Pohrana podataka o molekulama u sustavu za upravljanje bazom podataka PostgreSQL

Lukin, Josip

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija
Sveučilišni diplomski studij Informatika

Josip Lukin

Pohrana podataka o molekulama u sustavu za upravljanje bazom podataka PostgreSQL

Diplomski rad

Mentor: prof. dr. sc. Vedran Miletić

Rijeka, 1. veljače 2024.

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 1. veljače 2023.

Zadatak za diplomski rad

Pristupnik: Josip Lukin

Naziv diplomskog rada: Pohrana podataka o molekulama u sustavu za upravljanje bazom podataka PostgreSQL

Naziv diplomskog rada na eng. jeziku: Molecular data storage in PostgreSQL database management system

Sadržaj zadatka:

PostgreSQL se smatra najnaprednijim postojećim sustavom za upravljanje relacijskom bazom podataka otvorenog koda. Osim podrške za osnovne tipove podataka, omogućuje korištenje i valuta, nabrajanja, geometrijskih pojmova, mrežnih adresa, rezultata pretrage teksta, XML-a, JSON-a te brojnih drugih. Jedno od proširenja treće strane, dio projekta RDKit, omogućuje pohranu podataka o molekulama. Cilj rada je na primjeru razvoja vlastite aplikacije za rad s molekulama prikazati mogućnosti i način korištenja tog proširenja.

Mentor:

Doc. dr. sc. Vedran Miletić

Voditeljica za diplomske radove:

Prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 1. veljače 2023.

(potpis pristupnika)

Adresa: **Radmile Matejčić 2**
51000 Rijeka, Hrvatska

uniri

Sažetak

Diplomski rad usmjeren je na istraživanje i implementaciju rješenja za efikasnu pohranu, pretraživanje i manipulaciju podacima o molekulama koristeći sustav za upravljanje bazom podataka PostgreSQL. U radu je naglašena važnost integracije RDKit-a, otvorenog softverskog alata za kemoinformatiku, s PostgreSQL bazom podataka, čime se omogućava napredno rukovanje molekularnim strukturama unutar baze podataka. Rad detaljno opisuje proces izgradnje RDKit alata iz izvornog koda, modeliranja i populaciju baze podataka molekularnim podacima, implementaciju REST API-ja za pristup i manipulaciju podacima, te funkcionalnosti sustava u kontekstu različitih operacija nad molekulama, kao što su pretraga po strukturi, substrukturalna pretraga i izračunavanje molekulskih deskriptora.

Ključne riječi: PostgreSQL; molekule; RDKit; REST API

Sadržaj

Poglavlje 1

Uvod

U današnjem digitalnom dobu, web aplikacije zauzimaju ključnu ulogu u pružanju i pristupanju raznorodnim uslugama i informacijama. Omogućuju korisnicima interaktivno sudjelovanje u digitaliziranom poslovanju, online trgovini, društvenim mrežama, igrama, edukaciji i mnogim drugim domenama, bez potrebe za instalacijom posebnog softvera. Široku primjenu i rastuću popularnost web aplikacija potpomaže niz ključnih prednosti, uključujući dostupnost sa bilo kojeg mjesta i uređaja, skalabilnost koja omogućuje lako prilagođavanje rastućim potrebama, interoperabilnost sa različitim sustavima i platformama, visoku razinu sigurnosti, te jednostavnost ažuriranja i održavanja. Ovakav tehnološki razvoj nužno vodi prema inovacijama i specijalizacijama unutar web tehnologija, posebno u domenama koje zahtijevaju obradu i analizu specifičnih skupova podataka. Molekularni podaci su jedan od takvih skupova, te postoji potražnja za rješenjima koja omogućuju pohranu, indeksiranje, pretraživanje i analizu molekula u okviru web aplikacija. Diplomski rad fokusira se na razvoj web aplikacije koja omogućuje pohranu i analizu molekula, koristeći PostgreSQL bazu podataka s instaliranom RDKit ekstenzijom. RDKit je moćan alat za kemoinformatiku, koji omogućuje manipulaciju i analizu molekularnih struktura. Integracijom RDKit ekstenzije u PostgreSQL bazu podataka, rad pokazuje kako se napredne funkcionalnosti baza podataka mogu iskoristiti za implementaciju efikasne infrastrukture za obradu molekularnih podataka, koji je istovremeno jednostavan, siguran i skalabilan. Aplikacija za web pohranu i obradu molekularnih podataka izrađena je kroz čvrstu integraciju ključnih tehnologija uključujući PostgreSQL za upravljanje bazom podataka, RDKit za obradu molekularnih podataka, Spring Boot kao okvir za razvoj aplikacije i Java kao programski jezik. Posebno, razvijen je REST API koji omogućava laku integraciju s drugim aplikacijama/servisima. Rad je omogućuje dubok uvid u prednosti, izazove i mogućnosti integracije ovih tehnologija. Na taj način, diplomski rad ne samo da pruža vrijedan doprinos akademskoj zajednici i industriji kroz razvoj specijalizirane web aplikacije za pohranu i obradu molekularnih podataka, već također ističe potencijal za daljnja istraživanja i razvoj u ovom brzo rastućem tehnološkom području.

Poglavlje 2

Teorijska pozadina

Kemoinformatika je interdisciplinarno područje znanosti koje se bavi razvojem metoda i softvera za rješavanje kemijskih problema. Naziv računalna kemija se također nekad koristi za dijelove kemoinformatike koji se bave implementacijom metoda teorijske kemije na računalu [1]. Računalna kemija obuhvaća čitav niz računalnih tehnika usmjerenih prema razumijevanju strukture i svojstava molekula, materijala i velikih bioloških sustava [2]. Osnovni je segment suvremene znanosti ključne za razvoj i unapređenje tehnologija u području kemije, ali i ostalih znanstvenih disciplina. Eksponencijalni rast računalne snage, uz napredak u algoritmima i softverskim rješenjima, omogućio je analizu i predikciju svojstava molekula primjenom računalne kemije. Na taj se način znanstvenicima omogućava precizno modeliranje i simulacija ponašanja molekula na atomskoj razini, čime se pridonosi dubljem razumijevanju njihovih fizičkih i kemijskih svojstava, što je od posebne važnosti za razvoj novih lijekova, materijala s posebnim svojstvima te u razumijevanju složenih bioloških procesa. Kombinacijom računalne kemije s eksperimentalnom praksom omogućena je interpretacija složenih eksperimentalnih podataka, ali i dizajniranje i sinteza molekula s unaprijed definiranim svojstvima, što rezultira uštedom vremena i resursa u laboratorijskim istraživanjima. S obzirom na navedeno, pohrana i upravljanje molekularnim podacima u bazama podatak kao što je PostgreSQL sve je relevantnija, otvarajući novo polje učinkovitog upravljanja i analize znanstvenim podacima na globalnoj razini.

2.1 Rad s podacima o molekulama

Osnovni rad s podacima o molekulama uključuje upotrebu specifičnih formata i standarda za opisivanje kemijskih struktura, omogućujući njihovu digitalnu manipulaciju i analizu. Među najčešće korištene specifikacije za opisivanje kemijskih struktura molekula spadaju SMILES (Simplified Molecular Input Line Entry System), SMARTS (SMILES Arbitrary Target Specification), InChI (International Chemical Identifier) i MOLFILE formati.

- SMILES je tekstualni zapis koji na jednostavan način omogućava precizno reprezentira-

nje molekulske strukture pomoću niza ASCII znakova. SMILES može opisati strukturu molekule s obzirom na povezanost atoma, stereoizomteriju i pojedine funkcionalne skupine [3].

- SMARTS zapis je proširenje SMILES notacije, omogućujući definiranje generičkih molekularnih uzoraka i struktura. To je posebno korisno u pretrazi baza podataka molekula, gdje SMARTS može služiti za identificiranje molekula koje zadovoljavaju određene kemijske kriterije, kao što su prisutnost specifičnih funkcionalnih grupa ili skeleta. SMARTS notacija omogućava vrlo fleksibilno i moćno sredstvo za specifikaciju složenih kemijskih upita, što je čini nezamjenjivim alatom u kemoinformatičkim aplikacijama [4].
- InChI predstavlja jedinstveni identifikator za kemijske spojeve, koji je razvijen od strane Međunarodne unije za čistu i primijenjenu kemiju (IUPAC). Pruža standardiziran način opisa molekularne strukture koji je neovisan o softveru, što olakšava razmjenu podataka i pretragu [5].
- MOLFILE je format datoteke koji se koristi za opisivanje molekularnih struktura u 2D ili 3D formatu. Ovaj format sadrži detaljne informacije o položaju atoma, veza između njih, kao i informacije o stereoizometriji.

Korištenjem ovih i drugih srodnih specifikacija, omogućava efikasnu digitalizaciju kemijskih informacija, što je temelj za razvoj baza podataka molekula, algoritama za pretragu i analizu molekularnih struktura, te računalno potpomognuto dizajniranje lijekova.

2.2 RDKit

RDKit je kemoinformatički softver otvorenog koda dizajniran za olakšavanje manipulacije i analizu kemijskih struktura. Inicijalno je nastao 2000. godine unutar Rational Discovery projekta, a 2006. godine prebačen je na SourceForge pod licencom BSD, gdje se razvija i održava od strane međunarodne zajednice znanstvenika i programera kao jedan od ključnih alat u polju kemoinformatike, kemije i bioinformatike [6]. Nudi širok spektar funkcionalnosti, uključujući generiranje i pretraživanje molekularnih otisaka, izračunavanje fizičko-kemijskih svojstava, modeliranje molekularne dinamike, građenje i optimizacija konformera, klasifikacija i regresija pomoću neuronskih mreža, podrška za rad s 3D strukturama i mnogo drugih. Implementiran u C++, s Python i Java vezama, omogućuje širokom krugu programera i istraživača brzo i efikasno rješavanje različitih problema iz domena dizajna lijekova, materijala, bioinformatike, nanotehnologije itd. Jedna od značajki koju nudi je RDKit Cartridge, moćan i fleksibilan alat za upravljanje i analizu velikih skupova kemijskih podataka. Uključivanje RDKit-a u druge projekte otvorenog koda pokazuje njegovu prilagodljivost i ključnu funkciju u domenama kemoinformatike. Kompatibilnost s alatima za analitiku kao što je KNIME omogućuje korisnicima neprimjetno uključivanje funkcionalnosti kemoinformatike u svoje analize podataka. Podrška okvira za web razvoj putem Djanga i njegova dostupnost u web aplikacijama putem RDKit.js demonstriraju njegovu fleksibilnost.Korištenje RDKit-a u nizu drugih projekata otvorenog koda

naglašava njegovu važnost u unapređenju istraživanja kemijskih i molekularnih znanosti. Činjenica da brojni drugi projekti otvorenog koda koriste RDKit naglašava koliko je važan za napredak istraživanja kemijskih i molekularnih znanosti. Projekti kao što su Datamol, DockOnSurf i Scopy iskorištavaju RDKit za izgradnju baze podataka, optimizaciju geometrije i molekularnu manipulaciju za visoku propusnost i virtualni pregled. Dok biblioteke kao što su *stk* i *gpusimilarity* koriste RDKit za molekularni dizajn i pretraživanje sličnosti, Open Force Field Toolkit ga koristi za parametrizaciju polja sile. Primjena RDKit-a uključuje modeliranje kemijskih predmeta s CheTo-om, stvaranje baze podataka s *mmpdb*-om i strojno učenje u kemiji s DeepChem-om. Štoviše, RDKit olakšava stvaranje specijaliziranih alata kao što je ZINC za virtualni pregled, ChEMBL Beaker za internetske kemoinformatičke usluge i OCEAN za procjenu unakrsne reaktivnosti. Dodatno, podržava analitičke i instruktivne alate kao što je MolGears za kemoinformatiku bioaktivnih molekula i *sdf_viewer.py* za interaktivnu molekularnu vizualizaciju. Činjenica da je RDKit kompatibilan s Python skriptiranjem u bazama podataka i da se može integrirati s KNIME čvorovima pokazuje koliko je svestran i kako pomaže u pojednostavljenju keminoformatičkih procedura. Ova opsežna integracija i upotreba u različitim projektima ne samo da odražava robusnost i pouzdanost RDKita, već i njegovu ulogu u poticanju inovacija i suradnje unutar znanstvene zajednice. Stalna ažuriranja i doprinosi korisnika i programera osiguravaju da RDKit ostane na čelu kemoinformatičkih alata, olakšavajući istraživanje i razvoj u kemijskim i molekularnim znanostima.

