Bazy danych I – dokumentacja projektu

Piotr Libucha

1. Projekt koncepcji, założenia

1.1. Zdefiniowanie tematu projektu

Projekt reprezentuje uproszczoną bazę danych bliżej nieokreślonego sklepu. Baza pozwala na zbieranie podstawowych danych między innymi o klientach, produktach, zamówieniach i pracownikach. Celem systemu jest ułatwienie zarządzania sklepem.

1.2. Analiza wymagań użytkownika

Baza danych ma za zadanie przede wszystkim przechowywać podstawowe informacje potrzebne do prowadzenia działalności sklepu. Oprócz podstawowej funkcjonalności dodawania i przeglądania danych w tabelach, aplikacja generuje raporty z danymi takimi jak np. zestawienie produktów, które należy niezwłocznie zamówić, ranking najlepiej sprzedających się produktów, czy ranking najwydajniejszych pracowników. Raporty ułatwiają optymalizację działalności.

1.3. Zaprojektowanie funkcji

Oprócz podstawowej funkcjonalności dodawania i przeglądania danych, znaczna część funkcji skupia się na zapobieganiu przypadkom wprowadzania błędnych danych. Walidowane są wszystkie formularze pozwalające na modyfikację danych.

2. Projekt diagramów

2.1. Budowa i analiza diagramu przepływu danych (Data Flow Diagram)

Ze względu na prostotę projektu, przepływ danych jest całkowicie pionowy. Dane mogą być wprowadzane jedynie poprzez formularz w aplikacji. Aplikacja wywołuje funkcję dodającą wpis do bazy danych. Przed finalizacja transakcji wywoływane są jeszcze triggery walidujące.

2.2. Zaprojektowanie encji

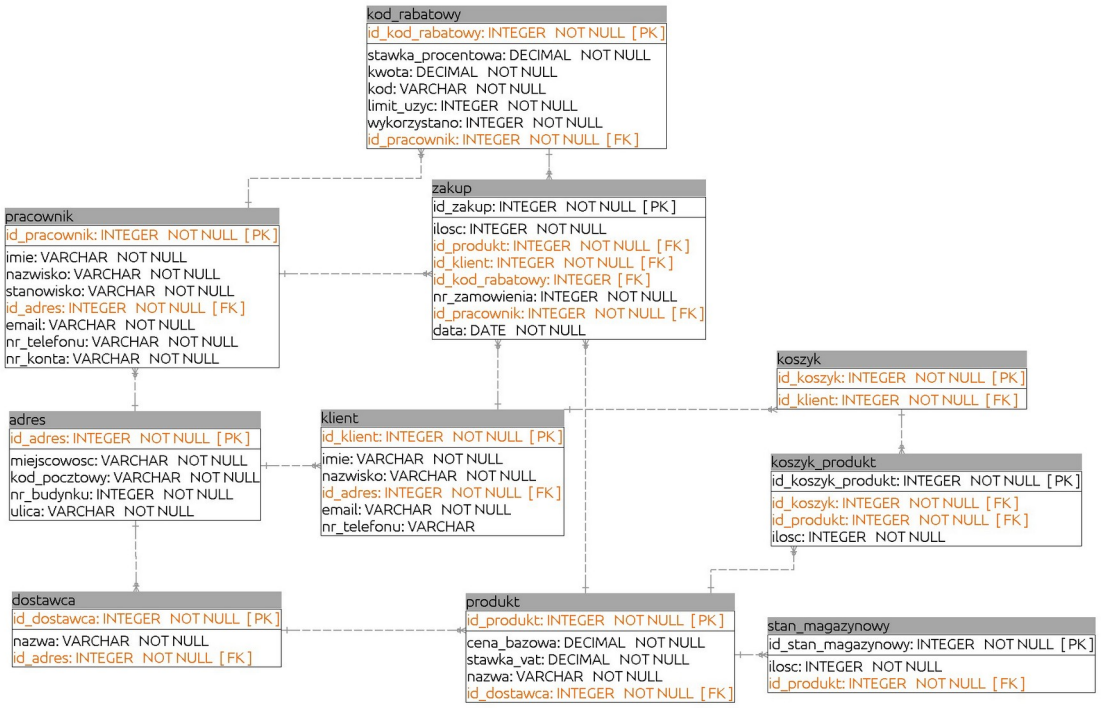
W bazie znajduje się 10 encji, są to:

- pracownik – Reprezentuje pracownika sklepu, zawiera jego dane Kontaktowe.
- klient – Reprezentuje klienta, zawiera jego dane kontaktowe.
- produkt – Przechowuje informacje o danym produkcie, takie jak jego nazwa, cena i stawka podatku VAT.
- dostawca – Zawiera informacje o dostawcy produktów.
- adres – Przechowuje informacje o adresie klienta, pracownika lub dostawcy.

