|  |  |
| --- | --- |
|  | Projekt Podstawy Baz Danych  Katedra Informatyki Akademii Górniczo-Hutniczej w Krakowie, studia zaoczne inżynierskie |

**EventCreator**

**BAZODANOWY SYSTEM ZARZĄDZANIA KONFERENCJAMI**

|  |  |
| --- | --- |
| Wykonali: | Tomasz BELCZYK |
|  | Michał DZIARMAGA |

Spis treści

[I Wstęp 4](#_Toc42253697)

[II Wymagania 4](#_Toc42253698)

[III Funkcjonalności 4](#_Toc42253699)

[IV Schemat bazy danych 5](#_Toc42253700)

[V Opisy tabel 6](#_Toc42253702)

[1.1. Tabela [dbo].Clients 6](#_Toc42253703)

[1.2. Tabela [dbo].Conferences 6](#_Toc42253704)

[1.3. Tabela [dbo].WorkshopTypes 7](#_Toc42253705)

[1.4. Tabela [dbo].ConferenceDates 8](#_Toc42253706)

[1.5. Tabela [dbo].ConferenceDatesPriceThresholds 9](#_Toc42253707)

[1.6. Tabela [dbo].DailyConferenceWorkshops 9](#_Toc42253708)

[1.7. Tabela [dbo].Participants 11](#_Toc42253709)

[1.8. Tabela [dbo].Payments 12](#_Toc42253710)

[1.9. Tabela [dbo].Payments 12](#_Toc42253712)

[1.10. Tabela [dbo].ClientInformations 14](#_Toc42253713)

[1.11. Tabela [dbo].ClientInformations 15](#_Toc42253714)

[VI Opis procedur 16](#_Toc42253715)

[1.1. Procedura [dbo].RegisterConference 16](#_Toc42253716)

[1.2. Procedura [dbo].RegisterClient 17](#_Toc42253717)

[1.3. Procedura [dbo].RegisterCompany 18](#_Toc42253718)

[1.4. Procedura [dbo].RegisterForConference 19](#_Toc42253719)

[1.5. Procedura [dbo].AddWorkshopType 20](#_Toc42253720)

[1.6. Procedura [dbo]. RegisterConferenceDateWorkshop 21](#_Toc42253721)

[1.7. Procedura [dbo]. FillCompanyParticipantsNames 22](#_Toc42253723)

[1.8. Procedura [dbo]. RegisterConferenceParticipantForWorkshop 23](#_Toc42253725)

[1.9. Procedura [dbo]. ClientPayment 25](#_Toc42253726)

[1.10. Procedura [dbo]. SetConferenceDatePrice 26](#_Toc42253727)

[1.11. Procedura [dbo]. SetConferenceDatePriceThreshold 26](#_Toc42253728)

[VII Opis fukcji 27](#_Toc42253729)

[1.1. Funkcja [dbo].TruncateDate 27](#_Toc42253730)

[1.2. Funkcja [dbo].GetConferencePrice 27](#_Toc42253731)

[1.3. Funkcja [dbo].GetConferenceDatePriceForClient 28](#_Toc42253732)

[1.4. Funkcja [dbo].GetClientPrice 29](#_Toc42253733)

[1.5. Funkcja [dbo].GetIntSetting 30](#_Toc42253734)

[1.6. Funkcja [dbo].GetVarcharSetting 30](#_Toc42253735)

[1.7. Funkcja [dbo].GetNumericSetting 30](#_Toc42253736)

[1.8. Funkcja [dbo].GetDatetimeSetting 31](#_Toc42253737)

[VIII Widoki 31](#_Toc42253738)

[1.1. Widok [dbo].GetConferenceParticipants 31](#_Toc42253739)

[1.2. Widok [dbo].GetClientsWithNotFilledParticipants 32](#_Toc42253740)

[1.3. Widok [dbo].GetClientsPayments 32](#_Toc42253741)

# Wstęp

EventCreator to bazodanowy system zarządzania konferencjami, który umożliwia dodawanie konferencji, rejestracje użytkowników, czy płatności. EventCreator powstał z myślą o niewielkich przedsiębiorstwach obsługujących i kreujących konferencje w celu uproszczenia działań związanych z organizacją i zarządzaniem tych wydarzeń.

# Wymagania

Aby skorzystać z systemu EventCreator wymagane są następujące narzędzia:

* Windows 10
* Microsoft SQL Server Management Studio

Oraz podstawowa znajomość zagadnień związanych z SQL Server.

# Funkcjonalności

System EventCreator umożliwia:

* Rejestrowanie klientów indywidualnych i firm
* Obsługa kilkudniowych i jednodniowych konferencji
* Obsługa warsztatów podczas konferencji
* Obsługa płatności i różnych cen konferencji

# Schemat bazy danych

1. **Schemat bazy danych.**

# Opisy tabel

## Tabela [dbo].Clients

CREATE TABLE [dbo].[Clients]

(

[Id] uniqueidentifier default newid() NOT NULL,

[FirstName] varchar(50) NULL ,

[LastName] varchar(50) NULL ,

[Created] datetime NOT NULL ,

[ClientInformationsId] uniqueidentifier NOT NULL ,

CONSTRAINT [PK\_Clients] PRIMARY KEY CLUSTERED ([Id] ASC)

);

GO

CONSTRAINT [FK\_Clients\_ClientInformations\_ClientInformationsID] FOREIGN KEY ([ClientInformationsId]) REFERENCES [dbo].[ClientInformations]([Id])

### Opis

Tabela zawiera niezbędne informacje o wszystkich klientach. W tabeli wprowadza się dane klientów indywidualnych oraz firm.

