Оглавление

[ЗАДАНИЕ 4](#_Toc74781833)

[Введение 4](#_Toc74781834)

[1. Реализация таблиц и связей 5](#_Toc74781835)

[2. Реализация запросов 6](#_Toc74781837)

[3. Реализация индексов. 7](#_Toc74781838)

[4. Реализация индексов 8](#_Toc74781839)

[5. Реализация процедур 9](#_Toc74781840)

[6. Реализация составной транзакции 14](#_Toc74781841)

[7. Реализация курсора на обновления 15](#_Toc74781842)

[8. Реализация функций 15](#_Toc74781843)

[9. Реализация распределения прав 16](#_Toc74781844)

[Интерфейс приложения 16](#_Toc74781845)

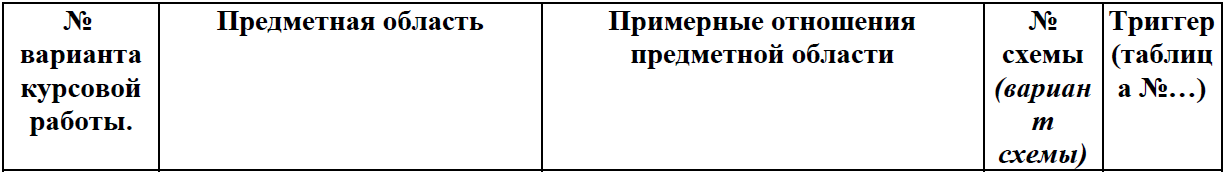
[Листинг кода 20](#_Toc74781846)

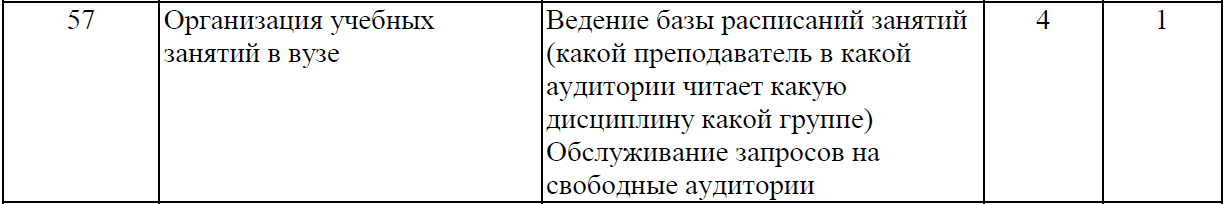
[Вывод 47](#_Toc74781847)

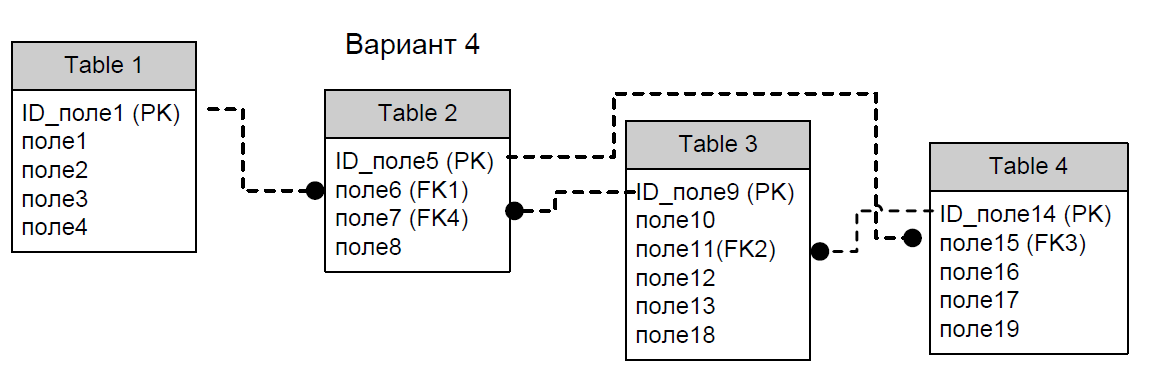
[Источники 47](#_Toc74781848)

# 

# ЗАДАНИЕ







Разработать клиент-серверное приложение, серверная часть которой реализована на любом SQL сервере, представляющая собой модель предметной области в соответствии с вариантом задания. В рамках заданной предметной области реализовать заданную (по варианту) схему отношений, т.е. выделить сущности и их атрибуты, так чтобы связи между сущностями соответствовали представленной схеме.

# Введение

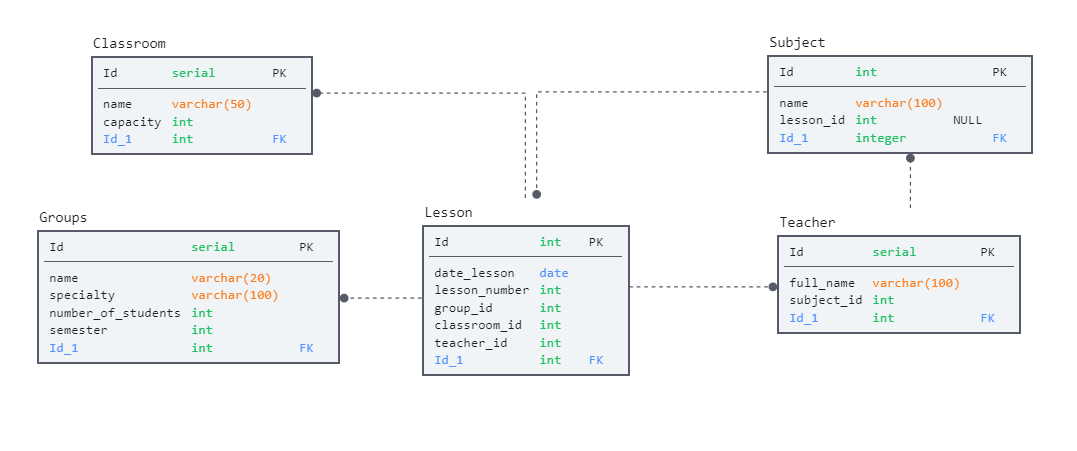
Базы данных — это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять.

Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию, начиная от даты рождения и заканчивая подробным описанием кулинарного рецепта. Однотипные поля разных записей образуют столбец таблицы.

Создав одну таблицу, вы уже получаете полноценную базу данных. Однако в реальной жизни структуры баз данных, а соответственно и способы их создания, намного сложнее.

# РЕАЛИЗАЦИЯ ТАБЛИЦ И СВЯЗЕЙ



# 1. Постоянные таблицы и связи между ними, количество таблиц и наличие связей должно соответствовать заданию, допускается увеличение числа таблиц и их полей для более адекватного представления предметной области.

create table Groups

(

    Id serial primary key,

    name varchar(20) not null unique,

    specialty varchar(100) not null,

    number\_of\_students int not null,

    semester int not null

);

create table Classroom

(

    Id serial primary key,

    name varchar(50) not null unique,

    capacity int not null

);

create table Subject

(

    Id serial primary key,

    name varchar(100) not null unique

);

create table Teacher

(

    Id serial primary key,

    full\_name varchar(100) not null,

    subject\_id integer references Subject(Id)

);

create table Lesson

(

    Id serial primary key,

    date\_lesson date not null,

    lesson\_number int not null,

    group\_id integer references Groups(Id) not null,

    classroom\_id integer references Classroom(Id) not null,

    teacher\_id integer references Teacher(Id) not null

);

alter table Subject

add lesson\_id integer references Lesson(Id) default null

# 2. В приложении реализовать не менее пяти запросов, включая (для демонстрации навыков работы)

**a. Составной многотабличный запрос с параметром, включающий соединение таблиц и CASE-выражение;**

select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Groups.name as group\_name, Classroom.name as classsroom\_name,

       (case when Groups.number\_of\_students > Classroom.capacity then 'students will not fit in the classroom'

        else 'students will fit in the classroom' end) as capacity\_check

from Lesson

     join Groups on Lesson.group\_id = Groups.Id

     join Classroom on Lesson.classroom\_id = Classroom.Id

**b. На основе обновляющего представления (многотабличного VIEW), в котором критерий упорядоченности задает пользователь при выполнении;**

create or replace view lesson\_veiw as

    select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Groups.name as group\_name,

