Specyfikacja wymagań

Wymagania użytkownika

1. Produkt musi:

   a) Umożliwiać wprowadzanie danych na temat zabiegów medycznych

   b) Logowanie dla zarejestrowanych i potwierdzonych użytkowników

   c) Generować raport dla administratora

2. Produkt powinien:

   a) Posiadać prosty i przejrzysty interfejs

   b) Wysyłać e-mail potwierdzający dla osób zakładających konto

3. Produkt mógłby

   a) Działać na różnych urządzeniach (mobilnych, desktopowych)

Wymagania biznesowe

1. Produkt musi:

   a) Być dedykowanym rozwiązaniem do zbierania danych medycznych z klinik w Polsce

   b) Mieć stronę internetową dostępną z sieci

   c) Być produktem opartym na narzędziach Open Source

2. Produkt powinien:

   a) Możliwie jak najtańszy w obsłudze i utrzymaniu

3. Produkt mógłby:

   a) Być dosyć łatwo rozszerzalny w przyszłości

Wymagania systemowe

1. Produkt musi:

   a) Być zabezpieczony, ponieważ posiada dane wrażliwe

   b) Posiadać bazę danych do przetrzymywania informacji

   c) Udostępniać graficzny interfejs użytkownika

   d) Obsługiwać wiele użytkowników jednocześnie

2. Produkt powinien:

a) Być łatwo przenaszalny

b) Być dostępny ciągle

c) Obsługiwać kilkudziesięciu pacjentów i do kilkuset zdarzeń rocznie

3. Produkt mógłby

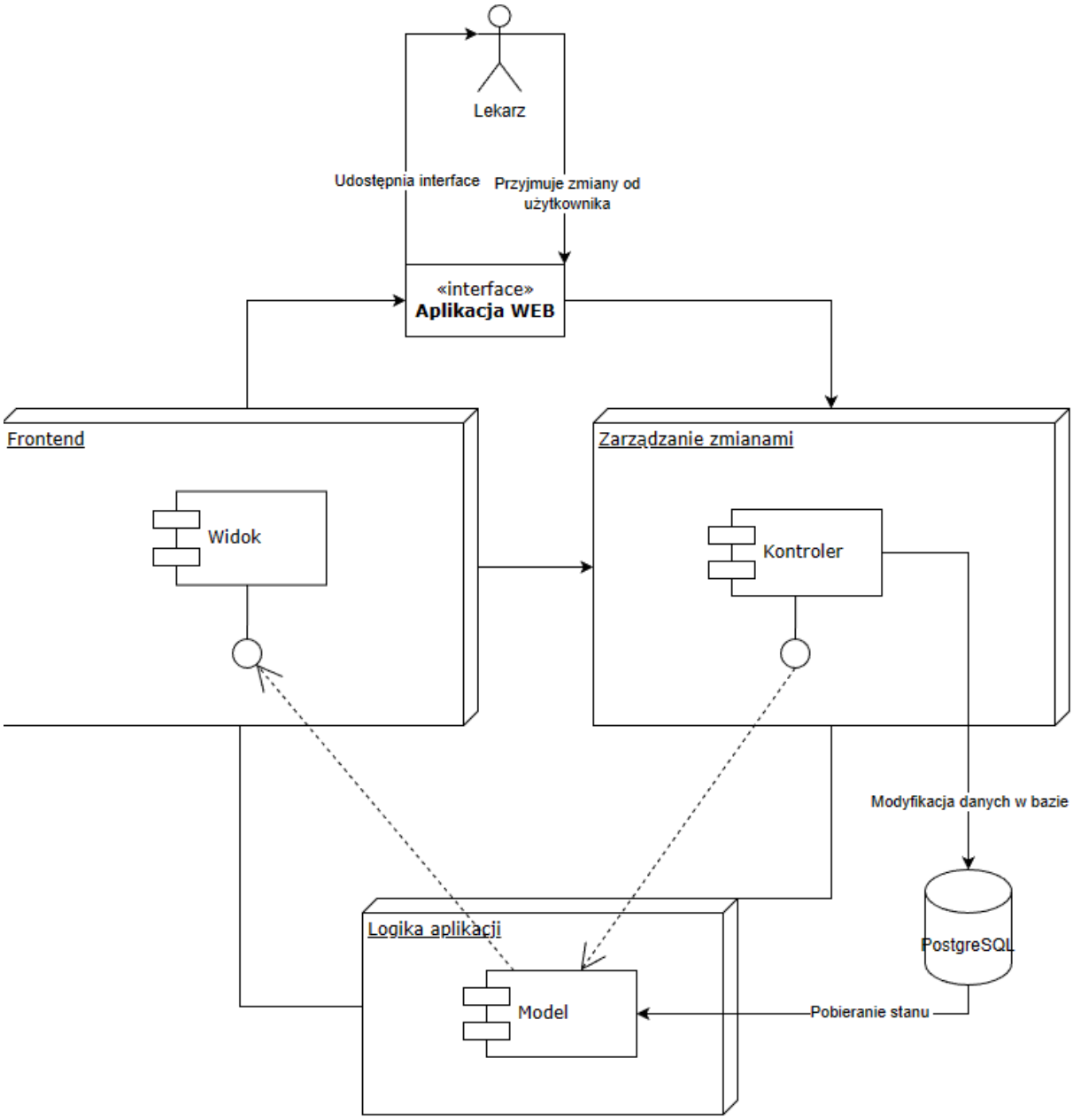
a) Posiadać pełną automatyzację

Specyfikacja przypadków użycia

System będzie używany przez bardzo wąską grupę ludzi. Opis przypadków użycia wraz z historyjkami został zamieszczony w pliku: Specyfikacja przypadków użycia, został również skonsultowany oraz zatwierdzony przez eksperta do spraw UE.

Definicja architektury

Wzorzec architektury



Rysunek 1: Wzorzec architektury

