

Dokumentacja projektu EasyVENTER

Bazy Danych 1

Patryk Będkowski Łukasz Szarejko
Szymon Skarzyński Marcin Grabysz

25 stycznia 2022

1 Ogólny opis rozwiązania

Koncepcja projektu została zainspirowana naszymi doświadczeniami z kupowania biletów na mecz Polska - Węgry w październiku 2021. System działał niesprawnie, był nieczytelny i ostatecznie musieliśmy kupić bilety w innym sektorze, niż planowaliśmy. Mając w pamięci te doświadczenia, postanowiliśmy zainteresować się tym tematem i stworzyć aplikację do kupowania biletów na wydarzenia sportowe i kulturalne. Stworzona przez nas baza danych i obsługująca ją aplikacja **EasyVENTER** (napisana w Javie przy użyciu biblioteki Swing) umożliwia wymianę informacji między organizatorami wydarzeń oraz klientami kupującymi bilety.

1.1 Możliwości organizatora

Użytkownik zalogowany jako organizator ma możliwość tworzenia nowych wydarzeń oraz ich późniejszego modyfikowania. Tworząc wydarzenie, organizator podaje nazwę wydarzenia, datę i czas rozpoczęcia oraz adres (ulica, miasto oraz kraj z obsługiwanej puli krajów). Aplikacja umożliwia również dywersyfikację cen, dlatego organizator może podzielić pulę biletów na sektory (tu kolejna inspiracja kupowaniem biletów na Stadionie Narodowym) i przypisać każdemu z sektorów unikalną nazwę, liczbę miejsc oraz bazową cenę.

Modyfikując wydarzenie, organizator może zmienić jego datę oraz adres. Opcja modyfikowania puli biletów i ich cen nie jest dostępna, żeby nie powodować konfliktów w sytuacjach, gdy klienci kupili już bilety na pewne wydarzenie.

1.2 Możliwości klienta

Użytkownik zalogowany jako klient może przeglądać dostępne wydarzenia i kupować bilety z dostępnej puli w wybranym sektorze. Jeden klient ma możliwość kupienia maksymalnie dziesięciu biletów w każdej z kategorii: dorosły, dziecko, VIP (kategorie różnią się ceną).

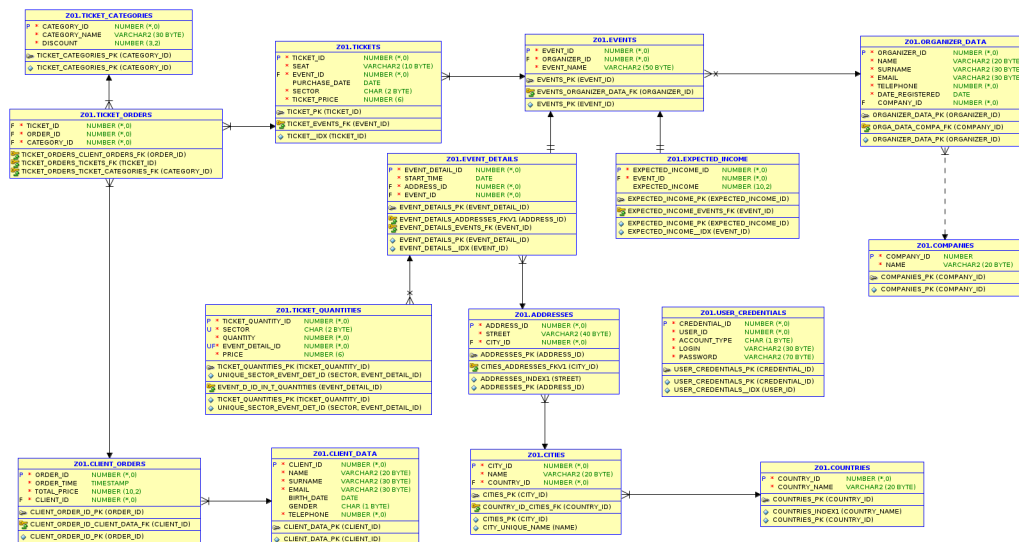
Klient może także przeglądać wydarzenia, na które posiada przynajmniej jeden bilet i anulować dowolną liczbę biletów. Anulowanie ostatniego biletu jest równoznaczne z rezygnacją z całego zamówienia, ale użytkownik może zakupić bilety na to wydarzenie ponownie, znajdując je wśród dostępnych wydarzeń.