Kolumny:

* Id – unikalny identyfikator klienta
* FirstName – Imię klienta (NULL w przypadku firmy)
* LastName – Nazwisko klienta (NULL w przypadku firmy)
* Created – data utworzenia klienta
* ClientInformationsId – Kolumna wiążąca do szczegółowych informacji o klientach

### Klucze

* Id – primary key
* ClientInformationsId – foreign key do tabeli ClientInformations (Id)

## Tabela [dbo].Conferences

CREATE TABLE [dbo].[Conferences]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[Name] varchar(200) NOT NULL ,

[Description] varchar(500) NOT NULL ,

[StartOfConference] datetime NOT NULL ,

[EndOfConference] datetime NOT NULL ,

CONSTRAINT [PK\_Conferences] PRIMARY KEY CLUSTERED ([Id] ASC)

);

GO

### Opis

Tabela zawierająca informacje o konferencjach.

Kolumny:

* Id – unikalny identyfikator konferencji
* Name – Nazwa konferencji
* Description – Opis szczegółowy konferencji
* StartOfConference – data rozpoczęcia konferencji
* EndOfConference – data zakończenia konferencji

### Klucze

* Id – primary key

## Tabela [dbo].WorkshopTypes

CREATE TABLE [dbo].[WorkshopTypes]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[Name] varchar(100) NOT NULL ,

[Description] varchar(500) NOT NULL ,

[Price] numeric(18,2) NULL ,

CONSTRAINT [PK\_Workshops] PRIMARY KEY CLUSTERED ([Id] ASC)

);

GO

### Opis

Tabela zawierająca informacje o dodanych warsztatach. Warsztaty w tej tabeli nie są w żaden sposób powiązane z konkretną konferencją.

Kolumny:

* Id – unikalny identyfikator warsztatu
* Name – nazwa warsztatu
* Description – opis szczegółowy warsztatu
* Price – koszt warsztatu (za uczestnika. NULL jeśli bezpłatny)

### Klucze

* Id – primary key

## Tabela [dbo].ConferenceDates

CREATE TABLE [dbo].[ConferenceDates]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ConferenceId] uniqueidentifier NOT NULL ,

[Date] datetime NOT NULL ,

[DayBasePrice] numeric(10,2) NULL,

CONSTRAINT [PK\_ConferenceDates] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_ConferenceDates\_Conferences\_ConferenceId] FOREIGN KEY ([ConferenceId]) REFERENCES [dbo].[Conferences]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_ConferenceDates\_ConferenceId] ON [dbo].[ConferenceDates]

(

[ConferenceId] ASC

)

GO

### Opis

Tabela zawierająca informacje o konkretnych datach konferencji z przedziału dat zdefiniowanych w konferencji.

### Kolumny:

* Id – unikalny identyfikator daty konferencji
* ConferenceId – unikalny identyfikator konferencji
* Date – konkretna data konferencji
* DayBasePrice – bazowa cena daty konferencji

### Klucze

* Id – primary key
* ConferenceId - foreign key do tabeli Conferences(Id)

## Tabela [dbo].ConferenceDatesPriceThresholds

CREATE TABLE [dbo].[ConferenceDatesPriceThresholds]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ConferenceDateId] uniqueidentifier NOT NULL ,

[ThresholdDate] datetime NOT NULL ,

[Price] numeric(10,2) NOT NULL ,

CONSTRAINT [PK\_ConferenceDatesPriceThresholds] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_ConferenceDatesPriceThresholds\_ConferenceDateId] FOREIGN KEY ([ConferenceDateId]) REFERENCES [dbo].[ConferenceDates]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_ConferenceDatesPriceThresholds\_ConferenceDateId] ON [dbo].[ConferenceDatesPriceThresholds]

(

[ConferenceDateId] ASC

)

GO

### Opis

Tabela zawierająca informacje o progach cenowych konferencji.

### Kolumny:

* Id – unikalny identyfikator progu cenowego
* ConferenceDateId – unikalny identyfikator daty konferencji
* ThresholdDate – data po której próg cenowy ma obowiązywać
* Price – wartość progu cenowego

### Klucze

* Id – primary key
* ConferenceDateId - foreign key do tabeli ConferenceDates(Id)

## Tabela [dbo].DailyConferenceWorkshops

CREATE TABLE [dbo].[DailyConferenceWorkshops]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ConferenceDateId] uniqueidentifier NOT NULL ,

[WorkshopId] uniqueidentifier NOT NULL ,

[ParticipantsLimit] numeric(3,0) NOT NULL ,

[Date] datetime NOT NULL ,

[StartOfWorkshop] datetime NOT NULL ,

[EndOfWorkshop] datetime NOT NULL ,

CONSTRAINT [PK\_ConferenceWorkshops] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_DailyConferenceWorkshops\_ConferenceDates\_ConferenceDateId] FOREIGN KEY ([ConferenceDateId]) REFERENCES [dbo].[ConferenceDates]([Id]),

CONSTRAINT [FK\_DailyConferenceWorkshops\_WorkshopTypes\_WorkshopId] FOREIGN KEY ([WorkshopId]) REFERENCES [dbo].[WorkshopTypes]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_DailyConferenceWorkshops\_ConferenceDateId] ON [dbo].[DailyConferenceWorkshops]

(

[ConferenceDateId] ASC

)

GO

CREATE NONCLUSTERED INDEX [IN\_DailyConferenceWorkshops\_WorkshopId] ON [dbo].[DailyConferenceWorkshops]

(

[WorkshopId] ASC

)

GO

### Opis

Tabela zawierająca warsztaty w konkretnym dniu daty konferencji.

