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 do zarządzenia nr 28/2016 r.
Rektora PW

Politechnika Warszawska

„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

1.1 Motywacja

1.2 Cel pracy

2 Podstawy teoretyczne

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

3 Wymagania funkcjonalne i niefunkcjonalne

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, przyśpieszyć dostęp do danych.

Pierwszą grupą docelową są lekarze, którzy będą poszukiwali możliwych chorób powiązanych z wariantem pacjenta, by wykryć niebezpieczeństwa i móc jak nawcześniej 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 umożliwiającen 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 umożliwiająca wprowadzanie wcześniej zdefiniowanych filtrów wymusiła stworzenie trzeciej klasy użytkowników, to jest administratorów, którzy będą zarządzać dostępem do próbek dla użytkowników oraz będą wprowadzali plik zmieniający strukturę filtrów.

Parę wymagań niefukcjonalnych jak dostęp wielu użytkowników na raz, czas odpowiedzi itp.

4 Istniejące rozwiązania

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

Czy dodać tu zdjęcia z tych aplikacji?

5 Wybór technologi

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 fukncyjny 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 a 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 lini 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 zarządania 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]:

- MqSQL
- 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 decyzje 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 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, ilość urządzeń pozwalających na korzystanie z przeglądarek internetowych pozwoliły na rozwój aplikacji internetowych oraz ich rozpowrzechnienie. Łatwość w rozbudowie, zarządzaniu i niskie ceny hostowania serwera aplikacyjnego spowodo-

wały 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 przyśpieszyć pracę programisty dzięki:

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

6 Przypadki użycia

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

6.1 Autoryzacja

6.1.1 Role

6.1.2 Rejestracja użytkownika

6.1.3 Logowanie użytkownika

6.2 Przeglądanie danych z sekwencjownowania DNA

6.2.1 Widok listy dostępnych próbek

6.2.2 Ekran dostępnych genomów

6.2.3 Filtrowanie danych

6.3 Panel administratora

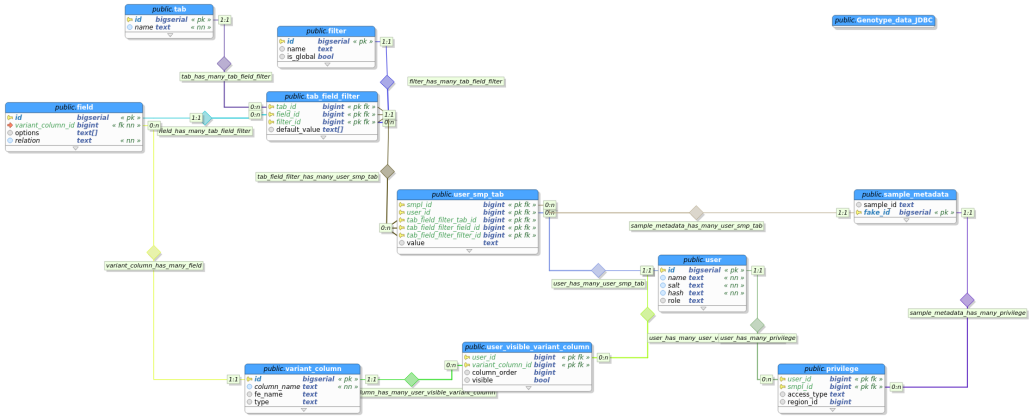
6.3.1 Zarządzanie filtrami

6.3.2 Zarządzanie rolami użytkowników

6.3.3 Zarządzanie widocznością próbek dla użytkowników

7 Schemat bazy danych

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



Rysunek 1. Schamat bazy danych

8 Opis implementacji

Czy nie połączyć opisu implementacji razem z przypadkami użycia? Kolejno opisując użycie aplikacji mógłbym opisać jak to się odbywa w kodzie.

9 Bezpieczeństwo aplikacji

9.1 Niebezpieczeństwa

9.2 Wykorzystanie protokołu https

Opisać jak od strony bardziej technicznej odbywa się zabezpieczanie haseł użytkownika (sól, sha512)

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).

Wykaz rysunków i tabel