1.3 Rejestracja użytkownika

Oprócz obsługi przykładowych użytkowników dostępnych od początku w bazie danych, aplikacja umożliwia dodawanie nowych klientów oraz organizatorów. Dane pobierane podczas rejestracji to: imię, nazwisko, login, hasło, adres email oraz telefon; klient dodatkowo: płeć, data urodzenia; organizator dodatkowo: nazwa firmy.

2 Najważniejsze tabele

Cała baza zawiera 15 tabel - to trochę więcej, niż byłoby konieczne i zgodne z intuicją, jednak podeszliśmy do projektu z myślą o ćwiczeniu i opanowaniu tworzenia oraz modyfikowania baz danych.



Rysunek 1: Model relacyjny wszystkich tabel

Najważniejsze tabele są opisane poniżej:

2.1 Events

Tabela łącząca ID wydarzenia z jego nazwą i ID organizatora. ID wydarzenia umożliwia uzyskanie szczegółów wydarzenia oraz informacji o kupionych biletach.

2.2 Tickets

Każdy wiersz tabeli reprezentuje jeden bilet - bilet przechowuje ID, ID wydarzenia, numer siedzenia, sektor, cenę i datę zakupu.

2.3 Ticket_orders

Tabela przechowuje zamówienia - każde zamówienie polega na kupieniu pewnej ilości biletów różnych kategorii na jedno wydarzenie przez jednego klienta.

2.4 Client_data

Tabela przechowuje informacje o użytkownikach będących klientami: ID, imię, nazwisko, email, data urodzenia, płeć, telefon.

2.5 Organizer_data

Tabela przechowuje informacje o użytkownikach będących organizatorami: ID, imię, nazwisko, email, telefon, data zarejestrowania, ID firmy.

3 Opis funkcji i procedur w bazie danych

3.1 Funkcje

- `calculate_total_price` - funkcja ta odpowiada za obliczanie całkowitej ceny zamówienia złożonego przez klienta. Pobiera podstawową cenę biletu, mnożnik zależny od wybranej kategorii oraz aktualny koszt zamówienia, a następnie liczy cenę biletu w danej kategorii i zwraca aktualny koszt zamówienia powiększony o cenę rozważanego biletu.
- `get_company` - funkcja ta zwraca nazwę firmy, do której należy organizator, którego login został podany jako argument funkcji.

3.2 Procedury

- `buy_ticket` - głównym zadaniem procedury `buy_ticket` stanowi zakupienie pojedynczego biletu dla danego zamówienia klienta.

Procedura przyjmuje 7 argumentów, z czego dwa dotyczą danych klienta, kolejne 3 dotyczą typu biletu do zakupienia, natomiast ostatnie 2 to kolejno numery ID zamówienia oraz zmienna typu OUT, która zwraca owy numer zamówienia.

Architektura tej procedury polega na tym, że wykonuje się ona kilka razy, jeżeli klient w ramach jednego zamówienia zakupi więcej niż jeden bilet. Przy pierwszym wywołaniu, do procedury przekazywany jest nieistniejący numer zamówienia. W takim przypadku system bazy danych wykrywa to i tworzy nowe zamówienie, którego id jest następnie zwracane na zewnątrz. Aplikacja w Javie, odbiera ID owego zamówienia i