W naszym projekcie korzystamy z wzorca architektury MVC (Model-View-Controller). Podany wzorzec oddziela aplikację od trzech grup składników: modeli, widoków i kontrolerów. Ułatwia to skalowanie aplikacji, ponieważ łatwiej jest kodować, debugować i testować pojedyncze zadania. Aplikacja udostępnia publiczny interfejs w postaci aplikacji WEB dostępnej dla każdej osoby połączonej z internetem.

Wybór padł na ASP MVC .NET z HTTPS, ponieważ generuje on bardzo prosty i łatwy w obsłudze interfejs oraz w duży sposób ułatwia tworzenie projektu, tworząc dużą ilość gotowych szablonów. Będzie zajmował się logiką aplikacji, interfejsem użytkownika oraz połączeniem wszystkiego z bazą PostgreSQL.

Cała aplikacja opiera się na podanej strukturze. Nasz system jest tworzony tak, aby w przyszłości umożliwić proste wdrożenie i łatwe do obsługi rozwiązanie. Planowane jest, aby w wersji końcowej baza danych została postawiona w dockerze z montowaną zewnętrzną pamięcią.

Interakcje pomiędzy elementami

Model – Widok

Model jest odpowiedzialny za logikę aplikacji. Łączy się on w sposób niejawny z modelem – uaktualnia on informacje generowane przez model.

Widok – Kontroler

Widok zajmuje się opisywaniem, w jaki sposób należy wyświetlić daną część interfejsu użytkownika. Odpowiada też, za jego wyświetlenie. Poprzez interakcję użytkownika z generowanym przez widok interfejsem wywołuje się odpowiednie, podpięte pod konkretne działania kontrolery. Kontroler może również wpłynąć wprost na widok – odświeżyć go.

Kontroler – Model

Celem kontrolera jest odbieranie, przetwarzanie oraz analiza danych wejściowych otrzymanych od użytkownika. W jednym momencie aplikacją steruje maksymalnie jeden kontroler, który na podstawie otrzymanych informacji zmienia stan określonego modelu.

