

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**

## **Abstract**



„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

<b>1</b>	<b>Wstęp</b>	<b>6</b>
1.1	Motywacja . . . . .	6
1.2	Cel pracy . . . . .	6
<b>2</b>	<b>Podstawy teoretyczne</b>	<b>7</b>
<b>3</b>	<b>Wymagania funkcjonalne i нефункционалне</b>	<b>8</b>
3.1	Wymagania funkcjonalne . . . . .	8
3.2	Wymagania нефункционалне . . . . .	9
<b>4</b>	<b>Istniejące rozwiązania</b>	<b>10</b>
<b>5</b>	<b>Wybór technologii</b>	<b>11</b>
5.1	Język programowania Scala . . . . .	11
5.2	System zarządzania bazą danych . . . . .	11
5.2.1	MySQL . . . . .	12
5.2.2	SQLite . . . . .	12
5.2.3	PostgreSQL . . . . .	12
5.2.4	Uzasadnienie wyboru PostgreSQL . . . . .	13
5.3	Slick . . . . .	13
5.4	JDBC . . . . .	13
5.5	Aplikacja przeglądarkowa . . . . .	13
<b>6</b>	<b>Przypadki użycia</b>	<b>15</b>
6.1	Autoryzacja . . . . .	15
6.1.1	Role . . . . .	15
6.1.2	Rejestracja i logowanie użytkownika . . . . .	15
6.2	Przeglądanie danych z sekwencjonowania DNA . . . . .	16
6.3	Panel administratora . . . . .	22
<b>7</b>	<b>Schemat bazy danych</b>	<b>25</b>
<b>8</b>	<b>Opis implementacji</b>	<b>28</b>
<b>9</b>	<b>Bezpieczeństwo aplikacji</b>	<b>29</b>

<b>10 Testy oraz wydajność</b>	<b>30</b>
<b>11 Wnioski i podsumowania</b>	<b>31</b>
<b>Literatura</b>	<b>31</b>

# **1 Wstęp**

## **1.1 Motywacja**

## **1.2 Cel pracy**

## **2 Podstawy teoretyczne**

*Czy jest sens pisać o tym? 1-2 strony o tym czym jest DNA, genomu, kodony, jak zmiana w DNA może wpływać na organizm i dlaczego warto zajmować się badaniem DNA*



### 3 Wymagania funkcjonalne i нефункционаłne

Określenie funkcjonalności dostępnych w budowanej aplikacji, rozpoczęto od określenia rodzajów użytkowników, którzy mają korzystać z oprogramowania tak by jak najlepiej dostosować system do ich potrzeb i przyspieszyć ich pracę.

Pierwszą grupą docelową są lekarze, którzy będą poszukiwali możliwych chorób powiązanych z wariantem pacjenta, aby wykryć niebezpieczeństwa i móc jak najwcześniej przeciwdziałać chorobom. Na dane będą patrzeć w kontekście jednego badanego pacjenta i należy umożliwić im łatwe ich rozróżnienie genotypów.

Drugim typem są analitycy, którzy będą analizować dane i zadawać odwrotne pytania, czyli będą starać się znaleźć warianty, które mogą być odpowiedzialne za konkretną chorobę.

Inną istotną kwestią wziętą pod uwagę był typ i wielkość danych, jakie mają być wyświetlane klientom. W trakcie projektowania architektury jako dane przykładowe zostały wybrane dane dla przykładowego transkryptu dostępne w aplikacji Exac [3]. Dane te posiadały 36 kolumn i oczywistym wydało się że obie grupy użytkowników będzie interesowała tylko część informacji o genotypie i należało by umożliwić im filtrację oraz zakrywanie niepotrzebnych danych. Jedna próbka liczyła sobie więcej niż 340000 wiersze co wymogło zaproponowanie funkcjonalności umożliwiających na poprawne i intuicyjne filtrowanie danych tak by klient otrzymywał tylko interesujące go rekordy.

#### 3.1 Wymagania funkcjonalne

Po zakończeniu analizy zostały określone następujące funkcjonalności. Aplikacja:

- 1) ma wygodny, prosty interfejs użytkownika,
- 2) rejestruje użytkowników,
- 3) autoryzuje użytkowników,
- 4) umożliwia wybór próbki do analizy,

- 5) pozwala na wprowadzenie wcześniej zdefiniowanych filtrów z panelu administratora,
- 6) wyświetla dane z sekwencjonowania DNA dla konkretnej próbki,
- 7) filtruje dane po stronie serwera i wysła je klientowi,
- 8) zlicza ilość danych przy zadanych filtrach i informuje klienta o wyniku,
- 9) umożliwia zmianę wartości filtrów,
- 10) pozwala na wyłączenie z filtracji dowolnej części filtrów,
- 11) zapisuje wartości filtrów oddzielnie dla każdego użytkownika,
- 12) sortuje dane po stronie klienta,
- 13) filtruje dane po stronie klienta,
- 14) udostępnia administratorowi możliwość zmiany dostępu do próbek każdego użytkownika,
- 15) daje możliwość zakrycia na stronie aplikacji części danych,