### Kolumny:

* Id – unikalny identyfikator warsztatu daty konferencji
* ConferenceDateId – unikalny identyfikator daty konferencji
* WorkshopId – unikalny identyfikator warsztatu
* ParticipantsLimit – limit uczestników
* Date – data dnia konferencji
* StartOfWorkshop – data z czasem rozpoczęcia warsztatu
* EndOfWorkshop – data z czasem zakończenia warsztatu

### Klucze

* Id – primary key
* ConferenceDateId - foreign key do tabeli ConferenceDates(Id)
* WorkshopId – foreign key do tabeli WorkshopTypes(Id)

## Tabela [dbo].Participants

CREATE TABLE [dbo].[Participants]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ConferenceDateId] uniqueidentifier NOT NULL ,

[ClientId] uniqueidentifier NOT NULL ,

[FirstName] varchar(50) NULL ,

[LastName] varchar(50) NULL ,

[RegistrationDate] datetime NOT NULL ,

[StudentId] varchar(50) NULL ,

CONSTRAINT [PK\_Participants] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_Participants\_ConferenceDates\_ConferenceDateId] FOREIGN KEY ([ConferenceDateId]) REFERENCES [dbo].[ConferenceDates]([Id]),

CONSTRAINT [FK\_Participants\_Clients\_ClientId] FOREIGN KEY ([ClientId]) REFERENCES [dbo].[Clients]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_Participants\_ConferenceDateId] ON [dbo].[Participants]

(

[ConferenceDateId] ASC

)

GO

CREATE NONCLUSTERED INDEX [IN\_Participants\_ClientId] ON [dbo].[Participants]

(

[ClientId] ASC

)

GO

### Opis

Tabela zawierająca informacje o uczestnikach konkretnych dni konferencji. Jeśli ClientId odnosi się do klienta indywidualnego, imię nazwisko są kopiowane.

### Kolumny:

* Id – unikalny identyfikator uczestnika
* ConferenceDateId – unikalny identyfikator daty konferencji
* ClientId – unikalny identyfikator klienta
* FirstName – imię uczestnika
* LastName – nazwisko uczestnika
* RegistrationDate – data zarejestrowania uczestnika w dacie konferencji
* StudentId – identyfikator legitymacji studenckiej

### Klucze

* Id – primary key
* ConferenceDateId - foreign key do tabeli ConferenceDates(Id)
* ClientId - foreign key do tabeli Clients(Id)

## Tabela [dbo].Payments

CREATE TABLE [dbo].[Payments]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ClientId] uniqueidentifier NOT NULL ,

[Payment] numeric(10,2) NOT NULL ,

[PaymentDate] datetime NOT NULL ,

CONSTRAINT [PK\_Payments] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_Payments\_Clients\_ClientId] FOREIGN KEY ([ClientId]) REFERENCES [dbo].[Clients]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_Payments\_ClientId] ON [dbo].[Payments]

(

[ClientId] ASC

)

GO

### Opis

Tabela zawierająca informacje o płatnościach klientów.

### Kolumny:

* Id – unikalny identyfikator płatności
* ClientId – unikalny identyfikator klienta
* Payment – wartość płatności
* PaymentDate – data płatności

### Klucze

* Id – primary key
* ClientId - foreign key do tabeli Clients(Id)

## Tabela [dbo].Payments

CREATE TABLE [dbo].[ParticipantWorkshops]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ParticipantId] uniqueidentifier NOT NULL ,

[DailyConferenceWorkshopId] uniqueidentifier NOT NULL ,

CONSTRAINT [PK\_ParticipantWorkshops] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_ParticipantWorkshops\_Participants\_ParticipantId] FOREIGN KEY ([ParticipantId]) REFERENCES [dbo].[Participants]([Id]),

CONSTRAINT [FK\_ParticipantWorkshops\_DailyConferenceWorkshops\_DailyConferenceWorkshopId] FOREIGN KEY ([DailyConferenceWorkshopId]) REFERENCES [dbo].[DailyConferenceWorkshops]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_ParticipantWorkshops\_ParticipantId] ON [dbo].[ParticipantWorkshops]

(

[ParticipantId] ASC

)

GO

CREATE NONCLUSTERED INDEX [IN\_ParticipantWorkshops\_DailyConferenceWorkshopId] ON [dbo].[ParticipantWorkshops]

(

[DailyConferenceWorkshopId] ASC

)

GO

### Opis

Tabela zawierająca informacje o warsztatach, na które uczestnik daty konferencji się zarejestrował.

### Kolumny:

* Id – unikalny identyfikator połączenia
* ParticipantId – unikalny identyfikator uczestnika
* DailyConferenceWorkshopId – unikalny identyfikator warsztatu daty konferencji

### Klucze

* Id – primary key
* ParticipantId - foreign key do tabeli Participants(Id)

## Tabela [dbo].ClientInformations

CREATE TABLE [dbo].[ClientInformations]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[ClientId] uniqueidentifier NOT NULL ,

[IsCompany] bit NOT NULL ,

[BirthDate] datetime NULL ,

[Email] varchar(200) NOT NULL ,

[Country] varchar(50) NOT NULL ,

[City] varchar(50) NOT NULL ,

[ZipCode] varchar(50) NOT NULL ,

[Street] varchar(50) NOT NULL ,

[TaxId] varchar(50) NULL ,

[CompanyName] varchar(100) NULL ,

CONSTRAINT [PK\_ClientInformations] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_ClientInformations\_Clients\_ClientId] FOREIGN KEY ([ClientId]) REFERENCES [dbo].[Clients]([Id])

);

GO

CREATE NONCLUSTERED INDEX [IN\_ClientInformations\_ClientId] ON [dbo].[ClientInformations]

(

[ClientId] ASC

)

GO

### Opis

Tabela zawierająca szczegółowe informacje o klientach.

