

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Informatyki

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Inżynieria Systemów Informatycznych

Przeglądarka danych uzyskanych z sekwencjonowania następnej
generacji (NGS)

Tomasz Kogowski

Numer albumu 261428

promotor
dr inż. Tomasz Gambin

Warszawa 2017

Streszczenie

Wraz z rozwojem technologii umożliwiających tanie i efektywne pozyskiwanie danych z sekwencjonowania DNA wzrosło zapotrzebowanie na stworzenie systemu umożliwiającego analizę wariantów genetycznych wykrytych u pacjentów oraz ich korelację z chorobą.

Praca prezentuje interfejs do rozproszonej bazy danych, zawierającej dane pozyskane z sekwencjonowania DNA. Aplikacja przeglądarkowa została opracowana w języku programowania Scala wraz z wykorzystaniem platformy programistycznej Play oraz platformy AngularJS. W pracy opisany został schemat bazy danych PostgreSQL, która przechowuje filtry oraz dane o użytkownikach. Poza przypadkami użycia przedstawione zostały także elementy, które umożliwiły redukcję czasu odpowiedzi serwera aplikacyjnego na żądanie HTTP oraz sposób zabezpieczenia haseł użytkowników razem ze sposobem zarządzania sesją. Cały system został zaprojektowany jako łatwy do rozbudowy.

Słowa kluczowe:

- DNA,
- Scala,
- medycyna,
- Play Framework,
- PostgreSQL,
- AngularJS,

Abstract

With the development of cheap and effective DNA sequencing methods a system for comprehensive analysis of genomic variant data became vital for many clinics and research facilities.

The result of this thesis is a user interface to data warehouse which stores genomic variant information. The web application was developed in Scala programming language with Play and AngularJS frameworks. This essay also explains all of the use cases, application security aspect and the optimization steps that have reduced server response time for HTTP requests. PostgreSQL database schema that was created to store user data and credentials is included here as well.

Keywords:

- DNA,
- Scala,
- medicine,
- Play Framework,
- PostgreSQL,
- AngularJS,



„załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta”

Spis treści

1	Wstęp	7
1.1	Motywacja	7
1.2	Cel pracy	7
1.3	Zawartość rozdziałów	8
2	Specyfika danych	9
3	Wymagania funkcjonalne i нефункционалне	11
3.1	Wymagania funkcjonalne	11
3.2	Wymagania нефункционалне	12
4	Istniejące rozwiązania	13
5	Wybór technologii	16
5.1	Język programowania Scala	16
5.2	System zarządzania bazą danych	16
5.2.1	MySQL	17
5.2.2	SQLite	17
5.2.3	PostgreSQL	17
5.2.4	Uzasadnienie wyboru PostgreSQL	18
5.3	Slick	18
5.4	JDBC	18
5.5	Aplikacja przeglądarkowa	19
6	Przypadki użycia	20
6.1	Autoryzacja	20
6.1.1	Role	20
6.1.2	Rejestracja i logowanie użytkownika	20
6.2	Przeglądanie danych z sekwencjonowania DNA	21
6.3	Panel administratora	30
7	Schemat bazy danych	33
8	Wydajność aplikacji	37
9	Bezpieczeństwo aplikacji	42

10 Częściowy opis implementacji	43
10.1 Struktura folderów	43
10.1.1 Pliki konfiguracyjne	43
10.1.2 Pliki napisane w języku Scala	44
10.1.3 Pliki części klienckiej	45
10.2 Opis implementacji autoryzacji i sesji	45
11 Wnioski i podsumowania	48
Literatura	48
Wykaz rysunków i tabel	53

1 Wstęp

1.1 Motywacja

W dzisiejszych czasach genetyka jest jednym z najważniejszych działów medycyny. Dzięki rozwojowi metod sekwencjonowania stała się możliwa skuteczna diagnostyka wielu chorób o podłożu genetycznym.

Równocześnie z rozwojem technologii sekwencjonowania DNA, wzrasta zapotrzebowanie na systemy pozwalające na przechowywanie oraz analizowanie zebranych informacji. Na świecie istnieje wiele publicznie dostępnych baz danych, z wariantami dziesiątek tysięcy pacjentów. Powstało również wiele systemów starających się umożliwić wygodne i efektywne przeglądanie oraz analizowanie list wariantów i adnotacji. Jednak do tej pory nie opracowano darmowego i uniwersalnego narzędzia pozwalającego na efektywne i szybkie znajdowanie wariantów odpowiedzialnych za choroby czy też, które spełniałoby wszystkie oczekiwania potencjalnych użytkowników takiego systemu.

Dlatego na Wydziale Elektroniki i Technik Informacyjnych oraz w Zakładzie Genetyki Medycznej Instytutu Matki i Dziecka w Warszawie powstał projekt mający na celu połączyć najnowsze technologie i praktyki dotyczące rozproszonych baz danych wraz z wiedzą naukową lekarzy oraz biologów, tak by stworzyć system umożliwiający wspomaganie ich codziennej pracy.

1.2 Cel pracy

Celem pracy inżynierskiej jest opracowanie interfejsu użytkownika do wspomnianej wcześniej rozproszonej bazy danych. Myślą przewodnią projektowania systemu było stworzenie aplikacji prezentującej dane z sekwencjonowania DNA, w postaci zaadnotowanej listy wariantów użytkownikowi, którą będzie można spersonalizować i będzie łatwo konfigurowalna. W celu ograniczenia pracy administratora systemu do całkowitego minimum zaimplementowano strukturę projektu tak, aby rozwijanie go poprzez dodawanie nowych funkcjonalności było prowadzone niskim kosztem.

Podczas projektowania aplikacji należało zwrócić uwagę na liczbę i rodzaj danych, co wymusiło zaprojektowanie specyficznego systemu filtrującego, w celu umożliwienia ograniczenia widocznych danych do tylko tych potrzebnych klientowi aplikacji.

1.3 Zawartość rozdziałów

Drugi rozdział pracy stara się przybliżyć specyfikę danych. Odbywa się to poprzez wyjaśnienie biologicznych aspektów pracy, to jest wyjaśnienia pochodzenia danych oraz terminologii używanej w pracy inżynierskiej. Następny rozdział przedstawia grupę docelowych użytkowników oraz zawiera zebrane wymagania funkcjonalne i нефункционалне aplikacji. W czwartym rozdziale przedstawione są istniejące rozwiązania problemu, wraz z argumentacją, dlaczego nie zostały wykorzystane. Następnie opisane zostaną technologie użyte do implementacji systemu oraz te odrzucone w czasie projektowania.

W szóstym rozdziale opisane zostały zaimplementowane funkcjonalności wraz z ilustracjami ich użycia. Kolejny rozdział skupia się na prezentacji zaprojektowanej bazy danych i wyjaśnienia jej schematu. Ósmy rozdział opisuje optymalizację działania głównej części aplikacji poprzez wyjaśnienia aspektów spowalniających odpowiedź serwera oraz wydłużających czas rysowania się elementów w przeglądarce. W dziewiątym rozdziale opisane zostały elementy odpowiedzialne za bezpieczeństwo aplikacji i jej użytkowników, a dziesiąty rozdział opisuje implementację części funkcjonalności oraz strukturę folderów projektu. Na końcu zawarto wnioski i podsumowania.

2 Specyfika danych

W celu zrozumienia specyfiki danych należy przedstawić ich pochodzenie i znaczenie.

DNA (kwas deoksyrybonukleinowy) [1] to związek organiczny zlokalizowany w jądrach komórkowych. Struktura DNA odpowiada między innymi za kodowanie białek [2] [3] pełniących w organizmie wiele ważnych funkcji (np. transport tlenu między tkankami, za co odpowiedzialna jest między innymi hemoglobina).

Podstawowymi elementami budującymi nasz kod genetyczny są nukleotydy [4] [5]. Sposób, w jaki są ustawione w sekwencji, odpowiada za ostateczną budowę białka. Genom ludzki ma w przybliżeniu 3 miliardy nukleotydów, z których około 50 milionów zajmuje się właściwym kodowaniem 25 tysięcy białek.

Warianty DNA to wszelkie różnice pomiędzy genomem osoby badanej, a publicznie dostępnym genomem referencyjnym.

Powyższa różnica może spowodować poważne zakłócenia w funkcjonowaniu naszego organizmu, a warianty są elementami, które starają się badać lekarze poprzez wyszukiwanie tych, które mogą powodować choroby. Białko powstaje w procesie transkrypcji [6], czyli przepisania DNA na RNA oraz translacji, w której RNA zamieniane jest na ciąg aminokwasów. Na potrzeby tej pracy, słowa „gen” i „transkrypt” będą używane zamiennie. Jeden wiersz w rozproszonej bazie danych, która stanowi główne źródło danych niniejszej aplikacji, zawiera dokładne informacje na temat danego wariantu DNA, w tym dane o:

- chromosomie owego wariantu,
- położeniu wariantu na chromosomie,
- referencyjnej wartości nukleotydu,
- wartości nukleotydu, na którą zmienił wariant,
- częstości występowania tego wariantu w referencyjnych bazach danych,
- genie i eksonie, w którym znajduje się wariant
- efekcie białkowym
- wynikach algorytmów predykcyjnych, np. SIFT, Polyphen2, MutationTaster itd.

- częstości występowania w kontrolnych bazach danych, np. gnomad ExAC, 1000 genomes itd.
- patogenności wariantu, z baz ClinVar, HGMD
- powiązaniu danego genu z chorobą (OMIM)

Potencjalna liczba kolumn może dochodzić do kilkuset i nie jest z góry ograniczona, dlatego system został zaprojektowany tak, by być niezależnym od modelu bazy danych wariantów.

3 Wymagania funkcjonalne i нефункционалне

Określenie funkcjonalności dostępnych w budowanej aplikacji, rozpoczęto od zdefiniowania rodzajów użytkowników, którzy mają korzystać z oprogramowania.

Grupą docelową są analitycy (przede wszystkim diagnosty laboratoryjni), którzy będą poszukiwali wariantów odpowiedzialnych za chorobę pacjenta, aby wykryć niebezpieczeństwa i móc jak najwcześniej postawić diagnozę. Dane będą rozpatrywać w kontekście jednego pacjenta i należy umożliwić łatwe rozróżnienie genotypów, czyli wystąpień wariantów u poszczególnych pacjentów.

Inną istotną kwestią wziętą pod uwagę był typ i wielkość danych, jakie mają być wyświetlane użytkownikom. W trakcie projektowania architektury jako dane przykładowe została pobrana lista wariantów dla jednego transkryptu dostępne w aplikacji ExAC instytutu Broad [10] [11]. Dane te podobnie jak te w rozproszonej bazie danych posiadały kilkadziesiąt kolumn i oczywistym wydało się, że różnych użytkowników będzie interesowała tylko część informacji o wariantach i należałoby umożliwić im filtrację oraz zakrywanie niepotrzebnych kolumn. Jedna próbka z docelowej bazy danych liczy sobie więcej niż 20000 wierszy, co wymogło zaproponowania funkcjonalności umożliwiające na poprawne i intuicyjne ich filtrowanie tak, aby użytkownik otrzymywał tylko interesujące go rekordy.

