

Bazy danych 1

Dokumentacja projektu

Adrian Furman
nr albumu 303745
Informatyka Stosowana

1. Projekt koncepcji, założenia

1.1. Zdefiniowanie tematu projektu

Za temat projektu wybrany został system informatyczny dla sieci laboratoriów diagnostycznych pozwalający na wygodne zarządzanie strukturą placówek oraz zatrudnionych pracowników, wykonywanie zleconych badań w laboratoriach oraz przeglądanie raportów przedstawiających osiągnięcia finansowe laboratoriów jak i wydajność pracowników. Ponadto system umożliwia przeglądanie dostępnej oferty badań dla klientów oraz składanie zamówień. Wyniki badań mogą zostać wyświetlone przez klientów po podaniu specjalnego kodu dostępu otrzymywanego po złożeniu zamówienia.

1.2. Analiza wymagań użytkownika

Dla klientów system oferuje poniższe możliwości:

- Przeglądanie oferty aktualnie dostępnych badań z podziałem na kategorie,
- składanie zamówień na wybrane badania
- przeglądanie wyników zamówionych zestawów badań po wprowadzeniu specjalnego kodu dostępu który wyświetlany jest po złożeniu zamówienia.

Dla pracowników sieci system udostępnia następujące funkcjonalności:

- wszystkie jak dla klienta, oraz dodatkowo:
- logowanie się do serwisu za pomocą otrzymanego od administratora adresu email oraz hasła,
- przeglądanie oraz wykonywanie (poprzez wprowadzenie wartości wynikowej) badań zleconych do laboratorium, w którym dany pracownik jest zatrudniony,
- przeglądanie raportów przedstawiających min. osiągnięcia finansowe każdej z placówek,
- operowanie wszystkimi dostępnymi zasobami sieci laboratoriów takimi jak: badania, laboratoria, punkty pobraniowe, kategorie badań, stanowiska, pracownicy z możliwością dodawania nowych, usuwania czy modyfikowania wybranych obiektów.

1.3. Zaprojektowanie funkcji

Baza danych za pomocą odpowiednio zdefiniowanych funkcji oraz pozostałych mechanizmów powinna sprawdzać poprawność wprowadzanych danych, dbać o odpowiednie aktualizowanie statusów zleconych badań, zapobiegać przed usuwaniem zasobów istotnych pod względem zachowania historii wyników klientów czy operacji finansowych, umożliwiać na wygodne pobieranie złożonych zbiorów kolumn. W tym celu przewidziane zostały funkcje odpowiedzialne za następujące zadania:

- automatyczne aktualizowanie statusu zlecenia badania po wprowadzeniu odpowiadającego mu wyniku przez pracownika laboratorium,
- blokowanie próby ponownego wprowadzenia wyniku do już zrealizowanego badania (potencjalnie możliwe w przypadku dwóch pracowników, którzy z braku koordynacji w zbliżonej chwili czasu wprowadzają wynik dla identycznego zlecenia badania),
- normalizowanie oraz sprawdzanie poprawności danych wprowadzanych do systemu w szczególności tych pochodzących od osób z zewnątrz - klientów (tutaj zgodnie z wytycznymi specjalną walidacją, wykraczającą poza ograniczenia <NOT NULL>, które w większości przypadków w zupełności wystarczają, objęta została jedna tabela przechowująca dane osobowe a więc w głównej mierze rekordy pochodzące z procesu składania zamówień, uzupełniane przez klientów - potencjalnie najbardziej narażone na ataki miejsce systemu),
- pobieranie odpowiednich wyników badań wraz z meta informacjami dla podanego kodu dostępu,
- sprawdzanie czy operacja usunięcia wskazanego badania nie naruszy historycznej informacji wykonanego zlecenia - jeśli tak by się stało operacja nie jest przeprowadzana oraz zwracany jest odpowiedni komunikat (podobnie jak w przypadku walidacji zaimplementowana została tylko jedna taka funkcja w celu zaprezentowania mechanizmu, dla reszty zasobów analogiczne operacje wyglądałyby bardzo podobnie co sztucznie powiększyłoby objętość kodu a nie wniosło żadnej informacji dydaktycznej),
- opakowanie złożonych zapytań np. generujących raporty w widoki o adekwatnych nazwach w celu wygody użytkownika systemu po stronie backendu aplikacji.