           Groups.specialty, Classroom.name as classroom\_name, Teacher.full\_name, Subject.name as subject\_name

from Lesson

      join Groups on Lesson.group\_id = Groups.Id

      join Classroom on Lesson.classroom\_id = Classroom.Id

      join Teacher on Lesson.teacher\_id = Teacher.Id

      join Subject on Teacher.subject\_id = Subject.Id

order by Lesson.date\_lesson

**c. Запрос, содержащий коррелированные и некоррелированные подзапросы в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);**

select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number,

        (select Groups.name from Groups

            where Groups.Id = group\_id) as group\_name,

        (select Classroom.name from Classroom

            where Classroom.Id = classroom\_id) as classroom\_name,

        (select Teacher.full\_name from Teacher

            where Teacher.Id = teacher\_id) as teacher\_name

from (select \* from Lesson

        where lesson\_number < 4) as Lesson

where (select Classroom.capacity from Classroom

            where Classroom.Id = classroom\_id) >

       (select Groups.number\_of\_students from Groups

            where Groups.Id = group\_id)

**d. Многотабличный запрос, содержащий группировку записей, агрегативные функции и параметр, используемый в разделе HAVING;**

select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Groups.name as group\_name,

       min(Groups.number\_of\_students) as number\_of\_students

from Lesson, Groups

    where Groups.Id = Lesson.group\_id

group by Lesson.Id, Groups.name

    having (min(Groups.number\_of\_students) < 26);

**e. Запрос, содержащий предикат ANY(SOME) или ALL;**

select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Classroom.name as classroom\_name

from Lesson

     join Classroom on Lesson.classroom\_id = Classroom.Id

     where Lesson.classroom\_id = any

     (select Classroom.Id from Classroom where Classroom.name = '321b')

# 3. Создать индексы для увеличения скорости выполнения запросов.

create index i\_group\_name on Groups using btree(name);

create index i\_classroom\_name on Classroom using btree(name);

create index i\_Teacher\_full\_name on Teacher using btree(full\_name);

create index i\_subject\_name on Subject using btree(name);

# 4. В таблице (в соответствии с вариантом) предусмотреть поле, которое заполняется автоматически по срабатыванию триггера при добавлении, обновлении и удалении данных, иметь возможность продемонстрировать работу триггера при работе приложения. Триггеры должны обрабатывать только те записи, которые были добавлены, изменены или удалены в ходе текущей операции (транзакции).

create table History

(

    Id serial primary key,

    group\_id int not null,

    operation varchar(100) not null,

    group\_name varchar(100) not null,

    create\_at date not null default current\_date

);

/\* добавление новой группы \*/

create or replace function Groups\_add\_history()

returns trigger

language plpgsql

as $function$

begin

    insert into History(group\_id, operation, group\_name)

    values (new.Id, 'add group ',new.name);

    return new;

end;

$function$;

create trigger tr\_group\_add

after insert on Groups

for each row execute procedure Groups\_add\_history()

/\* удаление группы \*/

create or replace function Groups\_deletion\_history()

returns trigger

language plpgsql

as $function$

begin

    insert into History(group\_id, operation, group\_name)

    values (old.Id, 'deleting a group ' , old.name);

    return old;

end;

$function$;

create trigger tr\_group\_deletion

after delete on Groups

for each row execute procedure Groups\_deletion\_history()

/\* изменение группы \*/

create or replace function Groups\_update\_history()

returns trigger

language plpgsql

as $function$

begin

    insert into History(group\_id, operation, group\_name)

    values (new.Id, 'group update ' , new.name);

    return new;

end;

$function$;

create trigger tr\_group\_update

after update on Groups

for each row execute procedure Groups\_update\_history()

# 5. Операции добавления, удаления и обновления реализовать в виде хранимых процедур (с параметрами) хотя бы для одной таблицы; для остальных допустимо использовать возможности связывания полей ввода в приложении с полями БД.

/\* добавление группы \*/

create or replace function add\_group(\_name text, \_specialty text, \_number\_of\_students int, \_semester int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    insert into Groups(name, specialty, number\_of\_students, semester)

                values(\_name, \_specialty, \_number\_of\_students, \_semester);

    return checker;

end;

$$;

/\* изменение группы \*/

create or replace function update\_group(\_Id int, \_name text, \_specialty text, \_number\_of\_students int, \_semester int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    update Groups

        set name = \_name,

            specialty = \_specialty,

            number\_of\_students = \_number\_of\_students,

            semester = \_semester

        where Id = \_Id;

    return checker;

end;

$$;

/\* удаление группы \*/

create or replace function delete\_group(\_Id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    delete from Groups

        where Id = \_Id;

    return checker;

end;

$$;

/\* добавление аудитории \*/

create or replace function add\_classroom(\_name text, \_capacity int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    insert into Classroom(name, capacity)

                values(\_name, \_capacity);

    return checker;

end;

$$;

/\* изменение аудитории \*/

create or replace function update\_classroom(\_Id int, \_name text, \_capacity int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    update Classroom

        set name = \_name,

            capacity = \_capacity

        where Id = \_Id;

    return checker;

end;

$$;

/\* удаление аудитории \*/

create or replace function delete\_classroom(\_Id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    delete from Classroom

        where Id = \_Id;

    return checker;

end;

$$;

/\* добавление дисциплины \*/

create or replace function add\_subject(\_name text, \_lesson\_id int default null)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    insert into Subject(name, lesson\_id)

                values(\_name, \_lesson\_id);

    return checker;

end;

$$;

/\* изменение дисциплины \*/

create or replace function update\_subject(\_Id int, \_name text, \_lesson\_id int default null)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    update Subject

        set name = \_name,

            lesson\_id = \_lesson\_id

        where Id = \_Id;

    return checker;

end;

$$;

select update\_subject(2,'Линейная алгебра')

/\* удаление дисциплины \*/

create or replace function delete\_subject(\_Id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    delete from Subject

        where Subject.Id = \_Id;

    return checker;

end;

$$;

/\* добавление преподователя \*/

create or replace function add\_teacher(\_full\_name text, \_subject\_id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    insert into Teacher(full\_name, subject\_id)

                values(\_full\_name, \_subject\_id);

    return checker;

end;

$$;

/\* изменение преподователя \*/

create or replace function update\_teacher(\_Id int, \_full\_name text, \_subject\_id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    update Teacher

        set full\_name = \_full\_name,

            subject\_id = \_subject\_id

        where Id = \_Id;

    return checker;

end;

$$;

/\* удаление преподователя \*/

create or replace function delete\_teacher(\_Id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    delete from Teacher

        where Id = \_Id;

    return checker;

end;

$$;

/\* добавление пары \*/

create or replace function add\_Lesson(\_date\_lesson date, \_lesson\_number int, \_group\_id int, \_classroom\_id int, \_teacher\_id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    if (\_lesson\_number > 10 or \_lesson\_number < 1) then

    raise exception 'неверно указан номер пары';

    checker := false;

    end if;

    if(checker = true) then

    insert into Lesson(date\_lesson, lesson\_number, group\_id, classroom\_id ,teacher\_id)

                values(\_date\_lesson, \_lesson\_number, \_group\_id, \_classroom\_id, \_teacher\_id);

    end if;

    return checker;

end;

$$;

/\* изменение пары \*/

create or replace function update\_Lesson(\_Id int, \_date\_lesson date, \_lesson\_number int, \_group\_id int, \_classroom\_id int, \_teacher\_id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    if (\_lesson\_number > 10 or \_lesson\_number < 1) then

    raise exception 'неверно указан номер пары';

    checker := false;

    end if;

    if(checker = true) then

    update Lesson

        set date\_lesson = \_date\_lesson,

            lesson\_number = \_lesson\_number,

            group\_id = \_group\_id,

            classroom\_id = \_classroom\_id,

            teacher\_id = \_teacher\_id

        where Id = \_Id;

    end if;

    return checker;

end;

$$;

/\* удаление пары \*/

create or replace function delete\_Lesson(\_Id int)

returns boolean

language plpgsql

as $$

declare

checker boolean:= true;

begin

    delete from Lesson

        where Lesson.Id = \_Id;

    return checker;

end;

$$;