2.3 PostgreSQL

PostgreSQL je snažan objektno-relacijski sustav baze podataka otvorenog koda koji koristi i nadograđuje SQL jezik uz mogućnost dodavanja brojnih funkcionalnosti koje pospješuju sigurno skladištenje i upravljanje kompleksnim setovima podataka. Porijeklo PostgreSQL-a seže u 1986. godinu, kad je bio dio projekta POSTGRES na Kalifornijskom sveučilištu u Berkeleyju, razvijajući se do danas više od 35 godina [7]. Zbog svoje dokazane arhitekture, pouzdanosti, sigurnosti podataka, opsežnih funkcionalnosti, mogućnosti proširenja i podrške zajednice otvorenog koda, PostgreSQL stekao je ugled visokokvalitetnog softvera. Podržava sve glavne operativne sustave, pridržava se ACID principa od 2001. godine [7] i nudi raznovrsne dodatke, poput postGIS-a za geo-prostorne podatke stoga je popularan izbor među korisnicima i organizacijama. PostgreSQL nudi različite značajke koje olakšavaju razvoj aplikacija, očuvanje integriteta podataka, izgradnju pouzdanih sistema i upravljanje podacima neovisno o njihovoj veličini. Kao besplatni i otvoreni softver, PostgreSQL se ističe svojom sposobnošću proširenja, omogućujući korisnicima da kreiraju vlastite tipove podataka, funkcije i čak koriste različite programske jezike bez potrebe za ponovnim kompajliranjem. PostgreSQL teži usklađivanju sa SQL standardima, osim u slučajevima kada bi to moglo utjecati na njegove ključne funkcionalnosti ili dovesti do lošeg dizajna. Podržava mnoge zahtjeve SQL standarda, ponekad uz malo modificiranu sintaksu ili funkcionalnost. S vremenom se mogu očekivati daljnji pomaci prema

usklađenosti. Od izdanja 16 verzije u rujnu 2023., PostgreSQL je usklađen s najmanje 170 od 179 obveznih značajki za SQL:2023 Core usklađenost. Važno je napomenuti da u ovom trenutku niti jedna relacijska baza podataka nije u potpunosti usklađena s ovim standardom [7]. Sustav je dokazano skalabilan, sposoban za upravljanje velikim količinama podataka i velikim brojem korisnika, s klasterima koji u produkciji rade s terabajtima podataka i specijaliziranim sustavima za upravljanje petabajtima podataka.

2.4 Java

Java je objektno-orijentirani programski jezik visoke razine koji je dizajniran da ima što manje implementacijskih ovisnosti [8]. Razvijen od strane Sun Microsystems (trenutačno dio Oracle Corporation), Java je brzo stekala popularnost u razvojnoj zajednici zbog toga što slijedi filozofiju "write once, run anywhere" (WORA) [9], što znači da se Java programi, jednom napisani, mogu izvoditi na bilo kojoj platformi koja podržava Java virtualnu mašinu (JVM). Ova portabilnost i platformska neovisnost čine Javu izuzetno privlačnom za razvoj aplikacija u širokom spektru okruženja, od ugradbenih uređaja i mobilnih telefona do velikih poslovnih sustava. Programski jezik koji je promijenio svijet računalnog programiranja, svoje korijene vuče iz ranih 90-ih godina prošlog stoljeća kada se u Sun Microsystemsu, tim predvođen Jamesom Goslingom, često nazivanim *"ocem Jave"*, upustio u projekt koji je prvobitno nazvan *Hrast* [10], po hrastu koji je stajao ispred Goslingovog ureda. Projekt *Hrast* je preimenovan u Java 1994. nakon što je pretraga zaštitnih znakova otkrila da Oak Technology polaže prava na naziv *Hrast*. Javu je službeno objavio Sun Microsystems u svibnju 1995. godine, označavajući revolucionarni korak u svijetu računalstva. U studenom 2006. Sun Microsystems donesena je ključna odluka da se Javu plasira kao besplatni softver otvorenog koda pod GNU General Public License (GPL), što je uvelike ubrzalo njezino usvajanje i razvoj. Godine 2010. Oracle Corporation je preuzima Sun Microsystems, postajući upravitelj Jave. Prijelaz nailazi na različite reakcije Java zajednice, posebno u vezi s Oracleovim rukovanjem Javom i njezinim ekosustavom.

Java je bila pod utjecajem drugih programskih jezika, posebno C++-a, od kojeg je preuzela sintaksu i koncepte [11], ali i uvela neke novine, kao što su automatsko upravljanje memorijom (garbage collection), iznimke (exceptions) i generički tipovi (generics). Dizajnirana je da bude neovisna o platformi, sigurna i sposobna za rad na malim uređajima s ograničenom procesorskom snagom i memorijom zahvaljujući Java Virtual Machine (JVM) — revolucionarnoj komponenti koja omogućuje pokretanje Java aplikacija na bilo kojem uređaju opremljenom JVM-om, bez obzira na temeljni hardver i operativni sustav. Jezik podržava objektno-orijentiranu paradigmu, što znači da je programiranje u Javi usmjereno na stvaranje i manipulaciju objektima koji predstavljaju instance klasa. Ključne karakteristike Java programskog jezika uključuju enkapsulaciju, nasljeđivanje i polimorfizam. Enkapsulacija omogućava skrivanje unutarnjih stanja objekta i izlaganje samo onih dijelova koji su potrebni za vanjsku interakciju. Nasljeđivanje

omogućava klasi da naslijedi metode i varijable od druge klase, dok polimorfizam omogućava objektima da se obrađuju kao njihovi roditeljski tipovi ili sučelja. Java je također poznata po svojoj bogatoj standardnoj biblioteci koja pruža programerima velik broj gotovih klasa i metoda za obavljanje raznih zadataka, uključujući mrežnu komunikaciju, rad s datotekama, grafičko korisničko sučelje i obradu XML-a. Ove biblioteke, zajedno s velikim brojem programskih okvira (framework) razvijenih za Javu, kao što su Spring, Hibernate i JavaServer Faces (JSF), olakšavaju razvoj aplikacija. Koristi se za razvoj web aplikacija, mobilnih aplikacija (posebno Android, koji primarno koristi Javu), poslovnih aplikacija do softvera za integrirane sustave. Kroz godine, nastavila je evoluirati kroz različite verzije, uvodeći nove značajke kao što su lambda izrazi, tokovi (streams), što omogućava pisanje čišćeg, efikasnijeg i održivog kod. Unatoč pojavi novih programskih jezika, poput Go, Kotlin, Scala, Swift, itd., i dalje je jedan od najpopularnijih i najpouzdanijih programskih jezika u svijetu softverskog razvoja.

2.5 Spring programski okvir

Programski okvir otvorenog koda Spring Framework, predstavlja jedan od vodećih programskih okvira za razvoj Java aplikacija, posebno usmjerenih na poslovne aplikacije. Olakšava razvoj samostalnih produkcijskih aplikacija koje se izvode na Java virtualnoj mašini (JVM) nudeći značajku ubacivanja ovisnosti (dependency injection) [12]. Ova značajka objektima omogućuje definiranje vlastitih ovisnosti koje Spring kontrolira i povezuje te tako omogućuje stvaranje modularnih aplikacija koje se sastoje od labavo povezanih komponenti koje su idealne za mikroservise i distribuirane mrežne aplikacije. Uz to, Spring nudi širok spektar podrška za razne aspekte razvoja aplikacija, uključujući manipulaciju podacima, pretvorbu tipa, validaciju, internacionalizaciju, MVC web razvoj, sigurnost, rad s RESTful web servisima i slično [13]. Jednostavno Spring oprema programere sveobuhvatnim paketom alata za izgradnju višeplatformskih, slabo povezanih Java aplikacija prikladnih za svako okruženje. Međutim, upravo zbog toga što je toliko opsežan i fleksibilan, sam Spring može biti težak za naučiti i primijeniti novim korisnicima. Postavljanje i konfiguracija Spring aplikacije zahtijeva detaljno razumijevanje kako različiti dijelovi Springa rade zajedno, a koji može oduzeti puno vremena. Upravo zbog toga dolazi do izražaja važnost Spring Boot-a, podokvira unutar Spring ekosustava, koji je dizajniran da automatizira proces konfiguracije i ukloni potrebu za repetitivnim konfiguracijskim kodom. Spring Boot omogućava brzo postavljanje i razvijanje nove Spring aplikacije, koristeći auto-konfiguraciju i pružajući niz starter paketa koji automatski konfiguriraju aplikaciju za različite potrebe. Eliminacijom XML konfiguracije, koja je bila karakteristična za rane verzije Springa, Spring Boot promovira konfiguraciju temeljenu na anotacijama, koja aplikacije čini jednostavnijima. Sve u svemu, Spring i Spring Boot zajedno pružaju moćan set alata za izgradnju robusnih, visoko dostupnih aplikacija koje se mogu lako skalirati i održavati.

