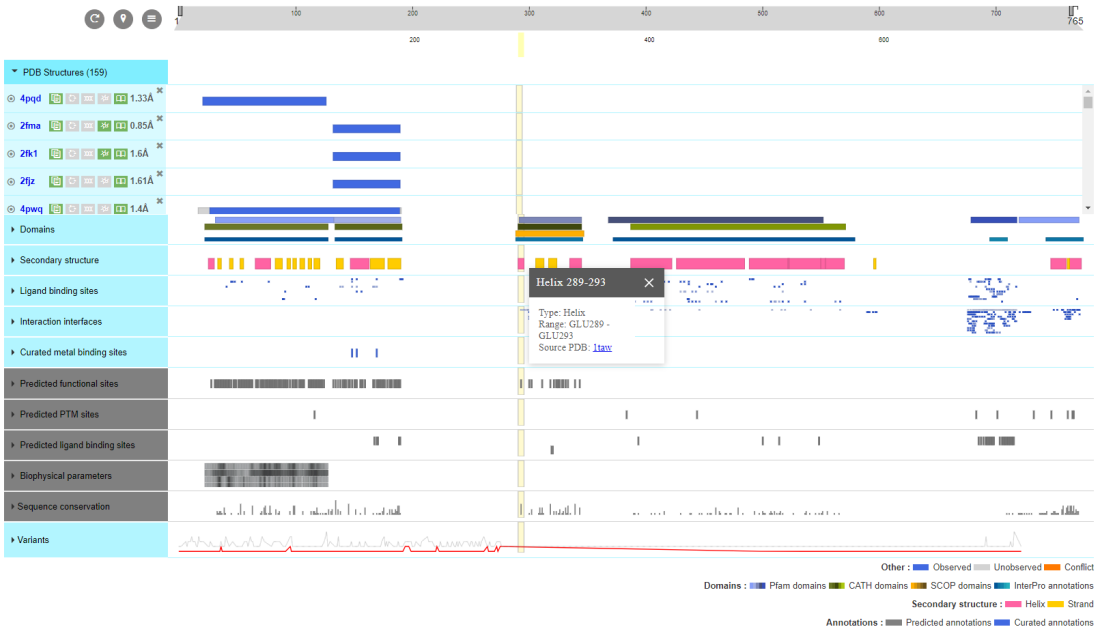
Ukázku ProtVista PDB představuje Obrázek 1.8.





PDB Entry	Method	Resolution	Chain	Positions	Links
1AAP	X-ray	1.50 Å	A/B	287-344	PDBe RCSB PDB PDBj PDBsum
1AMB	NMR		A	672-699	PDBe RCSB PDB PDBj PDBsum
1AMC	NMR		A	672-699	PDBe RCSB PDB PDBj PDBsum
1AML	NMR		A	672-711	PDBe RCSB PDB PDBj PDBsum
1BA4	NMR		A	672-711	PDBe RCSB PDB PDBj PDBsum
1BA6	NMR		A	672-711	PDBe RCSB PDB PDBj

Obrázek 1.7: Ukázka komponenty protvista-uniprot. Obrázek byl převzat z protvista-uniprot (Watkins).



Obrázek 1.8: Ukázka komponenty ProtVista PDB.

1.3.7 MolArt

MolArt (Hoksza a kol., 2018) je javascriptový nástroj umožňující prohlížení proteinové sekvence a anotací společně s 3D strukturou proteinu. Z tohoto projektu vychází naše práce.

2. Programátorská dokumentace

MolArt 2.0 se skládá ze tří částí. První část (`uniprot-nightingale`) používá komponenty z knihovny Nightingale (popsána v sekci 1.3.2) a má na starost vizualizaci proteinové sekvence s anotacemi. Druhá část (`MolstarPlugin`) představuje obal nad pluginem Mol* (popsán v sekci 1.3.4) a zajišťuje zobrazení 3D struktury proteinu. Třetí část (`MolArt`) zabezpečuje propojení mezi `uniprot-nightingalem` a `MolstarPluginem`. V této kapitole popíšeme tyto tři části a představíme způsob testování aplikace.

2.1 uniprot-nightingale

Modul `uniprot-nightingale` zajišťuje vizualizaci proteinové sekvence s anotacemi. V následujících sekcích popíšeme jak získáváme vstupní data. Dále představíme objektový model tohoto modulu a způsob jeho propojení s knihovnou Nightingale.

2.1.1 Vstupní data

`Uniprot-nightingale` získává zobrazovaná data z externích zdrojů nebo od uživatele, tyto dva způsoby lze kombinovat. Data z externích zdrojů se načítají pomocí REST API popsaných v sekci 1.2.2. Data poskytnutá uživatelem se definují pomocí konfigurační struktury, která je popsána v Kódu 2.4.

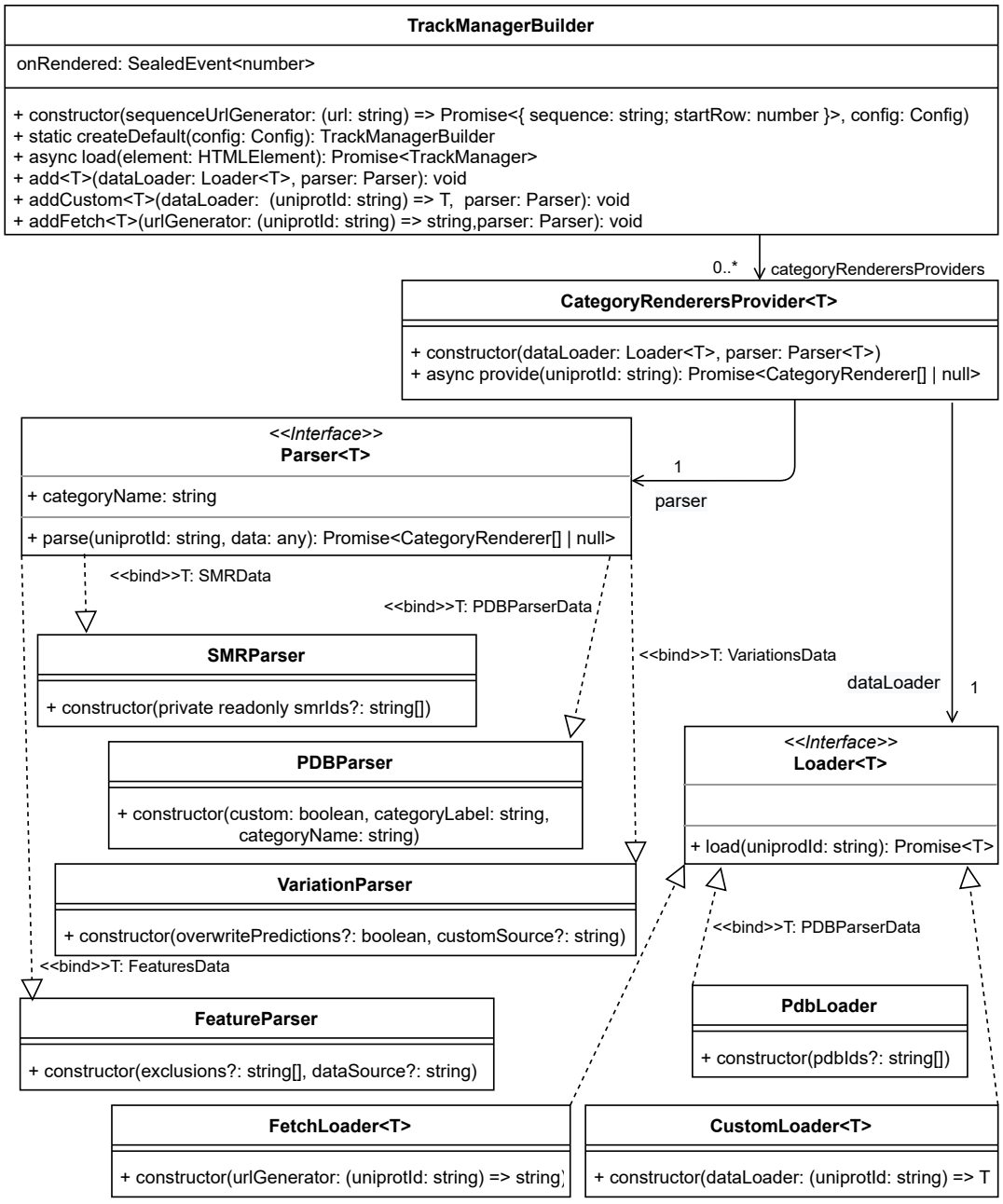
2.1.2 Objektový model

V této sekci popíšeme hlavní třídy, datové typy a vztahy mezi nimi v modulu `uniprot-nightingale`. Vizuální reprezentaci objektového modelu znázorňují obrázky 2.1 a 2.2.