# 6. Реализовать отдельную хранимую процедуру, состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана.

create or replace procedure next\_year\_groups(group\_id int)

language plpgsql

as $$

declare

    current\_year int;

begin

    select Groups.year into current\_year from Groups where Groups.Id = group\_id;

    if current\_year < 5  then

            update Groups

            set Groups.year = Groups.yaer + 1 where Groups.Id = group\_id;

    else

            delete from Groups

                where Groups.Id = group\_id;

    end if;

end; $$

# 7. В триггере или хранимой процедуре реализовать курсор на обновления отдельных данных.

create or replace function group\_update(

    \_id int,

    \_name text,

    \_specialty text,

    \_number\_of\_students int,

    semester int

)

returns boolean

language plpgsql

as $$

declare

checker boolean := true;

\_cursor cursor (cursor\_id bigint) for select \* from Groups where Groups.Id = cursor\_id;

begin

    open \_cursor(cursor\_id := \_id);

    move first from \_cursor;

    update Groups

        set

            Groups.name = \_name,

            Groups.specialty = \_specialty,

            Groups.number\_of\_students = \_number\_of\_students,

            Groups.semester = \_semestr

    where current of \_cursor;

    close \_cursor;

return checker;

end;

$$;

# 8. В запросе (из пункта 2 или в дополнительном к тому перечню) использовать собственную скалярную функцию, а в хранимой процедуре – векторную (или табличную) функцию. Функции сохранить в базе данных.

/\* Скалярная функция\*/

create or replace function get\_number\_of\_students(group\_name text)

returns int

language plpgsql

as $$

declare

\_number\_of\_students int;

begin

    select Groups.number\_of\_students into \_number\_of\_students from Groups

        where Groups.name = group\_name;

    return \_number\_of\_students;

end;

$$;

/\* Табличная функция \*/

create or replace function return\_table\_group(group\_name text)

returns table (\_id int, \_name text, \_specialty text)

language plpgsql

as $$

begin

    select Groups.Id, Groups.name, Groups.specialty from Groups

        where Groups.name = group\_name;

end;

$$;

# 9. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

create role Timetable\_Teacher;

create role Timetable\_Student;

create role Timetable\_Admin;

grant select, insert on all tables in schema public to Timetable\_Teacher;

grant select on all tables in schema public to Timetable\_Student;

grant select, update, insert, delete on all tables in schema public to Timetable\_Admin;

grant usage, select on all sequences in schema public to Timetable\_Admin;

create user T\_Teacher password '12345';

create user T\_Student password '12345';

create user T\_Admin password 'admin';

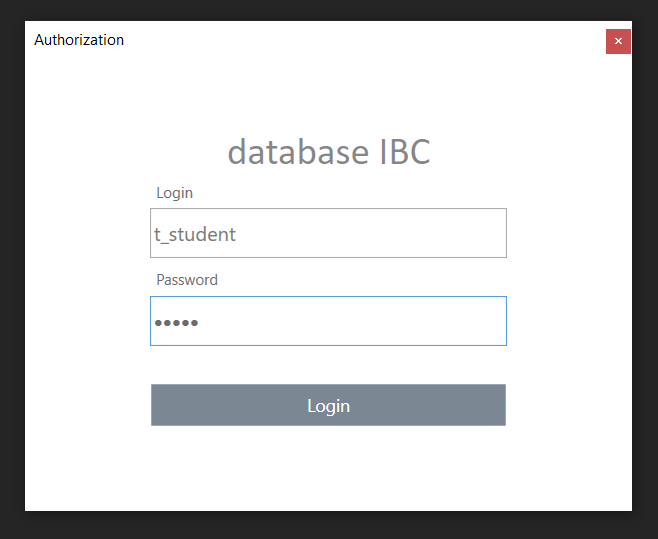
grant Timetable\_Teacher to T\_Teacher;

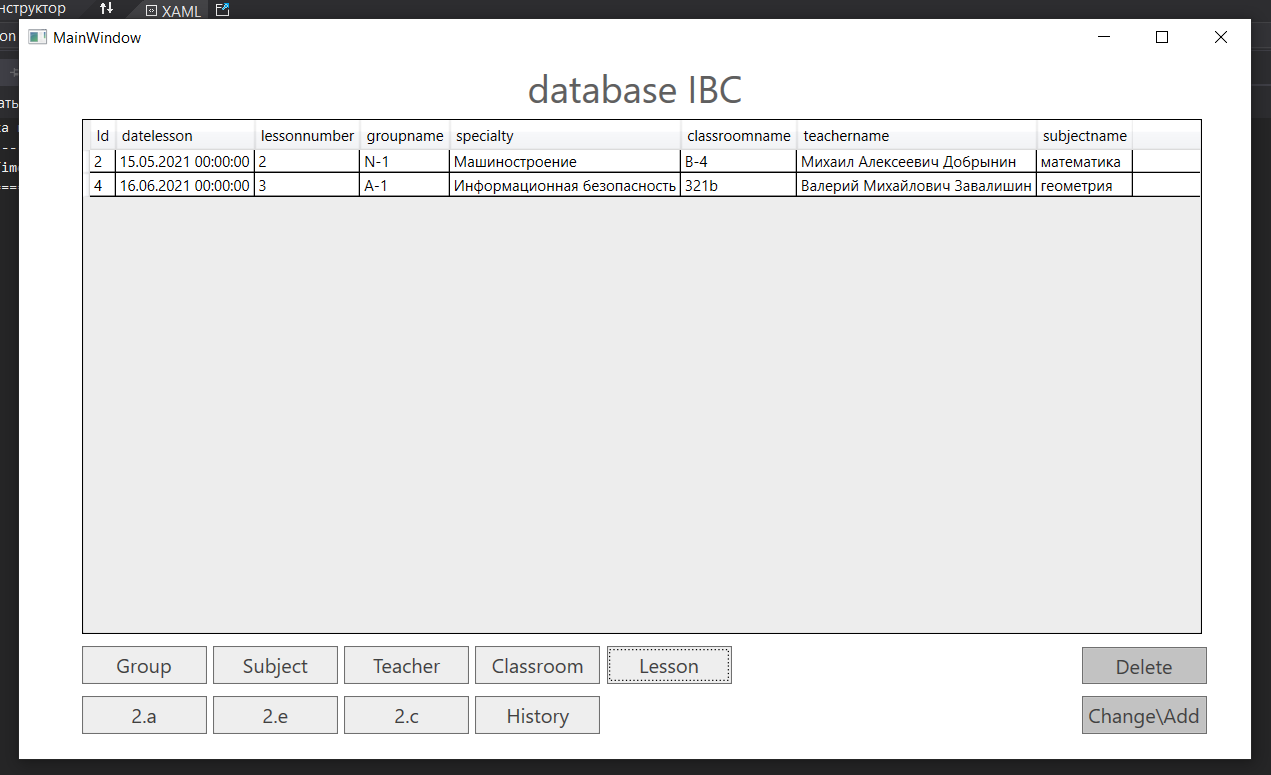
grant Timetable\_Student to T\_Student;

grant Timetable\_Admin to T\_Admin;

# Интерфейс приложения

Клиентское приложение реализовано на языке C# с использованием фреймворка WPF. Для взаимодействия приложения с бд использовалась технология ADO.NET