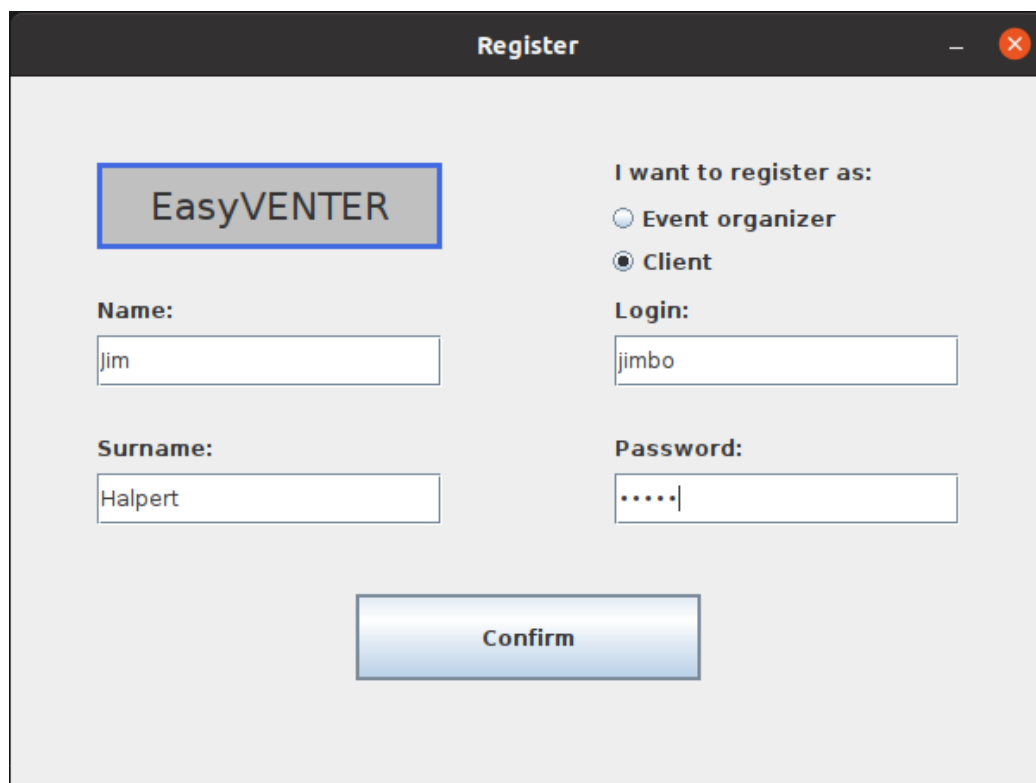
wywołuje tę procedurę ponownie, lecz teraz z istniejącym numerem zamówienia, dzięki czemu możliwe jest kupienie kilku biletów w ramach jednego zamówienia.

Przy zakupie wymaganego biletu, sprawdzane są założenia dotyczące, czy specyfikacja biletu jest poprawna, tzn. czy owy event, kategoria biletu istnieją, a także czy klient jest zarejestrowany w systemie. Procedura wywołuje również funkcję `calculate_total_price`, która oblicza cenę zakupionego biletu.

- `add_event` - procedura ta dodaje nowe wydarzenie do bazy danych, uzupełniając przy tym wymagane tabele, takie jak `addresses`, `event_details` itp.
- `remove_ticket` - usuwa bilet z zamówienia klienta, zmieniając przy tym koszt danego zamówienia.
- `register_organizer` - dodaje organizatora do bazy danych.
- `register_client` - analogicznie jak `register_organizer`, tylko dla klienta.
- `edit_event` - procedura ta edytuje możliwe do edycji informacje o wydarzeniu. Możliwe do edycji są czas i miejsce wydarzenia.
- `cancel_event` - usuwa dane wydarzenie z bazy danych. Ze względu na problem, przy projekcie z PAPu, jako jedyna nie jest używana w aplikacji. Jest jednak możliwością rozwoju projektu.
- `add_ticket` - dodaje bilet do nowo utworzonego wydarzenia, który może zostać zakupiony
- `add_ticket_quantity` - ustawia ilość biletów dla danego sektora i o danej cenie dla konkretnego wydarzenia.
- `add_ticket_to_order` - łączy zakupiony bilet ze złożonym przez klienta zamówieniem poprzez wpis do tabeli `ticket_orders`.

