



POLITECHNIKA WARSZAWSKA

Wydział Elektroniki i Technik Informacyjnych

Instytut Informatyki

Michał Haponiuk

nr albumu: 249371

PRACA DYPLOMOWA INŻYNIERSKA

Przeglądarka genomu

Praca wykonana pod kierunkiem

dr hab. inż. Robert Nowak

Warszawa, 27 stycznia 2016



Imię i nazwisko:

Michał Haponiuk

Kierunek:

Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia:

3 kwietnia 1992

Data rozpoczęcia studiów:

20 luty 2012

Życiorys

Urodziłem się 3 kwietnia 1992 roku w Kielcach. W 2008 roku rozpoczęłem naukę w II Liceum Ogólnokształcącym im. Jana Śniadeckiego w Kielcach, w klasie o podstawie programowej w zakresie rozszerzonym z matematyki, informatyki, fizyki i astronomii . W 2011 roku uzyskałem wykształcenie średnie oraz świadectwo dojrzałości. W następnym roku zostałem przyjęty na studia stacjonarne pierwszego stopnia, na Wydziale Elektroniki i Technik Informatycznych, kierunku Informatyka. Po dwóch latach studiów, obrałem specjalność Inżynierii Systemów Informatycznych.

.....
Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu 2016 r.

z wynikiem

Ogólny wynik studiów:

Dodatkowe wnioski i uwagi Komisji:

.....

.....

Streszczenie

Przeglądarka genomu

Celem niniejszej pracy inżynierskiej było utworzenie aplikacji pozwalającej na przechowywanie i interaktywne przeglądanie danych genetycznych pozyskiwanych z sekwencjonowania genomów. Praca rozpoczyna się wprowadzeniem czytelnika w świat bioinformatyki i zagadnień związanych z przeglądarkami genomów. Następnie został opisany proces projektowania i implementacji, oraz omówione zostały użyte technologie i narzędzia, ułatwiające realizację programu. Oprogramowanie było tworzone z myślą o możliwości dalszego rozszerzania i rozwoju.

Słowa kluczowe: przeglądarka genomów, bioweb, bioinformatyka, sekwencje

Abstract

Genome viewer

The purpose of this thesis was to create an application for storing and interactive browsing genetic data obtained from genome sequencing. The work begins with an introduction the reader into the world of bioinformatics and issues related to genome browsers. Thereafter, it was described the process of design and implementation. It discusses the technologies and tools which simplify accomplishment of the program. The software was written specifically about the possibility of further development.

Keywords: genome viewer, bioweb, bioinformatics, sequences

Spis treści

| | |
|---------------------------------------|----|
| 1. Cel i zakres pracy | 5 |
| 1.1 Układ pracy | 5 |
| 2. Przeglądarki genomów | 7 |
| 2.1 Mapowanie genomów | 7 |
| 2.1.1 Mapy genetyczne | 8 |
| 2.1.2 Mapy fizyczne | 9 |
| 2.1.3 Pojęcia | 9 |
| 2.2 Historia | 10 |
| 2.3 Oprogramowanie i narzędzia | 12 |
| 2.3.1 Bazy danych | 12 |
| 2.3.2 Usługi internetowe | 13 |
| 2.4 Najpopularniejsze przeglądarki | 13 |
| 2.4.1 Dostęp | 13 |
| 2.4.2 Architektura | 13 |
| 2.4.3 Istniejące rozwiązania | 14 |
| 2.5 Porównanie | 17 |
| 3. Projekt i implementacja | 18 |
| 3.1 Wymagania | 19 |
| 3.1.1 Wymagania funkcjonalne | 19 |
| 3.1.2 Wymagania niefunkcjonalne | 20 |
| 3.2 Architektura | 21 |
| 3.3 Warstwa trwałości | 22 |
| 3.3.1 PostgreSQL | 23 |
| 3.3.2 ORM | 23 |
| 3.4 Warstwa przetwarzania | 23 |
| 3.4.1 Python | 23 |
| 3.4.2 Django | 26 |
| 3.4.3 Gunicorn | 27 |
| 3.4.4 Nginx | 28 |

| | | |
|-----------|---|-----------|
| 3.5 | Warstwa prezentacji | 28 |
| 3.5.1 | HTML5 | 29 |
| 3.5.2 | JavaScript | 30 |
| 3.5.3 | AngularJS | 30 |
| 3.5.4 | Bootstrap | 31 |
| 3.6 | Inne narzędzia | 31 |
| 3.6.1 | Zintegrowane środowisko programistyczne | 31 |
| 3.6.2 | System kontroli wersji | 32 |
| 3.7 | Szczegóły implementacyjne | 33 |
| 3.7.1 | Dane | 34 |
| 3.7.2 | Budowanie i wdrażanie aplikacji | 35 |
| 3.7.3 | Import i export danych | 36 |
| 3.7.4 | Trasowanie | 37 |
| 3.8 | Testowanie | 37 |
| 3.8.1 | Testy jednostkowe | 38 |
| 3.8.2 | Testy funkcjonalne | 39 |
| 4. | Demonstracja programu | 42 |
| 5. | Podsumowanie | 45 |
| | Bibliografia | 47 |

1. Cel i zakres pracy

Informatyka jest aktualnie bardzo rozległą dziedziną nauki, z osiągnięć której szeroko korzystają inne dyscypliny naukowe a wśród nich genomika - nauka zajmująca się analizą genomu, której celem jest poznanie sekwencji materiału genetycznego, mapowanie genomu oraz określenie wszelkich zależności i interakcji wewnątrz genomu.

Cechy każdego żywego organizmu zapisane są na nośniku informacji genetycznej, którym w komórkach posiadających jądro komórkowe jest kwas dezoksyrybonukleinowy (DNA). DNA składa się z nukleotydów, których kolejność występowania warunkuje cechy organizmu. Kompletny zapis informacji genetycznej nosi nazwę genomu.

Dokładną budowę chemiczną z przestrzennym modelem cząsteczki DNA udało się opracować stosunkowo niedawno, bo w 1953 r. a odkrywcy - James D. Watson i Francis Circk, otrzymali Nagrodę Nobla. Jako pierwszy ukończono projekt poznania genomu bakteriofaga Φ -X174, mającego 5368 par zasad. Zsekwencjonował go Frederick Sanger w 1977 r. Pierwszy ukończony projekt dotyczący bakterii to projekt poznania genomu *Haemophilus influenzae* wykonany w TIGR (*The Institute for Genomic Research*) w 1995r. W 1988 roku grupa naukowców z 18 państw rozpoczęła badania nad ludzkim genomem w ramach projektu HGP (*Human Genome Project*). Wstępny opis genomu człowieka opublikowano w 2000 roku. Badacze odkrywają genomy także i innych organizmów (bakterii, roślin i zwierząt) a liczba ukończonych projektów ciągle wzrasta. Dostępność całych, często obszernych genomów organizmów, stworzyła potrzebę wizualizacji, zaspokajaną przez zastosowanie przeglądarki. Zaproponowana przeze mnie przeglądarka genomu wpisuje się w katalog istniejących projektów umożliwiając wysłanie i przeglądanie danych zlokalizowanych na komputerze użytkownika.

1.1 Układ pracy

Tekst składa się z pięciu rozdziałów. Na początku pracy zapoznaję czytelnika z podstawowymi zagadnieniami dotyczącymi bioinformatyki. W trakcie wprowadzenia w problematykę genomiki, opisuję pojęcia biologiczne bezpośrednio związane z projektowaną przeglądarką. Przedstawiam rys historyczny pokazujący jak rozwijał się przemysł bioinformatyczny w XX wieku. Poruszam kwestie technologii używanych w teraźniejszych przeglądarkach genomów

oraz przedstawiam rozwiązania dostępne na rynku. W kolejnym rozdziale opisuję wymagania funkcjonalne i niefunkcjonalne stawiane projektowanemu systemowi oraz opowiadam o zaprojektowanej architekturze i użytych technologiach. Omawiam również wybrane kluczowe decyzje implementacyjne. Czwarty rozdział zawiera demonstrację funkcjonalności aplikacji. Pracę kończy podsumowanie wraz z opisem potencjalnych możliwości rozwoju programu.

2. Przeglądarki genomów

Bioinformatyka jako dziedzina nauki łączy w sobie informatykę i matematykę w celu przetwarzania danych biologicznych. Dynamiczny rozwój biologii molekularnej oraz genomiki przyczynił się do powstania olbrzymiej ilości informacji. Konsekwencją jest utworzenie i zarządzanie rozwiniętymi bazami danych, w których mogą być gromadzone i sprawnie przeglądane dane biologiczne. Wykorzystywanie narzędzi bioinformatycznych istotnie redukuje czas projektowania i przeprowadzania doświadczeń oraz zmniejsza koszty. Na biologach została wywarta presja, aby w swojej pracy korzystali z komputerów i informatyki na etapach gromadzenia, wydobywania oraz przetwarzania informacji zawartych w DNA (rys.2.1), RNA i białkach. Cząsteczki te mogą być opisywane przez sekwencje symboli.



Rysunek 2.1: Prosty model DNA.

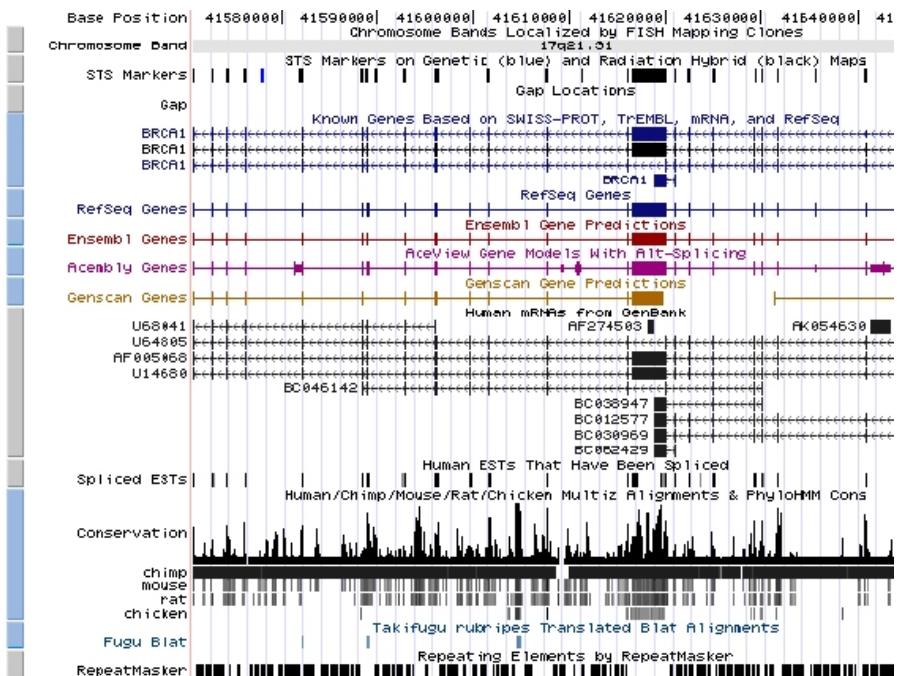
Źródło: <https://forensicdnaconsulting.files.wordpress.com/2013/07/canstockphoto2267476.jpg>

Przeglądarki genomów są narzędziem, które upraszczają i porządkują dane związane z genomem danego organizmu. Obecnie przeglądarki posiadają graficzny interfejs użytkownika. Umożliwiają naukowcom wizualizacje całych genomów z dodatkowymi informacjami pozywiskiwanymi zwykle z wielu źródeł¹. Różnią się one od zwykłych biologicznych baz danych, gdyż dane są opisane współrzędnymi w genomie. Rysunek 2.2 ilustruje typową wizualizację genomu dla przeglądarki UCSC (roz.2.4.3). Można przyjąć, że obrazuje się sekwencje wraz z adnotacjami.

2.1 Mapowanie genomów

Przeglądarka genomów opisuje położenie elementów w genomie, poczynając od chromosomów. Następnie ustalane jest położenie sekwencji lub adnotacji wewnątrz chromosomu

¹ Dane porównawcze (drzewa filogenetyczne, zmienności), laboratoryjne badania analityczne (sekwencje, skany), odnośniki do innych baz, przetworzone dane laboratoryjne.



Rysunek 2.2: Fragment typowej wizualizacji genomu przeglądarki UCSC. Wyświetlany zakres danych jest ułożony pionowo. Na samej górze, pokazany jest numer i pozycja na chromosomie. Kilka poniższych wierszy ilustruje dane dotyczące genów, które zostały doświadczalnie znalezione lub przewidziane różnymi metodami. Kolejne wiersze dotyczą poziomu ekspresji genów, a pod nimi znajduje się porównanie z genomami innych gatunków.

Źródło: https://genomics.soe.ucsc.edu/research/browser_overview

co tworzy mapę. Uwzględniając zjawisko rekombinacji rozróżniamy *mapy genetyczne*, wyrażające oddalenie między specyficznymi sekwencjami (markerami), oraz *mapy fizyczne*, określające pozycję genów w sposób bezpośredni, jako indeks. Rekordy zawarte w obu typach map uzupełniają się nawzajem, będąc podstawą do kolejnych badań nad ukształtowaniem i strukturą chromosomów.

2.1.1 Mapy genetyczne

Mapy genetyczne, wykorzystując analizę sprzężeń genetycznych, umożliwiają określenie rozłożenia genów i innych rozpoznawalnych markerów (sekwencji) w genomie oraz ustalając odległość genetyczną między nimi. Miara wyrażana jest w centymorganach² (cM).

² Jeden cM to taka odległość pomiędzy dwoma loci, że szansa na ich rozdzielenie w procesie rekombinacji genetycznej w ciągu jednego pokolenia (podczas jednorazowego wydarzenia, np. crossing-over) wynosi 1%. Centymorgany nie odzwierciedlają bezwzględnej odległości pomiędzy obszarami chromosomu zajmowanymi przez gen, ponieważ częstość rekombinacji jest różna w różnych rejonach chromosomów, na różnych chromosomach oraz w różnych organizmach.

2.1.2 Mapy fizyczne

Techniki biologii molekularnej są wykorzystywane w mapach fizycznych w celu bezpośredniego usytuowania poszczególnych sekwencji DNA w genomie. Jednostką w tych mapach są pary zasad³ (pz). Tablica 2.1 przedstawia mapę dwóch komplementarnych łańcuchów DNA. Oba mają długość 15 pz.

| | | | | | | | | | | | | | | | |
|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <i>Łańcuch 1</i> | A | T | C | G | A | T | T | G | A | G | C | T | C | T | A |
| <i>Łańcuch 2</i> | T | A | G | C | T | A | A | C | T | C | G | A | G | A | T |

Tablica 2.1: Mapa dwóch komplementarnych łańcuchów DNA.