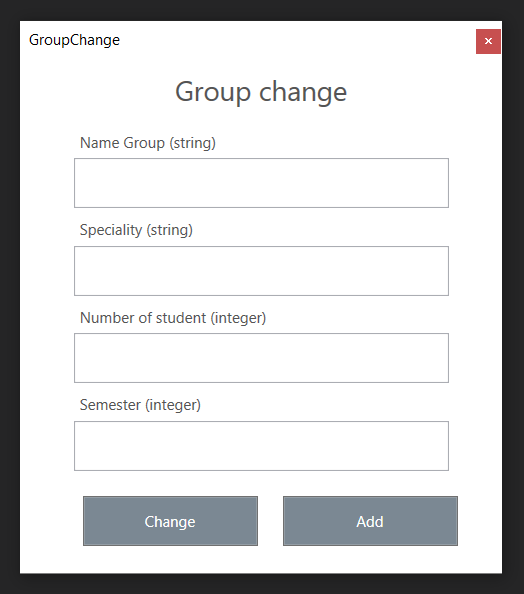


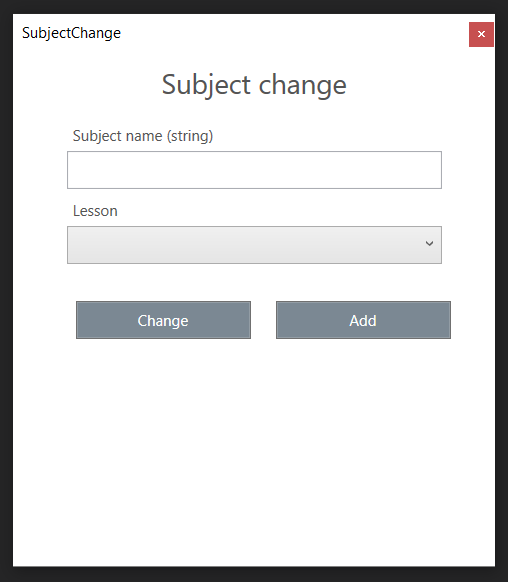


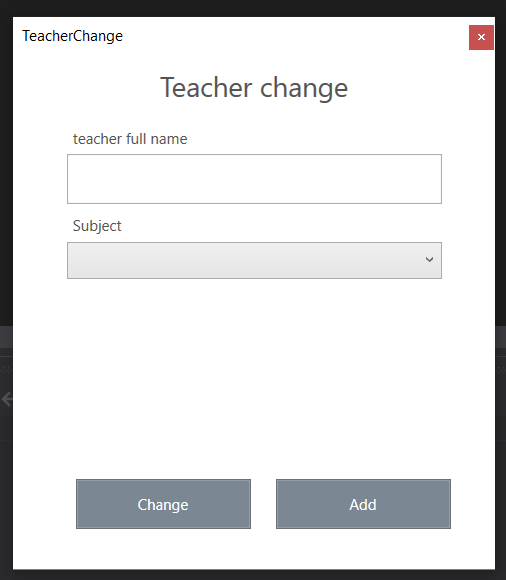
При нажатии на кнопки с названием таблиц, данные из бд отобразятся в DataGrid.

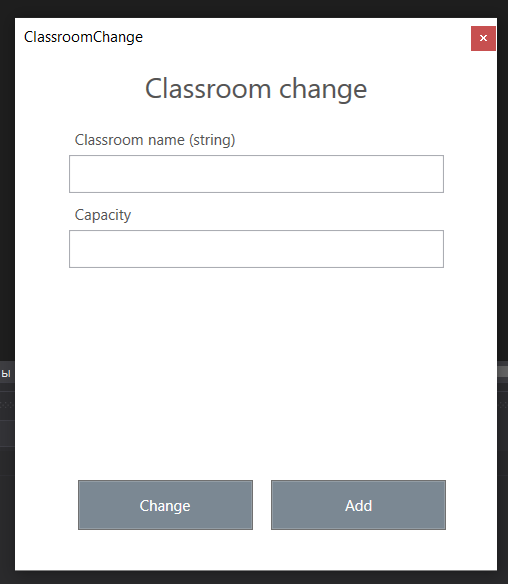
Для изменения записи нужно нажать на номер записи и после нажать на кнопку Change/Add.

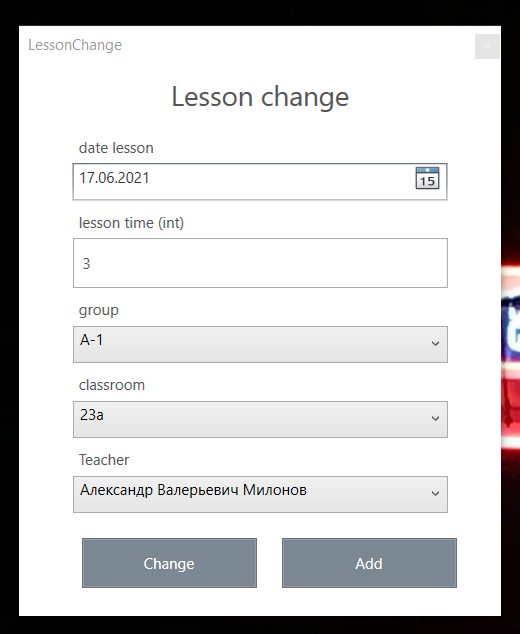
После этого откроется специальное окно











После заполнения полей можно изменить выбранную запись или добавить новую

# Листинг кода

Authorization.xaml

<Window x:Class="Timetable\_of\_classes.Authorization"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes"

mc:Ignorable="d"

Title="Authorization" Height="400" Width="500" MinHeight="400" MinWidth="500" MaxHeight="400" MaxWidth="500" WindowStyle="ToolWindow">

<Grid>

<StackPanel Margin="100,50,100,50">

<Label Height="40" Margin="0" Content="database IBC" VerticalContentAlignment="Top" HorizontalContentAlignment="Center" FontSize="31" FontFamily="Calibri" Opacity="0.5" Foreground="#FF0C0C0C" Background="White" />

<Label Height="30" Content="Login" VerticalContentAlignment="Bottom" HorizontalContentAlignment="Left" Foreground="#FF6C6C6C"></Label>

<TextBox x:Name="textBoxLogin" VerticalContentAlignment="Center" Margin="0,0,0,0" Height="40" FontSize="16" BorderThickness="1" Foreground="#FF797979" />

<Label Height="30" Content="Password" VerticalContentAlignment="Bottom" HorizontalContentAlignment="Left" Foreground="#FF6C6C6C"></Label>

<PasswordBox x:Name="textBoxPassword" Margin="0,0,0,15" Height="40" VerticalContentAlignment="Center" Foreground="#FF6C6C6C" SelectionTextBrush="#FF6C6C6C" SelectionBrush="#FF6C6C6C" />

<Button x:Name="authorizationButoom" Content="Login" Margin="0,15,0,15" Height="35" Background="#FF7B8893" BorderBrush="{x:Null}" ScrollViewer.VerticalScrollBarVisibility="Disabled" FontSize="14" Foreground="White" Click="authorizationButoom\_Click"/>

</StackPanel>

</Grid>

</Window>

Authorization.xaml.cs

using System;

using System.Text;

using System.Windows;

namespace Timetable\_of\_classes

{

/// <summary>

/// Логика взаимодействия для Authorization.xaml

/// </summary>

public partial class Authorization : Window

{

MainWindow window;

public Authorization(MainWindow mainWindow)

{

InitializeComponent();

window = mainWindow;

}

private void authorizationButoom\_Click(object sender, RoutedEventArgs e)

{

try

{

Connection.connectionString = $"Server=localhost;Port=5432;User Id={this.textBoxLogin.Text};Password={this.textBoxPassword.Password};Database=Timetable;";

byte[] bytes = Encoding.Default.GetBytes(Connection.connectionString);

Connection.connectionString = Encoding.UTF8.GetString(bytes);

Connection.sqlConnection = new Npgsql.NpgsqlConnection(Connection.connectionString);

Connection.sqlConnection.Open();

this.Close();

window.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Classroom.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Timetable\_of\_classes

{

public class Classroom

{

public Classroom(int id, string name, int capacity)

{

this.id = id;

this.name = name;

this.capacity = capacity;

}

public int id { get; set; }

public string name { get; set; }

public int capacity { get; set; }

}

}

ClassroomChange.xaml

<Window x:Class="Timetable\_of\_classes.ClassroomChange"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes"

mc:Ignorable="d"

Title="ClassroomChange" Height="450" Width="400" MinHeight="450" MinWidth="400" MaxHeight="450" MaxWidth="400" WindowStyle="ToolWindow">

<Grid>

<StackPanel>

<Label Content="Classroom change" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="50" Foreground="#FF545454" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="22"/>

<Label Content="Classroom name (string)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="classroomNameTextBox" HorizontalAlignment="Center" Height="30" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" SelectionBrush="#FF5D5D5D" Foreground="#FF545454" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="Capacity" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="classroomCapacityTextBox" Margin="0,0,0,170" HorizontalAlignment="Center" Height="30" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" SelectionBrush="#FF5D5D5D" Foreground="#FF545454" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<StackPanel Orientation="Horizontal">