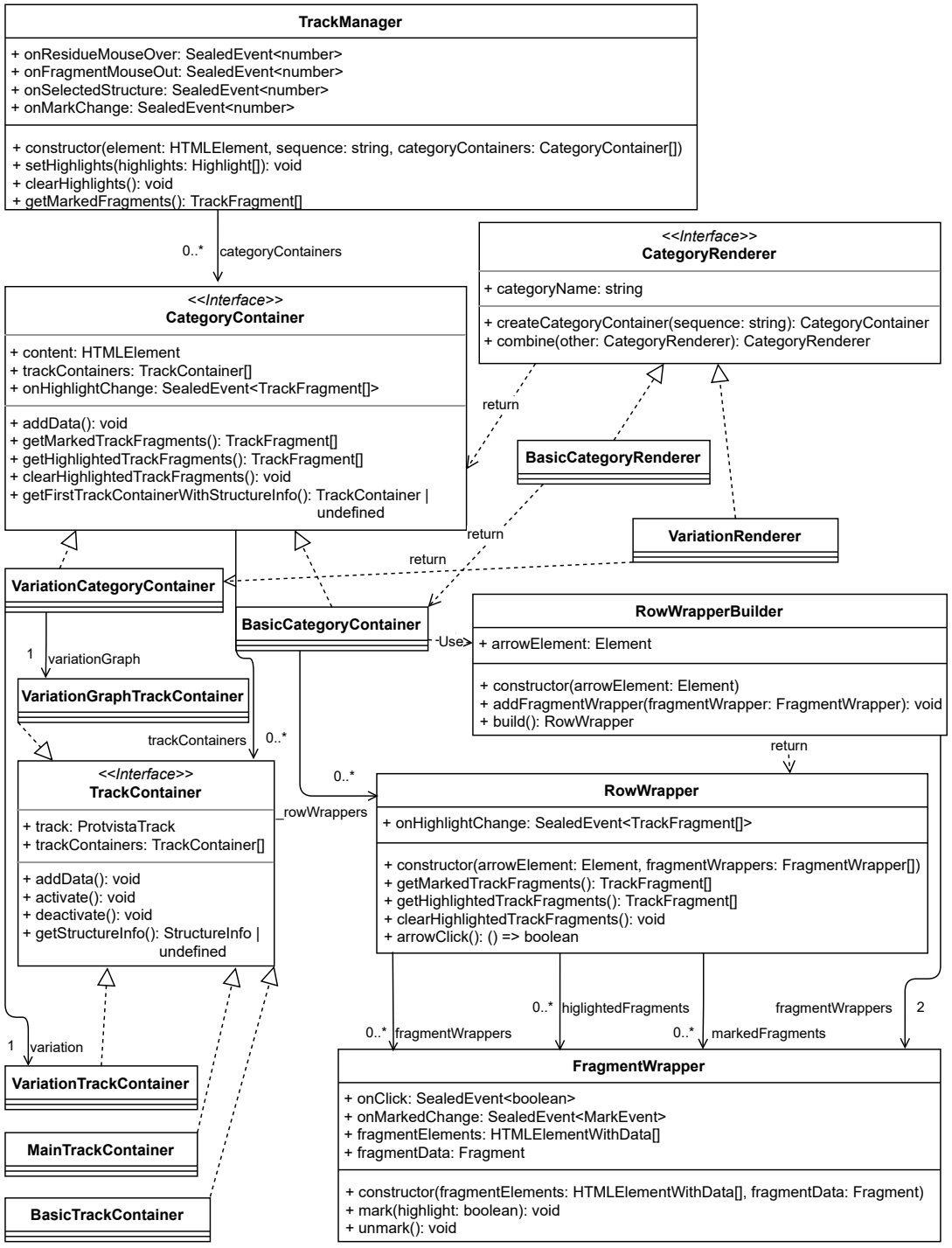
TrackManager

Nejdůležitější třídou, která řídí celou vizualizaci, je `TrackManager`. Tato třída vytváří elementy `ProtvistaNavigation` a `ProtvistaSequence` z knihovny Nightingale sloužící k zobrazování sekvence aminokyselin. Dále `TrackManager` spravuje všechny vykreslované kategorie a vytváří Nightingale komponentu `ProtvistaManager`, do které vloží výše zmíněné elementy (včetně kategorií). Komponenta `ProtvistaManager` koordinuje zvýrazňování (highlight), přibližování a oddalování řádků všech vložených elementů. Třída `TrackManager` pomocí `ProtvistaManageru` nastavuje zvýrazňování řádků. Další z funkcí `TrackManageru` je zobrazování a skrývání popisku anotace (tooltip).

Jedním z problémů, které třída `TrackManager` řeší, je, že pokud kategorie obsahuje mnoho řádků, tak tyto řádky zabírají na stránce příliš velkou část plochy a posunou ostatní kategorie mimo obrazovku. Tento problém jsme vyřešili nastavením maximálního zobrazovaného počtu řádků v kategorii (pro zobrazení zbylých řádků slouží posuvník). Přidáním posuvníku do elementu vykreslujícího tuto kategorii nastal kvůli změně velikosti daného elementu stejný problém, jako



Obrázek 2.1: První část objektového modelu komponenty `uniprot-nightingale`, která zobrazuje třídy podílející se na vytvoření `TrackManagera`.



Obrázek 2.2: Druhá část objektového modelu komponenty `uniprot-nightingale`, která zobrazuje `TrackManagera` a všechny jeho důležité závislosti.

lze vidět na Obrázku 1.8 ukazujícím komponentu ProtVista PDB, kde si můžeme všimnout nesprávného zarovnání žlutého zvýrazňovacího pruhu v kategorii, která obsahuje posuvník. Tuto nepřesnost jsme opravili použitím pluginu Overlay Scrollbars[1], který poskytuje posuvník neovlivňující velikost elementu. `TrackManager` umožňuje přihlásit se k odběru následujících událostí:

- `onResidueMouseOver.on(callback: (residueNumber: number) => void)` – Funkce *callback* se volá při najetí myší na anotaci. Tato událost obsahuje index aminokyseliny v sekvenci dle pozice kurzoru myši.
- `onFragmentMouseOut.on(callback: (data: void) => void)` – Funkce *callback* se volá při opuštění anotace myší.
- `onMarkChange.on(callback: (markedTrackFragments: TrackFragment []) => void)` – Funkce *callback* se volá při změně množiny vybraných anotací (například při kliknutí na anotaci se daná anotace přidá/odebere z množiny). Poskytuje informace o začátku, konci a barvě všech aktuálně vybraných anotací. Datovou strukturu pro reprezentaci těchto informací zobrazuje Kód 2.1.

```
1    type TrackFragment = {
2      //začátek anotace v sekvenci
3      readonly sequenceStart: number;
4      //konec anotace v sekvenci
5      readonly sequenceEnd: number;
6      //barva, kterou je anotace vykreslena
7      readonly color: string;
8    };
9
```

Kód 2.1: Datová struktura popisující informace o vybrané anotaci.

- `onSelectedStructure` se spouští po vybrání nové struktury. Poskytuje informace o struktuře obsahují PDB ID, ID terciární struktury, mapování první úrovně, URL k načtení strukturálních dat, nebo přímo data samotná a také formát strukturálních dat. Datovou strukturu obsahující tyto atributy definuje Kód 2.2.

```
1    //structureInfo obsahuje vždy právě jeden z atributů url a
     ↪ data
2    type StructureInfo = {
3      //PDB ID pro expermentální struktury nebo identifikátor
     ↪templátu pro predikované struktury
4      readonly pdbId: string;
5      //ID vybrané terciární struktury
6      readonly chain: string;
7      //záznam obsahující mapování první urovně pro všechny
     ↪terciární struktury pro dané PDB ID
8      readonly mapping: Mapping;
```

[1]Dokumentace pluginu Overlay Scrollbars se nachází na adrese `https://kingsora.github.io/OverlayScrollbars/#!documentation/options`.

```
9       //URL poskytující strukturální data
10      readonly url?: string;
11      //formát strukturálních dat, v MolArtu 2.0 používáme
    ↪pouze formáty PDB a mmCIF, ale ostatní vyjmenované formá
    ↪ty jsou také podporované
12      readonly format: "mmcif" | "cifCore" | "pdb" | "pdbqt" |
    ↪ "gro" | "xyz" | "mol" | "sdf" | "mol2";
13      //strukturální data
14      readonly data?: string;
15      //udává, zda se mají používat klasické indexy (label), č
    ↪i autorské (auth)
16      //v MolArtu 2.0 používáme auth indexy pro struktury ze
    ↪SMR databáze a label indexy pro struktury z PDB databáze
17      readonly idType: "label" | "auth";
18      //segmenty sekvence, které jsou namapované na pozorované
    ↪ části vybrané terciární struktury či templátu
19      readonly observedIntervals: { start: number; end: number
    ↪ }[];
20      //zdroj struktury
21      readonly source: "PDB" | "SMR" | "USER";
22    }
23    //klíč záznamu je ID terciární struktury
24    type Mapping = Record<string, ChainMapping>;
25
26    //mapování pro terciární strukturu
27    type ChainMapping = {
28      //ID terciární struktury používané ve formátu mmCIF
29      readonly structAsymId: string;
30      //mapování na sekvenci pro různé části struktury
31      readonly fragmentMappings: FragmentMapping[];
32    };
33
34    //mapování části struktury na sekvenci
35    type FragmentMapping = {
36      //ID molekuly používané ve formátu mmCIF
37      readonly entityId?: number;
38      //začátek dané části ve struktuře
39      readonly structureStart: number;
40      //konec dané části ve struktuře
41      readonly structureEnd: number;
42      //začátek dané části v sekvenci
43      readonly sequenceStart: number;
44      //konec dané části v sekvenci
45      readonly sequenceEnd: number;
46    };
47
```

Kód 2.2: Datová struktura popisující informace o vybrané terciární struktuře.

Třída `TrackManager` poskytuje následující API:

- `getMarkedFragments(): TrackFragment[]` vrátí seznam všech aktuálně vybraných anotací. Datovou strukturu použitou pro reprezentaci těchto anotací ukazuje Kód 2.1.
- `setHighlights(highlights: Highlight[])` nastaví uživatelské zvýraznění sekvence dle obdrženého parametru. Datovou strukturu představující zvýraznění v sekvenci zobrazuje Kód 2.3.

```
1   type Highlight = {
2     //začátek zýraznění v sekvenci
3     readonly sequenceStart: number;
4     //konec zýraznění v sekvenci
5     readonly sequenceEnd: number;
6     //barva zvýraznění (není-li barva vyplněná, použije se
    ↪defaultní barva)
7     readonly color?: string;
8   };
9
```

Kód 2.3: Datová struktura popisující zvýraznění sekvence.

- `clearHighlights()` zajistí zrušení uživatelských zvýraznění sekvence.
- getActiveStructureInfo(): StructureInfo | undefined vrací informace o aktuálně vybrané struktuře. Datovou strukturu obsahující tyto atributy definuje Kód 2.2.