2.6 REST API

REST API (Representational State Transfer Application Programming Interface) predstavlja arhitekturalni stil [12] odnosno skup pravila i konvencija za izgradnju web servisa. Osnovna ideja je omogućiti komunikaciju između klijentskih aplikacija i servera putem HTTP protokola, koristeći standardne HTTP metode poput GET, POST, PUT, DELETE i PATCH, koji su ekvivalentni osnovnim CRUD operacijama (Create, Read, Update, Delete) i omogućavaju manipulaciju resursima. Ključna karakteristika REST API-ja je bezstanje (stateless), što znači da svaki zahtjev od klijenta k serveru mora sadržavati sve informacije potrebne za njegovo izvršenje, tako da server nema potrebu čuvati kontekst ili stanje sesije. Ovo čini REST API veoma skalabilnim i svaka njegova komponenta je nezavisna. Podaci se razmjenjuju između klijenta i servera nekim od popularnih formata, najčešće se koriste JSON (JavaScript Object Notation) i XML (eXtensible Markup Language), s tim da JSON ima prednost zbog svoje jednostavnosti i čitljivosti. Primjena REST API-a nije ograničena samo na web aplikacije, nego se rabi u svim drugim tipovima aplikacija kao što su mobilne, desktop, IoT (Internet of Things) i to zbog jednostavnosti i visoke interoperabilnosti između različitih sustava i tehnologija. Kako bi se REST API efikasno implementirao važno je pratiti najbolje prakse u dizajniranju kao što su jasni i konzistentni nazivi resursa, efikasno upravljanje greškama, i korištenje HTTPS-a za sigurnu komunikaciju.

Poglavlje 3

Tehnička implementacija

Sljedeća implementacija odrađena je na laptopu Lenovo ThinkPad P1 Gen1 sa 2.60GHz Intel Core i7-8850H CPU i 32GB RAMa. Korišten je Windows 11 operacijski sustav uz WSL Ubuntu 20.04 na kojem je kompiliran RDKit i instalirana PostgreSQL baza podataka. Od ostalih softvera, za razvoj REST servisa korišten je IntelliJ IDEA razvojno okruženje i DBeaver kao GUI klijentska aplikacija za administriranje baze podataka.

3.1 RDKit Cartridge

RDKit Cartridge je ekstenzija za PostgreSQL koji osigurava napredno rukovanje molekularnim podacima i pristup širokom spektru funkcija za manipulaciju i analizu molekularnih podataka. Omogućuje izvršavanje složenih upita i analizu molekula izravno unutar baze podataka, što je ključno za istraživačke projekte u područjima poput farmaceutske industrije i biokemije. RDKit Cartridge ekstenzija donosi četiri nova tipa u PostgreSQL bazu podataka:

- mol - RDKit molekula koja se može kreirati iz SMILES direktno konverzijom
- qmol - RDKit molekula koja sadrži više značajki tj. kreira se iz SMARTS zapisa
- sfp - predstavlja vektorski otisak molekula rijetkog brojenja
- bfp - bitni vektorski otisak molekule

Osim pružanja raznih funkcija, RDKit Cartridge uvodi nove operatore koji se mogu koristiti u PostgreSQL bazi podataka.

Operatori za traženje sličnosti:

- # - operator koji se koristi za traženje sličnosti pomoću Dice sličnosti. Vraća je li Dice sličnost između dva otiska (bilo dvije `sfp` ili dvije `bfp` vrijednosti) veća od `rdkit.dice_threshold` - konfigurabilno svojstvo baze podataka
- <%> - koristi se za Tanimoto KNN pretraživanja (za vraćanje uređenih popisa susjeda)
- <#> - koristi se za Dice KNN pretraživanja (za vraćanje uređenih popisa susjeda)

Operatori za pretrage substrukture i točno traženje strukture

- `@>` - vraća je li mol/qmol s desne strane podstruktura mola s lijeve strane
- `<@` - vraća je li mol/qmol s lijeve strane podstruktura mola s desne strane
- `@=` - vraća jesu li dvije molekule iste

Operatori za usporedba molekula

- `<` - vraća je li lijeva molekula manja od desne
- `>` - vraća je li lijeva molekula veća od desne
- `=` - vraća je li lijeva molekula jednak desne molekule
- `<=` - vraća je li lijeva molekula manja ili jednak desnoj molekuli
- `>=` - vraća je li lijeva molekula veća ili jednak desnoj molekuli

Izgradnjom RDKita omogućuje se kreiranje ekstenzije, no proces podrazumijeva niz koraka koji uključuju preuzimanje RDKita, pripremu sustava, konfiguraciju i kompilaciju izvornog koda te postavljanje okruženja.Za preuzimanje posljednje verzije RDKita sa službenog repozitorija koristi se Git, sustav za kontrolu verzija koji omogućuje preuzimanje i ažuriranje koda projekata.

```
sudo apt-get install git
git clone https://github.com/rdkit/rdkit.git
```

Osim posljednje verzije, sa službenog repozitorija moguće je preuzeti prijašnje verzije koristeći neki od alata za dohvaćanje sadržaja sa poslužitelja na webu putem HTTP-a, HTTPS-a i FTP-a, poput `wget` ili `curl` alata. U tom slučaju potrebno je izmijeniti `Release_202X_XX_X` u sljedećoj naredbi s oznakom verzije koju se želi kompilirati.

```
sudo apt-get install wget
wget https://github.com/rdkit/rdkit/archive/refs/tags/Release_202X_XX_X.tar.gz
```

Prije preuzimanja potrebnih paketa potrebno je osvježit lokalni popis paketa i njihovih verzija prema repozitorijima definiranim u `/etc/apt/sources.list` datoteci i drugim `.list` datotekama unutar `/etc/apt/sources.list.d/` direktorija. Zatim, nadograditi sve instalirane pakete na najnovije verzije koje su dostupne prema ažuriranom popisu paketa.

```
sudo apt update && sudo apt upgrade -y
```

CMake verzija `3.1`, po službenoj dokumentaciji, najniža je verzija potrebna za kompilaciju RDKita. U okviru ovog rada koristit se verzija `3.16.3`.

```
sudo apt install cmake -y
cmake --version
cmake version 3.16.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

Temeljni C++ kod RDKit-a ažuriran je za korištenje suvremenih C++ standarda, posebno C++11,

od verzije izdane u ožujku 2018. !!!gcc compiler i dati primjere drugih i verzije!!! Posljedično, za potrebe kompilacije koda neophodne su verzije kompajlera novije od `4.8`, dok će se u okviru ovog rada primjenjivati verzija `10.5.0`.

```
gcc --version
gcc (Ubuntu 10.5.0-1ubuntu1~20.04) 10.5.0
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

g++ --version
g++ (Ubuntu 10.5.0-1ubuntu1~20.04) 10.5.0
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Prije kreiranja skripti za izgradnju RDKita, na sustavu je nužno imati PostgreSQL bazu podataka. U sklopu ovog rada koristit će se PostgreSQL verzije 12. Osim baze, za izgradnju ekstenzije ključne su i razvojne datoteke. Paket `postgresql-server-dev-12` pojednostavljuje proces izgradnje pružajući sve bitne alate potrebne za izgradnju projekata koji koriste PostgreSQL sučelje.

```
apt install postgresql
    postgresql-server-dev-12
    libboost-all-dev
    libfreetype6-dev
```

Tijekom kreiranja skripti za izgradnju, potrebne su određene vanjske biblioteke za omogućavanje različitih funkcionalnosti unutar RDKit-a. Paket `libboost-all-dev` sveobuhvatan je skup Boost biblioteka, koje poboljšavaju funkcionalnost C++. RDKit koristi Boost za mnoge različite aktivnosti, kao što je obrada nizova, upravljanje memorijom pomoću pametnih pokazivača i izvođenje matematičkih izračuna, posebno za algoritme grafova i kemijsku informatiku. Boost biblioteke pružaju čvrstu osnovu za složene strukture podataka i algoritme RDKit-a, osiguravajući učinkovitost i skalabilnost. Iako je RDKit primarno alat za kemoinformatiku, uključuje funkcionalnosti i za generiranje molekularnih vizualizacija i prikaza. Za takve vizualizacije potrebno je iscrtavanje teksta kao što je crtanje oznaka atoma, veza i slično. RDKit koristi izvrsne mogućnosti renderiranja teksta koje pruža biblioteka FreeType za generiranje točnih i jasnih molekularnih dijagrama i kemijskih struktura. Paket `libfreetype6-dev` preuzet je kako bi se jamčilo da se RDKit može kompilirati s podrškom za različite grafičke prikaze, što povećava korisnost alata u istraživačkim i razvojnim okruženjima gdje je molekularna vizualizacija neophodna.

```
cmake
  -D RDK_BUILD_PGSQL=ON
  -D RDK_BUILD_PYTHON_WRAPPERS=OFF
  -D PostgreSQL_CONFIG_DIR=/usr/lib/postgresql/12/bin
  -D PostgreSQL_INCLUDE_DIR=/usr/include/postgresql
  -D PostgreSQL_TYPE_INCLUDE_DIR=/usr/include/postgresql/12/server
  -D PostgreSQL_LIBRARY=/usr/lib/x86_64-linux-gnu/libpq.so ..
  -D RDK_BUILD_INCHI_SUPPORT=ON

make && sudo make install
```

Proces generiranja makefilesa i ostalih datoteka potrebnih za izgradnju RDKita, konfigurira se s nizom prekidača u naredbi `cmake`. Podrška za PostgreSQL uključuje se postavljanjem prekidača `RDK_BUILD_PGSQL`, koja upućuje procesu da uključi faze ili konfiguracije specifične za PostgreSQL. Upotreba RDKit Catridgea ne zahtjeva upotrebu Pythona stoga će se isključit postavljanjem `RDK_BUILD_PYTHON_WRAPPERS` prekidača. Prekidač `PostgreSQL_CONFIG_DIR` pokazuje na direktorij u kojem se nalaze binarne konfiguracijske datoteke PostgreSQLa, dok `PostgreSQL_INCLUDE_DIR` i `PostgreSQL_TYPE_INCLUDE_DIR` pokazuju na direktorije koji sadrže definicije i deklaracije potrebne za komunikaciju s PostgreSQLom. Konačno, prekidač `PostgreSQL_LIBRARY` navodi lokaciju datoteke `libpq.so`, C klijentska biblioteke za PostgreSQL potrebne za povezivanje, a koja omogućuje izravne pozive prema bazi podataka. Naredbom `make` softver se kompilira i povezuje prema makefiles i ostalih generiranih datotekama, koje specificiraju kompilaciju i procedure povezivanja dok se `make install` koristi za kopiranje izgrađenog softvera i svih potrebnih datoteka na za to određena mjesta u sustavu, nakon kompilacije. O uspješnom izvršavanju spomenutih naredbi izvještava posljednji ispis u konzoli.

```
=====================================================================
PostgreSQL cartridge installation succeeded.
=====================================================================
Restart the PostgreSQL service before using the PostgreSQL RDKit cartridge.
=====================================================================
```

