Sprawozdanie
Wydarzenia Rozrywkowe

Wykonał: Krzysztof Cięgotura i Jakub Kowalski

Opis zrealizowanych prac:

Projekt

Naszym projektem było zrobienie aplikacji desktopowej do zarządzania wydarzeniami rozrywki. Projekt został zrobiony przy użyciu narzędzia visual studio 2022 w szkielecie projektu WPF w języku C#. Aplikacja jest przeznaczona zarówno dla użytkowników, którzy przeglądają aktualne wydarzenia rozrywkowe jak i zarówno dla administratorów, którzy zarządzają danymi wydarzeniami.

Implementacja

Projekt wymagał od nas stworzenia dwóch baz danych, pierwsza do trzymania informacji na temat pracowników, którzy będą nadzorować dane wydarzenie, a druga baza jest przeznaczona do trzymania informacji na temat owych wydarzeń, na których cała aplikacja się opiera. Pierw aplikacja miała sztywną grafikę by tylko sprawdzać, że wszystko działa i nie obyło się pierw bez błędów które w porę rozpoznaliśmy i naprawiliśmy. Poprzez testy użyteczności dodaliśmy też skróty klawiszowe do wszystkich stron by obsługiwały takie intuicyjne przyciski jak escape, backspace, delete, tabulator oraz enter. Wprowadzono różne zabezpieczenia, do akcji które by mógłby przerwać prace użytkownika jak wchodzenie na inną stronę, kiedy są wpisywane informacje do formularza. Zabezpieczenia powstały też by dane wprowadzane do formularzów nie były błędne i wyskakują też komunikaty jak się będzie przechodziło dalej to aplikacja zwróci błąd z informacją jak ten błąd naprawić. Powstały komunikaty, gdy niektóre zmiany by nieomyłkowo nie usunąć czegoś z baz danych. Musieliśmy jakoś odseparować gości od administratorów to zrobiliśmy wbudowane konto, przez które trzeba się zalogować w oknie. By móc się zalogować trzeba wpisać login i hasło, które mają zawartość admin i admin.

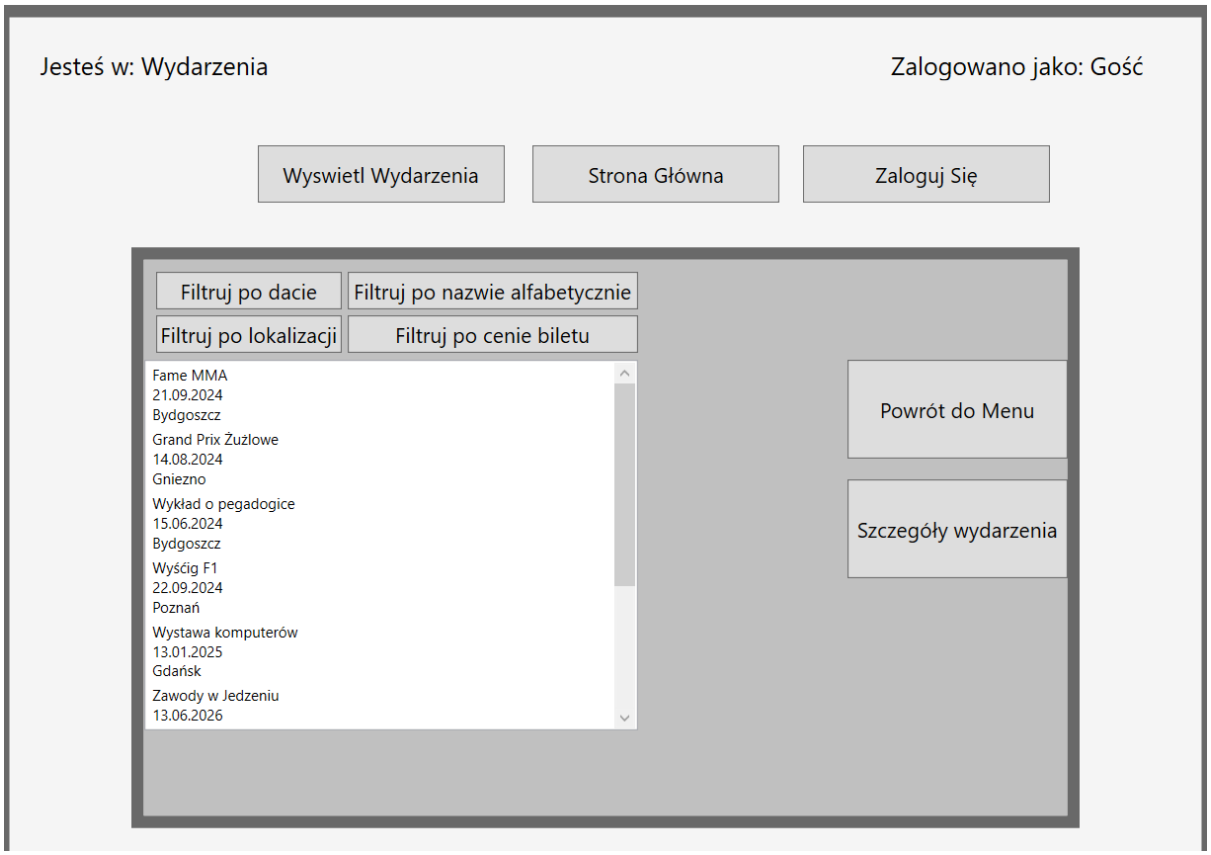
Testy

Aplikacja była stale testowana pod względem czy dany moduł działa prawidłowo. Szczególnie to były testy jednostkowe. Za każdym razem, gdy kończyliśmy modyfikować

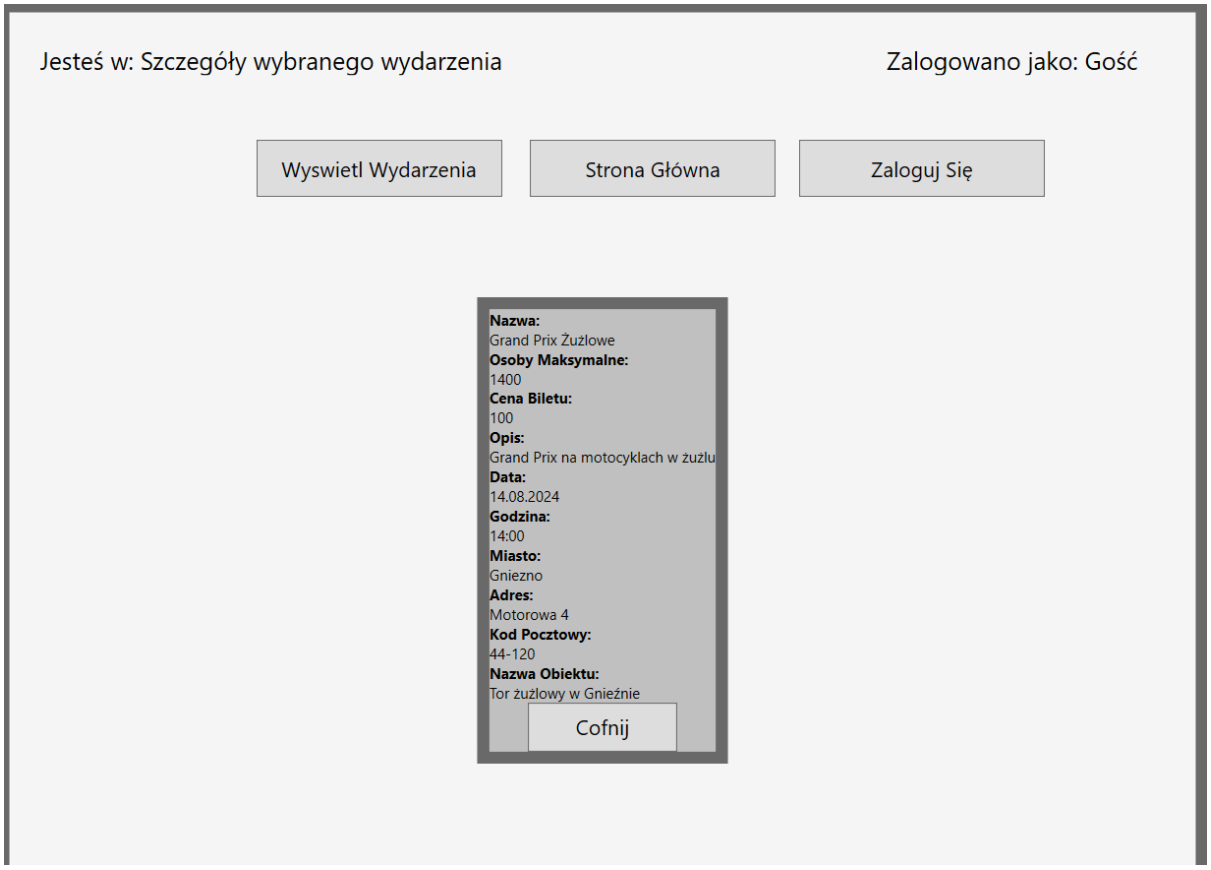
jeden z modułów to włączaliśmy aplikacje i korzystaliśmy z niej w taki sposób jaki użytkownik najprawdopodobniej by skorzystał.