TrackManagerBuilder

Třídu `TrackManagerBuilder` používáme k sestavení instance třídy `TrackManager`. V konstruktoru očekává konfiguraci a funkci generující sekvenci na základě identifikátoru sekvence (uniprotId). Formát konfigurace definuje Kód 2.4.

```
1  //nelze zadat současně uniprotId a sequence zároveň
2  //musí být poskytnut alespoň jeden z atributů uniprotId a sequence
3  //zadá-li se sequence musí být poskytnuto také
     ↪sequenceStructureMapping
4  type SequenceConfig = {
5    //identifikátor sekvence v databázi UniProtKB
6    readonly uniprotId?: string;
7    //ID experimentálních kvartérních struktur, které mají být
       ↪zobrazeny
8    //pokud není definován, zobrazí se všechny
9    readonly pdbIds?: string[];
10   //templáty predikovaných kvartérních struktur, které mají být
       ↪zobrazeny
11   //pokud není definován, zobrazí se všechny
12   readonly smrIds?: string[];
13   //definuje pořadí, v jakém se mají zobrazovat kategorie
14   //kategorie uvedené v tomto poli se zobrazují jako první a
       ↪ostatní až za nimi ve výchozím pořadí
```

```
15    readonly categoryOrder?: string[];
16    //udává kategorie, které se nemají zobrazovat
17    //výchozí hodnota je prázdné pole
18    readonly categoryExclusions?: string[];
19    //uživatelem poskytnutá data
20    //výchozí hodnota je prázdné pole
21    readonly customDataSources?: CustomDataSource[];
22    //udává, zda se mají v popisku mutací ukazovat predikce zadané u
        ↪živatelem (true), nebo načtené z externího zdroje (false)
23    //výchozí hodnota je false
24    readonly overwritePredictions?: boolean;
25    //uživatelem zadaná sekvence
26    readonly sequence?: string;
27    //uživatelem zadané mapování
28    readonly sequenceStructureMapping?: PDBParserData;
29  }
30
31  //musí být poskytnuto alespoň jeden z atributů url a data
32  type CustomDataSource = {
33    //název zdroje
34    readonly source: string;
35    //true, pokud se má na konec zadané URL přidat ".json"
36    //výchozí hodnota je false
37    readonly useExtension?: boolean;
38    //URL, ze které se načtou data, přidává se přípona ${uniprotId}
        ↪je-li definována
39    readonly url?: string;
40    //uživatelská data
41    readonly data?: CustomDataSourceData;
42  };
43
44  type CustomDataSourceData = {
45    //odpovídající sekvence
46    readonly sequence: string;
47    //data anotací
48    readonly features: CustomDataSourceFeature[];
49  };
50
51  type CustomDataSourceFeature = Feature | VariantWithCategory;
52
```

Kód 2.4: Datová struktura popisující konfiguraci `TrackManagerBuilderu`. Typy `Feature` a `VariantWithCategory` jsou popsány v příloze A.2.

`TrackManagerBuilder` obsahuje pole objektů `CategoryRenderersProvider`, které se skládají ze dvou atributů. První z nich (`Loader`) má na starosti načítání dat a druhý (`Parser`) je připraví do požadovaného formátu a vytvoří instanci třídy implementující interface `CategoryRenderer`, která produkuje instanci třídy implementující interface `CategoryContainer`. Pomocí získaných `CategoryCon-`

tainerů se sestaví TrackManager.

TrackManagerBuilder umožňuje přihlásit se k odběru následující události:

- onRendered.on(callback: (data: void) => void) – Funkce *callback* se volá po vytvoření TrackManageru.

Třída TrackManagerBuilder poskytuje následující API:

- add<T>(dataLoader: Loader<T>, parser: Parser<T>) přidává objekty typu CategoryRendererProvider do TrackManagerBuilderu, které se podílejí na vytvoření instance typu TrackManager.
- addCustom<T>(dataLoader: (uniprotId: string) => T, parser: Parser<T>) zjednodušuje použití funkce add tím, že není nutné vytvářet objekt typu Loader a lze místo něj použít lambda funkci.
- addFetch<T>(urlGenerator: (uniprotId: string) => string, parser: Parser<T> zjednodušuje použití funkce add v případě jednoduchého načítaní dat ve formátu JSON z externích zdrojů (pomocí javascriptové funkce *fetch*).
- load(element: HTMLElement): Promise<TrackManager> pomocí všech CategoryRenderersProviderů načte a naparsuje data a do daného HTMLElementu vloží TrackManager.

Výchozí sestavení TrackManagerBuilderu se základními CategoryRenderersProvidery je možné přes statickou funkci createDefault(config: SequenceConfig). Tato funkce načítá následující data (část *{uniprotId}* se nahrazuje identifikátorem sekvence):

- **Sekvenci aminokyselin** se načítá z adresy https://www.uniprot.org/uniprot/{uniprotId}.fasta.
- **Seznam predikovaných struktur** se načítá z adresy https://swissmodel.expasy.org/repository/uniprot/{uniprotId}.json?provider=swissmodel
- **Seznam experimentálních struktur** se načítá pomocí třídy PDBLoader, která je podrobně popsaná níže v této sekci.
- **Anotace typu features** se načítá z adresy https://www.ebi.ac.uk/proteins/api/features/{uniprotId}.
- **Anotace kategorie proteomics** se načítá z adresy https://www.ebi.ac.uk/proteins/api/proteomics/{uniprotId}.
- **Anotace kategorie antigen** se načítá z adresy https://www.ebi.ac.uk/proteins/api/antigen/{uniprotId}.
- **Anotace kategorie variation** se načítá z adresy https://www.ebi.ac.uk/proteins/api/variation/{uniprotId}.

Loader

Loader<T> je obecný interface definující třídy načítající data tak, aby je mohla zpracovat třída implementující interface Parser<T>. Předepisuje jednu metodu, která je definovaná jako load(uniprodId: string): Promise<T>.

Třídy implementující interface Loader<T> jsou následující:

- `CustomLoader<T>` v konstruktoru očekává lambda funkci, která vrací data typu `T` a jako parametr dostává identifikátor sekvence. Tuto lambda funkci volá v metodě `load` a její výsledek zabalí do `Promisu`.
- `FetchLoader<T>` v konstruktoru očekává generátor URL, ze které se v metodě `load` načítají data pomocí javascriptové funkce *fetch*.
- `PdbLoader` je specifická třída pro načítání experimentálních struktur. Generický typ `T` nahrazuje typem `PDBParserData`, který je popsán v příloze A.3. V konstuktoru může obdržet nepovinný parametr se seznamem PDB ID struktur, které mají být vykresleny. Pokud tento parametr není zadán, vykreslí se všechny struktury příslušné k danému identifikátoru sekvence.

  V následujících odstavcích se části adres obsahující *{uniprotId}*, respektive *{pdbId}* nahrazují identifikátorem sekvence, respektive PDB ID struktury.

  `PdbLoader` načítá seznam experimentálních struktur (jejich PDB ID) a identifikátory jim příslušných terciárních struktur z `https://www.ebi.ac.uk/pdbe/api/mappings/best_structures/{uniprotId}` a mapování druhé úrovně z `https://www.ebi.ac.uk/pdbe/api/mappings/{uniprotId}`. Dále pro každou kvartérní strukturu načítá pokrytí jejích terciárních struktur pozorovanými segmenty z `https://www.ebi.ac.uk/pdbe/api/pdb/entry/polymer_coverage/{pdbId}`. Informace z těchto zdrojů se agregují do formátu, který je čitelný pro `PdbParser`.

  Alternativní zdroj, který přímo obsahuje seznam terciárních struktur s daty agregovanými do pozorovaných a nepozorovaných úseků, je poskytován přes PDBe graph API (`https://www.ebi.ac.uk/pdbe/graph-api/uniprot/unipdb/{uniprotId}`). Tento zdroj však pro každé PDB ID popisuje pouze jednu terciární strukturu, proto jsme ho v našem projektu nepoužili.

Parser

`Parser<T>` je interface definující třídy, které připravují data z `Loaderů` tak, aby byla lépe zpracovatelná `CategoryRenderery` (`CategoryRenderer` má na starosti vytváření HTML elementů). Obsahuje jednu metodu definovanou jako `parse(uniprotId: string, data: T): Promise<CategoryRenderer[] | null>` a jeden atribut představující název kategorie zpracovávaných dat.

Třídy implementující interface `Loader<T>` jsou následující:

- `PdbParser` zpracovává data experimentálních struktur z PDBe-KB databáze (popsána v sekci 1.2.2) agregovaná pomocí `PdbLoaderu` nebo data struktur definovaných uživatelem. Z těchto dat vytvoří pro každou terciární strukturu pozorované a nepozorované úseky. Transformovaná data předává `BasicCategoryRendereru`. Generický typ `T` nahrazuje typem `PDBParserData`.
- `SMRParser` zpracovává data predikovaných struktur ze SMR databáze (popsána v sekci 1.2.2) a předává je `BasicCategoryRendereru`. V konstruktoru může obdržet nepovinný parametr představující ID templátů, které se mají vykreslovat. Pokud tento parametr není poskytnut, vykreslí se všechny templáty přítomné v daných datech. Generický typ `T` nahrazuje typem `SMRData`.

- `FeatureParser` upravuje data z externích zdrojů (popsaných výše v sekci *TrackManagerBuilder*) nebo data zadaná uživatelem přestavující anotace a předává je `BasicCategoryRendereru`. Generický typ `T` nahrazuje typem `FeaturesData`.
- `VariationParser` transformuje data z externích zdrojů (popsaných výše v sekci *TrackManagerBuilder*) nebo data zadaná uživatelem přestavující varianty (mutace) a předává je `VariationRendereru`. Generický typ `T` nahrazuje typem `VariationsData`.

CategoryRenderer

Interface `CategoryRenderer` předepisuje třídy sestavující HTML elementy příslušné k jedné kategorii anotací. Vytvořené HTML elementy se skládají do `TrackContainerů` (třída `TrackContainer` představuje jeden vykreslovaný řádek) spolu s daty odpovídajícími danému řádku. Z těchto `TrackContainerů` je vyroben `CategoryContainer` (`CategoryContainer` reprezentuje všechny řádky v jedné kategorii).

`CategoryRenderer` obsahuje jeden atribut představující název kategorie a dvě následující metody:

- `createCategoryContainer(sequence: string): CategoryContainer` sestaví HMTL elementy a předá je vytvářenému `CategoryContaineru`.
- `combine(other: CategoryRenderer): CategoryRenderer` zajišťuje kombinaci dat ze dvou `Rendererů`. Tato metoda se používá, pokud obdržíme data stejné kategorie z externích zdrojů a zároveň od uživatele nebo také pokud uživatel uvede více zdrojů s daty stejné kategorie.

Třídy implementující interface `CategoryRenderer` jsou následující:

- `VariationRenderer` vytváří elementy `ProtvistaVariationGraph` a `ProtvistaVariation` z knihovny Nightingale a element `VariationFilter` (upravená Nightingale komponenta `ProtvistaFilter`). První dvě z těchto komponent zajišťují vizualizaci mutací a třetí jejich filtrování. Pokud jsou přítomny uživatelem zadané mutace, tak `VariationRenderer` rozšiřuje základní nastavení `VariationFilteru` o uživatelský zdroj dat a důsledky mutace. Vytvořené elementy umístí do `VariationCategoryContaineru`.
- `BasicCategoryRenderer` se používá na vykreslení většiny řádků (kromě variant), které zobrazují anotace nebo pokrytí struktur. V konstruktoru obdrží seznam názvů řádků a jim příslušných anotací. Z agregovaných dat z tohoto seznamu je vytvořen Nightingale element `ProtvistaTrack`, který označujeme jako **agregovaný řádek**. Dále je pro každou položku ze seznamu vytvořen jeden `ProtvistaTrack` obsahující data dané položky. Z těchto `ProtvistaTracků` jsou sestaveny `TrackContainery`, které jsou vloženy do `BasicCategoryContaineru`.

