

Procedury

Add_course_module_async - PS

Procedura, która dodaje dany moduł z danymi do tabeli z modułami prowadzonymi online-asynchronicznie.

```
CREATE procedure add_course_module_async
    @ModuleID int,
    @RecordingLink nvarchar(100)
as begin
    begin try
        -- Sprawdzenie czy dany moduł istnieje
        if not exists(select 1 from Modules where ModuleID = @ModuleID)
        begin
            throw 50000, 'Moduł o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie poprawności podanego typu
        if not exists(select 1 from Modules inner join Types on Modules.TypeID = Types.TypeID
            where TypeName = 'Online Async' and ModuleID = @ModuleID)
        begin
            throw 50001, 'Podany moduł nie jest typu online-asynchronicznie', 1;
        end

        -- Dodanie danych
        insert Online_Async_Modules(ModuleID, RecordingLink)
        values (@ModuleID, @RecordingLink)
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;
```

Add_course_module_in_person - PS

Procedura, która dodaje informacje o module do tabeli z modułami stacjonarnymi.

```
CREATE procedure add_course_module_in_person
    @ModuleID int,
    @Classroom int,
    @TranslatorID int,
    @LanguageID int,
    @Limit int
as begin
    begin try
        -- Sprawdzenie czy dany moduł istnieje
        if not exists(select 1 from Modules where ModuleID = @ModuleID)
        begin
            throw 50000, 'Moduł o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie poprawności podanego typu
        if not exists(select 1 from Modules inner join Types on Modules.TypeID = Types.TypeID
            where TypeName = 'In-person' and ModuleID = @ModuleID)
        begin
            throw 50001, 'Podany moduł nie jest typu stacjonarnego', 1;
        end

        -- Sprawdzenie czy dany tłumacz istnieje
        if not exists(select 1 from Translators where TranslatorID = @TranslatorID) and @TranslatorID is not null
        begin
            throw 50002, 'Tłumacz o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dany język istnieje
        if not exists(select 1 from Languages where LanguageID = @LanguageID)
        begin
            throw 50003, 'Język o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy limit jest poprawnie wpisany
        if @Limit <= 0
        begin
            throw 50004, 'Limit nie może być wartością mniejszą bądź równą 0', 1;
        end
    end try
end;
```

```

-- Sprawdzenie czy podana sala jest wolna w tym okresie
declare @DateAndBeginningTime datetime = (select DateAndBeginningTime from Modules where ModuleID = @ModuleID)
declare @Duration time(0) = (select Duration from Modules where ModuleID = @ModuleID)

if dbo.check_classroom_availability(@Classroom, @DateAndBeginningTime, @Duration) = cast(1 as bit)
begin
    throw 50005, 'Sala w okresie trwania modułu nie dostępna', 1;
end

-- Sprawdzenie dostępności tłumacza
if dbo.check_translator_availability(@TranslatorID, @DateAndBeginningTime, @Duration) = cast(1 as bit)
begin
    throw 50006, 'Tłumacz w okresie danego modułu jest nie dostępny', 1;
end

-- Dodanie danych
insert In_person_Modules(ModuleID, Classroom, TranslatorID, LanguageID, Limit)
values (@ModuleID, @Classroom, @TranslatorID, @LanguageID, @Limit)
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Add_course_module_sync - PS

Procedura, która dodaje moduł do tabeli z podułami prowadzonymi online-synchronicznie.

```

CREATE procedure add_course_module_sync
    @ModuleID int,
    @MeetingLink nvarchar(100),
    @RecordingLink nvarchar(100),
    @TranslatorID int,
    @LanguageID int
as begin
    begin try
        -- Sprawdzenie czy dany moduł istnieje
        if not exists(select 1 from Modules where ModuleID = @ModuleID)
        begin
            throw 50000, 'Moduł o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie poprawności podanego typu
        if not exists(select 1 from Modules inner join Types on Modules.TypeID = Types.TypeID
            where TypeName = 'Online Sync' and ModuleID = @ModuleID)
        begin
            throw 50001, 'Podany moduł nie jest typu online-synchronicznie', 1;
        end

        -- Sprawdzenie czy dany tłumacz istnieje
        if not exists(select 1 from Translators where TranslatorID = @TranslatorID) and @TranslatorID is not null
        begin
            throw 50002, 'Tłumacz o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dany język istnieje
        if not exists(select 1 from Languages where LanguageID = @LanguageID)
        begin
            throw 50003, 'Język o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie dostępności tłumacza
        declare @DateAndBeginningTime datetime = (select DateAndBeginningTime from Modules where ModuleID = @ModuleID)
        declare @Duration time(0) = (select Duration from Modules where ModuleID = @ModuleID)

        if dbo.check_translator_availability(@TranslatorID, @DateAndBeginningTime, @Duration) = cast(1 as bit)
        begin
            throw 50004, 'Tłumacz w okresie danego modułu jest nie dostępny', 1;
        end

        -- Dodanie danych
        insert Online_Sync_Modules(ModuleID, MeetingLink, RecordingLink, TranslatorID, LanguageID)
        values (@ModuleID, @MeetingLink, @RecordingLink, @TranslatorID, @LanguageID)
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;

```

Add_course_module - PS

Procedura służąca do dodawania modułów do danego kursu.