2.1.3 Pojęcia

W pracy używam wybranych pojęć biologicznych:

- **Chromosom** jest strukturą zawierającą materiał genetyczny komórki. Zbudowany jest głównie z pojedynczej olbrzymiej liniowej cząsteczki DNA. Podczas podziału jądra chromosomy przyjmują zwartą strukturę i stają się widoczne w mikroskopie jako pojedyncze obiekty. Struktura chromosomu nie jest niezmienna, podlega on bowiem zmianom zwanym mutacjami⁴. [18, 20]

Komórki mogą być haploidalne, czyli zawierające po jednym chromosomie każdego typu, diploidalne, zawierające po dwa chromosomy danego typu, bądź poliploidalne - jeśli garnitur chromosomalny jest zwielokrotniony ponad dwukrotnie. Liczba chromosomów u różnych gatunków może być różna - od pojedynczych par aż do kilkuset par, ale zazwyczaj wynosi od kilku do kilkudziesięciu par.[18, 20]

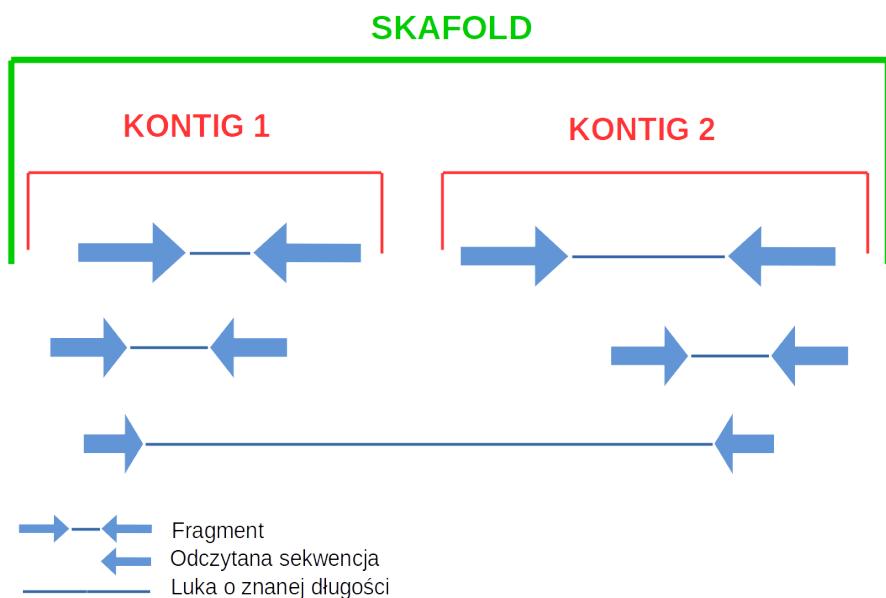
- **Kontig** to zbiór nakładających się na siebie fragmentów DNA. Ponieważ obecna technologia pozwala na bezpośrednie sekwencjonowanie tylko stosunkowo niedługich odcinków

³ Para zasad to dwie komplementarne zasady nukleotydów dwóch różnych nici kwasu nukleinowego, połączone wiązaniem wodorowym, zgodnie z następującą regułą: adenina(A) tworzy zawsze podwójne wiązanie wodorowe z tyminą (T) a cytozyną(C) tworzy potrójne wiązanie wodorowe z guaniną (G).

⁴ Mutacje genomowe powodują zaburzenia genetyczne lub zespoły chorobowe, takie jak zespół Downa czy zespół Turnera.

DNA, genom musi być podzielony na małe kawałki przed przystąpieniem do procesu sekwencjonowania. Zgrubnie koncepcja niektórych metod sekwencjonowania polega na podziale DNA na małe fragmenty sekwencji, następnie pogrupowaniu ich w kontigi, a na końcu montażu całego genomu.[19]

- **Skafold** w terminologii mapowania genomów jest to ciąg kontigów uporządkowanych w kolejności ale niekoniecznie związanych w jeden ciągły odcinek sekwencji. Zawiera sekwencje oddzielone lukami o znanej długości. [21]
- **Marker** to dowolny gen lub segment DNA o znanym położeniu na chromosomie, którego obecność można łatwo zidentyfikować. Markery genetyczne wykorzystywane są przy konstruowaniu map genetycznych jako swego rodzaju punkty odniesienia - aby ustalić położenie danego genu na mapie genetycznej, wystarczy ustalić odległość dzielącą go od dwóch markerów genetycznych.[20]



Rysunek 2.3: Nakładające się zbiorze sekwencji kontigów wraz z lukami w ujęciu skafoldu.

Źródło: https://upload.wikimedia.org/wikipedia/commons/6/6e/PET_contig_scaffold.png

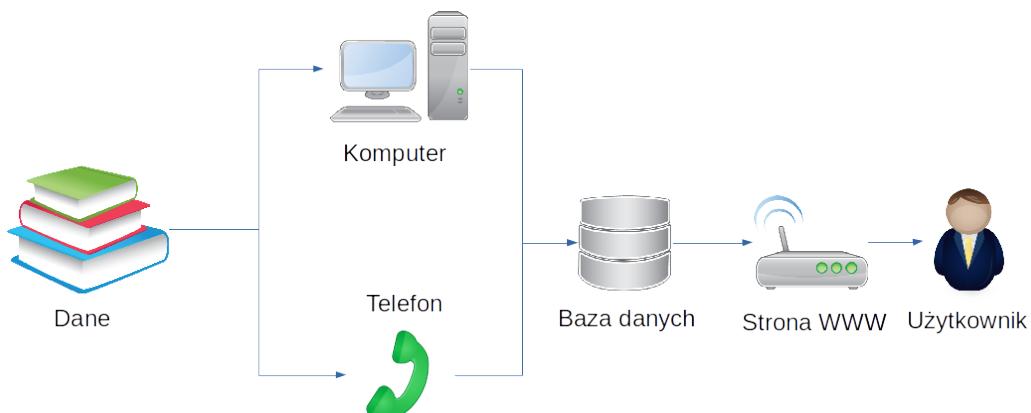
2.2 Historia

Załążki bioinformatyki sięgają lat osiemdziesiątych, kiedy to utworzono w Stanach Zjednoczonych bazę danych *GenBank*. Wraz z coraz częstszymi badaniami różnych organizmów i sekwencjonowaniem ich genomów przez badaczy, Amerykański Departament Energii zdecydował się na opracowanie ww. programu, aby gromadzić sekwencje DNA.

Początkowo użytkownicy wprowadzali sekwencje DNA do systemu używając specjalnych klawiatur posiadających jedynie cztery klawisze - A, C, T oraz G. Z biegiem czasu opracowano specjalne protokoły przekazywania danych, przyspieszające cały proces. Naukowcy mogli zadzwonić do *GenBanku* i używając komputera byli w stanie bezpośrednio wprowadzić dane o kolejnych fragmentach genomu do bazy (rys.2.4). Rozwój technologii internetowych diametralnie zmienił sposób przekazywania informacji pomiędzy ośrodkami naukowymi. Po utworzeniu serwisu WWW, naukowcy z całego świata otrzymali darmowy wgląd do zgromadzonych informacji.

Zarząd *GenBanku* przeniesiono do NCBI⁵ w National Institutes of Health. Gdy wystartował Projekt Poznania Ludzkiego Genomu, ilość danych odbierana przez *GenBank* rosła w tempie wykładniczym. Wraz z opracowaniem wydajnych metod sekwencjonowania wykorzystujących komputery, automatyczne sekwencjonery i zagadnienia robotyki, proces gromadzenia danych niesamowicie przyspieszył.

Równolegle prywatne firmy również zajęły się pokrewnymi badaniami tworząc własne ogromne zbiory danych. Aktualnie największe firmy potrafią w krótkim odstępie czasu (1 dnia) zdekodować dane genetyczne liczące nawet kilkadziesiąt milionów par zasad DNA. Ilość danych jaką uzyskano z pełnej sekwencji genomu ludzkiego to ponad 50 TB⁶



Rysunek 2.4: Historyczny schemat przepływu informacji o sekwencjach.

Źródło: <http://www.uwm.edu.pl/wnz/KBZ/Lipazy/obrazy/his.gif>

⁵ National Center for Biotechnology Information

⁶ Taka ilość danych zmieściłaby się na około 80 tys. płyt CD.

2.3 Oprogramowanie i narzędzia

Spektrum oprogramowania dostępnego dla bioinformatyki jest bardzo szerokie. Rozpoczyna się od prostych narzędzi z interfejsem terminalowym, a kończy na usługach internetywych i złożonych programach z możliwościami rozbudowanej interakcji z użytkownikiem.

Znaczna część narzędzi jest dostępna w formie darmowej lub otwartego oprogramowania. Następstwem popytu na nowe algorytmy przeznaczone do analizy był dynamiczny rozwój nieodpłatnych programów. Dzięki ideologii otwartego kodu źródłowego, grupy badawcze z całego świata mogą przyczynić się do tworzenia coraz to lepszych rozwiązań. Istotną zaletą darmowego oprogramowania jest budowanie standardów i integrowanie bioinformatycznych informacji.

2.3.1 Bazy danych

Bazy danych używane w przeglądarkach genomów zwykle są ogromnymi zbiorami informacji, w większości składającymi się z sekwencji. Ponieważ bazy danych są niezbędnym elementem w procesie badań naukowych, tworzenie ich jest jednym z najistotniejszych zagadnień bioinformatyki. Dane przechowywane w biologicznych bazach nie są niezmienne i nieodwołalne. Wszystkie sekwencje są wynikiem badań o określonej dokładności i w konsekwencji nie można wykluczyć, że zbiór danych nie zawiera nieprawidłowości.

Możliwość szybkiego i precyzyjnego wyszukiwania pożądanych informacji jest jedną z ważniejszych cech biologicznych baz danych. Najczęściej użytkownik oprócz obróbki danych ma również możliwość połączenia się przez internet z innymi bazami. Charakteryzują się one zazwyczaj także małą redundancją danych. By ułatwić pracę, badacze często mają możliwość wyboru sposobu wyświetlania wyszukanych danych oraz wydruku, bądź ich zapisu.

Pierwsze bazy danych zazwyczaj budowali biolodzy molekularni i biochemicy. W dzisiejszych czasach zwraca się uwagę, aby były one czytelne nie tylko dla specjalistów, ale i dla badaczy z nauk pokrewnych.

2.3.2 Usługi internetowe

Dla wielu zastosowań bioinformatycznych zostały utworzone interfejsy wykorzystujące protokoły SOAP⁷ i REST⁸. Dzięki takim rozwiązaniom, aplikacja działająca w jednej części świata, potrafi korzystać z algorytmów, danych i mocy obliczeniowej serwerów podpiętych do sieci w innych częściach globu. Brak obowiązku utrzymywania baz danych oraz zajmowania się oprogramowaniem stał się jednym z największych udogodnień. Usługi bioinformatyczne zostały zgrupowane w trzy podstawowe filary: dotyczące poszukiwania sekwencji, wielokrotnego przyrównywania sekwencji oraz analizy biologicznych sekwencji. Podział nastąpił z inicjatywy Europejskiego Instytutu Informatyki (EBI).

2.4 Najpopularniejsze przeglądarki

2.4.1 Dostęp

Obecnie dostępnych jest wiele przeglądarek. Dzięki ogólnoświatowej sieci komputerowej są szeroko używane przez badaczy. Czasem jednak możemy nie móc korzystać z przeglądarek *online*. Możliwe, że nie chcemy udostępniać danych innym użytkownikom, albo możemy dysponować zbyt dużą ich ilością, utrudniającą przesłanie na inny serwer. Rozwiązaniem problemu są przeglądarki *offline* wyświetlające dane z dysku, ale pobierające kontekst opisu genomu ze zdalnego serwera DAS⁹.

2.4.2 Architektura

Przeglądarki różnią się wewnętrzną organizacją. Jedne są scentralizowane, inne stawiają na rozproszony system przechowywania danych. Oba podejścia charakteryzują się wadami i zaletami, w zależności od rozpatrywanego punktu widzenia. Projekty zaprzegające dużą część społeczności naukowej wymagają wysiłku od dużej liczby osób, często powodując wygaszenie entuzjazmu w pracy oraz wymagają praktycznie ciągłej aktualizacji. Specjalizowane bazy

⁷ Simple Object Access Protocol - protokół komunikacyjny, korzystający z XML do kodowania wywołań.

⁸ Representational State Transfer - styl architektury oprogramowania wywiedziony z doświadczeń przy pisaniu specyfikacji protokołu HTTP dla systemów rozproszonych, wykorzystuje m.in. jednorodny interfejs oraz bezstanową komunikację.

⁹ *Distributed Annotation System* to protokół wymiany danych stworzony w celu publikowania oraz integracji danych biologicznych, w systemie rozproszonym.

danych z silną kuratelą zmagają się niekiedy z problemami finansowymi. Sprawowanie ciągłej kontroli nad systemem również opóźnia dostęp do danych. Praktyka pokazuje, że tylko nieliczne bazy danych rozwijające się w zamkniętych społecznościach mają szansę dobrze funkcjonować, co rzutuje na popularność użytkowania takich przeglądarek. Scentralizowane archiwa posiadają niekiedy niespójne albo wręcz żadne adnotacje¹⁰ genomów.

2.4.3 Istniejące rozwiązania

- *UCSC browser* autorstwa Jima Kenta, napisana w większości w języku C w 2000 roku. Udostępniana darmowo dla akademickich zastosowań. Dla komercyjnych zastosowań wymagana licencja *Kent Informatics*.
- *Gbrowse* - największa część kodu to Perl, z czasem coraz więcej JavaScriptu. Prace rozpoczęto w 2002 roku, funkcjonuje na licencji *PERL artistic license*.
- *ENSEMBL* - funkcjonuje na licencji Apache, dopuszczającej użycie kodu źródłowego zarówno na potrzeby wolnego oprogramowania, jak i zamkniętego oprogramowania komercyjnego. Znaczna część kodu napisana w Perlu, schemat bazy danych w MySQL.
- *Integrated Genome Browser* - projekt rozpoczęty przez firmę *Affymetrix*, jednak porzucony przez nią i rozwijany w środowisku akademickim. Przeglądarka napisana w Java, obecnie na prawach Academic free license.
- *Alamut* - przeglądarka z prostym interfejsem użytkownika z adnotacjami pobieranymi z publicznie dostępnych baz NCBI, EBI, UCSC. Zgodna z nomenklaturą HGVS (ang. *Human Genome Variation Society*). Funkcjonalność obliczeniowa oparta o narzędzia predykcyjne.
- *Argo Genome Browser* - narzędzie służące do wizualizacji i manualnego dodawania adnotacji genomów. Oprogramowanie open-source na licencji LGPL, napisane w języku Java.
- *ChIPMonk* - narzędzie opracowane w Instytucie Babraham Cambridge, służy do obrazowania i analizy tablic danych *ChIP-on-chip*. Kod wydany na licencji GPL v2. Projekt przestał być rozwijany.
- *Biodalliance* - szybka, interaktywna wizualizacja genomów w przeglądarce internetowej. Oparta o *JavaScript*, wspiera wiele najpopularniejszych formatów danych genetycznych. Rozwiązanie łatwe do wbudowania we własne aplikacje czy strony internetowe.
- *DNAexus* - przeglądarka do wizualizacji danych korzysta z technologii *Flash*. Jest narzędziem nowej generacji, mocno rozbudowanym w kategorii analizy i obrazowania sekwencji.