CategoryContainer

Interface `CategoryContainer` definuje rozhraní tříd, které spravují řádky jedné kategorie.

Interface `CategoryContainer` obsahuje následující atributy a metody:

- `content: HTMLElement` představuje HTML element vykreslující všechny řádky dané kategorie.
- `trackContainers: TrackContainer[]` je seznam obalů všech řádků v dané kategorii.
- `addData()` volá metodu `addData` na `TrackContainerech`.
- `getMarkedTrackFragments(): TrackFragment[]` poskytuje informace o začátku, konci a barvě všech aktuálně vybraných anotací v dané kategorii. Datovou strukturu pro reprezentaci těchto informací zobrazuje Kód 2.1.
- `getHighlightedTrackFragments(): TrackFragment[]` poskytuje informace o začátku, konci a barvě všech aktuálně zvýrazněných anotací v dané kategorii. Datovou strukturu pro reprezentaci těchto informací zobrazuje Kód 2.1.
- `clearHighlightedTrackFragments()` zruší zvýraznění anotací v dané kategorii.
- `getFirstTrackContainerWithStructureInfo(): TrackContainer | undefined` vrací ze seznamu první `TrackContainer`, který při volání funkce *getStructureInfo* nevrací `undefined`. Tato metoda se používá pro nalezení prvního řádku, který obsahuje informace o proteinové struktuře.

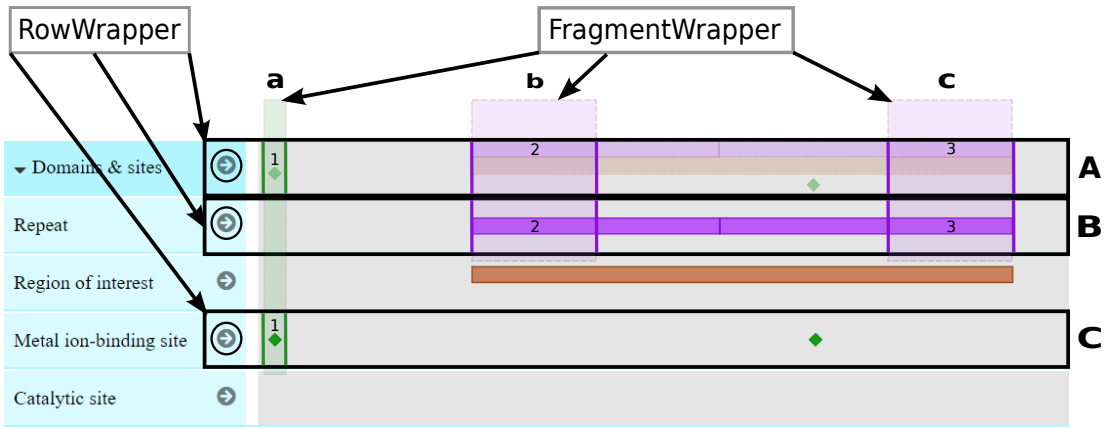
Interface `CategoryContainer` definuje následující událost:

- `onHighlightChange.on(callback: (markedTrackFragments: TrackFragment[]) => void)` – Očekává se, že funkce *callback* se volá při kliknutí na anotaci v dané kategorii. Poskytuje informace o začátku, konci a barvě všech aktuálně vybraných anotací v této kategorii. Datovou strukturu pro reprezentaci těchto informací zobrazuje Kód 2.1.

Třídy implementující interface `CategoryContainer` jsou následující:

- `VariationCategoryContainer` se používá k reprezentaci kategorie zobrazující mutace. Spravuje označování mutací vybraných kliknutím myši a obsahuje dva `TrackContainery - VariationTrackContainer` a `VariationGraphTrackContainer`. První z těchto `TrackContainerů` obsahuje `FixedProtvistaVariation` (upravený Nightingale element `ProtvistaVariation`) zobrazující informace o alternativní aminokyselině a druhý `TrackContainer` obsahuje Nightingale element `ProtvistaVariationGraph`, který vizualizuje počet mutací dané aminokyseliny.
- `BasicCategoryContainer` se používá k reprezentaci všech kategorií kromě mutací. Na rozdíl od `VariationCategoryContaineru` může obsahovat více `TrackContainerů` (řádků).

  V metodě `addData` třída `BasicCategoryContainer` zařídí propojení elementu zobrazujícího anotaci v agregovaném řádku s příslušným elementem v neagregovaném řádku. Propojení je implementováno tím způsobem, že oba tyto elementy jsou umístěny do instance třídy `FragmentWrapper`, která se stará o to, aby se při vybrání anotace označily oba příslušné elementy. Všechny `FragmentWrappery` příslušící k jednomu řádku jsou vloženy do instance třídy `RowWrapper`, která zajišťuje označování anotací pomocí kliknutí na šipku. Tato hierarchie je znázorněna na Obrázku 2.3.



Obrázek 2.3: Vizualizace funkce tříd `FragmentWrapper` a `RowWrapper`. `RowWrapper` **A** představuje agregovaný řádek a obsahuje `FragmentWrappery` **a**, **b**, **c**. `RowWrapper` **B** obsahuje `FragmentWrappery` **b**, **c**. `RowWrapper` **C** obsahuje `FragmentWrapper` **a**. Každý `RowWrapper` má také odkaz na element zobrazující šipku.

TrackContainer

Interface `TrackContainer` definuje třídy představující obal pro Nightingale komponenty ProtvistaTrack.

Interface `TrackContainer` obsahuje následující atributy a metody:

- `track: ProtvistaTrack` je Nightingale komponenta vykreslující daný řádek.
- `addData()` přidává data do Nightingale komponenty. Tato data se musejí do komponenty dodat až po jejím připojení do DOM stromu prohlížeče, protože jinak je vyhozena výjimka.
- `getStructureInfo(): StructureInfo | undefined` vrací `StructureInfo` daného řádku. `StructureInfo` je datová struktura představující informace o struktuře, její definici ukazuje Kód 2.2. Nejedná-li se o řádek zobrazující pokrytí struktury, tak tato metoda vrací `undefined`.
- `activate()` zvýrazní název řádku, pokud metoda `getStructureInfo` nevrací `undefined`.
- `deactivate()` odstraní zvýraznění názvu řádku, pokud metoda `getStructureInfo` nevrací `undefined`.

Interface `TrackContainer` definuje následující události:

- `onLabelClick.on(callback: (structureInfo: StructureInfo) => void)` – Očekává se, že se funkce *callback* volá po kliknutí na název řádku a poskytuje informace o struktuře na daném řádku ve formátu, který definuje Kód 2.2.

Třídy implementující interface `TrackContainer` jsou následující:

- `BasicTrackContainer` je základní implementací `TrackContaineru`, obsahuje jeden `ProtvistaTrack` a metoda `getStructureInfo` vrací výsledek dle parametru v konstruktoru.

- `MainTrackContainer` reprezentuje agregovaný řádek. Obsahuje dva `ProtvistaTrack` elementy. První z nich představuje řádek s anotacemi, druhý neobsahuje žádná data a zobrazuje se při rozbalení neagregovaných řádků. `ProtvistaTrack` bez dat používáme proto, aby nebylo přerušeno zvýraznění při skrytí `ProtvistaTracku` s anotacemi. V této implementaci metoda `getStructureInfo` vždy vrací `undefined`.
- `VariationGraphTrackContainer` reprezentuje řádek zobrazující graf počtu mutací aminokyselin. Obsahuje Nightingale komponentu `ProtvistaVariationGraph` (dědí od `ProtvistaTracku`). V této implementaci metoda `getStructureInfo` vždy vrací `undefined`.
- `VariationTrackContainer` reprezentuje řádek zobrazující informace o alternativní aminokyselině. Obsahuje `FixedProtvistaVariation` (upravený Nightingale element `ProtvistaVariation` dědící od `ProtvistaTracku`). V této implementaci metoda `getStructureInfo` vždy vrací `undefined`.

2.1.3 Integrace s Nightingale

V modulu `uniprot-nightingale` používáme k vizualizaci sekvence a anotací komponenty z knihovny Nightingale (již představena v sekci 1.3.2), které popisujeme v první sekci. Během vývoje projektu MolArt 2.0 jsme narazili na několik problémů s knihovnou Nightingale. Tyto problémy a jejich řešení si ukážeme ve druhé sekci. Ve třetí sekci popíšeme způsob, jakým jsme propojili javascriptovou knihovnu s naším typescriptovým projektem.

Nightingale komponenty

Níže představíme komponenty z knihovny Nightingale, které používáme v modulu `uniprot-nightingale`. Většinu z těchto komponent ukazuje Obrázek 2.4.

- **ProtvistaManager (protvista-manager)** – Každá Nightingale komponenta se automaticky registruje k nejbližšímu `ProtvistaManageru`, který odposlouchává jejich události a na základě nich nastavuje Nightingale komponentám atributy (například zvýraznění části sekvence, délku, začátek zobrazení a konec zobrazení).
- **ProtvistaNavigation (protvista-navigation)** – `ProtvistaNavigation` je komponenta určená k nastavování rozsahu zobrazovaného úseku sekvence. Při změně rozešle událost obsahující informace o novém začátku a konci zobrazení, které `ProtvistaManager` nastaví všem komponentám.
- **ProtvistaSequence (protvista-sequence)** – `ProtvistaSequence` vizualizuje sekvenci aminokyselin, která ji je nastavena jako atribut. Kódy aminokyselin se objevují jen pokud je sekvence dostatečně přiblížená, jinak se zobrazí pouze indexy aminokyselin dle nastavení atributu *numberofticks*.
- **ProtvistaTrack (protvista-track)** – `ProtvistaTrack` je základní komponenta pro vizualizaci anotací. Zobrazované anotace se nastavují pomocí atributu *data*. V každé anotaci musí být nastaven její začátek a konec, nepovinnými parametry jsou tvar a barva.

Obrázek 2.4: Ukázka Nightingale komponent. Všechny vyznačené komponenty jsou v DOM stromě prohlížeče potomci `ProtvistaManagera`.

`ProtvistaTrack` poskytuje dva typy zobrazení (*non-overlapping* a *overlapping*), které se nastavují atributem *layout*. *Overlapping* zobrazení ukáže všechny anotace na jednom řádku. *Non-overlapping* zobrazení vykreslí anotace tak, aby se nepřekrývaly.