```
CREATE procedure add_course_modules
    @TeacherID int,
    @CourseID int,
    @Name nvarchar(50),
    @Description nvarchar(max),
    @DateAndBeginningTime datetime,
    @Duration time(0),
    @TypeID int
as begin
    begin try
        -- Sprawdzenie poprawności wpisanych danych
        if not exists(select 1 from Employees where EmployeeID = @TeacherID)
        begin
            throw 50001, 'Nauczyciel o podanym ID nie istnieje', 1;
        end

        if not exists(select 1 from Courses where CourseID = @CourseID)
        begin
            throw 50002, 'Kurs o podanym ID nie istnieje', 1;
        end

        if not exists(select 1 from Types where TypeID = @TypeID)
        begin
            throw 50003, 'Typ o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy nauczyciel nie ma w tym czasie innych zajęć
        if dbo.check_teachers_availability(@TeacherID, @DateAndBeginningTime, @Duration) = cast(1 as bit)
        begin
            throw 50004, 'Podany nauczyciel ma w tym czasie inne zajęcia', 1;
        end

        -- Sprawdzenie czy moduł nakłada się z innym w tym samym kursie
        declare @EndDate DATETIME = DATEADD(MINUTE, DATEDIFF(MINUTE, 0, @Duration), @DateAndBeginningTime);
        IF EXISTS (
            SELECT 1
            FROM Modules
            WHERE CourseID = @CourseID
            AND (
                -- Sprawdzenie, czy istnieje moduł, którego czas pokrywa się z nowym modułem
                (@DateAndBeginningTime < DateAndBeginningTime AND @EndDate > DateAndBeginningTime)
                OR
                (@DateAndBeginningTime >= DateAndBeginningTime AND @DateAndBeginningTime < DATEADD(MINUTE, DATEDIFF(MINUTE, 0, Duration),
DateAndBeginningTime))
                OR
                (@EndDate > DateAndBeginningTime AND @EndDate <= DATEADD(MINUTE, DATEDIFF(MINUTE, 0, Duration), DateAndBeginningTime))
            )
        )
        BEGIN
            THROW 50005, 'Moduł nakłada się na istniejący moduł w tym kursie', 1;
        END

        -- W innych przypadkach można dodać moduł
        insert Modules (TeacherID, CourseID, Name, Description, DateAndBeginningTime, Duration, TypeID)
        values (@TeacherID, @CourseID, @Name, @Description, @DateAndBeginningTime, @Duration, @TypeID)
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;
```

Add_course - PS

Procedura służąca do dodawania kursu do oferty.

```
CREATE procedure add_course
    @CoordinatorID int,
    @Name nvarchar(30),
    @Description nvarchar(max),
    @StartDate date,
    @EndDate date,
    @Price money,
    @Status bit
```

```

as begin
begin try
begin transaction;

-- Sprawdzenie poprawności wpisywanych danych
if not exists(select 1 from Employees where EmployeeID = @CoordinatorID and
              PositionID = 4)

begin
throw 50001, 'Koordynator o danym ID nie istnieje lub nie jest koordynatorem kursów', 1;
end

if @Price < 0
begin
throw 50002, 'Cena nie może być mniejsza od 0', 1;
end

if @StartDate >= @EndDate
begin
throw 50003, 'Nie poprawnie wpisane daty', 1;
end

-- W innym przypadku możemy dodać
-- Rezerwacja ID w produktach
declare @NewProductID int;
declare @CategoryID int = (select CategoryID from Categories where Name = 'Course')

insert into Products (CategoryID, Status)
values (@CategoryID, @Status)

-- Pobranie ID po dodaniu do produktów
set @NewProductID = SCOPE_IDENTITY();

-- Dodanie do tabeli ze Wbinarami
insert Courses (CourseID, CoordinatorID, Name, Description, StartDate, EndDate, Price)
values (@NewProductID, @CoordinatorID, @Name, @Description, @StartDate, @EndDate, @Price)

commit transaction;
end try
begin catch
-- Wycofanie transakcji w przypadku błędu
if @@TRANCOUNT > 0
begin
rollback transaction;
end;

-- Przerzucenie ERRORa dalej
throw;
end catch
end;

```

Add_employee - PS

Procedura służąca do dodania nowego pracownika do systemu.

```

CREATE procedure add_employee
@FirstName nvarchar(50),
@LastName nvarchar(50),
@Phone varchar(15),
@email nvarchar(50),
@Address nvarchar(50),
@City nvarchar(30),
@PostalCode varchar(10),
@PositionID int
as
begin

begin try
-- Sprawdzenie czy dana pozycja istnieje
if not exists(select 1 from Employees_Positions where PositionID = @PositionID)
begin
throw 51000, 'Pozycja nie istnieje ', 1;
end

-- Wstawienie danych to tabeli
insert Employees (FirstName, LastName, Phone, Email, Address, City, PostalCode, PositionID)
values (@FirstName, @LastName, @Phone, @Email, @Address,
        @City, @PostalCode, @PositionID);
end try
begin catch
-- Przerzucenie błędu dalej
throw;
end

```

```
        end catch
    end;
```

Add_translator - PS

Procedura służąca na dodanie nowego tłumacza do systemu.

```
CREATE procedure add_translator
    @FirstName nvarchar(50),
    @LastName nvarchar(50),
    @Phone varchar(15),
    @Email nvarchar(50),
    @Address nvarchar(50),
    @City nvarchar(30),
    @PostalCode varchar(10)
as
begin
    begin try
        -- Wstawienie danych to tabeli
        insert Translators (FirstName, LastName, Phone, Email, Address, City, PostalCode)
        values (@FirstName, @LastName, @Phone, @Email, @Address,
            @City, @PostalCode);
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;
```

Add_user - PS

Procedura służąca do dodania nowego użytkownika do systemu.

```
create procedure add_user
    @FirstName nvarchar(50),
    @LastName nvarchar(50),
    @Phone varchar(15),
    @Email nvarchar(50),
    @Address nvarchar(50),
    @City nvarchar(30),
    @PostalCode varchar(10)
as
begin
    begin try
        -- Wstawienie danych to tabeli
        insert Users (FirstName, LastName, Phone, Email, Address, City, PostalCode)
        values (@FirstName, @LastName, @Phone, @Email, @Address,
            @City, @PostalCode);
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;
```

Add_webinar - PS

Procedura służąca do dodania nowych webinarów do oferty.