3.1 Wymagania funkcjonalne

Zostały określone następujące funkcjonalności. Aplikacja:

- 1) rejestruje użytkowników,
- 2) autoryzuje użytkowników,
- 3) umożliwia wybór próbki do analizy,
- 4) pozwala na wprowadzenie wcześniej zdefiniowanych filtrów z panelu administratora,
- 5) wyświetla dane (listę wariantów w postaci tabeli) z sekwencjonowania DNA dla konkretnej próbki,
- 6) filtruje dane po stronie serwera (na podstawie ustawionych przez użytkownika filtrów) i wysyła je użytkownikowi,

- 7) zlicza dane przy zadanych filtrach i informuje użytkownika o wyniku,
- 8) umożliwia zmianę wartości filtrów,
- 9) pozwala na wyłączenie z filtracji dowolnej części filtrów,
- 10) zapisuje wartości filtrów oddzielnie dla każdego użytkownika,
- 11) sortuje dane po stronie użytkownika,
- 12) filtruje dane po stronie użytkownika,
- 13) udostępnia administratorowi możliwość zmiany dostępu do próbek każdego użytkownika,
- 14) daje możliwość zakrycia na stronie aplikacji części danych,

Funkcjonalność umożliwiającą wprowadzanie wcześniej zdefiniowanych filtrów spowodowała stworzenie specjalnej klasy użytkowników, to jest administratorów, którzy będą zarządzali strukturą filtrów poprzez wprowadzenie odpowiedniego pliku ze specjalnie przygotowanego panelu administracyjnego oraz będą zarządzać dostępem do próbek dla użytkowników.

3.2 Wymagania niefunkcjonalne

Aplikacja:

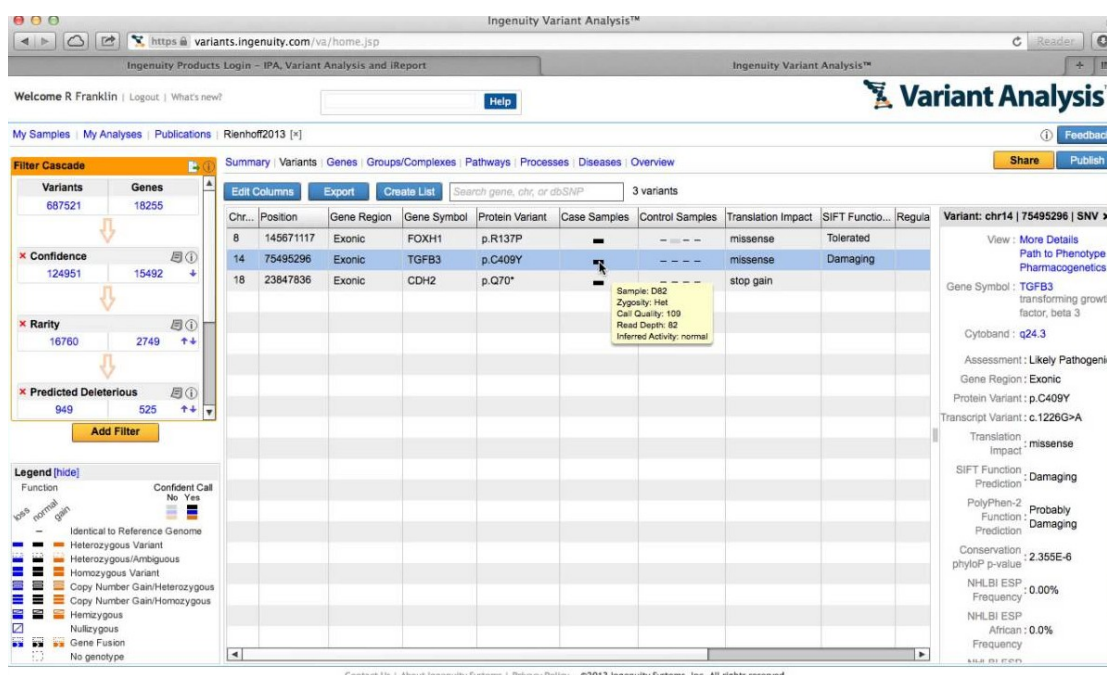
- 1) szyfruje wysyłane dane między klientem a serwerem za pomocą protokołu HTTPS,
- 2) wykorzystuje funkcję SHA-512 [30] do zabezpieczenia hasła użytkownika,
- 3) korzysta z „soli” przy wyliczaniu funkcji skrótu,
- 4) wykorzystuje darmowe oprogramowanie,

4 Istniejące rozwiązania

Konieczność posiadania aplikacji umożliwiającej analizę danych z genomów jest powszechnie znana od dłuższego czasu. W związku z powyższym powstało wiele rozwiązań, komercyjnych i niekomercyjnych, które starają się sprostać wymaganiom wszystkich użytkowników. W tej sekcji przedstawione zostaną najpopularniejsze opcje.

Ingenuity Variant Analysis Jednym z przodujących rozwiązań jest aplikacja Ingenuity Variant Analysis [7] firmy Qiagen. To zaawansowane narzędzie poza wyświetlaniem danych wariantów, umożliwia między innymi:

- zaawansowane filtrowanie,
- eksporty wyników do plików,
- udostępnienie swojego raportu innemu użytkownikowi,



Rysunek 1. Zrzut ekranu z aplikacji Ingenuity Variant Analysis. Strona przedstawia listę wariantów dla różnych genów oraz moduł filtrujący

Źródło elementów graficznych: <https://i.ytimg.com/vi/icQdej79vUc/maxresdefault.jpg>.

Jest to jednak oprogramowanie komercyjne, co wykluczyło go z dalszych rozważań.

ExAC Browser ExAC Browser jest aplikacją, tworzoną przez koalicję badaczy i informatyków, w celu udostępnienia danych kodujących białka szerszemu gronu badaczy. Dokonują tego poprzez harmonizację i agregację danych z wielu projektów. Jednymi z najmocniejszych zalet aplikacji jest prezentacja danych w postaci tabel, wykresów oraz umożliwianie eksportu danych. Mimo iż jest to wolne oprogramowanie, nie zdecydowano się na jego wykorzystanie, ponieważ:

- 1) podczas rozważań oprogramowanie było dopiero w wersji beta,
- 2) brakowało funkcjonalności filtrowania danych transkryptu,
- 3) brakowało rozróżnienia użytkowników, a to powodowało:
 - brak możliwości personalizacji aplikacji dla użytkownika,
 - brak kontroli nad dostępami do próbek,
- 4) ExAC browser jest przeznaczony przede wszystkim do prezentacji informacji o wariantach oraz ich częstości w populacji, jednak nie ma możliwości przeglądania danych pojedynczych pacjentów,

Save coverage plot
Save exon image
Save CNV image

All
Missense + LoF
LoF
☐ Include filtered (non-PASS) variants
☐ Invert (highlight rare variants)

Export table to CSV

Variant	Chrom	Position	Consequence	Filter	Annotation	Flags	Allele Count	Allele Number	Number of Homozygotes	Allele Frequency	
22-46594230 T / A	22	46594230		PASS	Intron		1	120270	0	0.000008315	
22-46594236 C / T	22	46594236	c.-42-3C>T	PASS	splice region		1	120124	0	0.000008325	
22-46594241 G / T	22	46594241	c.-40G>T	PASS	splice region		1	120166	0	0.000008322	
22-46594246 G / T	22	46594246		PASS	5' UTR		1	120188	0	0.000008320	
22-46594249 C / T	22	46594249		PASS	5' UTR		1	120180	0	0.000008321	
22-46594251 C / T	22	46594251		PASS	5' UTR		3	120174	0	0.00002496	
22-46594254 C / T	22	46594254		PASS	5' UTR		5	120306	0	0.00004156	
22-46594254 C / A	22	46594254		PASS	5' UTR		3	120306	0	0.00002494	
22-46594255 G / A	22	46594255		PASS	5' UTR		1	120320	0	0.000008311	
22-46594259 C / T	22	46594259		PASS	5' UTR		1	120410	0	0.000008305	
22-46594261 A / G	22	46594261		PASS	5' UTR		1	120436	0	0.000008303	
22-46594263 C / T	22	46594263		PASS	5' UTR		1	120452	0	0.000008302	
22-46594264 A / G	22	46594264		PASS	5' UTR		4	120468	0	0.00003320	
22-46594271 T / C (rs4253793)	22	46594271		PASS	5' UTR		90	120560	0	0.0007465	
22-46594277 C / T	22	46594277		PASS	5' UTR		2	120656	0	0.00001658	
22-46594279 C / T	22	46594279		PASS	5' UTR		46	120716	0	0.0003811	

Rysunek 2. Zrzut ekranu z aplikacji ExAC Browser. Strona przedstawia listę wariantów dla wybranego genu

VCF-Miner Kolejną aplikacją jest VCF-Miner’a [13] [14] pozwalający na pracę z próbkami w formacie VCF [15]. Jest to format pliku tekstowego używanego do przechowywania danych z sekwencjonowania DNA.

VCF-Miner’owi mimo swoich wielu zalet takich jak na przykład możliwość załadowania własnego pliku VCF, brakuje kilku istotnych funkcjonalności, które zostały zaimplementowane w aplikacji, której dotyczy ta praca inżynierska. Są nimi:

- 1) brak predefiniowanych filtrów,
- 2) trudna w obsłudze filtracja danych, to znaczy:
 - brak możliwości filtracji pobranych już danych po stronie przeglądarki,
 - brak możliwości wyłączenia jednego filtru bez usuwania go w całości,
 - możliwość usunięcia tylko ostatniego w kolejce filtru,
- 3) brak zapisu ukrytych kolumn — użytkownik przy każdej próbkce musi ponownie ukrywać bądź odsłaniać interesujące go kolumny,
- 4) brak ograniczanie dostępu do próbek — każdy użytkownik ma dostęp do wszystkich próbek,

CHROM	POS	ID	REF	ALT	#_Samples	Samples
10	5193864	rs11253021	C	T	50	NA06984 NA06985 NA06989 NA07000 NA07037 NA07048 NA07346 NA07347 NA07357 NA10847 NA11829 NA11830 NA11831 NA11832 NA11840 NA11881 NA11883 NA11918 NA11920 NA11930 NA11992 NA11995 NA12005 NA12044 NA12058 NA12144 NA12154 NA12155 NA12156 NA12249 NA12283 NA12342 NA12347 NA12383 NA12546 NA12716 NA12717 NA12718 NA12748 NA12749 NA12751 NA12775 NA12776 NA12778 NA12812 NA12815 NA12828 NA12872 NA12873 NA12892
10	5193865	rs61729616	G	A	1	NA06994
10	5194928	rs7097295	C	T	47	NA06984 NA06985 NA07000 NA07037 NA07346 NA07347 NA07357 NA10847 NA11829 NA11830 NA11831 NA11832 NA11840 NA11881 NA11883 NA11918 NA11920 NA11930 NA11992 NA11995 NA12005 NA12044 NA12058 NA12144 NA12154 NA12155 NA12156 NA12249 NA12283 NA12342 NA12347 NA12383 NA12546 NA12716 NA12717 NA12718 NA12748 NA12749 NA12751 NA12775 NA12776 NA12778 NA12812 NA12815 NA12828 NA12872 NA12873 NA12892
10	5531183	rs10904517	C	T	27	NA06984 NA06985 NA06989 NA07347 NA11831 NA11920 NA11930 NA11994 NA11995 NA12003 NA12045 NA12154 NA12156 NA12234 NA12275 NA12340 NA12414 NA12717 NA12748 NA12760 NA12814 NA12815 NA12829 NA12842 NA12873 NA12878 NA12890
10	14930835	rs45567440	C	A	1	NA12776
10	14936199	rs61749161	G	A	1	NA11843
10	14990599	rs41300676	G	A	1	NA07347
10	15016420	rs7076862	G	A	26	NA07000 NA07037 NA07051 NA07347 NA10851 NA11843 NA11893 NA11918 NA11930 NA12005 NA12043 NA12144 NA12154 NA12283 NA12286 NA12287 NA12347 NA12413 NA12489 NA12748 NA12762 NA12776 NA12814 NA12829 NA12843 NA12892
10	15016733	rs35441642	G	C	11	NA07357 NA11829 NA11831 NA11930 NA12003 NA12282 NA12413 NA12718 NA12749 NA12878 NA12891
10	15017475	rs41297018	C	T	5	NA07000 NA11994 NA12045 NA12144 NA12716
10	45119340	.	C	A	1	NA06986
10	46506921	rs59974223	C	T	22	NA07037 NA07347 NA11830 NA11881 NA11920 NA11930 NA12043 NA12058 NA12272 NA12273 NA12282 NA12341 NA12347 NA12400 NA12413 NA12489 NA12748 NA12749 NA12751 NA12814 NA12889 NA12892
10	46506984	rs3740294	G	A	22	NA07037 NA07347 NA11830 NA11881 NA11920 NA11930 NA12043 NA12058 NA12272 NA12273 NA12282 NA12341 NA12347 NA12400 NA12413 NA12489 NA12748 NA12749 NA12751 NA12814 NA12873 NA12892
10	46507084	rs2229967	G	T	39	NA06989 NA06994 NA07000 NA07048 NA07051 NA07357 NA10851 NA11832 NA11843 NA11918 NA11919 NA11992 NA11994 NA12044 NA12045 NA12144 NA12156 NA12286 NA12340 NA12348 NA12413 NA12414 NA12717 NA12718 NA12748 NA12751 NA12760 NA12761 NA12762 NA12763 NA12775 NA12812 NA12829 NA12842 NA12843 NA12872 NA12873 NA12890

Rysunek 3. Zrzut ekranu z aplikacji VCF-Miner. Strona przedstawia listę wariantów dla wybranego genu oraz moduł filtrujący

5 Wybór technologii

Platforma klastrowego przetwarzania danych — Apache Spark[9], z którą współpracować będzie aplikacja, została stworzona oraz udostępnia interfejs programistyczny w języku Scala. Naturalnym przez to wydało się wybranie tego języka programowania do stworzenia przeglądarki danych.