- **ProtvistaTooltip (protvista-tooltip)** – `ProtvistaTooltip` zobrazuje popis anotace.
- **ProtvistaFilter (protvista-filter)** – `ProtvistaFilter` je komponenta určená k filtrování dat. Nastavení komponenty, jejíž data chceme filtrovat, probíhá pomocí atributu *for*. Do atributu *filters* se vkládá pole typů filtrů. `ProtvistaFilter` používáme k filtrování dat v komponentách `ProtvistaVariationGraph` a `ProtvistaVariation`.
- **ProtvistaVariationGraph (protvista-variation-graph)** – `ProtvistaVariationGraph` vizualizuje pomocí grafu počet mutací příslušné aminokyseliny. Červeným grafem zobrazuje mutace způsobující nemoc a šedým grafem ostatní mutace. Rozšiřuje komponentu `ProtvistaTrack`.
- **ProtvistaVariation (protvista-variation)** – `ProtvistaVariation` pomocí matice zobrazuje alternativní aminokyselinu dané mutace. Rozšiřuje komponentu `ProtvistaTrack`.

Naše úpravy knihovny Nightingale

K zajištění správné funkcionality jsme provedli v knihovně Nightingale několik následujících změn:

- Přidání možnosti nastavit barvu zvýrazňované části sekvence. Této změny bylo docíleno úpravami Nightingale tříd `Region` a `TrackHighlighter`.
- Přidání možnosti nastavit hustotu zobrazovaných indexů aminokyselin v `ProtvistaSequence`. Této změny bylo docíleno přidáním nového atributu *numberofticks* do komponenty `ProtvistaSequence`.
- Oprava vyhazování výjimky při používání *non-overlapping* zobrazení a velkém počtu anotací, jejichž rozsahy se překrývají. Oprava spočívala v rozšíření třídy `NonOverlappingLayout` o omezení nastavení výšky elementu minimální kladnou hodnotou. Za tímto účelem jsme vytvořili třídy `LimitedTrack` a `LimitedNonOverlappingLayout`.
- Oprava vyhazování výjimky při přibližování dvojklikem. Tato chyba nastávala v některých případech při dvojkliku na element `ProtvistaTrack`. Pravděpodobná příčina je špatné vyhodnocení této události knihovnou D3, kterou Nightingale používá k vykreslování a správě HTML elementů. Jelikož přesný důvod vzniku výjimky se nám nepodařilo zjistit, zakázali jsme na Nightingale elementech `ProtvistaTrack`, `ProtvistaVariationGraph` a `ProtvistaVariation` přibližování pomocí dvojkliku. Tyto úpravy jsou implementovány v třídách `FixedProtvistaVariation`, `FixedVariationGraph` a `LimitedTrack`.
- Doplnění `ProtvistaTooltipu` o ikonu sloužící k zavření popisku. K tomuto účelu jsme vytvořili třídu `CloseableTooltip`, která přepisuje `ProtvistaTooltip` metodu `render`.

- Umožnění použití `ProtvistaFilteru` na filtrování dat ve více komponentách zároveň. Za tímto účelem jsme vytvořili třídu `VariationFilter`, které se nastavují filtrované komponenty pomocí atributu *multifor*. V této třídě jsou také přepsány `ProtvistaFilter` metody `getCheckBox` a `styles` tak, aby se HTML element správně vykresloval při používání typu filteru s více popisky.

  Abychom mohli `VariationFilter` použít na komponentu `ProtvistaVariationGraph`, bylo nutné změnit implementaci metody `_applyFilters`, aby přepočítávala počty mutací dle přefiltrovaných dat. Tuto úpravu zajišťuje třída `FixedVariationGraph`.

Umístění tříd implementujících tyto změny je popsáno v příloze A.1.

TypeScript

Abychom se vyhnuli častému používání univerzálního typu `any` při práci s převážně javacriptovou knihovnou Nightingale v našem typescriptovém projektu MolArt 2.0, vytvořili jsme pro Nightingale takzvaný declaration file. Declaration file je soubor s příponou d.ts, který popisuje typy pro projekt napsaný v JavaScriptu. V tomto souboru jsme zadefinovali námi využívané typy a třídy a jejich parametry a metody, což nám umožnilo využít výhod typovaného jazyka TypeScript. Tyto typy jsou uložené v samostatném modulu `nightingale-types`.

2.2 MolstarPlugin

`MolstarPlugin` je třída, která spravuje vizualizaci 3D struktury proteinu pomocí pluginu Mol* (uveden v sekci 1.3.4). Tato třída implementuje interface `StructureViewer<StructureConfig, Residue>` (popsáno v sekci 2.3.1), kde generický typ `StructureConfig` nahrazuje typem `MolstarPluginConfig` definovaným Kódem 2.5 a generický typ `Residue` nahrazuje typem `MolstarResidue` definovaným Kódem 2.6. V konstruktoru očekává HTML element, do kterého se vykreslí Mol* komponenta.

```
1  type MolstarPluginConfig = {
2    //nastevení uživatelského zvýraznění
3    extraHighlights: {
4      //název zvýraznění
5      label: string;
6      //nastavenení barvy, typu a velikosti zvýraznění na základě
     ↪Mol* parametrů
7      props: StructureRepresentationBuiltInProps;
8      residue?: {
9        //index prvního zvýrazněného rezidua
10       //pro predikované struktury používáme index zadaný autorem,
     ↪pro experimentální klasický index
11       residueNumFrom: number;
12       //index posledního zvýrazněného rezidua
13       //pro predikované struktury používáme index zadaný autorem,
     ↪pro experimentální klasický index
```

```
      residueNumTo: number;
    };
    //názvy zvýrazněných terciárních struktur
    //pro predikované struktury používáme název zadaný autorem,
    ↪pro experimentální klasický název
    chain?: string[];
    //názvy zvýrazněných atomů
    //pro predikované struktury používáme název zadaný autorem,
    ↪pro experimentální klasický název
    atom?: string[];
  }[];
};
```

Kód 2.5: Datová struktura popisující konfiguraci třídy `MolstarPlugin`.

```
type MolstarResidue = {
  //název aminokyseliny poskytnutý autorem
  authName: string;
  //pozice aminokyseliny v sekvenci poskytnutá autorem
  authSeqNumber: number;
  chain: {
    //identifikátor terciární struktury
    asymId: string;
    //identifikátor terciární struktury poskytnutý autorem
    authAsymId: string;
    entity: {
      //unikátní ID mmcif záznamu popisující molekulární entitu
      entityId: string;
      //Mol* index molekulární entity
      index: number;
    };
    //Mol* index terciární struktury
    index: number;
  };
  //Mol* index aminokyseliny
  index: number;
  //PDB insertion code
  insCode: string;
  //true pokud je aminokyselina mikroheterogenní
  isHet: boolean;
  //název aminokyseliny
  name: string;
  //pozice aminokyseliny v sekvenci
  seqNumber: number;
};
```

Kód 2.6: Datová struktura popisující reziduum aminokyseliny, která je vytvářena `MolstarPluginem`.

Vypínání a zapínání zobrazení zvýraznění, které uživatel zadefinoval v parametru *extraHiglights* v konfiguraci, je možné pomocí rozbalovacího seznamu. Pro zpřístupnění funkce výběru více položek ze seznamu zároveň jsme použili komponentu, kterou poskytuje plugin s názvem Bootstrap Multiselect[2].

2.3 MolArt

Třída `MolArt<StructureConfig, Residue>` se stará o propojení modulu `uniprot-nightingale` a `MolstarPluginu`. Její hlavní funkcí je synchronizace zvýrazňování v obou částech. V konstruktoru očekává objekt, který zajišťuje funkci nástroje pro zobrazování 3D struktury a implementuje interface `StructureViewer`. Jako druhý parametr konstruktoru očekává HMTL element, do kterého bude vložen modul `uniprot-nightingale` po zavolání funkce `loadConfig`.

V této sekci popíšeme předpokládané chování interface `StructureViewer` a veřejné rozhraní třídy `MolArt`.

2.3.1 StructureViewer

Aby bylo možné snadno zaměnit nástroj používaný k vizualizaci struktury proteinu, zavedli jsme interface `StructureViewer<StructureConfig, Residue>` definující API, které musí splňovat třídy zajišťující zobrazování struktury.

Interface `StructureViewer` obsahuje následující metody:

- `load(structureInfo: StructureInfo, config: StructureConfig): Promise<void>` se volá při žádosti o načtení nové struktury. Parametr *structureInfo* reprezentuje informace o struktuře, jeho formát definuje Kód 2.2. Parametr *config* představuje konfiguraci a je určen generickým typem interface, z důvodu, aby si každý nástroj k vizualizaci struktury mohl zadefinovat vlastní formát konfigurace.
- `overpaintFragments(fragments: TrackFragment[])` se volá při změně množiny vybraných anotací. Parametr *fragments* obsahuje informace o začátku, konci a barvě všech vybraných anotací. Datovou strukturu pro reprezentaci těchto informací zobrazuje Kód 2.1.
- `highlightMouseOverResidue(resNum?: number)` se volá při najetí na anotaci nebo při jejím opuštění. Parametr *resNum* představuje index nejbližší aminokyseliny v sekvenci dle kurzoru myši, pokud se jedná o událost opuštění anotace myší, tak není definován.
- `isLoaded(): boolean` by měla vracet informaci, zda načtena nějaká struktura. Výsledek této funkce se používá na API MolArtu 2.0.
- `highlight(resNum: number, chain?: string)` slouží k nastavení uživatelského zvýraznění rezidua ve struktuře. Parametr *resNum* představuje index rezidua v sekvenci a nepovinný parametr *chain* značí ID terciární struktury. Pokud není parametr *chain* definovaný, jsou zvýrazněny příslušná rezidua ve všech vykreslených terciárních strukturách.

[2]Dokumentace pluginu Bootstrap Multiselect se nachází na adrese `https://davidstutz.github.io/bootstrap-multiselect/`.

- `unhighlight()` slouží ke zrušení uživatelského zvýraznění rezidua ve struktuře.
- `focus(resNum: number, chain?: string, radius?: number)` slouží k přiblížení pohledu na reziduum dané parametrem *resNum*. Nepovinný parametr *chain* představuje ID terciární struktury a nepovinný parametr *radius* udává míru přiblížení.
- `getOuterElement(): HTMLElement` by měla vracet HTML element, ve kterém se nástroj k vizualizaci struktury nachází.
- `handleResize()` se volá při změně velikosti elementu, který je vracen metodou `getOuterElement`.