Określenie podstawowych mechanizmów technicznych

Serwer aplikacyjny

Serwer aplikacyjny postawiony na lokalnym serwerze WUMu, umożliwiający dostęp dla użytkowników zdalnych.

System bazy danych

Baza danych PostgreSQL postawiona na dockerze, z montowaną zewnętrzną pamięcią.

Mechanizmy zarządzania

Strona internetowa – dla użytkowników oraz administratorów danych. Zmiana systemowa – bezpośrednia zmiana w kodzie systemu.

Mechanizmy bezpieczeństwa

Dostęp do systemu posiadają tylko zarejestrowane i zaakceptowane przez administratorów osoby. Baza danych jest szyfrowana, a hasła użytkowników przetrzymywane jako hashe.

Specyfikacja analityczna

Model dziedziny

Głównym zadaniem naszej aplikacji jest gromadzenie danych na temat historii leczenia metodami nerkozastępczymi u dzieci – podanym leczeniem zajmuje się bardzo nieliczna grupa klinik w Polsce (jest ich zaledwie kilkanaście). Powinna umożliwiać osobom zarejestrowanym i zatwierdzonym przez administratora na wprowadzanie informacji o leczeniu. Administrator danych powinien w łatwy sposób zatwierdzać nowych użytkowników oraz generować raport z wprowadzonych danych. Cały system powinien być jak najprostszy i przyjazny użytkownikowi.

Słownik pojęć

Hash

Wynik działania funkcji skrótu, utrudniający złamanie danej informacji.

PostgreSQL

Jeden z najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych.

ASP MVC .NET

Rozbudowana struktura do tworzenia aplikacji internetowych i interfejsów API z użyciem wzorca architektonicznego Model-View-Controller. Tworzona głównie w językach C# oraz HTML.