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

### **3.2 Wymagania нефunkcjonalne**

Aplikacja:

- 1) szyfruje wysyłane dane między klientem a serwerem za pomocą protokołu HTTPS,
- 2) wykorzystuje funkcję SHA-512 do zabezpieczenia hasła użytkownika,
- 3) korzysta z "soli" przy wyliczaniu funkcji skrótu,

## 4 Istniejące rozwiązania

Exac broad institute, Exac Harvard Skupić się na tym iż systemy nie pozwalają na personalizację interfejsu dla użytkownika. Harvard udostępnia REST API nieprzyjazne użytkownikowi

**Czy dodać tu zdjęcia z tych aplikacji?**

## 5 Wybór technologii

Platforma klastrowego przetwarzania danych - Apache Spark[2], 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 [1]. 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 stworzenia 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[4]:

- 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[5] pozwalające na pełną kontrolę nad bazą danych oraz pisanie klasycznych zapytań SQL.

## 5.4 JDBC

Dopytać się o SQL engine by uzasadnić użycie czystego 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.

**Zalety aplikacji internetowych** Łatwość w dostępie do internetu i ilość urządzeń pozwalających na korzystanie z przeglądarek internetowych pozwoliły na rozwój aplikacji internetowych oraz ich rozpowszechnienie. Łatwość w rozbudowie, zarządzaniu i niskie ceny wynajmowania serwera aplikacyjnego spowodowały powstanie grupy platform programistycznych wspomagających ich budowę.

Narzędzia typu Ruby on Rails czy Spring Boot zdejmują z programisty obowiązek konfiguracji serwera HTTP od podstaw i umożliwiają rozpoczęcie pracy nad stronami aplikacji po kilku minutach.

**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 Angular** Angular jest opracowaną przez Google biblioteką wspomagającą tworzenie aplikacji przeglądarkowych na jednej stronie. Jej głównymi zaletami jest :

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

## 6 Przypadki użycia

Czy skupić się bardziej na opisie działania aplikacji czy raczej implementacji?

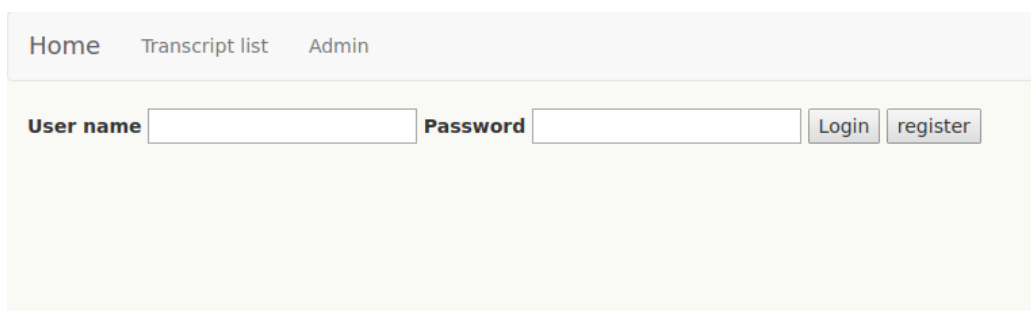
### 6.1 Autoryzacja

#### 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 1. Okno logowania i rejestracji



## 6.2 Przeglądanie danych z sekwencjonowania DNA

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

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

Home	Transcript list	Admin
<b>Sample id</b>		
id_1		
sample_id_17		
sample_id_18		
sample_id_2		

Rysunek 2. Lista próbek

**Tabela z danymi z sekwencjonowania DNA - lepsza nazwa? Czy pokazać tylko jedno zdjęcie całej strony?** Przechodzą na stronę z danymi dla konkretnej próbki, użytkownikowi prezentowane są dane w postaci tabelarycznej (Rysunek numer 3).

Chrom	Position	Allele Count European (Non-F...	Allele Number European (Non-F...	Homozygote Count European (N...	Allele Count Fl...	Allele Number Fl...	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 3. 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 największy 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 ilość pokazywanych wierszy została ograniczona do 300 i wprowadzono dodatkowy element widoczny w lewym dolnym rogu

rysunku numer 3, który umożliwia użytkownikowi poruszanie się po kolejnych stronach tabelki zmniejszając 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, umożliwiające na zawężanie zbioru danych po każdej kolumnie. Element odpowiadający za stronicowanie poprawnie zmniejsza ilość 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.