2. Projekt diagramów (konceptualny)

2.1. Zdefiniowanie encji oraz ich atrybutów

W projekcie przewidziane zostały następujące obiekty wraz z ich atrybutami:

- **dane_osobowe** (id, imie, nazwisko, pesel, data urodzenia),
- **stanowisko** (id, nazwa, pensja, opis),
- **pracownik** (id, dane_osobowe_id, email, hasło, data zatrudnienia, premia),
- **adres** (id, miasto, ulica, numer budynku/lokalu),
- **laboratorium_diagnostyczne** (id, adres_id, liczba aparatów),
- **punkt_pobran** (id, adres_id, laboratorium_id),
- **badanie** (id, nazwa, dolna granica, górna granica, jednostka, cena, materiał, rodzaj),
- **zamowienie_badan** (id, dane_osobowe_id, punkt_pobran_id, data, kod dostępu),
- **zlecenie_badania** (id, zamowienie_id, laboratorium_id, badanie_id, status, koszt),
- **wynik_badania** (id, data, wartosc, zlecenie_badania_id, pracownik_id),
- **kategoria_badan** (id, nazwa, opis),
- **adresplatnosc** (id, zamowienie_id, rodzaj, data, kwota, zrealizowana).

2.2. Zaprojektowanie relacji pomiędzy encjami

Do zaprojektowania oraz zwizualizowania relacji występujących pomiędzy zdefiniowanymi obiektami wykorzystane zostało narzędzie do projektowania diagramów ERD dbdiagram.io udostępnione na [tej stronie](#). Poniżej znajduje się grafika wygenerowana za pomocą wspomnianego narzędzia:

3. Projekt logiczny

3.1. Projektowanie tabel, kluczy indeksów

W folderze *sql* w pliku *create.sql* znajdują się polecenia odpowiedzialne za definiowanie słowników oraz utworzenie struktury tabel, relacje zostały utworzone na końcu pliku za pomocą poleceń ALTER TABLE. Finalna struktura prezentuje się następująco:

→ **dane_osobowe**

- ◆ **id SERIAL - element klucza głównego - unikalne**
- ◆ **pesel VARCHAR(11) - element klucza głównego, wymagane dokładnie 11 znaków będących cyframi - unikalne**
- ◆ imie VARCHAR - wymagane
- ◆ nazwisko VARCHAR - wymagane
- ◆ data urodzenia DATE - wymagane

→ **stanowisko**

- ◆ **id SERIAL - klucz główny**
- ◆ nazwa VARCHAR - wymagane
- ◆ pensja NUMERIC - wymagane
- ◆ opis VARCHAR - opcjonalny

→ **pracownik**

- ◆ **id SERIAL - klucz główny**
- ◆ dane_osobowe_id INT - klucz obcy odwołujący się do danych osobowych pracownika, wymagane
- ◆ stanowisko_id INT - klucz obcy odwołujący się do stanowiska, które piastuje pracownik, wymagane
- ◆ laboratorium_id INT - klucz obcy odnoszący się do laboratorium, w którym zatrudniony jest pracownik, wymagane
- ◆ email VARCHAR
- ◆ haslo VARCHAR - hash stworzony na podstawie wprowadzonego przy dodawaniu pracownika hasła, wymagane
- ◆ data_zatrudnienia DATE, domyślnie = now()
- ◆ premia NUMERIC - domyślnie = 0

→ **adres**

- ◆ **id SERIAL - klucz główny**
- ◆ miasto VARCHAR(50) - wymagane
- ◆ ulica VARCHAR(50) - wymagane