5.1 Język programowania Scala

W aplikacji użyto języka Scala w wersji 2.11.7 [8]. Jest to język programowania powstały w 2001 roku pod kierownictwem Martina Odersky'ego w Lozannie. Działa na Wirtualnej Maszynie Javy a do 2012 roku wspierała platformę .NET opracowaną przez firmę Microsoft.

Język ten nadaje się równie dobrze do krótkich, zwartych skryptów wywoływanych podobnie do skryptów języka Python, jak i do tworzenia wydajnych, ogromnych, bezpiecznych systemów sieciowych. Jest językiem łączącym cechy języków funkcyjnych oraz obiektowych. Nie jest jednak obligatoryjny funkcyjny styl programowania, do którego nie jest przyzwyczajona większość programistów.

Scala w swoim założeniu, nawiązuje do minimalizmu składni Lispa, to znaczy, że nie opiera się na składni, ale na funkcjach bibliotecznych. Nazwa ma podkreślić skalowalność języka, dzieje się tak dzięki możliwości tworzenia dodatkowych typów i struktur wyglądających jak nowa składnia języka. Zaletą języka jest również to, że dzięki kompatybilności z językiem Java mamy możliwość wykorzystania każdej linii kodu napisanej w owym języku.

5.2 System zarządzania bazą danych

Zadanie opracowania bazy danych przechowującej informacje konfiguracyjne, dane użytkowników oraz o użytkownikach było dużą częścią tworzenia systemu i wymagało wybrania odpowiedniego systemu zarządzania bazą danych. Model bazodanowy został zaprojektowany w modelu opartym na relacyjnej organizacji danych, przez co wybór ograniczył się do darmowych technologii realizujących relacyjne bazy danych.

Biorąc pod uwagę powyższe kryteria, można porównać najpopularniejsze systemami, są nimi[17]:

- MySQL,
- SQLite,
- PostgreSQL,

5.2.1 MySQL

Zalety

- proste i łatwe w obsłudze,
- wysoki poziom bezpieczeństwa,

Wady

- nie realizuje w pełni standardu SQL,
- problematyczny jednoczesny zapis i odczyt,

5.2.2 SQLite

Zalety

- zgodny ze standardem SQL,
- przenośny dzięki oparciu bazy o jeden plik,

Wady

- brak zarządzania użytkownikami i dostępami do danych,

5.2.3 PostgreSQL

Zalety

- zgodny ze standardem SQL,
- wsparcie dla współbieżności,
- pełne wsparcie dla transakcji,

Wady

- słaba wydajność,
- trudność instalacji dla początkujących użytkowników,

5.2.4 Uzasadnienie wyboru PostgreSQL

Po analizie ostateczny wybór systemem padł na PostgreSQL. To otwarte i darmowe oprogramowanie posiada bardzo dużą społeczność, której wiedza jest łatwo dostępna w internecie i posiada wiele narzędzi i bibliotek przeznaczonych do pracy z owym systemem. Istotny wpływ na decyzję miała również łatwość integracji PostgreSQL na inne systemy.

5.3 Slick

Pracę z bazą danych po stronie serwera aplikacyjnego znacznie ułatwia oprogramowanie pozwalające na odwzorowanie obiektowo-relacyjne tabel bazodanowych na obiekty języka programowania. Dzięki tej technice programista może traktować obiekty bazodanowe jak elementy kolekcji czy pola obiektów.

Takim narzędziem jest stworzone przez firmę Lightbend, Inc. oprogramowanie Slick[18] pozwalające na pełną kontrolę nad bazą danych oraz pisanie klasycznych zapytań SQL.

5.4 JDBC

Baza danych zawierająca dane o sekwencjonowaniu DNA jest kolumnowo zorientowaną bazą danych opartą na systemie Apache Kudu [19]. W przeciwieństwie do tradycyjnych systemów bazodanowych gdzie dane składowane są poziomo, czyli wierszami, w kolumnowych systemach dane przechowywane są pionowo — kolumnami. Kudu zajmuje się składowaniem danych, za wykonywanie zapytań SQL odpowiada całkowicie zintegrowane z Kudu oprogramowanie Impala [20] [21] udostępniające interfejs JDBC.

5.5 Aplikacja przeglądarkowa

Biorąc pod uwagę wymagania klientów oraz różnorodność używanych przez nich urządzeń należało wybrać odpowiedni rodzaj aplikacji klienckiej pozwalający na spełnienie wszystkich wymagań funkcjonalnych naszych użytkowników oraz jednocześnie będący łatwy w utrzymaniu i rozwijaniu.

Platforma programistyczna Play Platforma Play, stworzona w języku Scala jest środowiskiem do tworzenia aplikacji internetowych, która na celu ma przyspieszyć pracę programisty dzięki:

- strategii Konwencji Ponad Konfigurację,
- przeładowywania i ponownej kompilacji plików po edycji,
- wykorzystaniu wzorca Model-Widok-Kontroler,
- wykorzystaniu technologii REST,

Platforma programistyczna AngularJS AngularJS jest opracowaną przez Google biblioteką wspomagającą tworzenie aplikacji przeglądarkowych na jednej stronie. Jej głównymi zaletami jest:

- oddzielenie warstwy klienckiej od warstwy serwerowej,
- oddzielenie manipulacji modelu dokumentu HTML od logiki aplikacji,
- wykorzystaniu wzorca Model-Widok-Kontroler,

ECMASCRIPT 6 Część przeglądarkowa została napisana z wykorzystaniem dodatkowych funkcjonalności języka JavaScript w standardzie ECMAScript 2015 [22].

Lodash Biblioteka Lodash [23] jest zbiorem wielu użytecznych funkcji ułatwiających pracę z typami dostępnymi w języku JavaScript.

6 Przypadki użycia

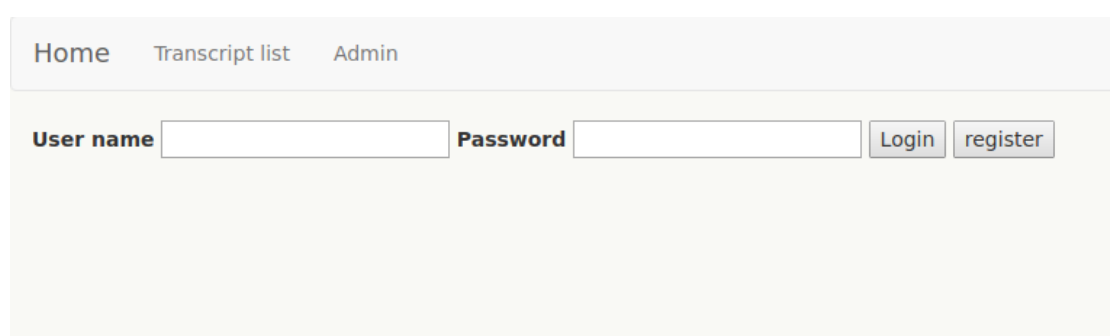
6.1 Autoryzacja

6.1.1 Role

Administratorem nazywana jest osoba mająca dostęp do bazy danych aplikacji, ustalająca widoczne dla użytkowników próbki oraz zarządzająca strukturą filtrów. Jednocześnie administrator ma takie same możliwości jak zwykły użytkownik, którymi są dostęp do określonych próbek, możliwość ich oglądania, filtrowania i ukrywania kolumn. Każdy użytkownik jest rozróżnialny w aplikacji dlatego, wymagana jest wcześniejsza rejestracja.

6.1.2 Rejestracja i logowanie użytkownika

Poniższy rysunek pokazuje widok startowy aplikacji. W celu przejścia do pozostałych funkcjonalności użytkownik musi się zalogować lub zarejestrować, jeśli nie ma jeszcze konta w aplikacji. Wskazane jest użycie szyfrowanego połączenia HTTPS, żeby uniknąć kradzieży hasła.



Home Transcript list Admin

User name Password Login register

Rysunek 4. Okno logowania i rejestracji

6.2 Przeglądanie danych z sekwencjonowania DNA

Po zalogowaniu się do aplikacji przed użytkownikiem pojawia się główna część aplikacji pozwalająca na dostęp do próbek oraz pracę na nich.

Ekran dostępnych genomów Każda próbka (odpowiadająca wynikom jednego pacjenta), dostępna w bazie danych posiada indywidualny identyfikator pozwalający na rozróżnienie jej od innych próbek. Ów identyfikator jest ciągiem znaków, który wyświetlany jest alfabetycznie posortowanej liście. Użytkownik widzi tylko próbki udostępnione przez administratora i ma możliwość spojrzenia dokładniej w dane poprzez kliknięcie identyfikator próbki, co przeniesie go do następnej strony prezentującej dane z sekwencjonowania DNA.

Home	Sample list	Admin
Sample id		
sample_id_1		
sample_id_3		
sample_id_4		
sample_id_5		
sample_id_6		
sample_id_7		
sample_id_8		

Rysunek 5. Lista dostępnych użytkownikowi próbek

Tabela z danymi z sekwencjonowania DNA Przechodząc na stronę z danymi dla konkretnej próbki, użytkownikowi prezentowane są dane w postaci tabelarycznej (Rysunek 6).