```
CREATE procedure add_webinar
    @Name nvarchar(30),
    @Description nvarchar(max),
    @DateAndBeginningTime datetime,
    @Duration time(0),
    @CoordinatorID int,
    @TeacherID int,
    @TranslatorID int,
    @Price int,
    @LanguageID int,
    @RecordingLink nvarchar(100),
    @MeetingLink nvarchar(100),
    @Status bit
as begin
    begin try
```

```

begin transaction;

-- Sprawdzenie poprawności wpisywanych danych
if not exists(select 1 from Employees where EmployeeID = @CoordinatorID and
              PositionID = 2)

begin
    throw 50001, 'Koordynator o danym ID nie istnieje lub nie jest koordynatorem webinarów', 1;
end

if not exists(select 1 from Employees where EmployeeID = @TeacherID)
begin
    throw 50002, 'Nauczyciel o danym ID nie istnieje', 1;
end

if @Price < 0
begin
    throw 50003, 'Cena nie może być mniejsza od 0', 1;
end

if not exists(select 1 from Languages where LanguageID = @LanguageID)
begin
    throw 50004, 'Język o danym ID nie istnieje', 1;
end

if not exists(select 1 from Translators where @TranslatorID = TranslatorID) and @TranslatorID IS NOT NULL
begin
    throw 50005, 'Tłumacz o danym ID nie istnieje', 1;
end

if dbo.check_translator_language(@TranslatorID, @LanguageID) = cast(0 as bit)
begin
    throw 50006, 'Para tłumacz-język nie istnieje', 1;
end

-- W innym przypadku możemy dodać
-- Rezerwacja ID w produktach
declare @NewProductID int;
declare @CategoryID int = (select CategoryID from Categories where Name = 'Webinar')

insert Products (CategoryID, Status)
values (@CategoryID, @Status)

-- Pobranie ID po dodaniu do produktów
set @NewProductID = SCOPE_IDENTITY();

-- Dodanie do tabeli ze Webinarami
insert Webinars (WebinarID, Name, Description, DateAndBeginningTime, Duration, TeacherID, TranslatorID, Price, LanguageID,
RecordingLink, MeetingLink, CoordinatorID)
values (@NewProductID, @Name, @Description, @DateAndBeginningTime, @Duration, @TeacherID, @TranslatorID, @Price, @LanguageID,
@RecordingLink, @MeetingLink, @CoordinatorID)

commit transaction;
end try
begin catch
    if @@TRANCOUNT > 0
    begin
        rollback transaction;
    end;

    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Assign_language_to_translator - PS

Procedura służąca do dodania języka do tłumacza (języka, którego tłumaczy).

```

CREATE procedure assign_translator_to_languages
    @TranslatorID int,
    @LanguageID int
as begin
    begin try
        -- Sprawdzenie czy język o danym ID istnieje
        if not exists(select 1 from Languages where LanguageID = @LanguageID)
        begin
            throw 50001, 'Język nie istnieje', 1;
        end

        -- Sprawdzenie czy tłumacz o danym ID istnieje
        if not exists(select 1 from Translators where TranslatorID = @TranslatorID)

```

```

begin
    throw 50002, 'Tłumacz nie istnieje', 1;
end

-- Sprawdzenie czy taki wpis już istnieje
if dbo.check_translator_language(@TranslatorID, @LanguageID) = cast(1 as bit)
begin
    throw 50003, 'Taka para już istnieje', 1;
end

-- W innych przypadkach dodajemy parę
insert Translators_Languages (TranslatorID, LanguageID)
values (@TranslatorID, @LanguageID)
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Delete_language_from_translator - PS

Procedura służąca do usunięcia języka tłumaczowi (języka, którego tłumaczył).

```

CREATE procedure delete_language_from_translator
    @TranslatorID int,
    @LanguageID int
as begin
    begin try
        -- Sprawdzenie czy tłumacz istnieje
        if not exists(select 1 from Translators where TranslatorID = @TranslatorID)
        begin
            throw 50001, 'Tłumacz o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy język istnieje
        if not exists(select 1 from Languages where LanguageID = @LanguageID)
        begin
            throw 50002, 'Język o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dana para istnieje
        if dbo.check_translator_language(@TranslatorID, @LanguageID) = cast(0 as bit)
        begin
            throw 50003, 'Taka para nie istnieje', 2;
        end

        -- W innym przypadku ją usuwamy
        delete from Translators_Languages
        where TranslatorID = @TranslatorID and LanguageID = @LanguageID;
    end try
    begin catch
        -- Przerzucenie błędu dalej
        throw;
    end catch
end;

```

Update_module_type - PS

Procedura umożliwiająca edytowanie typu danego modułu w razie pomyłki. Jeżeli zostały już dodane dodatkowe dane do którejś tabeli z kategorii to dane stamtąd są usuwane.

```

CREATE procedure update_module_type
    @ModuleID int,
    @TypeID int
as begin
    begin try
        -- Sprawdzenie czy moduł istnieje
        if not exists(select 1 from Modules where ModuleID = @ModuleID)
        begin
            throw 50000, 'Moduł o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dany typ istnieje
        if not exists(select 1 from Types where TypeID = @TypeID)
        begin
            throw 50001, 'Podany typ nie istnieje', 1;
        end
    end try

```

```

-- Sprawdzenie czy nie zmieniasz na ten sam typ
if exists(select 1 from Modules where ModuleID = @ModuleID and TypeID = @TypeID)
begin
    throw 50002, 'Zmieniasz typ modułu na ten sam typ', 1;
end

declare @OldTypeID int = (select TypeID from Modules where ModuleID = @ModuleID)
declare @OldTypeName varchar(20);
set @OldTypeName = (select TypeName from Types where TypeID = @OldTypeID)

-- Aktualizacja typu
update Modules
set TypeID = @TypeID
where ModuleID = @ModuleID