### Kolumny:

* Id – unikalny identyfikator informacji
* ClientId – unikalny identyfikator klienta
* IsCompany – pole informujące czy klient jest firmą czy osobą (0 – klient indywidualny, 1 – firma)
* BirthDate – data urodzin (NULL jeśli firma)
* Email – email kontaktowy klienta
* Country – kraj klienta
* City – miasto klienta
* ZipCode – kod pocztowy klienta
* Street - ulica klienta
* TaxId – NIP firmy (NULL jeśli osoba)
* CompanyName – nazwa firmy (NULL jeśli osoba)

### Klucze

* Id – primary key
* ClientId - foreign key do tabeli Clients(Id)

## Tabela [dbo].ClientInformations

CREATE TABLE [dbo].[Settings]

(

[Id] uniqueidentifier default newid() NOT NULL ,

[Name] varchar(100) UNIQUE NOT NULL,

[Description] varchar(500) NULL,

[VarcharValue] varchar(200) NULL,

[NumericValue] numeric(38,23) NULL,

[IntValue] int NULL,

[DatetimeValue] datetime NULL,

CONSTRAINT [PK\_Settings] PRIMARY KEY CLUSTERED ([Id] ASC)

);

GO

CREATE NONCLUSTERED INDEX [IN\_Settings\_Name] ON [dbo].[Settings]

(

[Name] ASC

)

GO

### Opis

Tabela zawierająca ustawienia.

### Kolumny:

* Id – unikalny identyfikator ustawienia
* Name – nazwa ustawień
* Description – opis ustawień
* VarcharValue – wartość ustawienia jako varchar(200)
* NumericValue – wartość ustawienia jako numeric(32,23)
* IntValue – wartość ustawienia jako int
* DatetimeValue – wartość ustawienia jako data

### Klucze

* Id – primary key

# Opis procedur

## Procedura [dbo].RegisterConference

CREATE PROCEDURE [dbo].RegisterConference(@name varchar(200),

@description varchar(500),

@startOfConference datetime,

@endOfConference datetime)

AS

BEGIN

DECLARE @sql varchar(1000);

DECLARE @error varchar(100);

DECLARE @newId uniqueidentifier = newid();

SET @startOfConference = [dbo].TruncateDate(@startOfConference);

SET @endOfConference = [dbo].TruncateDate(@endOfConference);

if(@name is null or @name = '')

THROW 50001, 'Nazwa konferencji nie może byc pusta', 1;

if(@startOfConference > @endOfConference)

THROW 50001, 'Konferencja nie może zaczynać się później niż się kończy.', 1;

INSERT INTO [dbo].Conferences(Id, Name, Description, StartOfConference, EndOfConference)

VALUES(@newId,@name,@description,@startOfConference,@endOfConference);

WHILE(@startOfConference <= @endOfConference)

BEGIN

INSERT INTO [dbo].ConferenceDates(ConferenceId,Date) VALUES(@newId,@startOfConference)

SET @startOfConference= DATEADD(day, 1, @startOfConference);

END

SELECT @newId;

END;

GO

### Opis

Procedura tworząca konferencje.

### Parametry:

* @name – nazwa konferencji
* @description – opis konferencji
* @startOfConference – data startu konferencji
* @endOfConference – data zakończenia konferencji

## Procedura [dbo].RegisterClient

CREATE PROCEDURE [dbo].RegisterClient(@firstName varchar(50),

@lastName varchar(50),

@birthDate datetime,

@email varchar(200),

@country varchar(50),

@city varchar(50),

@zipCode varchar(50),

@street varchar(50))

AS

BEGIN

DECLARE @clientId uniqueidentifier = newid();

DECLARE @clientInformationsId uniqueidentifier = newid();

if(@firstName is null or @firstName = '')

THROW 50001, 'Imię nie może być puste.', 1;

if(@lastName is null or @lastName = '')

THROW 50001, 'Nazwisko nie może być puste.', 1;

if(@email is null or @email = '')

THROW 50001, 'Email nie może być pusty.', 1;

if(@country is null or @country = '')

THROW 50001, 'Kraj nie może być pusty.', 1;

if(@city is null or @city = '')

THROW 50001, 'Miasto nie może być puste.', 1;

if(@zipCode is null or @zipCode = '')

THROW 50001, 'Kod pocztowy nie może być pusty.', 1;

if(@street is null or @street = '')

THROW 50001, 'Ulica nie może być pusta.', 1;

if(@birthDate is null)

THROW 50001, 'Data urodzin nie może być pusta.', 1;

if(@birthDate > DATEADD(yy, -18, SYSDATETIME()))

THROW 50002, 'Aby się zarejestrować użytkownik musi byc pełnoletni.', 1;

INSERT INTO [dbo].Clients(Id, FirstName, LastName, Created, ClientInformationsId)

VALUES(@clientId, @firstName, @lastName, SYSDATETIME(), @clientInformationsId);

INSERT INTO [dbo].ClientInformations(Id, ClientId, IsCompany, BirthDate, Email, Country, City, ZipCode, Street)

VALUES(@clientInformationsId, @clientId, 0, @birthDate, @email, @country, @city, @zipCode, @street)

SELECT @clientId;

END;

GO

### Opis

Procedura tworząca klienta.

### Parametry:

* @firstName – imię klienta
* @lastName – nazwisko klienta
* @birthDate – data urodzin klienta
* @email – email klienta
* @country – kraj klienta
* @city – miasto klienta
* @zipCode – kod pocztowy klienta
* @street – ulica klienta

## Procedura [dbo].RegisterCompany

CREATE PROCEDURE [dbo].RegisterCompany(@companyName varchar(100),

@taxId varchar(50),

@email varchar(200),

@country varchar(50),

@city varchar(50),

@zipCode varchar(50),

@street varchar(50))

AS

BEGIN

DECLARE @clientId uniqueidentifier = newid();

DECLARE @clientInformationsId uniqueidentifier = newid();

if(@companyName is null or @companyName = '')

THROW 50001, 'Nazwa firmy nie może być pusta.', 1;

if(@taxId is null or @taxId = '')

THROW 50001, 'NIP firmy nie może być pusty.', 1;

if(@email is null or @email = '')

THROW 50001, 'Email nie może być pusty.', 1;

if(@country is null or @country = '')

THROW 50001, 'Kraj nie może być pusty.', 1;

if(@city is null or @city = '')

THROW 50001, 'Miasto nie może być puste.', 1;

if(@zipCode is null or @zipCode = '')

THROW 50001, 'Kod pocztowy nie może być pusty.', 1;

if(@street is null or @street = '')

THROW 50001, 'Ulica nie może być pusta.', 1;

INSERT INTO [dbo].Clients(Id, Created, ClientInformationsId)

VALUES(@clientId, SYSDATETIME(), @clientInformationsId);

INSERT INTO [dbo].ClientInformations(Id, ClientId, IsCompany, Email, Country, City, ZipCode, Street, TaxId, CompanyName)

VALUES(@clientInformationsId, @clientId, 1, @email, @country, @city, @zipCode, @street, @taxId, @companyName);

SELECT @clientId;

END

GO

### Opis

Procedura tworząca klienta firmowego.