Chrom	Position	Allele Count European (Non-F...	Allele Number European (Non-F...	Homozygote Count European (N...	Allele Count Fi...	Allele Number Fi...	Homozyg...
22	46644047	1	52664	0	0	4884	
22	46644046	0	52534	0	0	4884	
22	46644047	1	52664	0	0	4884	
22	46644046	0	52534	0	0	4884	
22	46644041	1	50662	0	0	4730	
22	46644041	1	50662	0	0	4730	
22	46644047	1	52664	0	0	4884	
22	46644046	0	52534	0	0	4884	
22	46644041	1	50662	0	0	4730	
22	46644047	1	52664	0	0	4884	
22	46644046	0	52534	0	0	4884	
22	46644041	1	50662	0	0	4730	
22	46644204	0	45122	0	0	4322	
22	46644202	1	45802	0	0	4378	
22	46644199	0	47728	0	0	4586	
22	46644198	0	48086	0	0	4620	
22	46644194	0	48652	0	0	4656	
22	46644188	2	50554	0	0	4828	
22	46644178	0	54226	0	0	5164	
22	46644177	3397	55608	98	235	5264	

Rysunek 6. Tabela z danymi z sekwencjonowania DNA

Po zakończeniu implementacji wyświetlania tabeli z danymi z sekwencjonowania DNA zauważono problemy z wydajnością. Długi czas tworzenia się elementów HTML tabeli i łatwo zauważalne zawieszanie się przeglądarki były elementami nie do przyjęcia dla codziennej pracy użytkownika. Pod obserwację wzięto:

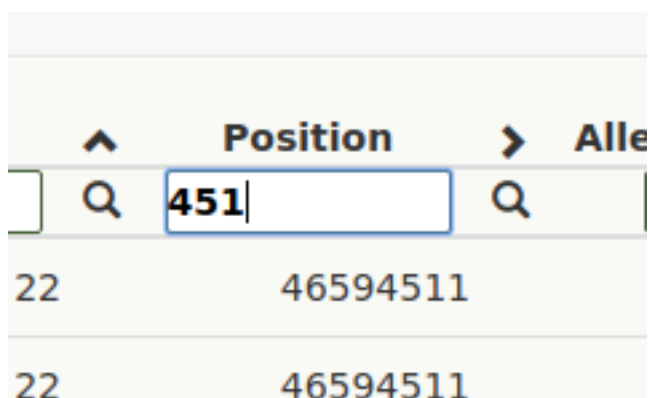
- czas odpowiedzi serwera na zapytanie HTTP,
- czas wygenerowania i wykonania zapytania SQL,
- czas wygenerowania przez przeglądarkę elementów HTML tabeli,

Wykonane testy wykazały, że duży narzut czasowy na ładowanie się strony, miał ostatni element, to znaczy rysowanie tabeli przez przeglądarkę.

Wada ta została zniwelowana poprzez wprowadzenie paginacji zwanej też stronicowaniem. Maksymalna liczba pokazywanych wierszy została ograniczona do 300 i wprowadzono dodatkowy element widoczny w lewym dolnym rogu Rysunku 6, który umożliwia użytkownikowi poruszanie się po kolejnych stronach tabelki, zmniejsz-

szając narzut pamięci operacyjnej wymaganej do wygenerowania całości tabeli.

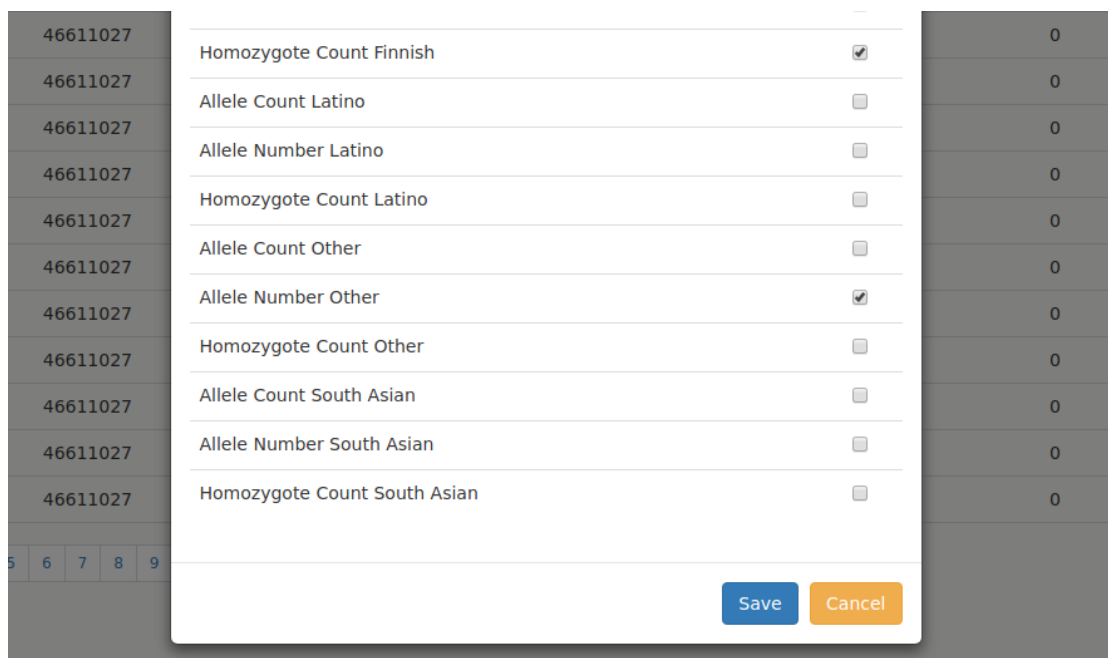
Ważną funkcjonalnością z punktu widzenia użytkownika jest możliwość filtracji pobranych już wierszy z bazy danych. W celu zwiększenia możliwości wyszukiwania, każda kolumna posiada oddzielne pole filtrujące, umożliwiając na zawężanie zbioru danych po każdej kolumnie. Element odpowiadający za stronicowanie poprawnie zmniejsza liczbę dostępnych stron przy dynamicznym zmniejszaniu się wyświetlanych danych. Dodatkową opcją działającą po stronie przeglądarki jest funkcjonalność sortowania rosnąco bądź malejąco jednej kolumny. Służy do tego strzałka po lewej stronie od nazwy kolumny, kliknięcie w ikonkę bądź nazwę kolumny zmienia sortowanie.



	^	Position	>	Alle
	Q	451	Q	
22		46594511		
22		46594511		

Rysunek 7. Sortowanie i filtrowanie pobranych danych po stronie przeglądarki

Ustawianie widoczności kolumn Przejrzystość danych i dostęp tylko do potrzebnych informacji jest kluczową wartością dla użytkowników. Mnogość kolumn, prowadząca do przedstawiania wielu informacji niepotrzebnych wszystkim użytkownikom uniemożliwia osiągnięcie tego efektu. Wychodząc naprzeciw tym oczekiwaniom, zaimplementowano opcję umożliwiającą klientom aplikacji ukrywanie dowolnej kolumny. Po kliknięciu w specjalny guzik umiejscowiony po prawej stronie elementu stronicującego, wyskakuje okienko z listą kolumn, które użytkownik może odznaczyć, co spowoduje zniknięcie danej kolumny z tabeli. Wybór ukrytych kolumn może być zapisany w bazie danych tak by przy ponownym wejściu na tę stronę aplikacji, użytkownik nie musiał kolejny raz ukrywać nieinteresujących go kolumn. Po rejestracji użytkownik ma widoczne wszystkie kolumny i musi sam je odznaczyć.



Rysunek 8. Okno z listą widocznych kolumn tabeli z przyciskami wyboru, zapisu i zamknięcia okna

Filtrowanie danych Rysunek 9 przedstawia moduł filtrujący dane, znajdujący się w lewej części strony aplikacji. Podstawowym elementem tworzącym filtr jest tak zwane „pole”, odnosi się ono do jednej kolumny bazodanowej, z którą łączy ją jedna z poniższych relacji:

- mniejsze („less”),
- większe („greater”),
- mniejsze równe („less than”),
- większe równe („greater than”),
- równe („equals”),

Użytkownik wprowadza własną wartość do pola bądź wybiera ją z listy rozwijanej. Dodatkową możliwością jest pole mające wartość domyślną ustaloną przez administratora. Wartość każdego pola może być zapisane w bazie danych, zapis jest oddzielny, dla każdego użytkownika, każdego filtru oraz każdej próbki. Grupa pól tworzy właściwy „filtr”, który może być wyłączony z filtracji dzięki przyciskowi wyboru umiejscowionego po prawej stronie nazwy. Wizualnie filtr jest wyciemniony jak na przykładzie. Grupa filtrów tworzy tak zwany panel, które są oddzielnymi bytami w bazie danych, o własnych nazwach i filtrach. W górnej części panelu widoczne są przyciski z nazwami paneli. Aktywny panel wyróżnia się od zielonym kolorem od nieaktywnych o kolorze szarym.

Filters

Tab 1

Tab 2

Tab 3

Filter 11 ☒

Chrom greater than

22

Position greater

46594236

Allele Count East Asian greater

Filter 12 ☒

Reference equals

T

Chrom less than

23

Allele Number greater

8650

Filter 13 ☒

Annotation equals

intron

Allele Number Latino greater

0

Filter

Count

Save

Get all

Rysunek 9. Zrzut widoku paneli filtrujących

Filtrowanie odbywa się zgodnie z kolejnością aktywnych filtrów, to znaczy najpierw filtrujemy dane, używając pól pierwszego aktywnego filtra, następnie te dane filtrujemy, korzystając z drugiego etc. Ilustruje to przykład z Rysunku 10, a Tabela 1 przedstawia rozmiar danych, który zostałby pobrany dla każdego filtra.

Przypadek 1 na diagramie ilustruje sytuację, gdy tylko jeden filtr jest aktywny. Wynikiem filtracji jest wtedy 3840, czyli tyle ile ograniczają warunki Filtru numer 3. Przypadek 2 przedstawia zdarzenie, gdy włączone są dwa filtry: „Filtr 1” oraz „Filtr 3”. Następuje filtracja danych względem najwyższego filtra, a następnie ostatniego, z pominięciem wszystkich filtrów pomiędzy nimi. Dane są ograniczone w tym przypadku do 3584. Ostatnia rubryka na diagramie ilustruje moment, gdy wszystkie filtry są aktywne. W takiej sytuacji najpierw filtruje się dane względem warunków „Filtr 1”, następnie „Filtru 2” i na końcu „Filtru 3”. W ten sposób ograniczamy liczbę pobranych rekordów do 768.

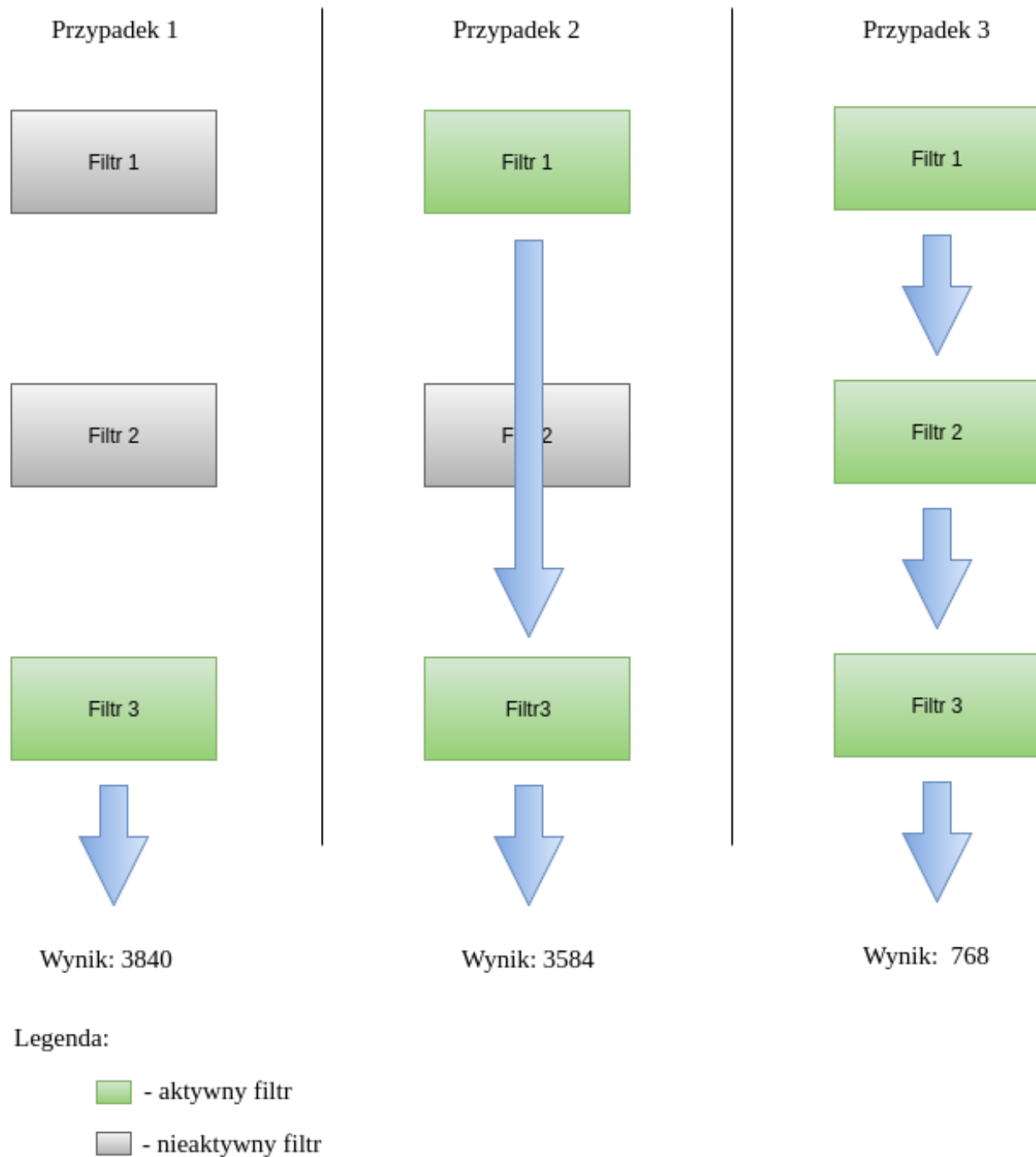
	Filtr 1	Filtr 2	Filtr 3
Wyniki zliczania danych dla każdego przykładowego filtra	19712	1792	3840

Tabela 1. Wyniki zliczania danych dla każdego przykładowego filtra

Przycisk z etykietą „Count” umożliwia policzenie ilości wierszy zwróconych przy zadanych wartościach pól. Pozwala to na sprawdzenie rozmiaru danych przed właściwym pobraniem całego zbioru danych. Efekt takiego działania jest widoczny na Rysunku 11.

