

2022/2023



Aplikacja do kupowania biletów

PROGRAMOWANIE OBIEKTOWE

SEBASTIAN KOLAŃSKI

LABORATORIUM 2

Nr albumu 122942

Spis treści

1.	Wprowadzenie	2
1.1	Cel oraz ogólny opis aplikacji.....	2
1.2	Wymagania systemowe oraz konfiguracja.....	2
1.3	Baza danych – diagram ERD oraz opis tabel.....	3
1.3.1	Opis tabeli „events”	3
1.3.2	Opis tabeli „users”	4
1.3.3	Opis tabeli „bought”	5
1.4	Diagram przepływu danych	6
2.	Interfejs użytkownika i funkcjonalności	7
2.1	Okno logowania.....	7
2.2	Panel główny	8
2.3	Panel zarządzania bazą danych	9
2.4	Panel do zakupu biletów	10
2.5	Panel użytkownika.....	11
3.	Dokumentacja API	12
3.1	Klasy w programie	12
3.2	Metody w „HelloController”	13
3.3	Metody w „MainPanelController”	14
4.	Bezpieczeństwo, zalety i ograniczenia aplikacji	18
4.1	Bezpieczeństwo	18
4.2	Zalety	18
4.3	Ograniczenia	18

1. Wprowadzenie

1.1 Cel oraz ogólny opis aplikacji

Celem aplikacji o nazwie „NEW EVENT” jest możliwość zakupu biletu na wydarzenia oraz zarządzanie bazą danych, w której są zawarte informacje o nadchodzących wydarzeniach. W przyszłości, chciałbym dodać możliwość komentowania przez użytkowników wydarzeń i dyskutowanie o nich. W aplikacji postawiłem na schludny, prosty w obsłudze interfejs, który będzie zrozumiały i czytelny zarówno dla zwykłego użytkownika jak i dla osoby, która będzie nią zarządzać. Całość kodu źródłowego została zrealizowana w języku JavaFX, z interfejsem JDBC, umożliwiającym połączenie z bazą danych. Do zaprojektowania interfejsu użyłem aplikacji SceneBuilder, która znacznie przyspieszyła ten proces i była przydatna przy ewentualnych modernizacjach. Dodatkowo zaimportowałem bibliotekę JFoenix, która przekłada się jedynie na wygląd aplikacji. Oprócz pobierania informacji z bazy danych na temat wydarzeń, dodałem funkcję umożliwiającą logowanie i rejestrowanie użytkowników z wykorzystaniem połączenia z bazą danych.

1.2 Wymagania systemowe oraz konfiguracja

Żeby uruchomić aplikację będziemy musieli zainstalować na naszym komputerze:

- JAVA 9
- XAMPP – przykładowa aplikacja do zarządzania bazą danych.

Żeby skonfigurować nasz program, będziemy musieli zaimportować dołączoną bazę danych za pomocą programu XAMPP. Używamy przeglądarki, wprowadzamy adres „localhost/phpmyadmin”. Tworzymy nową bazę danych o nazwie „market” , a w zakładce import przekazujemy dołączony plik „market.sql”.

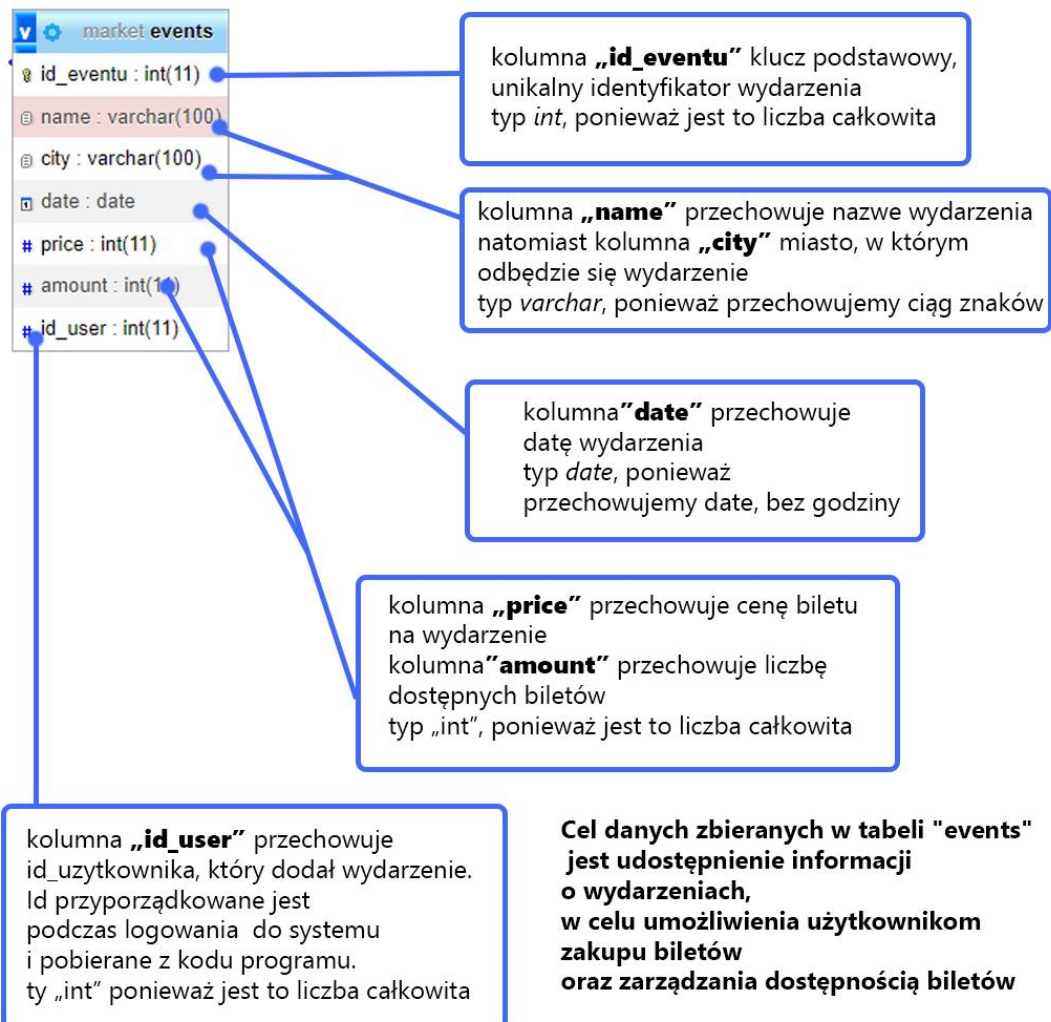
Żeby uruchomić program używamy „New Event.exe”, który znajduje się w głównym katalogu z aplikacją