Prije korištenja ekstenzije važno je stopirati instancu baze podataka te pokrenuti dobivenu skriptu `pgsql_install.sh`. Cilj skripte je dodati mogućnost kreiranje ekstenzije RDKita na instanci PostgreSQLa, osiguravajući da su sve potrebne datoteke u odgovarajućim mapama kako bi proširenje moglo ispravno funkcionirati.

```
sudo service postgresql stop
  * Stopping PostgreSQL 12 database server                    [ OK ]

sudo sh build/Code/PgSQL/rdkit/pgsql_install.sh
```

```
+ test -n
+ cp build/Code/PgSQL/rdkit/rdkit--4.4.0.sql
    /usr/share/postgresql/12/extension
+ cp Code/PgSQL/rdkit/rdkit.control
    /usr/share/postgresql/12/extension
+ cp build/Code/PgSQL/rdkit/update_sql/rdkit--3.8--4.0.sql
    build/Code/PgSQL/rdkit/update_sql/rdkit--4.0--4.0.1.sql
    build/Code/PgSQL/rdkit/update_sql/rdkit--4.0.1--4.1.sql
    build/Code/PgSQL/rdkit/update_sql/rdkit--4.1--4.1.1.sql
    build/Code/PgSQL/rdkit/update_sql/rdkit--4.1.1--4.2.sql
    build/Code/PgSQL/rdkit/update_sql/rdkit--4.2--4.3.0.sql
    build/Code/PgSQL/rdkit/update_sql/rdkit--4.3.0--4.4.0.sql
    /usr/share/postgresql/12/extension
+ cp build/Code/PgSQL/rdkit/librdkit.so
    /usr/lib/postgresql/12/lib/rdkit.so
```

3.2 Baza podataka

Naredbama `createuser` i `createdb` kreira se korisnik i baza podataka u koju je potrebno učitati podatke o molekulama i na koju se kasnije spaja Spring aplikacija. Novo kreiranom korisniku dodjeljuju se sve privilegije nad novo kreiranom bazom pomoću naredbe `grant`.

```
sudo -u postgres createuser -P -e rdkit
Enter password for new role:
Enter it again:
CREATE ROLE rdkit PASSWORD 'md51a221773d79e46439a076b0aebb447ad'
    NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT LOGIN;

sudo -u postgres createdb rdkit

sudo -u postgres psql
psql (12.17 (Ubuntu 12.17-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# grant all privileges on database rdkit to rdkit;
GRANT
```

Kreira se nova schema `emolecules` nad bazom podataka te instalira RDKit ekstenzija. Tek tad moguće je kreirati tablice koje sadrže molekule kao poseban tip podataka, koristiti razne funkcije i operatore za rad s molekulama.

```
sudo -u postgres
    psql -c 'CREATE SCHEMA emolecules AUTHORIZATION rdkit' rdkit
CREATE SCHEMA

sudo -u postgres
    psql -c 'CREATE EXTENSION rdkit SCHEMA emolecules' rdkit
CREATE EXTENSION
```

Nakon preuzimanja PostgreSQLa, kreiranja baze podataka i korisnika te uspješne instalacije RDKit ekstenzije, sljedeći korak uključuje konfiguriranje i pripremu baze podataka za učitavanje podataka o molekulama te omogućavanje udaljenog pristupa bazi podataka. Baza podataka napunit će se podacima iz `version.smi.gz` datoteke odnosno SMILES datoteke komercijalno dostupnih kemijskih spojeva preuzeta s emolecules.com web sjedišta. Za učitavanje ovih podataka potrebno je nekoliko sati iz razloga što datoteka sadrži podatke o više od 18 milijuna kemijskih spojeva. U svrhu poboljšavanja performansi tijekom učitavanja i kreiranja indeksa promijenjeno je nekoliko PostgreSQL konfiguracijskih parametara. Uređivanje datoteke `postgresql.conf`, koja se obično nalazi u `\etc\postgresql\<version>\main` direktoriju, najosnovniji je način konfiguracije postavljanjem određenih parametara.

```
synchronous_commit = off   # synchronization level;
full_page_writes = off     # recover from partial page writes

shared_buffers = 6144MB    # min 128kB
work_mem = 1024MB          # min 64kB
```

Asinkroni commit ubrzava propusnost za manje transakcije, ali povećava rizik od gubitka najnovijih transakcija u slučaju rušenja baze podataka. Ubrzava transakcije eliminirajući potrebu čekanja da se Write-Ahead Logging (WAL) zapisi spremaju na disk prije potvrde uspjehu o zapisa. Način commita po transakciji moguće je kontrolirati putem parametra `synchronous_commit`, omogućujući ravnotežu između performansi i trajnosti transakcije. Isključivanje parametra `full_page_writes` ubrzava normalan rad baze podataka, ali može uzrokovati tiho ili nepovratno oštećenje podataka u slučaju kvara sustava. Rizik je da bi u slučaju nestanka struje ili pada sustava to moglo oštetiti podatke koji su nepovratno oštećeni. Posljedično, onemogućavanje ovog parametra poželjno je ako se cijela baza može rekreirati koristeći vanjske podatke. Parametar `shared_buffers` konfigurira količinu memorije dodijeljenu za međuspremnike dijeljene memorije. Njegova zadana vrijednost je 32 MB, iako postavke kernela mogu uzrokovati promjenu. Parametar mora biti postavljena na 128 KB ili više. Postavljanje parametra na 25% sistemske memorije preporučuje se za optimalnu izvedbu, posebno na namjenskim poslužiteljima baze podataka s 1 GB ili više RAM-a.Međutim, budući da se PostgreSQL oslanja na predmemoriju OS-a, izdvajanje više od 40% memorije rijetko poboljšava brzinu. Parametar `work_mem` određuje maksimalnu memoriju za operacije prije korištenja prostora na disku. Vi-

šestruke sesije mogu uvelike povećati ukupnu upotrebu memorije, a složeni upiti mogu izvršiti više operacija odjednom, od kojih svaka može koristiti točno vrijednost parametra `work_mem` prije nego što se prebaci na disk.

```
psql -c 'create table raw_data
    (id SERIAL, smiles text, emol_id integer, parent_id integer)' rdkit

zcat version.smi.gz |
sed '1d; s/\\/\\\\/g' |
psql -c
"copy raw_data(smiles,emol_id,parent_id)
    from stdin with delimiter ' '" rdkit
COPY 18238702
```

Nakon konfiguracije baze podataka, iznad navedenim naredbama kreira se pomoćna tablica `raw_data` sa sirovim podaci o kemijskim spojevima, koji su spremljeni kao `text` tip podataka. Sljedeći korak uključuje kreiranje nove tablice `mol` koja će sadržavati podatke o spojevima kao `mol` tip podataka korištenjem `mol_from_smiles` funkcije koja transformira SMILES tekst. Tablica `mol` sadržavat će samo spojeve iz `raw_data` tablice koje RDKit prihvaća. Naposljetku kreira se GiST indeks. Generalizirano stablo pretraživanja, ili GiST, je uravnotežena, prilagodljiva struktura stabla dizajnirana za podršku različitih tehnika indeksiranja, kao što su R- i B+-stabla. Njegova je glavna prednost što uklanja potrebu za poznavanjem baze podataka dopuštajući stručnjacima za određene vrste podataka stvaranje jedinstvenih tipova podataka i metoda pristupa.

```
SELECT *
INTO mols
FROM (
    SELECT id,
        mol_from_smiles(smiles::cstring) m
    FROM raw_data
    ) tmp
WHERE m IS NOT NULL;
SELECT 18208222
Time: 6674770.496 ms (01:51:14.770)

CREATE INDEX molidx ON mols USING GiST(m);
CREATE INDEX
Time: 6272983.677 ms (01:44:32.984)
```

Nakon punjenja baze podataka s podacima o molekulama, preostaje omogućiti udaljeni pristup bazi kako bi se aplikacija povezala na bazu. U već spomenutoj datoteci `postgresql.conf`

postavlja se parametar `listen_addresses` na `*` naređujući PostgreSQLu da prima konekcija sa svih IP adresa. Nadalje, u datoteci `pg_hba.conf` potrebno je dodati novi redak, prikazan niže, kako bi se dopustila veza s određene IP adrese koristeći MD5 autentifikaciju kao način authorizacije.

```
host  all       all       172.29.64.1/32        md5
```

3.3 MolAPI

Aplikacija MolApi predstavlja servis koji integrira funkcionalnosti RDKit Cartridge-a u web okruženje, omogućavajući korisnicima pristup naprednim alatima za kemoinformatiku putem standardnih web protokola.MolApi omogućava laku integraciju s različitim klijentskim aplikacijama, uključujući web aplikacije, mobilne aplikacije i softverske alate, nudeći fleksibilan i pristupačan način za izvođenje složenih kemijskih pretraga, analiza i transformacija molekula. Za razvoj i implementacija servisa koršteno je razvojno okruženju IntelliJ. IntelliJ IDEA razvojno okruženje namijenjeno je razvoju Java aplikacija te pruža potrebnu programsku i korisničku podršku za rad kako s Java web aplikacijama tako i sa Spring Boot programskim okvirom. Projekt je kreiran pomoću Spring Initializr-a, alata za brzo postavljanje i konfiguraciju Spring projekata koji generira osnovnu strukturu projekta i preuzima potrebne ovisnosti. Alat nudi odabir build alata, verzije Spring Boot-a i Jave, vrstu paketiranja i željene ovisnosti te zahtijeva unos metapodataka o projektu. MolAPI aplikacija korisiti Spring Boot 3.0.2 s Javom 17 te Maven kao build alat koji će graditi JAR vrstu paketa. Ovisnosti aplikacije definirane su u `pom.xml` datoteci projekta.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
```

```
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.10.1</version>
    </dependency>
</dependencies>
```

Jednostavnu konfiguraciju povezivanja na bazu podataka u Spring Boot aplikacijama omogućuje Spring Data JPA putem uređivanja `application.properties` ili `application.yml` datoteke. Datoteka omogućava centralizirano upravljanje konfiguracijskim postavkama aplikacije, uključujući detalje za povezivanje s bazom podataka.

```
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show_sql=true

spring.datasource.url=
jdbc:postgresql://${DATABASE_IP_ADDRESS}:${DATABASE_PORT}/${DATABASE_NAME}
spring.datasource.username=${USER}
spring.datasource.password=${PASSWORD}
```

Opcija `spring.jpa.hibernate.ddl-auto=none` u iznad prikazanoj datoteci onemogućava Hibernate-u (Java programskom okviru otvorenog koda za mapiranje objekata u bazu podataka koji dolazi s Spring Data JPA) automatsku obradu strukture baze podataka prilikom pokretanja aplikacije. Hibernate neće pokušavati kreirati, modificirati ili brisati tablice baze podataka što je iznimno korisno u produkcijskim okruženjima gdje je potrebna potpuna kontrola nad strukturom baze podataka. Opcije `spring.jpa.properties.hibernate.format_sql` i `spring.jpa.properties.hibernate.show_sql` omogučavaju formatiranje i ispisivanje svakog SQL upita koji Hibernate generira na konzolu. Ova opcija vrlo je korisna pri otklanjanju pogrešaka aplikacije, ali u produkcijskom okruženju može negativno utjecati na performanse i veličinu log datoteka. Opcija `spring.datasource.url` definira JDBC URL za povezivanje s PostgreSQL bazom podataka. Varijable `${DATABASE_IP_ADDRESS}`, `${DATABASE_PORT}` i `${DATABASE_NAME}` koriste se za ubacivanje IP adrese, porta i naziva baze podataka, što omo-

gućava fleksibilnost i olakšava konfiguraciju u različitim okruženjima. Varijable `${USER}` i `${PASSWORD}` sadržavaju vrijednosti kreirane u prethodnom poglavlju, prilikom kreiranja baze podataka i korisnika. U prethodnom poglavlju također je kreirana tablica s podacima o molekulama stoga je potrebno kreirati Java klasu koja će predstavljati entitete.