- kod_rabatowy – Reprezentuje kod rabatowy pozwalający na obniżenie ceny produktu. Rabat może być zarówno procentowy, jak i wyrażony konkretną kwotą. Kod może być wykorzystany skończoną ilość razy.
- zakup – Zawiera informacje o zakupie pewnej ilości pojedynczego towaru z pojedynczego zamówienia.
- koszyk – Reprezentuje wirtualny koszyk z produktami, które klient zamierza kupić lub którymi jest zainteresowany.
- koszyk_produkt – Zawiera informację o produktach w koszyku.
- stan_magazynowy – Przechowuje informacje o ilości produktów w magazynie sklepu.

2.3. Zaprojektowanie relacji między encjami

Relacje między encjami zostały zaprezentowane na diagramie ERD (Entity Relationship Diagram), znajdującym się poniżej.



3. Projekt logiczny

3.1. Projekt tabel kluczy i indeksów

Tabele zostały zaprojektowane i stworzone zgodnie z przedstawionym diagramem ERD. Kod SQL użyty do generacji utworzenia tabel w bazie znajduje się w pliku sql/create_tables.sql oraz stanowi część pliku sql/create_all.sql. Poniżej znajdują się się tabelki z dokładnym opisem atrybutów wszystkich encji oraz ograniczeń danych, jakie mogą się w nich znaleźć (słowniki).

pracownik					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_pracownik	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
imie	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃ][a-ząęółśżźćń]*$	Imię
nazwisko	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃ][a-ząęółśżźćń]*$	Nazwisko
stanowisko	VARCHAR	NOT NULL		^[a-ząęółśżźćń]*$	Stanowisko pracownika
id_adres	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny adresu
email	VARCHAR	NOT NULL		[a-z]*@[a-z]*\.[a-z]*$	Email
nr_teleofnu	VARCHAR	NOT NULL		^[0-9]{9}$	Numer telefonu
nr_konta	VARCHAR	NOT NULL		^[0-9]{26}$	Numer konta bankowego

klient					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_klient	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
imie	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃ][a-ząęółśżźćń]*$	Imię
nazwisko	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃ][a-ząęółśżźćń]*$	Nazwisko
id_adres	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny adresu
email	VARCHAR	NOT NULL		[a-z]*@[a-z]*\.[a-z]*$	Email
nr_teleofnu	VARCHAR			^[0-9]{9}$	Numer telefonu

produkt					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_produkt	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
cena_bazowa	DECIMAL	NOT NULL		> 0	Cena bazowa / netto
stawka_vat	DECIMAL	NOT NULL		>= 0	Stawka VAT, jaką opodatkowany jest produkt
nazwa	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃa-ząęółśżźćń0-9\-@[:space:]]*$	Nazwa własna produktu
id_dostawca	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny dostawcy

dostawca					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_dostawca	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
nazwa	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃa-ząęółśżźćń\.\-0-9[:space:]]*$	Nazwa własna produktu
id_adres	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny adresu

adres					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_adres	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
miejscowosc	VARCHAR	NOT NULL		^[A-ZĄĘÓŁŚŻŹĆŃ][a-ząęółśżźćń[:space:]]*$	Miejscowość
kod_pocztowy	VARCHAR	NOT NULL		^[0-9]{2}-[0-9]{3}$	Kod pocztowy
nr_budynku	INTEGER	NOT NULL		> 0	Numer budynku
ulica	VARCHAR	NOT NULL		> 0	Ulica