**Ustawianie widoczności kolumn** Przejrzystość danych i dostęp tylko do potrzebnych informacji jest kluczową wartością dla użytkowników. Mnogość kolumn, porowadząca do przedstawianie 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 z tabeli. Selekcja może być zapisana w bazie danych tak by przy ponownym wejściu na tą stronę aplikacji, użytkownik nie musiał kolejny raz ukrywać nieinteresujących go kolumn. Po rejestracji użytkownik ma widoczne wszystkie kolumny i musi sam je odznaczyć.

## Select visible columns

Chrom	<input checked="" type="checkbox"/>
Position	<input checked="" type="checkbox"/>
RSID	<input type="checkbox"/>
Reference	<input type="checkbox"/>
Alternate	<input type="checkbox"/>
Consequence	<input type="checkbox"/>
Protein Consequence	<input type="checkbox"/>
Transcript Consequence	<input type="checkbox"/>

Rysunek 4. Lista próbek

**Filtrowanie danych** Rysunek numer 5 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 relacja typu:

- 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 z etykietą "Inactive". Wizualnie filtr jest wyciemniony jak na przykładzie. Grupa filtrów tworzą tak zwane tabki, **zastanawia mnie użycie tu tego słowa, czy nie ma lepszej nazwy** które są oddzielnymi bytami w bazie danych, o własnych nazwach i filtrach. W górnej części panelu widoczne są przyciski z nazwami tabów. Aktywny tab wyróżnia się od zielonym kolorem od nieaktywnych o kolorze szarym.

Filtrowanie odbywa się zgodnie z kolejnością aktywnych filtrów, to znaczy najpierw filtrujemy dane używając pól pierwszego aktywnego filtru, następnie te dane filtrujemy korzystając z drugiego i tak dalej.

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 przedstawia rysunek o numerze 6. 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.

## Filters

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

**Inactive** ☒ Filter 11  
**Chrom greater than**  
  
**Position greater**

**Inactive** ☐ Filter 12  
**Reference equals**  
  
**Chrom less than**  
  
**Allele Number greater**

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

Rysunek 5. Filtry

Inactive ☐ Filter 12

**Reference equals**

C ▼

**Chrom less than**

23

**Allele Number greater**

42920

This selection reduce list length to 15360

Rysunek 6. Wynik zliczania danych

### 6.3 Panel administratora

Użytkownik o roli administratora może wejść do oddzielnej strony aplikacji (rysunek 7), gdzie może przesyłać plik o formacie xlsx o ustalonej strukturze i załadować nowe filtry. Ma też możliwość zmian widoczności próbek dla użytkowników oraz ich ról.

Home	Transcript list	Admin
Upload		
User name	Role	
user1	user	
cxz	user	
cxz2	user	

Rysunek 7. Część panelu administratora z listą użytkowników

**Zarządzanie filtrami** Zmiana struktury filtrów odbywa się z panelu administratora poprzez wysłanie z przeglądarki na serwer plik arkusza kalkulacyj-

nego o ustalonym formacie. Przykładowe dane są widoczne na rysunku 8. W pierwszej kolumnie podaje się nazwę tabki, do której przypisywany jest filtr o nazwie z drugiej kolumny. 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 zostały 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	<u>Chrom</u>	Greater than	22	
3	Tab 1	Filter 11	Position	Greater		23686, 245345, 456846
4	Tab 1	Filter 12	<u>Reference</u>	Equals	C	C,G
5	Tab 1	Filter 12	<u>Chrom</u>	Less than	23	
6	Tab 1	Filter 12	Allele Number	Greater	42920	
7	Tab 2	Filter 21	<u>Chrom</u>	Greater than	22	
8	Tab 2	Filter 21	Position	Greater		23686
9	Tab 2	Filter 22	<u>Reference</u>	Equals	G	
10	Tab 2	Filter 22	<u>Chrom</u>	Less than	23	22,23,24
11	Tab 3	Filter 31	Allele Number	Greater	23686	