<Button Content="Change" Margin="50,0,0,0" HorizontalAlignment="Left" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click"/>

<Button Content="Add" Margin="15,0,0,0" HorizontalAlignment="Center" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click\_1"/>

</StackPanel>

</StackPanel>

</Grid>

</Window>

ClassroomChange.xaml.cs

using Npgsql;

using System;

using System.Windows;

namespace Timetable\_of\_classes

{

/// <summary>

/// Логика взаимодействия для ClassroomChange.xaml

/// </summary>

public partial class ClassroomChange : Window

{

public ClassroomChange()

{

InitializeComponent();

}

private async void Button\_Click(object sender, RoutedEventArgs e)

{

try

{

string sqlExpression = $"select update\_classroom({Connection.currentId}, '{this.classroomNameTextBox.Text}', {this.classroomCapacityTextBox.Text})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromClassroom();

this.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private async void Button\_Click\_1(object sender, RoutedEventArgs e)

{

try

{

string sqlExpression = $"select add\_classroom('{this.classroomNameTextBox.Text}', {this.classroomCapacityTextBox.Text})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromClassroom();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Connection.cs

using Npgsql;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

namespace Timetable\_of\_classes

{

public enum ObjectName { Classroom, Group, Lesson, Teacher, Subject }

public static class Connection

{

public static ObjectName objectName;

public static DataGrid GroupGrid;

public static string connectionString { get; set; }

public static NpgsqlConnection sqlConnection { get; set; }

public static int currentId { get; set; }

public static List<Group> SelectFromGroup()

{

List<Group> groups = new List<Group>();

try

{

string sqlExpression = "SELECT \* FROM Groups";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string name = (string)reader.GetValue(1);

string specialty = (string)reader.GetValue(2);

int number\_of\_students = (int)reader.GetValue(3);

int semester = (int)reader.GetValue(4);

groups.Add(new Group(id, name, specialty, number\_of\_students, semester));

}

reader.Close();

objectName = ObjectName.Group;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

return groups;

}

public static List<Lesson> SelectfromLesson()

{

List<Lesson> lesson = new List<Lesson>();

try

{

string sqlExpression = "SELECT \* FROM lesson\_veiw";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string date\_lesson = reader.GetValue(1).ToString();

int lesson\_number = (int)reader.GetValue(2);

string group\_name = reader.GetValue(3).ToString();

string specialty = reader.GetValue(4).ToString();

string classroom\_name = reader.GetValue(5).ToString();

string teacher\_name = reader.GetValue(6).ToString();

string subject\_name = reader.GetValue(7).ToString();

lesson.Add(new Lesson(id, date\_lesson, lesson\_number, group\_name, specialty, classroom\_name, teacher\_name, subject\_name));

}

reader.Close();

objectName = ObjectName.Lesson;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

return lesson;

}

public static List<Subject> SelectfromSubject()

{

List<Subject> subjects = new List<Subject>();

try

{

string sqlExpression = "SELECT \* FROM Subject";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string name = reader.GetValue(1).ToString();

string subject\_id = reader.GetValue(2).ToString();

subjects.Add(new Subject(id, name, subject\_id));

}

reader.Close();

objectName = ObjectName.Subject;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

return subjects;

}

public static List<Teacher> SelectFromTeacher()

{

List<Teacher> teachers = new List<Teacher>();

try

{

string sqlExpression = "select Teacher.Id, Teacher.full\_name, Subject.name " +

"from Teacher " +

"left join Subject on Teacher.subject\_id = Subject.Id";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string full\_name = reader.GetValue(1).ToString();

string subject\_name = reader.GetValue(2).ToString();

teachers.Add(new Teacher(id, full\_name, subject\_name));

}

reader.Close();

objectName = ObjectName.Teacher;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

return teachers;

}

public static List<Classroom> SelectFromClassroom()

{

List<Classroom> classrooms = new List<Classroom>();

try

{

string sqlExpression = "select \* from Classroom";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string name = reader.GetValue(1).ToString();

int capacity = (int)reader.GetValue(2);

classrooms.Add(new Classroom(id, name, capacity));

}

reader.Close();

objectName = ObjectName.Classroom;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

return classrooms;

}

public static T ConvertFromDBVal<T>(object obj)

{

if (obj == null || obj == DBNull.Value)

{

return default(T);

}

else

{

return (T)obj;

}

}

public static List<A2Class> A2()

{

List<A2Class> ts = new List<A2Class>();

try

{

string sqlExpression = $"select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Groups.name as group\_name, Classroom.name as classsroom\_name, " +

"(case when Groups.number\_of\_students > Classroom.capacity then 'students will not fit in the classroom' " +

"else 'students will fit in the classroom' end) as capacity\_check " +

"from Lesson " +

"join Groups on Lesson.group\_id = Groups.Id " +

"join Classroom on Lesson.classroom\_id = Classroom.Id";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int Id = (int)reader.GetValue(0);

string date\_lesson = reader.GetValue(1).ToString();

int lesson\_number = (int)reader.GetValue(2);

string group\_name = reader.GetValue(3).ToString();

string classsroom\_name = reader.GetValue(4).ToString();

string capacity\_check = reader.GetValue(5).ToString();

ts.Add(new A2Class(Id, date\_lesson, lesson\_number, group\_name, classsroom\_name, capacity\_check));

}

reader.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

return ts;

}

public static List<D2Class> D2()

{

List<D2Class> ts = new List<D2Class>();

try

{

string sqlExpression = "select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Groups.name as group\_name, " +

"min(Groups.number\_of\_students) as number\_of\_students " +

"from Lesson, Groups " +

"where Groups.Id = Lesson.group\_id " +

"group by Lesson.Id, Groups.name " +

"having(min(Groups.number\_of\_students) < 25); ";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int Id = (int)reader.GetValue(0);

string date\_lesson = reader.GetValue(1).ToString();

int lesson\_number = (int)reader.GetValue(3);

string group\_name = reader.GetValue(4).ToString();

int number\_of\_students = (int)reader.GetValue(5);

ts.Add(new D2Class(Id, date\_lesson, lesson\_number, group\_name, number\_of\_students));

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ts;

}

public static List<E2Class> E2()

{

List<E2Class> ts = new List<E2Class>();

try

{

string sqlExpression = "select Lesson.Id, Lesson.date\_lesson, Lesson.lesson\_number, Classroom.name as classroom\_name " +

"from Lesson " +

"join Classroom on Lesson.classroom\_id = Classroom.Id " +

"where Lesson.classroom\_id = any " +

"(select Classroom.Id from Classroom where Classroom.name = '34a')";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int Id = (int)reader.GetValue(0);

string date\_lesson = reader.GetValue(1).ToString();

int lesson\_number = (int)reader.GetValue(2);

string classroom\_name = reader.GetValue(3).ToString();

ts.Add(new E2Class(Id, date\_lesson, lesson\_number, classroom\_name));

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ts;

}

public static List<History> histories()

{

List<History> history = new List<History>();

try

{

string sqlExpression = "SELECT \* FROM History";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int Id = (int)reader.GetValue(0);

int groupId = (int)reader.GetValue(1);

string operation = (string)reader.GetValue(2);

string createAt = (string)reader.GetValue(3).ToString();

history.Add(new History(Id, groupId, operation, createAt));

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return history;

}

}

}

Group.cs

namespace Timetable\_of\_classes

{

public class Group

{

public Group(int Id, string name, string specialty, int number\_of\_students, int semester)

{

this.Id = Id;

this.name = name;

this.specialty = specialty;

this.number\_of\_students = number\_of\_students;

this.semester = semester;

}

public int Id { get; set; }

public string name { get; set; }

public string specialty { get; set; }

public int number\_of\_students { get; set; }

public int semester { get; set; }

}

}

GroupChange.xaml

<Window x:Class="Timetable\_of\_classes.GroupChange"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes"

mc:Ignorable="d"

Title="GroupChange" Height="450" Width="400" MinHeight="450" MinWidth="400" MaxHeight="450" MaxWidth="400" WindowStyle="ToolWindow">

<Grid>

<StackPanel>

<Label Content="Group change" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="50" Foreground="#FF545454" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="22"/>