1.3 Baza danych – diagram ERD oraz opis tabel

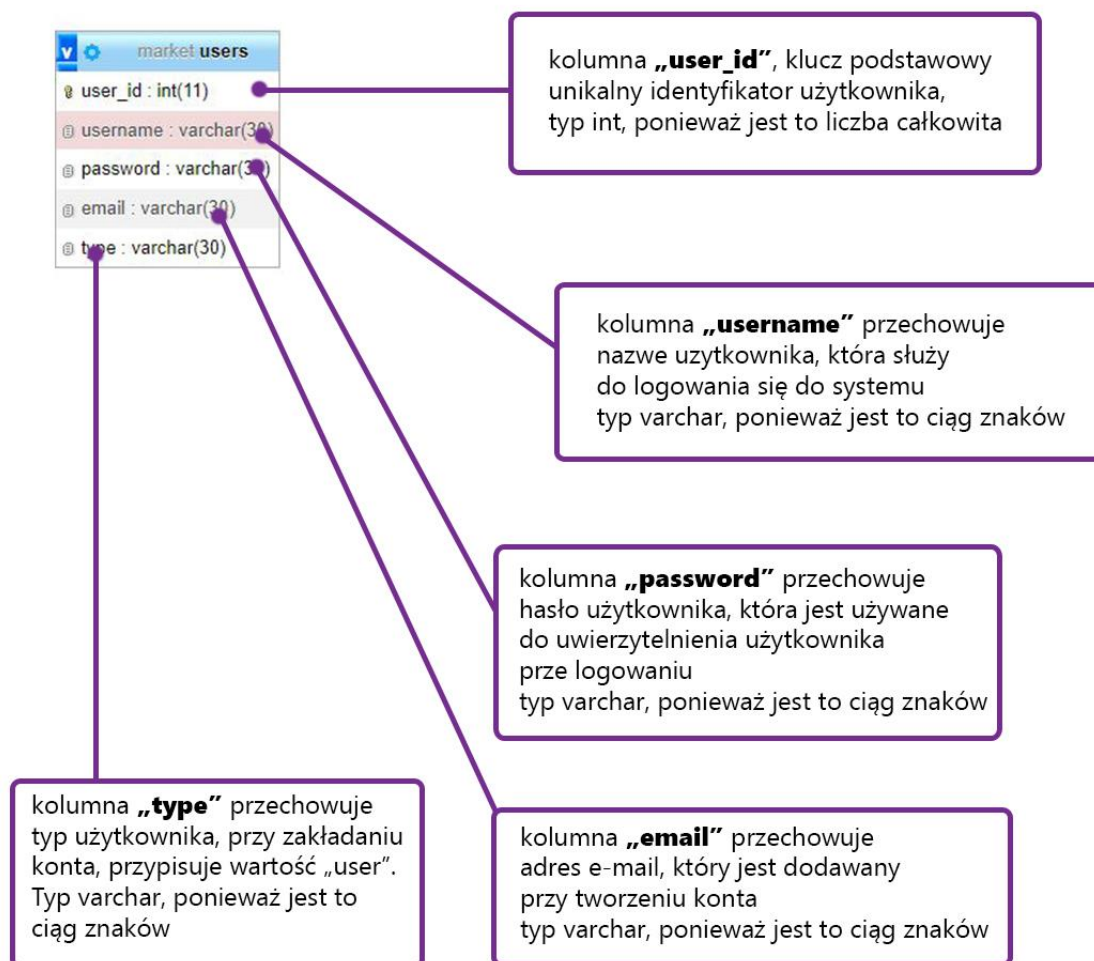
Baza danych składa się z trzech tabel połączonych relacjami.



