

Uniwersytet Rzeszowski

Kolegium Nauk Przyrodniczych

Bazy danych

Dokumentacja aplikacji bazodanowej

Wykonali: Ewa Janisz, Patryk Dyndał

Rzeszów, 28.06.2020

Spis plików:

1. Główne klasy zarządzające aplikacją, ścieżka: **.\gui\Projekt-Bazy-Danych\src\sample:**

- Main.java
- Database.java
- Session.Java

2. Pliki bazy danych, ścieżka: **.\sql:**

- create_procedures.sql
- create_user.sql
- export.sql
- set_initial_values.sql
- sql_create_tables.sql

3. Klasy Java tworzące obiekty wzorujące tablice z bazy sql, ścieżka: **.\gui\Projekt-Bazy-Danych\src\entity:**

- pozwolenia.java
- priorytet.java
- projekt.java
- role.java
- typ_zgloszenia.java
- utworzone.java
- uzytkownik.java
- zamkniete.java
- zarzadzanie_projektem.java
- zgloszenia.java

4. Pliki Java FXML, ścieżka: **.\gui\Projekt-Bazy-Danych\src\sample:**

- dodaj.fxml
- dodaj_zgloszenie.fxml
- edytuj.fxml
- edytuj_projekt.fxml
- lista_projektow.fxml
- login.fxml
- nowe_konto.fxml

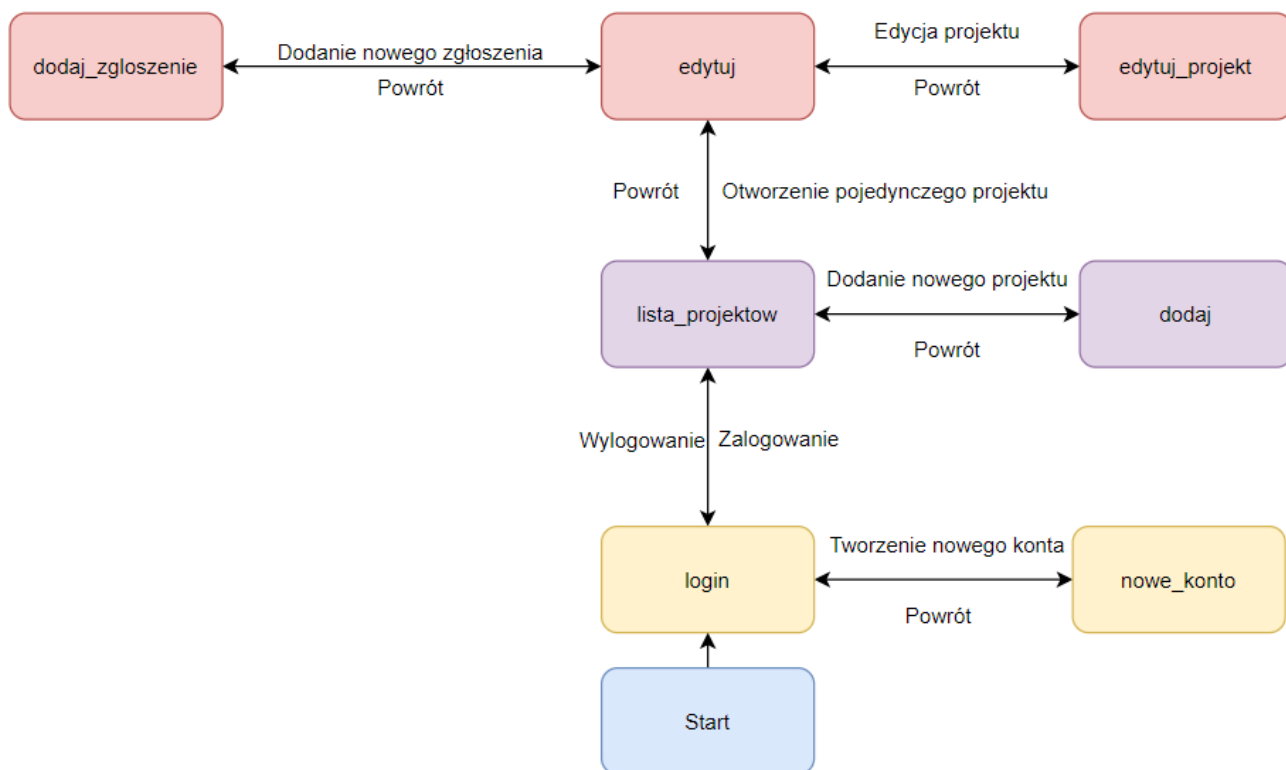
5. Kontrolery okien FXML, ścieżka: **.\gui\Projekt-Bazy-Danych\src\sample:**