kod_rabatowy					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_kod_rabatowy	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
stawka_procentowa	DECIMAL	NOT NULL		>= 0	Stawka procentowa, procent jest odliczany od całkowitej kwoty zamówienia
kwota	DECIMAL	NOT NULL		>= 0	Kwota odliczana od całkowitej kwoty zamówienia
kod	VARCHAR	NOT NULL		^[A-Za-z0-9]*$	Kod
limit_uzyc	INTEGER	NOT NULL		>= 0	Limit ilości zamówień, do których kod może zostać użyty
wykorzystano	INTEGER	NOT NULL		>= 0	Ilość razy, ile kod został wykorzystany
id_pracownik	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny pracownika, który utworzył kod

zakup					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_zakup	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
ilosc	INTEGER	NOT NULL		> 0	Ilość sztuk lub jednostek zakupionego produktu
id_produkt	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny produktu
id_klient	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny klienta
id_kod_rabatowy	INTEGER		Foreign	Foreign key	Klucz główny użytego kodu rabatowego, o ile został użyty
nr_zamówienia	INTEGER	NOT NULL		>= 0	Numer zamówienia
id_pracownik	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny pracownika, który sprzedał produkt
data	DATE	NOT NULL			Data zakupu

koszyk					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_koszyk	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
id_klient	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny klienta, do którego należy koszyk

koszyk_produkt					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_koszyk_produkt	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
id_koszyk	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny koszyka
id_produkt	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny produktu
ilosc	INTEGER	NOT NULL		> 0	Ilość sztuk lub jednostek produktu w koszuku

stan_magazynowy					
Atrybut	Typ	Własności	Key	Ograniczenia	Opis
id_stan_magazynowy	INTEGER	NOT NULL AUTO INCREMENT	Primary	> 0	Klucz główny, generowany automatycznie
ilosc	INTEGER	NOT NULL		>= 0	Ilość sztuk lub jednostek produktu w magazynie
id_produkt	INTEGER	NOT NULL	Foreign	Foreign key	Klucz główny produktu

3.3. Analiza zależności funkcyjnych i normalizacja tabel

Każda komórka zawiera jedną wartość. Wszystkie rekordy są unikalne ze względu na zastosowanie unikalnych, automatycznie inkrementowanych, kluczy głównych. Żaden klucz nie zależ od żadnego zbioru relacji lub wartości komórek. Wartości kolumn nie będących kluczami zależą od kluczy głównych i opisują rekordy wskazywane przez klucze główne. Wzajemne zależności między kolumnami nienależącymi do kluczy nie występują z wyjątkiem tabel kod_rabatowy i stan_magazynowy. W tych dwóch tabelach została zastosowana celowa denormalizacja. Reszta bazy jest w trzeciej postaci normalnej.

3.4. Denormalizacja struktury tabel

Tabela kod_rabatory denromalizuje tabelę. Jest to zabieg celowy, dzięki któremu nie trzeba wykonywać drogiej operacji liczenia ilości użyć danego kodu za każdym razem, kiedy potrzebna jest znajomość liczby użyć kodu. Zamiast tego, użyty został trigger, odpowiednio inkrementujący atrybut wykorzystano. Analogicznie, ilość produktów w tabeli stan_magazynowy, jest zmieniana odpowiednimi triggerami.

3.5. Projekt operacji na danych

Użytkownik korzystający z aplikacji ma możliwość dodawania rekordów za pomocą formularzy, do każdej tabeli w bazie. Wszystkie wprowadzane dane są walidowane za pomocą triggerów, zgodnie ograniczeniami opisanymi w punkcie 3.2. Zostały wprowadzone również inne triggery, opisane poniżej. Użytkownik może również przeglądać informacje zawarte we wszystkich tabelach w bazie. Poza podglądem tabel, użytkownik ma dostęp do kilku widoków.

3.5.1. Triggery o działaniu innymi niż podstawowa walidacja

- validate_stan_magazynowy() - Oprócz walidacji, sprawdza, czy w tabeli jest już wpis dotyczący danego produktu. Jeżeli takowy istnieje, to jest on zastępowany nową wersją.
- validate_zakup() - Oprócz podstawowej walidacji, sprawdza, czy w magazynie znajduje się odpowiednia ilość towaru. Sprawdza również, czy podany kod rabatowy może zostać użyty (czy nie wyczerpano limitu).
- after_insert_zakup() - Zmniejsza ilość produktu w magazynie po dodaniu nowego zakupu, oraz, jeżeli użyto kodu rabatowego, zwiększa licznik wykorzystania kodu.
- Pozostałe 7 triggerów realizuje jedynie walidację zgodną z tabelami w punkcie 3.2.