- ◆ numer VARCHAR(10) - numer budynku, typ znakowy ze względu np na: "34A", wymagane

→ **laboratorium_diagnostyczne**

- ◆ **id SERIAL - klucz główny**
- ◆ adres_id INT - klucz obcy odnoszący się do adresu danego laboratorium, wymagane
- ◆ liczba_aparatow INT - liczba aparatów wykonujących badania znajdujących się w danym laboratorium, domyślnie = 10

→ **punkt_pobran**

- ◆ **id SERIAL - klucz główny**
- ◆ adres_id INT - klucz obcy odnoszący się do adresu danego punktu pobrań, wymagane
- ◆ laboratorium_id INT - klucz obcy odnoszący się do laboratorium, pod które podlega dany punkt pobrań, wymagane

→ **zamowienie_badan**

- ◆ **id SERIAL - klucz główny**
- ◆ dane_osobowe_id INT - klucz obcy odnoszący się do danych osobowych wprowadzonych podczas składania zamówienia, wymagane
- ◆ punkt_pobran_id INT - klucz obcy odwołujący się do punktu pobrań, w którym dokonano pobrania materiału niezbędnego do wykonania badań, wymagane
- ◆ data DATE - data złożenia zamówienia, domyślnie = now()
- ◆ kod_dostepu VARCHAR - unikatowy ciąg znaków alfanumerycznych wygenerowany za pomocą biblioteki uuid, zwracany jest do klienta po złożeniu zamówienia i pozwala na przyszłe wyświetlanie wyników, wymagane

→ **zlecenie_badiania**

- ◆ **id SERIAL - klucz główny**
- ◆ zamowienie_id INT - klucz obcy nawiązujący do zamówienia badań, z którego pochodzi konkretne zlecenie, wymagane
- ◆ laboratorium_id INT - klucz obcy odwołujący się do laboratorium, w którym wykonane ma zostać badanie, wprowadzany na podstawie punktu pobrań w którym pobrany został materiał do badań, wymagane
- ◆ badanie_id INT - klucz obcy odwołujący się do konkretnego badania z oferty, którego dotyczy dane zlecenie
- ◆ status status_badiania - typ słownikowy służący do rozróżniania badań oczekujących na wykonanie od już zrealizowanych
- ◆ koszt NUMERI - cena pobrana z badania w momencie składania zamówienia badań, istotny ze względu na potencjalne rozbudowanie

systemu o promocje czasowe aby zachować historyczną informację o dokładnej cenie w jakiej kupione zostało badanie

→ **badanie**

- ◆ **id SERIAL - klucz główny**
- ◆ nazwa VARCHAR(30)
- ◆ wartosc_min NUMERIC - dolna granica przedziału prawidłowych wartości, wymagane
- ◆ wartosc_max NUMERIC - górna granica przedziału prawidłowych wartości, wymagane
- ◆ jednostka VARCHAR(10) - jednostka, w której podane są liczby dotyczące tego badania, wymagane
- ◆ cena NUMERIC - aktualna cena badania w ofercie, wymagane
- ◆ material material_biologiczny
- ◆ rodzaj rodzaj_pracowni - rodzaj pracowni do której w laboratorium trafi konkretne badanie

→ **kategoria_badan**

- ◆ **id SERIAL - klucz główny**
- ◆ nazwa VARCHAR(20), wymagane
- ◆ opis VARCHAR(200), opcjonalne

→ **badanie_kategoria**

- ◆ **badanie_id INT - klucz główny**
- ◆ **kategoria_id INT - klucz główny**

→ **wynik_badania**

- ◆ **id SERIAL - klucz główny**
- ◆ data TIMESTAMP - moment wykonania badania, wymagane
- ◆ wartosc NUMERIC - wartosc wyniki badania, wymagane
- ◆ zlecenie_badania_id INT - klucz obcy odwołujący się do konkretnego zlecenia badania, do którego przypisany jest ten wynik, wymagane
- ◆ pracownik_id INT - klucz obcy odwołujący się do pracownika, który wykonał badania, wymagane

