Platforma do wspomagania adnotacji wariantów spicingowych dla danych z sekwencjonowania następnej generacji

Dokumentacja projektowa PZSP2

Wersja 0.1.1

13-11-2023

Semestr 23Z

Zespół nr 1 w składzie:
Jarczewski Marcin
Jażdzyk Jakub
Kaszyński Dawid
Szawerda Mikołaj

Mentor zespołu: mgr inż. Klara Borowa

Spis treści

1 Wprowadzenie

1.1 Cel projektu

Celem projektu jest stworzenie zaawansowanej platformy wspomagającej adnotację wariantów splicingowych w danych uzyskanych przez sekwencjonowanie następnej generacji (NGS). Platforma ta ma na celu umożliwienie efektywnego przetwarzania i analizy wariantów genetycznych, które mogą mieć potencjalne znaczenie kliniczne, szczególnie tych, które mogą zaburzać proces splicingu. System będzie zdolny do szybkiego przetwarzania wariantów typu SNV dla całego genomu oraz umożliwi przeliczenie online wariantów typu INDEL.

1.2 Wstępna wizja projektu

Projekt zakłada opracowanie **modularnej** platformy, zdolnej do skutecznego radzenia sobie z wyzwaniami związanymi z analizą ogromnych ilości danych genetycznych generowanych przez technologie NGS. Platforma ta będzie wyposażona w funkcje umożliwiające identyfikację i interpretację wariantów splicingowych, korzystając z kilku wybranych algorytmów, oraz oferującą możliwość **równoległego przetwarzania danych** online za pomocą kolejki zdarzeń. Platforma zostanie zaprojektowana w sposób umożliwiający łatwe aktualizacje i dodawanie nowych funkcji, w miarę pojawiania się nowych algorytmów i rozwoju technologii. Modułowa architektura platformy **umożliwi integrację** z zewnętrznymi, już istniejącymi procesami np. system powiadomień o końcu przetwarzania danych. Dodatkowo, platforma będzie zawierała **interfejs REST API** oraz uproszczony interfejs graficzny, co ułatwi interakcję użytkowników z systemem.

2 Metodologia wytwarzania

Organizacja pracy w projekcie

3 Analiza wymagań

3.1 Wymagania użytkownika i biznesowe

- **Wymagania Biznesowe:**
  - ➔ Cele biznesowe: System ma na celu dostarczenie narzędzia do analizy wariantów splicingowych, skierowanego do laboratoriów.
  - ➔ Problemy do rozwiązania: Brak dostępnego narzędzia do efektywnej analizy wariantów splicingowych oraz potrzeba integracji z istniejącymi pipeline'ami przetwarzania danych bioinformatycznych.
- **Wymagania Użytkowe:**
  - ➔ Potrzeby użytkowników: System musi oferować intuicyjny interfejs pozwalający na przeliczanie wariantów genetycznych z wykorzystaniem plików w formacie .vcf, lub .csv, oraz komunikację z głównym systemem przez Rest API.
  - ➔ Cechy użytkowe: Interfejs użytkownika musi być prosty i intuicyjny, a także zapewniać możliwość przesyłania danych genetycznych w odpowiednim formacie.
- **Wymagania Systemowe:**
  - ➔ Skalowalność: System musi obsłużyć duże pliki i ilości danych.

➔ Wydajność: Mechanizm cache'owania (Redis) jest niezbędny do poprawy wydajności operacji oraz szybkiego odczytywania najczęściej przetwarzanych danych.
➔ System musi być zaprojektowany i zaimplementowany w sposób umożliwiający łatwą rozbudowę o nowe funkcje. Proces rozbudowy kodu powinien być intuicyjny i nie wymagać gruntownych zmian w istniejącym kodzie.

3.2 Wymagania funkcjonalne i niefunkcjonalne

Wymagania funkcjonalne:

- Przetwarzanie i analiza wariantów splicingowych
- Wsparcie dla formatów VCF, CSV
- Integracja z istniejącymi pipeline'ami
- Modularna architektura
- Interfejs REST API
- Przechowywanie wyników PostgreSQL
- Prosty interfejs graficzny
- Obsługa wariantów genetycznych w kontekście genomu bazowego hg38
- Przetwarzanie niezależnych wariantów
- Adnotacje z istniejących algorytmów obsługiwanych przez workery
- Wstępne przeliczanie SNV dla wszystkich zasad azotowych (A,C,T,G) per algorytm i cachowanie
- Cachowanie w INDEL bez wstępnego przeliczania (za dużo możliwości)