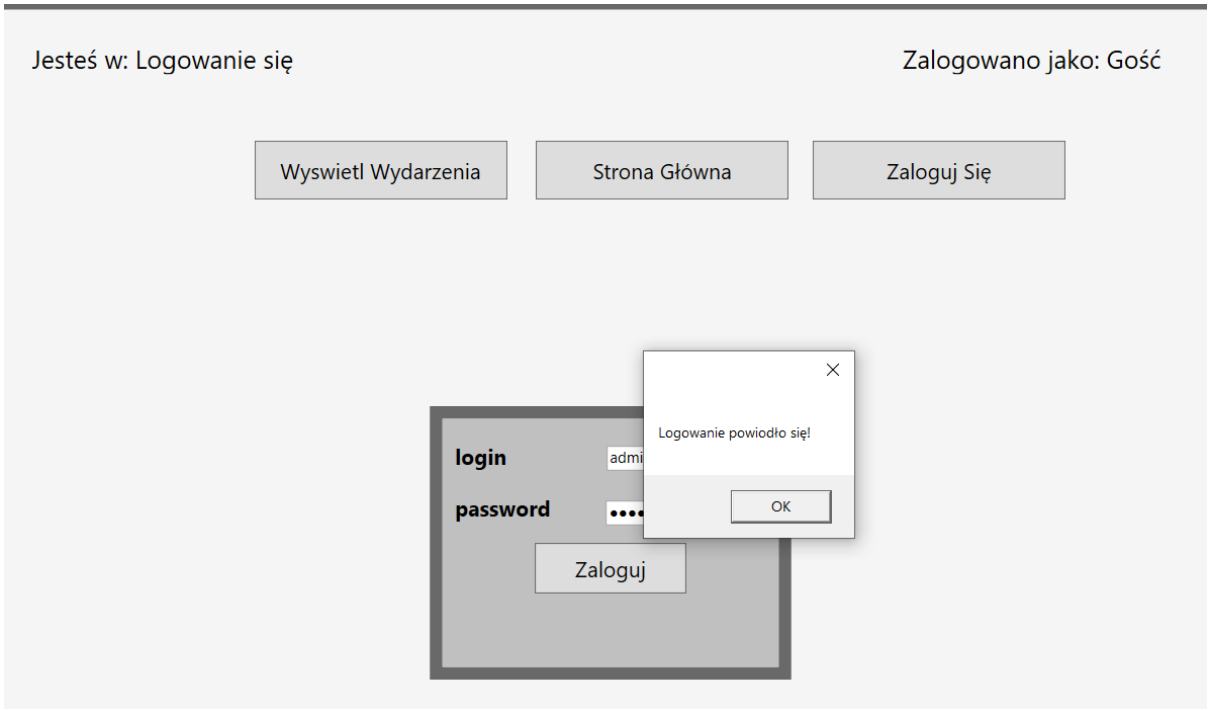
Przedstawienie funkcji aplikacji

Wyświetlanie listy wydarzeń z poziomu gościa

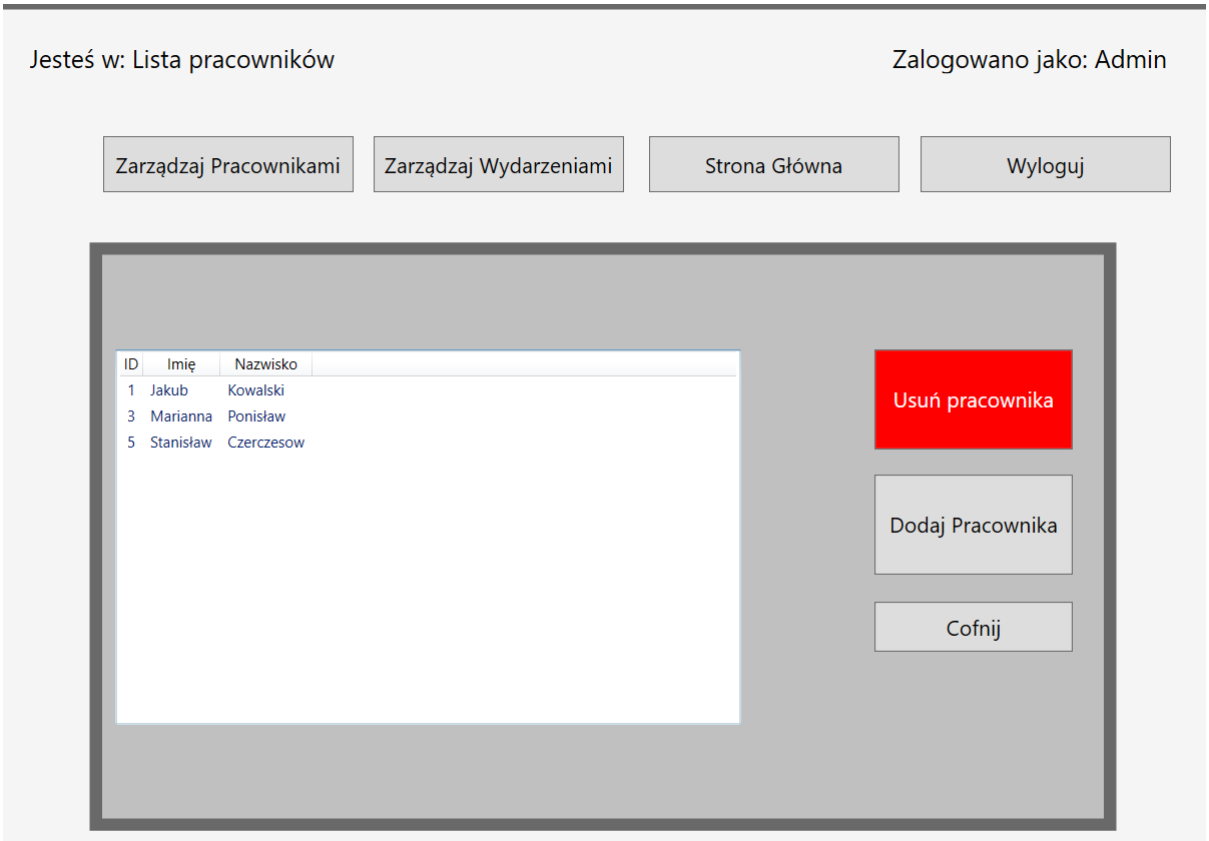


Wyświetlanie szczegółów poprzez podwójnie kliknięcie albo wciśnięcie w przycisk ”szczegóły wydarzenia”. Można też sortować listę wydarzeń poprzez nazwę wydarzenia, po dacie, po lokalizacji jak i po cenie biletu którego na pierwszy rzut oka nie widać, ponieważ jest ta informacja schowana w szczegółach by nie zrazić przyszłego uczestnika wydarzenia.