Debugować

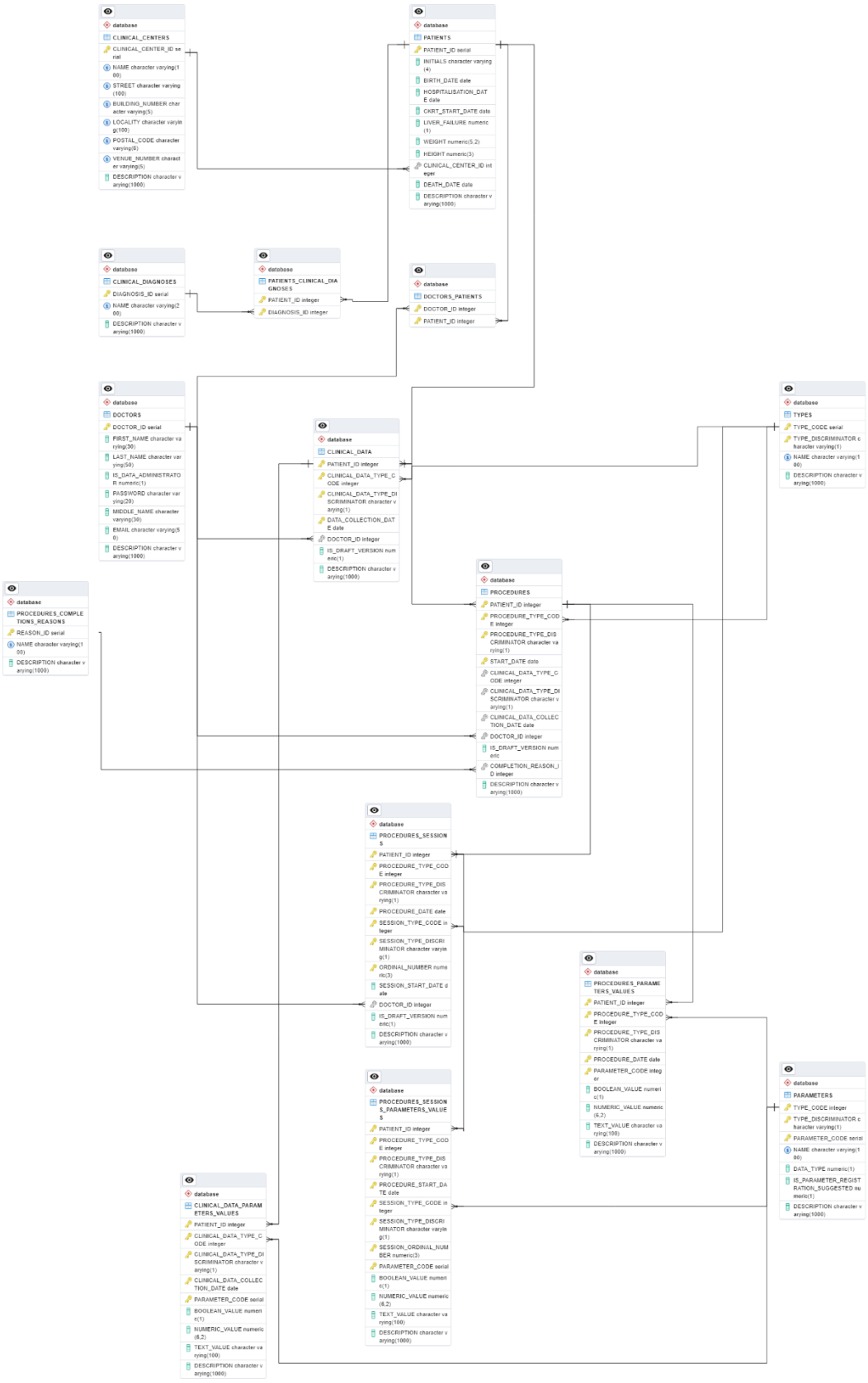
Proces systematycznego redukowania liczby błędów w oprogramowaniu.

Docker

Oprogramowanie służące do realizacji wirtualizacji na poziomie systemu operacyjnego.

Model pojęciowy struktury informacyjnej

Dostępny również: model pojęciowy struktury informacyjnej.



Rysunek 2: Model struktury informacyjnej

Specyfikacja projektowa

Określenie metod realizacji

Język programowania

C#, HTML, CSS.