¹⁰ Adnotacja jest daną powiązaną z jakimś elementem (np. sekwencją), dostarczającą nowych informacji na temat tego elementu - np. położenie eksonów, zawartość par GC, mutacje.

- *GeneWall* - przeglądarka genomów przeznaczona na urządzenia mobilne. W wersji podstawowej można przeglądać jedynie genom człowieka, natomiast w wariantie profesjonalnym aplikacji mamy możliwość uploadu własnych plików z danymi.
- *Gaggle Genome Browser* - otwarte narzędzie do mapowania mocno skondensowanych danych na współrzędne genomu. Oprogramowanie przeznaczone do obsługi dużych zbiorów danych, pozwala na łatwy import plików użytkownika. Współpracuje z innymi bioinformacyjnymi narzędziami zawartymi w framework'u Gaggle.
- *Genestack* - internetowy, genomiczny system operacyjny. Pozwala wyszukiwać i importować dane z wielu publicznych baz danych. Potrafi konwertować dane wejściowe użytkowników z wielu formatów. Udostępnia zestaw narzędzi dla programistów, do łatwego tworzenia własnych przeglądarek.
- *GenomeView* - samodzielna przeglądarka i edytor genomu nowej generacji. Obecnie rozwijana przez społeczność Broad Institute. Zapewnia interaktywną wizualizację sekwencji, adnotacji, możliwość porównań na wielu poziomach, mapowań i wielu innych. Dzięki systemowi wtyczek, istnieje możliwość rozszerzenia funkcjonalności przeglądarki.
- *GenomeMaps* - wysokowydajna przeglądarka z interfejsem opartym o HTML5 i CSS3. W dużej mierze implementowana z użyciem biblioteki Javascript *JSorolla*. Dane pozyskuje korzystając z usług REST bazy *CellBase*. Uruchamia się we wszystkich nowszych przeglądarkach internetowych, nie wymagając od użytkownika instalacji dodatkowych komponentów.
- *Genome Wowser* - aplikacja przeznaczona na urządzenia mobilne iPad, wydana przez CBMI (ang. Center for Biomedical Informatics) w Szpitalu Dziecięcym w Filadelfii. Pozwala przeglądać popularną bazę *UCSC Genome Browser*.
- *Genomic HyperBrowser* - jest wolnym oprogramowaniem na licencji *GNU GPL v3*. Produkt skupiony głównie na analizie statystycznej elementów w genomie. Tworzony z wykorzystaniem platformy *Galaxy*.
- *Genoverse* - przenośna, konfigurowalna, przeglądarka oparta o Javascript i HTML5. Pozwala na eksplorację danych w dynamiczny, interaktywny sposób. Dane są prezentowane w przeglądarce, dzięki czemu może być łatwo instalowana na własnych stronach i pokazywać dane z wielu źródeł - gromadzonych online i lokalnie.
- *GenPlay* - szybkie i łatwe w użyciu narzędzie do analizy i przetwarzania sekwencji napisane w Javie. Uruchamia się na większości najpopularniejszych systemów operacyjnych.

Prace nad projektem rozpoczęto na Kolegium Medycznym Alberta Einstein'a Uniwersytetu Yeshiva w Nowym Yorku. Przeglądarka aktualnie w fazie testowania, używana przez studentów.

- *Integrated Microbial Genomes* - zaawansowany system wspierający analizę i adnotacje mikrobiologicznych zbiorów danych i metadanych genetycznych zgromadzonych w *DOE's Joint Genome Institute*. Współpracuje z wieloma instytucjami.
- *Microbial Genomic Viewer* - łatwe w obsłudze narzędzie do interaktywnej wizualizacji wyników analizy porównawczej genomów.
- *NextBio Genome Browser* - interaktywna aplikacja pozwalająca na wizualizację zależności pomiędzy prywatnymi lub publicznymi zbiorami danych biologicznych różnych typów.
- *Persephone* - rozbudowana aplikacja nowej generacji szeroko używana przez bioinformatyków i genetyków. Możliwość bezpłatnego korzystania z wersji *trial* przez 30 dni.
- *PlantGDB* - przeglądarka z zestawem narzędzi analitycznych wraz ze zbiorami danych genetycznych wielu roślin.
- *STAR* - zintegrowane środowisko do zarządzania i wizualizacji danych sekwencjonowania. Płynność działania aplikacji w przeglądarce internetowej zapewnia połączenie technologii JavaScript, HTML5 i asynchronicznej komunikacji do wymiany danych.
- *TGAC Browser* - nowa open-source'owa przeglądarka genomów obrazująca adnotacje z bazy danych *Ensembl*. Wyprodukowana przez *Centrum Analizy Genomów* w Wielkiej Brytanii (ang. *The Genome Analysis Centre, UK*).
- *Ugene* - darmowa platforma bioinformatyczna wspomagająca użytkowników w pracach nad sekwencjami. Oferuje narzędzia do analizy danych, przypisów, porównań itp. Dane wejściowe mogą być składowane lokalnie albo udostępnione z innych źródeł. Napisana w C++ z wykorzystaniem biblioteki Qt. Funkcjonuje na licencji GPL.
- *VISTA Enhancer Browser* - kompleksowy zestaw baz danych, narzędzi, serwerów na potrzeby analizy porównawczej sekwencji genomów. Istnieją 2 sposoby na korzystanie z przeglądarki: można wysyłać własne sekwencje i dopasowania do analizy, bądź sprawdzać z wstępnie przetworzonymi danymi całych genomów różnych gatunków. Projekt rozwijany we współpracy z wieloma instytucjami.

2.5 Porównanie

Duża część z dostępnych narzędzi oferujących gromadzenie danych sekwencji genomów to potężne platformy biologiczne oferujące funkcjonalność znacznie wykraczającą poza możliwości przechowywania i wyświetlania danych sekwencji. Dostępne przeglądarki potrafią korzystać z zewnętrznych, publicznie udostępnionych zbiorów danych w celu przeprowadzania m.in. zaawansowanych analiz porównawczych wykorzystując różnego typu metadane i annotations. Aplikacja stworzona w ramach przedmiotowej pracy inżynierskiej nie może konkurować z innymi przeglądarkami genomów z kilku powodów. Ważąca była niewątpliwie konieczność budowy oprogramowania samodzielnie i w ograniczonym czasie. Wiele dostępnych aplikacji rozwijano w ramach uczelni akademickich, angażując w postęp rzesze naukowców z różnych dziedzin. Mimo to, tworzenie tak rozbudowanych systemów zajmowało grupom co najmniej kilka lat.

Prezentowany w przedmiotowej pracy system ma za zadanie umożliwić wysłanie i przeglądanie danych zlokalizowanych na komputerze użytkownika, nie oferując przeprowadzania np. analiz statystycznych. Możliwe jest jedynie znakowanie wybranych odcinków sekwencji markerami.

Prezentowana aplikacja wykazuje pewne podobieństwa do dostępnych przeglądarek pod względem zastosowanej architektury. Pomimo, że nie jest to reguła, coraz więcej podmiotów odpowiedzialnych za oprogramowanie decyduje się na rozwiązań oparte o technologie uniezależniające końcowego użytkownika od posiadania konkretnej platformy sprzętowej. Technologie internetowe obsługiwane przez przeglądarki WWW wspierające HTML5 są rozwiązaniem zastosowanym również przez innych autorów.

Część dostępnych przeglądarek, tak jak w przypadku prezentowanej aplikacji, jest dworzona w postaci otwartego oprogramowania z udostępnionym kodem źródłowym. Niektóre aplikacje są płatne, rozwijane przez prywatne firmy, zazwyczaj oferujące możliwość skorzystania z bezpłatnego okresu próbnego w celu przetestowania programu.

3. Projekt i implementacja

Dane genetyczne przedstawiane są typowo w postaci zbiorów łańcuchów znaków, gdzie każdy ciąg jest sekwencją symboli z danego alfabetu. Reprezentacja łańcuchowa odzwierciedla fakt, że kwasy nukleinowe (DNA i RNA - cząsteczki przechowujące informacje genetyczne), są biopolimerami¹ nukleotydów². Kwasy nukleinowe odgrywają decydującą rolę jako nośnik informacji genetycznej nie tylko o wszystkich elementach budowy organizmu ale również determinują jego rozwój i funkcjonowanie. W sekwencji nukleotydów nazwanej kodem genetycznym jest zakodowana informacja o strukturze białek - zasadniczych składnikach struktur komórkowych oraz czynnikach przemian biochemicznych (enzymy, hormony). Białka zdobudowane są z aminokwasów połączonych wiązaniem peptydowym. Ustalono, że kod genetyczny jest:

- trójkowy, tzn. że jeden aminokwas jest zawsze kodowany przez trzy nukleotydy (kodon),
- nadmiarowy, tzn. że jeden aminokwas może być kodowany przez różne kodony
- uniwersalny, tzn. że konkretny kodon odpowiada temu samemu aminokwasowi prawie u wszystkich gatunków
- bezprzecinkowy, tzn. że pomiędzy nukleotydami w kodzie genetycznym nie ma nukleotydów, które pełniłyby funkcję pauzy
- niezachodzący, tzn. że nukleotydy konkretnego kodonu nie biorą udziału w kodowaniu dwóch aminokwasów

Aplikacje wykorzystywane do analizy danych genetycznych, korzystają zazwyczaj z ogromnych woluminów i złożonych algorytmów. Wysoka wydajność, elastyczność oraz interfejs użytkownika z wykorzystaniem przeglądarki internetowej to cechy aplikacji, które można osiągnąć łącząc ze sobą kilka języków programowania. Różnorodność systemów i użytkowników sprawia, że przenośność oprogramowania jest równie ważna. Badacze preferują korzystanie z aktualizowanych automatycznie, graficznych interfejsów użytkownika dostępnych z poziomu przeglądarek internetowych.

¹ Polimer występujący naturalnie w organizmach żywych, produkowany przez nie.

² Podstawowy składnik strukturalny kwasów nukleinowych.

3.1 Wymagania

„Wymaganiem jest każda właściwość oprogramowania potrzebna użytkownikowi do zaspokojenia jego potrzeb. Wymaganiem jest każda właściwość oprogramowania niezbędna do zatwierdzenia gotowego produktu. Wymaganiem jest każdy zapis dokumentujący właściwości określone w dwóch poprzednich zdaniach.” [8]

Zacytowane słowa prezentują najistotniejsze idee inżynierii wymagań. Określić je można na wiele sposobów. Mogą różnić się zakresem którego dotyczą oraz poziomem szczegółowości. Wymagania najczęściej odnoszą się do różnych właściwości systemu i zwyczajowo klasyfikujemy je na dwa sposoby:

- wymagania funkcjonalne
- wymagania niefunkcjonalne

3.1.1 Wymagania funkcjonalne

Cechy systemu określane przez wymagania funkcjonalne, specyfikują zestaw funkcji realizowanych przez projektowany system:

— Przeglądanie organizmów

Użytkownik rozpoczynając używanie aplikacji ma możliwość wyboru organizmu będącego obiektem dalszej analizy. Prezentacja jest w formie tabelarycznej.

— Przeglądanie chromosomów

Materiał genetyczny organizmu zawarty jest w zespole chromosomów. Użytkownik ma możliwość wyboru chromosomu danego organizmu. Widok jest w formie graficznej oraz tabelarycznej.

— Przeglądanie skafoldów

Jedną z cech koniecznych do precyzyjnego i szybkiego odszukiwania pożdanego fragmentu genomu jest możliwość nawigowania po genomie w kontekście skafoldów, będących częścią informacji genetycznej zawartej w chromosomie.

— **Interaktywna nawigacja po sekwencji chromosomu**

Użytkownik może przeglądać dane sekwencji zawartej w chromosomie w formie interaktywnego widoku pozwalającego na:

- przybliżanie, oddalanie
- przewijanie w kierunku horyzontalnym
- wprowadzenie granic wyświetlanego przedziału
- wyświetlenie sekwencji ustalonego widoku

Dodatkowo system wyświetla podgląd informujący o lokalizacji przeglądanego fragmentu genomu z perspektywy całego chromosomu. Użytkownik ma do dyspozycji podziałkę z jednostkami fizycznymi. System umożliwia naniesienie na widok wybranych markerów przyporządkowanych do przeglądanego chromosomu.

— **Dodawanie i usuwanie organizmów**

Możliwość dodania do aplikacji nowego organizmu ze źródeł danych pozyskanych z procesu sekwencjonowania genomu. Użytkownik ma możliwość usunięcia istniejącego organizmu z aplikacji.

3.1.2 Wymagania niefunkcjonalne

Wymogi niefunkcjonalne inaczej znane jako wymagania jakościowe określają ograniczenia, przy zachowaniu których system powinien realizować swoje funkcje. Wymagania projektowanego systemu:

— **Wydajność oraz skalowalność**

System wspiera obsługę wielu połączonych użytkowników jednocześnie. Ograniczenia czasowe uzależnione są głównie od posiadanego serwerowego sprzętu komputerowego. Maszyna kliencka nie musi być wysoce wydajna. System jest zdolny do zwiększenia wydajności poprzez dołączanie kolejnych zasobów do maszyny serwerowej.

— **Połączenie z siecią**

Projektowana aplikacja jest przystosowana do pracy w sieci. Użytkownicy z różnych maszyn wewnętrz sieci mają możliwość korzystania z oprogramowania.

— **Sprzęt komputerowy**

Do korzystania z systemu konieczny jest komputer bądź inne urządzenie posiadające

przeglądarkę internetową. Do uruchomienia części serwerowej aplikacji potrzebny komputer klasy PC.

— **Oprogramowanie**

Przeglądarka internetowa użytkowników końcowych wspiera HTML5. Maszyna serwerowa posiada system operacyjny z rodziny uniksopodobnych oparty na jądrze Linux.

— **Przenośność**

Użytkownik ma możliwość korzystania z aplikacji niezależnie od używanego systemu operacyjnego.

— **Aktualizacja oprogramowania**

Wykonywanie aktualizacji oprogramowania wyłącznie po stronie serwera. Aktualne moduły klienckie ładowane podczas uruchomienia programu.