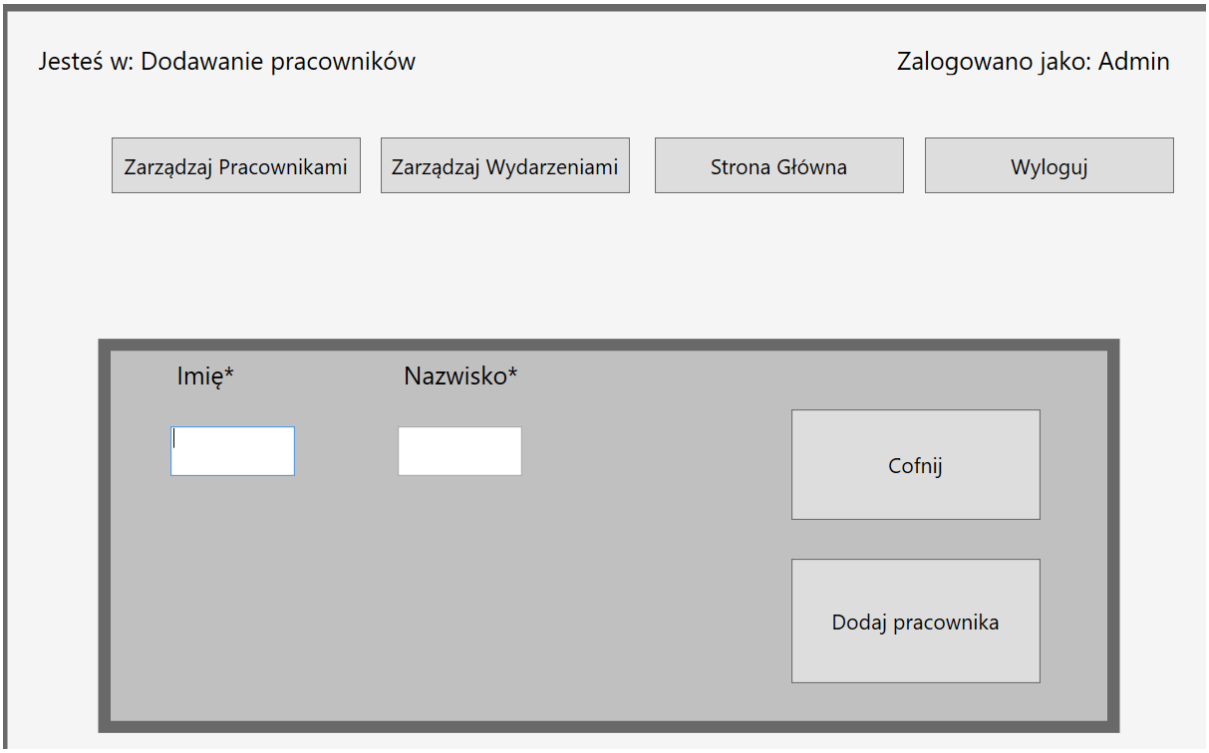




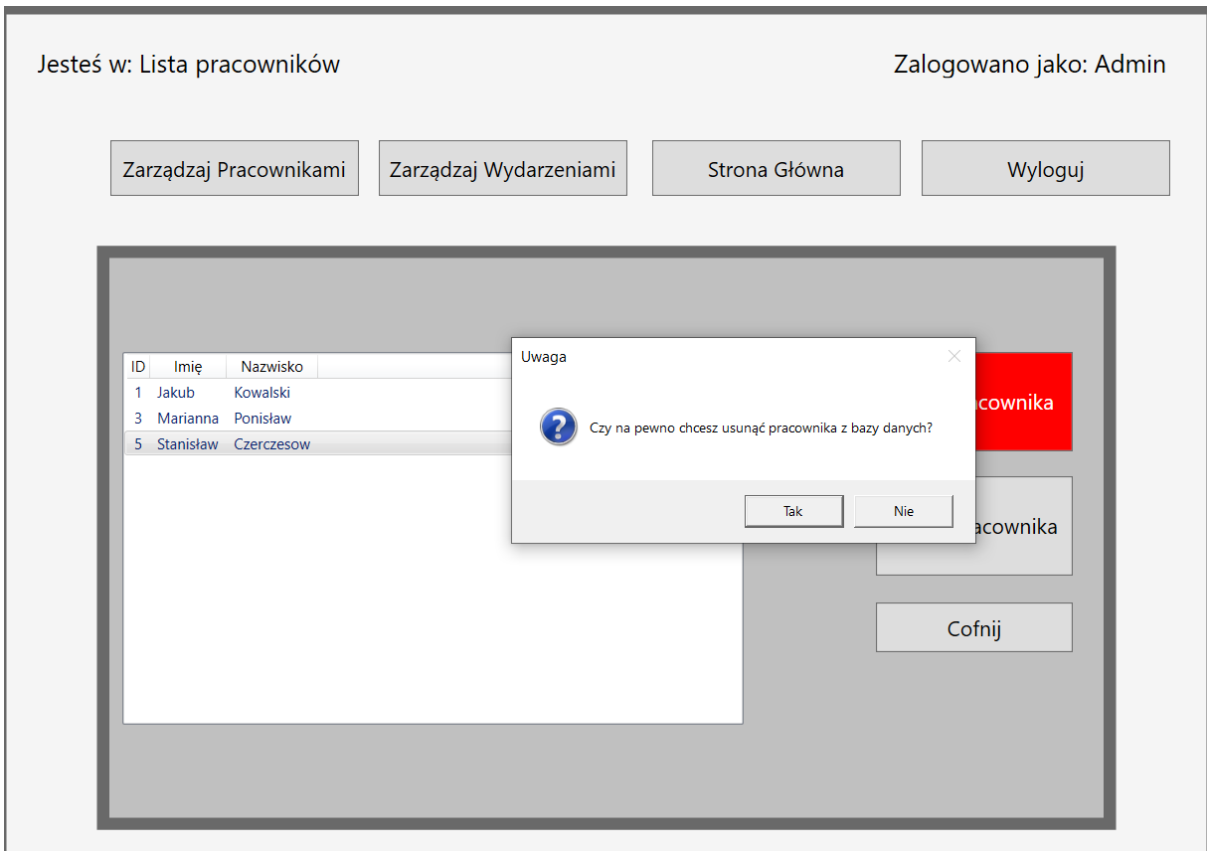
Logowanie się.



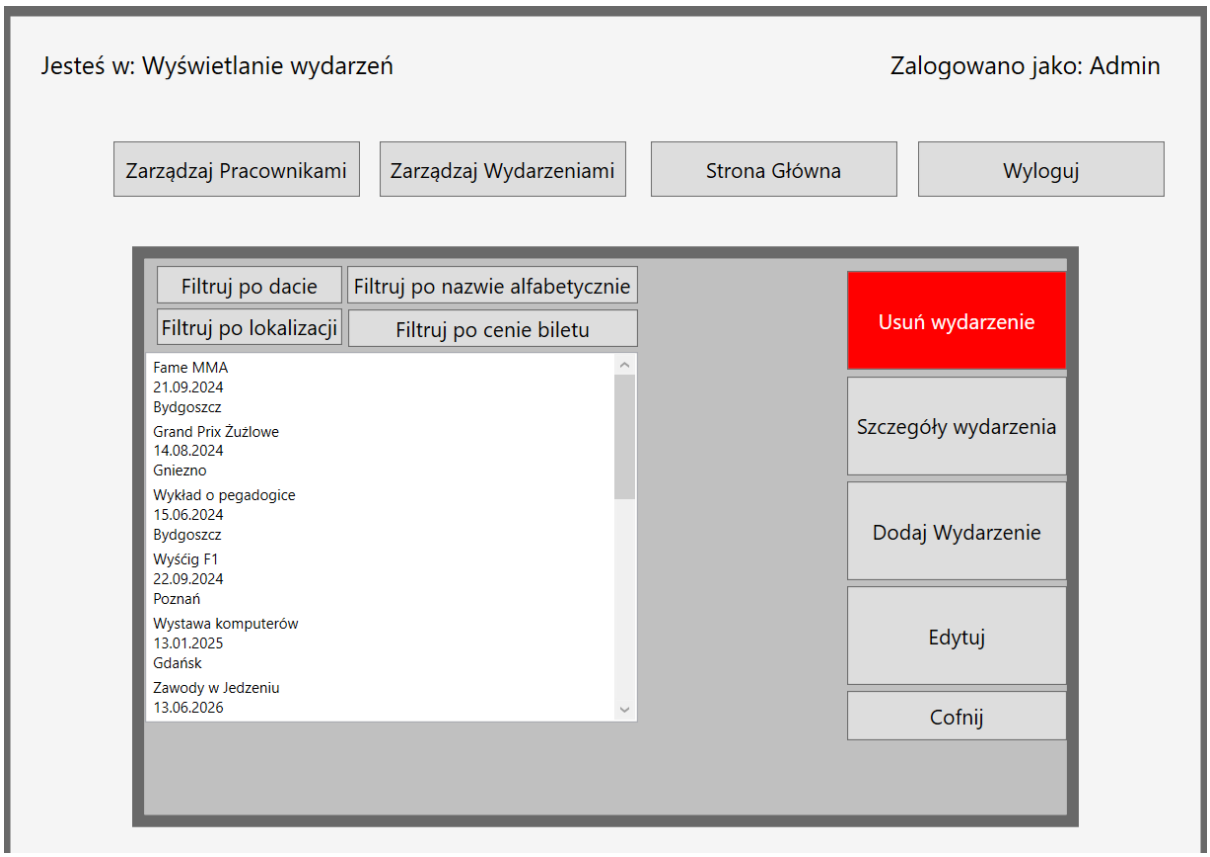
W zarządzaniu pracowników można usunąć pracownika lub dodać go.



Tak wygląda okno do dodawania pracownika.