4 Wygląd interfejsu graficznego

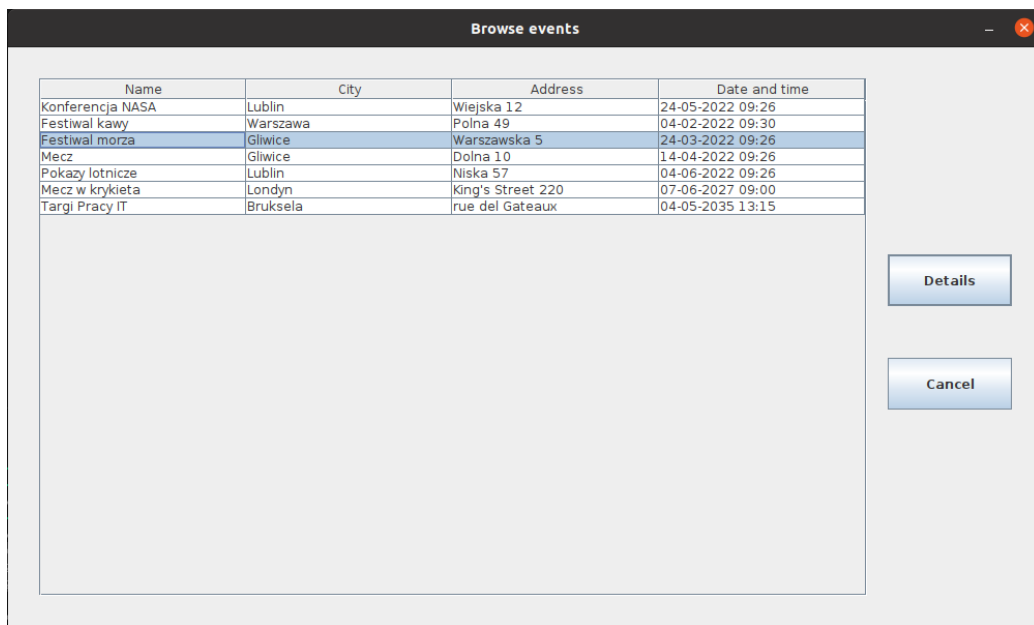
Poniżej przedstawione jest kilka przykładowych ujęć interfejsu graficznego aplikacji EasyVENTER.



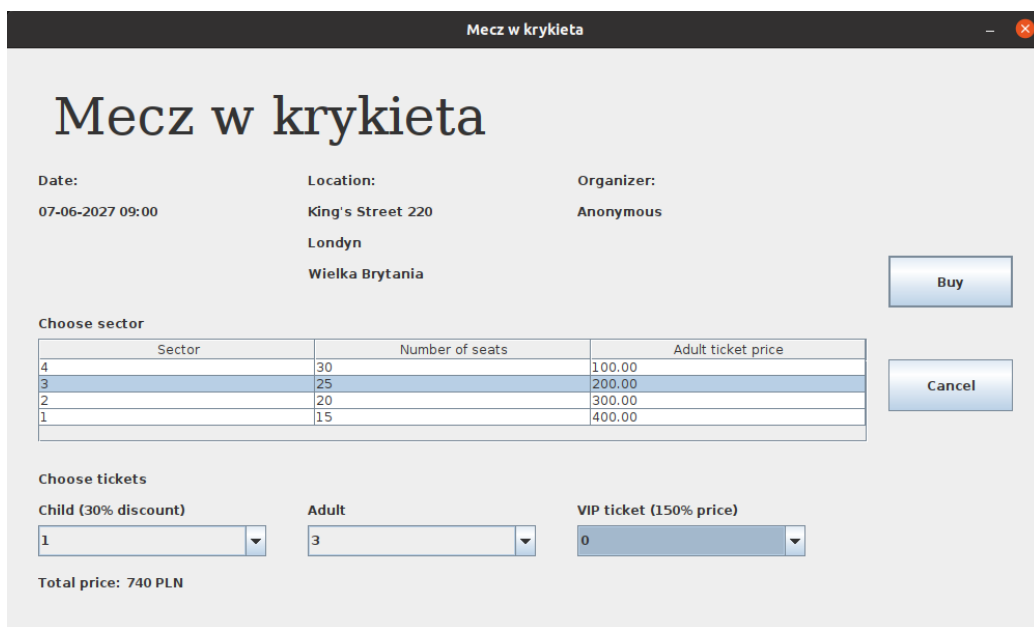
The image shows a window titled "Register" with a dark header bar containing the title and standard window controls (minimize, maximize, close). The main content area is light gray and contains the following elements:

- A logo box with the text "EasyVENTER" and a blue border.
- A section titled "I want to register as:" with two radio button options: "Event organizer" and "Client". The "Client" option is selected.
- Form fields for "Name:" (containing "jim") and "Surname:" (containing "Halpert").
- Form fields for "Login:" (containing "jimbo") and "Password:" (containing ".....").
- A large blue "Confirm" button at the bottom center.

Rysunek 2: Ekran rejestracji użytkownika



Rysunek 3: Ekran przeglądania dostępnych wydarzeń



Rysunek 4: Ekran kupowania biletów na wydarzenie

5 Analiza rozwiązania

5.1 Ograniczenia

Baza danych została zaprojektowana w taki sposób, aby wprowadzić przejrzysty podział informacji. Wyróżnić możemy pewne grupy tabel, które ułatwiają przechowywanie powiązanych ze sobą danych np. zamówienia klientów (`client_orders`) oraz informacje o poszczególnych biletach w tych w zamówieniach (`ticket_orders`). Zaletą takiego rozwiązania jest przechowywanie powiązanych ze sobą informacji w mniej zgrupowany sposób. Wadą natomiast jest potrzeba budowy bardziej skomplikowanych zapytań w celu uzyskania dostępu do większej liczby informacji jednocześnie. W rozwijanym przez nas programie wykorzystującym ową bazę, takowe zapytania stanowią zdecydowaną mniejszość, co przeważało nad decyzją wprowadzenia większej ilości tabel.

5.2 Możliwości rozwoju

Dzięki tabeli `expected_income` możemy w przyszłości wzbogacić nasz program o udostępnianie organizatorom informacji, jakiego przychodu mogą się spodziewać przy obecnym obłożeniu ich wydarzenia.

W tabeli `ticket_categories` możemy wprowadzić więcej kategorii biletów, jeśli zajdzie taka konieczność - bazowo udostępniliśmy tylko trzy kategorie (dorosły, dziecko, VIP), ale można sobie wyobrazić dużo więcej; wystarczy choćby spojrzeć na kilkanaście zniżek oferowanych przez PKP: uczniowska, studencka, kombatancka, itp.

5.3 Inne wnioski

Dzięki wydzieleniu loginów i haseł do osobnej tabeli `user_credentials` nie ma potrzeby pobierania wszystkich danych użytkowników podczas logowania. Inaczej jest podczas rejestracji, ale też logowanie użytkownika jest operacją występującą znacznie częściej niż rejestrowanie nowego.

Mamy wrażenie, że baza danych jest przejrzysta i łatwa w użytkowaniu. Wiele tabel zapewnia rozproszenie informacji, ale programista pracujący z

tabelą intuicyjnie odnajdzie potrzebne dane dzięki klarownym nazwom i dostępnemu diagramowi relacyjnemu. Użytkownik ma do dyspozycji prosty w obsłudze i estetyczny interfejs.