Można również pobrać wszystkie dane wybranej próbki za pomocą guzika z napisem „Get all”.

Za ustalanie struktury filtrów odpowiadają administratorzy i w podrozdziale im poświęconym opisane zostanie zarządzanie filtrami.



Rysunek 10. Diagram działania panelu filtrującego

Filters

Tab 1**Tab 2****Tab 3**

Filter 11 ☒
Chrom greater than
22
Position greater
46594236
Allele Count East Asian greater

Selection reduce list size to 19712

Filter 12 ☐
Reference equals
T
Chrom less than
23
Allele Number greater
8650

Filter 13 ☒
Annotation equals
intron
Allele Number Latino greater
0
Selection reduce list size to 3584

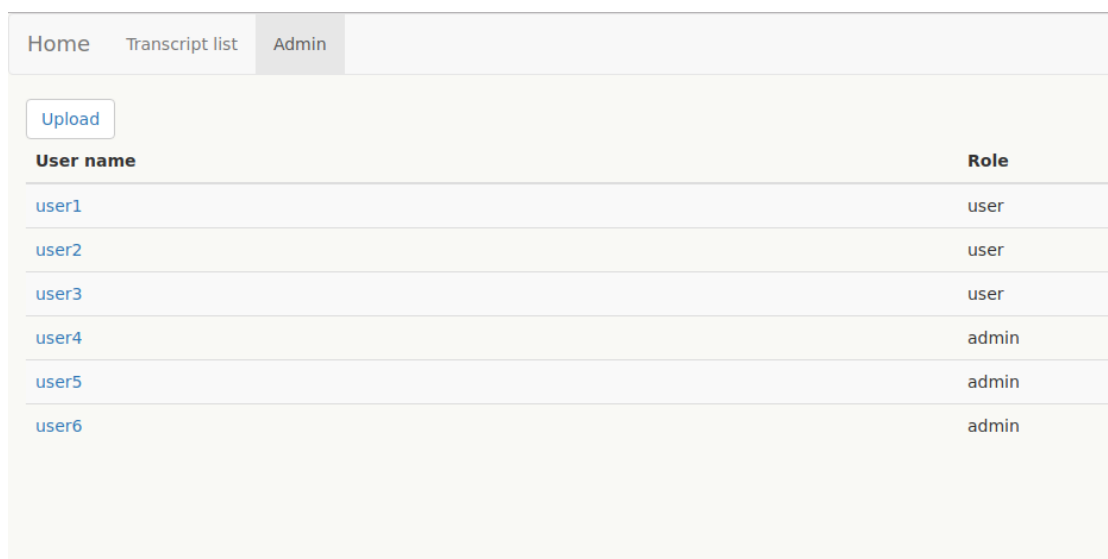
Filter**Count****Save****Get all**

Rysunek 11. Komponent paneli filtrujących po zliczeniu danych

6.3 Panel administratora

Użytkownik o roli administratora może wejść do oddzielnej strony aplikacji (Rysunek 12). Administratorowi prezentowana jest lista zarejestrowanych użytkowników wraz z ich rolami. Każdy element list jest hiperłączem prowadzącym do strony poświęconej konkretnemu użytkownikowi i jego uprawnieniom.

Przycisk o etykiecie „Upload” służy administratorowi do zmiany struktury filtrów, o czym będzie mowa w następnym paragrafie.



The screenshot shows a web application interface for an administrator. At the top, there is a navigation bar with three tabs: 'Home', 'Transcript list', and 'Admin'. The 'Admin' tab is currently selected. Below the navigation bar, there is a button labeled 'Upload'. Underneath the button is a table with two columns: 'User name' and 'Role'. The table contains six rows of user data.

User name	Role
user1	user
user2	user
user3	user
user4	admin
user5	admin
user6	admin

Rysunek 12. Panel administratora z listą użytkowników

Zarządzanie filtrami Zmiana struktury filtrów odbywa się z poziomu panelu administratora poprzez wysłanie z przeglądarki na serwer pliku arkusza kalkulacyjnego o ustalonym formacie. Przykładowe dane są widoczne na Rysunku 13. W pierwszej kolumnie podaje się nazwę zakładki, do której przypisywany jest filtr o nazwie podanej w drugiej kolumnie. W trzeciej kolumnie należy podać nazwę kolumny, do której odnosić się będzie pole filtru, a następnie relację między wprowadzaną wartością a kolumną. Opcjonalnie może być podana domyślna wartość pola oraz kilka wartości oddzielone przecinkami, między którymi użytkownik będzie mógł wybierać w aplikacji. Nazwy kolumn oraz relacje są dostępne pod listą rozwijaną w arkuszu w celu ułatwienia pracy administratora. Załadowanie pliku usuwa wcześniejsze filtry oraz zapisane wartości użytkowników.

	A	B	C	D	E	F
1	Tab name	Filter name	Variant column name	Relation	Default value	Options
2	Tab 1	Filter 11	Chrom	Greater than	22	
3	Tab 1	Filter 11	Position	Greater		23686, 245345, 456846
4	Tab 1	Filter 12	Reference	Equals	C	C,G
5	Tab 1	Filter 12	Chrom	Less than	23	
6	Tab 1	Filter 12	Allele Number	Greater	42920	
7	Tab 2	Filter 21	Chrom	Greater than	22	
8	Tab 2	Filter 21	Position	Greater		23686
9	Tab 2	Filter 22	Reference	Equals	G	
10	Tab 2	Filter 22	Chrom	Less than	23	22,23,24
11	Tab 3	Filter 31	Allele Number	Greater	23686	

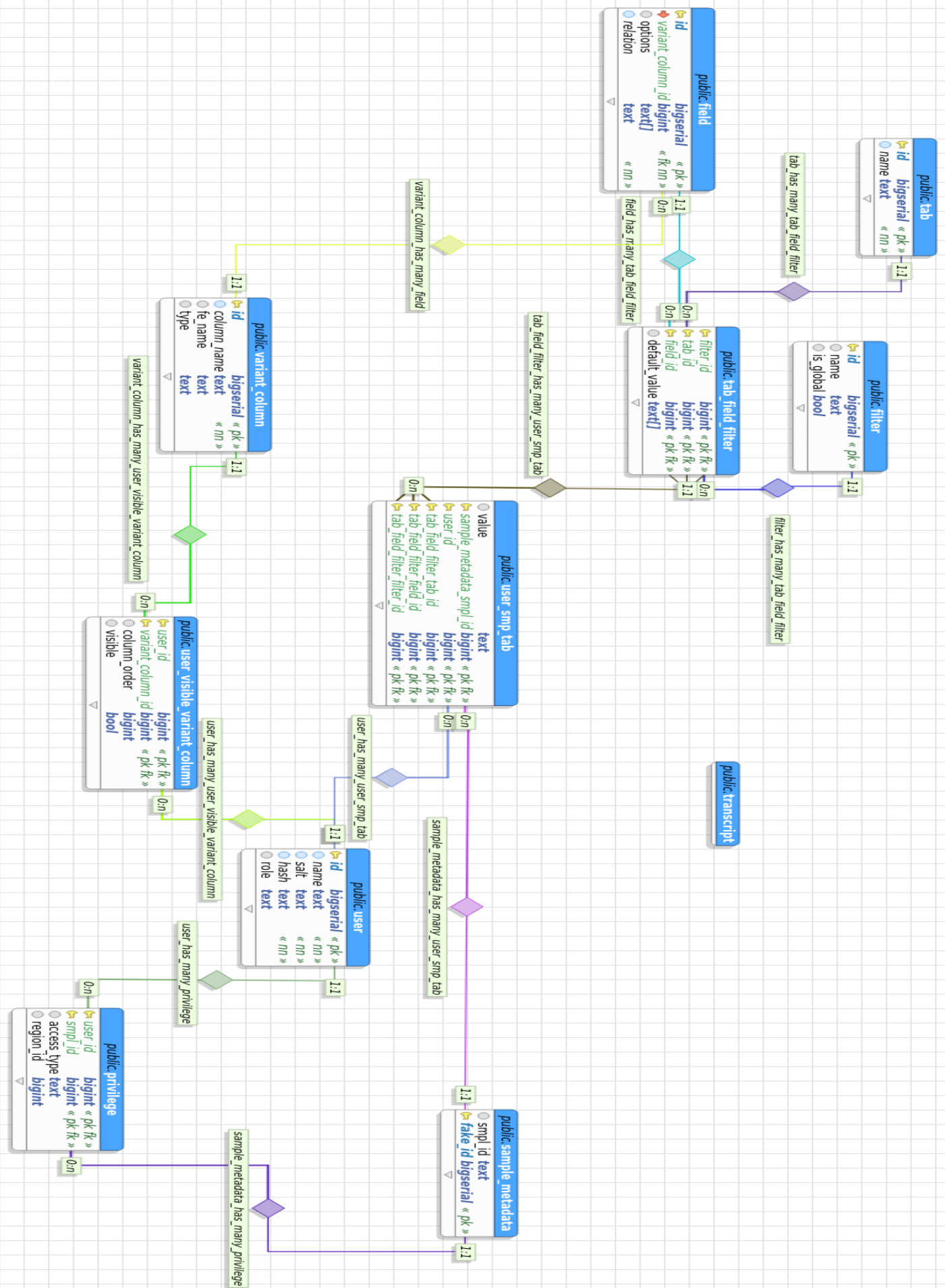
Rysunek 13. Przykładowa zawartość pliku konfiguracyjnego dla komponentu filtrującego

Zarządzanie widocznością próbek dla użytkowników Po wybraniu użytkownika z listy widocznej na Rysunku 12, administratorowi ukazuje się lista wszystkich dostępnych próbek (Rysunek 14). Administrator zaznacza w przyciskach wyboru, które próbki będą dostępne do wglądu wybranemu użytkownikowi. By zatwierdzić wybór, należy kliknąć przycisk z etykietą „Save”. Po zarejestrowaniu się do aplikacji, użytkownik nie jest przypisany do żadnej próbki.