Wymagania niefunkcjonalne:

- Skalowalność
- Odporność na błędy
- Bezpieczeństwo
- Wydajność przez cachowanie
- Asynchroniczna komunikacja
- Elastyczność projektu

3.3 Przypadki użycia

[na poziomie ogólnym i rozszerzonym

- diagramy aktywności/stanów dla skomplikowanych przypadków

- związki pomiędzy przypadkami użycia

- diagram przypadków użycia]

3.4 Potwierdzenie zgodności wymagań

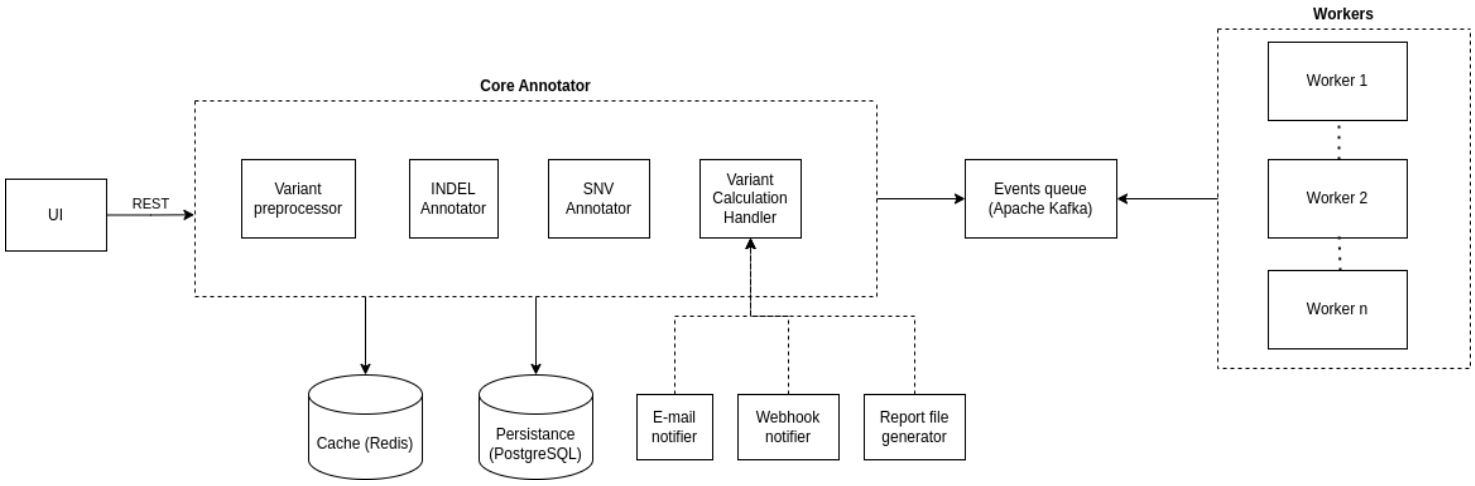
Zatwierdzam specyfikację wymagań, jako spełniających potrzeby Klienta.	…… ……. Data i podpis Właściciela tematu
Uwagi	

4 Definicja architektury

System opeira się na asynchronicznym rozdzieleniu pracy pomiędzy workery, którymi są kontenery dockerowe. To rozwiązanie jest kompatybilne z:

- dostawcami chmurowymi aws/gcp/azure
- kubernetes
- docker swarm

Dodatkowo w trakcie przetwarzania danych może być bardzo dużo, wszystkich danych SNV jest rzędu 1 miliarda lecz wysyłane do kolejki i przetwarzane w pojedynczym workerze będzie jedynie czwórka liczb wraz z genomem referencyjnym co zapewni nam niski narzut na przesyłanie danych.



System o architekturze rozproszonej składa się z następujących komponentów:

- UI

Prosty graficzny interfejs użytkownika umożliwiający przeliczanie wariantów genetycznych z wykorzystaniem przesyłania pliku w odpowiednim formacie (na przykład .vcf). Komunikuje się z głównym systemum za pomocą interfejsu REST.

- Core Annotator

Modularny system odpowiedzialny za delegację zadań przeliczania wariantów genetycznych typu SNV oraz INDEL, a także za komunikację z UI oraz istniejącymi potokami przetwarzania danych bioinformatycznych. Dla danego wariantu wejściowego, jeżeli nie został on nigdy wcześniej wyliczony to deleguje żądanie obliczenia tego wariantu z wykorzystaniem wybranego algorytmu na kolejkę (Apache Kafka).
Udostępnia możliwość integracji z zewnętrzynymi systemami przetwarzania zarówno metodą pull (możliwość sprawdzania dostępności wyników za pomocą identyfikatora żądania przeliczenia wariantów), a także push za co odpowiada moduł Variant Calculation Handler konsumujący powiadomienia od workerów o zakończonej pracy i odpowiednio jej wyniki obsługujący.