Języki baz danych

PL/pgSQL

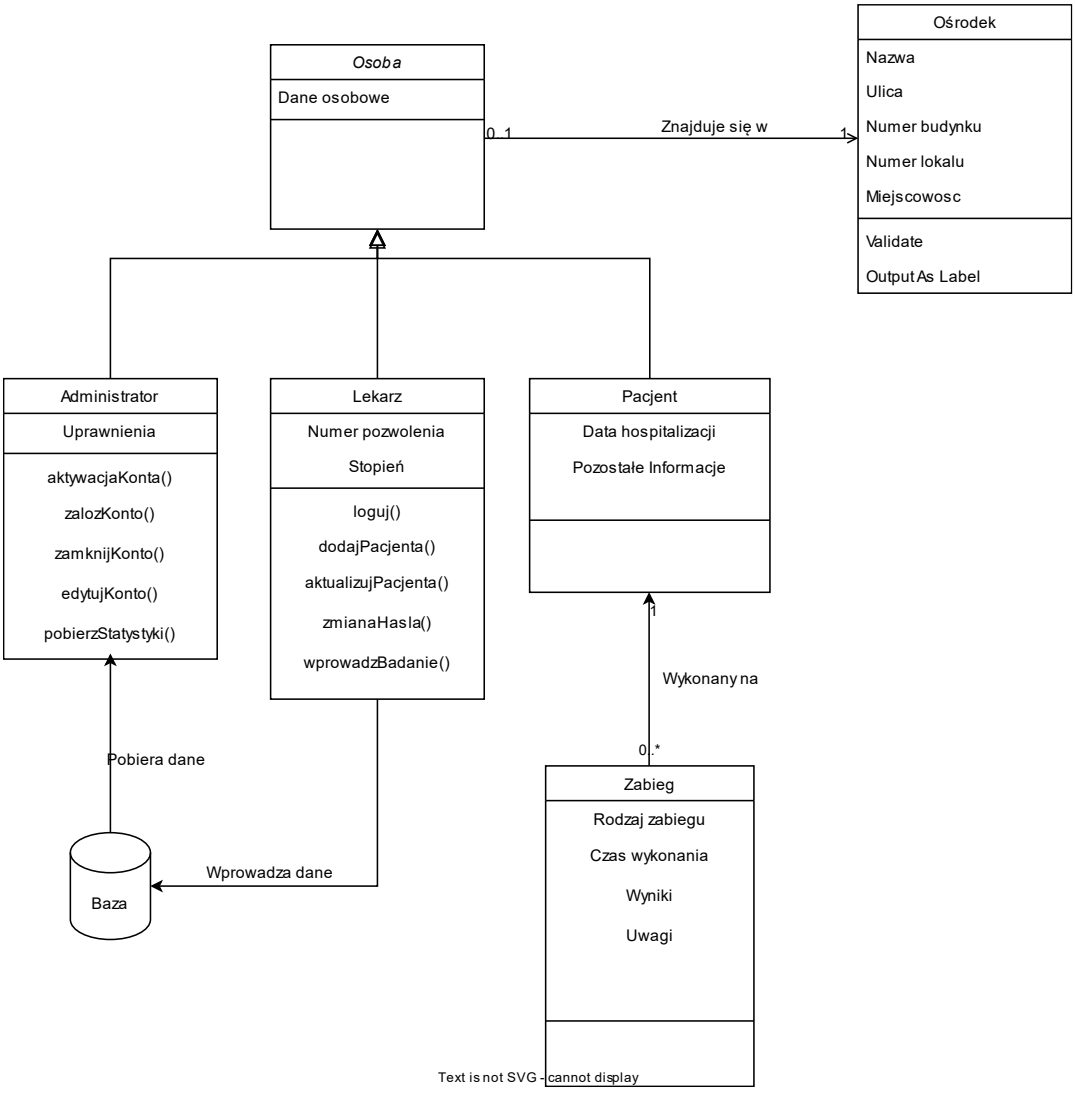
Środowisko programowania, uruchamiania, testowania, wdrażania