-- Usunięcie danych ze starego typu
if @OldTypeName = 'In-person'
begin
    if exists(select 1 from In_person_Modules where ModuleID = @ModuleID)
    begin
        -- Informacje zostały dodane i można usunąć informacje
        delete from In_person_Modules
        where ModuleID = @ModuleID
    end
end

if @OldTypeName = 'Online Sync'
begin
    if exists(select 1 from Online_Sync_Modules where ModuleID = @ModuleID)
    begin
        -- Informacje zostały dodane i można usunąć informacje
        delete from Online_Sync_Modules
        where ModuleID = @ModuleID
    end
end

if @OldTypeName = 'Online Async'
begin
    if exists(select 1 from Online_Async_Modules where ModuleID = @ModuleID)
    begin
        -- Informacje zostały dodane i można usunąć informacje
        delete from Online_Async_Modules
        where ModuleID = @ModuleID
    end
end
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Update_recordinglink_module_sync - PS

Procedura służąca do dodania linku do nagrania w modułach prowadzonych online-synchronicznie.

```

create procedure update_recordinglink_module_sync
    @ModuleID int,
    @RecordingLink nvarchar(100)
as begin
    begin try
        -- Sprawdzenie czy dany moduł istnieje
        if not exists(select 1 from Modules where ModuleID = @ModuleID)
        begin
            throw 50001, 'Podany moduł nie istnieje', 1;
        end

        -- Sprawdzenie czy moduł został dodany do tabeli z online-synchronicznie
        if not exists(select 1 from Online_Sync_Modules where ModuleID = @ModuleID)
        begin
            throw 50002, 'Moduł nie został dodany do modułów synchronicznych', 1;
        end

        -- Zaktualizowanie linku do nagrania
        update Online_Sync_Modules
        set RecordingLink = @RecordingLink
        where ModuleID = @ModuleID
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
    end catch
end;

```



```
        throw;  
    end catch  
end;
```

Update_recording_webinar - PS

Procedura służąca do dodania linku do nagrania w webinarach.

```
create procedure update_recordinglink_webinar  
    @WebinarID int,  
    @RecordingLink nvarchar(100)  
as begin  
    begin try  
        -- Sprawdzenie czy dany webinar istnieje  
        if not exists(select 1 from Webinars where WebinarID = @WebinarID)  
        begin  
            throw 50001, 'Podany webinar nie istnieje', 1;  
        end  
  
        -- Zaktualizowanie linku do nagrania  
        update Webinars  
        set RecordingLink = @RecordingLink  
        where WebinarID = @WebinarID  
    end try  
    begin catch  
        -- Przerzucenie ERRORa dalej  
        throw;  
    end catch  
end;
```

Add_meeting - MS

Procedura służy do dodania spotkania studyjnego.

```
create procedure add_meeting(  
    @TeacherID int,  
    @SubjectID int,  
    @ReunionID int,  
    @DateAndBeginningTime datetime,  
    @Duration time(0),  
    @Price money,  
    @TypeID int,  
    @Status bit  
)  
as begin  
    begin try  
        begin transaction;  
  
        -- Sprawdzenie poprawności wpisanych danych  
        if not exists(select 1 from Employees where EmployeeID = @TeacherID)  
        begin  
            throw 50001, 'Nauczyciel o danym ID nie istnieje', 1;  
        end  
  
        if not exists(select 1 from Subjects where SubjectID = @SubjectID)  
        begin  
            throw 50002, 'Przedmiot o danym ID nie istnieje', 1;  
        end  
  
        if not exists(select 1 from Studies_Reunion where ReunionID = @ReunionID)  
        begin  
            throw 50003, 'Zjazd o danym ID nie istnieje', 1;  
        end  
  
        if @Price < 0  
        begin  
            throw 50004, 'Cena nie może być mniejsza od 0', 1;  
        end  
  
        -- Rezerwacja ID w produktach  
        declare @NewProductID int;  
        declare @CategoryID int = (select CategoryID from Categories where Name = 'Meeting')  
  
        insert Products (CategoryID, Status)  
        values (@CategoryID, @Status)  
  
        -- Pobranie ID po dodaniu do produktów
```

```

set @NewProductID = SCOPE_IDENTITY();

-- Wstawienie danych to tabeli
insert Meetings (MeetingID, TeacherID, SubjectID, ReunionID, DateAndBeginningTime, Duration, Price,TypeID)
values ( @NewProductID,@TeacherID, @SubjectID, @ReunionID, @DateAndBeginningTime, @Duration, @Price, @TypeID);

commit transaction;
end try
begin catch
-- Wycofanie transakcji w przypadku błędu
if @@TRANCOUNT > 0
begin
rollback transaction;
end;

-- Przerzucenie ERRORa dalej
throw;
end catch
end;

```

Add_meeting_async - MS

Procedura służy do dodawania informacji o spotkaniu online-asynchronicznym do tabeli z tymi spotkaniami.

```

CREATE procedure add_meeting_async
@MeetingID int,
@RecordingLink nvarchar(100)
as begin
begin try
-- Sprawdzenie czy dane spotkanie istnieje
if not exists(select 1 from Meetings where MeetingID = @MeetingID)
begin
throw 50000, 'Spotkanie o podanym ID nie istnieje', 1;
end

-- Sprawdzenie poprawności podanego typu
if not exists(select 1 from Meetings inner join Types on Meetings.TypeID = Types.TypeID
where TypeName = 'Online Async' and MeetingID = @MeetingID)
begin
throw 50001, 'Podane spotkanie nie jest typu online-asynchronicznie', 1;
end

-- Dodanie danych do tabeli
insert Online_Async_Meetings(MeetingID, RecordingLink)
values (@MeetingID, @RecordingLink)
end try
begin catch
-- Przerzucenie ERRORa dalej
throw;
end catch
end;