- Workers

Workery są odpowiedzialne za wykonywanie złożonych algorytmów przetwarzania danych genetycznych. Konsumują one żądania z kolejki, wyliczają rozwiązanie i dodają na kolejkę powiadomienia o skończeniu przez siebie pracy.

System opiera się na asynchronicznym rozdzieleniu pracy pomiędzy workery, którymi są kontenery dockerowe. To rozwiązanie jest kompatybilne z:

- dostawcami chmurowymi aws/gcp/azure
- kubernetes
- docker swarm

Dodatkowo w trakcie przetwarzania danych może być bardzo dużo, wszystkich danych SNV jest rzędu 1 miliarda lecz wysyłane do kolejki i przetwarzane w pojedynczym workerze będzie jedynie czwórka liczb wraz z genomem referencyjnym co zapewni nam niski narzut na przesyłanie danych.

- Events queue

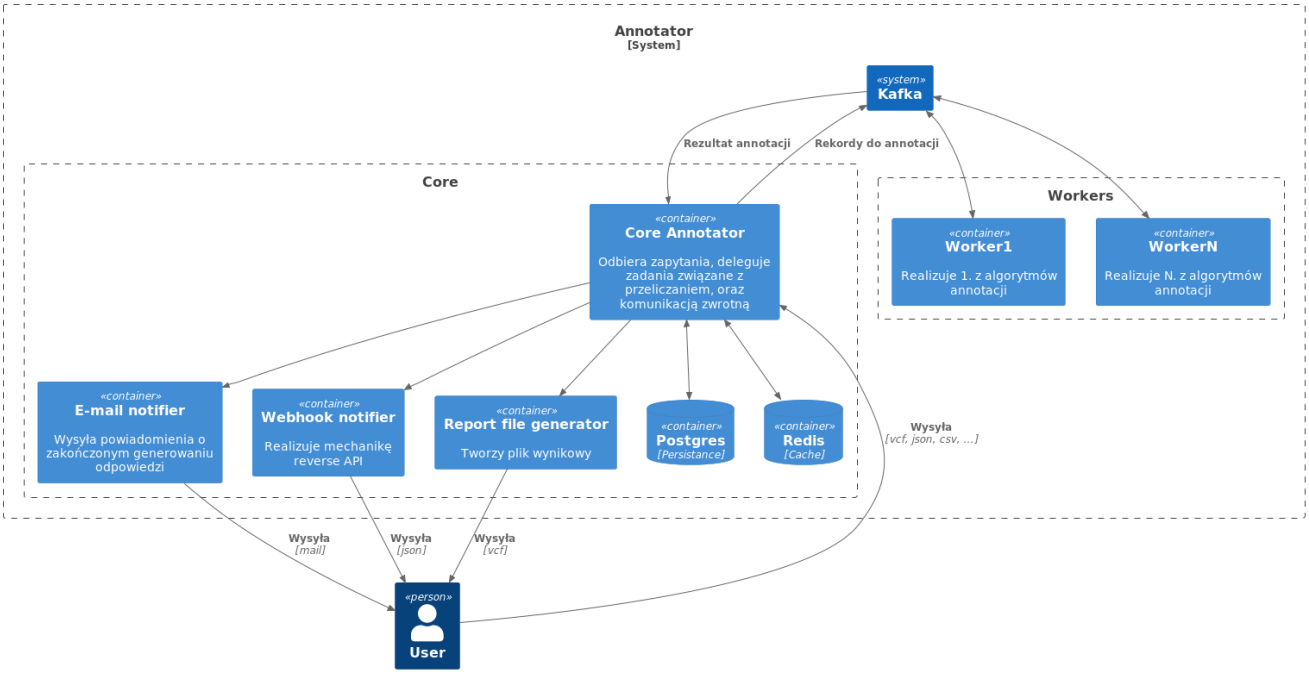
Message broker (Apache Kafka) umożliwiający zachowanie małego wzajemnego sprzężenia serwisów w systemie.