Visual Studio, Docker, serwer zewnętrzny

Logiczny model danych

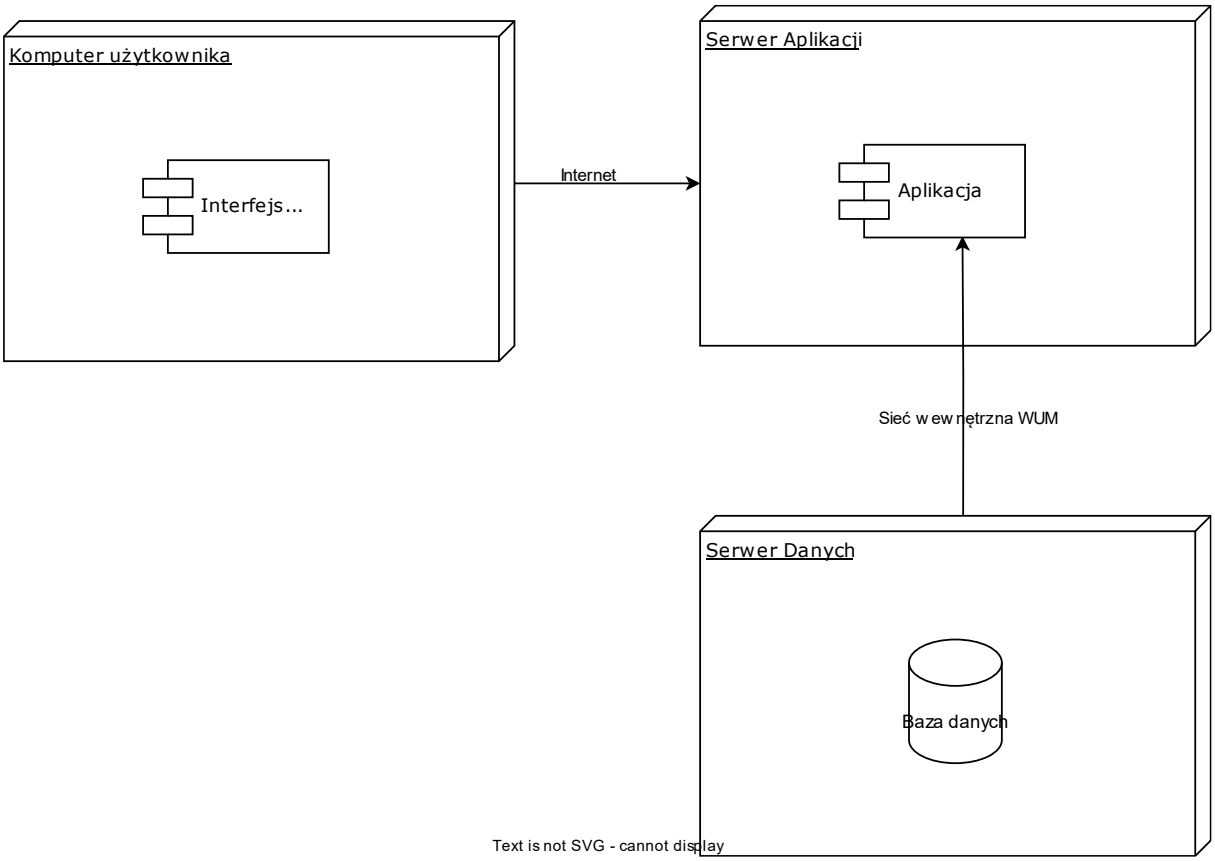
Model danych jest zbyt duży, aby wstawić go do dokumentacji w czytelny sposób. Można go znaleźć pod linkiem: logiczny model danych.

Model obiektowy systemu



Rysunek 3: Diagram klas

Model struktury systemu



Rysunek 4: Diagram wdrożenia

Projekt standardu interfejsu użytkownika

Mockupy naszego systemu wraz z relacjami pomiędzy poszczególnym ekranami i przypadkami użycia. Link do projektu: mockupy,

Specyfikacja testów

Scenariusze testów funkcjonalnych

W ramach testów funkcjonalnych zostały przeprowadzone poniższe testy:

Accept_new_user – zaakceptowanie nowego użytkownika przez administratora.

Creater_account_new_user – stworzenie nowego konta użytkownika i wysłanie e-maila z potwierdzeniem.

Login_user – zalogowanie się zaakceptowanego użytkownika.

Create_new_patient – stworzenie nowego pacjenta przez użytkownika.

Add_new_procedure – stworzenie nowej ankiety dla zabiegu.

Generate_raport – generowanie raportu z wprowadzonych danych

Został również stworzony skrypt sqlowy, który testował strukturę danych w bazie. Znajduje się on tutaj: skrypt.

Podręcznik użytkownika

Otwórz w przeglądarce adres IP naszej strony. Pojawi się okno z logowaniem do naszego systemu. Należy wpisać podane dane i wcisnąć przycisk zaloguj. Aby móc korzystać z aplikacji wcześniej należy się zarejestrować i zgłosić do administratora z prośbą o zatwierdzenie. Po zalogowaniu, wyświetlą się podstawowe informacje o koncie oraz przyciski z możliwością wylogowania lub usunięcia konta. W zakładce pacjenci znajdują się wcześniej wprowadzone przez użytkownika osoby. Można tam również wprowadzić nowego podopiecznego. Pod zakładką ankiety można zobaczyć oraz wprowadzić nowe ankiety dla swoich pacjentów. Zakładka rozpoznanie kliniczne umożliwia zobaczenie lub wprowadzenie nowego rozpoznania. Zakładka zabiegi umożliwia sprawdzenie lub dodanie nowego zabiegu dla danego pacjenta.