3.2 Architektura

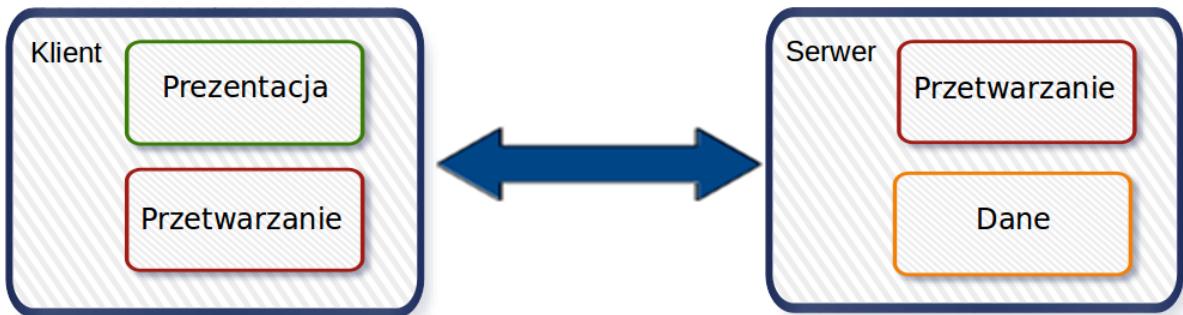
Aplikacja została napisana w oparciu o architekturę wielowarstwową skupiającą się na odseparowaniu od siebie części programu pełniących różne role w systemie. Takie podejście ułatwia pielęgnację kodu, dając możliwość modyfikacji poszczególnych warstw programu nie wpływając na inne.

Jedną z kluczowych cech wpływających na zdolność utrzymania dobrze zorganizowanego kodu jest jego czytelność. Programiści są zgodni, że model aplikacji trójwarstwowej ułatwia utrzymanie „czystego” kodu.

Popularnym rozwiązaniem w aplikacjach sieciowych jest koncepcja *klient-serwer*, dzieląca program na część prezentacyjną, obsługującą dane oraz część odpowiadającą za logikę biznesową aplikacji.

Umiejscowienie modułów obliczeniowych po stronie serwera zwalnia klienta z konieczności posiadania wydajnej maszyny, gdyż jego główne zadanie to prezentacja przetworzonych danych odebranych od serwera poprzez połączenie sieciowe. W dzisiejszych czasach praktycznie każda stacja robocza posiada jednostki obliczeniowe pozwalające na proste przetwarzanie. Sensownym podejściem wydaje się zrzucenie odpowiedzialności za operacje takie jak generowanie rysunków czy walidację danych na moduł klienta. Serwer musi być przystosowany do

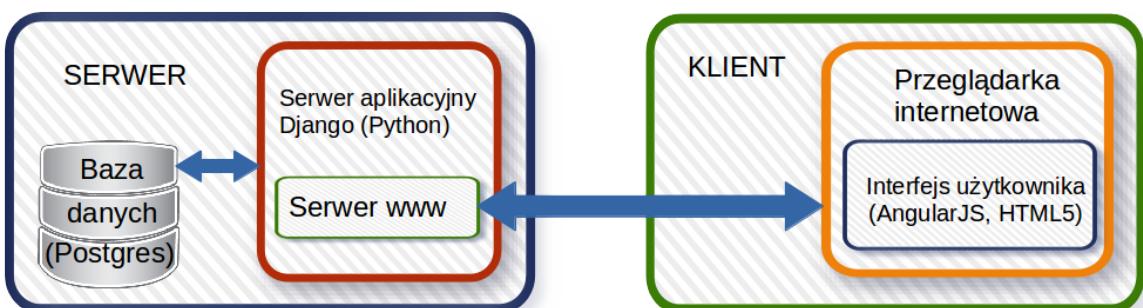
obsługi wielu klientów jednocześnie, zatem taka taktyka zmniejszy jego obciążenie do minimum. Rysunek 3.1 prezentuje rozmieszczenie poszczególnych modułów aplikacji.



Rysunek 3.1: Model aplikacji trójwarstwowej projektowanego systemu.

Źródło: <http://www.hindawi.com/journals/bmri/2014/253013/fig1/>

Oprogramowanie zostało zbudowane w oparciu o framework *bioweb*³ przystosowany do tworzenia aplikacji operujących na danych genetycznych. Łączy ze sobą wiele technologii, korzystając z zalet języków kompilowanych i interpretowanych. Rysunek 3.2 przedstawia ogólny schemat projektowanej aplikacji z uwzględnieniem zastosowanych narzędzi.



Rysunek 3.2: Schemat struktury projektowanej aplikacji.

Źródło: <http://bioweb.sourceforge.net/architecture.png>

3.3 Warstwa trwałości

Do prawidłowego i efektywnego funkcjonowania aplikacji potrzebna jest możliwość gromadzenia danych cyfrowych zgodnie z przyjętymi zasadami wynikającymi z analizy wymagań. Warstwa danych składa się stan aplikacji, umożliwiając jego odczyt i modyfikację. Operacje modyfikujące dane składające się z wielu elementarnych operacji powinny być przeprowadzane wszystkie jako całość albo wcale. Pożądaną cechą tej warstwy jest możliwość obsługi nowych żądań, bądź zgłaszania niepowodzenia odczytu lub zapisu, podczas obsługi aktualnych żądań.

³ <http://bioweb.sourceforge.net/pl/index.html>

Na tym poziomie najczęściej korzysta się z gotowych, zaimplementowanych już systemów bazodanowych lub systemów plików.

3.3.1 PostgreSQL

Wykorzystanym systemem bazodanowym w pracy jest *PostgreSQL* (zwany *Postgres*). Jest to potężny obiektowo-relacyjny system open source, z historią ponad 15 lat aktywnego rozwoju. Wypracował silną reputację niezawodności, spójności danych i poprawności. Działa na większości systemów operacyjnych takich jak *UNIX*, *Mac OS X* czy *Windows*.

3.3.2 ORM

Django dostarcza maper obiektowo relacyjny (ang. *Object Relational Mapping*) pozwalający operować na zgromadzonych danych, praktycznie bez potrzeby wykorzystywania języka *SQL*, jednak nie wykluczający takiej możliwości. Bogate API umożliwia przeprowadzanie nawet mocno skomplikowanych zapytań w stosunkowo prosty sposób. Często przy korzystaniu z ORM pojawiają się liczne problemy wydajnościowe jednak w omawianym projekcie nie sprawiły one problemu. Korzystanie z mapera okazało się wygodne i dość intuicyjne, jednocześnie będąc mało podatnym na popełnianie błędów przez programistę.

3.4 Warstwa przetwarzania

W niniejszym podrozdziale omawiam narzędzia i technologie programistyczne, użyte w pracy na poziomie warstwy przetwarzania. Warstwa ta odpowiada za logikę biznesową i wykonywane operacje w systemie. Pełni rolę kontrolera - komunikuje się z warstwą danych i prezentacji, koordynując pracę nad całością systemu. Jej głównym zadaniem jest przetwarzanie żądań przychodzących od klientów i zwrócenie im oczekiwanej odpowiedzi. Odpowiedni dobór narzędzi może w znacznym stopniu uprościć i przyspieszyć proces wytwarzania oprogramowania.

3.4.1 Python

Python to język programowania wysokiego poziomu, który łączy w sobie zalety tradycyjnych języków, takich jak *C/C++* i *Java*, z łatwością i szybkością tworzenia aplikacji w językach skryptowych, takich jak *Ruby* czy *VBScript*. Daje to jego użytkownikom możliwość tworzenia aplikacji, które mogą rozwiązywać różnorodne problemy.



Rysunek 3.3: Logo języka programowania *Python*

Źródło: <http://www.zerotech.com/image/page3/python.png>

Przyglądając się logo języka (Rys.3.3), można przypuszczać, że nazwa języka pochodzi od zwierzęcia - gatunku węża. Nic bardziej mylnego. Twórca *Pythona* - holenderski programista Guido van Rossum, był fanem serialu komediowego „Latający Cyrk Monty Pythona” (ang. „*Monty Python's Flying Circus*”), który emitowano w latach siedemdziesiątych przez głównego brytyjskiego publicznego nadawcę radiowo-telewizyjnego BBC.

Python realizuje kilka paradygmatów programowania, pozwalając programistom na stosowanie jednocześnie. Wspiera programowanie obiektowe, strukturalne oraz funkcyjne. W celu zarządzania dynamicznie przydzieloną pamięcią używa metod automatycznego zbierania nieużytków (ang. *garbage collection*).

Zaletą *Pythona* jest jego liczna społeczność. Posiada bardzo dużą liczbę użytkowników⁴, co ułatwia często rozwiązywanie napotykanych problemów. Istnieje bowiem spora szansa na to, że już ktoś kiedyś miał ten sam problem co my i pomocne informacje zmieścił w sieci.

Powodem użycia tego języka, oprócz wspomnianej popularności jest:

— **Jakość oprogramowania**

Dla wielu, jedną z najważniejszych cech *Pythona* jest jego spójność i wysoka ogólna jakość. Kod ma być czytelny i prosty w utrzymaniu, o wiele prostszy od tradycyjnych języków skryptowych. Uniformizm sprawia że jest łatwy do zrozumienia. *Python* ma głębokie wsparcie dla bardziej zaawansowanych mechanizmów ponownego użycia takich jak programowanie funkcyjne czy programowanie zorientowane obiektowo.

— **Efektywnosć pracy**

Stosowanie *Pythona* przyspiesza proces wytwarzania oprogramowania wiele razy w porównaniu do statycznie typowanych języków takich jak *C*, *C++* czy *Java*. Kod zajmuje zazwyczaj od jednej trzeciej do jednej piątej rozmiaru kodu napisanego w innym języku

⁴ Według autora książki *Learning Python 5th Edition*, w 2013 roku było mniej więcej milion użytkowników *Pythona*.[9]

kompilowanym. Oznacza to, że jest mniej do pisania, mniej do debugowania, a także mniej do utrzymywania. Programy tworzone w *Pythonie* uruchamiają się natychmiast, bez długich komplikacji i nie wymagają kroków takich jak np. linkowanie.

— Przenośność programów

Większość programów napisanych w *Pythonie* można uruchomić bez zmian na wszystkich najpopularniejszych platformach komputerowych. Przenoszenie kodu pomiędzy *Linuxem* a *Windowsem* polega zwykle na po prostu skopiowaniu plików programu pomiędzy maszynami. *Python* oferuje wiele opcji do implementacji przenośnych graficznych interfejsów, aplikacji webowych czy też programów łączących się z bazami danych. W dużej mierze wspierane są nawet operacje korzystające z interfejsów systemów operacyjnych, takie jak uruchamianie procesów czy działania na lokalnym systemie plików w ujednolicony dla użytkownika sposób.

— Biblioteki

Rozpoczynając użytkowanie *Pythona* dostajemy na starcie zbiór sporej biblioteki standardej oferującej przenośną funkcjonalność. Ta biblioteka wspiera szereg zadań programistycznych na poziomie aplikacji - od dopasowywania wzorców regularnych do eleganckich rozwiązań sieciowych. Firmy trzecie oferują mocno rozbudowane narzędzia do budowy stron internetowych, rozwoju gier i wiele innych. Dla przykładu - rozszerzenie *NumPy* zostało niejednokrotnie określone mocniejszym rozwiązaniem od *Matlaba* (w dodatku darmowym) w kontekście programowania numerycznego.

— Integracja z komponentami

Skrypty *Pythona* z łatwością mogą komunikować się z innymi częściami aplikacji za pomocą różnorodnych mechanizmów integracyjnych. Takie integracje pozwalają na wywoływanie wewnątrz kodu *Pythona*, instrukcji napisanych w *C*, bądź *C++*. Potrafimy korzystać z skryptów też analogicznie w drugą stronę (wykonanie kodu *Pythona* wewnątrz np. *C++*). Współpraca z fragmentami aplikacji napisanymi w *Javie* czy komponentami *.NET* również jest możliwa. *Python* potrafi współdziałać z urządzeniami poprzez COM (ang. *Component Object Model*), porty szeregowe czy interfejsy sieciowe takie jak *SOAP* czy *CORBA*.

— Frajda

Ze względu na łatwość obsługi i zestaw wbudowanych narzędzi, *Python* może wnieść do programowania więcej przyjemności niż przykrych obowiązków. Pomimo, że są to korzyści niematerialne, ich wpływ na produktywność jest dość ważnym atutem.

Bardzo pomocny w pisaniu pracy okazał się interaktywny interpreter *Pythona*. Pozwala on na m.in. testowanie linijek kodu bez konieczności tworzenia, edycji i uruchamiania źródłowego pliku. Powłoka języka nie tylko sprawdza poprawność napisanego kodu, ale także pozwala wykorzystać inne funkcje związane z pracą z kodem, jak np. sprawdzanie struktur danych. Doceniając zalety idei interaktywnego interpretera warto zaznajomić się z narzędziem *IPython*. Od tradycyjnego interpretera różni się zakresem możliwości - oferuje dodatkowo takie cechy jak:

- dostęp do powłoki systemowej
- numerowanie linii
- automatyczne wcięcia
- historię poleceń
- profilowanie kodu pod względem wydajności i zajętości pamięci.

3.4.2 Django

Każdy programista tworząc aplikację internetową zmaga się z takimi samymi trudnościami na pewnych etapach tworzenia oprogramowania. Ludzie projektują frameworki webowe, aby zautomatyzować czynności powtarzające się przy projektowaniu stron internetowych, dzięki czemu możemy się skupić na pisaniu aplikacji bez konieczności „wymyślania koła na nowo”. Jednym z nich jest właśnie *Django*.



Rysunek 3.4: Logo frameworku *Django*.

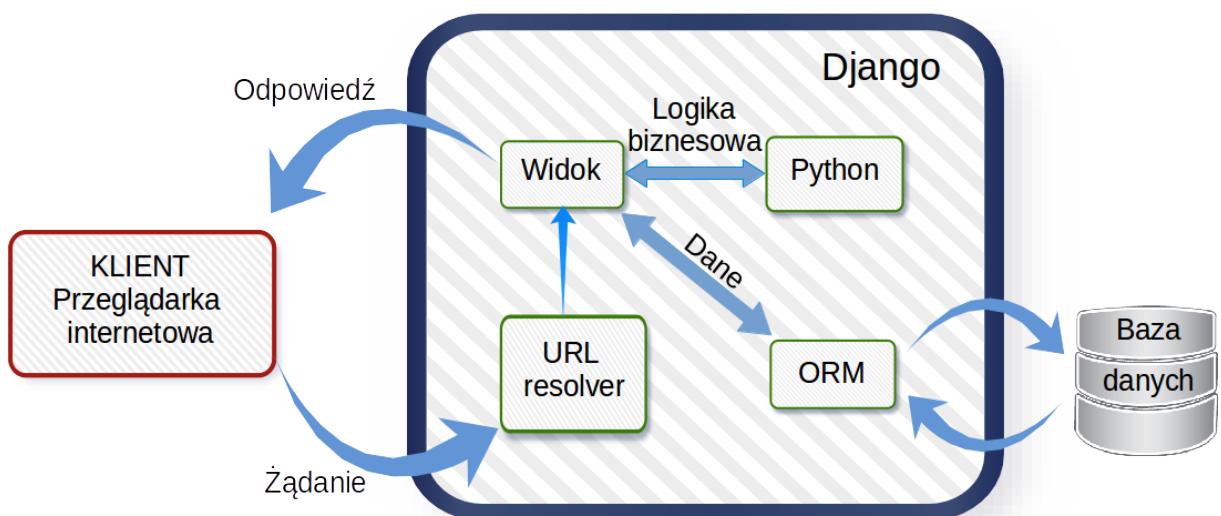
Źródło: <http://www.slothparadise.com/wp-content/uploads/2014/11/django.png>

Django jest w pełni darmowym, open-source'owym projektem rozpoczętym w 2003 roku, napisanym w całości w Pythonie. Nazwa frameworku pochodzi od imienia jednego z najwybitniejszych gitarzystów wszech czasów - francuza Django Reinhardta. Jest prosty, ale i elastyczny zarazem, dzięki czemu pozwala na projektowanie różnorodnych rozwiązań bez niepotrzebnych kosztów za pomocą względnie niedużej ilości kodu.