3.5.2. Widoki

- zamowienie – Zestawia zamówienia na podstawie tabeli zakup.
- Łączna kwota udzielonych rabatów według kodu – Dla każdego kodu rabatowego liczy łączną kwotę, o jaką zostały obniżone ceny zakupionych produktów
- Zamówienia o największej łącznej kwocie przed vat – Ranking zamówień (z widoku zamowienie) pod względem łącznej ceny netto

- Pracownicy według największej sprzedaży – Ranking pracowników pod względem sumy wszystkich kwot netto zapłaconych przez klientów, którym wystawiali rachunek. Nie uwzględnia rabatów.
- Produkty według wygenerowanego przychodu – Ranking produktów pod względem sumy zapłaconej za wszystkie zakupione sztuki.
- Podatek do zapłacenia według stawki VAT – Liczy całkowitą zapłaconą kwotę podatku VAT dla każdej stawki procentowej.
- Produkty do zamówienia ASAP – Produkty, które powinny zostać zamówione jak najszybciej, ponieważ ilość sztuk danego produktu w magazynie jest mniejsza niż łączna ilość sztuk w koszykach klientów, co może skutkować niemożliwością realizacji zamówień.

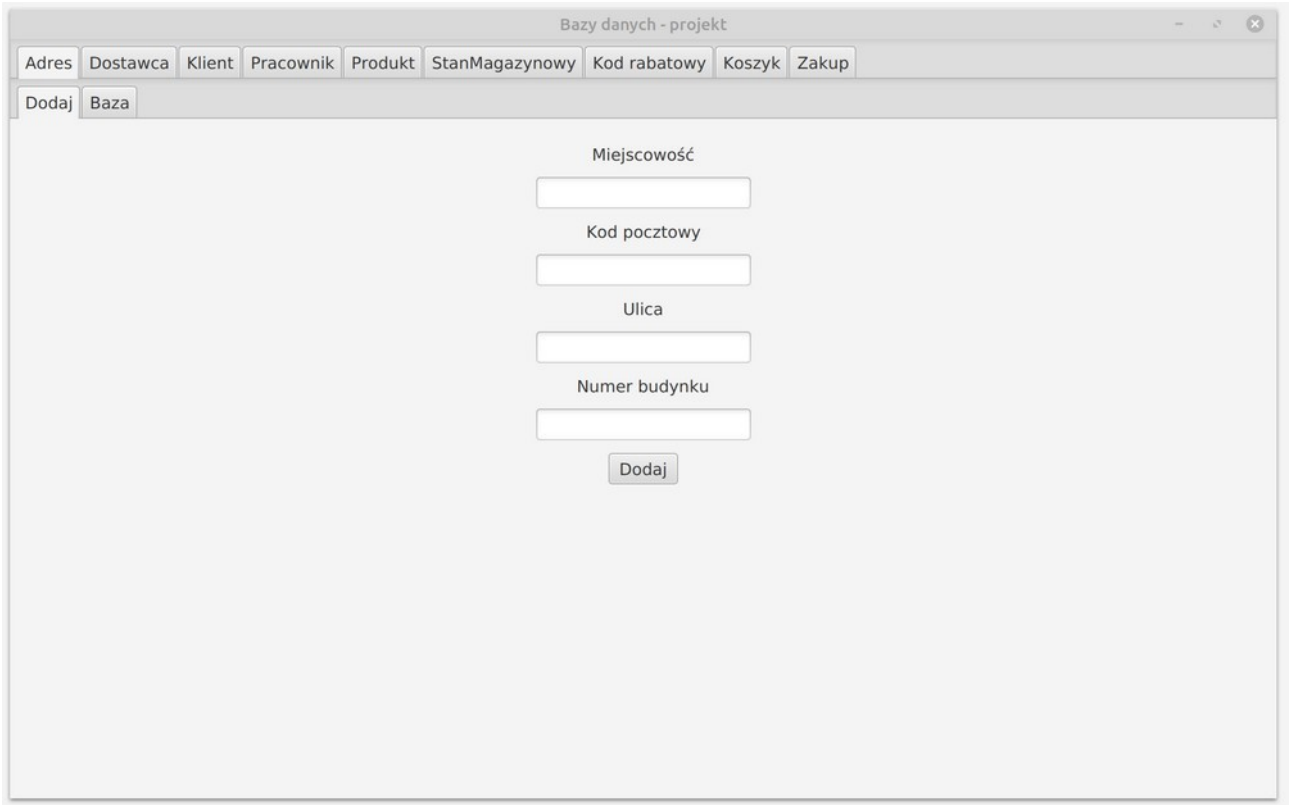
3.5.3. Funkcje

Każdy formularz ma odpowiadającą mu funkcję służącą do wprowadzania danych.

4. Projekt funkcjonalny

4.1. Interfejsy do obsługi danych

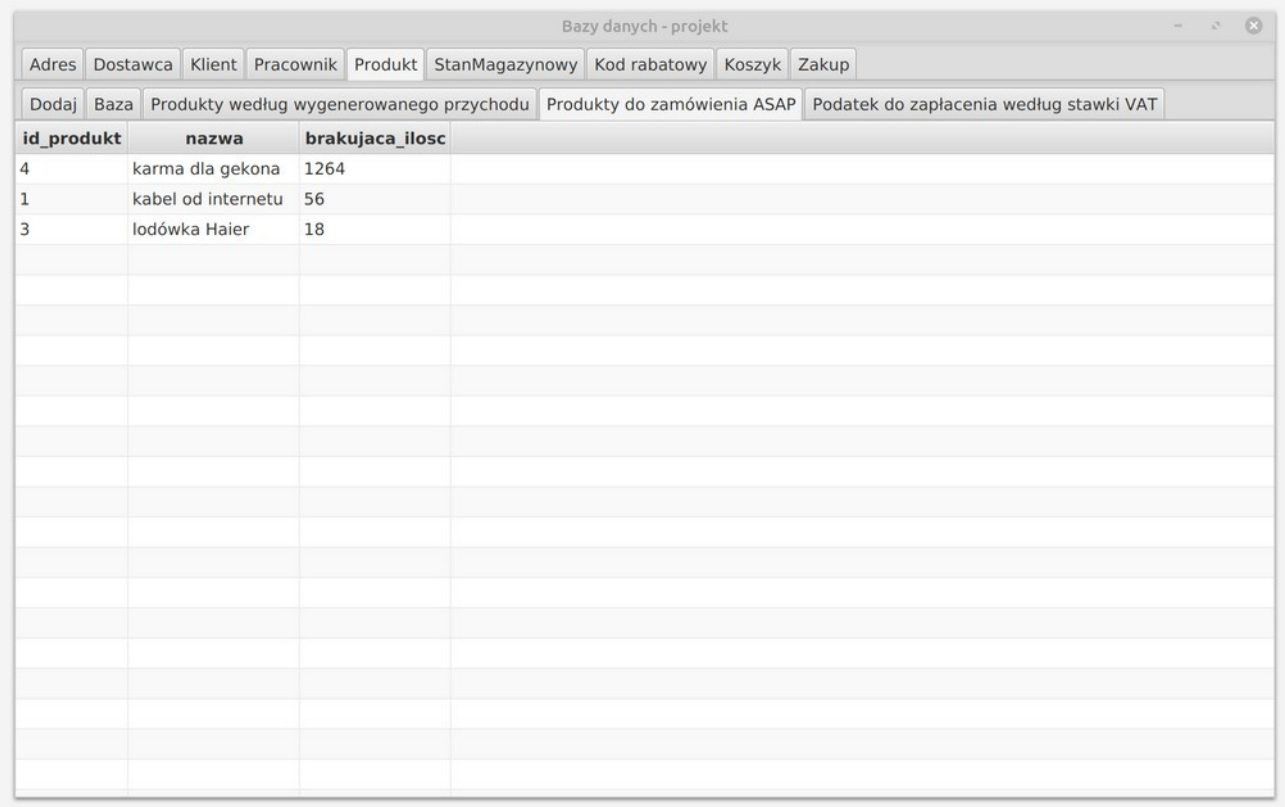
W aplikacji mamy dostępne zakładki dla wszystkich tabel. Po wybraniu zakładki dla tabeli, możemy wybrać podzakładkę z formularzem do wprowadzania danych oraz jedną lub więcej zakładek do przeglądania danych



Screen 1: Formularz do wprowadzania danych adresowych.