1.3.1 Opis tabeli „events”

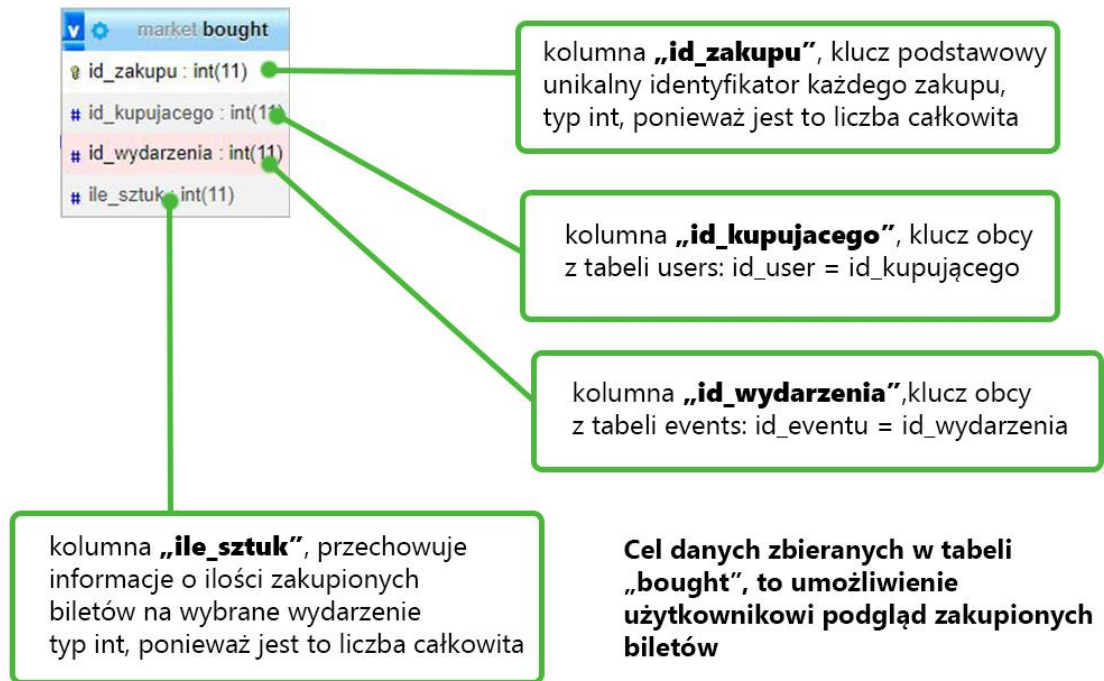


1.3.2 Opis tabeli „users”

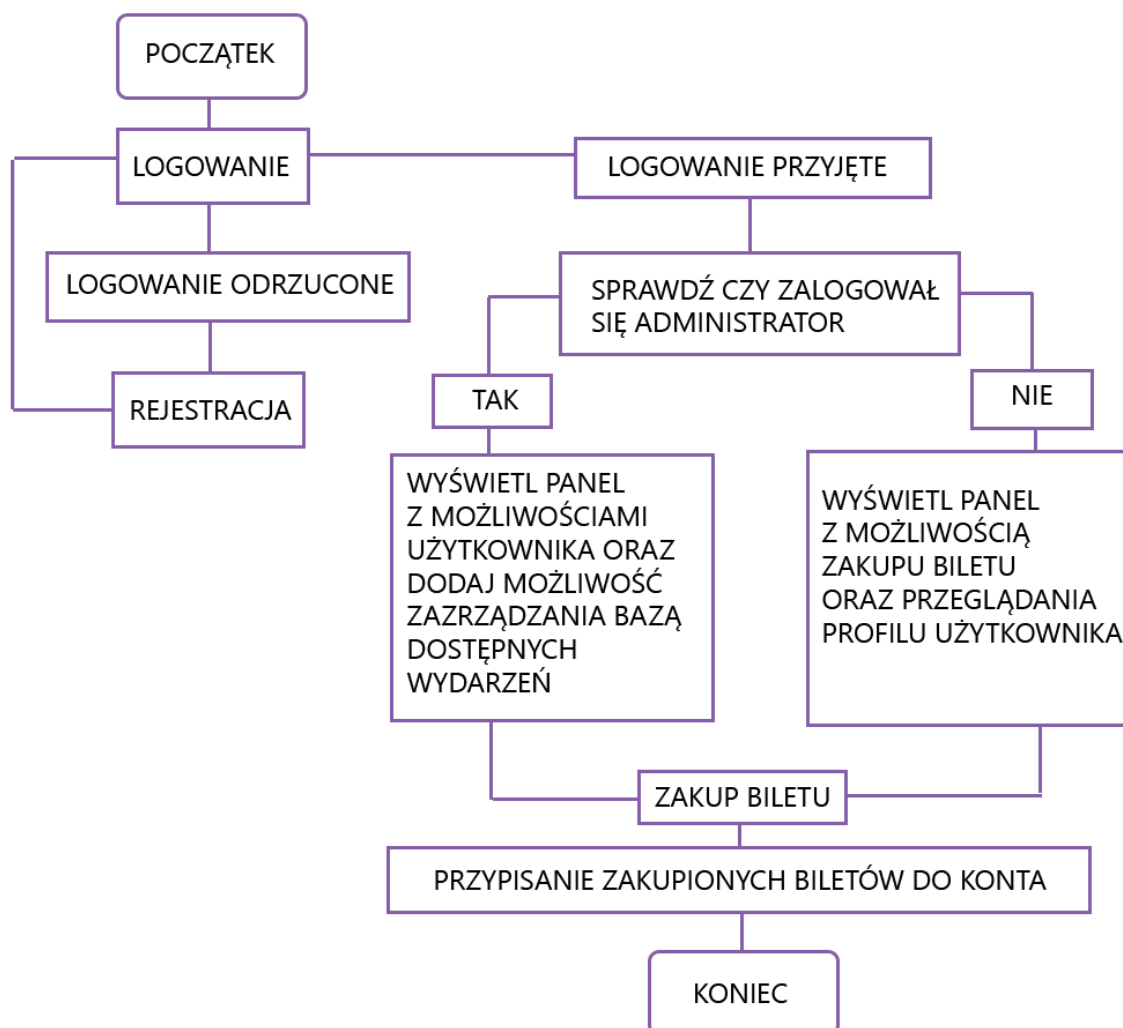


Dane te są zbierane, aby umożliwić użytkownikom logowanie się do systemu, personalizację treści i udostępnianie odpowiednich funkcji zależnie od poziomu dostępu oraz do komunikacji z użytkownikami

1.3.3 Opis tabeli „bought”

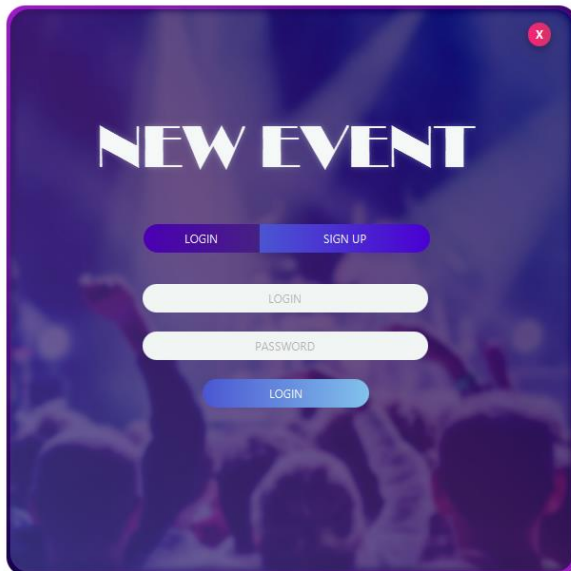


1.4 Diagram przepływu danych