Home Transcript list Admin	
Sample id	Visible
sample1	<input checked="" type="checkbox"/>
sample2	<input checked="" type="checkbox"/>
sample3	<input checked="" type="checkbox"/>
sample4	<input type="checkbox"/>
sample5	<input type="checkbox"/>
sample6	<input type="checkbox"/>
sample7	<input type="checkbox"/>
sample8	<input type="checkbox"/>
<div>Save</div>	

Rysunek 14. Panel administratora, konfigurujący listę udostępnionych użytkownikowi próbek

7 Schemat bazy danych



Rysunek 15. Schemat bazy danych

Przed rozpoczęciem implementacji aplikacji przeglądarkowej zaprojektowano strukturę relacyjnej bazy danych. Na Rysunku 15 przedstawiono schemat stworzony w narzędziu pgModeler [27]. Dzięki temu darmowemu narzędziu wygenerowany został skrypt tworzący bazę danych.

Dane o użytkowniku Tabela „user” zawiera rekordy o użytkownikach. Są to:

- 1) nazwy, jakie wybrali przy rejestracji i jakimi posługują się w aplikacji („name”),
- 2) rolę, jaką przydzielili im administratorzy systemu („role”),
- 3) „sól” użytą do wygenerowania skrótu i potrzebną do autoryzacji („salt”),
- 4) wynik funkcji skrótu z ich hasła („hash”),

Dane z sekwencjonowania DNA Podczas trwania okresu implementacji systemu, przykładowe dane z serwisu ExAC zostały załadowane do oddzielnej tabeli bazodanowej o nazwie „transcript”. Mimo iż owa tabela znajdowała się w tej samej bazie danych co dane o filtrach czy użytkownikach, aplikacja łączyła się oddzielnym połączeniem. Dodatkowo do pobierania danych wykorzystywała interfejs JDBC, tak by, jak najlepiej symulować produkcyjne połączenie aplikacji, czyli połączenie do rozproszonej bazy danych.

Sztuczne identyfikatory próbek Do implementacji strony prezentującej dane użytkownikowi (podrozdział 6.2) użyto funkcjonalności platformy AngularJS, umożliwiającej przekazywanie parametrów poprzez część adresu URL. Rozróżnianie próbek odbywa się względem ich nazw, które są dowolnymi ciągami znaków. Biorąc pod uwagę konieczność zabezpieczenia aplikacji przed znakami specjalnymi, które mogłyby się pojawić w nazwie oraz potrzebę użycia identyfikatora próbki jako klucza obcego w części tabel systemu, postanowiono dodać dodatkową liczbową kolumnę („fake_id”), której wykorzystanie ułatwiło implementację systemu.

Dostęp do próbek W tabeli „privilege” przechowywane są informacje o dostępie użytkownika do danej próbki. Brak rekordu wiążącego użytkownika z daną próbką jest w aplikacji rozumiany jako brak dostępu do danej próbki, przez co po zarejestrowaniu się użytkownika, nie ma On dostępu do żadnej próbki.

Lista kolumn System był projektowany z myślą, by ograniczyć w przyszłości konieczność zmian w kodzie aplikacji i by był konfigurowalny z jednego pliku dostępnego administratorowi. Z tym przeświadczeniem została dodana tabela „variant_column”, która będzie zawierać informacje o kolumnach tabeli z danymi DNA. Poza nazwami tych kolumn, które będą wykorzystywane do generowania zapytań pobierających dane, w tej encji znajdują się nazwy wyświetlające się użytkownikowi aplikacji („fe_name”) oraz informacje o typach tych kolumn, służące do sprawdzania poprawności danych wejściowych od klientów.

Widoczność kolumn Za składowanie informacji o widocznych kolumnach na stronie prezentującej użytkownikowi dane o sekwencjonowaniu, odpowiada tabela „user_visible_variant_column”. Zapisane w niej są:

- 1) identyfikator użytkownika ("user_id"),
- 2) identyfikator kolumny wariantu ("variant_column_id"),
- 3) dana typu boolowskiego ("visible") informująca o widoczności kolumny w następujący sposób:
 - prawda — kolumna jest widoczna,
 - fałsz — kolumna nie jest widoczna,

Przechowywanie schematu filtrów W trakcie analizy wymagań i projektowaniu systemu wydzielono trzy oddzielne obiekty tworzące moduł filtrujące dane, to znaczy panele, filtry i pola. Ułatwiło to zaprojektowanie odpowiednich encji bazodanowych.

Tabela odpowiadająca panelowi („tab”) poza identyfikatorem („id”) przechowuje również unikalną nazwę („name”) widoczną w górnej części panelu filtrującego. Podobnie skonstruowana jest encja odpowiadająca jednemu filtrowi („filter”) z tą różnicą, iż nazwa („name”) nie jest unikalna.

Encja będąca odpowiednikiem jednego pola („field”) również została dodana do schematu bazy danych, z następującymi kolumnami:

- 1) identyfikator („id”),
- 2) identyfikator kolumny wariantu („variant_column_id”), powiązanej z polem,
- 3) relacja między polem a kolumną wariantu,
- 4) opcjonalne wartości ukazywane użytkownikowi pod listą rozwijaną,

Za łączenie wszystkich trzech wymienionych encji, odpowiada tabela „tab_field_filter”, posiadająca identyfikatory każdej z encji oraz domyślne wartości zadeklarowane przez administratora.

Zapisywanie wartości filtrów dla użytkownika Ostatnią zaprojektowaną tabelą jest tabela „user_smp_tab”, Funkcjonalność zapisania wartości dla każdego pola filtru, oddzielnie dla każdej próbki jest zrealizowana poprzez przechowywanie identyfikatora panelu, filtru i pola elementu filtracji, razem z identyfikatorem próbki i wybraną wartością.

8 Wydajność aplikacji

Dane używane podczas implementacji do sprawdzenia działania aplikacji pochodziły z Instytutu Broad. Do testów wybrano transkrypt ENST00000407236 [12], którego dane były dostępne w pliku CSV (wyeksportowane z bazy danych ExAC). Rozmiar danych wynosił 530 wierszy, każdy posiadający 37 atrybutów. Plik został załadowany do lokalnej bazy danych, do tabeli „transcript”. Należy zaznaczyć, iż spodziewana liczba wierszy dla jednej próbki wynosi około 20000. W celu sprawdzenia zachowania się aplikacji dla większych rozmiarów danych, postanowiono kilkakrotnie zmnożyć liczbę wierszy i zapisać je jako oddzielne próbki.

Zaobserwowano dzięki temu brak płynności przy przewijaniu tabeli, tak zwane „zacinanie się” strony przeglądarki oraz nieakceptowalne wydłużenie się czasu oczekiwania na wyrysowanie się wszystkich elementów. W celu znalezienia przyczyny i poprawienia sprawności wytypowano następujące elementy działania aplikacji podejrzewane o duże narzuty czasowe:

- rozmiar przesyłanych danych,
- czas odpowiedzi na żądanie HTTP,
- czas generowania modelu w części klienckiej,
- czas dodawania elementów do DOM-u przeglądarki przez platformę AngularJS,

Środowisko testowe stanowił komputer typu laptop z procesorem firmy Intel model i3-3110M wraz z 8 GB pamięci operacyjnej. Wirtualna maszyna Javy, z której korzystała aplikacja, posiadała sterę o wielkości 512 MB z możliwością dynamicznego zwiększenia do maksymalnie 2 GB. Przeglądarkami, na których testowano aplikację była przeglądarka Google Chrome w wersji 60.0.3112.113 oraz Mozilla Firefox w wersji 55.0.2. Każdy przypadek był mierzony pięć razy.

Rozmiar przesyłanych danych Rozmiar przesyłanych danych został jako pierwszy poddany analizie i ewentualnej optymalizacji, ponieważ może znacząco wpływać na pozostałe obiekty badań.

W Tabeli 2 przedstawiono rozmiar odpowiedzi HTTP względem rozmiaru wybranej próbki DNA.

		1920 wierszy	20224 wierszy	51200 wierszy
rozmiar HTTP	odpowiedzi	3.5MB	37.3MB	93.6MB

Tabela 2. Rozmiar odpowiedzi HTTP względem rozmiaru wybranej próbki DNA

Dane wysyłane z serwera były to dane tekstowe w formacie JSON. Wysyłanymi model danych stanowiła lista obiektów DataRowDTO, który odpowiadał jednemu wierszowi tabeli i gdzie każdy z nich zawierał listę obiektów DataCellDTO (pole o nazwie „row”). Zawartość tych obiektów stanowiła nazwa odpowiadającej kolumny wiersza („columnName”) oraz wartość („value”).

Dla zmniejszenia długości przesyłanego ciągu znaków zmieniono logikę po stronie JavaScript’owej tak by można było wysłać liczbowy identyfikator kolumny zamiast jej nazwy oraz zmieniono nazwy pól obiektów w następujący sposób:

- row -> r
- columnName -> id -> i
- value -> v

Zmiany rozmiaru wysyłanych danych prezentuje Tabela 3.

		1920 wierszy	20224 wierszy	51200 wierszy
rozmiar HTTP	odpowiedzi	1.3MB	13.8MB	34.2MB

Tabela 3. Rozmiar odpowiedzi HTTP względem rozmiaru wybranej próbki DNA, po optymalizacji przesyłanego modelu

Czas trwania żądania HTTP Poniżej w Tabeli 4 przedstawiany jest średni czas trwania całego żądania HTTP.

	1920 wierszy	20224 wierszy	51200 wierszy
czas trwania żądania HTTP	5.66s	9.54s	19.21s
czas przetwarzania danych na serwerze	4.85s	7.88s	17.37s

Tabela 4. Czasy trwania żądania HTTP oraz czasy przetwarzania danych pobranych z bazy danych (w tym czas połączenia z bazą danych) względem rozmiaru danych.

Warto zwrócić uwagę jak dużą częścią całego żądania stanowił czas przetwarzania danych. Oprócz pobrania danych z bazy danych, serwer również przygotowywał powstałe obiekty do wysłania. Przekształcał je do formatu JSON, przed wysłaniem klientowi. Jak łatwo się domyślić, lokalna baza danych oraz przede wszystkim lokalny serwer będzie miał zdecydowanie lepsze wyniki od produkcyjnego serwera i należało zająć się ograniczeniem czasu pracy serwera aplikacyjnego. W tym celu wykorzystano mechanizm pamięci podręcznej (z ang. cache). Oprogramowanie, z którego skorzystano, nosi nazwę Ehcache [16] i zostało wybrane, ponieważ jest wspierane przez oprogramowanie Play i jednocześnie jest najczęściej wykorzystywaną implementacją pamięci podręcznej dla aplikacji w języku Java.

Tabela 5 przedstawia średni czas trwania całego żądania HTTP oraz czas przetwarzania danych pobranych z bazy danych (w tym czas połączenia z bazą danych) względem rozmiaru danych po zaimplementowaniu mechanizmu pamięci podręcznej.

	1920 wierszy	20224 wierszy	51200 wierszy
czas trwania żądania HTTP	0.61s	2.55s	4.57s
czas przetwarzania danych	0.14s	0.12s	0.16s

Tabela 5. Czasy trwania żądania HTTP oraz czasy przetwarzania danych pobranych z bazy danych (w tym czas połączenia z bazą danych) względem rozmiaru danych po zaimplementowaniu mechanizmu pamięci podręcznej.

Czas generowania modelu w części klienckiej Pobrany obiekt typu JSON po zakończeniu zapytania HTTP był przekształcany do innej formy między innymi funkcje biblioteki Lodash. Poniżej w Tabeli 6 przedstawione są pomiary czasu przekształceń względem rozmiaru obiektu.

	1.3MB	13.8MB	34.2MB
przekształcenia modelu w przeglądarce	0.11s	0.29s	0.91s

Tabela 6. Czasy przekształceń w przeglądarce klienckiej względem rozmiaru obiektu

Osiągnięte wyniki testów uznano za zadowalające i nie postanowiono próbować optymalizować czasu wykonania owych operacji.

Czas dodawania elementów do DOM-u przeglądarki przez platformę AngularJS Doświadczenie z oprogramowaniem AngularJS wskazywało, iż ten element może być odpowiedzialny za główny narzut czasowy przy generowanie się tabeli. Zdecydowano się wprowadzić element stronicujący prezentowane dane oraz zmierzyć czas potrzebny na wygenerowanie się elementów zależnie od ilości wizualnie dostępnych wierszy. W Tabeli 7 przedstawione są wyniki.