4.2. Wizualizacja danych

W podzakładkach mamy możliwość przeglądania danych. Nieprzetworzone dane z tabel znajdują się w zakładkach zatytułowanych „Baza”. Pozostałe zawierają tabele z widoków opisanych w podpunkcie 3.5.2.



Screen 2: Tabela widoku z produktami do zamówienia ASAP

5. Dokumentacja

5.1. Wprowadzanie danych

W pliku sql/populate_tables.sql znajduje się kod SQL wprowadzający do tabel dane testowe. Polecenia te są zawarte również w sql/create_all.sql. Dane te zostały wygenerowane z użyciem dołączonego programu, którego od znajduje się w pliku sql_generator/generate_populate_tables.py. Użytkownik aplikacji wprowadza dane jedynie z użyciem formularzy w aplikacji.

5.2. Instrukcja obsługi aplikacji

5.2.1. Aplikację uruchamia się poleceniem java -jar bd_project.jar [args], znajdując się w katalogu z plikiem bd_project.jar. W miejsce [args] należy podać kolejno:

- url – URL do bazy
- user – Nazwa użytkownika bazy
- password – Hasło użytkownika user

W przypadku nie podania któregoś z argumentów, zostaną użyte wartości domyślne, takie jak poniżej:

- url – jdbc:postgresql://localhost:5432/db_project

- user – db_project_role
- password – password

5.2.2. Dane w aplikacji wprowadza się za pomocą odpowiednich formularzy. W przypadku, kiedy dane nie przejdą walidacji, pod formularzem zostanie wyświetlona na czerwono informacja o błędzie.

5.3. Dokumentacja techniczna

5.3.1. Do stworzenia interfejsu użytkownika została wykorzystana biblioteka JavaFX. Interfejs został zaprojektowany przy użycia narzędzia Scene Builder i jest zdefiniowany w pliku src/main/resources/com/example/bd_project/main.fxml. Połączenie z bazą danych jest realizowane przy użyciu paczki JDBC.

5.3.2. Klasa DatabaseController zestawia połączenie z bazą danych i służy jako interfejs. Zawarte w niej metody publiczne pośredniczą w interakcjach z bazą. Sygnatury metod publicznych oddają ich funkcjonalność i sposób użycia na tyle dobrze, że nie potrzeba dalszych komentarzy.

5.3.3. Klasa Contoller obsługuje graficzny interfejs użytkownika. Pola z adnotacjami @FXML odnoszą się do kontrolek GUI. Metody z adnotacjami @FXML są callbackami wywoływanymi przez kontrolki GUI. Metody populateTable* wypełniają tabele danymi pozyskanymi z bazy. Są używane w callbackach GUI.

5.3.4. Klasy dziedziczące po klasie abstrakcyjnej TableElement reprezentują pojedyncze rekordy z tabel w bazie danych. Klasa SelectResultRecord również dziedziczy po TableElement i jest używana w przypadkach, kiedy nie zależy nam na zachowaniu abstrakcyjnej reprezentacji rekordów wyniku zapytania do bazy.

5.4. Przygotowanie projektu do testowania

5.4.1. Zacząć należy od utworzenia odpowiedniego użytkownika w bazie PostgreSQL (można też wykorzystać istniejącego).

5.4.2. Następnie należy przygotować bazę poprzez wywołanie kodu w pliku sql/create_all.sql. Bazę można przygotować również wywołując po kolei kod z plików:

- sql/create_tables.sql
- sql/populate_tables.sql
- sql/create_insert_functions.sql
- sql/create_triggers.sql
- sql/create_views.sql

Oba sposoby prowadzą do jednakowego rezultatu. Dane testowe zawarte w pliku sql/populate_tables.sql można zastąpić innymi. Dane można wygnerować z użyciem konfigurowalnego skryptu sql_generator/generate_populate_tables.py

5.4.3. Powyższe kroki wystarczają do przygotowania aplikacji do uruchomienia. Aplikację należy uruchomić zgodnie z instrukcją w punkcie 5.2.1, z danymi zgodnymi z punktem 5.4.1.