```

Add_meeting_in_person - MS

Procedura służy do dodawania informacji o spotkaniu stacjonarnym do tabeli z tymi spotkaniami.

```

CREATE procedure add_meeting_in_person
@MeetingID int,
@Classroom int,
@TranslatorID int,
@LanguageID int,
@Limit int
as begin
begin try
-- Sprawdzenie czy dane spotkanie istnieje
if not exists(select 1 from Meetings where MeetingID = @MeetingID)
begin
throw 50000, 'Spotkanie o podanym ID nie istnieje', 1;
end

-- Sprawdzenie poprawności podanego typu
if not exists(select 1 from Meetings inner join Types on Meetings.TypeID = Types.TypeID
where TypeName = 'In-person' and MeetingID = @MeetingID)
begin
throw 50001, 'Podane spotkanie nie jest typu stacjonarnego', 1;
end

```

```

-- Sprawdzenie czy dany tłumacz istnieje
if not exists(select 1 from Translators where TranslatorID = @TranslatorID) and @TranslatorID is not null
begin
    throw 50002, 'Tłumacz o podanym ID nie istnieje', 1;
end

-- Sprawdzenie czy dany język istnieje
if not exists(select 1 from Languages where LanguageID = @LanguageID)
begin
    throw 50003, 'Język o podanym ID nie istnieje', 1;
end

-- Sprawdzenie czy limit jest poprawnie wpisany
if @Limit <= 0
begin
    throw 50004, 'Limit nie może być wartością mniejszą bądź równą 0', 1;
end

-- Sprawdzenie dostępności tłumacza
declare @DateAndBeginningTime datetime = (select DateAndBeginningTime from Meetings where MeetingID = @MeetingID);
declare @duration time(0) = (select Duration from Meetings where MeetingID = @MeetingID);

if dbo.check_translator_availability(@TranslatorID, @DateAndBeginningTime, @Duration) = cast(1 as bit)
begin
    throw 50006, 'Tłumacz w okresie trwania danego spotkania jest nie dostępny', 1;
end

-- Sprawdzenie czy dana sala jest dostępna
if dbo.check_classroom_availability(@Classroom, @DateAndBeginningTime, @Duration) = cast(1 as bit)
begin
    throw 50007, 'Sala w danym terminie nie jest dostępna', 1;
end

-- Dodanie danych
insert In_person_Meetings(MeetingID, Classroom, TranslatorID, LanguageID, Limit)
values (@MeetingID, @Classroom, @TranslatorID, @LanguageID, @Limit)
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Add_meeting_sync - MS

Procedura służy do dodania informacji o spotkaniu stacjonarnym do tabeli z tymi spotkaniami.

```

CREATE procedure add_meeting_sync
    @MeetingID int,
    @MeetingLink nvarchar(100),
    @RecordingLink nvarchar(100),
    @TranslatorID int,
    @LanguageID int
as begin
    begin try
        -- Sprawdzenie czy dane spotkanie istnieje
        if not exists(select 1 from Meetings where MeetingID = @MeetingID)
        begin
            throw 50000, 'Spotkanie o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie poprawności podanego typu
        if not exists(select 1 from Meetings inner join Types on Meetings.TypeID = Types.TypeID
            where TypeName = 'Online Sync' and MeetingID = @MeetingID)
        begin
            throw 50001, 'Podane spotkanie nie jest typu online-synchronicznie', 1;
        end

        -- Sprawdzenie czy dany tłumacz istnieje
        if not exists(select 1 from Translators where TranslatorID = @TranslatorID) and @TranslatorID is not null
        begin
            throw 50002, 'Tłumacz o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dany język istnieje
        if not exists(select 1 from Languages where LanguageID = @LanguageID)
        begin
            throw 50003, 'Język o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie dostępności tłumacza
    end try

```

```

declare @DateAndBeginningTime datetime = (select DateAndBeginningTime from Meetings where MeetingID = @MeetingID);
declare @duration time(0) = (select Duration from Meetings where MeetingID = @MeetingID);

if dbo.check_translator_availability(@TranslatorID, @DateAndBeginningTime, @Duration) = cast(1 as bit)
begin
    throw 50004, 'Tłumacz w okresie danego spotkania jest niedostępny', 1;
end

-- Dodanie danych
insert Online_Sync_Meetings(MeetingID, MeetingLink, RecordingLink, TranslatorID, LanguageID)
values (@MeetingID, @MeetingLink, @RecordingLink, @TranslatorID, @LanguageID)
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Add_reunion - PS

Procedura służy do dodawania zjazdu do danych studiów wraz z podaniem jego czasu odbycia się.

```

create procedure add_reunion
    @StudiesID int,
    @StartDate date,
    @EndDate date,
    @Price money
as begin
    begin try
        -- Sprawdzenie poprawności wpisanych danych
        if not exists(select 1 from Studies where StudiesID = @StudiesID)
        begin
            throw 50001, 'Studia o podanym ID nie istnieją', 1;
        end

        if @StartDate >= @EndDate
        begin
            throw 50002, 'Data startowa nie może być późniejsza niż data końca', 1;
        end

        if @Price <= 0
        begin
            throw 50003, 'Cena za zjazd nie może być ujemna lub równa 0', 1;
        end

        insert Studies_Reunion(StudiesID, StartDate, EndDate, Price)
        values (@StudiesID, @StartDate, @EndDate, @Price)
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end

```

Add_studie - PS

Procedura pozwala na dodanie studiów do bazy wraz z jej wszystkimi informacjami.

```

create procedure add_studie
    @CoordinatorID int,
    @Name nvarchar(30),
    @Description nvarchar(max),
    @StartDate date,
    @EndDate date,
    @Price money,
    @Status bit
as begin
    begin try
        begin transaction;

        -- Sprawdzenie poprawności wpisywanych danych
        if not exists(select 1 from Employees where EmployeeID = @CoordinatorID and
            PositionID = 3)

        begin
            throw 50001, 'Koordynator o danym ID nie istnieje lub nie jest koordynatorem studiów', 1;
        end
    end try