	1920 wierszy	20224 wierszy	51200 wierszy
300 wierszy	3.81s	4.33s	5.78s
1000 wierszy	11.82s	13.05s	16.07s
5000 wierszy	35.4s	~4min	>5min

Tabela 7. Czasy dodawania elementów do DOM-u HTML przez platformę AngularJS

Powyższe wyniki najlepiej prezentują skalę problemu. Ostatecznie zaimplementowano stronicowanie z 300 wierszami, duży wpływ na tę decyzję miał również fakt, iż przy 1000 widocznych wierszach straty wydajności i brak płynności przy przewijaniu suwakami widocznej tabelki uniemożliwiałyby efektywną pracę.

Po zaimplementowaniu funkcjonalności ukrywania kolumn ponowiono testy przy ukrytych 17 kolumnach. AngularJS zapewnia dwie dyrektywy, które można było zastosować przy tej funkcjonalności. Są to:

- ng-show bądź ng-hide — które ukrywają dany element poprzez style CSS
- ng-if — która nie dodaje elementu do DOM-u przeglądarki

Poniższa Tabela 8 ilustruje wyniki uzyskane z dyrektywą ng-if przy ukrytych 17 kolumnach.

	1920 wierszy	20224 wierszy	51200 wierszy
300 wierszy	3.73s	4.74s	5.1s
1000 wierszy	10.86s	15.69s	18.93s
5000 wierszy	19.61s	~3min	>5min

Tabela 8. Czasy dodawania elementów do DOM-u HTML przez platformę AngularJS

Porównanie ostatecznych wyników (całkowity czas do wyświetlenia użytkownikowi) po wprowadzeniu elementów optymalizacyjnych przy wprowadzeniu paginacji do każdego badania, ilustruje poniższa tabela.

	1920 wierszy	20224 wierszy	51200 wierszy
przed optymalizacją	16.45s	24.22s	50s
po optymalizacji (pierwsze pobranie)	11.26s	14.33s	33.38s
po optymalizacji (cache)	6.6s	9.2s	18.74s

Tabela 9. Czasy wyrysowanie 300 wierszy względem ilości wierszy przed i po optymalizacji

9 Bezpieczeństwo aplikacji

Badania przeprowadzone przez firmę Norton by Symantec [24] pokazują jak dużym zjawiskiem, jest współdzielenie przez ludzi haseł między różnymi serwisami, czy to skrzynką poczty elektronicznej, serwisem społecznościowym czy też kontem bankowym. Użytkownicy ufają twórcom aplikacji, iż dołożą wszelkich starań w celu zabezpieczenia ich danych.

Do zapewnienia bezpieczeństwa haseł zostało zaimplementowane, co następuje:

- połączenia klienta z serwerem — aplikacja została przygotowana do obsługi protokołu HTTPS, zapewniającym szyfrowane połączenie i ochronę danych autoryzacyjnych
- przechowywanie hasła — baza danych przechowuje hasło przekształcone za pomocą funkcji mieszającej SHA-512, dla zwiększenia bezpieczeństwa wykorzystuje się wygenerowaną „sól”

Opis implementacji tych funkcjonalności zostanie opisany w następnym rozdziale.

SQL injection Z kolei opublikowany w 2017 roku raport OWASP [25] informuje, iż najwięcej ataków na aplikacje internetowe odbywa się poprzez tak zwane SQL injection (z ang.). Opracowany system generował dynamicznie zapytania SQL przez co, mógł być narażony na te niebezpieczeństwa i został odpowiednio zabezpieczony. Przed tymi zagrożeniami czyhającymi na relacyjną bazę danych PostgreSQL chroni aplikację oprogramowanie Slick, które dynamicznie sprawdza typowanie i upewniając się, iż wprowadzone parametry zostaną wcześniej przekształcone na odpowiednie typy. Baza danych z danymi z sekwencjonowania DNA łączy się z serwerem za pomocą interfejsu JDBC, który udostępnia tę samą funkcjonalność poprzez wykorzystanie klasy PreparedStatement [26].

10 Częściowy opis implementacji

10.1 Struktura folderów

Folder o nazwie *src* zawiera pliki źródłowe projektu. Platforma Play opiera na strategii Konwencji ponad Konfigurację oraz na architekturze Model-Widok-Kontroler, dlatego po stworzeniu nowego projektu, otrzymujemy podstawowy kontroler i widok wraz z kilkoma plikami konfiguracyjnymi. W tej pracy postanowiono wykorzystać rady twórców narzędzia Play odnośnie do ułożenia struktury plików, których opis wraz z opisem niektórych plików znajduje się poniżej.

10.1.1 Pliki konfiguracyjne

Zbudowaniem całego projektu zajmuje się narzędzie Scala Build Tool [29]. Zależności wymagane do skompilowania i uruchomienia projektu umieszczono w pliku *build.sbt* oraz w plikach folderu *project*.

Na tym samym poziomie znajduje plik arkusza kalkulacyjnego — *input_file.xlsx*, w którym podano plik konfiguracyjny paneli filtrów, z przykładowymi wartościami. Pozostałe pliki odpowiedzialne za konfigurację aplikacji, znajdują się w folderze *conf*. Znajdziemy w nim:

- 1) *application.conf* — przechowujący między innymi:
 - informacje o adresie bazy danych,
 - nazwę i hasło użytkownika bazodanowego,
 - nazwę tabeli z danymi z sekwencjonowania DNA,
 - ustawienia ciasteczek,
- 2) *generated.keystore* — wygenerowany klucz do szyfrowania połączenia,
- 3) *routes* — odwzorowanie adresu URL z żądania HTTP, na odpowiedni kontroler i jego metodę

W tej sekcji należy również wspomnieć o folderze *lib*, zawierającym sterownik JDBC do oprogramowania PostgreSQL.

10.1.2 Pliki napisane w języku Scala

Do folderu *app* trafiły pliki z rozszerzeniem *.scala* oraz pliki odpowiedzialne za generowanie widoku przesyłanego jako plik HTML do przeglądarek klientów aplikacji.

Folder *controllers* zawiera w sobie kontrolery aplikacji. W zaprojektowanym systemie stworzono następujące kontrolery:

- 1) *Admin* — odpowiadający za akcje administratorów,
- 2) *Application* — odpowiadający za akcje użytkowników,
- 3) *Authorization* — odpowiadający za autoryzację użytkowników,
- 4) *Upload* — odpowiadający za załadowanie nowego pliku z definicją panelu filtrującego,

Dalej, w folderze *models* znalazły się odwzorowania tabel bazodanowych, w postaci klas języka Scala a w katalogu *repository* umieszczone zostały definicje repozytoriów — obiektów, które pośredniczą między warstwą łączącą się z bazą danych a warstwą aplikacji wykorzystującą odwzorowane już struktury z bazy danych. Repozytoria zajmują się przede wszystkim odwzorowywaniem tabel bazodanowych na modele z folderu *models* ale również filtracją ich, łączeniem czy też usuwaniem bądź edycją.

W katalogu *views* umieszczone zostały widoki aplikacji. Warto przypomnieć wykorzystaniu narzędzia AngularJS i podejścia jednej strony aplikacji. Pliki odpowiedzialne za zmianę strony w przeglądarce internetowej są umieszczone w oddzielnym folderze i będą opisane w następnej sekcji.

Katalog *utils* zawiera pliki z wszelkimi narzędziowymi klasami. Znajdziemy tam folder *dtos* z definicjami obiektów, które są przesyłane między serwerem a klientem. Metody wykorzystywane podczas autoryzacji użytkowników zostały umieszczone w pliku *Secured* w podfolderze *security* a w *services* znajdziemy serwisy między innymi odpowiedzialne za zapis do pamięci podręcznej czy też do pobierania wartości zapisanych w plikach konfiguracyjnych.

10.1.3 Pliki części klienckiej

Pliki HTML, JavaScript oraz kaskadowych arkuszy stylów umiejscowiono w folderze *public*. Katalog *javascripts* zawiera w sobie biblioteki, wykorzystywane przez część kliencką oraz aplik *app.js*, w którym łączone są wszystkie zależności obiektów platformy AngularJS oraz konfiguracja powstałego programu. Dla ułatwienia pracy i przejrzystości projektu starano się wydzielić każdy element budujący stronę. Każdy taki komponent został umieszczony w podfolderze katalogu *modules* razem ze swoim kontrolerem i plikiem HTML odpowiedzialnym za jego strukturę.

Wszystkie funkcje odpowiedzialne za generowanie żądań HTTPS aplikacji wraz z tymi odpowiedzialnymi za zapis do pamięci podręcznej przeglądarki zostały umieszczone w katalogu *services*, a elementy narzędziowe w *utils*.

10.2 Opis implementacji autoryzacji i sesji

Zabezpieczenia aplikacji zostały zaimplementowane po obu stronach aplikacji. Obie części wykorzystują mechanizm sesji do rozróżnienia użytkowników i do sprawdzenia ich uprawnień.

Część przeglądarkowa Przed przedstawieniem działania autoryzacji, należy opisać sposób działania sesji w narzędziu Play. Jest to wymagane, ponieważ sesja jest ściśle powiązana z zabezpieczeniami. Play do rozróżniania podłączonych klientów wykorzystuje klucz sesji zapisany do ciasteczka HTTP pod domyślną nazwą „PLAY_SESSION”. Wartością owego klucza jest wygenerowany dla każdego użytkownika pseudolosowy ciąg znaków. Przykład widoczny jest na Rysunku 16.

Name	Value
Request Cookies	
PLAY_SESSION	40df53b062cb84f0062533c8b0d9bb8b60cad876-username=tkogowsk
authenticated	true
Response Cookies	
authenticated	true
role	admin

Rysunek 16. Zrzut z aplikacji Chrome DevTools listy ciasteczek używanych w aplikacji

Platforma Play umożliwia dodawanie dodatkowych ciasteczek do odpowiedzi serwera i ta funkcjonalność posłużyła do przekazywania informacji o roli użytkownika przy każdym zapytaniu HTTP wysyłanym po logowaniu.

Program napisany w języku JavaScript zajmuje się zabezpieczeniem dostępu do funkcjonalności systemu zależnie od roli użytkownika otrzymanej poprzez ciasteczko. Każda próba dostania się, na którąkolwiek stronę aplikacji przez niezalogowanego użytkownika skutkuje przekierowaniem do widoku logowania. Dzieje się tak dzięki sprawdzaniu, czy użytkownik jest zalogowany przy tworzeniu każdego kontrolera AngularJS'owego i przed wykonaniem jakichkolwiek dodatkowych połączeń z serwerem.

Dzięki zastosowaniu mechanizmu sesji aplikacja przeglądarkowa nie odstępuje niczym od standardów dzisiejszych rozwiązań. Użytkownik ma możliwość odświeżania strony czy też wykorzystania przycisku „Wstecz” przeglądarki bez potrzeby ponownej autoryzacji.

Część serwera Metody serwera udostępnione klientowi do autoryzacji znajdują się w klasie kontrolera *Authorization*. Funkcja *register* sprawdza dostępność przesłanej nazwy użytkownika, a następnie wywołuje metodę z obiektu klasy *UserRepository*. Następuje wygenerowanie „soli” i za pomocą implementacji funkcji mieszającej SHA-512 uzyskujemy skrót nieodwracalny, który zapisujemy do bazy danych wraz z „solą”.

By uzyskać dostęp do pozostałych funkcjonalności, użytkownik musi się zalogować. Umożliwia to metoda *login*, która rozpoczyna ponowne wywołanie funkcji mieszającej dla przesłanego hasła oraz zapisanej w bazie danych „soli”. Pozytywny wynik logowania skutkuje w przesłaniu do przeglądarki odpowiedzi razem z ciasteczkiem *authenticated*. Ciasteczko to ma ustawiony czas życia na pół godziny i jest ponownie ustawiane na ten czas przy każdym odwołaniu do serwera.