→ **platnosc**

- ◆ **id SERIAL - klucz główny**
- ◆ zamowienie_id INT - klucz obcy odwołujący się do zamówienia badań, z którego pochodzi płatność, wymagane
- ◆ rodzaj typ_platnosc - np. przelew, domyślnie = przelew
- ◆ data TIMESTAMP - data dokonania płatności, domyślnie = now()
- ◆ kwota NUMERIC - całkowity koszt zamówienia badań, wymagane

3.2. Słowniki danych

Słowniki danych zdefiniowane zostały we wspomnianym pliku *sql/create.sql*.

3.3. Analiza zależności funkcyjnych i normalizacja tabel

Projekt struktury bazy danych spełnia wymagania trzech postaci normalnych. Tabele opisują pojedyncze obiekty, wartości atrybutów są atomowe, kolejność wierszy nie ma znaczenia, żadna niekluczowa kolumna nie jest częściowo funkcyjnie zależna od klucza potencjalnego, atrybuty niekluczowe nie są funkcyjnie zależne od innych atrybutów niekluczowych. Każda tabela posiada unikalny klucz główny, który jednoznacznie identyfikuje zasób.

3.4. Denormalizacja struktury tabel

Nie wymagana.

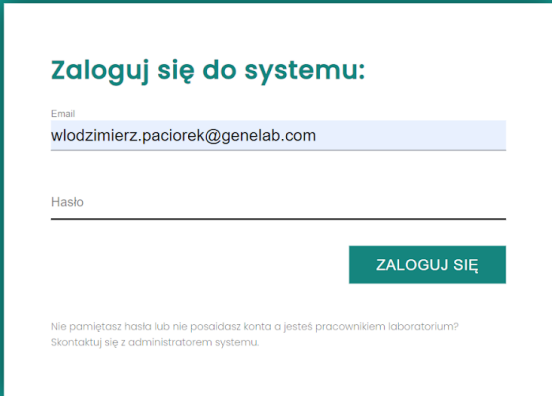
3.5. Zaprojektowanie operacji na danych

Wszystkie procedury składowane oraz definicje triggerów umieszczone zostały w pliku *sql/functions.sql*. Zdefiniowane widoki znajdują się natomiast w pliku *sql/views.sql*. Funkcje, których nazwa czy treści komunikatów błędów nie oddają w sposób wystarczający ich przeznaczenia opatrzone zostały dodatkowym komentarzem wyjaśniającym.

4. Projekt funkcjonalny

4.1. Interfejsy do prezentacji, edycji i obsługi danych

Zarządzanie poszczególnymi zasobami odbywa się poprzez formularze i listy na dedykowanych podstronach panelu administracyjnego. Aby móc zarządzać danymi należy uprzednio się zalogować:



Zaloguj się do systemu:

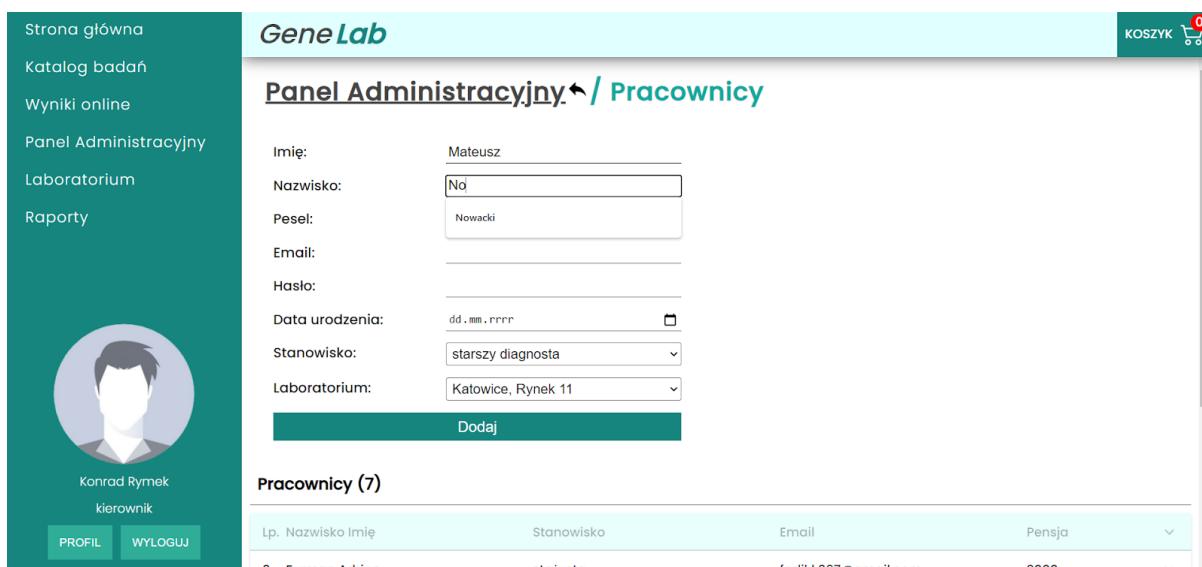
Email
włodzimierz.paciorek@genelab.com

Hasło

ZALOGUJ SIĘ

Nie pamiętasz hasła lub nie posiadasz konta a jesteś pracownikiem laboratorium?
Skontaktuj się z administratorem systemu.

Poniżej pokazana została przykładowy panel zarządzania pracownikami:



GeneLab KOSZYK 0

Panel Administracyjny / Pracownicy

Imię: Mateusz

Nazwisko:

Pesel:

Email:

Hasło:

Data urodzenia:

Stanowisko:

Laboratorium:

Dodaj

Pracownicy (7)

Lp.	Nazwisko Imię	Stanowisko	Email	Pensja	
0	Furman Adrian	stażysta	fadikk367@gmail.com	2000	▼

4.2. Wizualizacja danych

Wgląd w dane poszczególnych zasobów możliwy jest poprzez panel administracyjny. Ponadto aplikacja pozwala na wyświetlanie wyników badań online po podaniu kodu dostępu:

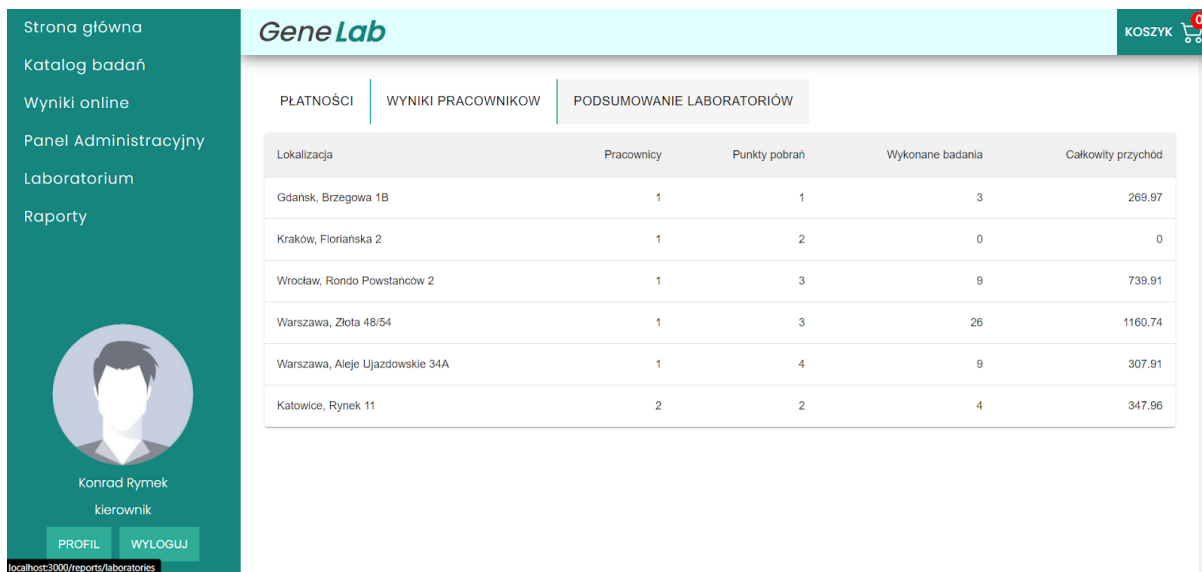
Nazwa badania	Norma	Jednostka	Wynik
Adrenalina	120 - 140	mg/cm3	121 OK
Albumina	20 - 40	uq	43 ▲
anty-TG	30 - 39	ug	33 OK
Białko całkowite	23 - 29	g/kg	24 OK
Cholesterol HDL	40 - 50	ug	43 OK
Jod, ilościowo	170 - 200	mg	187 OK