### Parametry:

* @companyName – nazwa firmy
* @taxId – NIP firmy
* @email – email klienta
* @country – kraj klienta
* @city – miasto klienta
* @zipCode – kod pocztowy klienta
* @street – ulica klienta

## Procedura [dbo].RegisterForConference

CREATE PROCEDURE [dbo].RegisterForConference(@clientId uniqueidentifier,

@conferenceDateId uniqueidentifier,

@participantsCount int = 1)

AS

BEGIN

DECLARE @i int = 1;

DECLARE @isCompany bit, @firstName varchar(50), @lastName varchar(50);

DECLARE @participantId uniqueidentifier = null;

DECLARE @addedParticipants table(ParticipantId uniqueidentifier, FirstName varchar(50), LastName varchar(50));

if((SELECT Count(\*) from [dbo].Clients where Id = @clientId) = 0)

THROW 50002, 'Klient o podanym id nie istnieje.', 1;

if((SELECT Count(\*) from [dbo].ConferenceDates where Id = @conferenceDateId) = 0)

THROW 50002, 'Konferencja o podanym id nie istnieje.', 1;

if((SELECT Date FROM [dbo].ConferenceDates where Id = @conferenceDateId) < SYSDATETIME())

THROW 50002, 'Nie ma możliwości na rejestracje na dzień konferencji który już się skończył lub trwa.', 1;

SELECT @isCompany = IsCompany,

@firstName = FirstName,

@lastName = LastName

FROM [dbo].Clients c join [dbo].ClientInformations ci ON c.Id = ci.ClientId

WHERE c.Id = @clientId

if(@isCompany = 0)

BEGIN

SET @participantId = newid();

INSERT INTO [dbo].Participants(Id,

ConferenceDateId,

ClientId,

RegistrationDate,

FirstName,

LastName) VALUES(@participantId,

@conferenceDateId,

@clientId,

SYSDATETIME(),

@firstName,

@lastName)

INSERT INTO @addedParticipants values(@participantId, @firstName, @lastName)

END

if(@isCompany = 1)

BEGIN

WHILE(@i <= @participantsCount)

BEGIN

SET @participantId = newid();

INSERT INTO [dbo].Participants(Id, ConferenceDateId, ClientId, RegistrationDate)

VALUES(@participantId, @conferenceDateId, @clientId, SYSDATETIME())

INSERT INTO @addedParticipants values(@participantId, null, null)

SET @i = @i + 1;

END

END

SELECT \* from @addedParticipants

END

GO

### Opis

Procedura rejestrująca klienta na konkretną konferencję danego dnia.

### Parametry:

* @clientId – unikalny identyfikator klienta
* @conferenceDateId – unikalny identyfikator daty konferencji
* @participantsCount – liczba uczestników podawana w przypadku firmy

## Procedura [dbo].AddWorkshopType

CREATE PROCEDURE [dbo].AddWorkshopType(@name varchar(100),

@description varchar(500),

@price numeric(18,2))

AS

BEGIN

DECLARE @workshopTypeId uniqueidentifier = newid();

if(@name is null or @name = '')

THROW 50001, 'Nazwa warsztatu nie może być pusta.', 1;

if(@price is null or @price <= 0)

THROW 50001, 'Nieprawidłowa cena (<= 0).', 1;

INSERT INTO [dbo].WorkshopTypes(Id, Name, Description, Price)

VALUES(@workshopTypeId, @name, @description, @price);

SELECT @workshopTypeId

END

### Opis

Procedura dodająca warsztaty.

### Parametry:

* @name – nazwa warsztatu
* @description – opis warsztatu
* @price – opcjonalna cena warsztatu

## Procedura [dbo]. RegisterConferenceDateWorkshop

CREATE PROCEDURE [dbo].RegisterConferenceDateWorkshop(@conferenceDateId uniqueidentifier,

@workshopTypeId uniqueidentifier,

@startOfWorkshop datetime,

@endOfWorkshop datetime,

@participantsLimit numeric(3,0)

)

AS

BEGIN

DECLARE @dailyConferenceWorkshopId uniqueidentifier;

DECLARE @startOfConference datetime = null;

DECLARE @conferenceDayDate datetime;

(select @startOfConference = StartOfConference,

@conferenceDayDate = Date

from [dbo].ConferenceDates cd join [dbo].Conferences c on cd.ConferenceId = c.Id

where cd.Id = @conferenceDateId)

if(@startOfConference is null or SYSDATETIME() > @startOfConference)

THROW 50001, 'Konferencja o podanym id nie istnieje lub konferencja juz się zakończyła.', 1;

if((SELECT 'EXISTS' from [dbo].WorkshopTypes where Id = @workshopTypeId) is null)

THROW 50001, 'Warsztat o podanym id nie istnieje.', 1;

if(@startOfWorkshop > @endOfWorkshop)

THROW 50001, 'Konferencja nie może się zaczynać później niż się kończy.', 1;

if(@participantsLimit <= 0)