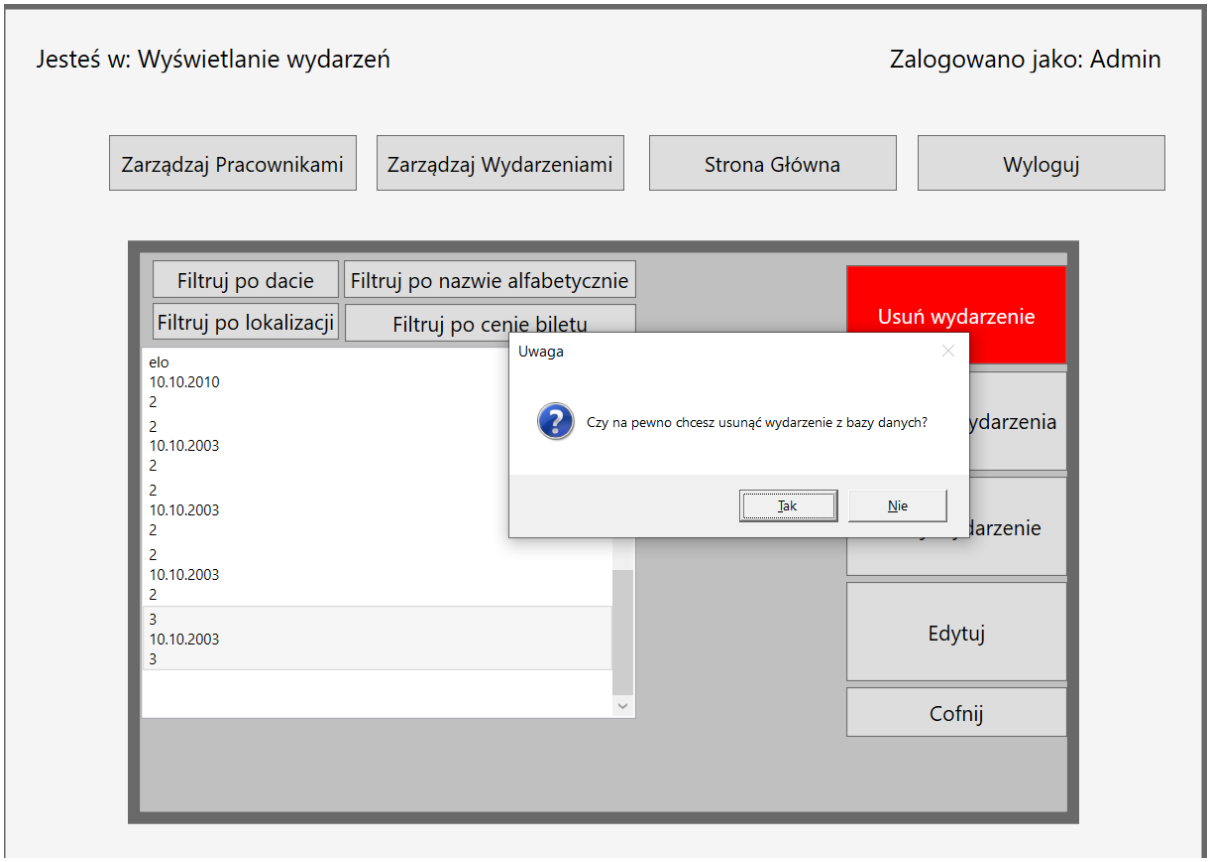


A tak wygląda powiadomienie, które mówi nam czy aby na pewno chcemy usunąć z bazy danych pracownika

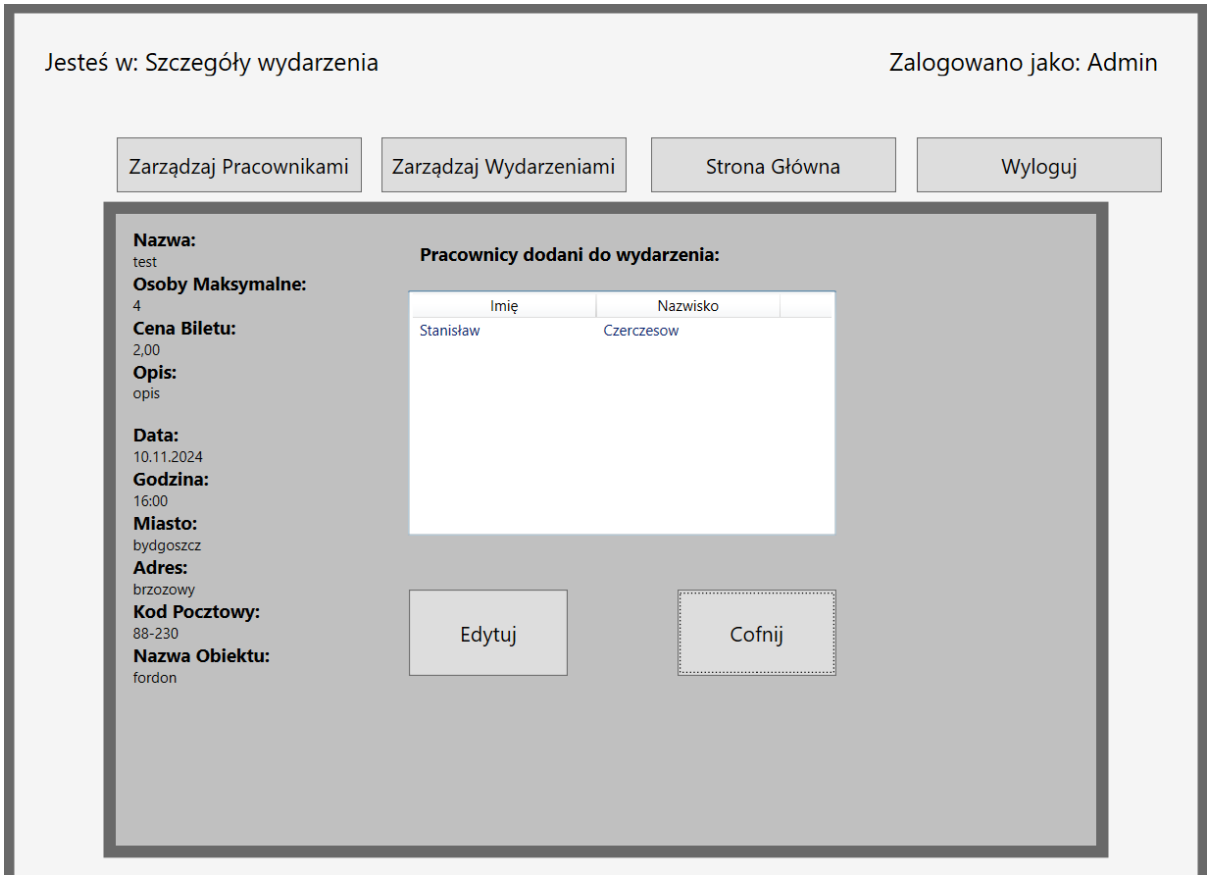


Tak wygląda zarządzanie wydarzeniami. Można tutaj sortować wydarzenia, dodawać,

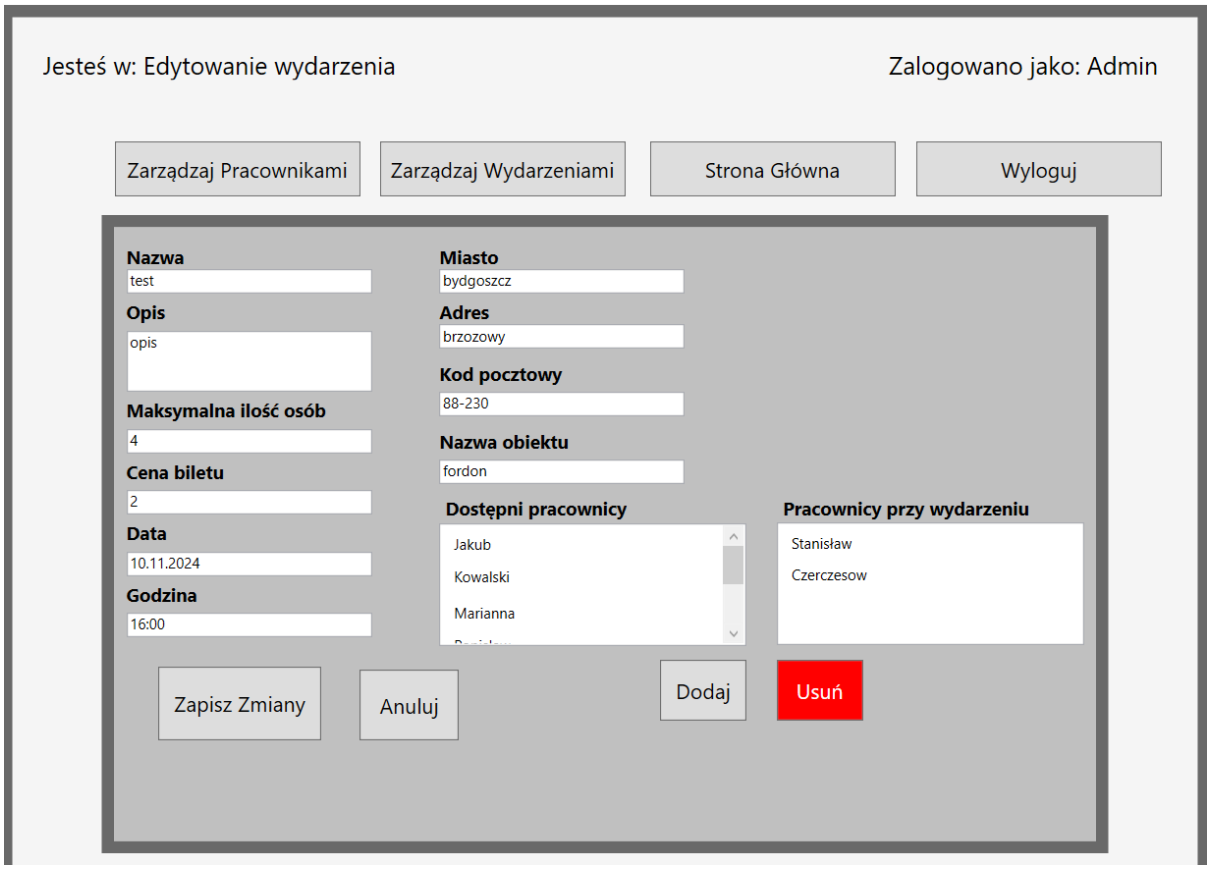
usuwać jak i edytować. Poprzez naciśnięcie klawisza backspace to usuwa się wraz z powiadomieniem wydarzenie