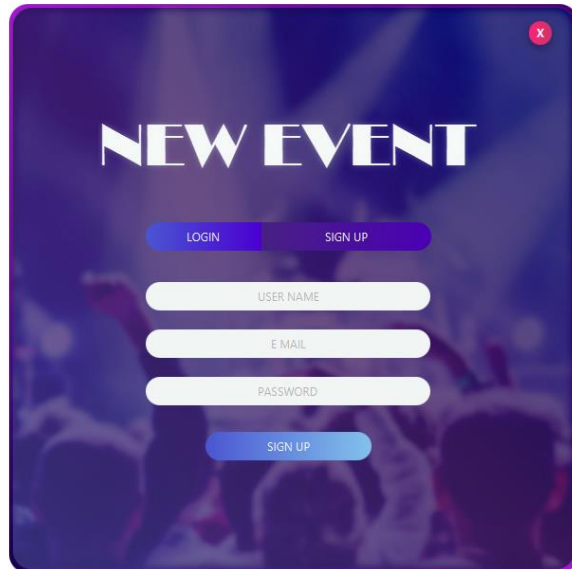


2. Interfejs użytkownika i funkcjonalności

2.1 Okno logowania

The login panel features a dark blue background with a faint image of a crowd. At the top, the text "NEW EVENT" is displayed in white. Below it, there are two tabs: "LOGIN" (active) and "SIGN UP". Under the "LOGIN" tab, there are two white input fields labeled "LOGIN" and "PASSWORD". Below these fields is a blue "LOGIN" button. A red "X" icon is in the top right corner.