Interface `StructureViewer` definuje následující události:

- `onLoaded.on(callback: (data: void) => void)` – Očekává se, že se funkce *callback* volá po načtení struktury.
- `onHover.on(callback: (residue: Residue | undefined) => void)` – Očekává se, že se funkce *callback* volá po najetí myší na reziduum aminokyseliny a poskytuje informace o daném reziduu nebo po opuštění rezidua myší – v takovém případě vrací `undefined`. Datový typ `Residue` je určen generickým typem interface.
- `onHighlightChange.on(callback: (highlights: Highlight[]) => void)` – Očekává se, že se funkce *callback* volá při zvýraznění části struktury způsobené najetím myší na reziduum. Poskytuje informace o zvýraznění ve formátu, který podporuje modul `uniprot-nightingale`. Datovou strukturu pro reprezentaci těchto informací zobrazuje Kód 2.3.

2.3.2 Veřejné rozhraní

Třída `MolArt` poskytuje následující API:

- `loadConfig(config: Config<StructureConfig>): Promise<void>` na základě konfigurace zadané v parametru vykoná všechny potřebné kroky k tomu, aby zobrazil `uniprot-nightingale` modul a definovaný `StructureViewer`. V této funkci se vytvoří objekt typu `TrackManager`, který spravuje část aplikace vykreslující sekvenci a její anotace. Formát konfigurace zobrazuje Kód 2.7.

```
1  type Config<StructureConfig> = {
2    structure: StructureConfig;
3    sequence: SequenceConfig;
4  };
```

Kód 2.7: Datová struktura popisující konfiguraci třídy `MolArt`. Artibut *structure* představuje konfiguraci nástroje, který zobrazuje 3D strukturu a jeho typ je definovaný generickým typem `StructureVieweru`. Příklad struktury konfigurace pro `MolstarPlugin` ukazuje Kód 2.5. Artibut *sequence* udává konfiguraci modulu `uniprot-nightingale` a jeho typ definuje Kód 2.4.

- `isStructureLoaded(): boolean` vrací výsledek volání funkce `isLoaded` na objektu typu `StructureViewer` poskytnutého v konstruktoru.
- `highlightInStructure(resNum: number, chain?: string)` volá metodu `highlight` na objektu typu `StructureViewer` poskytnutého v konstruktoru se zadanými parametry.
- `unhighlightInStructure()` volá metodu `unhighlight` na objektu typu `StructureViewer` poskytnutého v konstruktoru.
- `unhighlightInSequence()` zruší uživatelské zvýraznění v sekvenci.
- `highlightInSequence(higlight: Highlight)` nastaví uživatelské zvýraznění v sekvenci. Formát datové struktury `Highlight` definuje Kód 2.3.
- `focusInStructure(resNum: number, radius = 0, chain?: string)` volá metodu `focus` na objektu typu `StructureViewer` poskytnutého v konstruktoru se zadanými parametry.
- `getStructureController(): StructureViewer<StructureConfig, Residue>` vrací objekt typu `StructureViewer` poskytnutý v konstruktoru.
- `getSequenceController(): TrackManager | undefined` vrací aktuální `TrackManager`.
- `getSequenceStructureRange(): Interval[]` vrací segmenty sekvence, které jsou namapované na pozorované části aktuální terciární struktury. Datová struktura `Interval` obsahuje atributy *start* a *end*, které značí začátek a konec segmentu.

Třída `MolArt` umožňuje přihlásit se k odběru následujících událostí:

- `onSequenceMouseOn.on(callback: (residueNumber: number) => void)` – Funkce *callback* se volá poté, co objekt typu `TrackManager` vytvořený ve funkci `loadConfig` spustí událost `onResidueMouseOver` a přeposílá index aminokyseliny obdržený z této události.
- `onSequenceMouseOff.on(callback: (data: void) => void)` – Funkce *callback* se volá poté, co objekt typu `TrackManager` vytvořený ve funkci `loadConfig` spustí událost `onFragmentMouseOut`.
- `onStructureMouseOff.on(callback: (data: void) => void)` – Funkce *callback* se volá poté, co objekt typu `StructureViewer` zadaný v konstruktoru spustí událost `onHover` s hodnotou parametru `undefined`.
- `onStructureMouseOn.on(callback: (residue: Residue) => void)` – Funkce *callback* se volá poté, co objekt typu `StructureViewer` zadaný v konstruktoru spustí událost `onHover` s definovaným parametrem typu `Residue`. Tento parametr je následně přeposlán do funkce *callback*. Datový typ `Residue` je určený generickým parametrem `StructureVieweru`.
- `onStructureLoaded.on(callback: (data: void) => void)` – Funkce *callback* se volá poté, co objekt typu `StructureViewer` zadaný v konstruktoru spustí událost `onLoaded`.
- `onSequenceViewerReady.on(callback: (data: void) => void)` – Funkce *callback* se volá poté, co objekt typu `TrackManager` vytvořený ve funkci `loadConfig` spustí událost `onRendered`.

2.4 Kompilace a debugování

Moduly `uniprot-nightingale` a `MolArt` (společně s `MolstarPluginem`) byly publikovány do registru npm[3], což je správce open-source software ve formě balíčků. Balíček obsahující moduly `MolArt` a `MolstarPlugin` je nazván `molart2`. Odkazy na námi zveřejněné balíčky se nachází v příloze A.5. Npm také poskytuje stejnojmenného klienta pro příkazovou řádku, kterého používáme k instalaci všech balíčku (příkaz `npm i`), které náš projekt využívá, a ke spouštění skriptů pro kompilaci či nastartování programu.

Pro spuštění vývojové verze aplikace lze použít příkaz `npm run start`, jenž spustí testovací server, na který se dá připojit debugovací nástroj (v našem případě Visual Studio Code). Pro projekt `uniprot-nightingale` běží tento server na adrese `http://localhost:1340/` a pro projekt `molart2` je spuštěn na adrese `http://localhost:1341/`. Pro vytvoření produkční verze používáme příkaz `npm run build`, který vloží zkompilované soubory do adresáře dist. Běžící aplikaci si lze prohlédnout na adresách `https://scheuerv.github.io/molart/` a `https://scheuerv.github.io/uniprot-nightingale/`.

K vytvoření produkčního balíčku a spuštění vývojové verze aplikace používáme program Webpack[4] s pluginem ts-loader (zajišťuje překlad z JavaScriptu do TypeScriptu).

Pro publikování balíčku do npm kompilujeme projekt pomocí příkazu `tsc` přičemž dodržujeme sémantické verzování. Příkaz `tsc` přeloží soubory do jazyka JavaScript a ke každému vytvoří deklarační soubor popisující datové typy, následně výstup vloží do adresáře lib.

2.5 Unit testy

Pro testování aplikace jsme použili typescriptovou verzi testovacího frameworku Jest s názvem ts-jest[5]. Testy se pouští příkazem `npm run test`, který následně spustí program jest.

Testy je pokrytá většina funkcí a tříd, které nepracují s DOM objekty. Umístění souborů s testy je popsáno v příloze A.1.

[3]Dokumentace a více informací k npm se nachází na adrese `https://docs.npmjs.com/`.

[4]Bližší informace o Webpacku jsou dostupné na adrese `https://webpack.js.org/`.

[5]Dokumentace ts-jest se nachází na adrese `https://kulshekhar.github.io/ts-jest/docs/`.

3. Uživatelská dokumentace

Tato kapitola ukazuje používání základních funkcí MolArtu 2.0. Jakým způsobem se plugin používá, jeho konfigurace a veřejné rozhraní je popsáno v kapitole 2 (Programátorská dokumentace). Také se zde nezabýváme ovládáním pluginu Mol*, které je popsáno v uživatelské dokumentaci Mol*[1].

3.1 Přepínání struktur

Načtení nové struktury do pluginu Mol* je možné dvěma způsoby – kliknutím na název řádku se strukturou nebo modrou část obdélníku reprezentujícího danou strukturu (znázorněno na Obrázku 3.1 pod čísly **1a** a **1b**). Při načítání predikované struktury pomocí kliknutí na název řádku (zde představuje ID templátu) se vykreslí náhodná struktura přítomná na daném řádku. Název s aktuálně načtenou strukturou je zobrazen tučným písmem a segment sekvence namapovaný na strukturu je zvýrazněn šedým pruhem přes všechny řádky.

3.2 Zvýrazňování v uniprot-nightingale

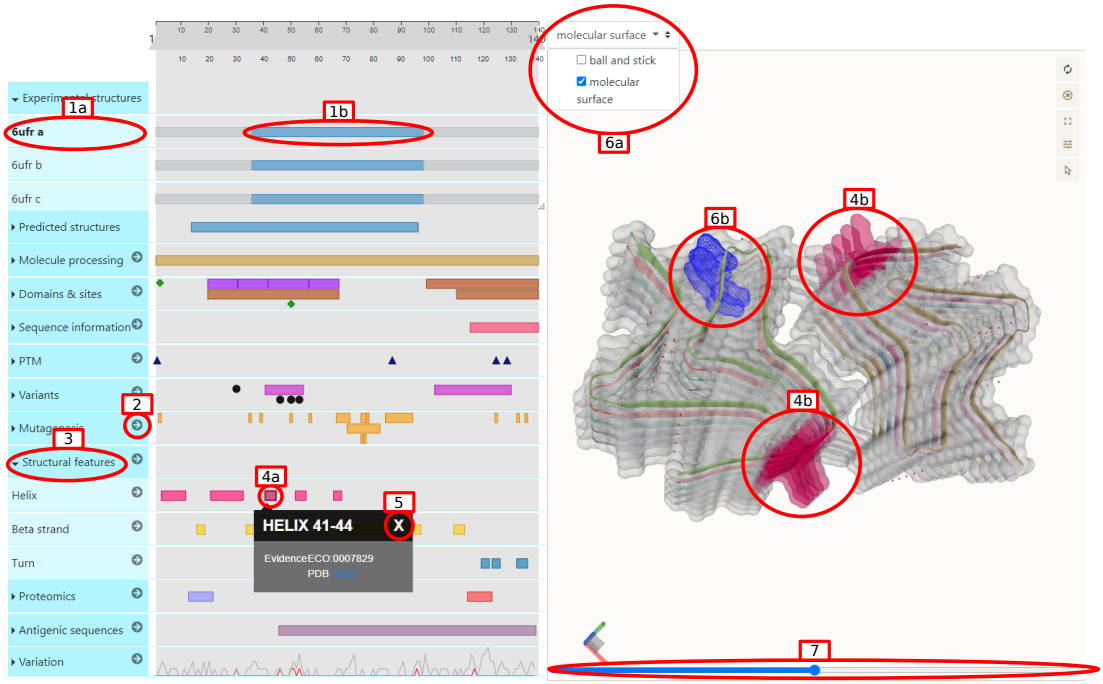
Každou zobrazenou anotaci v MolArtu 2.0 lze zvýraznit. Po spuštění MolArt 2.0 se všechny anotace z jedné kategorie zobrazují na agregovaných řádcích. Pro zobrazení neagregovaných řádků, které obsahují anotace jednoho typu, klikneme na název kategorie (rozbalená kategorie je znázorněna na Obrázku 3.1 pod číslem **3**).