```
package hr.uniri.molapi.model;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Table(name = "mols")
@Getter
@Setter
@ToString
@AllArgsConstructor
@NoArgsConstructor
public class Mol {
  @Id
  @Column(name = "id")
  @GeneratedValue(strategy = GenerationType.SEQUENCE,
      generator = "mols_generator")
  @SequenceGenerator(name = "mols_generator", s
      equenceName = "mol_seq", allocationSize = 1)
  private Integer id;

  @Column(name = "m")
  private String smiles;

  public Mol(String smiles) {
      this.smiles = smiles;
  }
}
```

U pozadini Spring Boot skenira sve klase koje koriste `@Entity` anotaciju i definira ih kao klase entiteta. Anotacija `@Table(name = "mols")` specificira naziv tablice u bazi podataka s kojom se ovaj entitet mapira omogućavajući precizno mapiranje na specifičnu tablicu, posebno ako naziv klase ne odgovara nazivu tablice u bazi podataka. Anotacije `@Getter`, `@Setter` i `@ToString` iz biblioteke Lombok automatski generiraju pripadne *getter*, *setter* i *toString* metode eliminirajući *boilerplate* kod. Anotacije `@AllArgsConstructor` i `@NoArgsConstructor` također

eliminiraju nepotreban kod generirajući konstruktor koji koristi sve atribute i prazan konstruktor. Atribut `id` označen je s više anotacija: `@Id` koji označava da se atribut koristi kao primarni ključ, `@Column(name = "id")` mapira atribut na stupac *"id"*, `@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "mols_generator")` specificira da se vrijednost ovog polja u bazi podataka automatski sekvencijalno generira, dok `@SequenceGenerator(name = "mols_generator", sequenceName = "mol_seq", allocationSize = 1)` definira generator sekvence. Atribut `smiles` mapirano je na stupac *"m"*. Pošto će se u sklopu ovog rada koristiti samo jedna tablica odnosno prikazati će se sve funkcije RDKit Cartridgea nad jednim (`mol`) od dva novouvedena tipa podataka molekula, u aplikaciji MolAPI samo klasa `Mol` predstavlja entitet te je ujedno i jedina klasa koja se nalazi u paketu `model`. Ostale datoteke u paketu su enumeracije RDKit Cartridge funkcija.

```
src
\-- main
    |-- java
    |  -- hr
    |      -- uniri
    |          -- molapi
    |               |-- MolApiApplication.java
    |               |-- fingerprint
    |               |   |-- generate
    |               |   |   |-- FpGenerateController.java
    |               |   |   |-- FpGenerateRepository.java
    |               |   |   |-- FpGenerateRepositoryImpl.java
    |               |   |   |-- FpGenerateRequest.java
    |               |   |   |-- FpGenerateService.java
    |               |   |  -- FpGenerateServiceImpl.java
    |               |   |-- io
    |               |   |   |-- FpInputOutputController.java
    |               |   |   |-- FpInputOutputRepository.java
    |               |   |   |-- FpInputOutputRepositoryImpl.java
    |               |   |   |-- FpInputOutputService.java
    |               |   |  -- FpInputOutputServiceImpl.java
    |               |  -- work
    |               |       |-- FpWorkController.java
    |               |       |-- FpWorkRepository.java
    |               |       |-- FpWorkRepositoryImpl.java
    |               |       |-- FpWorkRequest.java
    |               |       |-- FpWorkService.java
    |               |      -- FpWorkServiceImpl.java
```

```
|               |-- model
|               |   |-- Mol.java
|               |  -- enums
|               |       |-- ConnectivityDescriptor.java
|               |       |-- Descriptor.java
|               |       |-- FingerprintMethod.java
|               |       |-- MolFromFormat.java
|               |       |-- MolToFormat.java
|               |       |-- Operation.java
|               |       |-- Similarities.java
|               |      -- Validation.java
|               |-- molecule
|               |   |-- connectivityDescriptors
|               |   |   |-- MolConDescController.java
|               |   |   |-- MolConDescRepository.java
|               |   |   |-- MolConDescRepositoryImpl.java
|               |   |   |-- MolConDescRequest.java
|               |   |   |-- MolConDescService.java
|               |   |  -- MolConDescServiceImpl.java
|               |   |-- descriptors
|               |   |   |-- MolDescController.java
|               |   |   |-- MolDescRepository.java
|               |   |   |-- MolDescRepositoryImpl.java
|               |   |   |-- MolDescRequest.java
|               |   |   |-- MolDescService.java
|               |   |  -- MolDescServiceImpl.java
|               |   |-- io
|               |   |   |-- MolInputOutputController.java
|               |   |   |-- MolInputOutputRequest.java
|               |   |   |-- MolIoRepository.java
|               |   |   |-- MolIoRepositoryImpl.java
|               |   |   |-- MolIoService.java
|               |   |  -- MolIoServiceImpl.java
|               |   |-- mcs
|               |   |   |-- MolMcsRepository.java
|               |   |   |-- MolMcsRepositoryImpl.java
|               |   |   |-- MolMcsService.java
|               |   |   |-- MolMcsServiceImpl.java
```

```
|               |   |  -- MolMscController.java
|               |   |-- substructureOperations
|               |   |   |-- MolSubOpController.java
|               |   |   |-- MolSubOpRepository.java
|               |   |   |-- MolSubOpRepositoryImpl.java
|               |   |   |-- MolSubOpService.java
|               |   |  -- MolSubOpServiceImpl.java
|               |   |-- substructureSearch
|               |   |   |-- MolSubSearchController.java
|               |   |   |-- MolSubSearchRepository.java
|               |   |   |-- MolSubSearchRepositoryImpl.java
|               |   |   |-- MolSubSearchService.java
|               |   |  -- MolSubSearchServiceImpl.java
|               |  -- validation
|               |       |-- MolValidateController.java
|               |       |-- MolValidateRepository.java
|               |       |-- MolValidateRepositoryImpl.java
|               |       |-- MolValidateRequest.java
|               |       |-- MolValidateService.java
|               |      -- MolValidateServiceImpl.java
|               |-- other
|               |   |-- GeneralMolController.java
|               |   |-- GeneralMolJpaRepository.java
|               |   |-- GeneralMolRepository.java
|               |   |-- GeneralMolRepositoryImpl.java
|               |   |-- GeneralMolService.java
|               |   |-- GeneralMolServiceImpl.java
|               |  -- VersionRDKitController.java
|              -- utils
|                   |-- Const.java
|                   |-- ExecuteMethod.java
|                   |-- MolRowMapper.java
|                   |-- SimpleJdbcCallFactory.java
|                  -- TriFunction.java
-- resources
     |-- application.properties
     |-- static
    -- templates
```

Aplikacija koristi slojevitu arhitekturu s odvojenim slojevima za pristup podacima, logiku i izlaganje enpointova, odnosno za svaku skupinu RDKit Cartridge funkcionalnosti postoji zasebni paket u kojem se nalazi zasebni kontroler, servis i repozitorij sloj. Takva arhitektura predstavlja čest obrazac dizajna u razvoju softvera, posebno u Spring aplikacijama, koja razdvaja odgovornosti unutar aplikacije na jasno definirane slojeve. Ovakav pristup lakšava održavanje koda, testiranje i skaliranje aplikacije. Paketi u aplikaciji podjeljeni su skupine po uzoru na vrstu funkcionalnosti koje pruža RDKit Cartridge.

3.3.1 Molekularni otisci

Molekularni otisci ključni su kemoinformatički alati za virtualni pregled i mapiranje kemijskog prostora. Među različitim vrstama otisaka, substrukturni otisci najbolje funkcioniraju za male molekule kao što su lijekovi, dok su otisci atomskih parova poželjniji za velike molekule kao što su peptidi [14].

Generiranje

Kao što i sam naziv paketa sugerira, `generate` izlaže endpoint preko kojeg se može pristupiti RDKit Cartridge funkcionalnosti za izradu molekularnih otisaka:

- POST /fingerprint/generate

```
@RestController
@RequestMapping("fingerprint")
public class FpGenerateController {
  private final FpGenerateService fpGenerateService;

  @Autowired
  public FpGenerateController(FpGenerateService fpGenerateService) {
      this.fpGenerateService = fpGenerateService;
  }

  @PostMapping("/generate")
  public ResponseEntity<PGobject> generate(
      @RequestBody final FpGenerateRequest request) {
      return ResponseEntity.ok().body(fpGenerateService.generate(request));
  }
}

public class FpGenerateRequest {
    private String smiles;
```

```
    private String method;
    private Integer radius;
}
```

Anotacija `@RestController` kombinira anotacije `@Controller` i `@ResponseBody` označavajući da se radi o klasi kontrolera koji je spreman obraditi web zahtjev i da će svaka metoda automatski serijalizirati povratne objekre u HTTP odgovore. `@RequestMapping` denfinira URL za sve metode unutar ovog kontrolera. Anotacija `Autowired` iznad konstruktora označava da Spring treba ubaciti (dependency injection) instancu `FPGenerateService` klase prilikom kreiranja kontrolera. Metoda `generate` prihvaća `FpGenerateRequest` objekt kao tijelo zahtjeva radi oznake `@RequestBody`. Metoda će zahtjev odbiti ukoliko ne sadržava `FpGenerateRequest` objekt u tijelu zahtjeva. Tip povratne vrijednosti je `ResponseEntity<PGobject>` što omogućava manupulaciju HTTP odgovorima, kao što je statusni kod i tijelo odgovora, dok `PGobject` sugerira da se radi o tipu objekta koji se koristi za rad s PostgreSQL bazom podataka. Metoda nadalje poziva `generate` metodu servisnog sloja prosljeđujući `FpGenerateRequest` objekt.

```
@Override
public PGobject generate(FpGenerateRequest request) {
  FingerprintMethod method =
      FingerprintMethod.valueOf(request.getMethod());
  Mol mol = new Mol(request.getSmiles());
  Integer radius =
      request.getRadius() != null ? request.getRadius() : DEFAULT_RADIUS;

  return switch (method) {
      case morgan_fp, morganbv_fp, featmorgan_fp, featmorganbv_fp ->
        execute(mol, radius, method.name(),
                                      fpGenerateRepository::generate);
      case rdkit_fp, atompair_fp, atompairbv_fp,
      torsion_fp, torsionbv_fp, layered_fp, maccs_fp ->
        execute(mol, method.name(), fpGenerateRepository::generate);
  };
}
```

Metoda `generate` servisnog sloja, ovisno o vrsti tražene metode, određuje koju metodu repozitorijskog sloja pozvati. Naime neke metode za generiranje molekularnih otisaka, osim same molekule, prihvaćaju dodatni parametar radijus, koji iznos 2 ako nije proslježen. Nadalje metoda poziva `generate` metodu repozitorijskog sloja. Važno je naglasiti da je klasa `FpGenerateServiceImpl` označena sa `@Service` anotacijom označavajući Spring Boot-u da se radi o servisnom sloju i što će rezultirati da prilikom skeniranja klasa kreaira bean, odnosno instancu

klase kako bi ubacivanje (depenency injection) ovisnosti bilo moguće.