Sposób działania

W momencie gdy serwer otrzymuje żądanie, dostarcza je do *Django*, aby ten wywnioskował do czego się odnosi. Framework weryfikuje początkowo adres strony i uzgadnia co robić dalej. Modułem odpowiedzialnym za sprawdzenie adresu jest *urlresolver*. Jego działanie nie jest zbyt skomplikowane, bowiem sprowadza się do odnalezienia wzorca regularnego pasującego do adresu URL. Gdy odnajdzie pierwszy pasujący wzorzec przekazuje żądanie dalej, do stosowej funkcji nazywanej *widokiem*.

W funkcjach widoków implementowana jest logika biznesowa przetwarzająca żądanie i generująca odpowiedź (ang. *response*). Na tym etapie następuje najczęściej interakcja z bazą danych poprzez ORM, obejmująca operacje takie jak wyszukiwanie oraz modyfikacje rekordów danych zawartych w warstwie danych aplikacji. Wyprodukowaną odpowiedź *Django* wysyła do przeglądarki użytkownika. Uproszczony schemat działania ilustruje rysunek 3.5.



Rysunek 3.5: Schemat ilustrujący w uproszczeniu sposób funkcjonowania *Django*.

3.4.3 Gunicorn

Zalecanym przez twórców *Django* sposobem na wdrożenie aplikacji opartej o ich autorski framework, jest skorzystanie z WSGI (ang. *Web Server Gateway Interface*). Aplikacje internetowe napisane w języku *Python* projektowane były najczęściej pod kątem jednego z pośród wielu interfejsów do komunikacji z serwerem www. Często miały problem ze współpracą z ww. serwerami i vice versa. WSGI został stworzony jako interfejs niskiego poziomu w komunikacji między nimi w celu promowania wspólnej podstawy dla rozwoju przenośnych aplikacji internetowych napisanych z użyciem Pythona.



Rysunek 3.6: Logo serwera *Gunicorn*.

Źródło: http://gunicorn.org/images/large_gunicorn.png

*Gunicorn*⁵ jest serwerem HTTP implementującym interfejs WSGI dla systemu UNIX. Jest bliźniaczym oprogramowaniem przeniesionym z projektu *Unicorn* skierowanego dla języka *Ruby*. Serwer *Gunicorn* jest kompatybilny z szeroką gamą frameworków internetowych. Podczas działania używa niewielkiej ilości zasobów oraz jest dość szybki, co czyni go niewątpliwie atrakcyjnym rozwiązaniem.

3.4.4 Nginx

W środowisku produkcyjnym projektowana aplikacja korzysta z *Nginx*. *Nginx* jest serwerem WWW oraz serwerem proxy dla HTTP i IMAP/POP3. Twórcą jest Igor Sysojew. Dzięki swojej niezawodności, wysokiej wydajności i możliwości prostej, wysoce elastycznej konfiguracji, zyskał uznanie na całym świecie. Obecnie znajduje się na liście wśród 3 najczęściej używanych serwerów WWW, konkuruje z serwerem *Apache* czy produktami *Microsoftu*.



Rysunek 3.7: Logo serwera Nginx.

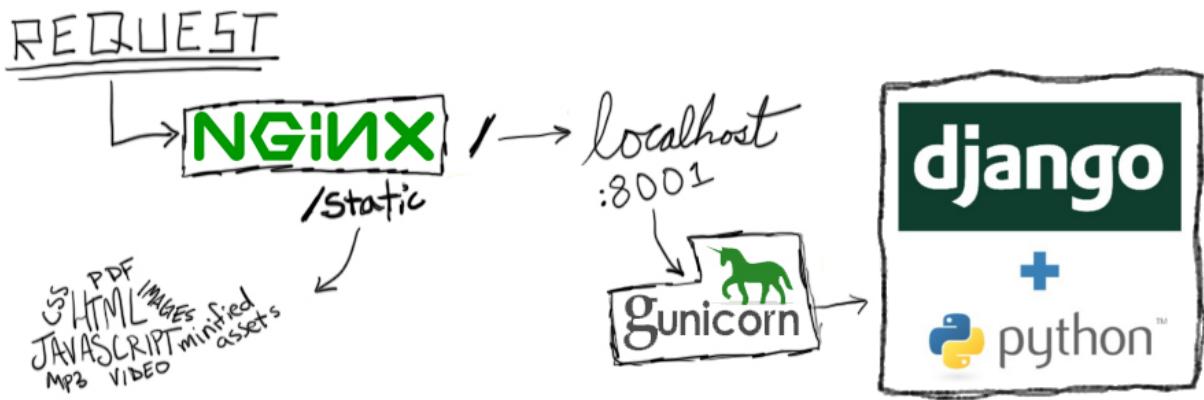
Źródło: <http://logonoid.com/images/nginx-logo.png>

Zaprojektowany został by stawić czoło problemowi *C10K*, polegającemu na obsłużeniu 10 tysięcy użytkowników jednocześnie. W przeciwieństwie do tradycyjnej architektury serwerów, *Nginx* do obsługi żądań nie używa osobnych wątków. Wykorzystuje architekturę sterowaną zdarzeniami. Dzięki asynchroniczności, nawet w trakcie dużego obciążenia korzysta z relatywnie małych przewidywanych ilości pamięci.

3.5 Warstwa prezentacji

Zadaniem warstwy prezentacji jest pobieranie żądań od użytkownika oraz wizualizacja wyników przetwarzania. Zasadniczą czynnością, którą wykonuje jest konwersja danych z postaci akceptowalnej przez system do formy czytelnej dla człowieka i odwrotnie. W przypadku

⁵ Green Unicorn



Rysunek 3.8: Schemat prezentujący przepływ żądania http w warstwie przetwarzania.

Źródło: <https://raw.githubusercontent.com/realpython/flask-deploy/master/images/localhost2.jpg>

zastosowania graficznego interfejsu użytkownika (ang. *graphical user interface - GUI*), warstwa ta monitoruje i interpretuje stan urządzeń wskazujących oraz dostarcza właściwych elementów graficznych.

3.5.1 HTML5

Jest to język, który został opracowany głównie w celu tworzenia i prezentowania internetowych stron www. Jest kolejną wersją języka HTML, w której kładziono nacisk na m.in. wprowadzenie nowych elementów zwiększających możliwości interaktywne i multimedialne prezentowanych treści.



Rysunek 3.9: Logo języka HTML5.

Źródło: https://www.w3.org/html/logo/downloads/HTML5_Logo_512.png

Canvas

Jedną z nowinek jakie wprowadzono w HTML5 jest element *Canvas*. Pozwala na dynamiczne rysowanie grafik 2D i 3D przy pomocy zazwyczaj Javascriptowych skryptów bez

konieczności instalacji dodatkowych wtyczek. W projektowanej przeze mnie przeglądarce stanowi istotny element, będący główną częścią widoku w graficznym interfejsie użytkownika.

3.5.2 JavaScript

Do wprowadzenia interaktywności na stronie korzystam z *JavaScriptu*. Umożliwia modyfikowanie zawartości kodu znaczników stron, za pomocą których strony te mogą być wyświetlane w przeglądarkach www. Cennym zastosowaniem jest dodanie przetwarzania po stronie klienta, nie wymagającego logiki serwera. Zdolność dynamicznego reagowania na zdarzenia użytkownika polepsza ogólne odczucia końcowego odbiorcy, czyniąc aplikację bardziej atrakcyjną.

3.5.3 AngularJS

Angular to framework JavasCriptowy wzbogaczający kod HTML przetwarzany przez przeglądarkę internetową. Dostarcza podstawy ułatwiające tworzenie rozbudowanych, bogatych aplikacji internetowych opartych o wzorzec projektowy MVC (ang. *Model-View-Controller*). Stosowanie go w projektach upraszcza organizację kodu, sprawia że oprogramowanie staje się łatwe w konserwacji i rozbudowie. Angular jest biblioteką typu open source, sponsorowaną i rozwijaną przez *Google*. Zastosowanie znajduje w największych i najbardziej skomplikowanych programach sieciowych ale również doskonale się spisał przy implementacji aplikacji będącej przedmiotem pracy inżynierskiej.



Rysunek 3.10: Logo frameworku AngularJS.

Źródło: <https://angularjs.org/img/AngularJS-large.png>

3.5.4 Bootstrap

Stworzenie profesjonalnej, nowoczesnej strony internetowej od zera wymaga dużo czasu i wysiłku od programisty. Dzisiejsze strony powinny działać szybko, być eleganckie i elastyczne. W dobie urządzeń mobilnych, coraz częściej wymaga się aby i na nich aplikacje poprawnie działały. Cytując autora książki wprowadzającej w świat Bootstrapa:

„Bootstrap ułatwia projektantom i programistom pracę, oferując szeroką gamę gotowych komponentów HTML oraz elastyczną strukturę w postaci systemu siatkowego, dzięki którym tworzenie profesjonalnych, responsywnych szablonów stron internetowych jest bardzo proste i trwa zdecydowanie krócej niż w przypadku tradycyjnych rozwiązań.” [25]

Bootstrap okazał się bardzo użyteczny przy implementacji elementów graficznych, nadając im elegancki wygląd i miejscami ożywiając je płynnymi animacjami. Próba tworzenia samodzielnie takich komponentów zajęłaby zdecydowanie zbyt wiele czasu.



Rysunek 3.11: Logo frameworku Bootstrap.

Źródło: http://shoplivechat.com/wp-content/uploads/2014/08/tech_bootstrap.png

3.6 Inne narzędzia

3.6.1 Zintegrowane środowisko programistyczne

Zastosowanie aplikacji IDE (ang. *Integrated Development Environment*) ułatwia i przyspiesza proces tworzenia kodu m.in. na etapach takich jak modyfikowanie, testowanie oraz konserwowanie oprogramowania. Programy będące zintegrowanymi środowiskami programistycznymi oferują złożoną, wieloaspektową funkcjonalność wspierającą programistę w wielu czynnościach. Omawiane aplikacje pomagają na poziomach:

- edycji i analizy kodu źródłowego
- komplikacji kodu źródłowego
- debugowania

- testowania
- refaktoryzacji kodu źródłowego
- tworzenia zasobów programu takich jak elementy graficzne, ikony, obrazy itd.
- tworzenia baz danych, komponentów.

Używanym przeze mnie środowiskiem jest produkt *PyCharm* firmy *JetBrains*. Posiada wszystkie cechy profesjonalnego narzędzia programistycznego. Jednym z powodów motywujących mnie do wyboru tego środowiska był fakt, że wspiera wiele frameworków, w tym użytych przeze mnie *Django* oraz *AngularJS*.



Rysunek 3.12: Logo firmy JetBrains i zintegrowanego środowiska programistycznego PyCharm.

Źródła: http://away3d.com/images/carousel/logo_jetbrains.png
<http://blog.pirx.ru/media/files/2015/type-hinting-talk/media/pycharm-logo.png>

Oferowana spora liczba wtyczek pozwala znacznie rozszerzyć możliwości aplikacji. Ciekawą funkcjonalnością okazała się możliwość instalacji wtyczki do *Google Chrome* pozwalającej na debugowanie kodu JavaScript w przeglądarce internetowej, sterując sesją bezpośrednio z oprogramowania *JetBrains*. W trakcie debugowania *PyCharm* wspiera również edytowanie kodu HTML i CSS „na żywo”, utrzymując aktualny obiektowy model dokumentu - DOM (ang. *Dokument Object Model*).

Używałem środowiska w wydaniu 5-tym *Professional Edition*, dzięki czemu mogłem korzystać z pełnej wersji programu. Możliwość bezpłatnego użytkowania uzyskałem na podstawie licencji studenckiej.

3.6.2 System kontroli wersji

System kontroli wersji⁶ to narzędzie pozwalające kontrolować zmiany przeprowadzane na plikach. W dowolnej chwili umożliwia odtworzenie którejkolwiek z wcześniejszych wersji

⁶ Inaczej system kontroli rewizji (ang. *revision control system*).



Rysunek 3.13: Logo systemu kontroli wersji *Git* i serwisu hostingowego *GitHub*.

Źródła: <https://git-scm.com/images/logo@2x.png>
<http://lumiinsight.com/wp-content/uploads/2015/11/GitHub.jpg>

projektu. Narzędzie pozwala na łatwy nadzór pracy nad projektem wykonywanym przez kilka osób. Podstawowe możliwości jakie oferuje użytkownikom to m.in.:

- przywrócenie plików do poprzedniej wersji
- odtworzenie stanu całego projektu
- porównywanie wprowadzonych zmian
- dowiedzenie się kto i kiedy modyfikował dany fragment projektu.

Zastosowanie narzędzia sprawia, że nawet jeśli zostanie popełniony błąd, bądź dane zostaną utracone, w stosunkowo łatwy sposób można przywrócić stan projektu do pożądanej formy.

Ze względu na dynamicznie rosnącą popularność w ostatnich latach, zdecydowałem się skorzystać z rozproszonego systemu kontroli rewizji *Git*. W celu ułatwienia koordynacji pracy, repozytorium zostało umieszczone w serwisie *GitHub*. Jest to jedno z najbardziej popularnych rozwiązań hostingowych projektów open source. Piki repozytorium projektu znajdują się pod adresem:

<https://github.com/mhaponiu/przegladarkaGenomow.git>

W trakcie trwania projektu wykonałem 105 commitów.

3.7 Szczegóły implementacyjne

Szacuję, że implementacja programu wymagała około 600 roboczo godzin. Łącznie napisałem ponad 4 tysiące linii kodu, przy czym wykonanie poszczególnych warstw przetwarzania i prezentacji zajęły odpowiednio: 2500 i 1600 linii kodu.