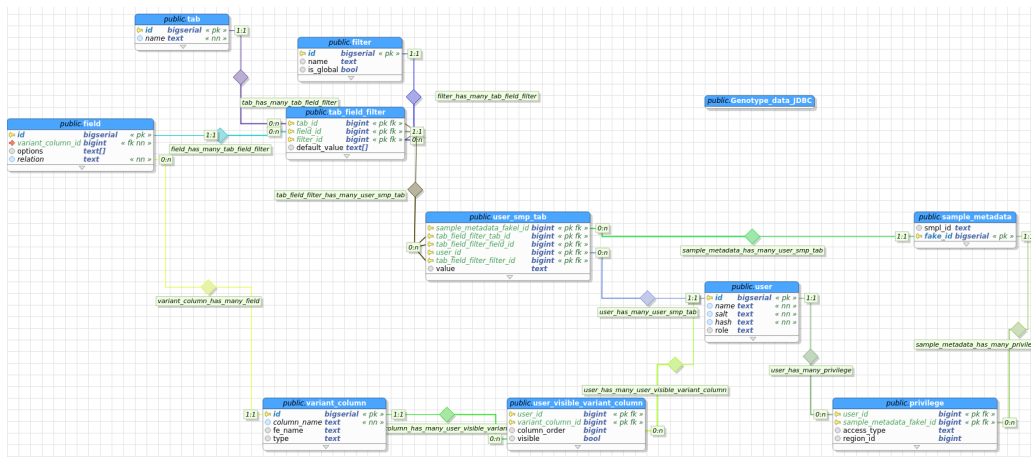
Rysunek 8. Przykładowa zawartość pliku konfiguracyjnego dla filtrów



**Zarządzanie widocznością próbek dla użytkowników** Po wybraniu użytkownika z listy widocznej na rysunku numer 7, administratorowi ukazuje się lista wszystkich dostępnych próbek (rysunek numer 9). Administrator zaznacza w przyciskach wyboru, które próbki będą dostępne do wglądu wybranemu użytkownikowi. Po zarejestrowaniu się do aplikacji, użytkownik nie jest przypisany do żadnej próbki.

Home Transcript list Admin	
Sample id	Visible
sample_id_18	<input checked="" type="checkbox"/>
id_1	<input checked="" type="checkbox"/>
sample_id_2	<input checked="" type="checkbox"/>
sample_id_17	<input type="checkbox"/>
sample_id_3	<input type="checkbox"/>
sample_id_7	<input type="checkbox"/>
sample_id_10	<input type="checkbox"/>
Save	

Rysunek 9. Panel dostępności próbek dla użytkownika



Rysunek 10. Schemat bazy danych

## 7 Schemat bazy danych

Schemat jest bardzo duży, może załączyć w częściach w dodatku i odsyłać tam czytelnika?

Przed rozpoczęciem implementacji aplikacji przeglądarkowej zaprojektowano strukturę relacyjnej bazy danych. Na rysunku nr 10 przedstawiono schemat stworzony w narzędziu pgModeler[10]. 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 "Genotype\_data\_JDBC". 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) planowano użyć funkcjonalności platformy Angular, 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.

**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

## 8 Opis implementacji

Czy w ogóle opisywać implementację? Jeśli tak to jak dokładnie? Czy nie połączyć opisu implementacji razem z przypadkami użycia? Kolejno opisując użycie aplikacji mógłbym opisać jak to się odbywa w kodzie.

## 9 Bezpieczeństwo aplikacji

Badania przeprowadzone przez firmę Norton by Symantec [7] pokazują jak dużym zjawiskiem jest współdzielenie przez ludzi haseł między różnymi serwisami, czy to skrzynką mejlową, 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

**SQL injection** Z kolei opublikowany w 2017 roku raport OWASP [8] 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 czających 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 tą samą funkcjonalność poprzez wykorzystanie klasy PreparedStatement [9].

## **10 Testy oraz wydajność**

## **11    Wnioski i podsumowania**



## Literatura

- [1] École Polytechnique Fédérale - Scala documentation, Available at: <http://docs.scala-lang.org/> (Accessed: 10 August 2017).
- [2] The Apache Software Foundation - Apache Spark Available at: <https://spark.apache.org/> (Accessed: 10 August 2017).
- [3] Exac Browser Data - Exome Aggregation Consortium Available at: <http://exac.broadinstitute.org/> (Accessed: 10 August 2017).
- [4] 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).
- [5] Lightbend, Inc Slick documentation. Available at: <http://slick.lightbend.com/docs/> (Accessed: 10 August 2017).
- [6] The Apache Software Foundation - Apache Kudu <https://kudu.apache.org/> (Accessed: 20 August 2017)
- [7] Norton Cybersecurity Insight Report Available at: <https://us.norton.com/norton-cybersecurity-insights-report-global> (Accessed: 20 August 2017).
- [8] OWASP Top 10 Application Security Risks - 2017 Available at: [https://www.owasp.org/index.php/Top\\_10\\_2017-Top\\_10](https://www.owasp.org/index.php/Top_10_2017-Top_10) (Accessed: 20 August 2017).
- [9] Oracle - Java™ Platform Standard Ed. 8 documentation <https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html> (Accessed: 20 August 2017).
- [10] Raphael A. Silva - pgModeler Available at: <https://pgmodeler.com.br/> (Accessed: 24 August 2017).

## **Wykaz rysunków i tabel**