```
@Override
public PGobject generate(Mol mol, Integer radius, String methodName) {
    return simpleJdbcCallFactory
            .getSimpleJdbcCall(methodName)
            .executeFunction(PGobject.class, mol.getSmiles(), radius);
}

@Override
public PGobject generate(Mol mol, String methodName) {
    return simpleJdbcCallFactory
            .getSimpleJdbcCall(methodName)
            .executeFunction(PGobject.class, mol.getSmiles());
}
```

Metode `generate` servisnog sloja zatim koriste `simpleJdbcFactory` za poziv funkcije nad bazom podataka. Metode vraćaju dobiveni rezultat u obliku `PGobject` objekta te se isti propagira do kotrolera gdje se vaća kao HTTP odgovor. Bitno je primjetiti da se za poziv funkcije koristi `simpleJdbcCallFactory` objekt. Tvornički dizajn obrazac koristi se za stvaranje objekata bez izravnog pozivanja konstruktora. Umjesto toga stvaranje objekta odvija se u posebnoj klasi `SimpleJdbcCallFactory`.

```
@Component
public class SimpleJdbcCallFactory {
  private final JdbcTemplate jdbcTemplate;
  private final Map<String, SimpleJdbcCall>
                          cache = new ConcurrentHashMap<>();

  @Autowired
  public SimpleJdbcCallFactory(JdbcTemplate jdbcTemplate) {
      this.jdbcTemplate = jdbcTemplate;
  }

  public SimpleJdbcCall getSimpleJdbcCall(String functionName) {
      return cache.computeIfAbsent(functionName, k -> createSimpleJdbcCall
      (functionName));
  }

  private SimpleJdbcCall createSimpleJdbcCall(String functionName) {
      // Configure and return a new SimpleJdbcCall instance
```

```
        return new SimpleJdbcCall(jdbcTemplate)
                .withSchemaName("emolecules")
                .withFunctionName(functionName);
    }
}
```

Potrebno je istaknuti da kad se `simpleJdbcCall` instancira, funkciju nad bazuom koju poziva više se ne može mijenjati. Ovakav pristup omogućuje centralizirano i efikasno upravljanje objektima `simpleJdbcCall`, smanjujući redudanciju i povečavajući performanse. Keširanjem instanci `simpleJdbcCall`, aplikacija smanjuje broja objekata koji treba stvoriti i konfigurirati, što dovodi do smanjenja opterećenja na *garbage collector* te korištenjem `ConcurrentHashMap` mape pruža siguran pristup cache-u u multithreaded okruženju izbjegavajući potencijalne probleme s konkurentnim threadovima.

Primjeri zahtjeva i odgovora

- morgan_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "CC(C)(C)OC(=O)N1C[C@@H](C(=O)O)[C@H](c2cccnc2)C1",
    "method": "morgan_fp",
    "radius": 3
}'
http://172.29.64.1:8080/fingerprint/generate

{"type":"sfp",
"value":"\\x0100000004000000ffffffff37000000f71fab050100000040...",
"null":false}
```

- morganbv_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "Nc1nc(-c2ccccc2)c(CCC(=O)O)c(=O)[nH]1",
    "method": "morganbv_fp"
}'
http://172.29.64.1:8080/fingerprint/generate

{"type":"bfp",
```

"value":"\\x00000010000000000100010400000000000508000080000200024208O1..."
"null":false}

- featmorgan_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "O=C(O)CCSc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1",
    "method": "featmorgan_fp",
    "radius": 4
}'
http://172.29.64.1:8080/fingerprint/generate
```

{"type":"sfp",
"value":"\\x0100000004000000ffffffff43000000000000000300000001000000...",
"null":false}

- featmorganbv_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "O=c1[nH]c(SCc2cccc3ccccc23)nc(-c2ccccc2)c1Cc1ccccc1",
    "method": "featmorganbv_fp",
    "radius": 4
}'
http://172.29.64.1:8080/fingerprint/generate
```

{"type":"bfp",
"value":"\\x7501100050080c0000024100004000c20908000800000000c81060048...",
"null":false}

- rdkit_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "O=C(O)CSc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1",
    "method": "rdkit_fp"
}'
http://172.29.64.1:8080/fingerprint/generate
```

```
{"type":"bfp",
"value":"\\xbf8010581c01992a450379cee8a56d6c3a35aba7a2144050ef445712...",
"null":false}
```

- atompair_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "O=C(O)CSc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1",
    "method": "atompair_fp"
}'
http://172.29.64.1:8080/fingerprint/generate

{"type":"sfp",
"value":"\\x0100000004000000000800006800000004000000030000000b0000000...",
"null":false}
```

- atompairbv_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "CCOC(=O)CCc1c(-c2ccccc2)[nH]c(=S)[nH]c1=O",
    "method": "atompairbv_fp"
}'
http://172.29.64.1:8080/fingerprint/generate

{"type":"bfp",
"value":"\\x37000300101000000101000001001000003003000000000000003000007...",
"null":false}
```

- torsion_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "CCOC(=O)CCc1c(-c2ccccc2)nc(N)[nH]c1=O",
    "method": "torsion_fp"
}'
http://172.29.64.1:8080/fingerprint/generate

{"type":"sfp",
```

"value":"\\x0100000004000000000400001700000049000000010000005000001 00...",
"null":false}

- torsionbv_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "CCCCOC(=O)Cc1c(-c2ccccc2)nc(N)[nH]c1=O",
    "method": "torsionbv_fp"
}'
http://172.29.64.1:8080/fingerprint/generate
```

{"type":"bfp",
"value":"\\x013000100000030000000000000000000010000001000000000000 0...",
"null":false}

- layered_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "CCCSc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1",
    "method": "layered_fp"
}'
http://172.29.64.1:8080/fingerprint/generate
```

{"type":"bfp",
"value":"\\x164097f283919db0bd4aadd9838de40f30651804a37d1a8bd941176b1...",
"null":false}

- maccs_fp

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "smiles": "CCCCSc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1",
    "method": "layered_fp"
}'
http://172.29.64.1:8080/fingerprint/generate
```

{"type":"bfp",
"value":"\\x164097f283919db0bd4aadd9838de40f30651804a37d1a8bd323613...",

```
"null":false}
```

Učitavanje i izvoz

Paket `io` pruža rad sa molekularnim otiscima na način da omogućuje konstrukciju `bfp` otiska molekule iz binarnog zapisa otiska te konverziju binarnog zapisa otiska u `bfp` otiska. Vrijedi istaknuti da se kod konstrukcije bfp-a iz binarnog zapisa, klijentu vraća zapis kodiran Base64 vrstom širiranja.

```
@Override
public String bfpToBinaryText(String bfp) {
  byte[] byteArray = fpInputOutputRepository.bfpToBinaryText(bfp);
  return Base64.getEncoder().encodeToString(byteArray);
}
```

Paket izlaže sljedeće endpointve:

- POST fingerprint/io/bfpToBinaryText

```
curl -X POST
-H "Content-Type: text/plain"
-d '\\xeda90d0f12075823851ebae7a029215fb2498bdae3761892ad5d4ff2106131...'
http://172.29.64.1:8080/fingerprint/io/bfpToBinaryText

XHhlZGE5MGQwZjEyMDc1ODIzODUxZWJhZTdhMDI5MjE1ZmIyNDk4YmRhZTM3N4ODkNGZjM...
```

- POST fingerprint/io/bfpFromBinaryText

```
curl -X POST
-H "Content-Type: text/plain"
-d 'XHhlZGE5MGQwZjEyMDc1ODIzODUxZWJhZTdhMDI5MjE1ZmIyNDk4YmRhZTM3NjE4OTJ...'
http://172.29.64.1:8080/fingerprint/io/bfpFromBinaryText

{"type":"bfp",
"value":"\\x5848686c5a4745354d4751775a6a45794d4463314f44497a4a54a685...",
"null":false}
```

Rad s molekularnim otiscima

Rdkit Cartridge za rad s molekularnim otiscima pruža funkcije koje uspoređuju otiske dvaju molekularnih otisaka i računa njihovu sličnost. Sličnost molekularnih otisaka računa se funkcijama `tanimoto_sml` i `dice_sml` dostupnih na endpointu:

- POST fingerprint/similarity

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "fp1": "\\x0100000004000000ffffffff1c0000004be599060106a22010009...",
    "fp2": "\\x0100000004000000ffffffff2e0000003a39a10001000d02014df...",
    "similarity": "tanimoto_sml"
}'
http://172.29.64.1:8080/fingerprint/similarity

0.11206896551724138
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "fp1": "\\x0100000004000000ffffffff1c0000004be599060100200000b89f...",
    "fp2": "\\x0100000004000000ffffffff2e0000003a39a100010000000034df...",
    "similarity": "dice_sml"
}'
http://172.29.64.1:8080/fingerprint/similarity

0.20155038759689922
```

Osim računanja sličnosti, RDKit Cartridge posjeduje funkciju za računanje dužine, odnosno broja bitova u bitnom vektorskom otisku molekule, dostupnoj na endpointu:

- POST /fingerprint/size

```
curl -X POST
-H "Content-Type: text/plain"
-d '\\xeda90d0f12075823851ebae7a029215fb2498bdae3761892ad5d4fc2101317c...'
http://172.29.64.1:8080/fingerprint/size

2064
```

Štoviše, RDKit nudi mogućnost vršenja operacija zbrajanja i oduzimanja dvaju molekularnih otisaka. Navedene operacije vratit će novonastali otisak na endpointu:

-POST /fingerprint/operation

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "fp1": "\\x0100000004000000ffffffff2b0000003a39a1000192d0201000...",
    "fp2": "\\x0100000004000000ffffffff2c0000003a39a10001000004034df...",
```

```
    "operation": "add"
}'
http://172.29.64.1:8080/fingerprint/operation

{"type":"sfp",
"value":"\\x0100000004000000ffffffff320000003a39a1000200000202004...",
"null":false}
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "fp1": "\\x0100000004000000ffffffff2b00000302010000004034d..",
    "fp2": "\\x0100000004000000ffffffff2c0000002d02010000004034...",
    "operation": "subtract"
}'
http://172.29.64.1:8080/fingerprint/operation

{"type":"sfp",
"value":"\\x0100000004000000ffffffff140000004034df03f01ffffffff...",
"null":false}
```

Nadalje moguće je provjeriti jesu li svi elementi veći ili manji od danog argumenta na endpointovima:

- POST /fingerprint/allValuesLt

```
curl -X POST
-H "Content-Type: text/plain"
-d '\x0100000004000000ffffffff140000004034df0503000000ac63f01ffa9...'
http://172.29.64.1:8080/fingerprint/allValuesLt?lessThan=10

true
```

- POST /fingerprint/allValuesGt

```
curl -X POST
-H "Content-Type: text/plain"
-d '\x0100000004000000ffffffff140000004034df0503000000ac63ff6fcab...'
http://172.29.64.1:8080/fingerprint/allValuesGt?greaterThan=10

false
```

3.3.2 Molekule

Molekula je najmanja jedinica kemijske tvari koja može postojati neovisno, čuvajući kemijska svojstva te tvari. Sastoji se od dva ili više atoma koji su međusobno povezani kemijskim vezama [15]. Atomi unutar molekule mogu biti istog elementa, kao što je u slučaju kisika (O2), ili različitih elemenata, formirajući spojeve kao što je voda (H2O), gdje dva atoma vodika (H) dijele elektrone s jednim atomom kisika (O).