Nierozważnym byłoby zajmowanie się zabezpieczeniem akcji tylko poprzez przesyłanie dodatkowego ciasteczka. Ataki hakerów mające na celu uzyskanie informacji o tożsamości i haseł klientów różnorodnych aplikacji są na porządku dziennym a zmiana wartości ciasteczka, jest bardzo łatwa.

Za sprawdzenie, czy dany użytkownik ma uprawnienia pozwalające na dostęp do żądanej funkcjonalności, odpowiada cecha (ang. trait) *Secured*. Klasy kontrolerów, które wykorzystują ową cechę, mogą każdą swoją akcję poprzedzić odpowiednim sprawdzeniem uprawnień dostępu za pomocą następujących metod:

- *withAuth* — pozwala na dostęp przy poprawnym kluczu sesji,
- *withUser* — pozwala na dostęp przy poprawnym kluczu sesji i roli użytkownika połączonej z sesją,
- *withAdmin* — pozwala na dostęp przy poprawnym kluczu sesji i roli administratora połączonej z sesją,

Dzięki tej technice, dodawanie nowych ról, przez co zabezpieczanie kolejnych funkcjonalności ogranicza się do dodania jednej metody.

By uzyskać nieuprawniony dostęp do metody zabezpieczonej cechą *Secured*, należy posiadać klucz sesji zalogowanego w tym samym momencie użytkownika z odpowiednią rolą.

Najprostszą drogą do uzyskania tych danych jest podsłuchanie wysyłanych żądań HTTP. Utrudnić to ma za zadanie wprowadzenie HTTPS i zablokowanie wysyłania ciasteczek, gdy serwer wykryje połączenie HTTP.

11 Wnioski i podsumowania

Zaimplementowany system spełnia założenia projektu — aplikacja umożliwia dostęp do próbek DNA na żądanie, pozwala na filtracje danych i zapisywanie wartości w filtrach. System umożliwia rozróżnianie różnych typów użytkowników i ograniczenie dostępu do danych bądź funkcjonalności zależnie od ich roli. Dodatkowo dba o bezpieczeństwo tożsamości wszystkich użytkowników i przechowywanie ich haseł. Co więcej, udało się stworzyć architekturę pozwalającą na łatwą rozbudowę i dodawanie nowych funkcjonalności.

Postarano się na zoptymalizowanie czasu odpowiedzi na zapytanie HTTPS, jednak prawdziwe testy, które odbędą się w Zakładzie Genetyki Medycznej Instytutu Matki i Dziecka w Warszawie odpowiedzą na pytanie, czy nie będzie potrzeby dalszych modyfikacji. Poza zmniejszeniem rozmiaru przesyłanych danych możliwe jest wykorzystanie pamięci podręcznej przeglądarki czy też bazy danych, które pojawiają się w przeglądarkach [28].

Rejestracja domeny, na której uruchomiony będzie serwer, w urzędzie certyfikacji jest najważniejszym elementem, od którego należy rozpocząć prace w następnym etapie rozwoju systemu.

W tym samym czasie można również rozpocząć pracę nad kolejnymi funkcjonalnościami aplikacji. Przykładami takich usprawnień mogłyby być:

- umożliwienie tworzenie własnych paneli z filtrami,
- eksport przefiltrowanych danych do plików,
- dodanie wykresów ilustrujących dane,

Literatura

- [1] Wikimedia Foundation, Inc. — DNA Available at: <https://en.wikipedia.org/wiki/DNA> (Accessed: 10 August 2017).
- [2] Wikimedia Foundation, Inc. — Genetic code Available at: https://en.wikipedia.org/wiki/Genetic_code (Accessed: 10 August 2017).
- [3] Irobalieva, R. N. et al. Structural diversity of supercoiled DNA. *Nat. Commun.* 6:8440 doi: 10.1038/ncomms9440 (2015) Available at: <https://www.nature.com/articles/ncomms9440> (Accessed: 5 August 2017).
- [4] Wikimedia Foundation, Inc. — Nucleotide Available at: <https://en.wikipedia.org/wiki/Nucleotide> (Accessed: 10 August 2017).
- [5] Kerstin Quandt, Kornelie Frech et al.: new fast and versatile tools for detection of consensus matches in nucleotide sequence data, *Nucleic Acids Research*, Volume 23, Issue 23, 1 January 1995, Pages 4878–4884, Available at: <https://doi.org/10.1093/nar/23.23.4878> (Accessed: 10 August 2017).
- [6] Wikimedia Foundation, Inc. — Transcription Available at: [https://en.wikipedia.org/wiki/https://en.wikipedia.org/wiki/Transcription_\(biology\)](https://en.wikipedia.org/wiki/https://en.wikipedia.org/wiki/Transcription_(biology)) (Accessed: 10 August 2017).
- [7] QIAGEN — Ingenuity Variant Analysis Available at: <https://www.qiagenbioinformatics.com/products/ingenuity-variant-analysis/> (Accessed: 10 August 2017).
- [8] École Polytechnique Fédérale — Scala documentation, Available at: <http://docs.scala-lang.org/> (Accessed: 10 August 2017).
- [9] The Apache Software Foundation — Apache Spark Available at: <https://spark.apache.org/> (Accessed: 10 August 2017).
- [10] Exac Browser Data — Exome Aggregation Consortium Available at: <http://exac.broadinstitute.org/> (Accessed: 10 August 2017).
- [11] Lek, Monkol et al.— Analysis of protein-coding genetic variation in 60,706 humans Available at:

- <http://www.nature.com/nature/journal/v536/n7616/full/nature19057.html> (Accessed: 10 August 2017).
- [12] Exac Browser Data — Exome Aggregation Consortium Available at: <http://exac.broadinstitute.org/transcript/ENST00000407236> (Accessed: 10 August 2017).
- [13] Steven N. Hart, Patrick Duffy et al.— VCF-Miner Available at: <https://github.com/Steven-N-Hart/vcf-miner> (Accessed: 10 August 2017).
- [14] Steven N. Hart, Patrick Duffy et al.— VCF-Miner: GUI-based application for mining variants and annotations stored in VCF files Available at: <https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbv051> (Accessed: 10 August 2017).
- [15] IGSF — VCF (Variant Call Format) version 4.0 Available at: <http://www.internationalgenome.org/wiki/Analysis/vcf4.0/> (Accessed: 10 August 2017).
- [16] Terracotta, Inc.— Ehcache Available at: <http://www.ehcache.org/> (Accessed: 10 August 2017).
- [17] Hostovita sp. z o.o. —Porównanie relacyjnych SZBD: SQLite, MySQL, PostgreSQL Available at: <https://hostovita.pl/blog/porownanie-relacyjnych-systemow-zarzadzania-bazami-danych-sqlite-mysql-postgresql/> (Accessed: 10 August 2017).
- [18] Lightbend, Inc — Slick documentation. Available at: <http://slick.lightbend.com/docs/> (Accessed: 10 August 2017).
- [19] The Apache Software Foundation — Apache Kudu Available at: <https://kudu.apache.org/> (Accessed: 20 August 2017)
- [20] The Apache Software Foundation — Apache Impala Available at: <https://impala.apache.org/>(Accessed: 20 August 2017)
- [21] Victor Bittorf, Marcel Kornacker et al.— Impala: A modern, open-source SQL engine for Hadoop. Available at: <http://web.eecs.umich.edu/~mozafari-fall2015/eecs584/papers/impala.pdf> (Accessed: 20 August 2017)

- [22] Ecma International — ECMAScript 2015 Available at: <https://www.ecma-international.org/ecma-262/6.0/> (Accessed: 20 August 2017)
- [23] JS Foundation — Lodash Available at: <https://lodash.com/> (Accessed: 20 August 2017)
- [24] Norton Cybersecurity Insights Report Available at: <https://us.norton.com/norton-cybersecurity-insights-report-global> (Accessed: 20 August 2017).
- [25] OWASP Top 10 Application Security Risks 2017 Available at: https://www.owasp.org/index.php/Top_10_2017-Top_10 (Accessed: 20 August 2017).
- [26] Oracle— Java™ Platform Standard Ed. 8 documentation <https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html> (Accessed: 20 August 2017).
- [27] Raphael A. Silva — pgModeler Available at: <https://pgmodeler.com.br/> (Accessed: 24 August 2017).
- [28] World Wide Web Consortium — IndexedDB Available at: <https://www.w3.org/TR/IndexedDB/> (Accessed: 24 August 2017).
- [29] Lightbend Inc. — Scala Build Tool Available at: <http://www.scala-sbt.org/> (Accessed: 24 August 2017).
- [30] Wikimedia Foundation, Inc. — Secure Hash Algorithms Available at: https://en.wikipedia.org/wiki/Secure_Hash_Algorithms (Accessed: 24 August 2017).

Opis zawartości CD

Na płycie CD znajduje się, co następuje:

- 1) katalog *src* z plikami źródłowymi projektu,
- 2) plik *dokument.pdf* z pisemną pracą dyplomową,
- 3) plik *Instrukcja uruchomienia aplikacji z przykładowymi danymi* z instrukcją instalacji systemu wraz z bazą danych,
- 4) skrypt *createDatabase.sql* tworzący tabele bazodanowe,
- 5) skrypt *inz_data.sql* tworzący tabele bazodanowe i wypełniający je przykładowymi danymi,

Wykaz rysunków i tabel

Spis rysunków

1	Zrzut ekranu z aplikacji Ingenuity Variant Analysis. Strona przedstawia listę wariantów dla różnych genów oraz moduł filtrujący	13
2	Zrzut ekranu z aplikacji ExAC Browser. Strona przedstawia listę wariantów dla wybranego genu	14
3	Zrzut ekranu z aplikacji VCF-Miner. Strona przedstawia listę wariantów dla wybranego genu oraz moduł filtrujący	15
4	Okno logowania i rejestracji	20
5	Lista dostępnych użytkownikowi próbek	21
6	Tabela z danymi z sekwencjonowania DNA	22
7	Sortowanie i filtrowanie pobranych danych po stronie przeglądarki . .	23
8	Okno z listą widocznych kolumn tabeli z przyciskami wyboru, zapisu i zamknięcia okna	24
9	Zrzut widoku paneli filtrujących	26
10	Diagram działania panelu filtrującego	28
11	Komponent paneli filtrujących po zliczeniu danych	29
12	Panel administratora z listą użytkowników	30
13	Przykładowa zawartość pliku konfiguracyjnego dla komponentu filtrującego	31
14	Panel administratora, konfigurujący listę udostępnionych użytkownikowi próbek	32
15	Schemat bazy danych	33
16	Zrzut z aplikacji Chrome DevTools listy ciasteczek używanych w aplikacji	45

Spis tabel

1	Wyniki zliczania danych dla każdego przykładowego filtru	27
2	Rozmiar odpowiedzi HTTP względem rozmiaru wybranej próbki DNA	38
3	Rozmiar odpowiedzi HTTP względem rozmiaru wybranej próbki DNA, po optymalizacji przesyłanego modelu	38

4	Czasy trwania żądania HTTP oraz czasy przetwarzania danych pobranych z bazy danych (w tym czas połączenia z bazą danych) względem rozmiaru danych.	39
5	Czasy trwania żądania HTTP oraz czasy przetwarzania danych pobranych z bazy danych (w tym czas połączenia z bazą danych) względem rozmiaru danych po zaimplementowaniu mechanizmu pamięci podręcznej.	39
6	Czasy przekształceń w przeglądarce klienckiej względem rozmiaru obiektu	40
7	Czasy dodawania elementów do DOM-u HTML przez platformę AngularJS	40
8	Czasy dodawania elementów do DOM-u HTML przez platformę AngularJS	41
9	Czasy wyrysowanie 300 wierszy względem ilości wierszy przed i po optymalizacji	41