Aby móc wyświetlić wyniki badań oczywiście należy wcześniej złożyć na nie zamówienie, wybierając interesujące nas pozycje w katalogu badań:

Lp.	Nazwa	Cena
0.	anty-TG	199.99 PLN
1.	Białko całkowite	34.99 PLN
2.	Cholesterol HDL	34.99 PLN
3.	Cholesterol LDL bezpośrednio	34.99 PLN
4.	Cystatyna C	29.99 PLN
5.	Jod, ilościowo	12.99 PLN
6.	Mocznik	24.99 PLN
7.	Panel jelitowy	24.99 PLN

Dodatkowo dla zalogowanych pracowników przewidziane zostało generowanie raportów podsumowujących wyniki finansowe osiągnęte przez poszczególne

laboratoria, statystyki pracowników czy zestawienie wszystkich opłat za zamówienia. Poniżej przykładowy raport wyników finansowych laboratoriów:

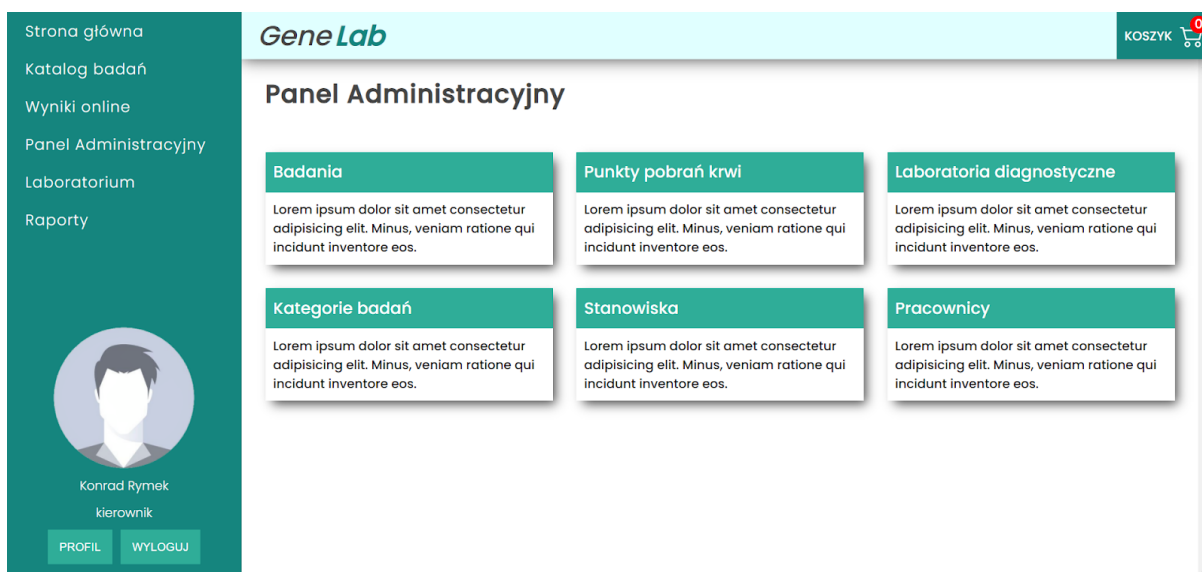


The screenshot displays the 'GeneLab' application interface. On the left is a dark teal sidebar with navigation links: 'Strona główna', 'Katalog badań', 'Wyniki online', 'Panel Administracyjny', 'Laboratorium', and 'Raporty'. Below these links is a user profile for 'Konrad Rymek', 'kierownik', with buttons for 'PROFIL' and 'WYLOGUJ'. The main content area has a light blue header with the 'GeneLab' logo and a 'KOSZYK' icon. Below the header are three tabs: 'PŁATNOŚCI', 'WYNIKI PRACOWNIKÓW', and 'PODSUMOWANIE LABORATORIÓW'. The 'PODSUMOWANIE LABORATORIÓW' tab is active, showing a table with the following data:

Lokalizacja	Pracownicy	Punkty pobrań	Wykonane badania	Całkowity przychód
Gdańsk, Brzegowa 1B	1	1	3	269.97
Kraków, Floriańska 2	1	2	0	0
Wrocław, Rondo Powstańców 2	1	3	9	739.91
Warszawa, Żłota 48/54	1	3	26	1160.74
Warszawa, Aleje Ujazdowskie 34A	1	4	9	307.91
Katowice, Rynek 11	2	2	4	347.96

4.3. Zdefiniowanie panelu sterowania aplikacji