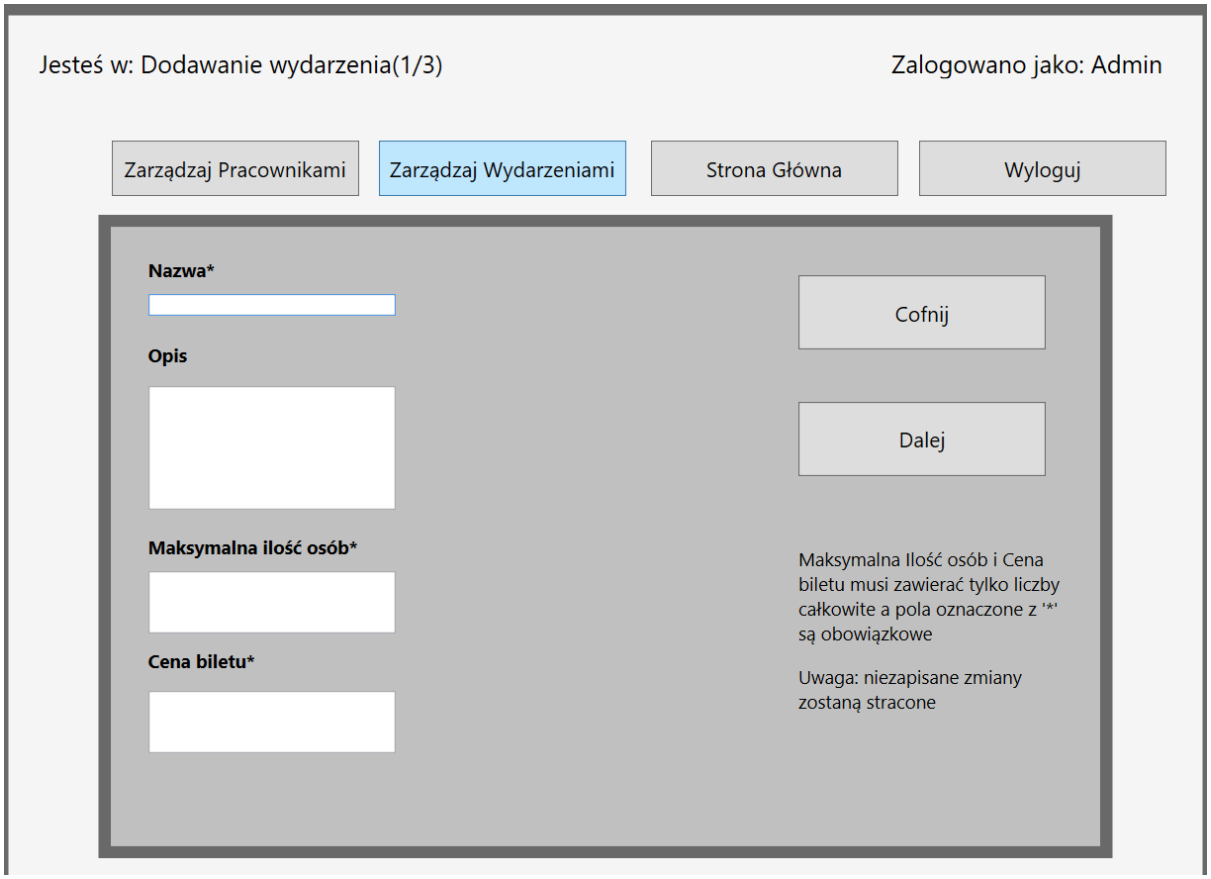
Powiadomienie jak będzie chciało się usunąć wydarzenie.



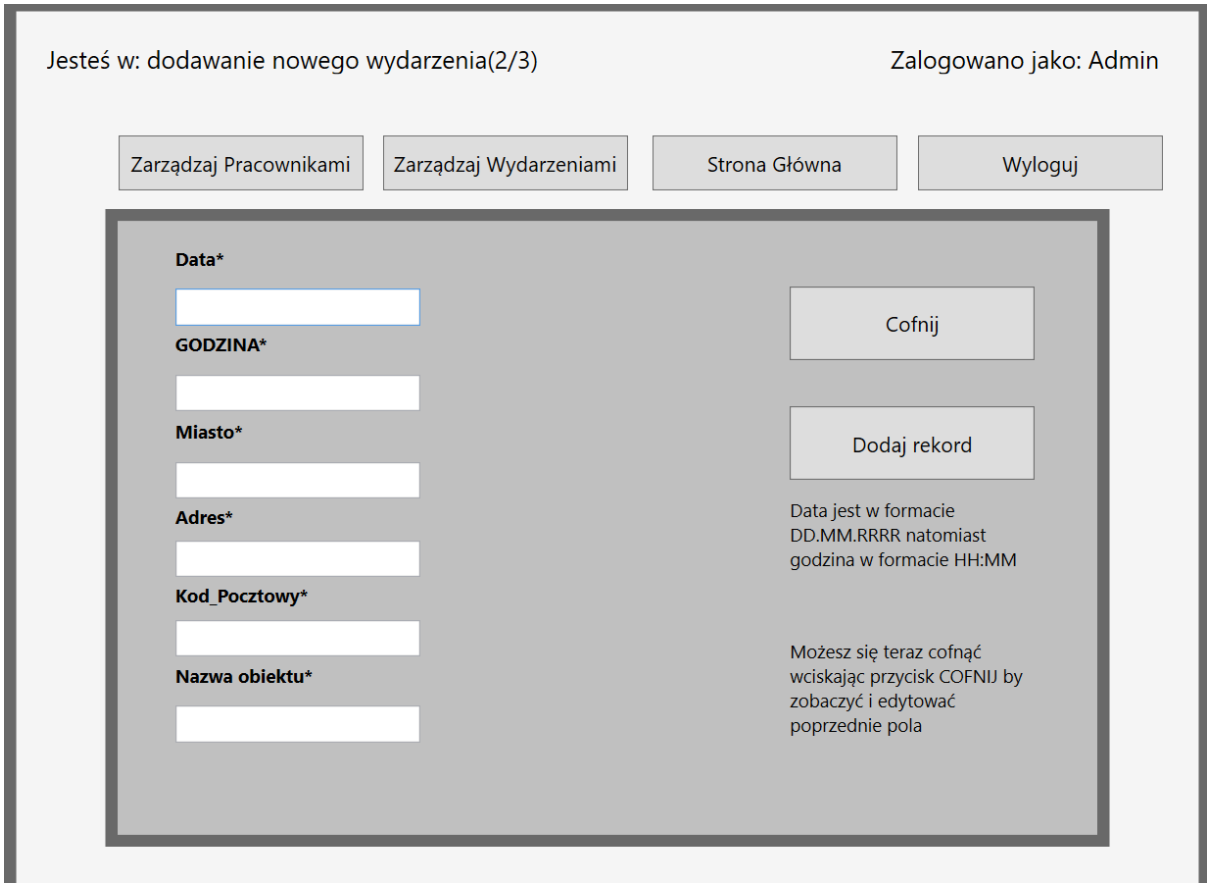
To jest strona, która się wyświetla po wciśnięciu danego wydarzenia dwa razy albo wciśnięciu przycisku szczegółów. Gdy wciśniemy ”edytuj” to przeniesie nas strony, w której można edytować. Kliknięcie escape albo przycisku ”cofnij” cofnie nas do wyświetlaniu wydarzeń.



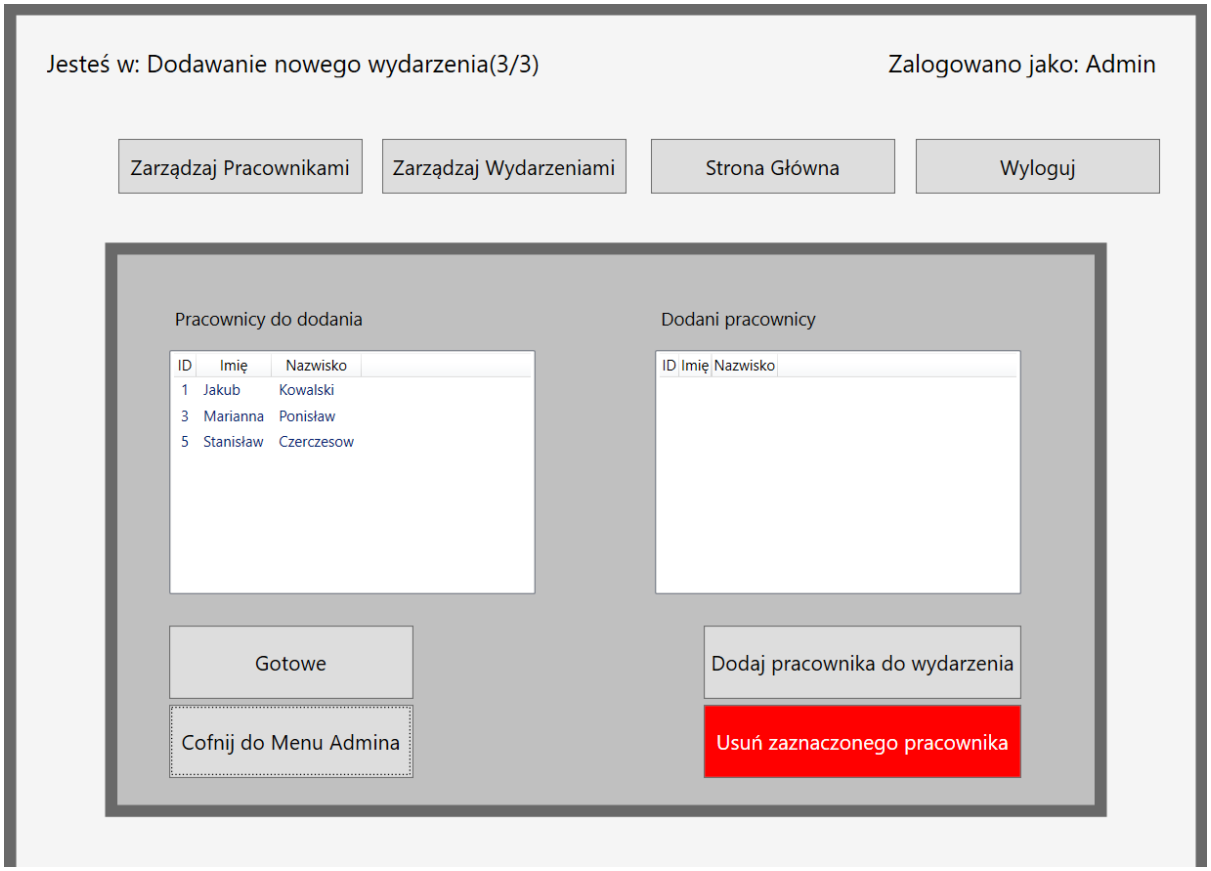
Strona w której można edytować parametry wydarzenia jak usuwać i dodawać pracowników do obsługi. Kliknięcie podwójnie w pracowników powoduje dodanie i usunięcie ich z danego wydarzenia, to też można zrobi przyciskami ”dodaj” ”usuń”. Przyciski ”anuluj” odrzucają jakiekolwiek zmiany a przycisk ”zapisz zmiany” powoduje zapisanie. Do tej strony można się dostać poprzez szczegóły wydarzenia -> edytuj jak i zarówno kliknięciu ”edytuj” w wyświetlaniu wszystkich wydarzeń.