<Label Content="Name Group (string)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="nameGroupTextBox" HorizontalAlignment="Center" Height="40" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" SelectionBrush="#FF5D5D5D" Foreground="#FF545454" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="Speciality (string)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="specialityTextBox" HorizontalAlignment="Center" Height="40" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" Foreground="#FF545454" SelectionBrush="#FF5D5D5D" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="Number of student (integer)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="numberOfStudentTextBox" HorizontalAlignment="Center" Height="40" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" Foreground="#FF545454" SelectionBrush="#FF5D5D5D" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="Semester (integer)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="semesterTextBox" HorizontalAlignment="Center" Height="40" Margin="0,0,0,20" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" Foreground="#FF545454" SelectionBrush="#FF5D5D5D" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<StackPanel Orientation="Horizontal">

<Button Content="Change" HorizontalAlignment="Left" Margin="50,0,0,0" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click"/>

<Button Content="Add" HorizontalAlignment="Left" Margin="20,0,0,0" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click\_1"/>

</StackPanel>

</StackPanel>

</Grid>

</Window>

GroupChange.xaml.cs

using Npgsql;

using System;

using System.Data.SqlClient;

using System.Windows;

namespace Timetable\_of\_classes

{

/// <summary>

/// Логика взаимодействия для GroupChange.xaml

/// </summary>

public partial class GroupChange : Window

{

string nameGroup;

string speciality;

int numberOfStudent;

int semester;

private string sqlExpression;

public GroupChange()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

UpdateOrAdd("update\_group");

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

UpdateOrAdd("add\_group");

}

private async void UpdateOrAdd(string EXEC)

{

try

{

if (!string.IsNullOrEmpty(this.nameGroupTextBox.Text))

{

this.nameGroup = this.nameGroupTextBox.Text;

}

else

{

MessageBox.Show("name group is empty");

return;

}

if (!string.IsNullOrEmpty(this.specialityTextBox.Text))

{

this.speciality = this.specialityTextBox.Text;

}

else

{

MessageBox.Show("speciality is empty");

return;

}

if (!int.TryParse(this.numberOfStudentTextBox.Text, out numberOfStudent))

{

MessageBox.Show("number Of Student is empty");

return;

}

if (!int.TryParse(this.semesterTextBox.Text, out semester))

{

MessageBox.Show("semester is empty");

return;

}

if (EXEC == "update\_group")

{

sqlExpression = $"select update\_group ({Connection.currentId},'{this.nameGroup}', '{this.speciality}',{this.numberOfStudent},{this.semester})";

}

else if (EXEC == "add\_group")

{

sqlExpression = $"select add\_group ('{this.nameGroup}', '{this.speciality}',{this.numberOfStudent},{this.semester})";

}

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromGroup();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

HelperClasses.cs

using System;

namespace Timetable\_of\_classes

{

public class A2Class

{

public A2Class(int Id, string date\_lesson, int lesson\_number, string group\_name, string classsroom\_name, string capacity\_check)

{

this.Id = Id;

this.date\_lesson = date\_lesson;

this.lesson\_number = lesson\_number;

this.group\_name = group\_name;

this.classsroom\_name = classsroom\_name;

this.capacity\_check = capacity\_check;

}

public int Id { get; set; }

public string date\_lesson { get; set; }

public int lesson\_number { get; set; }

public string group\_name { get; set; }

public string classsroom\_name { get; set; }

public string capacity\_check { get; set; }

}

public class D2Class

{

public D2Class(int Id, string date\_lesson, int lesson\_number, string group\_name, int number\_of\_students)

{

this.Id = Id;

this.date\_lesson = date\_lesson;

this.lesson\_number = lesson\_number;

this.group\_name = group\_name;

this.number\_of\_students = number\_of\_students;

}

public int Id { get; set; }

public string date\_lesson { get; set; }

public int lesson\_number { get; set; }

public string group\_name { get; set; }

public int number\_of\_students { get; set; }

}

public class C2Class

{

public C2Class(int Id, string classroomName, int capacity, string groupName, string fullNameTeacher)

{

this.Id = Id;

this.classroomName = classroomName;

this.capacity = capacity;

this.groupName = groupName;

this.fullNameTeacher = fullNameTeacher;

}

public int Id { get; set; }

public string classroomName { get; set; }

public int capacity { get; set; }

public string groupName { get; set; }

public string fullNameTeacher { get; set; }

}

public class History

{

public History(int Id, int groupId, string operation, string createAt)

{

this.Id = Id;

this.groupId = groupId;

this.operation = operation;

this.createAt = createAt;

}

public int Id { get; set; }

public int groupId { get; set; }

public string operation { get; set; }

public string createAt { get; set; }

}

public class E2Class

{

public E2Class(int Id, string date\_lesson, int lesson\_number, string classroom\_name)

{

this.Id = Id;

this.date\_lesson = date\_lesson;

this.lesson\_number = lesson\_number;

this.classroom\_name = classroom\_name;

}

public int Id { get; set; }

public string date\_lesson { get; set; }

public int lesson\_number { get; set; }

public string classroom\_name { get; set; }

}

}

Lesson.cs

namespace Timetable\_of\_classes

{

public class Lesson

{

public Lesson(int Id, string date\_lesson, int lesson\_number, string group\_name, string specialty, string classroom\_name, string teacher\_name, string subject\_name)

{

this.Id = Id;

this.date\_lesson = date\_lesson;

this.lesson\_number = lesson\_number;

this.group\_name = group\_name;

this.specialty = specialty;

this.classroom\_name = classroom\_name;

this.teacher\_name = teacher\_name;

this.subject\_name = subject\_name;

}

public int Id { get; set; }

public string date\_lesson { get; set; }

public int lesson\_number { get; set; }

public string group\_name { get; set; }

public string specialty { get; set; }

public string classroom\_name { get; set; }

public string teacher\_name { get; set; }

public string subject\_name { get; set; }

}

}

LessonChange.xaml

<Window x:Class="Timetable\_of\_classes.LessonChange"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes"

mc:Ignorable="d"

Title="LessonChange" Height="480" Width="400" MinHeight="480" MinWidth="400" MaxHeight="480" MaxWidth="400" WindowStyle="ToolWindow">

<Grid>

<StackPanel>

<Label Content="Lesson change" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="50" Foreground="#FF545454" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="22"/>

<Label Content="date lesson" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<DatePicker x:Name="calendar" IsDropDownOpen="True" SelectedDateFormat="Short" Width="300" Height="30" SelectedDateChanged="calendar\_SelectedDateChanged" />

<Label Content="lesson time (int)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="timeTextBox" HorizontalAlignment="Center" Height="40" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" Foreground="#FF545454" SelectionBrush="#FF5D5D5D" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="group" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<ComboBox x:Name="comboBoxGroupId" HorizontalAlignment="Center" Width="300" Height="30" SelectionChanged="comboBoxGroupId\_SelectionChanged"/>

<Label Content="classroom" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<ComboBox x:Name="comboBoxClassroomId" HorizontalAlignment="Center" Width="300" Height="30" SelectionChanged="comboBoxClassroomId\_SelectionChanged"/>

<Label Content="Teacher" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<ComboBox x:Name="comboBoxTeacherId" HorizontalAlignment="Center" Width="300" Height="30" Margin="0,0,0,20" SelectionChanged="comboBoxTeacherId\_SelectionChanged"/>

<StackPanel Orientation="Horizontal">

<Button Content="Change" HorizontalAlignment="Left" Margin="50,0,0,0" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click" />

<Button Content="Add" HorizontalAlignment="Left" Margin="20,0,0,0" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click\_1" />

</StackPanel>

</StackPanel>

</Grid>

</Window>

LessonChange.xaml.cs

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace Timetable\_of\_classes

{

/// <summary>

/// Логика взаимодействия для LessonChange.xaml

/// </summary>

public partial class LessonChange : Window

{

Dictionary<int, string> keyValueGroup;

Dictionary<int, string> keyValueClassroom;

Dictionary<int, string> keyValueTeacher;

string currentGroup = null;

string currentClassroom = null;

string currentTeacher = null;

string currentDate = null;

string sqlExpression;

public LessonChange()