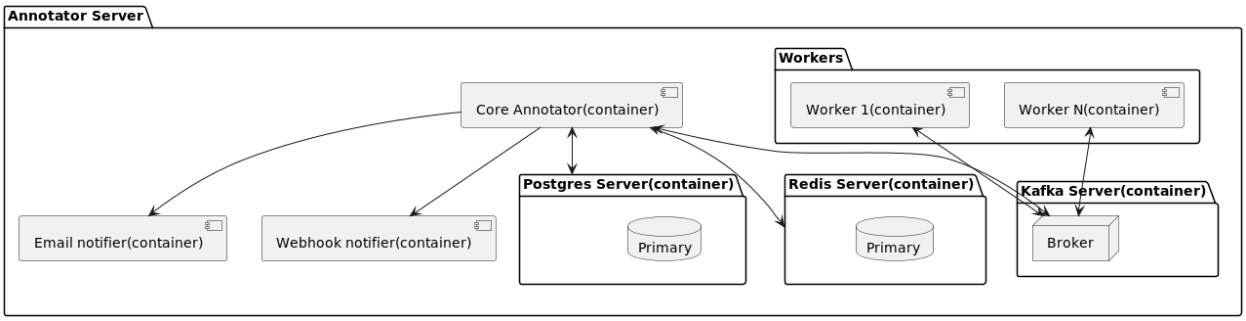
- Persitance

Baza danych (PostgreSQL) zapisująca wyniki żądań obliczeń wariantów użytkownika oraz przechowująca wszystkie dotychczas wyliczone warianty SNV i INDEL względem genomu referencyjnego.

- Cache

Pamięć podręczna (Redis) umożliwiająca szybkie odczytywanie najczęściej przewijających się wariantów.





Rozwiązanie na cele developmentu i finalnej prezentacji będzie uruchamiane z użyciem docker compose. Skonteneryzowany będzie więc główny moduł rozwiązania, poszczególne algorytm liczące adnotacje, bazy danych i serwer kafka.

W przypadku produkcyjnej realizacji rozwiązania dzięki użycia brokera wiadomości, moduł “Core Annotator” jak i poszczególne Workery mogą być dowolnie skalowane horyzontalnie.

5 Dane trwałe

5.1 Model logiczny danych

[diagramy]

5.2 Przetwarzanie i przechowywanie danych

[opis planowanego sposobu implementacji: jaka baza danych lub inne oprogramowanie, jak będzie realizowany dostęp do danych (użyte API itp.), czy planowane jest programowanie w b.d. (wyzwalacze, mechanizmy optymalizacji zapytań, itp.)]

6 Specyfikacja analityczna i projektowa

Obowiązkowo odnośnik do repozytorium kodu (jedno repozytorium na projekt, jeżeli więcej proszę uzasadnić)

Obowiązkowo określenie metod realizacji: języki programowania, frameworki, środowisko programowania/ uruchamiania/ wdrażania, środowisko ciągłej integracji]

Obowiązkowo Diagram klas lub model pojęciowy struktury informacyjnej: E-R l

Opcjonalnie model struktury systemu (diagram wdrożenia)

Opcjonalnie specyfikacja realizacji przypadków użycia: diagramy sekwencji lub współpracy

Obowiązkowo statystyki: licaba plików, linie kodu, liczba testów jednostkowych

7 Projekt standardu interfejsu użytkownika

[wykorzystanie narzędzi do modelowania oraz tworzenia makiet warstwy prezentacyjnej (np. storyboards, wireframes, wireflows, mockups, prototypes etc)]

8 Specyfikacja testów

[standardy obsługi błędów i sytuacji wyjątkowych

rodzaje testów, specyfikacja i opis sposobu realizacji poszczególnych rodzajów testów, scenariusze testowe

miary jakości testów]

9 Wirtualizacja/konteneryzacja

10 Bezpieczeństwo

11 Podręcznik użytkownika

[instrukcja użycia funkcjonalności systemu]

12 Podręcznik administratora

[- instrukcja budowy systemu z kodu źródłowego

- instrukcja instalacji i konfiguracji systemu

- instrukcja aktualizacji oprogramowania

- instrukcja zarządzania użytkownikami i uprawnieniami

- instrukcja tworzenia kopii zapasowych i odtwarzania systemu

- instrukcja zarządzania zasobami systemu]

13 Podsumowanie

[Krytyczna analiza osiągniętych wyników, mocne i słabe strony

Możliwe kierunki rozwoju]

14 Bibliografia

[Wykaz materiałów źródłowych, opis zgodny ze standardem sporządzania opisów bibliograficznych - https://bg.pw.edu.pl/index.php/przypisy-i-bibliografia]

Zatwierdzam dokumentację.	…… ……. Data i podpis Mentora