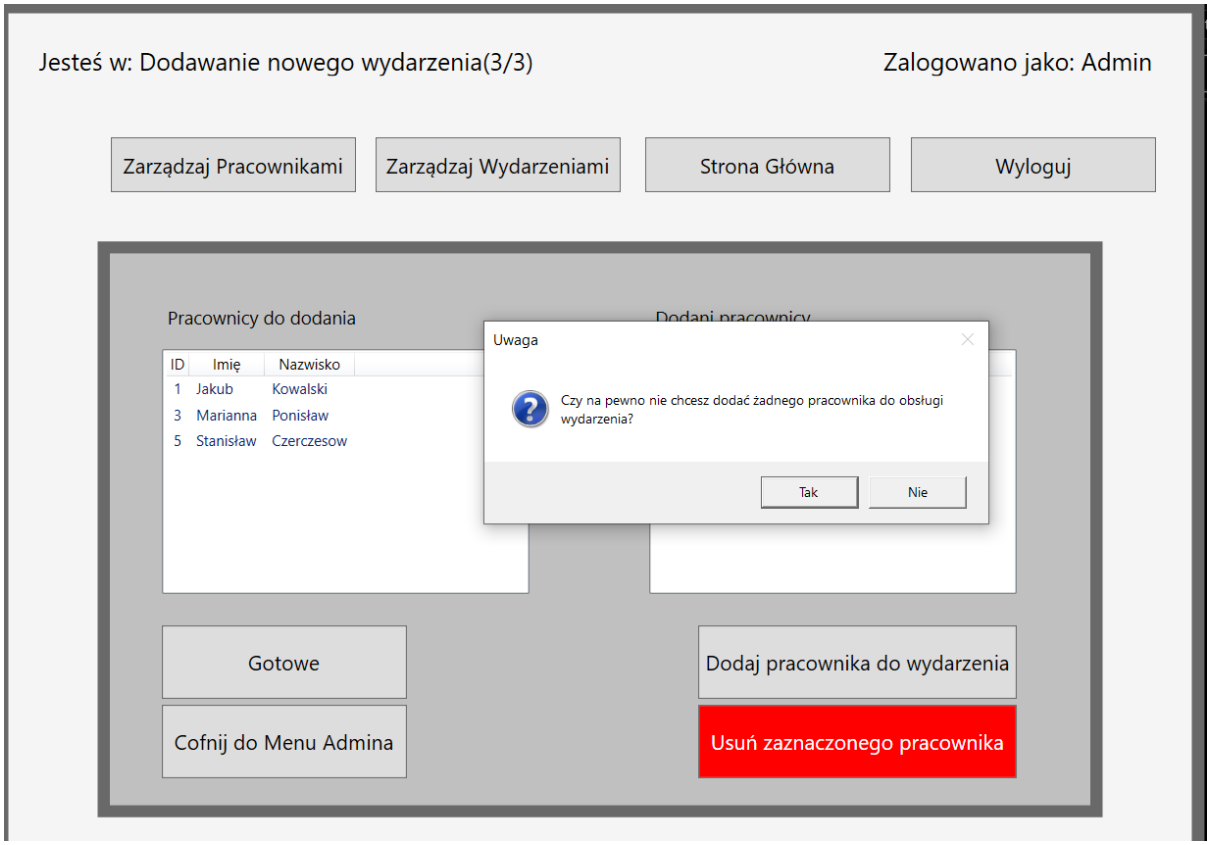
To jest strona, która jest podzielona na 3 segmenty które będą się wyświetlać, jak się kliknie Dalej. Zastosowaliśmy tutaj umowną zasadę, że pola, które trzeba wypełnić oznaczyliśmy gwiazdką ’*’. Przyciski takie jak tabulator działają i nasza klawiatura będzie się skupiać na kolejnych polach do pisania.



Tutaj z kolei, gdy się cofnie przyciskiem albo klawiszem escape to nasze wcześniejsze dane co wpisaliśmy do formularza zostaną zachowane. Godzinę można wpisać tylko cyfry i jeden znak ':' który oddziela godziny od minut, natomiast data jest w formacie DD.MM.RRRR.



Ostatnim zaś etapem jest dodawanie pracowników do wydarzenia. Można ich dodać jak i usuwać podwójnie klikając na nich albo używaniu przycisków. Gdy będziemy chcieli opuścić tą stronę to aplikacja nasz powiadomi czy na pewno chcemy wydarzenie bez żadnego pracownika.



Działa to zarówno, gdy będziemy chcieli wcisnąć escape, przyciski ”gotowe” albo ”cofnij do menu admina” lub na byle jaki przycisk przy nawigacji. Przyciski ”gotowe” i ”cofnij do menu admina” niczym się nie różnią, ale dawają użytkownikowi poczucie, że mają większą kontrolę. W każdej chwili użytkownik może się wylogować powracając do trybu gościa.

Przedstawienie ważnych fragmentów kodu

Jednym z najbardziej powtarzalnych fragmentów kodu to jest przekierowanie na inną podstronę które jest wyrażane w taki sposób:

```
private void ButtonGlowna_Click(object sender, RoutedEventArgs e)
{
    mainWindow.MainFrame.NavigationService.Navigate(new glowna());
    mainWindow.labelStatus.Content = "Jesteś w: Strona Główna";
}
```

Kod przekierowuje użytkownika na inną podstronę zmieniając przy tym status systemu, gdzie się znajduje.

```
private void Button_Usun_Click(object sender, RoutedEventArgs e)
{
    if (listBoxWydarzenia.SelectedItem is Wydarzenie wydarzenie)
    {
        MessageBoxResult result = MessageBox.Show(
    "Czy na pewno chcesz usunąć wydarzenie z bazy danych?",
    "Uwaga",
    MessageBoxButton.YesNo,
    MessageBoxImage.Question);
        if (result == MessageBoxResult.Yes)
        {
                int idWydarzenia = wydarzenie.Id_Wydarzenia;
                // Wywołaj metodę usuwającą pracownika z bazy danych na podstawie idPracownika
                UsunWydarzenie(idWydarzenia);

                // Odśwież listę pracowników
                wydarzenia.Clear();

                WczytajWydarzenia();

        }
    }
}
```

To jest kod, który usuwa z bazy danych wydarzenie permanentnie i aplikacja wysyła zapytanie w postaci okienka czy aby na pewno to miał na myśli.

```
    private void Button_zaloguj_click(object sender, RoutedEventArgs e)
    {
        if (textbox_login.Text == login && textbox_password.Password == password)
        {
            Esc.logowanie = 1;
            MessageBox.Show("Logowanie powiodło się!");
            mainWindow.labelzalogowano.Content = "Zalogowano jako: Admin";
            mainWindow.labelStatus.Content = "Jesteś w: Strona Główna";
            mainWindow.NavigationFrame.NavigationService.Navigate(new NawigacjaAdmin());
            mainWindow.MainFrame.NavigationService.Navigate(new glowna_admin());

        }
        else
        {
            MessageBox.Show("Błędne hasło lub login");
            textbox_login.Focus();
            textbox_login.SelectAll();
            Esc.logowanie = 0;
        }
    }
```