Zarządzanie strukturą oraz ofertą sieci laboratoriów odbywa się z poziomu panelu administracyjnego, w którym dostępne są kafelki prowadzące na dedykowanej podstronie do zarządzania wybranymi zasobami:



Proces wykonywania zleconych badań przez pracowników realizowany jest w zakładce Laboratorium, która pokazuje oczekujące na wykonanie badania w

przypisanym do zalogowanego pracownika laboratorium. Wykonanie badania polega na wprowadzeniu wyniku w okienku:

Strona główna
Katalog badań
Wyniki online
Panel Administracyjny
Laboratorium
Raporty

Konrad Rymek
kierownik
PROFIL WYLOGUJ

GeneLab

KOSZYK 0

Laboratorium: Warszawa, Złota 48/54
Liczba aparatów: 33

Wynik badania [X]

Badanie: Witamina D
Numer zlecenia badania: 49

Wynik: 14 ug

ZATWIERDŹ

Lp.	Nazwa	Material	id
0	Białko całkowite	HW	44
1	Fosfataza kwaśna	maz	45
2	Homocysteina	HW	46
3	Kreatynina	HW	47
4	Witamina B12	HW	48
5	Witamina D	HW	49

4.4. Makropolecenia

Nie zaimplementowano.

5. Dokumentacja

5.1. Wprowadzanie danych

Wprowadzanie danych odbywa się poprzez formularze dostępne w różnych częściach aplikacji. Przed rozpoczęciem pracy z aplikacją należy zainicjalizować bazę danych wstępnymi danymi (objaśnione w kolejnym podpunkcie).

5.2. Dokumentacja użytkownika

Aby móc rozpocząć interakcję z aplikacją należy najpierw zainicjalizować strukturę bazy danych wraz z niezbędnymi funkcjami oraz widokami, a następnie wprowadzić zbiór przykładowych danych. W tym celu w utworzonej bazie danych należy uruchomić skrypt `sql/all.sql` zawierający wszystkie wspomniane definicje struktur, funkcji, widoków oraz instrukcje INSERT w odpowiedniej kolejności.

W celu uruchomienia aplikacji niezbędne jest środowisko uruchomieniowe Node.js, które może pobrać z [oficjalnej strony](#) (najlepiej w najnowszej wersji LTS).