Podręcznik administratora

Budowa systemu z kodu źródłowego

https://learn.microsoft.com/en-us/dotnet/core/deploying/deploy-with-cli.

Aplikacja

Baza danych

W celu uruchomienia bazy danych konieczne jest uruchomienie kontenera z obraz Postgresa. Link do repozytorium z obrazami kontenerów postgresa: https://hub.docker.com/_/postgres

Przy uruchamianiu kontenera należy pamiętać o tym, aby dołączyć do niego wolumen. W przeciwnym przypadku, błąd kontenera powodujący zamknięcie obecnego i stworzenie nowego lub aktualizacja obrazu spowodują utratę danych. Oficjalna instrukcja dotycząca montowania wolumenów w kontenerach dockera znajduje się pod tym linkiem:

https://docs.docker.com/storage/volumes/#create-and-manage-volumes

Przykładowe polecenie służące do stworzenia bazy danych znajduje się w repozytorium z projektem: https://gitlab-stud.elka.pw.edu.pl/mkowal21/pzsp2/-/blob/main/Database/create_database_container.txt

W przypadku użycia należy dostosować następujące parametry do swoich potrzeb:

1. *–name* – w podanym przykładzie nazwa tworzonego kontenera to *pzsp2-db*
2. *-p abcd:5432* – zamiast *abcd* należy podać port na którym baza danych ma być dostępna z zewnątrz kontenera. Port *5432* jest domyślnym portem dla Postgresa i taką wartość należy podać po dwukropku, aby ten port był udostępniany na zewnątrz
3. *-e POSTGRES_PASSWORD=xyz* – flaga *-e* oznacza przekazywanie zmiennych i konieczne jest tutaj przekazanie hasła dostępu do bazy danych. Domyślnie tworzony jest użytkownik o nazwie *postgres*

Aktualizacja bazy danych

W ramach aktualizacji silnika bazy danych konieczne jest sprawdzanie dostępności nowych wersji obrazów dockerowych na stronie oficjalnego repozytorium: https://hub.docker.com/_/postgres

W momencie pojawienia się nowej wersji obrazu konieczne jest przebudowanie obrazu zainstalowanego na działającym serwerze. Podczas uruchamiania kontenera z nową wersją obrazu należy zwrócić pamiętać o zamontowaniu wolumenu z danymi.

Tworzenie kopii zapasowych i odtwarzanie bazy danych

Narzędzia

W celu tworzenia kopii zapasowych bazy danych administrator powinien zapoznać się z instrukcją do narzędzia *pg_dump* dostępną na stronie PostgreSQL:

https://www.postgresql.org/docs/current/backup-dump.html

Narzędzie *pg_dump* umożliwia zarówno tworzenie kopii zapasowych jak również odtwarzanie bazy danych.

Szczególną uwagę należy zwrócić na konieczność wykonywania kopii bazy danych z poziomu użytkownika mającego dostęp do całego schematu bazy danych, np. domyślny użytkownik postgres lub dowolny użytkownik stworzony przez administratora, któremu nadane zostały odpowiednio szerokie uprawnienia.

Zalecenia

Zalecamy wykonywanie kopii zapasowej bazy danych przynajmniej raz w miesiącu, np. ostatniego dnia miesiąca. Z racji na przyrostowy charakter bazy danych, tj. dodawania nowych danych tylko z nielicznymi modyfikacjami danych już istniejących w systemie, nie ma konieczności trzymania wszystkich wykonanych kopii. W zależności od rozmiaru danych i dostępnych zasobów dyskowych można np. przyjąć strategię przechowywania ostatnich 6 kopii, co będzie odpowiadało ostatnim 6 miesiącom. Ponadto, można rozważyć opcję przechowania skompresowanej kopii bazy danych na koniec 2 ubiegłych lat.