Kod który odpowiada za logowanie i przechodzenie między stanem gościa a administratora.

```
private void DalejLokClick(object sender, RoutedEventArgs e)
{
    if (ValidateInputFields())
    {
        string data = DataTextBox.Text;
        string godzina = GodzinaTextBox.Text;
        string miasto = MiastoTextBox.Text;
        string adres = AdresTextBox.Text;
        string kod_pocztowy = Kod_Pocztowy_TextBox.Text;
        string nazwa_obiektu = NazwaObiektu_TextBox.Text;

        // Dodawanie wydarzenia do bazy danych
        string connectionString = "Data Source=wydarzenia.db;Version=3;";
        using (SQLiteConnection connection = new SQLiteConnection(connectionString))
        {
            connection.Open();
            string insertQuery = "INSERT INTO Wydarzenie (Nazwa, Osoby_Maks, Cena_BILETU, OPIS, DATA, GODZINA, MIASTO, ADRES, KOD_POCZTOWY, NAZWA_OBIEKTU) " +
                                 "VALUES (@nazwa, @maks_osob, @cena, @opis, @data, @godzina, @miasto, @adres, @kod_pocztowy, @nazwa_obiektu);";
            using (SQLiteCommand insertCommand = new SQLiteCommand(insertQuery, connection))
            {
                insertCommand.Parameters.AddWithValue("@nazwa", nazwa);
                insertCommand.Parameters.AddWithValue("@maks_osob", maks_osob);
                insertCommand.Parameters.AddWithValue("@cena", cena);
                insertCommand.Parameters.AddWithValue("@opis", opis);
                insertCommand.Parameters.AddWithValue("@data", data);
                insertCommand.Parameters.AddWithValue("@godzina", godzina);
                insertCommand.Parameters.AddWithValue("@miasto", miasto);
                insertCommand.Parameters.AddWithValue("@adres", adres);
                insertCommand.Parameters.AddWithValue("@kod_pocztowy", kod_pocztowy);
                insertCommand.Parameters.AddWithValue("@nazwa_obiektu", nazwa_obiektu);

                insertCommand.ExecuteNonQuery();

                // Pobierz ID ostatnio dodanego wydarzenia
                string lastInsertIdQuery = "SELECT last_insert_rowid()";
                using (SQLiteCommand lastInsertIdCommand = new SQLiteCommand(lastInsertIdQuery, connection))
                {
                    lastInsertId = Convert.ToInt32(lastInsertIdCommand.ExecuteScalar());
                    MessageBox.Show($"Dodano Wydarzenie. Nowo dodany rekord ma Id_Wydarzenia: {lastInsertId}");
                }
            }
        }

        // Przejdź do strony dodawania pracowników do wydarzenia
        mainWindow.MainFrame.NavigationService.Navigate(new DodawaniePracownikowDoWydarzenia(lastInsertId));
        mainWindow.labelStatus.Content = "Jesteś w: Dodawanie nowego wydarzenia(3/3)";
    }
}
```

Kod który odpowiada za dodawanie kolejnego wydarzenia do bazy danych.

```
1 odwołanie
private void ButtonFiltrujPoLokalizacji_Click(object sender, RoutedEventArgs e)
{
    var sortedWydarzenia = wydarzenia
    .OrderBy(w => w.MIASTO)
    .ToList();

    listBoxWydarzenia.ItemsSource = sortedWydarzenia;
}

1 odwołanie
```

Kod który odpowiada za sortowanie wydarzeń względem lokalizacji.

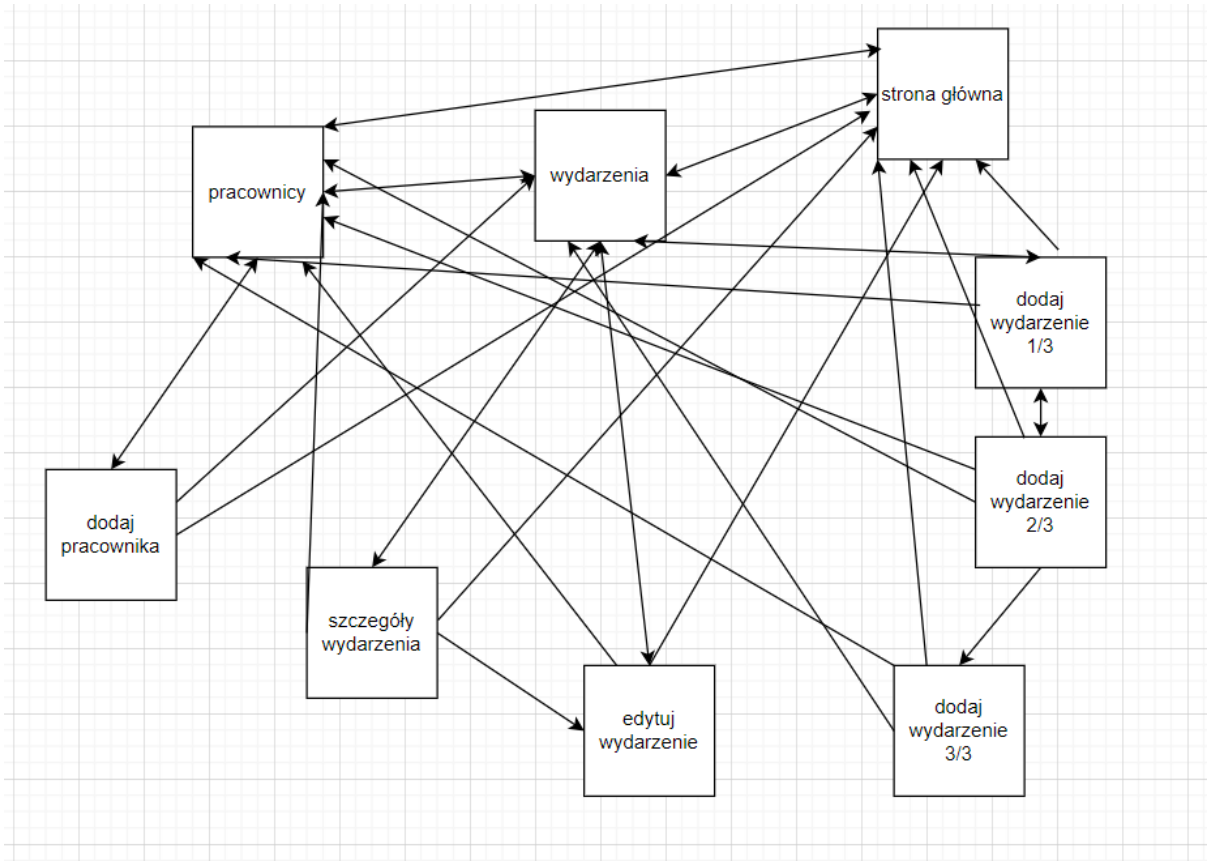
Wszystkie inne kody są pochodnymi tych kodów lub lekko zmodyfikowane.

Projekt interfejsu użytkownika

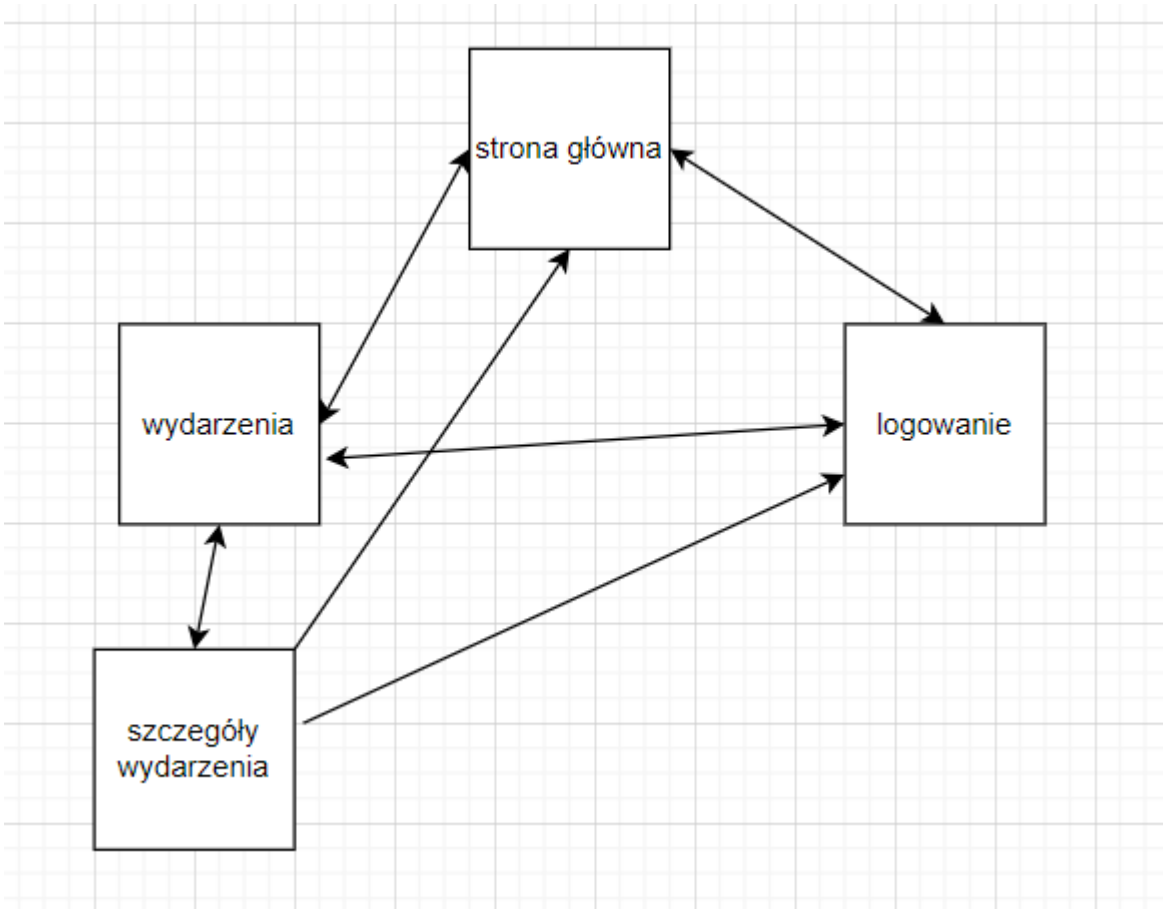
Najbardziej dla naszej aplikacji pasował wzorzec zakładek. Nasze główne okno postawało aktywne a na niej rzutowaliśmy 2 podstrony. Jedna podstrona odpowiadała za nawigację, czyli była zakładką, która zmieniała swoją zawartość zależną od tego czy zalogowany był admin lub nie był zalogowany tak zwany gość. Drugą podstroną była nasz główny komponent, czyli rzeczywista zawartość aplikacji.