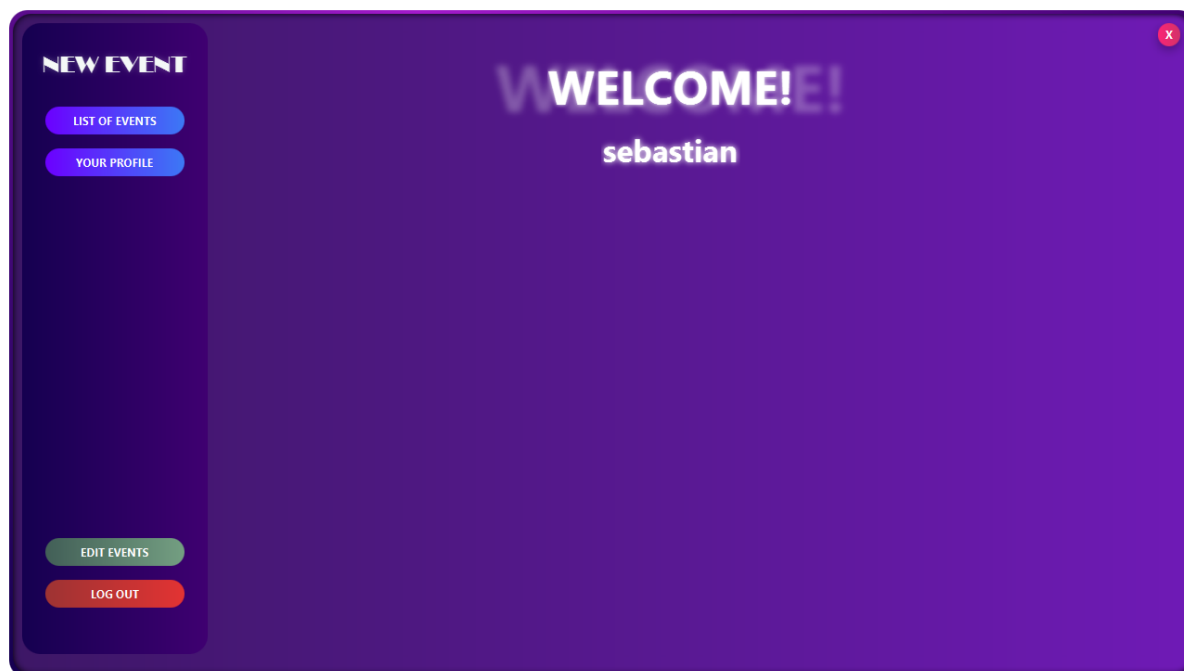
Panel logowania

The registration panel has the same dark blue background and "NEW EVENT" header. It features two tabs: "LOGIN" and "SIGN UP" (active). Under the "SIGN UP" tab, there are three white input fields labeled "USER NAME", "E MAIL", and "PASSWORD". Below these fields is a blue "SIGN UP" button. A red "X" icon is in the top right corner.

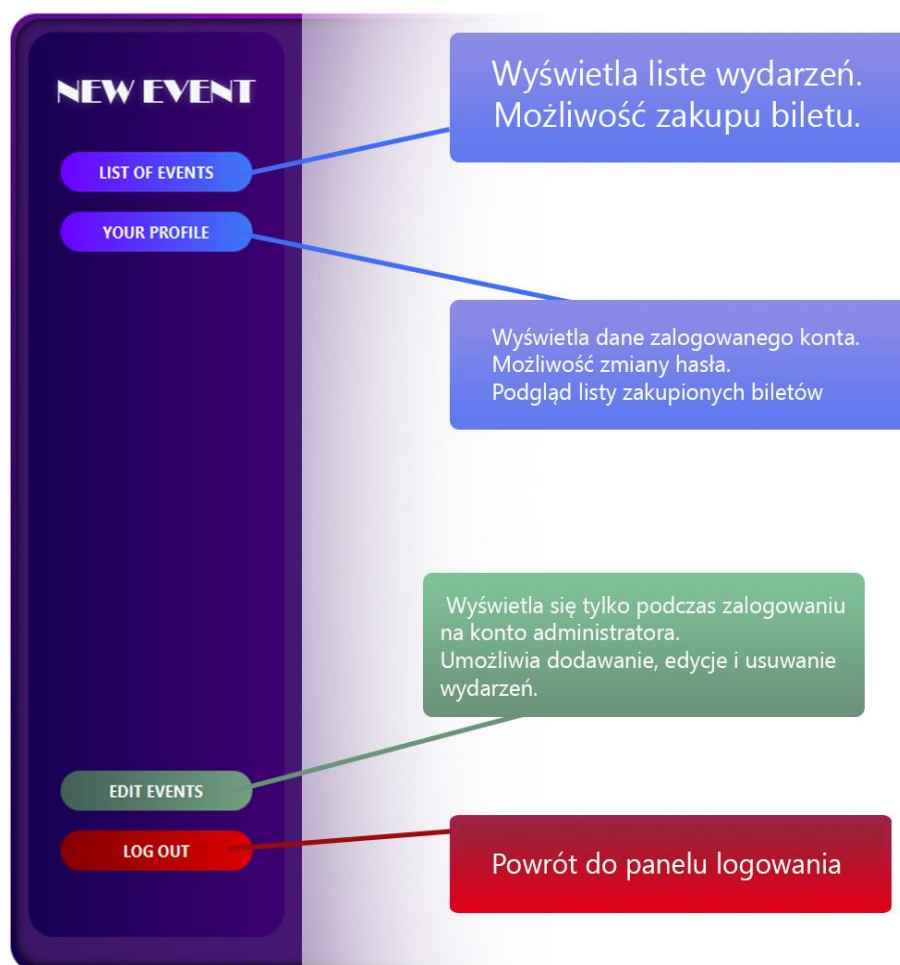
Panel rejestracji

Program po uruchomieniu pokazuje ekran logowania. Żeby zalogować się musimy podać nasze dane logowania(tj. login i hasło), z wcześniej utworzonego konta. Jeśli jednak jeszcze tego nie zrobiliśmy, przechodzimy do ekranu rejestracji, w którym po podaniu nazwy użytkownika, e-mailu oraz hasła stworzymy nowe konto.