Po rozpakowaniu katalogu z plikami źródłowymi należy przejść do katalogu `server` i z jego poziomu uruchomić polecenie `npm install` w celu zainstalowania wszystkich niezbędnych zależności projektu.

W kolejnym kroku należy uzupełnić plik `.env` standardowymi danymi pozwalającymi na połączenie się z bazą danych (min. `PGHOST`, `PGUSER` itp).

Na koniec pozostając w katalogu `server` należy uruchomić aplikację za pomocą polecenia `npm start`. Aplikacja wystartuje korzystając ze zbudowanej wersji produkcyjnej plików frontendowych i będzie dostępna pod adresem `localhost:8000`. W razie problemów z ładowaniem stron, które mogą być spowodowane nie perfekcyjnym serwowaniem plików produkcyjnych, można uruchomić frontend aplikacji w trybie developerskim. Aby to zrobić należy przejść do folder `client` będącego sąsiadem wcześniej wspomnianego folderu `server` oraz z jego poziomu wykonać analogiczne polecenia `npm install` oraz `npm start`. W tym przypadku aplikacja będzie dostępna na porcie 3000.

Po uruchomieniu aplikacji, jeśli chcemy korzystać z systemu w roli pracownika musimy się zalogować. Sugerowane konto do testowania aplikacji to:

email: konrad.rymek@genelab.com
hasło: password

Jest to konto kierownika laboratorium w Warszawie przy Złotej 48/54. Z jego poziomu możemy zarządzać wszystkimi zasobami za pomocą formularzy w podstronach do których przechodzimy przez panelu administracyjnym. (Dla uproszczenia sprawdzania projektu wszyscy pracownicy mają dostęp do panelu administratora).

W momencie składania zamówień w roli klienta zaleca się wybranie jednego z punktów podlegających pod wspomniane laboratorium (np. Warszawa, Tylna 41F) aby zlecenia badań pojawiły nam się w zakładce Laboratorium na obecnie zalogowanym koncie pracownika - w przypadku wybraniu punktu pobrań podlegającego pod inne laboratorium, aby zobaczyć zlecenia musimy zalogować się na konto pracownika laboratorium pod które podlega wybrany przez nas punkt pobrań.

Przykładowy kod dostępu go już gotowych wyników badań:

kod: 8d7d5323-6cd4-431f-ac83-fb8877021065

5.3. Opracowanie dokumentacji technicznej

Projekt został zrealizowany jako aplikacja webowa. Do części frontendowej wykorzystana została biblioteka React pozwalająca na tworzenie rozbudowanych dynamicznych interfejsów. Obsługa stanu aplikacji klienta stworzona została z wykorzystaniem biblioteki redux oraz jej adaptera do Reacta - react-redux.

Do implementacji serwera użyto biblioteki Express, która pozwoliła w prosty i intuicyjny sposób zaimplementować niezbędne dla części klienckiej REST API.

Struktura kodu zarówno po stronie klienta jak i serwera pozwala na proste rozszerzanie aplikacji o kolejne moduły. Po stronie klienta za poszczególne gałęzie routingu odpowiadają kontrolery o adekwatnych nazwach. Niektóre z nich są zagnieżdżone w innych kontrolerach co w połączeniu z metodami HTTP tworzy transparentne API.

Przykładowe rozszerzenie aplikacji o moduł związany np. z magazynem odczynników polegałoby na zaimplementowaniu nowego kontrolera z niezbędnymi handlerami endpointów oraz podpięciu go pod serwer aplikacji w pliku Server.js poprzedzając go adekwatnym prefixem ścieżki. Po stronie

frontendu wystarczyłoby dodać kolejną podstronę w folderze src/pages oraz uwzględnić ją w routingu głównym aplikacji - plik App.js.

5.4. Wykaz literatury

1. Dokumentacja Node.js [link](#)
2. Dokumentacja biblioteki React [link](#)
3. Dokumentacja biblioteki Express [link](#)
4. Dokumentacja bazy danych PostgreSQL [link](#)