Zvýraznění anotací a jim odpovídajícím částem struktury je možné kliknutím na danou anotaci. Zvýraznění anotace ukazuje Obrázek 3.1 pod čísly **4a** a **4b**. Při kliknutí s držením klávesy Shift je rozsah anotace zvýrazněn pruhem přes všechny řádky. Zvýraznění probíhá stejnou barvou jakou má daná anotace (v sekvenci i ve struktuře). Při překrývání intervalů zvýrazněných anotaci se jejich barvy smíchají. Kliknutím na šipku vedle názvu řádku ze zvýrazní všechny anotace přítomné na daném řádku. Šipku lze vidět na Obrázku 3.1 pod číslem **2**. Zvýraznění se ruší opětovným kliknutím na anotaci nebo na šipku. Pokud chceme zrušit pouze zvýrazněné pruhy, klikneme kamkoliv mimo anotaci v rámci řádků, které zobrazují anotace nebo pokrytí struktury.

Šipka příslušící řádku s mutacemi má specifické chování. Kliknutím na ni se vybarví namapovaná část struktury odstíny šedé. Čím tmavší odstín tím více mutací pro danou aminokyselinu existuje.

Najetím myší na anotaci se pomocí žlutého pruhu zvýrazní aminokyselina odpovídající pozici kurzoru myši (odpovídající aminokyselina ze zvýrazní také v pluginu Mol*).

[1] Uživatelská dokumentace Mol* se nachází na `https://molstar.org/viewer-docs/`.



Obrázek 3.1: Ukázka uživatelského rozhraní.

3.3 Zvýrazňování v Mol*

V pluginu Mol* probíhá zvýrazňování přes klikání na anotace nebo najetím myší na strukturu (odpovídající aminokyselina se zvýrazní také v části zobrazující sekvenci s anotacemi).

Zvýrazňovat pomocí klikání na anotace lze pouze ty části struktury, které jsou namapované na sekvenci. Nenamapované úseky vykresluje Mol* šedě a části struktury, které nebyly pozorovány v experimentu, zobrazuje přerušovanou čarou.

Pokud jsou v konfiguraci MolArtu 2.0 zadána nějaká uživatelská zvýraznění, lze je zapínat a vypínat pomocí rozbalovacího seznamu umístěného nad levým horním rohem Mol* pluginu. Toto je znázorněno na Obrázku 3.1 pod čísly **6a** a **6b**.

Průhlednost povrchu molekuly se nastavuje pomocí posuvníku v dolní části Mol* pluginu. Posuvník je ukázán na Obrázku 3.1 pod číslem **7**.

3.4 Filtrování mutací

Zobrazované mutace lze filtrovat pomocí filtrů vlevo od řádku, který zobrazuje mutace. Jsou přítomny dva druhy filtrů – podle důsledku mutace a podle zdroje dat. Pokud jsou v konfiguraci MolArtu 2.0 zadány mutace uživatelem, tak se tyto typy filtrů mohou rozšířit o další uživatelské filtry. Kliknutím na název nebo ikonku filtru se daný filtr aktivuje (lze aktivovat i více filtrů najednou, to můžeme vidět na Obrázku 3.2, kde jsou aktivní filtry značeny červeným obdélníkem), opětovným kliknutím se filtr deaktivuje. K návratu filtrů do původního nastavení, kde se zobrazují všechny mutace, lze použít ikonku šipky. Šipku lze vidět na Obrázku 3.2, kde je zvýrazněna červeným kolečkem. Při aktivaci/deaktivaci filtru

Obrázek 3.2: Ukázka uživatelského rozhraní filtru mutací.

či při kliknutí na šipku se zruší zvýraznění všech mutací.

3.5 Popis anotace

Popis anotace se zobrazuje najetím na anotaci myší nebo kliknutím na ni. Při zobrazení popisu bez kliknutí popis zmizí po opuštění anotace myší. Při zobrazení popisu kliknutím se popis zavírá křížkem v jeho pravém horním rohu (lze ho vidět na Obrázku 3.1 pod číslem **5**).

3.6 Přibližování

Změna přiblížení proteinové sekvence a anotací je možná pomocí kolečka myši nebo posuvem okrajů pravítka v horní části modulu zobrazujícího sekvenci. Při dostatečném přiblížení se na řádku umístěném pod pravítkem zobrazí kódy aminokyselin.

Závěr

Cílem této práce byl vývoj nové verze javascriptového webového pluginu MolArt v jazyce TypeScript, s rozšířenou funkcionalitou a možností snadno nahradit 3D vizualizační komponentu. To se nám povedlo vytvořením pluginu MolArt 2.0, který zachovává všechny důležité funkce MolArtu, například automatické načtení experimentálních i predikovaných struktur, možnost zadání vlastních anotací či mapování uživatelem, synchronizace zvýrazňování v obou vizualizačních komponentách a možnost nastavení uživatelského zvýraznění ve 3D struktuře. MolArt 2.0 navíc poskytuje možnost zvýrazňování více vybraných anotací zároveň a k vizualizaci používá novější knihovny.

Jedna z největších překážek byla, že nová verze knihovny používané k vizualizaci proteinové sekvence nezachovala všechny funkcionality původní verze. Tato nová verze již neposkytuje pouze jednu komponentu, která zajišťuje celou vizualizaci, jak to bylo u verze předchozí, ale nabízí několik menších vizualizačních podkomponent, které bylo nutné správně zkombinovat. Přestože existuje několik projektů používajících zmiňovanou knihovnu k podobnému účelu, žádný z nich nesplňoval všechny naše požadavky a nedosahoval dostatečné kvality abychom ho mohli integrovat do našeho pluginu. Z tohoto důvodu jsme vytvořili modul s názvem uniprot-nightingale.

Možným rozšířením do budoucna je obohacení grafického uživatelského rozhraní, například o možnost pokročilejšího nastavení zvýrazňování nebo o možnost zadání ID sekvence, která má být načtena. Jako další vylepšení se nabízí výměna komponenty vizualizující strukturu nebo schopnost zobrazení a propojení více vizualizačních komponent zároveň.

Seznam použité literatury

Bartovská, L. a Šišková, M. (2005). Vodíková vazba (vodíkový můstek). online. URL http://147.33.74.135/knihy/uid_es-001/hesla/vodikova_vazba.html. Co je co v povrchové a koloidní chemii - elektronický výkladový slovník [cit. 2021-06-30].

Berman, H., Henrick, K. a Nakamura, H. (2003). Announcing the worldwide protein data bank. *Nature Structural & Molecular Biology*, **10**(12), 980–980. doi: 10.1038/nsb1203-980. URL https://doi.org/10.1038/nsb1203-980.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. a Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, **28**(1), 235–242. ISSN 0305-1048. doi: 10.1093/nar/28.1.235. URL https://doi.org/10.1093/nar/28.1.235.

Bienert, S., Waterhouse, A., de Beer, T., Tauriello, G., Studer, G., Bordoli, L. a Schwede, T. (2016). The SWISS-MODEL Repository—new features and functionality. *Nucleic Acids Research*, **45**(D1), D313–D319. ISSN 0305-1048. doi: 10.1093/nar/gkw1132. URL https://doi.org/10.1093/nar/gkw1132.

Commons, W. (2007). The four levels of protein structure (primary, secondary, tertiary, and quaternary). URL https://commons.wikimedia.org/wiki/File:Figure_03_04_09.jpg. File: Figure_03_04_09.jpg [cit. 2021-07-08].

Dana, J. M., Gutmanas, A., Tyagi, N., Qi, G., O'Donovan, C., Martin, M. a Velankar, S. (2018). SIFTS: updated Structure Integration with Function, Taxonomy and Sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. *Nucleic Acids Research*, **47**(D1), D482–D489. ISSN 0305-1048. doi: 10.1093/nar/gky1114. URL https://doi.org/10.1093/nar/gky1114.

Deshpande, M. Pdb protvista viewer. online. URL https://github.com/PDBeurope/protvista-pdb. [cit. 2021-07-03].

EBI. Nightingale. online. URL https://ebi-webcomponents.github.io/nightingale/. [cit. 2021-07-03].

EBI (2021). Current release statistics. URL https://www.ebi.ac.uk/uniprot/TrEMBLstats. [cit. 2021-07-19].

EMBL-EBI. Protvista: User guide. online. URL https://ebi-uniprot.github.io/ProtVista/userGuide.html. [cit. 2021-07-03].

Hoksza, D., Gawron, P., Ostaszewski, M. a Schneider, R. (2018). MolArt: a molecular structure annotation and visualization tool. *Bioinformatics*, **34**(23), 4127–4128. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty489. URL https://doi.org/10.1093/bioinformatics/bty489.

MEFANET (2018). Klasifikace a struktura proteinů. online. URL https://www.wikiskripta.eu/w/Klasifikace_a_struktura_protein%C5%AF. [cit. 2021-07-09].

MEFANET (2019a). Nukleotid. online. URL https://www.wikiskripta.eu/w/Nukleotid. [cit. 2021-06-30].

MEFANET (2019b). Bílkoviny (1. lf uk, nt). online. URL https://www.wikiskripta.eu/w/Bílkoviny_(1._LF_UK,_NT). [cit. 2021-07-09].

PDBe-KB consortium (2019). PDBe-KB: a community-driven resource for structural and functional annotations. *Nucleic Acids Research*, **48**(D1), D344–D353. ISSN 0305-1048. doi: 10.1093/nar/gkz853. URL https://doi.org/10.1093/nar/gkz853.

Sehnal, D. (2019). Litemol. online. URL https://github.com/dsehnal/LiteMol. [cit. 2021-07-03].

Sehnal, D., Bittrich, S., Deshpande, M., Svobodová, R., Berka, K., Bazgier, V., Velankar, S., Burley, S. K., Koča, J. a Rose, A. S. (2021). Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*, **49**(W1), W431–W437. ISSN 0305-1048. doi: 10.1093/nar/gkab314. URL https://doi.org/10.1093/nar/gkab314.

The UniProt Consortium (2020). UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, **49**(D1), D480–D489. ISSN 0305-1048. doi: 10.1093/nar/gkaa1100. URL https://doi.org/10.1093/nar/gkaa1100.

Watkins, X. protvista-uniprot. online. URL https://github.com/ebi-webcomponents/protvista-uniprot. [cit. 2021-07-03].