3.7.1 Dane

Struktura danych została tak zaprojektowana aby umożliwiała przechowywanie informacji o organizmach, chromosomach, skafoldach, sekwencjach, markerach i ich znaczeniach, możliwe dokładnie oddając ich naturę.

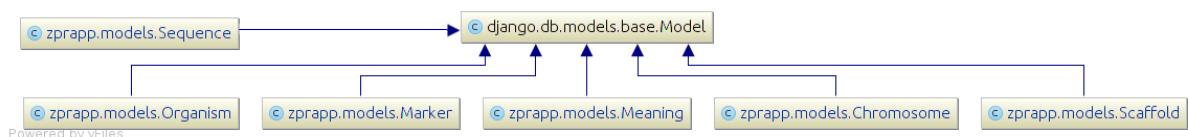
Baza została wygenerowana przez *Django* z wykorzystaniem *mapera obiekto relacyjnego* (ORM) na podstawie przygotowanych klas znajdujących się w pliku *models.py*. Diagram klas przedstawiający strukturę obiektową został pokazany na rys.3.14.

Każdy organizm posiada zbiór chromosomów, które mogą się składać z wielu skafoldów. Skafoldy mają określoną kolejność w chromosmie, znamy dodatkowo ich położenie i długość. Ciąg zasad składający się na sekwencje jest ciałem skafoldu. Markery mogą być nanoszone na wybrany odcinek chromosomu z przypisany znaczeniem. Istotę obrazuje diagram zależności modelu (rys.3.15). Utworzona struktura relacyjna zaprezentowana została na rys.3.16.

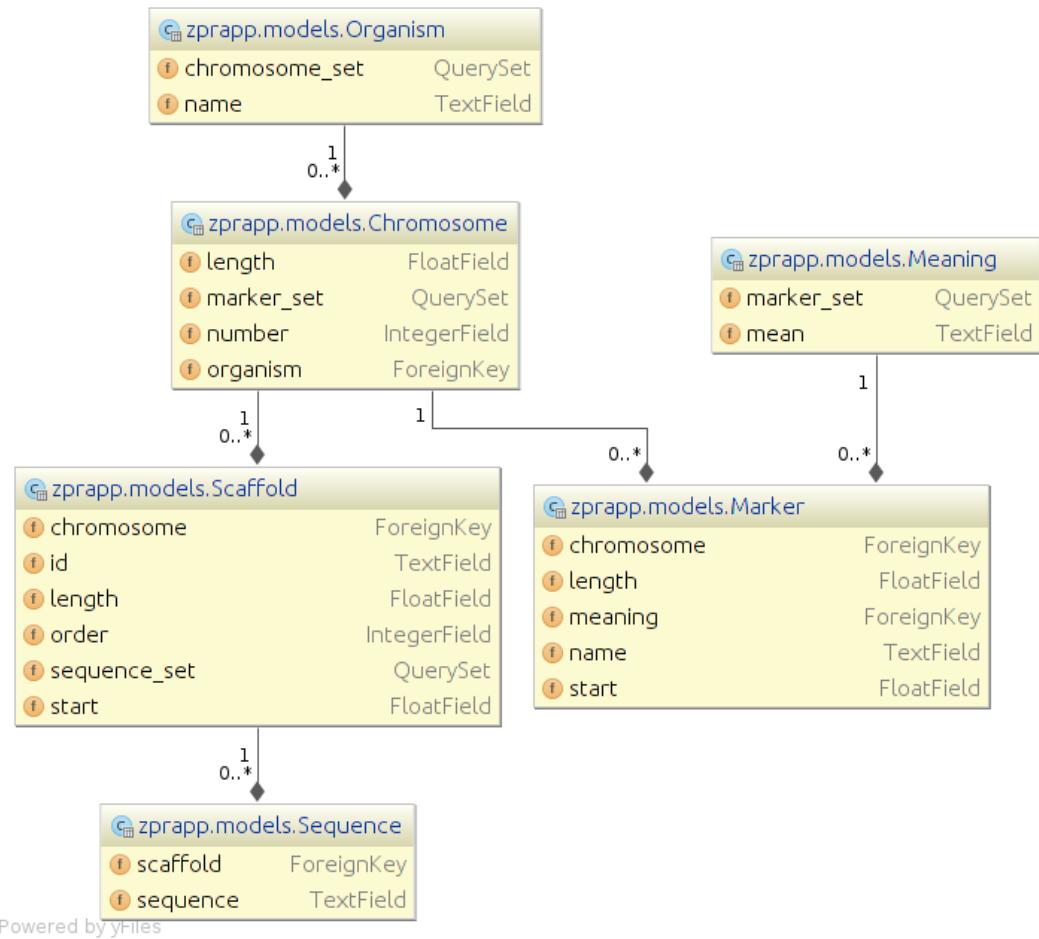
Genom ogórka

Baza danych została zainicjowana danymi o genomie ogórka z projektu *WebOmicsViewer* autorstwa *Piotra Róża*. Do realizacji zadania został mi udostępniony backup bazy danych ww. pracy. Napotkałem problemy w trakcie migracji danych, ponieważ okazały się one nie-spójne. Aby rozwiązać problem, zmuszony byłem dokonać analizy angażującej plik płaski xls, zawierający dodatkowe informacje o skorelowanych markerach z chromosomami. Moduł przeglądarki odpowiadający za inicjację bazy znajduje się w pakiecie *zpr.database*. Diagram klas został przedstawiony na rys.3.17. Klasa *DatabaseGenome* agreguje obiekty pozostałych klas odpowiedzialnych za prawidłowe wyselekcjonowanie i wstawienie danych do docelowej bazy.

W bazie przechowywanych jest 7 chromosomów ogórka, składających się łącznie z 964 skafoldów. Średnia długość przechowywanych skafoldów wynosi blisko 222 tysiące, zaś suma wszystkich sekwencji ponad 204 miliony par zasad. System bazodanowy gromadzi 72MB danych.



Rysunek 3.14: Diagram klas modelu danych.

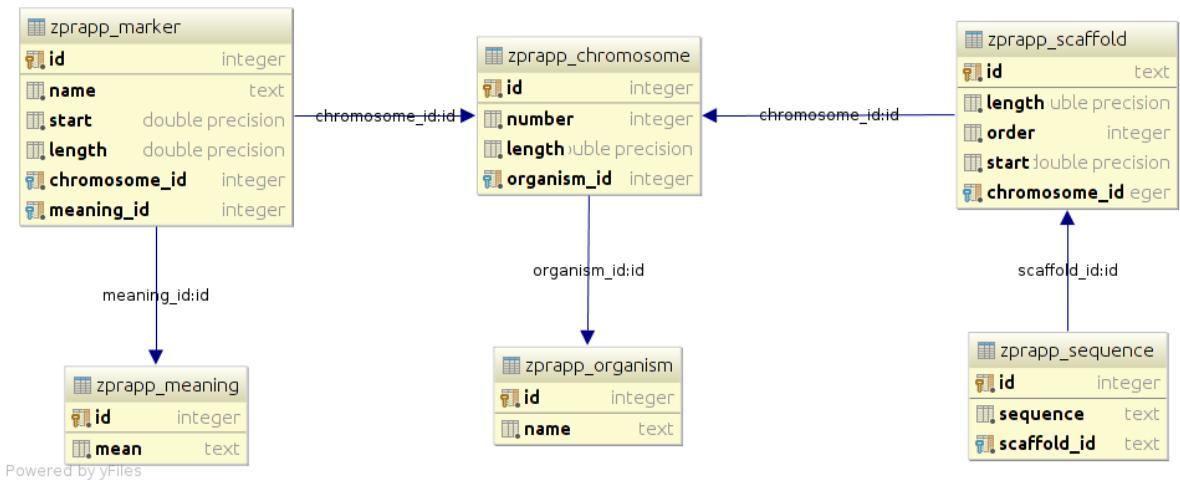


Rysunek 3.15: Diagram zależności modelu danych.

3.7.2 Budowanie i wdrażanie aplikacji

Wdrażanie aplikacji na nową maszynę jest procesem żmudnym i podatnym na błędy. Ręczne konfigurowanie środowiska zajmuje zazwyczaj wiele czasu. Niektóre czynności dają się z powodzeniem zautomatyzować. Użyteczne klasy do tego celu znajdują się w module `build_tools`. Klasa `AppBuilder` zajmuje się m.in.:

- utworzeniem drzewa katalogów projektu
- pobraniem plików potrzebnych do inicjacji bazy danych
- ustawieniem parametru `debug` (istotne przy wdrażaniu na środowisko produkcyjne)
- utworzeniem środowiska `virtualenv`, dzięki któremu mamy możliwość posiadania na tej samej maszynie kilku wersji tej samej biblioteki.



Rysunek 3.16: Diagram fizycznej struktury danych.

Klasa *Deployer* zajmuje się głównie przygotowaniem aplikacji i narzędzi do wdrożenia produkcyjnego. Generuje plik konfiguracyjny dla serwera WWW *Nginx* oraz ze względów bezpieczeństwa tworzy nowy tajny klucz zabezpieczeń *Django*. Obie klasy zostały przedstawione na rys.3.18. Aplikacja została wdrożona na serwerze uczelnianym. Dostępna jest pod adresem:

<http://smyrna.ise.pw.edu.pl:9002/>

3.7.3 Import i export danych

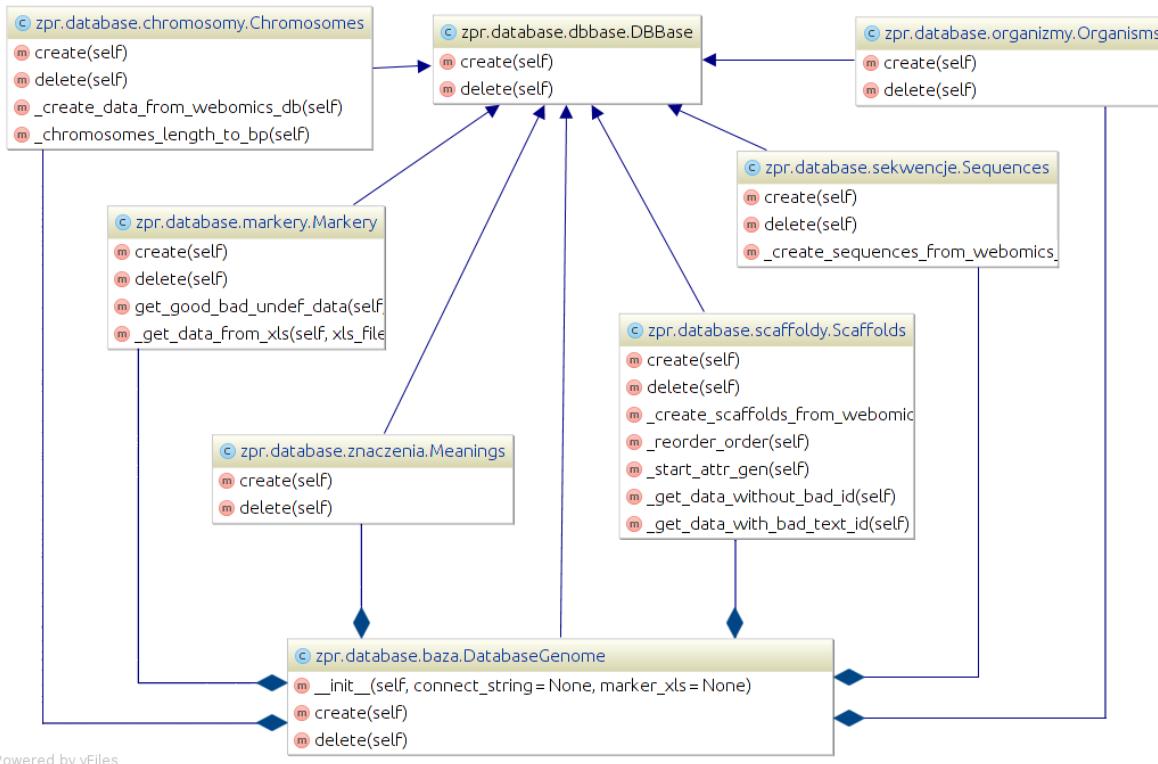
Jednym z wymagań funkcjonalnych była możliwość przesyłania na serwer danych dotyczących nowych organizmów. Dane sekwencji powinny być zgodne z formatem *FASTA*⁷. Plik fasta powinien zawierać linię opisu zaczynającą się od znaku >, a w niej identyfikator sekwencji. W kolejnych wierszach ma znajdować się sekwencja (ATCG), aż do pojawienia się kolejnej linii ze znakiem >, bądź osiągnięcia końca pliku.

Pozostałe pliki powinny być przesłane na serwer w formacie *GFF*⁸. Każdy rekord pliku GFF powinien rozpoczynać się *id obiektu*, a kończyć ewentualnym *id obiektu* nadziednego, którego dotyczy. W środku wymagane są kolejne atrybuty klasy.

Realizacja omawianego zadania wymaga po odebraniu plików, przeprowadzenia odpowiedniego procesu parsowania dokumentów po stronie serwera. Klasa potrafiąca generować pliki fasta i gff, oraz je odbierać i wkładać do bazy, znajduje się w module *zprapp.import_export*, którego diagram klas został zaprezentowany na rys.3.19. Jeśli okaże się,

⁷ https://en.wikipedia.org/wiki/FASTA_format

⁸ <https://genome.ucsc.edu/FAQ/FAQformat.html#format3>



Rysunek 3.17: Diagram klas modułu odpowiedzialnego za inicjację bazy danymi ogórką.

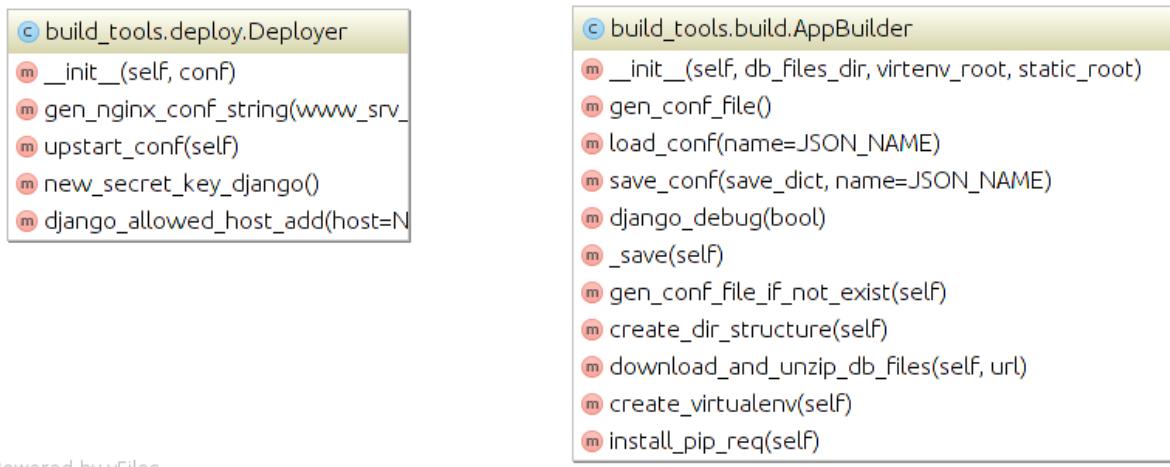
że pliki nie spełniają wymaganych zależności, zostanie wygenerowany wyjątek informujący klienta na jakim etapie przetwarzania wydarzył się błąd.