```

```

if @Price < 0
begin
    throw 50002, 'Cena nie może być mniejsza od 0', 1;
end

if @StartDate >= @EndDate
begin
    throw 50003, 'Nie poprawnie wpisane daty', 1;
end

-- W innym przypadku możemy dodać
-- Rezerwacja ID w produktach
declare @NewProductID int;
declare @CategoryID int = (select CategoryID from Categories where Name = 'Studies')

insert into Products (CategoryID, Status)
values (@CategoryID, @Status)

-- Pobranie ID po dodaniu do produktów
set @NewProductID = SCOPE_IDENTITY();

insert Studies (StudiesID, CoordinatorID, Name, Description, StartDate, EndDate)
values (@NewProductID, @CoordinatorID, @Name, @Description, @StartDate, @EndDate)

commit transaction;
end try
begin catch
    -- Wycofanie transakcji w przypadku błędu
    if @@TRANCOUNT > 0
    begin
        rollback transaction;
    end;

    -- Przerzucenie ERRORa dalej
    throw;
end catch
end

```

Delete_user_from_product - MS

Procedura usuwa użytkownika z danych studiów/kursu/webinaru. Dodatkowo w przypadku kursów usuwamy go ze wszystkich modułów, a w przypadku studiów ze wszystkich spotkań studyjnych.

```

create procedure delete_user_from_product
    @UserID int,
    @ProductID int
as begin
    begin try
        -- Sprawdzenie czy użytkownik istnieje
        if not exists(select 1 from Users where UserID = @UserID)
        begin
            throw 50001, 'Użytkownik o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dana para istnieje
        if dbo.check_user_enrollment_for_product (@UserID, @ProductID) = cast(0 as bit)
        begin
            throw 50002, 'Taka para nie istnieje', 2;
        end

        -- W innym przypadku ją usuwamy

        -- Produkt to STUDIA
        if exists(select 1 from Studies where StudiesID = @ProductID)
        begin
            delete from Users_Studies
            where UserID = @UserID and StudiesID = @ProductID

            -- Stworzenie tabeli ze wszystkimi spotkaniami ze wszystkich zjazdów z danych studiów
            declare @StudiesReunionsMeetingsIDs table (
                MeetingID int
            )

            insert @StudiesReunionsMeetingsIDs
            select m.MeetingID from Studies s
            join Studies_Reunion r on s.StudiesID = r.StudiesID
            join Meetings m on r.ReunionID = m.ReunionID
            where s.StudiesID = @ProductID

            delete uma
            from Users_Meetings_Attendance uma

```

```

        join @StudiesReunionsMeetingsIDs srm on uma.MeetingID = srm.MeetingID
        where uma.UserID = @UserID;
    end
    -- Produkt to KURS
    else if exists(select 1 from Courses where CourseID = @ProductID)
    begin
        delete from Users_Courses
        where UserID = @UserID and CourseID = @ProductID

        -- Pobrać tabele z modułami danego kursu
        declare @CourseModuleID table (
            ModuleID int
        )

        -- Zebranie ID modułów danego kursu
        insert @CourseModuleID (ModuleID)
        select ModuleID from Modules where Modules.CourseID = @ProductID

        delete ump
        from Users_Modules_Passes ump
        join @CourseModuleID cm on ump.ModuleID = cm.ModuleID
        where ump.UserID = @UserID;
    end
    -- Produkt to WEBINAR
    else
    begin
        delete from Users_Webinars
        where UserID = @UserID and WebinarID = @ProductID;
    end
end try
begin catch
    -- Przerzucenie błędu dalej
    throw;
end catch
end;

```

Set_extended_payment_deadline - MS

Procedura pozwala na ustawienie daty przedłużonego terminu podzamówienia.

```

create procedure set_extended_payment_deadline (
    @SubOrderID int,
    @ExtendedPaymentDeadline date
)
as begin
    begin try
        -- Sprawdzenie czy podzamówienie istnieje
        if not exists(select 1 from Orders_Details where SubOrderID = @SubOrderID)
        begin
            throw 50000, 'Podzamówienie o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy data przedłużonego terminu jest późniejsza od początkowej daty
        if ((select PaymentDeadline from Orders_Details where SubOrderID = @SubOrderID) > @ExtendedPaymentDeadline)
        begin
            throw 50001, 'Podana data jest nieprawidłowa', 1;
        end

        -- Aktualizacja daty przedłużonego terminu
        update Orders_Details
        set ExtendedPaymentDeadline = @ExtendedPaymentDeadline
        where SubOrderID = @SubOrderID
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;

```

Add_order - MS

Procedura pozwala na dodanie zamówienia.

```

CREATE procedure add_order
    @UserID int,
    @OrderDate date,
    @PaymentLink nvarchar(100),

```

```

    @LastID int output
as begin
    begin try
        -- Sprawdzenie czy dany użytkownik istnieje
        if not exists(select 1 from Users where UserID = @UserID)
        begin
            throw 50000, 'Użytkownik o podanym ID nie istnieje', 1;
        end

        -- Dodanie danych do tabeli
        insert Orders(UserID, OrderDate, PaymentLink)
        values (@UserID, @OrderDate, @PaymentLink)

        SET @LastID = SCOPE_IDENTITY();
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;

```

Add_suborder - MS

Procedura pozwala na dodanie podzamówienia.

```

CREATE procedure add_suborder
    @OrderID int,
    @PaymentDeadline date,
    @ExtendedPaymentDeadline date,
    @PaymentDate date,
    @FullPrice money,
    @ProductID int,
    @Payment money
as begin
    begin try
        -- Sprawdzenie, czy dane zamówienie istnieje
        if not exists(select 1 from Orders where OrderID = @OrderID)
        begin
            throw 50000, 'Zamówienie o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie, czy dany produkt istnieje
        if not exists(select 1 from Products where ProductID = @ProductID)
        begin
            throw 50000, 'Produkt o podanym ID nie istnieje', 1;
        end

        -- Dodanie danych do tabeli
        insert Orders_Details(OrderID, PaymentDeadline, ExtendedPaymentDeadline, PaymentDate, FullPrice, ProductID, Payment)
        values (@OrderID, @PaymentDeadline, @ExtendedPaymentDeadline,
            @PaymentDate, @FullPrice, @ProductID, @Payment)
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;