Należy również pamiętać o tym, że kopia danych przechowywana w tym samym miejscu, co baza danych nie jest kopią zapasową. W przypadku np. pożaru serwerowni wszystkie dane przechowywane w ten sposób zostaną utracone. Zalecamy, aby kopie zapasowe danych były przechowywany przynajmniej w odległości kilku kilometrów od siebie, a najlepiej około 30 kilometrów od siebie na wypadek katastrofy naturalnej, np. powodzi Warszawy. Taki warunek odległości spełniało by przechowywanie danych w dwóch różnych klinikach / szpitalach w Polsce – oczywiście o ile przepisy prawne na to pozwalają. Jeśli powyższe opcje będą niemożliwe z punktu widzenia prawa, to zalecamy rozważenie opcji przechowywania danych w przynajmniej dwóch budynkach Warszawskiego Uniwersytetu Medycznego, np. baza danych na kampusie na Banacha i kopia na zaszyfrowanym dysku twardym na kampusie na Lindleya.

Zarządzanie użytkownikami i uprawnieniami

Aplikacja

Zarządzanie użytkownikami systemu odbywa się z poziomy aplikacji administratora danych. Zgodnie z wymaganiami dostęp do aplikacji przyznawany jest tylko po zatwierdzeniu konta przez administratora danych.

Baza danych

Skrypty przygotowane do stworzenia bazy danych nie tworzą dodatkowych użytkowników. Jedynym stworzonym użytkownikiem będzie domyślny użytkownik *postgres* wraz z hasłem dostępu podanym podczas tworzenia kontenera.

Należy zwrócić uwagę na to, ze wszystkie obiekty w bazie danych tworzone są w schemacie database, który zostanie utworzony podczas wykonywania przygotowanych skryptów. W przypadku konieczności dodania nowych użytkowników do bazy danych konieczne będzie przydzielenie im odpowiednich uprawnień do tego schematu lub obiektów z tego schematu. Warto również rozważyć utworzenie

synonimów do obiektów dla nowych użytkowników. Ułatwi to zarządzanie dostępem w przypadku zmiany struktury bazy danych oraz pozwala w bardziej precyzyjny sposób kontrolować dostęp do zawartości bazy danych.

Poniżej znajdują się linki do dokumentacji Postgresa omawiających następujące aspekty:

1. Tworzenie nowych użytkowników: https://www.postgresql.org/docs/8.0/sql-createuser.html
2. Nadawanie uprawnień użytkownikom: https://www.postgresql.org/docs/current/sql-grant.html
3. Tworzenie synonimów do obiektów bazy danych: https://www.postgresql.org/message-id/440D446E.7040509@cybertec.at

Administrowanie systemem

W celu administrowania systemem bazy danych zalecane jest korzystanie z jednego z dwóch wymienionych niżej narzędzi:

1. *PgAdmin* – oficjalne narzędzie do administrowania bazami danych Postgres. Posiada najwięcej narzędzi dedykowanych bazom danych Postgres. Opórcz pisania standardowych zapytań umożliwia m. in. wygodny wgląd w obiekty oraz dane składowane w bazie danych. Ponadto, posiada narzędzie do monitorowania wydajności i obciążenia bazy danych.
2. *DBeaver* – narzędzie typu open-source. Pozwala na zarządzanie bazami danych od różnych dostawców. Również pozwala na wgląd do danych i obiektów składowanych w bazie danych.

Wszystkie potrzebne informacje oraz dokumentacja znajduje się na naszym repozytorium: https://gitlab-stud.elka.pw.edu.pl/mkowal21/pzsp2/-/tree/main.