2.2 Panel główny



Po zalogowaniu, program zmienia scenę na ekran główny, w którym występują funkcje



2.3 Panel zarządzania bazą danych

The screenshot shows the 'NEW EVENT' application interface. On the left is the 'ADMIN PANEL' with input fields for ID (for edit only), NAME, CITY, DATE, PRICE, and AMOUNT, and buttons for ADD, UPDATE, DELETE, REFRESH, and BACK. On the right is a table of events with columns: ID, NAME, CITY, DATE, PRICE, AMOUNT, and ADDED BY. The table contains 17 rows of data.

ID	NAME	CITY	DATE	PRICE	AMOUNT	ADDED BY
23	Metallica	Warszawa	2026-02-27	300	5	admin
24	Eminem	Kraków	2023-02-24	400	50	admin
25	Sanah	Gdynia	2023-03-04	500	10	admin
26	Metallica	Warszawa	2026-02-27	300	5	admin
27	Eminem	Kraków	2023-02-24	400	50	admin
28	Sanah	Gdynia	2023-03-04	500	10	admin
29	Metallica	Warszawa	2026-02-27	300	0	admin
30	Eminem	Kraków	2023-02-24	400	50	admin
31	Sanah	Gdynia	2023-03-04	500	10	admin
32	Metallica	Warszawa	2026-02-27	300	5	admin
33	Eminem	Kraków	2023-02-24	400	50	admin
34	Sanah	Gdynia	2023-03-04	500	10	admin
35	Metallica	Warszawa	2026-02-27	300	5	admin
36	Eminem	Kraków	2023-02-24	400	50	admin
37	Sanah	Gdynia	2023-03-04	500	10	admin
38	Metallica	Warszawa	2026-02-27	300	5	admin
39	Eminem	Kraków	2023-02-24	400	50	admin
40	Sanah	Gdynia	2023-03-04	500	10	admin
41	Metallica	Warszawa	2026-02-27	300	5	admin
42	Eminem	Kraków	2023-02-24	400	50	admin

Po kliknięciu „EDIT EVENTS” uruchamia się nam scena, w której wyświetla się panel zarządzania oraz lista wydarzeń wpisanych do bazy danych. Oprócz wyszukiwania funkcjami które ta scena obsługuje jest

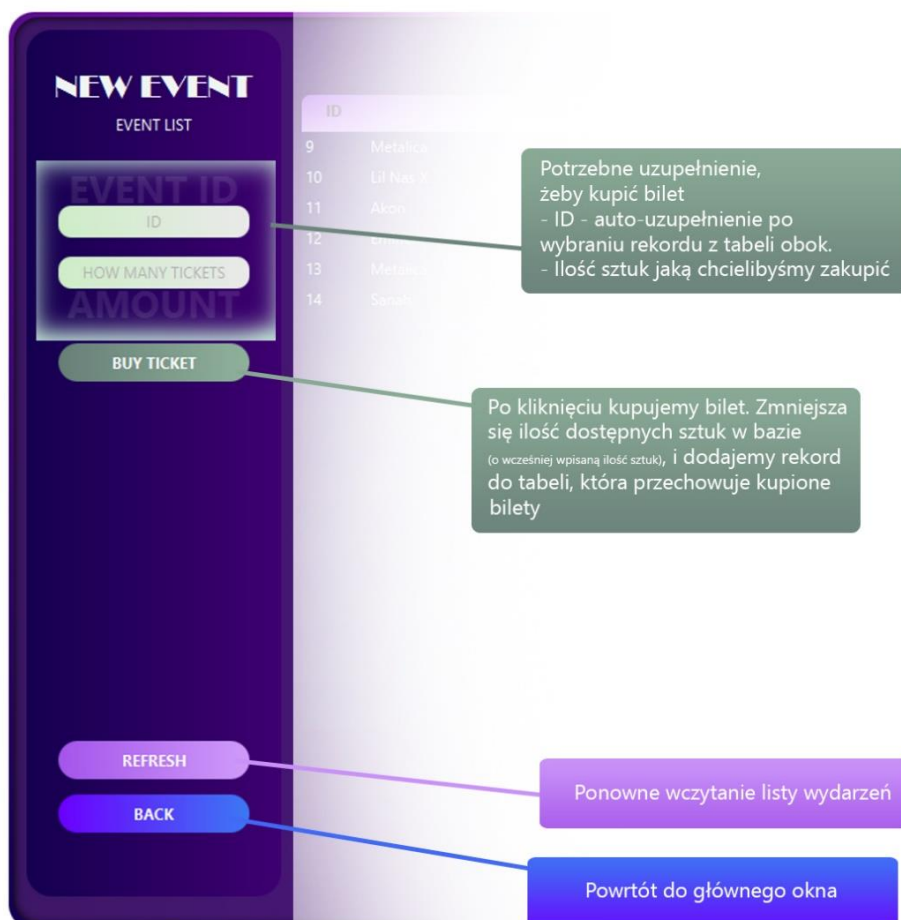
The diagram shows the 'NEW EVENT' application interface with callouts explaining the functions of the buttons:

- ID (for edit only)**: Pola służące do dodawania oraz edytowania wartości z tabeli. Auto-upełnienie po wybraniu rekordu
- ADD**: Dodaje rekord do tabeli, jeśli wszystkie dane z formularza zostały wypełnione poprawnie
- UPDATE**: Edytuje rekord, jeśli wszystkie dane z formularza zostały wypełnione poprawnie. (Przydatne przed edytowaniem, użycie auto-upełnienia po wybraniu rekordu)
- DELETE**: Usuwa wybrany rekord z tabeli.
- REFRESH**: Ponowne wczytanie listy wydarzeń
- BACK**: Powrót do głównego okna

2.4 Panel do zakupu biletów



Wcześniej dodane wydarzenia, możemy przeglądać również jako zwykły użytkownik. Służy do tego przycisk „LIST OF EVENTS”, który przenosi nas do okna z listą eventów. Oprócz wyszukiwania w tym oknie dostępnymi funkcjami są:



2.5 Panel użytkownika

NEW EVENT
YOUR PROFILE

CHANGE PASSWORD

YOUR USERNAME: **admin**
E-MAIL: **s.kolanski0@gmail.com**
TYPE: **admin**

PURCHASED TICKETS

NAME	CITY	DATE	AMOUNT
Metallica	Warszawa	2026-02-27	5

REFRESH
BACK

NEW EVENT
YOUR PROFILE

CHANGE PASSWORD

PO KLIKNIECIU

NEW EVENT
YOUR PROFILE

CHANGE PASSWORD

NEW PASSWORD

ACCEPT

YOUR USERNAME: **hej**
E-MAIL: **hej**
TYPE: **admin**

PURCHASED TICKETS

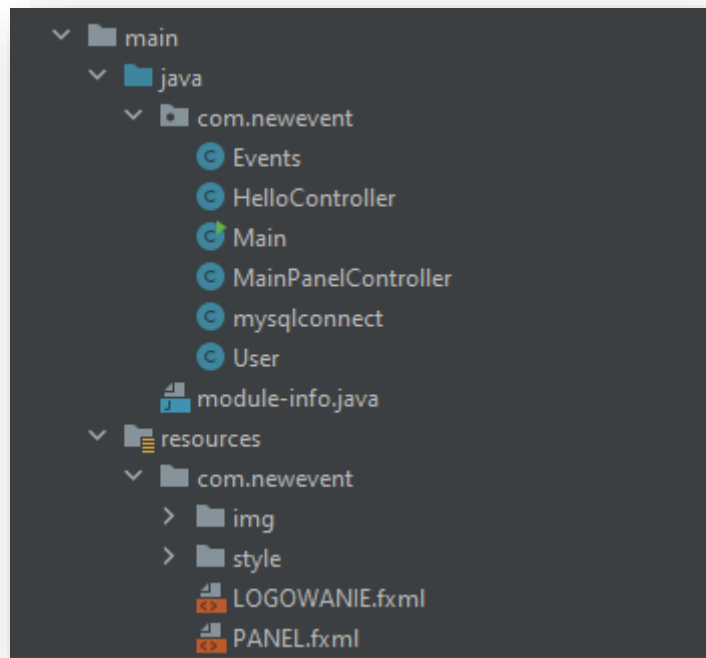
NAME	CITY	DATE	AMOUNT
Sanah	Kraków	2023-05-18	1

Wyświetla dane zalogowanego użytkownika oraz informacje o naszych biletach, takie jak ilość zakupionych sztuk.

Po wprowadzeniu nowego hasła oraz kliknięciu „ACCEPT” aktualizujemy hasło naszego konta.

3. Dokumentacja API

3.1 Klasy w programie



- Klasa **Events** jest skonstruowana na potrzeby przekazywania informacji o wydarzeniach.
- Klasa **HelloController** jest skonstruowana na potrzeby obsługi zdarzeń okna logowania, tj. logowanie do aplikacji i rejestracja użytkownika, przez dodanie informacji o nim do bazy danych.
- Klasa **Main** obsługuje uruchomienie sceny okna logowania.
- Klasa **MainPanelController** jest skonstruowana na potrzeby obsługi sceny po poprawnym zalogowaniu przez użytkownika.
- Klasa **mysqlconnect** jest skonstruowana na potrzeby obsługi zdarzenia połączenia programu z bazą danych.
- Klasa **User** nie ma zbyt wielu funkcji, potrzebna jedynie do zwrócenia nazwy użytkownika przy zmianie sceny.

3.2 Metody w „HelloController”

```
1 usage
public void LoginpaneShow(){...}

1 usage
public void Signuppaneshow(){...}

2 usages
private static String your_login;

2 usages
public static String getUsername(){...}

1 usage
@FXML
private void Login (ActionEvent event) throws Exception{...}

1 usage
public void add_users(ActionEvent event){...}

1 usage
@FXML
private void handleclose(ActionEvent event) { System.exit( status: 0); }
```

- Metoda **Loginpaneshow()** oraz **Signuppaneshow()** umożliwia przełączenie między oknem logowania a oknem rejestracji
- Metoda **getUsername()** zwraca login
- Metoda **Login()** po przekazaniu kliknięcia, w guzik, który jest na ekranie logowania, wykonuje się procedura: łączenie z bazą, sprawdzanie podanych loginów, a jeśli wszystko się zgadza, przełączanie na okno główne.
- Metoda **add_users()** po przekazaniu kliknięcia, w guzik, który jest na ekranie rejestracji, wykonuje się procedura: łączenie z bazą, dodanie użytkownika do bazy, jeśli dane się zgadzają, po sukcesie tej operacji przechodzimy na okno logowania.