```

Update_recordinglink_meeting_sync - PS

Procedura pozwalająca na dodanie, czy zmianę liku do spotkania dla spotkania studyjnego.

```

CREATE procedure add_suborder
    @OrderID int,
    @PaymentDeadline date,
    @ExtendedPaymentDeadline date,
    @PaymentDate date,
    @FullPrice money,
    @ProductID int,
    @Payment money
as begin
    begin try
        -- Sprawdzenie, czy dane zamówienie istnieje
        if not exists(select 1 from Orders where OrderID = @OrderID)
        begin
            throw 50000, 'Zamówienie o podanym ID nie istnieje', 1;
        end
    end

```

```

-- Sprawdzenie, czy dany produkt istnieje
if not exists(select 1 from Products where ProductID = @ProductID)
begin
    throw 50000, 'Produkt o podanym ID nie istnieje', 1;
end

-- Dodanie danych do tabeli
insert Orders_Details(OrderID, PaymentDeadline, ExtendedPaymentDeadline, PaymentDate, FullPrice, ProductID, Payment)
values (@OrderID, @PaymentDeadline, @ExtendedPaymentDeadline,
        @PaymentDate, @FullPrice, @ProductID, @Payment)
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Update_meeting_type - PS

Procedura pozwalająca na zmianę typu spotkania studyjnego. W przypadku wpisania już danych do tabeli z konkretną kategorią spotkania, dane już wpisane są usuwane.

```

CREATE procedure update_meeting_type
    @MeetingID int,
    @TypeID int
as begin
    begin try
        -- Sprawdzenie czy moduł istnieje
        if not exists(select 1 from Meetings where @MeetingID = MeetingID)
        begin
            throw 50000, 'Spotkanie o podanym ID nie istnieje', 1;
        end

        -- Sprawdzenie czy dany typ istnieje
        if not exists(select 1 from Types where TypeID = @TypeID)
        begin
            throw 50001, 'Podany typ nie istnieje', 1;
        end

        -- Sprawdzenie czy nie zmieniasz typu spotkania na ten sam typ
        if exists(select 1 from Meetings where @MeetingID = MeetingID and @TypeID = TypeID)
        begin
            throw 50002, 'Zmieniasz typ spotkania na ten sam typ co był ustawiony', 1;
        end

        declare @OldType int = (select TypeID from Meetings where MeetingID = @MeetingID)
        declare @OldTypeName varchar(20);
        set @OldTypeName = (select TypeName from Types where TypeID = @OldType)

        -- Aktualizacja typu
        update Meetings
        set TypeID = @TypeID
        where MeetingID = @MeetingID

        -- Usunięcie danych ze starego typu
        if @OldTypeName = 'In-person'
        begin
            if exists(select 1 from In_person_Meetings where MeetingID = @MeetingID)
            begin
                -- Informacje zostały dodane i można usunąć informacje
                delete from In_person_Meetings
                where MeetingID = @MeetingID
            end
        end

        if @OldTypeName = 'Online Sync'
        begin
            if exists(select 1 from Online_Sync_Meetings where MeetingID = @MeetingID)
            begin
                -- Informacje zostały dodane i można usunąć informacje
                delete from Online_Sync_Meetings
                where MeetingID = @MeetingID
            end
        end

        if @OldTypeName = 'Online Async'
        begin
            if exists(select 1 from Online_Async_Meetings where MeetingID = @MeetingID)
            begin
                -- Informacje zostały dodane i można usunąć informacje
                delete from Online_Async_Meetings
            end
        end
    end try
end

```



```

        where MeetingID = @MeetingID
    end
end
end try
begin catch
    -- Przerzucenie ERRORa dalej
    throw;
end catch
end;

```

Add_subject - PS

Procedura pozwalająca na dodawanie przedmiotów do studiów.

```

create procedure add_subject
    @StudiesID int,
    @Name nvarchar(50),
    @Description nvarchar(max)
as begin
    begin try
        -- Sprawdzenie poprawności wpisanych danych
        if not exists(select 1 from Studies where StudiesID = @StudiesID)
        begin
            throw 50001, 'Studia o podanym ID nie istnieją', 1;
        end

        -- Dodanie danych
        insert Subjects (StudiesID, Name, Description)
        values (@StudiesID, @Name, @Description)
    end try
    begin catch
        -- Przerzucenie błędu dalej
        throw;
    end catch
end

```

Add_practice - PS

Procedura umożliwiająca dodanie praktykanta do listy z praktykami.

```

create procedure add_practice
    @Description nvarchar(max),
    @CompanyName nvarchar(30),
    @Country nvarchar(30),
    @City nvarchar(30),
    @Address nvarchar(50),
    @Phone nvarchar(20),
    @Email nvarchar(50)
as begin
    begin try
        -- Dodaawanie danych
        insert Practices (Description, CompanyName, Country, City, Address, Phone, Email)
        values (@Description, @CompanyName, @Country, @City, @Address, @Phone, @Email)
    end try
    begin catch
        -- Przerzucenie błędu dalej
        throw;
    end catch
end

```

Assign_user_to_practice - PS

Procedura pozwala na przypisanie studenta do danych praktyk odrazu z jakąś oceną (czy zaliczył czy nie).