3.7.4 Trasowanie

Aby korzystanie z aplikacji było wygodne, warstwa prezentacji została stworzona w idei stron typu *one-page*. Podejście to cechuje wysoki poziom estetyki oraz duża interaktywność interfejsu. Niestety najczęściej przy takim rozwiązaniu pozbawiamy użytkownika możliwości zapisywania zakładek, ponieważ wszystkie operacje wykonują się dynamicznie bez zmiany adresu URL. Na szczęście *AngularJS* jest narzędziem umożliwiającym uporanie się z tym problemem. Listing 3.20 przedstawia funkcję konfigurującą trasowanie URL w przeglądarce. Do każdego widoku został przyporządkowany osobny kontroler ułatwiając pracę nad kodem.

3.8 Testowanie

Etap testowania oprogramowania to czas, w którym badamy całość programu lub jego poszczególne komponenty, uruchamiając kod w kontrolowanych warunkach i sprawdzając wyniki. Przeprowadzając testy sprawdzamy czy wyniki są zgodne z oczekiwaniemi użytkownika, gwarantując tym samym wymaganą jakość oprogramowania. Niejednokrotnie podczas



Rysunek 3.18: Diagram klas modułu *build_tools*.

implementacji, zastosowanie testów pomogło mi na wczesnym etapie wykryć defekty aplikacji, nie propagując błędów. Testy przeprowadzałem na dwóch poziomach:

- testowania jednostkowego
- testowania funkcjonalnego.

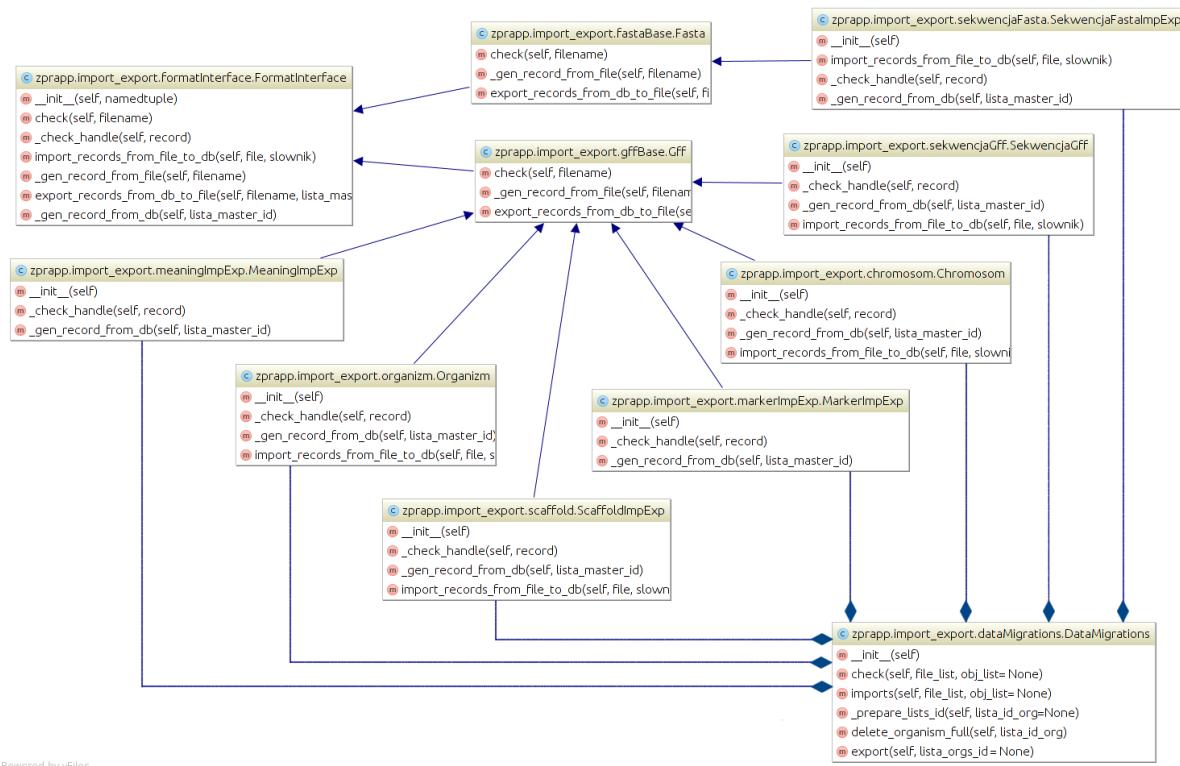
3.8.1 Testy jednostkowe

Przedmiotem testowania jednostkowego (ang. *unit testing*) są podstawowe jednostki programu. Postacie sprawdzanych komponentów różnią się w zależności od implementacji. Weryfikować możemy poprawność pojedynczych funkcji, metod czy klas zapisanych w języku obiektowym. Celem testowania na tym poziomie jest potwierdzenie poprawności wykonywanych fragmentów aplikacji jednocześnie próbując znaleźć jak największą liczbę błędów. Przeprowadziłem łącznie 26 testów jednostkowych.

Przykład testu widoku

Funkcja widoku *ajaxSeqSection(request)* będąca po stronie serwera ma za zadanie zwrócić fragment sekwencji chromosomu, który użytkownik zażąda korzystając z graficznego interfejsu użytkownika. Odnajdując potrzebne skafoldy musi je przyciąć w odpowiedni sposób, połączyć w całość w zadanym zakresie wypełniając luki między nimi i odesłać do klienta.

Aby proces przeprowadzać w stałych kontrolowanych warunkach, przed każdym uruchomieniem zestawu testów dla omawianego widoku tworzona jest testowa baza danych o strukturze identycznej jak w środowisku produkcyjnym lecz z innymi, niezmieniającymi się danymi. Funkcja realizująca wspomnianą czynność przedstawiona na listingu 3.21



Rysunek 3.19: Diagram klas modułu *import_export*.

Przeprowadzanie eksperymentów dla wszystkich możliwych danych wejściowych byłoby bardzo niewygodne. Projektując testy dla omawianego przykładu starałem się tak dobrać zbiór przypadków testowych aby efektywnie sprawdzić wszystkie aspekty poprawności funkcji widoku. Wyniki 13 przypadków testowych przedstawione na rys.3.22.

3.8.2 Testy funkcjonalne

Tego rodzaju testy sprawdzają z zewnątrz sposób działania całej aplikacji. Nazywane są często *testami czarnej skrzynki*, ponieważ osoba testująca nie korzysta z wiedzy dotyczącej wewnętrznych komponentów sprawdzanego systemu, lecz opiera się na założeniach funkcjonalnych jakie aplikacja powinna spełniać zgodnie z założeniami wynikającymi z wymagań.

Do przeprowadzenia testów funkcjonalnych skorzystałem z narzędzia *Selenium*, które pozwala na zaprzegnięcie do nich prawdziwej przeglądarki WWW. Łącznie wykonałem 12 testów funkcjonalnych. Przebieg przykładowego testu symuluje potencjalne działania użytkownika w trakcie korzystania z aplikacji.

Przykład

Scenariusz przeprowadzonego testu przedstawionego na listingu 3.23:

```

function cucRouteConfig($routeProvider) {
    $routeProvider.
        when('/organizmy', {
            controller: OrganizmKontroler,
            templateUrl: 'templates/organizmy.html'
        }).
        when('/organizm/:id_org/chromosomy', {
            controller: ChromosomKontroler,
            templateUrl: 'templates/chromosomy.html'
        }).
        when('/organizm/:id_org/chromosom/:id_chr/scaffold/:id_sc/sekwencja', {
            controller: SekwencjaKontroler,
            templateUrl: 'templates/sekwencja.html'
        }).
        otherwise({
            redirectTo: '/organizmy'
        });
}

```

Listing. 3.20: Funkcja konfigurująca trasowanie URL w przeglądarce WWW.

```

class GetSequenceSectionScaffCanvasTest(TestCase):
    @classmethod
    def setUpTestData(cls):
        #          0 1 2 3 4 5      0 1 2 3 4 5 6 7
        # chromosom  # # A G T C A # # # G T C A G T C # # #
        # index     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
        o = Organism(name="organizm_testowy1", id=1)
        chr = Chromosome(organism=o, number=1, length=21, id=1)
        cls.chr_id = chr.id
        sc1 = Scaffold(chromosome=chr, length=5, order=0, start=3, id=1)
        seq1 = Sequence(scaffold=sc1, sequence="AGTCA", id=1)
        sc2 = Scaffold(chromosome=chr, length=7, order=1, start=11, id=2)
        seq2 = Sequence(scaffold=sc2, sequence="GTCAGTC", id=2)
        for obj in chain([o, chr, sc1, sc2, seq1, seq2]):
            obj.save()

```

Listing. 3.21: Funkcja tworząca testową bazę danych dla zestawu testów *GetSequenceSectionScaffCanvasTest*

1. Użytkownik wchodzi na stronę główną aplikacji.
2. Zauważa ładnie wyświetlony pasek z tytułem aplikacji.
3. Wybiera pierwszy organizm.
4. Decyduje się na wybór drugiego chromosomu.
5. Ustawia parametry widoku aby pokazywały część chromosomu od indeksu 10 do 20.
6. Wybiera pożądany skafold.
7. Kliką w przycisk służący do kopiowania w celu zapisania sekwencji skafoldu do schowka systemowego.
8. Wyłącza aplikację.

| | | |
|----|--|------|
| OK | zprapp.tests.GetSequenceSectionScaffCanvasTest | 98ms |
| OK | test_url_resolver_to_get_sequence_section | 3ms |
| OK | test_view_ajaxSeqSection_1 | 12ms |
| OK | test_view_ajaxSeqSection_2 | 7ms |
| OK | test_view_ajaxSeqSection_3 | 7ms |
| OK | test_view_ajaxSeqSection_4 | 6ms |
| OK | test_view_ajaxSeqSection_5 | 8ms |
| OK | test_view_ajaxSeqSection_6 | 6ms |
| OK | test_view_ajaxSeqSection_7 | 6ms |
| OK | test_view_ajaxSeqSection_8 | 7ms |
| OK | test_view_ajaxSeqSection_9 | 9ms |
| OK | test_view_ajaxSeqSection_10 | 9ms |
| OK | test_view_ajaxSeqSection_11 | 9ms |
| OK | test_view_ajaxSeqSection_12 | 9ms |

Rysunek 3.22: Wynik testów jednostkowych widoku *ajaxSeqSection* prezentowany w IDE.

Rezultaty testu są pozytywne, wykonanie zajęło ponad 7 sekund (rys.3.24).

```
def test_copy_seq_chosen_scaff(self):
    self.browser.get(self.live_server_url)
    self.browser.set_window_size(1200, 600)
    bar = self.browser.find_element_by_class_name('navbar-brand')
    self.assertEqual(bar.text, u'PRZEGŁĄDARKA GENOMÓW')
    self.browser.find_element_by_link_text('Organizm_testowy_1').click()
    self.browser.find_element_by_link_text('chromosom_2').click()
    inputbox_from = self.browser.find_element_by_id("view_from")
    inputbox_from.clear()
    inputbox_from.send_keys("10")
    inputbox_to = self.browser.find_element_by_id("view_to")
    inputbox_to.clear()
    inputbox_to.send_keys("20")
    self.browser.find_element_by_id('button_insert_data').click()
    self.browser.find_element_by_link_text('scaffold_2').click()
    self.browser.find_element_by_id("button_copy").click()
    root = tk.Tk()
    root.withdraw()
    self.assertEqual(root.clipboard_get(), "GTCAGTC")
```

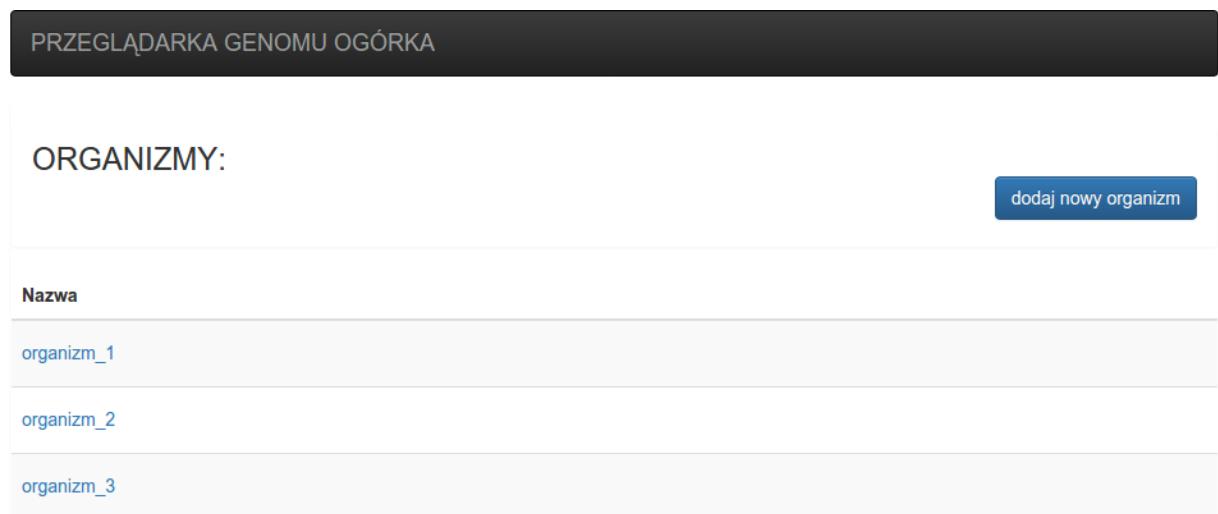
Listing. 3.23: Przykładowy test funkcjonalny aplikacji.

OK test_copy_seq_chosen_7s 703ms

Rysunek 3.24: Wynik przykładowego testu funkcjonalnego w środowisku PyCharm.