THROW 50001, 'Limit uczestników nie może być <= 0.', 1;

if([dbo].TruncateDate(@startOfWorkshop) <> [dbo].TruncateDate(@conferenceDayDate)

or [dbo].TruncateDate(@endOfWorkshop) <> [dbo].TruncateDate(@conferenceDayDate))

THROW 50002, 'Data warsztatów nie zgadza się z data konferencji.', 1;

SET @dailyConferenceWorkshopId = newid();

INSERT INTO [dbo].DailyConferenceWorkshops(Id, ConferenceDateId, WorkshopId, ParticipantsLimit, Date, StartOfWorkshop, EndOfWorkshop)

VALUES (@dailyConferenceWorkshopId,

@conferenceDateId,

@workshopTypeId,

@participantsLimit,

@conferenceDayDate,

@startOfWorkshop,

@endOfWorkshop)

SELECT @dailyConferenceWorkshopId;

END

GO

### Opis

Procedura dodająca warsztat do konkretnej daty konferencji.

### Parametry:

* @conferenceDateId – unikalny identyfikator daty konferencji
* @workshopTypeId – unikalny identyfikator warsztatu
* @startOfWorkshop – data z czasem rozpoczęcia warsztatu
* @endOfWorkshop – data z czasem zakończenia warsztatu
* @participantsLimit – limit uczestników

## Procedura [dbo]. FillCompanyParticipantsNames

CREATE PROCEDURE [dbo].FillCompanyParticipantsNames(@participants ParticipantsNameType READONLY)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @participantId uniqueidentifier, @firstName varchar(50), @lastName varchar(50), @studentId varchar(50);

DECLARE participantsCursor CURSOR LOCAL FAST\_FORWARD FOR SELECT \* FROM @participants;

OPEN participantsCursor;

FETCH NEXT FROM participantsCursor INTO @participantId, @firstName, @lastName, @studentId;

WHILE @@FETCH\_STATUS = 0

BEGIN

UPDATE [dbo].Participants set FirstName = @firstName,

LastName = @lastName,

StudentId = @studentId where Id = @participantId;

FETCH NEXT FROM participantsCursor INTO @participantId, @firstName, @lastName, @studentId;

END

CLOSE participantsCursor;

DEALLOCATE participantsCursor;

SELECT p1.Id,

p1.ClientId,

p1.ConferenceDateId,

p1.FirstName,

p1.LastName,

p1.RegistrationDate,

p1.StudentId

FROM [dbo].Participants p1 INNER JOIN @participants p2 ON p1.Id = p2.ParticipantId -- Validation and result in one. Since no participant will be updated

-- if doesn't exist in table

END

GO

### Opis

Procedura uzupełniająca dane uczestników w przypadku klienta firmowego

### Parametry

* @participants – tabela składająca się z:

@participantId – unikalny identyfikator uczestnika

@firstName – imię uczestnika

@lastName – nazwisko uczestnika

@studentId – opcjonalny identyfikator legitymacji

## Procedura [dbo]. RegisterConferenceParticipantForWorkshop

CREATE PROCEDURE [dbo].RegisterConferenceParticipantForWorkshop(@participantId uniqueidentifier,

@dailyConferenceWorkshopId uniqueidentifier)

AS

BEGIN

DECLARE @participantWorkshopId uniqueidentifier;

IF((SELECT Count(\*) FROM [dbo].Participants WHERE Id = @participantId) = 0)

THROW 50001, 'Uczestnik o podanym id nie istnieje.', 1;

IF((SELECT Count(\*) FROM [dbo].DailyConferenceWorkshops WHERE Id = @dailyConferenceWorkshopId) = 0)

THROW 50001, 'Warsztat konferencji o podanym id nie istnieje.', 1;

IF((SELECT Count(\*) from [dbo].ParticipantWorkshops where DailyConferenceWorkshopId = @dailyConferenceWorkshopId)

>= (SELECT ParticipantsLimit from [dbo].DailyConferenceWorkshops where Id = @dailyConferenceWorkshopId))

THROW 50002, 'Limit uczestników został osiągnięty.', 1;

IF((SELECT 'Registered'

FROM [dbo].ParticipantWorkshops

WHERE ParticipantId = @participantId

AND DailyConferenceWorkshopId = @dailyConferenceWorkshopId) is not null)

THROW 50002, 'Uczestnik jest już zarejestrowany na warsztat o podanym id.', 1;

IF((Select Count(\*) FROM [dbo].ParticipantWorkshops pw

INNER JOIN (

SELECT dcw.Id, dcw.StartOfWorkshop, dcw.EndOfWorkshop

FROM [dbo].DailyConferenceWorkshops dcw

WHERE [dbo].TruncateDate(dcw.Date) = (SELECT [dbo].TruncateDate(dcw2.Date)

FROM [dbo].DailyConferenceWorkshops dcw2

WHERE dcw2.Id = @dailyConferenceWorkshopId)

) dcw

ON pw.DailyConferenceWorkshopId = dcw.Id

WHERE pw.ParticipantId = @participantId

AND (SELECT dcw3.StartOfWorkshop

FROM [dbo].DailyConferenceWorkshops dcw3

WHERE dcw3.Id = @dailyConferenceWorkshopId) BETWEEN DATEADD(SECOND,-1,dcw.StartOfWorkshop) AND DATEADD(SECOND, 1, dcw.EndOfWorkshop)) -- we don't want

-- to include border values

> 0)

THROW 50002, 'Uczestnik jest już zarejestrowany na inny warsztat równolegle odbywający się.', 1;

SET @participantWorkshopId = newid();

INSERT INTO [dbo].ParticipantWorkshops(Id, DailyConferenceWorkshopId, ParticipantId)

VALUES(@participantWorkshopId, @dailyConferenceWorkshopId, @participantId)

SELECT @participantWorkshopId;

END

GO

### Opis

Procedura rejestrująca uczestnika konferencji na dany warsztat.