- dodajController.java
- edytujController.java
- lista_projektow_Controller.java
- loginController.java

- nowe_kontoController.java

Schemat działania:

Poruszanie się po aplikacji. Wewnątrz komórek wpisane są nazwy plików, opis ich funkcjonalności znajduje się poniżej.

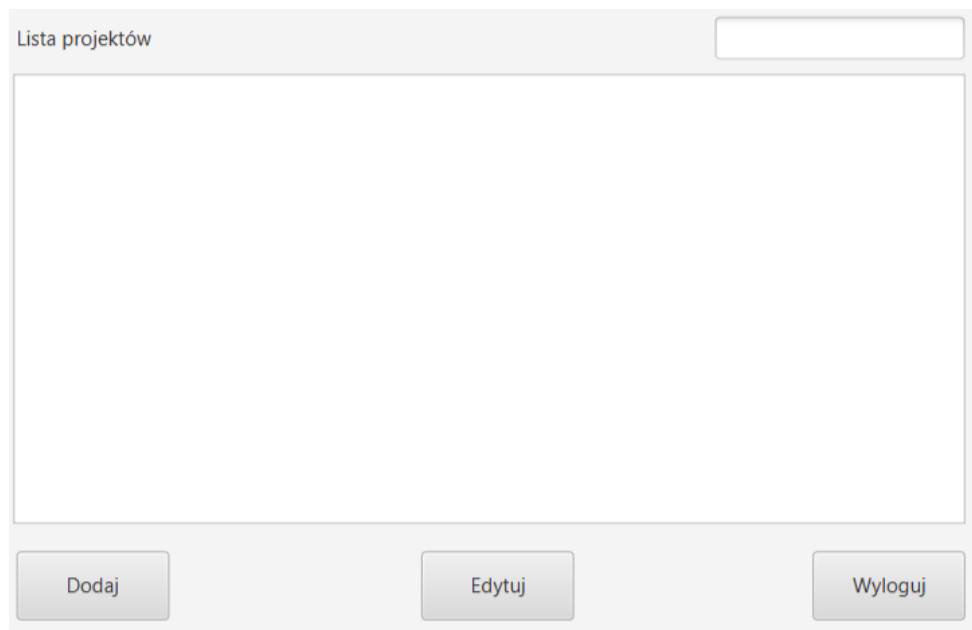


Rys 1.1 - schemat poruszania się wewnątrz aplikacji

Użytkownik startuje z panelu logowania, gdzie może się zalogować bądź stworzyć nowe konto (po czym zalogować się za jego pomocą).

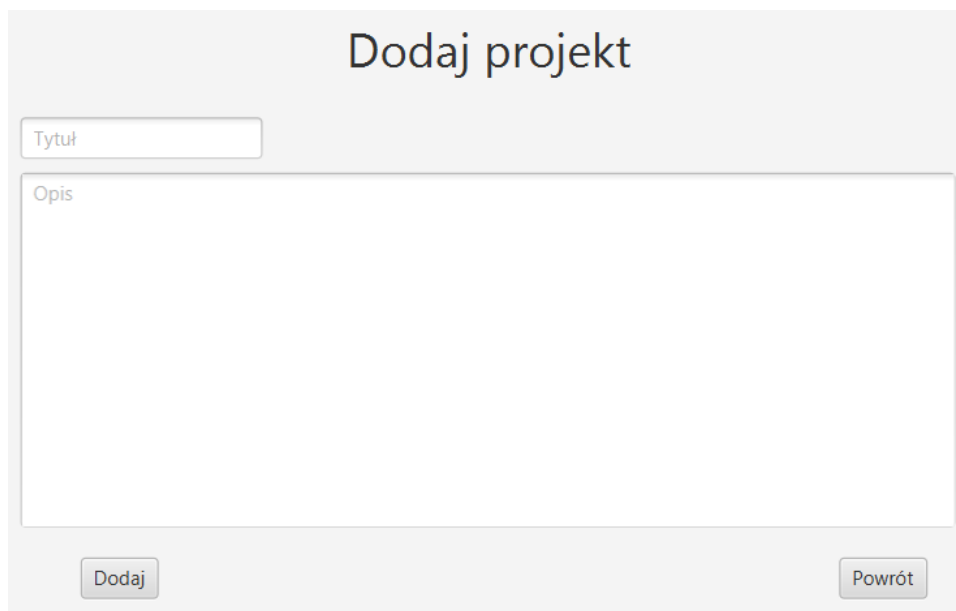
Rys. 1.2 - Okna logowania oraz dodania nowego konta