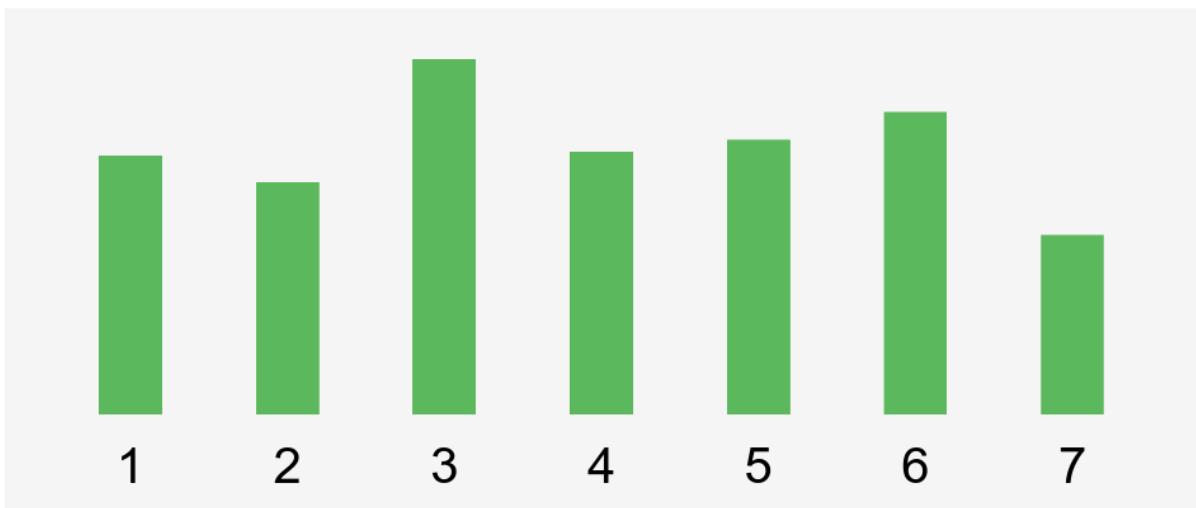
4. Demonstracja programu

Na stronie startowej aplikacji, wyświetlane są dostępne organizmy w bazie danych oraz przycisk umożliwiający uruchomienie okna dialogowego pozwalającego na dodanie nowego organizmu (rys.4.1). Wybierając organizm, przechodzimy do widoku chromosomów (rys.4.2). Przejście do następnego widoku (rys.4.3) następuje po kliknięciu w wybrany słupek reprezentujący chromosom. W tej części aplikacji nawigujemy po chromosomie za pomocą przycisków przybliżania i poruszania się na boki w celu odnalezienia pożądanego fragmentu genomu. Potrafimy nanosić na główny pasek poszczególne markery. Aby odczytać fragment wyświetlonej sekwencji należy nacisnąć przycisk z okiem, bądź wybrać z tabeli skafold, którego zawartość będziemy chcieć pobrać.



Rysunek 4.1: Widok strony startowej.

PRZEGŁĄDARKA GENOMU OGÓRKA

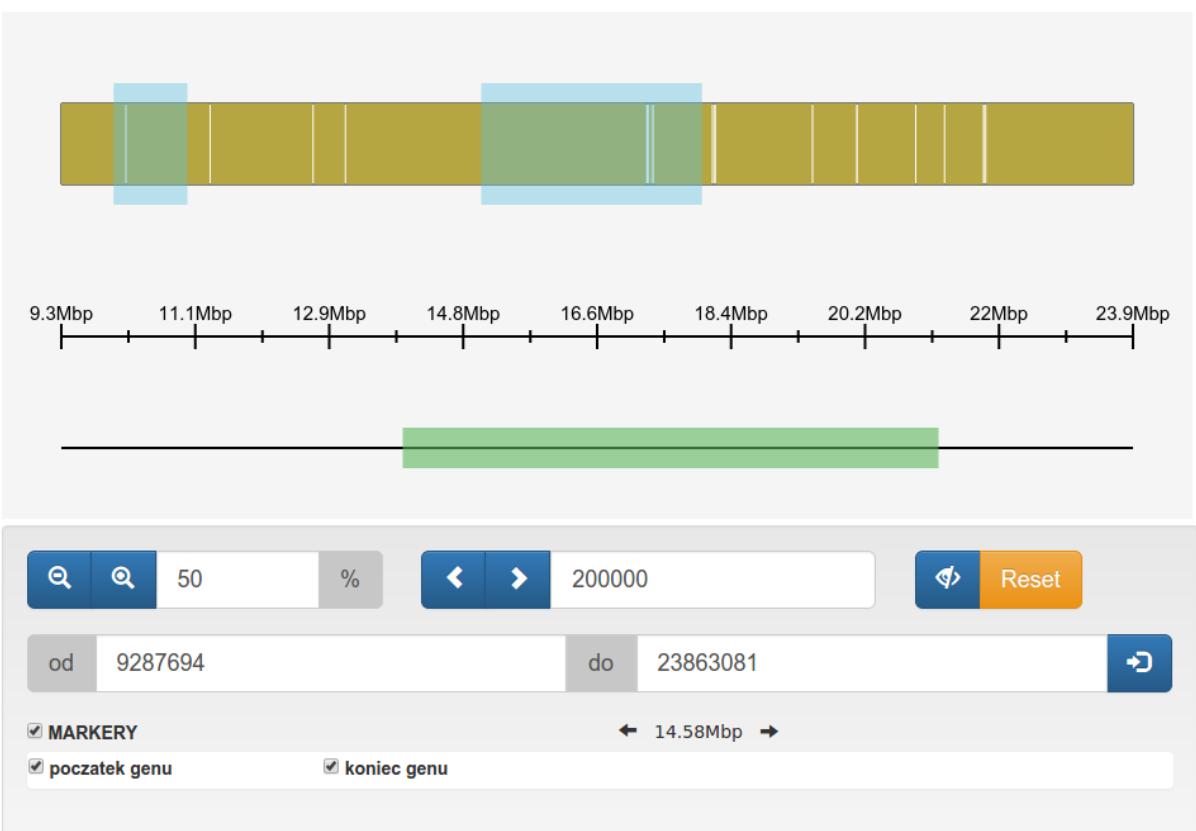


CHROMOSOMY:

| Chromosome | Długość |
|-------------|----------|
| chromosom 1 | 29150775 |
| chromosom 2 | 26165221 |
| chromosom 3 | 40056285 |
| chromosom 4 | 29601718 |
| chromosom 5 | 30950768 |
| chromosom 6 | 34089568 |
| chromosom 7 | 20250815 |

Rysunek 4.2: Widok chromosomów organizmu.

PRZEGŁĄDARKA GENOMU OGÓRKA



SCAFFOLDY (id_chr 1) (łącznie 105; wyświetcone 35):

| Scaffold | Długość | Start | Order |
|-----------------|---------|----------|-------|
| scaffold 360295 | 1219766 | 8937946 | 3 |
| scaffold 3656 | 2135 | 10166213 | 4 |
| scaffold 360300 | 1128022 | 10176849 | 5 |

Rysunek 4.3: Widok skafoldów chromosomu.

5. Podsumowanie

Wartość przedmiotowej pracy inżynierskiej w opinii autora można przedstawić w dwóch zasadniczych wymiarach.

Po pierwsze, zaproponowana przeglądarka genomów, może być wykorzystywana jako użyteczny element pomocniczy w procesach badawczych, które w istotny sposób przyczyniają się do postępu cywilizacyjnego. Osiągnięcia genomiki dają np. możliwość pożądanych modyfikacji roślin, usprawniają kryminalistykę i sądownictwo (analiza śladów biologicznych, identyfikacja zmarłych, ustalanie ojcostwa), pozwalają na określanie stopnia pokrewieństwa pomiędzy gatunkami oraz wpływają na rozwój precyzyjnej diagnostyki medycznej opartej na metodach genetyki molekularnej.

Inna, osobista wartość wynika z faktu poszerzenia umiejętności autora w zakresie tworzenia aplikacji sieciowych. W trakcie pisania pracy napotkałem wiele problemów, których rozwiązywanie nauczyło mnie pokory i pozwoliło poznać wiele ciekawych technologii. Zapoznanie się z całym wachlarzem nowoczesnych narzędzi było istotnym etapem na drodze do zawodu programisty i stanowiło ważne uzupełnienie wiedzy zdobytej podczas studiów na Wydziale Elektroniki i Technik Informacyjnych. Nabycie umiejętności będę wykorzystywał w przyszłych projektach, doskonalać się w sztuce inżynierijnej.

Otwartą możliwością i kwestią ewentualnej przyszłości pozostaje rozwinięcie funkcjonalności zaproponowanej przeglądarki o możliwość przeprowadzania analiz statystycznych i porównawczych z wykorzystaniem publicznie dostępnych baz danych. Architektura przeglądarki została tak zaprojektowana by możliwe było w przyszłości implementowanie wysokowydajnych algorytmów z użyciem języków kompilowanych takich jak *C++*. Dzięki bibliotece *Boost*, część serwerowa napisana w Pythonie z powodzeniem może być łączona z kodem *C++*, oferując ogromne możliwości rozbudowy aplikacji, która w mojej opinii posiada potencjał aby w przyszłości, po rozszerzeniu funkcjonalności, konkurować z innymi projektami.

Jednym z najpopularniejszych warzyw w Polsce i na świecie jest ogórek. Duży popyt powoduje, że ogórek był i nadal pozostaje ważnym obiektem upraw a więc i badań. Ogórek należy

do rodziny dyniowatych, jest spokrewniony z melonem, arbuzem i dynią. W celu zsekwencjonowania jego genomu, w 2008r. zostało powołane Polskie Konsorcjum Sekwencjonowania Genomu Jądrowego Ogórka (rys.5.1). Konsorcjum jest wspólnym przedsięwzięciem naukowców i przedsiębiorców z różnych instytucji chętnych do współpracy w procesie eksploracji genomu ogórka. Głównym zadaniem wskazanego konsorcjum jest postęp nad składaniem i wykorzystaniem sekwencji genomu do badań i aplikacji. Istnieje szansa, że przeglądarka stworzona w ramach pracy dyplomowej będzie w przyszłości intensywnie używana przez ww. związek przedsiębiorstw. Możliwe, że aplikacja po rozszerzeniu funkcjonalności i optymalizacji interfejsu użytkownika, jako narzędzie, przyczyni się do utrzymania hodowli ogórka w Polsce na poziomie konkurencyjnym do światowego.[27, 28]



Rysunek 5.1: Logo Polskiego Konsorcjum Sekwencjonowania Genomu Jądrowego Ogórka (ang. *Polish Consortium of Cucumber Genome Sequencing*).

Bibliografia

- [1] Sanger F, Air GM, Barrell BG, Brown NL, Coulson AR, Fiddes CA, Hutchison CA, Slocombe PM, Smith M., *Nucleotide sequence of bacteriophage phi X174 DNA*, Nature. 1977 Feb 24;265(5596):687-95.
- [2] Robert M. Nowak *Polyglot programming the applications to analyze genetic data* BioMed Research International, vol. 2014, doi:10.1155/2014/253013, <http://www.hindawi.com/journals/bmri/2014/253013> 10 października 2015
- [3] Bartek Wilczyński *Architektura dużych projektów bioinformatycznych*
<http://bioputer.mimuw.edu.pl/~bartek/APB/wyk3-browsers.pdf>,
17 sierpnia 2015
- [4] Krzysztof Sacha *Inżynieria oprogramowania* PWN, Warszawa 2010
- [5] Jan Paweł Jastrzębski *Wykład 2 bioinformatyka 2007/2008*
http://ebiolog.pl/graf/Wyklady/biotech/W2/Bioinformatyka_W2.pdf,
16 sierpnia 2015
- [6] Anna Leśniewska *Przeglądarki genomowe (2)*
http://www.cs.put.poznan.pl/alesniewska/BABD/W_2015-05-06.pdf,
17 sierpnia 2015
- [7] Uniwersytet Warmińsko Mazurski w Olsztynie *Historia Bioinformatyki...*
<http://www.uwm.edu.pl/wnz/KBZ/Lipazy/bioinfo-historia.html>
- [8] IEEE/ANSI Std 610, Glossary of Software Engineering Terminology, IEEE Computer Society Press 1990.
- [9] Mark Lutz *Learning Python, 5th Edition*
O'Reilly Media, June 2013. ISBN: 978-1-449-35573-9.
- [10] *Django Girls*
<http://tutorial.djangogirls.org/pl/django/index.html> 15 stycznia 2015
- [11] *Django*
<https://www.djangoproject.com/> 15 stycznia 2015
- [12] *Gunicorn*
<http://gunicorn.org/> 14 stycznia 2015
- [13] *Nginx*
<https://www.nginx.com/resources/wiki/> 14 stycznia 2015

- [14] JetBrains *PyCharm*
<https://www.jetbrains.com/pycharm/>, 14 stycznia 2015
- [15] Jeff Forcier, Paul Bissex, Wesley Chun *Python i Django. Programowanie aplikacji webowych.*
Helion, 2009. ISBN 978-83-246-2225-2.
- [16] Włodzimierz Gajda *Git. Rozproszony system kontroli wersji*
Helion, 2013. ISBN 978-83-246-7305-6.
- [17] e-Biotechnologia *Co można wyciągnąć ze zwykłej sekwencji?, Mapowanie genomów, Para zasad, Budowa białek*
<http://www.e-biotechnologia.pl/>
- [18] Wikipedia(PL) *Bioinformatyka, Centymorgan, Kwas rybonukleinowy, SOAP, REST, Apache License, Biopolimery, Nukleotydy, Białka, Struktura drugorzędowa białka, PyCharm, Django, Python, Chromosom, JavaScript, HTML5, Testowanie oprogramowania, Genom*
<http://pl.wikipedia.org/>
- [19] Wikipedia(EN) *Genome browser, WSGI, Contig*
<http://en.wikipedia.org/>
- [20] PWN *Chromosomy, Marker genetyczny*
<http://encyklopedia.pwn.pl/haslo/chromosomy;3886021.html>
- [21] Medicinenet *Definition of Scaffold*
<http://www.medicinenet.com/script/main/art.asp?articlekey=25223>
21 stycznia 2015
- [22] IT PWN *Po co dzielić aplikacje na warstwy?, Dlaczego warto stosować różne języki programowania jednocześnie?*
<http://it.pwn.pl/Artykuly/Programowanie/> 20 stycznia 2016
- [23] Jon Duckett *JavaScript and JQuery: Interactive Front-End Web Development*
John Wiley & Sons, Inc., Indianapolis, Indiana, 2014. ISBN 978-83-283-0126-9.
- [24] Adam Freeman *AngularJS Profesjonalne techniki*
Helion, 2014. ISBN 978-83-283-0200-6.
- [25] Syed Fazle Rahman *Bootstrap. Tworzenie interfejsów stron WWW. Technologia na start!*
Helion, 2015. ISBN 978-83-283-0514-4.
- [26] Harry J.W. Percival *TDD w praktyce. Niezawodny kod w języku Python*
- [27] Zbigniew Przybecki, Rafał Wójcicki, Stefan Malepszy *Sekrety ogórka nareszcie ujawnione - genom ogórka zsekwencjonowany*, tom 36, 2009 suplement nr 25 (19-31)
- [28] PCC Genomics
<http://csgenome.sggw.pl/en-us/> 27 stycznia 2015