### Parametry:

* @participantId – unikalny identyfikator uczestnika
* @dailyConferenceWorkshopId – unikalny identyfikator daty konferencji

## Procedura [dbo]. ClientPayment

CREATE PROCEDURE [dbo].ClientPayment(@clientId uniqueidentifier,

@conferenceId uniqueidentifier,

@payment numeric(10,2))

AS

BEGIN

DECLARE @paymentId uniqueidentifier = newid();

DECLARE @conferenceActualPrice numeric(10,2);

DECLARE @conferenceEndOfConference datetime;

DECLARE @error varchar(100);

SELECT @conferenceEndOfConference = EndOfConference FROM [dbo].Conferences WHERE Id = @conferenceId;

IF(@conferenceEndOfConference IS NULL)

THROW 50001, 'Konferencja o podanym id nie istnieje.', 1;

IF(@conferenceEndOfConference < SYSDATETIME())

THROW 50002, 'Nie można opłacić konferencji która już się ukończyła', 1;

IF(@payment <=0)

THROW 50002, 'Błędna kwota.', 1;

EXEC @conferenceActualPrice = [dbo].GetClientPrice @clientId = @clientId, @conferenceId = @conferenceId;

IF(@payment != @conferenceActualPrice)

BEGIN

SET @error = 'Błędna kwota konferencji. Koszt całości to ' + CAST(@conferenceActualPrice AS VARCHAR(15));

THROW 50002, @error, 1;

END

INSERT INTO [dbo].Payments(Id, ClientId, Payment, PaymentDate)

VALUES(@paymentId, @clientId, @payment, SYSDATETIME());

SELECT @paymentId;

END;

GO

### Opis

Procedura finalizująca płatność klienta.

### Parametry:

* @clientId – unikalny identyfikator klienta
* @conferenceId – unikalny identyfikator konferencji
* @payment – wartość płatności (musi zgadzać się z całkowitym kosztem konferencji dla klienta)

## Procedura [dbo]. SetConferenceDatePrice

CREATE PROCEDURE [dbo].SetConferenceDatePrice(@conferenceDateId uniqueidentifier,

@price numeric(10,2))

AS

BEGIN

UPDATE [dbo].ConferenceDates set DayBasePrice = @price WHERE Id = @conferenceDateId;

SELECT \* FROM [dbo].ConferenceDates WHERE Id = @conferenceDateId;

END

GO

### Opis

Procedura ustawiająca bazową cenę daty konferencji.

### Parametry:

* @conferenceDateId – unikalny identyfikator daty konferencji
* @price – cena

## Procedura [dbo]. SetConferenceDatePriceThreshold

CREATE PROCEDURE [dbo].SetConferenceDatePriceThreshold(@conferenceDateId uniqueidentifier,

@thresholdDate datetime,

@price numeric(10,2))

AS

BEGIN

DECLARE @thresholdId uniqueidentifier = newid();

INSERT INTO [dbo].ConferenceDatesPriceThresholds(Id, ConferenceDateId, ThresholdDate, Price)

VALUES(@thresholdId, @conferenceDateId, @thresholdDate, @price);

END

GO

### Opis

Procedura ustawiająca próg cenowy dla konkretnej daty konferencji

### Parametry:

* @conferenceDateId – unikalny identyfikator daty konferencji
* @thresholdDate – data progu cenowego
* @price – wartość progu cenowego

# Opis fukcji

## Funkcja [dbo].TruncateDate

CREATE FUNCTION [dbo].TruncateDate(@date datetime)

returns datetime

as

begin

declare @truncatedDate datetime = (SELECT CAST(@date AS DATE));

return @truncatedDate;

end;

GO

### Opis

Funkcja zwracająca datę bez czasu.

### Parametry:

* @date – data

## Funkcja [dbo].GetConferencePrice

CREATE FUNCTION [dbo].GetConferencePrice(@conferenceDateId uniqueidentifier)

RETURNS numeric(10,2)

AS

BEGIN

DECLARE @dayBasePrice numeric(10,2);

DECLARE @actualPrice numeric(10,2);

SELECT @dayBasePrice = DayBasePrice

FROM [dbo].ConferenceDates

WHERE Id = @conferenceDateId;

SELECT @actualPrice = cdpt.Price

FROM [dbo].ConferenceDatesPriceThresholds cdpt

WHERE cdpt.ConferenceDateId = @conferenceDateId

AND cdpt.ThresholdDate = (SELECT MAX(cdpt2.ThresholdDate)

FROM [dbo].ConferenceDatesPriceThresholds cdpt2

WHERE cdpt2.ConferenceDateId = @conferenceDateId

AND [dbo].TruncateDate(SYSDATETIME()) > [dbo].TruncateDate(cdpt2.ThresholdDate));

IF(@actualPrice is null)

return @dayBasePrice;

return @actualPrice

END;

GO

### Opis

Funkcja zwracająca cenę dla konferencji na aktualny dzień (data systemowa). Funkcja uwzględnia progi cenowe.