Nasz model nawigacyjny było pełne połączenie. Nawigacja zawsze była widoczna to też mogliśmy w każdej chwili powrócić do innej strony. Niektóre strony można byłoby wejść

tylko z poziomy jednej strony. Tak wygląda schemat połączeń dla konta admina:



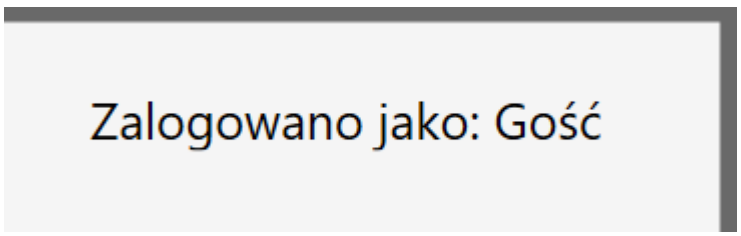
A tak wygląda z punktu widzenia użytkownika:



Zdecydowaliśmy się na takie połączenie by być zgodnym z Heurestyką Nielsona między innymi na 3 zasadę by użytkownik aplikacji czuł kontrole i miał swobodę działania, bo może się jednym a gdzie nie gdzie kilkoma kliknięciami przedostać się na inną stronę. Też do naszej oprawy graficznej dodaliśmy zgodnie z pierwszą zasadą zawsze widoczny status systemy który wyświetla w lewym górnym rogu na jakiej stronie aktualnie użytkownik się znajduje. Po prawej stronie znajduje się jako kto aktualnie jest się zalogowanym a tak to wygląda w naszej aplikacji.

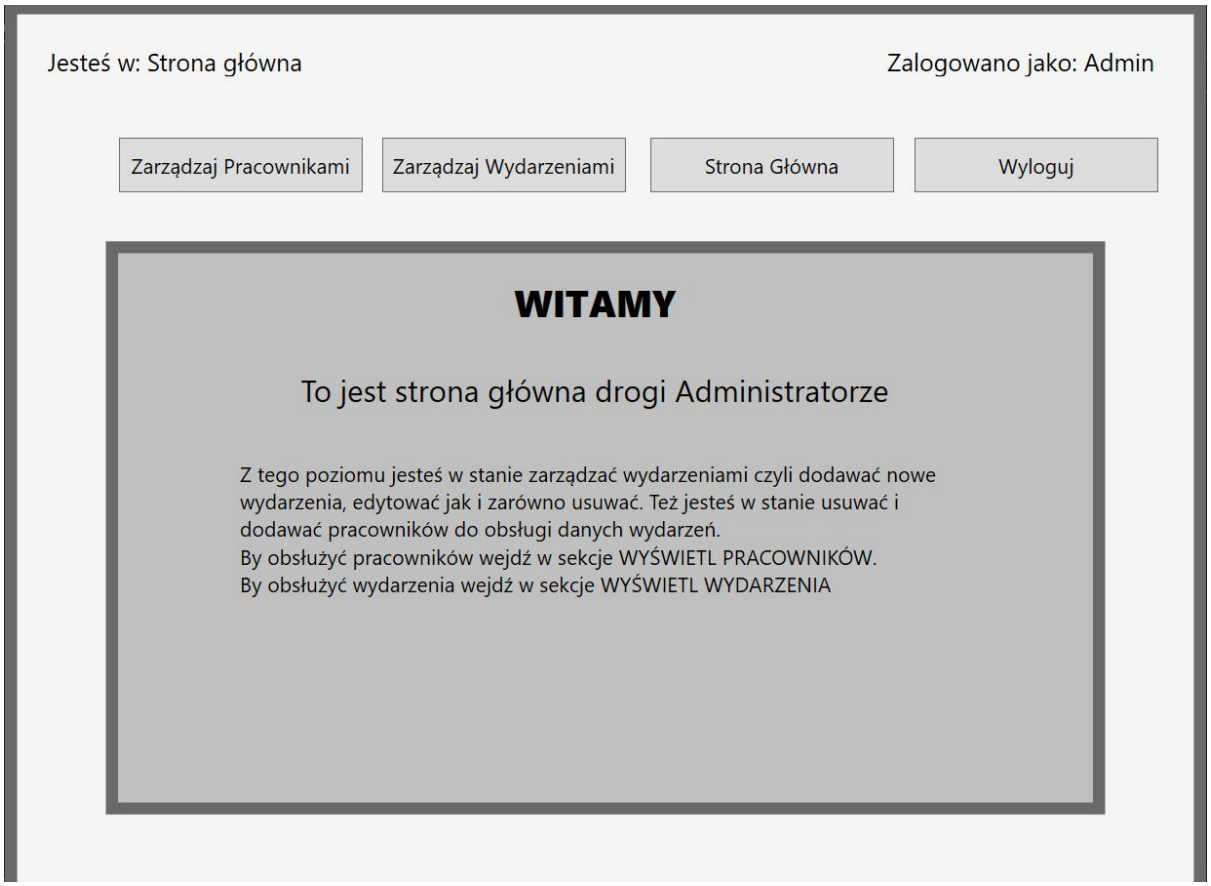
Status systemu:

Status logowania:



Dodaliśmy też strony główne dla dwóch kont by można było je wykorzystać jako powitanie użytkownika jak i trochę wprowadzenie go co może zrobić w naszej aplikacji.

Strona główna dla konta admina:



Strona główna dla gościa:

Użyliśmy odcieni szarości by wszystkie kolory były przyjemne dla oka i nie raziły nas. Wszystkie przyciski które powodują usunięcie czegoś zostały pokolorowane na czerwono by użytkownik przypadkowo nie usunął czegoś.

Testy użyteczności

Gdy zostało zorganizowane spotkanie by przetestować nawzajem aplikacje to nie spodziewaliśmy się nasi testerzy wykryją tyle błędów a były im między innymi:

1. Komórka, gdzie się wpisywał tekst zwiększała swoją objętość, gdy za dużo się wpisało
2. Można było wpisać litery tam, gdzie powinny być cyfry i aplikacja się wysypywała
3. Główne okno się zamykała i otwierała
4. Brak skrótów klawiszowych jak i podwójnych kliknięć

I też były inne błędy które były podobne. Aplikacja wydawała się niekompletna i miała dosyć błędów, ale przynajmniej przez te testy wiedzieliśmy na czym mogliśmy się skupić i jak je naprawić.

Weryfikacja wstępnych wymagań

Aplikacja została zaprojektowana z myślą o obsłudze wydarzeń rozrywkowych poprzez umożliwienie użytkownikom dodawania, usuwania oraz edytowania wydarzeń. Kluczowym aspektem było zapewnienie, aby lista wydarzeń była przejrzysta i łatwa w nawigacji, umożliwiając użytkownikom sortowanie według ich preferencji.

Wymagania funkcjonalne:

1. **Dodawanie, usuwanie i edytowanie wydarzeń:** Aplikacja ma umożliwiać użytkownikom dodawanie nowych wydarzeń, usuwanie istniejących oraz edytowanie szczegółów wydarzeń, takich jak data, opis i lokalizacja.
2. **Przejrzysta lista wydarzeń:** Interfejs użytkownika powinien prezentować listę wydarzeń w sposób klarowny i intuicyjny, umożliwiając szybkie wyszukiwanie i sortowanie po różnych kryteriach, takich jak data, lokalizacja czy cena biletu.

Wymagania niefunkcjonalne:

1. **Zabezpieczenia i niezawodność:** aplikacja powinna być zabezpieczona przez wprowadzaniem błędnych danych do formularzu
2. **Wydajność i elastyczność:** Szata graficzna powinna być przyjemna dla oka i wyglądać dość profesjonalnie. Prosta obsługa i jednoznaczne przyciski.

Perspektywy dalszego rozwoju projektu

Moglibyśmy dodać też kalendarz by móc lepiej i szybciej wstawiać datę do kolejnego wydarzenia. Też myśleliśmy by dodać opcję mapy, na której zawsze by była wyświetlane wydarzenie z dużą dokładnością i napisanym wejściem, całkowitym polem rozrywki. Byłaby ona interaktywna i wyświetlała też najbliższe wydarzenia względem twojej lokalizacji wraz z dojazdem. Dobrą opcją by była możliwość obsługiwania kupna biletu poprzez naszą aplikacje.