3.3 Metody w „MainPanelController”

```
//////////////////////////////////// do panelu głównego

2 usages
public void Paneadminshow() throws SQLException {...}

3 usages
public void Panewelcomeshow() {...}

2 usages
public void Paneeventshow(){...}

2 usages
public void Paneuserprofileshow(){...}
```

- Metody te przełączają widoki okien. Ponieważ cały ekran startowy obsługuje jeden plik FXML.

```
//////////////////////////////////// for all

2 usages
public void logout() throws IOException {...}

4 usages
@FXML
private void handleclose(ActionEvent event) { System.exit( status: 0); }
```

- Metoda **logout()** zmienia okno na ekran logowania.
- Metoda **handleclose()** służy do obsługi przycisku zamknięcia programu.

```
//////////////////////////////////// do panelu z profilem użytkownika
```

```
1 usage
```

```
public ObservableList<Bought> getDataBought(){...}
```

```
2 usages
```

```
public void Panechangepasswordshow() {...}
```

```
2 usages
```

```
public void Panechangepasswordhide() {...}
```

```
1 usage
```

```
public void ChangePassword(){...}
```

```
1 usage
```

```
public void showPurchasedTickets(){...}
```

- Metoda **getDataBought()** łączy się z bazą danych, wybiera odpowiednie rekordy za pomocą zapytania SELECT oraz wstawia je do ObservableList.
- Metoda **Panechangepasswordshow()** oraz **Panechangepasswordhide()** umożliwia przełączanie się widoku (pokazane w ostatnim zdjęciu przy opisach interfejsu).
- Metoda **ChangePassword()** aktualizuje bazę danych po zmianie hasła przez użytkownika.
- Metoda **showPurchasedTickets()** uzupełnia listę w oknie profilu użytkownika.


```

//////////////////////////////////// do panelu admina

4 usages
public void showEvents() throws SQLException {...}
1 usage
public void addEvents() {...}
2 usages
@FXML
void getSelected(MouseEvent event) {...}
1 usage
public void Edit() {...}
1 usage
public void Delete() {...}

```

Metody te służą do zarządzania panelem administratora.

- **showEvents()** pokazuje listę wydarzeń.
- **addEvents()** dodaje wydarzenia po uzupełnieniu formularza i kliknięciu przycisku add,
- **getSelected()** służy do auto-uzupełniania po wybraniu rekordu z listy,
- **Edit()** obsługuje aktualizację bazy po kliknięciu przycisku update,
- **Delete()**, usuwa zaznaczony rekord z bazy danych.

```

//////////////////////////////////// do panelu z kupowaniem biletu

2 usages
public void showEventsForUser() {...}
1 usage
public void getSelectedtobuy(){...}
1 usage
public void buyTicket(){...}

```

Metody te służą do pokazywania listy dostępnych wydarzeń i zakupu biletu.

- **showEventsForUser()**, wyświetla listę z wydarzeniami
- **getSelectedtobuy()**, przekazuje ID w pole formularza, po wybraniu rekordu z listy
- **buyTicket()**, aktualizuje bazę danych po zakupie biletu.

```

@FXML
public void searchEvents(int choice) {...}

1 usage
public void validationAdd() { validationAdminPane( choice: 1); }

1 usage
public void validationUpdate() { validationAdminPane( choice: 2); }

2 usages
public void validationAdminPane(int choice){...}

```

- **searchEvents()** – wyszukuje wydarzeń po nazwie lub mieście
- **validationAdd()** oraz **validationUpdate()** korzysta z **validationAdminPane()** który ustala poprawność wprowadzonych danych podczas dodawania/edycji wydarzeń

```

////////////////////////// przy włączaniu

6 usages
int userid;
4 usages
String rola;
2 usages
String mail;
4 usages
String currentUsername;

@FXML
private void initialize(){...}
}

```

Po włączeniu okna wykonuje się **initialize**, które wyświetla nazwe użytkownika na ekranie głównym, oraz uzupełnia zmienne, które znajdują się powyżej metody.

4. Bezpieczeństwo, zalety i ograniczenia aplikacji

4.1 Bezpieczeństwo

Dla bezpieczeństwa aplikacji, został wykonany system uwierzytelniania użytkowników za pomocą hasła oraz wprowadzenie funkcji użytkownika i administratora, żeby zwykły użytkownik nie mógł ingerować w zarządzanie bazą danych.

4.2 Zalety

Zaletą aplikacji na pewno jest to, że nie powinna sprawiać kłopotu w obsłudze dla zwykłego użytkownika oraz w prosty sposób możemy zarządzać nią jako administrator.

4.3 Ograniczenia

Ograniczeniami aplikacji z pewnością jest brak znacznej rozbudowy, jednakże w przyszłości chciałbym ją rozbudować dodając wszelakie funkcje.