```

create procedure assign_user_to_practice
    @UserID int,
    @StudiesID int,
    @PracticeID int,
    @Present bit
as begin
    begin try
        -- Sprawdzenie poprawności danych
        if not exists(select 1 from Users where UserID = @UserID)
        begin
            throw 50001, 'Użytkownik o podanym ID nie istnieje', 1;
        end
    end

```

```

if not exists(select 1 from Studies where StudiesID = @StudiesID)
begin
    throw 50002, 'Studia o podanym ID nie istnieją', 1;
end

if not exists(select 1 from Practices where PracticeID = @PracticeID)
begin
    throw 50003, 'Praktyki o podanym ID nie istnieją', 1;
end

if dbo.check_user_enrollment_for_product(@UserID, @StudiesID) = cast(0 as bit)
begin
    throw 50004, 'Użytkownik nie jest zapisany na dane studia', 1;
end

-- Dodanie danych
insert Users_Practices_Attendance (UserID, StudiesID, PracticeID, Present)
values (@UserID, @StudiesID, @PracticeID, @Present)
end try
begin catch
    -- Przerzucenie błędu dalej
    throw;
end catch
end

```

Update_product_status - PS

Procedura pozwalająca na aktualizację statusu produktu.

```

create procedure update_product_status
    @ProductID int,
    @Status bit
as begin
    begin try
        -- Sprawdzenie poprawności danych
        if not exists(select 1 from Products where ProductID = @ProductID)
        begin
            throw 50001, 'Produkt o danym ID nie istnieje', 1;
        end

        -- Aktualizacja danych
        update Products
        set Status = @Status
        where ProductID = @ProductID
    end try
    begin catch
        -- Przerzucenie błędu dalej
        throw;
    end catch
end

```

Set_student_grade - PS

Procedura pozwalająca na ustawienie oceny studentowi za studia.

```

create procedure set_student_grade
    @StudiesID int,
    @UserID int,
    @Grade int
as begin
    begin try
        -- Sprawdzenie poprawności danych
        if not exists(select 1 from Studies where StudiesID = @StudiesID)
        begin
            throw 50001, 'Studia o podanym ID nie istnieją', 1;
        end

        if not exists(select 1 from Users where UserID = @UserID)
        begin
            throw 50002, 'Użytkownik o podanym ID nie istnieje', 1;
        end

        if dbo.check_user_enrollment_for_product(@UserID, @StudiesID) = cast(0 as bit)
        begin
            throw 50003, 'Użytkownik nie jest zapisany na podane studia', 1;
        end
    end try

```

```

if @Grade < 2 or @Grade > 5
begin
    throw 50004, 'Podana ocena jest z nie dopuszczalnego przedziału [2,5]', 1;
end

-- Ustawienie oceny
update Users_Studies
set Grade = @Grade
where StudiesID = @StudiesID and UserID = @UserID
end try
begin catch
    -- Przerzucenie błędu dalej
    throw;
end catch
end

```

Set_student_practice_attendance - PS

Procedura pozwala na aktualizację obecności na danych praktykach studentowi.

```

create procedure set_student_practice_attendance
    @UserID int,
    @StudiesID int,
    @PracticeID int,
    @Present bit
as begin
    begin try
        -- Sprawdzenie poprawności danych
        if not exists(select 1 from Users where UserID = @UserID)
        begin
            throw 50001, 'Użytkownik o podanym ID nie istnieje', 1;
        end

        if not exists(select 1 from Studies where StudiesID = @StudiesID)
        begin
            throw 50002, 'Studia o podanym ID nie istnieją', 1;
        end

        if not exists(select 1 from Practices where PracticeID = @PracticeID)
        begin
            throw 50003, 'Praktyki o podanym ID nie istnieją', 1;
        end

        if not exists(select 1 from Users_Practices_Attendance where StudiesID = @StudiesID and PracticeID = @PracticeID
            and UserID = @UserID)
        begin
            throw 50004, 'Użytkownik o podanych studiach i praktykach nie ma wpisu w tabeli', 1;
        end

        -- Ustawienie obecności
        update Users_Practices_Attendance
        set Present = @Present
        where UserID = @UserID and PracticeID = @PracticeID and @StudiesID = StudiesID
    end try
    begin catch
        -- Przerzucenie błędu dalej
        throw;
    end catch
end

```

Set_student_meeting_attendance - PS

Procedura pozwala na ustawianie obecności studentom na spotkaniach.

```

create procedure set_student_meeting_attendance
    @UserID int,
    @MeetingID int,
    @Present bit
as begin
    begin try
        -- Sprawdzenie poprawności danych
        if not exists(select 1 from Users where UserID = @UserID)
        begin
            throw 50001, 'Użytkownik o podanym ID nie istnieje', 1;
        end
    end try

```

```

if not exists(select 1 from Meetings where MeetingID = @MeetingID)
begin
    throw 50002, 'Spotkanie o podanym ID nie istnieje', 1;
end

if not exists(select 1 from Users_Meetings_Attendance where MeetingID = @MeetingID and @UserID = UserID)
begin
    throw 50003, 'Para student-spotkanie nie istnieje', 1;
end

-- Ustawienie obecności
update Users_Meetings_Attendance
set Present = @Present
where UserID = @UserID and MeetingID = @MeetingID
end try
begin catch
    -- Przerzucenie błędu dalej
    throw;
end catch
end

```

Set_user_module_attendance - PS

Procedura pozwala na ustawienie czy dany użytkownik zdał dany moduł czy nie.

```

create procedure set_user_module_passes
@UserID int,
@ModuleID int,
@Passed bit
as begin
begin try
    -- Sprawdzenie poprawności danych
    if not exists(select 1 from Users where UserID = @UserID)
    begin
        throw 50001, 'Użytkownik o podanym ID nie istnieje', 1;
    end

    if not exists(select 1 from Modules where ModuleID = @ModuleID)
    begin
        throw 50002, 'Moduł o podanym ID nie istnieje', 1;
    end

    if not exists(select 1 from Users_Modules_Passes where ModuleID = @ModuleID and @UserID = UserID)
    begin
        throw 50003, 'Para użytkownik-moduł nie istnieje', 1;
    end

    -- Ustawienie obecności
    update Users_Modules_Passes
    set Passed = @Passed
    where UserID = @UserID and ModuleID = @ModuleID
end try
begin catch
    -- Przerzucenie błędu dalej
    throw;
end catch
end

```