Po zalogowaniu, użytkownik przechodzi do okna wyświetlającego listę projektów oraz ich tytuły, zaś w prawym górnym rogu wyświetlany jest id użytkownika



Rys. 1.3 - lista_projektow.fxml

W tym miejscu użytkownik może dodać nowy projekt poprzez naciśnięcie przycisku “Dodaj” przenoszące go do pliku **dodaj**, gdzie należy podać tytuł oraz opis projektu.



Rys. 1.4 - dodaj.fxml

Natomiast po wybraniu projektu i naciśnięciu przycisku “Edytuj” otwierane jest nowe okno **edytuj**, zawierające informacje o wybranym projekcie oraz listę zgłoszeń dotyczących go. Od góry pola zawierają, tytuł projektu, jego opis oraz listę zgłoszeń.

The screenshot shows a window titled 'edytuj.fxml'. It contains a form with the following elements:

- A text input field for the title.
- A larger text input field for the description.
- A label 'Lista zgłoszeń' above a list box.
- Three buttons at the bottom: 'Edytuj projekt', 'Dodaj zgłoszenie', and 'Powrót'.

Rys. 1.5 - edytuj.fxml

Naciśnięcie przycisku “Edytuj projekt” przenosi użytkownika do bliźniaczego okna z Rys. 1.4 pozwalającego na edycję danych wybranego projektu, natomiast przycisk “Dodaj zgłoszenie” pozwala przejść do okna zawierającego rozbudowane menu, wymagające podania tytułu zgłoszenia, jego opisu, podsumowania oraz wybrania z dostępnych opcji typu i priorytetu.

The screenshot shows a window titled 'Dodaj zgłoszenie'. It contains a form with the following elements:

- Fields for 'Tytuł', 'Typ' (with a dropdown arrow), and 'Priorytet' (with a dropdown arrow).
- A large text input field for 'Opis'.
- A text input field for 'Podsumowanie'.
- Two buttons at the bottom: 'Dodaj' and 'Powrót'.

Rys. 1.6 - dodaj_zgloszenie.fxml

Opis plików:

Po uruchomieniu, publiczna metoda klasy **Main** – **start()** tworzy obiekt rodzica do którego ładowany jest plik .fxml odpowiedzialny za okno logowania, element początkowy działania programu.

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        primaryStage.setTitle("Aplikacja do Baz Danych");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }

    public static void main(String[] args) {
        // Database db = new Database();
        // db.test_connect();
        Launch(args);
    }
}
```

Rys. 1.7 - klasa Main aplikacji

Cała logika każdego okna znajduje się wewnątrz klasy kontrolera, np.. dla okna login, logika znajduje się w klasie **loginController**.

```
public class loginController {
    Database db = new Database();

    @FXML
    private TextField user_nick;

    @FXML
    void new_account(ActionEvent event) { db.changeWindow(event, resourceName: "nowe_konto.fxml"); }

    @FXML
    void zaloguj(ActionEvent event) {
        String nick = user_nick.getText();

        if (db.get_user(nick)){
            db.changeWindow(event, resourceName: "lista_projektow.fxml");
        }

        else{
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Błąd");
            alert.setHeaderText("Niepoprawne dane");
            alert.setContentText("Użytkownik nie został znaleziony.");