### Parametry:

* @conferenceDateId – unikalny identyfikator daty konferencji

## Funkcja [dbo].GetConferenceDatePriceForClient

CREATE FUNCTION [dbo].GetConferenceDatePriceForClient(@clientId uniqueidentifier,

@conferenceDateId uniqueidentifier)

RETURNS NUMERIC(10,2)

AS

BEGIN

DECLARE @totalConferenceDatePrice numeric(10,2);

DECLARE @totalWorkshopsPrice numeric(10,2);

DECLARE @groupedConferenceDateId uniqueidentifier;

DECLARE @studentPercentageDiscount int = 0;

EXEC @studentPercentageDiscount = [dbo].GetIntSetting @settingName = 'StudentPercentageDiscount';

SELECT @totalWorkshopsPrice = SUM(totalPrices.TotalWorkshopsPrice),

@totalConferenceDatePrice = SUM(totalPrices.ConferenceDatePrice)

FROM (SELECT p.Id ParticipantId,

SUM(ISNULL(CASE

WHEN p.StudentId is null

THEN wt.Price

ELSE wt.Price - wt.Price \* (CAST(@studentPercentageDiscount as numeric(3,0)) / 100)

END, 0)) TotalWorkshopsPrice,

[dbo].GetConferencePrice(cd.Id) ConferenceDatePrice

FROM [dbo].ParticipantWorkshops pw

JOIN [dbo].DailyConferenceWorkshops dcw on pw.DailyConferenceWorkshopId = dcw.Id

JOIN [dbo].WorkshopTypes wt on wt.Id = dcw.WorkshopId

JOIN [dbo].ConferenceDates cd on cd.Id = dcw.ConferenceDateId

JOIN [dbo].Participants p on p.Id = pw.ParticipantId

WHERE p.ClientId = @clientId

AND dcw.ConferenceDateId = @conferenceDateId

GROUP BY p.Id, cd.Id) as totalPrices

RETURN CAST(ROUND(ISNULL(@totalConferenceDatePrice,0) + ISNULL(@totalWorkshopsPrice,0),2) AS numeric(10,2));

END;

GO

### Opis

Funkcja zwracająca całkowity koszt daty konferencji dla klienta

### Parametry:

* @clientId – unikalny identyfikator klienta
* @conferenceDateId – unikalny identyfikator daty konferencji

## Funkcja [dbo].GetClientPrice

CREATE FUNCTION [dbo].GetClientPrice(@clientId uniqueidentifier,

@conferenceId uniqueidentifier)

RETURNS numeric(10,2)

AS

BEGIN

DECLARE @totalPrice numeric(10,2);

SELECT @totalPrice = SUM(ISNULL([dbo].GetConferenceDatePriceForClient(@clientId, cd.Id),0))

FROM [dbo].ConferenceDates cd

WHERE cd.ConferenceId = @conferenceId

RETURN @totalPrice;

END

GO

### Opis

Funkcja zwracająca całkowity koszt konferencji dla klienta. Sumowane są wszystkie daty konferencji, na które klient jest zarejestrowany (i uczestnicy).

### Parametry:

* @clientId – unikalny identyfikator klienta
* @conferenceId – unikalny identyfikator konferencji

## Funkcja [dbo].GetIntSetting

CREATE FUNCTION [dbo].GetIntSetting(@settingName varchar(100))

RETURNS INT

AS

BEGIN

DECLARE @intValue int;

SELECT TOP(1) @intValue = IntValue FROM [dbo].Settings WHERE Name = @settingName;

return @intValue;

END

GO

### Opis

Funkcja zwracająca ustawienie jako int.

### Parametry:

* @settingName – unikalna nazwa ustawienia

## Funkcja [dbo].GetVarcharSetting

CREATE FUNCTION [dbo].GetVarcharSetting(@settingName varchar(100))

RETURNS VARCHAR(200)

AS

BEGIN

DECLARE @varcharValue varchar(200);

SELECT TOP(1) @varcharValue = VarcharValue FROM [dbo].Settings WHERE Name = @settingName;

return @varcharValue;

END

GO

### Opis

Funkcja zwracająca ustawienie jako varchar.

### Parametry:

* @settingName – unikalna nazwa ustawienia

## Funkcja [dbo].GetNumericSetting

CREATE FUNCTION [dbo].GetNumericSetting(@settingName varchar(100))

RETURNS NUMERIC(38,23)

AS

BEGIN

DECLARE @numericValue numeric(38,23);

SELECT TOP(1) @numericValue = NumericValue FROM [dbo].Settings WHERE Name = @settingName;

return @numericValue;

END

GO

### Opis

Funkcja zwracająca ustawienie jako numeric.

### Parametry:

* @settingName – unikalna nazwa ustawienia

## Funkcja [dbo].GetDatetimeSetting

CREATE FUNCTION [dbo].GetDatetimeSetting(@settingName varchar(100))

RETURNS DATETIME

AS

BEGIN

DECLARE @datetimeValue datetime;

SELECT TOP(1) @datetimeValue = DatetimeValue FROM [dbo].Settings WHERE Name = @settingName;

return @datetimeValue;

END

GO

### Opis

Funkcja zwracająca ustawienie jako datetime.

### Parametry:

* @settingName – unikalna nazwa ustawienia

# Widoki

## Widok [dbo].GetConferenceParticipants

CREATE VIEW [dbo].GetConferenceParticipants

AS

SELECT cd.ConferenceId, cd.Id ConferenceDateId, p.Id ParticipantId, p.FirstName, p.LastName, p.ClientId, p.StudentId

FROM [dbo].ConferenceDates cd

JOIN [dbo].DailyConferenceWorkshops dcw on dcw.ConferenceDateId = cd.Id

JOIN [dbo].Participants p on p.ConferenceDateId = dcw.ConferenceDateId

GO

### Opis

Widok zwracający wszystkich zarejestrowanych uczestników konferencji.

## Widok [dbo].GetClientsWithNotFilledParticipants

CREATE VIEW [dbo].GetClientsWithNotFilledParticipants

AS

SELECT c.Id, Count(\*) NotFilledParticipantsCount

FROM [dbo].Clients c LEFT JOIN [dbo].Participants p on c.Id = p.ClientId

WHERE p.FirstName is null or p.LastName is null

GROUP BY c.Id

GO

### Opis

Widok zwracający wszystkich uczestników zarejestrowanych na klienta, którzy nie mają wypełnionych danych (miejsca są zarezerwowane).

## Widok [dbo].GetClientsPayments

CREATE VIEW [dbo].GetClientsPayments

AS

SELECT \* FROM [dbo].Payments

GO

### Opis

Widok zwracający wszystkie płatności.