Učitavanje, izvoz i validacija

U kontekstu molekula, RDKit omogućuje učitavanje i izvoz molekularnih struktura, kao i njihovu validaciju. Molekulu, u sklopu RDKita moguće je kreirati na više načina. U prethodnim poglavljima spomenuto je kako se zapis molekule može kreirati koristeći standardne zapise za opisivanje kemijskih struktura molekula. RDKit proširuje način kreiranja zapisa molekula u bazu omogučujući kreiranje zapisa o molekulama putem JSON objekta, CTAB-a (blok zapis molekule), binarnog zapisa strukture molekula, korištenjem (Chemaxon Extended SMILES) i CXSMARTS (Chemaxon Extended SMARTS). Kreiranje molekula dostupno je na endpointu:

- POST molecule/molFrom

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "molecule" : "CCCCCCCCCCOC(=O)Cc1c(O)nc2ccccn2c1=O",
  "format" : "smiles"
}'
http://172.29.64.1:8080/molecule/molFrom

CCCCCCCCCCOC(=O)Cc1c(O)nc2ccccn2c1=O
```

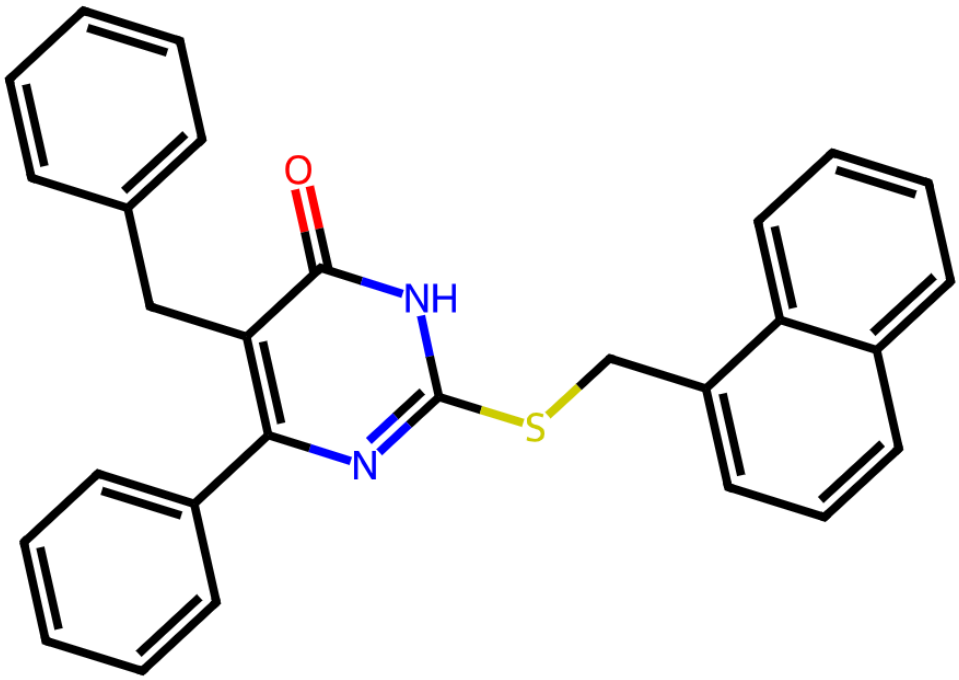
```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "molecule" : "[#6]-[#6](=[#7]-[#7](...)(-[#7]):[#6]:[#6]:1",
  "format" : "smarts"
}'
http://172.29.64.1:8080/molecule/molFrom

[#6]-[#6](=[#7]-[#7]-[#6]1:[#7]:[#6](-[#6]):[#6]2:[#6]:[#6](-[#6
]):[#8]:[#6]:2:[#7]:1)-[#6]1:[#6]:[#6]:[#6](-[#7]):[#6]:[#6]:1
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "molecule" : "\n  RDKit  2D\n\n 28 31  0 (...) 16  1  0\nM  END\n",
  "format" : "ctab"
}'
http://172.29.64.1:8080/molecule/molFrom

CC(C)CCC[C@@H](C)[C@H]1CC[C@H]2[C@@H]3CC=C4CC(Cl)CC[C@]4(C)[C@H]3CC[C@]12C
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule": {
    "defaults": {
        "atom": {...},
        "bond": {...}
    },
    "molecules": [
        {
            "atoms": [...],
            "bonds": [...],
            "extensions": [...]
        }
    ],
    "rdkitjson": {
        "version": 11
    }
},
    "format": "json"
}'
http://172.29.64.1:8080/molecule/molFrom

C=CCc1c(-c2ccccc2)nc(SCC(=O)O)[nH]c1=O
```

Osim učitavanja molekula koristeći razne formate, iste je moguće i izvoziti na endpointu:

- POST /molecule/molTo



Slika 3.1: Prikaz molekule u SVG formatu

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"O=c1[nH]c(SCc2cccc3ccccc23)nc(-c2ccccc2)c1Cc1ccccc1",
    "format": "svg"
}'
http://172.29.64.1:8080/molecule/molTo

<?xml version='1.0' encoding='iso-8859-1'?>
<svg>
...
</svg>
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"O=c1[nH]c(SCc2cccc3ccccc23)nc(-c2ccccc2)c1Cc1ccccc1",
    "format": "json"
}'
http://172.29.64.1:8080/molecule/molTo
```

```
{
"defaults": {
"molecules": [
    {
    "atoms": [...]
    "bonds": [...]
    "extensions": [
            {
            "aromaticAtoms": [...]
            "aromaticBonds": [...]
            "atomRings": [...]
            "formatVersion": 2,
            "name": "rdkitRepresentation",
            "toolkitVersion": "2024.03.1pre"
            }
        ]
    }
],
"rdkitjson": {
    "version": 11
}
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "format": "v3kctab"
}'
http://172.29.64.1:8080/molecule/molTo


     RDKit          2D

  0  0  0  0  0  0  0  0  0  0999 V3000
M  V30 BEGIN CTAB
M  V30 COUNTS 26 28 0 0 0
M  V30 BEGIN ATOM
M  V30 1 C 5.250000 -1.299038 0.000000 0
(...)
```

```
M  V30 26 O 1.500000 -2.598076 0.000000 0
M  V30 END ATOM
M  V30 BEGIN BOND
M  V30 1 1 1 2
(...)
M  V30 28 1 20 15
M  V30 END BOND
M  V30 END CTAB
M  END
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "format": "pkl"
}'
http://172.29.64.1:8080/molecule/molTo

\xefbeadde0000000010000000010000000000000001a0000001c00000080....
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "format": "cxsmarts"
}'
http://172.29.64.1:8080/molecule/molTo

[#6]-[#6](-[#16]-[#6]1:[#7]:[#6](...):[#7H]:1)-[#6](=[#8])-[#8]
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "format": "smarts"
}'
http://172.29.64.1:8080/molecule/molTo

[#6]-[#6](-[#16]-[#6]1:[#7]:[#6](....):[#7H]:1)-[#6](=[#8])-[#8]
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "format": "cxsmiles"
}'
http://172.29.64.1:8080/molecule/molTo

CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O
```

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "format": "smiles"
}'
http://172.29.64.1:8080/molecule/molTo

CC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O
```

Validirati strukture molekula u sklopu RDKita moguće je pomoću četiri funkcije `is_valid_smiles`, `is_valid_ctab`, `is_valid_smarts` i `is_valid_mol_pkl` koje se prosljeđuju parametrom `validateMethod` i dostupne su na endpointu:

- POST /molecule/validate

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"COc1ccc(CSc2nc(-c3ccccc3)c(Cc3ccccc3)c(=O)[nH]2)cc1Br",
    "validationMethod": "is_valid_smiles"
}'
http://172.29.64.1:8080/molecule/validate

true
```

Deskriptori

Molekularni deskriptori mogu se definirati kao matematički prikazi svojstava molekula koje generiraju algoritmi [16]. Numeričke vrijednosti molekularnih deskriptora koriste se za kvantitativno opisivanje fizikalnih i kemijskih informacija o molekulama. Deskriptori RDKit Cartrigde u aplikaciji MolAPI dostupni su na endpointu:

- POST /molecule/descriptor

Funkcija `mol_amw` računa prosječnu molekularnu težinu molekule.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"COc1ccc(CSc2nc(-c3ccccc3)c(Cc3ccccc3)c(=O)[nH]2)cc1Br",
    "descriptor": "mol_amw"
}'
http://172.29.64.1:8080/molecule/descriptor

493.426
```

Funkcija `mol_exactmw` računa točnu molekularnu težinu molekule.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_exactmw"
}'
http://172.29.64.1:8080/molecule/descriptor

408.15076
```

Funkcija `mol_logp` računa particijski koeficijent molekule.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_logp"
}'
http://172.29.64.1:8080/molecule/descriptor

4.7632
```

Funkcija `mol_tpsa` računa topološku polarnu površinu za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_tpsa"
```

```
}'
http://172.29.64.1:8080/molecule/descriptor

83.05
```

Funkcija `mol_labuteasa` vraća Labuteovu približnu površinu (ASA) za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_labuteasa"
}'
http://172.29.64.1:8080/molecule/descriptor

173.77988
```

Funkcija `mol_fractioncsp3` vraća udio ugljika koji su sp3 hibridizirani.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_fractioncsp3"
}'
http://172.29.64.1:8080/molecule/descriptor

0.26086956
```

Funkcija `mol_hba` vraća broj Lipinski akceptora H-veze (tj. broj Os i Ns) za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_hba"
}'
http://172.29.64.1:8080/molecule/descriptor

5.0
```

Funkcija `mol_hbd` vraća broj Lipinski donora H-veze (tj. broj Os i Ns koji imaju barem jedan H) za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_hbd"
}'
http://172.29.64.1:8080/molecule/descriptor

2.0
```

Funkcija `mol_numatoms` vraća ukupan broj atoma u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numatoms"
}'
http://172.29.64.1:8080/molecule/descriptor

53.0
```

Funkcija `mol_numheavyatoms` vraća broj teških atoma u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numheavyatoms"
}'
http://172.29.64.1:8080/molecule/descriptor

29.0
```

Funkcija `mol_numrotatablebonds` vraća broj rotirajućih veza u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numrotatablebonds"
}'
http://172.29.64.1:8080/molecule/descriptor
```