Watkins, X., Garcia, L. J., Pundir, S., Martin, M. J. a Consortium, U. (2017). ProtVista: visualization of protein sequence annotations. *Bioinformatics*, **33**(13), 2040–2041. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx120. URL https://doi.org/10.1093/bioinformatics/btx120.

Wikipedia (2021a). Binding site — Wikipedia, the free encyclopedia. online. URL https://en.wikipedia.org/wiki/Binding_site. [cit. 2021-07-03].

Wikipedia (2021b). FASTA format — Wikipedia, the free encyclopedia. online. URL http://en.wikipedia.org/w/index.php?title=FASTA\format&oldid=1021800055. [cit. 2021-07-09].

Wikipedia (2021c). Functional group — Wikipedia, the free encyclopedia. online. URL http://en.wikipedia.org/w/index.php?title=Functional\group&oldid=1032136186. [cit. 2021-07-09].

Wikipedia (2021d). Peptide bond — Wikipedia, the free encyclopedia. online. URL http://en.wikipedia.org/w/index.php?title=Peptide\bond&oldid=1023121594. [cit. 2021-07-08].

Seznam obrázků

1.1 Peptidická vazba 4
1.2 Úrovně struktur proteinů 6
1.3 Ukázka mapování první úrovně 12
1.4 Ukázka zvýrazněných a vybraných anotací 13
1.5 Ukázka nástroje MolArt 2.0 15
1.6 Ukázka nástroje ProtVista 16
1.7 Ukázka komponenty protvista-uniprot 17
1.8 Ukázka komponenty ProtVista PDB 18

2.1 Objektový model komponenty `uniprot-nightingale` 1. část . . . 20
2.2 Objektový model komponenty `uniprot-nightingale` 2. část . . . 21
2.3 Vizualizace funkce tříd `FragmentWrapper` a `RowWrapper` 30
2.4 Ukázka Nightingale komponent 32

3.1 Ukázka uživatelského rozhraní 41
3.2 Ukázka uživatelského rozhraní filtru mutací 42

Seznam použitých zkratek

EBI European Bioinformatics Institute 9

JSON JavaScript Object Notation 4, 8, 20

PDB Protein Data Bank 7, 10–12, 31

PDBe PDB in Europe 10, 16, 21, 31

PDBe-KB PDBe - Knowledge Base 10, 11, 21

PDBj PDB Japan 10

PIR Protein Information Resource 9

RCSB Research Collaboratory for Structural Bioinformatics 10

SIB Swiss Institute of Bioinformatics 9

SIFTS Structure Integration with Function, Taxonomy and Sequence 10

SMR SWISS-MODEL Repository 10, 11, 21

UniParc UniProt Archive 9

UniProt Universal Protein 6, 9, 12, 31

UniProtKB UniProt Knowledgebase 9–11

UniRef UniProt Reference Clusters 9

wwPDB Worldwide PDB 10

A. Přílohy

A.1 Zdrojové kódy

Součástí přílohy jsou zdrojové kódy k modulům `uniprot-nightingale`, `molart` a `nightingale-types` rozdělené do stejnojmenných adresářů.

Struktura přílohy se zdrojovými soubory je následující:

- `/molart/src` - Zdrojové kódy Molartu 2.0.
- `/molart/src/examples` - Příklady použití a konfigurace MolArtu 2.0.
- `/molart/test` - Unit testy pro Molart 2.0.
- `/uniprot-nightingale/src` - Zdrojové kódy uniprot-nightingale modulu.
- `/uniprot-nightingale/src/examples` - Příklady použití a konfigurace uniprot-nightingale modulu.
- `/uniprot-nightingale/src/protvista` - Zdrojové kódy upravující chování Nightingale komponent.
- `/uniprot-nightingale/test` - Unit testy pro uniprot-nightingale modul.
- `/nightingale-types/index.d.ts` - Soubor deklarující datové typy pro Nightingale komponenty.

A.2 Typy popisující uživatelské anotace

```
type Feature = {
  readonly type: string;
  readonly category?: string;
  readonly description?: string;
  readonly alternativeSequence?: string;
  readonly begin: string;
  readonly end: string;
  readonly xrefs?: Xref[];
  readonly evidences?: Evidence[];
  readonly unique?: boolean;
  readonly matchScore?: number;
  readonly color?: string;
}
type VariantWithCategory = VariantWithSources & {
  readonly category: string;
};

type VariantWithSources = Partial<Variant> & {
  readonly type: string;
  readonly otherSources?: Record<string, OtherSourceData>;
  readonly tooltipContent?: TooltipContent;
  readonly description?: string;
```

```
  readonly customSource?: string;
  readonly color?: string;
  readonly variant: AminoAcid;
  readonly begin: string;
  readonly end: string;
  readonly start: string;
};

type OtherSourceData = {
  readonly predictions?: Prediction[];
  readonly description?: string;
  readonly evidences?: Evidence[];
  readonly consequenceType?: ConsequenceType;
  readonly xrefs?: Xref[];
};

type Variant = {
  type: string;
  alternativeSequence?: AminoAcid;
  begin: string;
  end: string;
  xrefs: Xref[];
  consequenceType: ConsequenceType;
  wildType: AminoAcid;
  predictions?: Prediction[];
  sourceType: SourceType;
  association?: Association[];
  evidences?: Evidence[];
  ftId?: string;
};
enum AminoAcid {
  A = "A",
  C = "C",
  D = "D",
  E = "E",
  Empty = "*",
  F = "F",
  G = "G",
  H = "H",
  I = "I",
  K = "K",
  L = "L",
  M = "M",
  N = "N",
  Nl = "NL",
  P = "P",
  Q = "Q",
  R = "R",
```

```
  S = "S",
  T = "T",
  V = "V",
  W = "W",
  Y = "Y"
}
type Association = {
  name: string;
  dbReferences?: Xref[];
  evidences: Evidence[];
  disease: boolean;
  description?: string;
};
type Xref = {
  name: string;
  id: string;
  url: string;
  alternativeUrl?: string;
};
enum Source {
  CLINVAR = "ClinVar",
  ESP = "ESP",
  EXAC = "ExAC",
  GENOMES1K = "1000Genomes",
  BOVINE_SNP50 = "BovineSNP50",
  BOVINE_LD = "BovineLD",
  BOVINE_HD = "BovineHD",
  EQUINE_SNP50 = "EquineSNP50",
  CHICKEN_600K = "Chicken600K",
  KG_HQ = "1kg_hq",
  COSMIC = "cosmic curated",
  REVIEWED = "reviewed",
  UNIPROT = "UniProt",
  DBSNP = "dbSNP",
  ENSEMBL = "Ensembl",
  ENSEMBL_PLANTS = "EnsemblPlants",
  VECTORBASE = "VectorBase",
  REFSEQ = "RefSeq",
  DBGAP = "dbGaP",
  DDD = "DDD",
  PHARMCOGKB = "PharmGKB",
  ENSEMBL_FUNGI = "EnsemblFungi",
  ENSEMBL_METAZOA = "EnsemblMetazoa",
  GNOMAD_V2 = "gnomAD_v2.0",
  GNOMAD_V3 = "gnomAD_v3.0",
  TCGA = "NCI-TCGA",
  TCGA_COSMIC = "NCI-TCGA Cosmic",
  DECIPHER = "ddG2P",
```

```
  TOPMED = "TOPMed",
  GNOMAD = "gnomAD",
  PHARMGKB = "pharmgkb",
  SGRP = "SGRP",
  SGD = "SGD",
  JEFFARES_SNPS = "Jeffares_SNPs",
  JEFFARES_INDELS = "Jeffares_Indels",
  ENSEMBL_VIRUSES = "EnsemblViruses"
}
type Evidence = {
  code: string;
  source: Xref;
};
enum ConsequenceType {
  Empty = "-",
  Frameshift = "frameshift",
  InframeDeletion = "inframe deletion",
  Insertion = "insertion",
  Missense = "missense",
  StopGained = "stop gained"
}
type Prediction = {
  predictionValType: PredictionValType;
  predictorType: string;
  score: number;
  predAlgorithmNameType: PredAlgorithmNameType;
  sources: Source[];
  version?: string;
};
enum PredAlgorithmNameType {
  PolyPhen = "PolyPhen",
  Sift = "SIFT"
}
enum PredictionValType {
  Benign = "benign",
  Deleterious = "deleterious",
  DeleteriousLowConfidence = "deleterious - low confidence",
  PossiblyDamaging = "possibly damaging",
  ProbablyDamaging = "probably damaging",
  Tolerated = "tolerated",
  ToleratedLowConfidence = "tolerated - low confidence",
  Unknown = "unknown"
}
enum SourceType {
  LargeScaleStudy = "large_scale_study",
  Mixed = "mixed",
  UniProt = "uniprot"
}
```

Kód A.1: Na řádku 1 začíná datová struktura představující obecnou anotaci a na řádku 14 začíná typ definující uživatelskou anotaci typu mutace.

A.3 Datový typ popisující uživatelské pokrytí sekvence a mapování

```
type PDBParserData = PDBParserItem[];

type PDBParserItem = {
  readonly end: number;
  readonly chain_id: string;
  readonly pdb_id: string;
  readonly start: number;
  readonly unp_end: number;
  readonly coverage?: number;
  readonly polymer_coverage: PolymerCoverage;
  readonly unp_start: number;
  readonly experimental_method?: string;
  readonly tax_id?: number;
  readonly tax_ids?: number[];
  readonly structure: StructureData;
  readonly mappings: ParserMapping;
};

type StructureData = {
  format: "mmcif" | "pdb";
  data?: string;
  url?: string;
};

type PolymerCoverage = Record<
  string,
  {
    readonly molecules: Molecule[];
  }
>;

type Molecule = {
  readonly entity_id: number;
  readonly chains: ChainData[];
};

type ChainData = {
  readonly observed: Observed[];
  readonly chain_id: string;
```

```
40 };
41
42 type Observed = {
43   readonly start: {
44     readonly residue_number: number;
45   };
46   readonly end: {
47     readonly residue_number: number;
48   };
49 };
```

Kód A.2: Na řádku 1 začíná datová struktura popisující uživatelské pokrytí sekvence a mapování.

A.4 Git repozitáře

Veřejně dostupné repozitáře, na kterých jsou umístěny zdrojové kódy vyvinuty v rámci této práce:

- `https://github.com/scheuerv/uniprot-nightingale`
- `https://github.com/scheuerv/molart`
- `https://github.com/scheuerv/nightingale-types`

A.5 Npm

V rámci práce jsme vytvořili dva balíčky s knihovnami přístupné přes npm.

- `https://www.npmjs.com/package/uniprot-nightingale`
- `https://www.npmjs.com/package/molart2`