            alert.showAndWait();
        }
    }
}
```

Rys.1.8 - Przykładowa klasa kontrolera

Wnętrze tej klasy (jak i następnych kontrolerów) składa się głównie z metod obsługujących elementy okna takie jak przyciski, pola tekstowa itp., oraz odwołań do klasy **Database**, odpowiadającej za tworzenie/usuwanie/modyfikowanie rekordów bazy danych. Dla przykładu, klasa **loginController.java** podczas inicjalizacji tworzy prywatny obiekt klasy **Database** nazwany **db**, oraz obiekt klasy `FXML.TextField` nazwany **user_nick**, odwołujący się do pola tekstowego będącego elementem okna logowania.

Następnie metoda **new_account()** pozwala na przejście do kolejnego okna poprzez wywołanie metody **db.changeWindow()**, zaś metoda **zaloguj()** pobiera wpisany tekst ze stworzonego wcześniej obiektu pola tekstowego i korzystając z metod klasy **Database**, porównuje go z istniejącymi rekordami.

W celu zapewnienia skalowalności, wszystkie kontrolery korzystają z podobnego schematu, dlatego zamiast opisywać każdy z nich przejdziemy do klasy **Database**.

Database jest sercem aplikacji, gdyż lwią część logiki znajduje się właśnie w tej klasie, która wykorzystywana jest następnie przez wszystkie kontrolery.

Początkowo inicjalizuje ona wszystkie klasy z folder **entity**, będące odpowiednikami tabel w bazie danych, posiadającymi pola odpowiadające kolumnom danych tabel wraz z poprawnymi typami danych.

Następnie inicjalizowana jest sesja z bazą danych, poniżej zaś widzimy serię metod odpowiadających za zarządzanie rekordami w bazie danych. Jako że do zrozumienia działania wszystkich wystarczy omówienie pojedynczej metody, zajmiemy się metodą **add_issue()**:

```

public boolean add_issue( String tytuł, String opis, String typ, String prioritet, String podsumowanie, String ID_Projektu ) {
    String UserID = sample.Session.getInstance().getUserId();

    System.out.println("Dodaje zgłoszenie dla użytkownika "+UserID);

    ProcedureCall call = session.createStoredProcedureCall( s: "Add_Issue");
    call.registerParameter( i: 1, String.class, ParameterMode.IN); // Tytuł
    call.registerParameter( i: 2, String.class, ParameterMode.IN); // Opis
    call.registerParameter( i: 3, String.class, ParameterMode.IN); // Typ
    call.registerParameter( i: 4, String.class, ParameterMode.IN); // Prioritet
    call.registerParameter( i: 5, String.class, ParameterMode.IN); // Podsumowanie
    call.registerParameter( i: 6, String.class, ParameterMode.IN); // ID Projektu
    call.registerParameter( i: 7, String.class, ParameterMode.IN); // ID Użytkownika

    call.setParameter( i: 1, tytuł);
    call.setParameter( i: 2, opis);
    call.setParameter( i: 3, typ);
    call.setParameter( i: 4, prioritet);
    call.setParameter( i: 5, podsumowanie);
    call.setParameter( i: 6, ID_Projektu);
    call.setParameter( i: 7, UserID);

    boolean error = false;

    try {
        call.executeUpdate();
    } catch (Exception e){
        error = true;
        e.getMessage();
        e.getStackTrace();
    }

    return !error;
}

```

Rys. 1.9 - przykładowa metoda działająca z bazą danych

Pierwszym krokiem jest pobranie z aktywnej sesji nazwy użytkownika (głównie w celach logów/debugu), następnie zaś stworzenie obiektu sesji i zarejestrowanie oraz ustawienie odpowiednich parametrów przekazanych przez kontroler danego okna (dodaj_zgłoszenie w tym przypadku). Na końcu zaś wewnątrz bloku try-catch wykonywana jest próba uaktualnienia bazy danych, blok ten wymagany jest, gdyż bez niego w razie wystąpienia błędu z połączeniem doszłoby do całkowitego crashu aplikacji.

Pozostałe metody edytujące inne tabele w bazie danych działają na identycznym schemacie, dopasowanym do wymagań danej tabeli. Poza tymi metodami w klasie **Database** znajduje się jeszcze metoda odpowiedzialna za przechodzenie pomiędzy oknami, działająca podobnie do metody **start()** klasy **Main**.


```
public void changeWindow(ActionEvent event, String resourceName) {  
    Parent parent = null;  
    try {  
        parent = (AnchorPane) FXMLLoader.Load(getClass().getResource(resourceName));  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
  
    assert parent != null;  
    Scene scene = new Scene(parent);  
  
    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
  
    stage.setScene(scene);  
}
```

Rys. 1.10 - metoda przechodzenia między oknami