{

InitializeComponent();

this.keyValueGroup = new Dictionary<int, string>();

this.keyValueClassroom = new Dictionary<int, string>();

this.keyValueTeacher = new Dictionary<int, string>();

sqlExpression = "SELECT \* FROM Groups";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string nameGroup = (string)reader.GetValue(1);

this.keyValueGroup.Add(id, nameGroup);

}

reader.Close();

this.comboBoxGroupId.ItemsSource = this.keyValueGroup.Values;

sqlExpression = "SELECT \* FROM Classroom";

NpgsqlCommand command1 = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader1 = command1.ExecuteReader();

while (reader.Read())

{

int id = (int)reader1.GetValue(0);

string nameClassroom = (string)reader1.GetValue(1);

this.keyValueClassroom.Add(id, nameClassroom);

}

reader.Close();

this.comboBoxClassroomId.ItemsSource = this.keyValueClassroom.Values;

sqlExpression = "SELECT \* FROM Teacher";

NpgsqlCommand command2 = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader2 = command2.ExecuteReader();

while (reader.Read())

{

int id = (int)reader2.GetValue(0);

string nameTeacher = (string)reader2.GetValue(1);

this.keyValueTeacher.Add(id, nameTeacher);

}

reader.Close();

this.comboBoxTeacherId.ItemsSource = this.keyValueTeacher.Values;

}

private void calendar\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

DateTime selectedDate = (DateTime)calendar.SelectedDate;

this.currentDate = selectedDate.ToString("yyyy-MM-dd");

}

private void comboBoxGroupId\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

this.currentGroup = this.comboBoxGroupId.SelectedItem.ToString();

}

private void comboBoxClassroomId\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

this.currentClassroom = this.comboBoxClassroomId.SelectedItem.ToString();

}

private void comboBoxTeacherId\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

this.currentTeacher = this.comboBoxTeacherId.SelectedItem.ToString();

}

private async void UpdateOdAddTeacher(string EXEC)

{

int groupt\_id = keyValueGroup.FirstOrDefault(x => x.Value == this.currentGroup).Key;

int classroom\_id = keyValueClassroom.FirstOrDefault(x => x.Value == this.currentClassroom).Key;

int teacher\_id = keyValueTeacher.FirstOrDefault(x => x.Value == this.currentTeacher).Key;

if (EXEC == "update\_lesson")

{

sqlExpression = $"select {EXEC} ({Connection.currentId},'{this.currentDate}',{this.timeTextBox.Text},{groupt\_id},{classroom\_id},{teacher\_id})";

}

else if (EXEC == "add\_lesson")

{

sqlExpression = $"select {EXEC} ('{this.currentDate}',{this.timeTextBox.Text},{groupt\_id},{classroom\_id},{teacher\_id})";

}

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectfromLesson();

this.Close();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

UpdateOdAddTeacher("update\_lesson");

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

UpdateOdAddTeacher("add\_lesson");

}

}

}

MainWindows.xaml

<Window x:Class="Timetable\_of\_classes.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes" xmlns:col="clr-namespace:System.Collections;assembly=mscorlib"

mc:Ignorable="d"

Title="MainWindow" Height="600" Width="1000" Closing="Window\_Closing">

<Grid Background="White">

<Label Content="database IBC" HorizontalAlignment="Center" VerticalAlignment="Top" FontSize="30" Foreground="#FF5F5F5F" Margin="401,0,352,0" Width="231"/>

<DataGrid x:Name="GroupGrid" Margin="50,50,39.5,100.4" BorderBrush="Black" Background="#FFEDEDED" MouseUp="GroupGrid\_MouseUp"/>

<Button Content="Group" HorizontalAlignment="Left" Margin="50,0,0,60.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click"/>

<Button Content="Subject" HorizontalAlignment="Left" Margin="155,0,0,60.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_2"/>

<Button Content="Teacher" HorizontalAlignment="Left" Margin="260,0,0,60.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_3"/>

<Button Content="Classroom" HorizontalAlignment="Left" Margin="365,0,0,60.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_4"/>

<Button Content="Lesson" HorizontalAlignment="Left" Margin="470,0,0,60.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_13"/>

<Button Content="Change\Add" HorizontalAlignment="Left" Margin="850,0,0,20.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFC2C2C2" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_5"/>

<Button Content="Delete" HorizontalAlignment="Left" Margin="850,0,0,60" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFC2C2C2" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_7"/>

<Button Content="2.a" HorizontalAlignment="Left" Margin="50,0,0,20.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_8"/>

<Button Content="2.c" HorizontalAlignment="Left" Margin="260,0,0,20.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_10"/>

<Button Content="History" HorizontalAlignment="Left" Margin="365,0,0,20.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_11"/>

<Button Content="2.e" HorizontalAlignment="Left" Margin="155,0,0,20.4" VerticalAlignment="Bottom" Width="100" Height="30" Background="#FFEEEEEE" Foreground="#FF404040" FontSize="16" Click="Button\_Click\_12"/>

</Grid>

</Window>

MainWindow.xaml.cs

using Npgsql;

using System;

using System.Data.SqlClient;

using System.Windows;

namespace Timetable\_of\_classes

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

//public enum ObjectName { Classroom, Group, LessonNumber, Teacher, Subject }

public partial class MainWindow : Window

{

Authorization authorization;

GroupChange groupChange;

SubjectChange subjectChange;

TeacherChange teacherChange;

ClassroomChange classroomChange;

LessonChange lessonChange;

private string sqlExpression;

public MainWindow()

{

InitializeComponent();

this.authorization = new Authorization(this);

this.authorization.Show();

Connection.GroupGrid = this.GroupGrid;

this.Hide();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

this.GroupGrid.ItemsSource = Connection.SelectFromGroup();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

this.GroupGrid.ItemsSource = Connection.SelectfromSubject();

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

this.GroupGrid.ItemsSource = Connection.SelectFromTeacher();

}

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

this.GroupGrid.ItemsSource = Connection.SelectFromClassroom();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

Connection.sqlConnection.Close();

}

private void GroupGrid\_MouseUp(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

if (Connection.objectName == ObjectName.Classroom)

{

if (GroupGrid.SelectedItem is Classroom)

{

Classroom path = GroupGrid.SelectedItem as Classroom;

MessageBox.Show("id: " + path.id.ToString());

Connection.currentId = path.id;

Connection.objectName = ObjectName.Classroom;

return;

}

}

if (Connection.objectName == ObjectName.Group)

{

if (GroupGrid.SelectedItem is Group)

{

Group path = GroupGrid.SelectedItem as Group;

MessageBox.Show("id: " + path.Id.ToString());

Connection.currentId = path.Id;

Connection.objectName = ObjectName.Group;

return;

}

}

if (Connection.objectName == ObjectName.Subject)

{

if (GroupGrid.SelectedItem is Subject)

{

Subject path = GroupGrid.SelectedItem as Subject;

MessageBox.Show("id: " + path.id.ToString());

Connection.currentId = path.id;

Connection.objectName = ObjectName.Subject;

return;

}

}

if (Connection.objectName == ObjectName.Teacher)

{

if (GroupGrid.SelectedItem is Teacher)

{

Teacher path = GroupGrid.SelectedItem as Teacher;

MessageBox.Show("id: " + path.id.ToString());

Connection.currentId = path.id;

Connection.objectName = ObjectName.Teacher;

return;

}

}

}

private void Button\_Click\_5(object sender, RoutedEventArgs e)

{

if (Connection.objectName == ObjectName.Group)

{

this.groupChange = new GroupChange();

this.groupChange.Owner = this;

this.groupChange.Show();

}

if (Connection.objectName == ObjectName.Subject)

{

this.subjectChange = new SubjectChange();

this.subjectChange.Owner = this;

this.subjectChange.Show();

}

if (Connection.objectName == ObjectName.Teacher)

{

this.teacherChange = new TeacherChange();

this.teacherChange.Owner = this;

this.teacherChange.Show();

}

if (Connection.objectName == ObjectName.Classroom)

{

this.classroomChange = new ClassroomChange();

this.classroomChange.Owner = this;

this.classroomChange.Show();

}

if (Connection.objectName == ObjectName.Lesson)

{

this.lessonChange = new LessonChange();

this.lessonChange.Owner = this;

this.lessonChange.Show();