```
9.0
```

Funkcija `mol_numheteroatoms` daje broj heteroatoma u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numheteroatoms"
}'
http://172.29.64.1:8080/molecule/descriptor

6.0
```

Funkcija `mol_numrings` vraća broj prstenova u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numrings"
}'
http://172.29.64.1:8080/molecule/descriptor

3.0
```

Funkcija `mol_numaromaticrings` vraća broj aromatskih prstenova u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numaromaticrings"
}'
http://172.29.64.1:8080/molecule/descriptor

3.0
```

Funkcija `mol_numsaturatedrings` vraća broj zasićenih prstenova u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
```

```
    "descriptor": "mol_numsaturatedrings"
}'
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Funkcija `mol_numaromaticheterocycles` daje broj aromatskih heterocikla u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numaromaticheterocycles"
}'
http://172.29.64.1:8080/molecule/descriptor

1.0
```

Funkcija `mol_numaliphaticheterocycles` vraća broj alifatskih (barem jedna nearomatska veza) heterocikla u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numaliphaticheterocycles"
}'
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Funkcija `mol_numsaturatedheterocycles` vraća broj zasićenih heterocikla u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numsaturatedheterocycles"
}'
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Funkcija `mol_numaromaticcarbocycles` vraća broj aromatskih karbocikla u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numaromaticcarbocycles"
}'
http://172.29.64.1:8080/molecule/descriptor

2.0
```

Funkcija `mol_numaliphaticcarbocycles` vraća broj alifatskih (barem jedna nearomatska veza) karbocikla u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numaliphaticcarbocycles"
}'
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Funkcija `mol_numsaturatedcarbocycles` vraća broj zasićenih karbocikla u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numsaturatedcarbocycles"
}'
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Funkcija `mol_numspiroatoms` vraća broj spiro atoma u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
    "molecule":"CCCCC(Sc1nc(-c2ccccc2)c(Cc2ccccc2)c(=O)[nH]1)C(=O)O",
    "descriptor": "mol_numspiroatoms"
}'
```

```
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Funkcija `mol_numbridgeheadatoms` vraća broj atoma mosta u molekuli.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "molecule":"COc1ccc(CSc2nc(-c3ccccc3)c(Cc3ccccc3)c(=O)[nH]2)cc1[N+](=O)[O-]",
  "descriptor": "mol_numbridgeheadatoms"
}'
http://172.29.64.1:8080/molecule/descriptor

0.0
```

Podatke o jedinstvenim identifikator za kemijske spojeve, koji je razvijen od strane Međunarodne unije za čistu i primijenjenu kemiju moguće je na endpointovima:

- POST /molecule/inchi
- POST /molecule/inchiKey

Funkcija `mol_inchi` vraća InChI za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "smiles":"COc1ccc(CSc2nc(-c3ccccc3)c(Cc3ccccc3)c(=O)[nH]2)cc1[N+](=O)[O-]"
}'
http://172.29.64.1:8080/molecule/inchi

InChI=1S/C25H21N3O4S/c1-32-22-13-12-18(15-21(22)28(30)31)16-33-25-26-2
(19-10-6-3-7-11-19)20(24(29)27-25)14-17-8-4-2-5-9-17/h2-13,15H,14,16H2,1H3,
(H,26,27,29)
```

Funkcija `mol_inchikey` vraća InChI ključ za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "smiles":"COc1ccc(CSc2nc(-c3ccccc3)c(Cc3ccccc3)c(=O)[nH]2)cc1[N+](=O)[O-]"
}'
http://172.29.64.1:8080/molecule/inchiKey
```

```
HPLXKQPLNRFVLE-UHFFFAOYSA-N
```

Deskriptori povezanosti

Deskriptori povezanosti, također poznati kao topološki indeksi, vrsta su molekularnih deskriptora koji se koriste u poljima kemijske teorije grafova, molekularne topologije i matematičke kemije. Ovi deskriptori pružaju informacije o topologiji kemijskog spoja na temelju njegovog molekularnog grafa. Deskriptori povezanosti RDKit Cartrigde u aplikaciji MolAPI dostupni su na endpointu:

- POST /molecule/connectivityDescription

Funkcija `mol_chiXv` vraća vrijednost ChiXv za molekulu gdje X može biti raspona od 0 do 4 odnosno potrebno je proslijediti odgovarajući naziv metode u zahtjevu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O",
  "connectivityDescriptor": "mol_chi0v"
}'
http://172.29.64.1:8080/molecule/connectivityDescription?

16.569637
```

Funkcija `mol_chiXn` vraća vrijednost ChiXn za molekulu gdje X može biti raspona od 0 do 4 odnosno potrebno je proslijediti odgovarajući naziv metode u zahtjevu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O",
  "connectivityDescriptor": "mol_chi2n"
}'
http://172.29.64.1:8080/molecule/connectivityDescription

6.109051
```

Funkcija `mol_kappaX` vraća vrijednost kappaX za molekulu gdje X može biti raspona od 0 do 3 odnosno potrebno je proslijediti odgovarajući naziv metode u zahtjevu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
```

```
  "smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O",
  "connectivityDescriptor": "mol_kappa3"
}'
http://172.29.64.1:8080/molecule/connectivityDescription

4.7232943
```

Funkcija `mol_phi` vraća vrijednost Kier Phi za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O",
  "connectivityDescriptor": "mol_phi"
}'
http://172.29.64.1:8080/molecule/connectivityDescription

6.456631
```

Funkcija `mol_hallkieralpha` vraća Hall-Kier alfa vrijednost za molekulu.

```
curl -X POST
-H "Content-Type: application/json"
-d '{
  "smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O",
  "connectivityDescriptor": "mol_hallkieralpha"
}'
http://172.29.64.1:8080/molecule/connectivityDescription

-3.39
```

Maksimalna zajednička struktura

Maksimalna zajednička struktura (MCS) definirana je kao najveća substruktura koja se pojavljuje u dvije ili više molekula.

- POST /molecule/mcs/fmcs

```
curl -X POST
-H "Content-Type: application/json"
-d '[
  {"smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O"},
  {"smiles": "C=CCn1c(SCC(=O)Nc2ccc(Cl)cc2)nc2sc3c(c2c1=O)CC(C(C)C)OC3"}
```

```
]'
http://172.29.64.1:8080/molecule/mcs/fmcs

[#6]-,:[#6](:[#6]-[#6]:,-[#6]:,-[#6]:,-[#6]):[#6](...)=[#8]
```

Pretraga po substrukturi

Pretraga molekula po substrukturi je proces identifikacije molekula koje sadrže specifičnu strukturnu komponentu ili uzorak unutar veće molekulske strukture. RDKit Cartridge nudi dvije funkcije pretrage po strukturi `substruct` i `substruct_count`. Prva funkcija prima dvije molekule te vraća je li prva molekula substruktura druge.

- POST /molecule/substructure

```
curl -X POST
-H "Content-Type: application/json"
-d '[
  {"smiles": "C=CCc1c(-c2ccccc2)nc(SCc2ccc(OC)c([N+](=O)[O-])c2)[nH]c1=O"},
  {"smiles": "C=CCn1c(SCC(=O)Nc2ccc(Cl)cc2)nc2sc3c(c2c1=O)CC(C(C)C)OC3"}
]'
http://172.29.64.1:8080/molecule/substructure

false
```

Druga funckija vraća broj podudaranja podstrukture između druge i prve molekule te prihvaća treći argument označavajući jesu li podudaranja jedinstvena. Zadana vrijednost jedinstvenog podudaranja je `true`.

- POST /molecule/substructureCount

```
curl -X POST
-H "Content-Type: application/json"
-d '[
{"smiles": "CCCCCSc1nc(C)cc(O)n1"},
{"smiles": "COc1ccc(Cn2c(=O)cnn([C@@H]3OCC(OC(C)=O)c2=O)cc1[N+](=O)[O-]"}
]'
http://172.29.64.1:8080/molecule/substructureCount?uniquified=false

0
```

Poglavlje 4

Zaključak

Kroz diplomski rad, prikazano je kako softveri otvorenog koda mogu unaprijediti pohranu i analizu molekularnih podataka, čime može pridonijeti napretku u području kemoinformatike i u širem znanstvenom i istraživačkom kontekstu. Rad demonstrira uspješnu integraciju RDKit-a unutar PostgreSQL baze podataka i kako se REST API može koristiti za pristup i upravljanje molekularnim podacima, omogućavajući tako integraciju s različitim aplikacijama i servisima. Rad ističe važnost i efikasnost korištenja naprednih funkcionalnosti baza podataka i programskih okvira u obradi specijaliziranih podataka kao što su molekule te pruža temelje za daljnje razvijanje prikazane aplikacje. Primjena predloženog rješenja može imati širok spektar upotrebe, od farmaceutskih istraživanja do akademske edukacije, gdje je potrebna detaljna analiza i pohrana molekularnih podataka.

Literatura

[1] V. Miletić, "Kemoinformatika i računalna kemija." 01. veljače 2024. Available: https://group.miletic.net/hr/nastava/materijali/kemoinformatika-povijest-osnove/

[2] M. Lipovac, "Metodama računalne kemije moguće je dobiti precizan uvid u ponašanje i karakteristike promatranih sustava." 05. veljače 2024. Available: https://www.hrvatski-fokus.hr/2017/12/14401/

[3] "Toggle the table of contents simplified molecular-input line-entry system." 8. veljače 2024. Available: https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system

[4] "SMILES arbitrary target specification." 8. veljače 2024. Available: https://en.wikipedia.org/wiki/SMILES_arbitrary_target_specification

[5] "International chemical identifier." 8. veljače 2024. Available: https://en.wikipedia.org/wiki/International_Chemical_Identifier

[6] G. Landrum and other RDKit contributors, "An overview of the RDKit." 02. veljače 2024. Available: https://www.rdkit.org/docs/Overview.html

[7] "About PostgreSQL." 8. veljače 2024. Available: https://www.postgresql.org/about/

[8] "Java (programming language)." 8. veljače 2024. Available: https://en.wikipedia.org/wiki/Java_(programming_language)

[9] "Write once, run anywhere." 8. veljače 2024. Available: https://en.wikipedia.org/wiki/Write_once,_run_anywhere)

[10] "Oak (programming language)." 8. veljače 2024. Available: https://en.wikipedia.org/wiki/Oak_(programming_language)

[11] "What is java?" 8. veljače 2024. Available: https://www.ibm.com/topics/java

[12] M. Behler, "What is spring framework?" 8. veljače 2024. Available: https://www.marcobehler.com/guides/spring-framework

[13] "Spring framework." 8. veljače 2024. Available: https://spring.io/projects/spring-framework

[14] D. P. &. J. R. A. Capecchi, "One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome," *Journal of Cheminformatics*, vol. 12, no. 43, pp. 1–1, 2020.

[15] "Molekula." 8. veljače 2024. Available: `https://hr.wikipedia.org/wiki/Molekula`

[16] "Molecular descriptor." 8. veljače 2024. Available: `https://www.sciencedirect.com/topics/medicine-and-dentistry/molecular-descriptor`