}

}

private async void Button\_Click\_7(object sender, RoutedEventArgs e)

{

try

{

if (Connection.objectName == ObjectName.Classroom)

{

if (GroupGrid.SelectedItem is Classroom)

{

Classroom path = GroupGrid.SelectedItem as Classroom;

Connection.currentId = path.id;

Connection.objectName = ObjectName.Classroom;

sqlExpression = $"select delete\_classroom ({path.id})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromClassroom();

return;

}

}

if (Connection.objectName == ObjectName.Group)

{

if (GroupGrid.SelectedItem is Group)

{

Group path = GroupGrid.SelectedItem as Group;

Connection.currentId = path.Id;

Connection.objectName = ObjectName.Group;

sqlExpression = $"select delete\_group ({path.Id})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromGroup();

return;

}

}

if (Connection.objectName == ObjectName.Subject)

{

if (GroupGrid.SelectedItem is Subject)

{

Subject path = GroupGrid.SelectedItem as Subject;

Connection.currentId = path.id;

Connection.objectName = ObjectName.Subject;

sqlExpression = $"select delete\_subject ({path.id})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectfromSubject();

return;

}

}

if (Connection.objectName == ObjectName.Teacher)

{

if (GroupGrid.SelectedItem is Teacher)

{

Teacher path = GroupGrid.SelectedItem as Teacher;

Connection.currentId = path.id;

Connection.objectName = ObjectName.Teacher;

sqlExpression = $"select delete\_teacher ({path.id})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromTeacher();

return;

}

}

if (Connection.objectName == ObjectName.Lesson)

{

if (GroupGrid.SelectedItem is Lesson)

{

Lesson path = GroupGrid.SelectedItem as Lesson;

Connection.currentId = path.Id;

Connection.objectName = ObjectName.Lesson;

sqlExpression = $"select delete\_lesson ({path.Id})";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromTeacher();

return;

}

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Button\_Click\_8(object sender, RoutedEventArgs e)

{

Connection.GroupGrid.ItemsSource = Connection.A2();

}

private void Button\_Click\_11(object sender, RoutedEventArgs e)

{

Connection.GroupGrid.ItemsSource = Connection.histories();

}

private void Button\_Click\_12(object sender, RoutedEventArgs e)

{

Connection.GroupGrid.ItemsSource = Connection.E2();

}

private void Button\_Click\_13(object sender, RoutedEventArgs e)

{

Connection.GroupGrid.ItemsSource = Connection.SelectfromLesson();

}

}

}

Subject.cs

namespace Timetable\_of\_classes

{

public class Subject

{

public Subject(int id, string name, string classroom\_id = null)

{

this.id = id;

this.name = name;

this.classroom\_id = classroom\_id;

}

public int id { get; set; }

public string name { get; set; }

public string classroom\_id { get; set; }

}

}

SubjectChange.xaml.cs

<Window x:Class="Timetable\_of\_classes.SubjectChange"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes"

mc:Ignorable="d"

Title="SubjectChange" Height="450" Width="400" MinHeight="450" MinWidth="400" MaxHeight="450" MaxWidth="400" WindowStyle="ToolWindow">

<Grid>

<StackPanel>

<Label Content="Subject change" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="50" Foreground="#FF545454" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="22"/>

<Label Content="Subject name (string)" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="nameSubjectTextBox" HorizontalAlignment="Center" Height="30" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" SelectionBrush="#FF5D5D5D" Foreground="#FF545454" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="Lesson" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<ComboBox x:Name="comboBoxLessonId" HorizontalAlignment="Center" Width="300" Height="30" Margin="0,0,0,30" SelectionChanged="comboBoxLessonId\_SelectionChanged"/>

<StackPanel Orientation="Horizontal">

<Button Content="Change" HorizontalAlignment="Left" Margin="50,0,0,0" VerticalAlignment="Top" Width="140" Height="30" Foreground="White" Background="#FF7B8893" Click="Button\_Click"/>

<Button Content="Add" HorizontalAlignment="Left" Margin="20,0,0,0" VerticalAlignment="Top" Width="140" Height="30" Foreground="White" Background="#FF7B8893" Click="Button\_Click\_1"/>

</StackPanel>

</StackPanel>

</Grid>

</Window>

SubjectChange.cs

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Timetable\_of\_classes

{

/// <summary>

/// Логика взаимодействия для SubjectChange.xaml

/// </summary>

public partial class SubjectChange : Window

{

string nameSubject;

public Dictionary<int, string> keyValueLesson;

string currentLesson = null;

private string sqlExpression;

public SubjectChange()

{

InitializeComponent();

try

{

this.keyValueLesson = new Dictionary<int, string>();

string sqlExpression = "SELECT \* FROM Lesson";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string date\_lesson = (string)reader.GetValue(1).ToString();

int lesson\_number = (int)reader.GetValue(2);

this.keyValueLesson.Add(id, date\_lesson + " " + lesson\_number.ToString());

}

reader.Close();

this.comboBoxLessonId.ItemsSource = this.keyValueLesson.Values;

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

UpdateOrAddSubject("update\_subject");

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

UpdateOrAddSubject("add\_subject");

}

private async void UpdateOrAddSubject(string EXEC)

{

try

{

if (!string.IsNullOrEmpty(this.nameSubjectTextBox.Text))

{

this.nameSubject = this.nameSubjectTextBox.Text;

}

else

{

MessageBox.Show("name subject is empty");

return;

}

if (EXEC == "update\_subject")

{

if (string.IsNullOrEmpty(this.comboBoxLessonId.Text))

{

sqlExpression = $"select {EXEC} ({Connection.currentId},'{this.nameSubject}')";

}

else

{

int subjectId = keyValueLesson.FirstOrDefault(x => x.Value == this.currentLesson).Key;

sqlExpression = $"select {EXEC} ({Connection.currentId},'{this.nameSubject}',{subjectId})";

}

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectfromSubject();

this.Close();

}

else if(EXEC == "add\_subject")

{

if (string.IsNullOrEmpty(this.comboBoxLessonId.Text))

{

sqlExpression = $"select {EXEC} ('{this.nameSubject}')";

}

else

{

int subjectId = keyValueLesson.FirstOrDefault(x => x.Value == this.currentLesson).Key;

sqlExpression = $"select {EXEC} ('{this.nameSubject}',{subjectId})";

}

NpgsqlCommand command2 = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command2.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectfromSubject();

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void comboBoxLessonId\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

this.currentLesson = this.comboBoxLessonId.SelectedItem.ToString();

}

}

}

Teacher.cs

namespace Timetable\_of\_classes

{

public class Teacher

{

public Teacher(int id, string full\_name, string subject\_name)

{

this.id = id;

this.full\_name = full\_name;

this.subject\_name = subject\_name;

}

public int id { get; set; }

public string full\_name { get; set; }

public string subject\_name { get; set; }

}

}

TeacherChange.xaml.cs

<Window x:Class="Timetable\_of\_classes.TeacherChange"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Timetable\_of\_classes"

mc:Ignorable="d"

Title="TeacherChange" Height="500" Width="450" MinHeight="550" MinWidth="400" MaxHeight="550" MaxWidth="400" WindowStyle="ToolWindow">

<Grid>

<StackPanel>

<Label Content="Teacher change" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="50" Foreground="#FF545454" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="22"/>

<Label Content="teacher full name" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<TextBox x:Name="teacherFullName" HorizontalAlignment="Center" Height="40" TextWrapping="Wrap" VerticalAlignment="Top" Width="300" SelectionBrush="#FF5D5D5D" Foreground="#FF545454" VerticalContentAlignment="Center" Padding="5,0,0,0"/>

<Label Content="Subject" HorizontalAlignment="Center" VerticalAlignment="Top" Width="300" Height="30" Foreground="#FF545454" VerticalContentAlignment="Bottom"/>

<ComboBox x:Name="comboBoxSubjects" HorizontalAlignment="Center" Width="300" Height="30" Margin="0,0,0,30" SelectionChanged="comboBoxSubjects\_SelectionChanged" />

<StackPanel Orientation="Horizontal">

<Button Content="Change" HorizontalAlignment="Left" Margin="50,0,0,0" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click"/>

<Button Content="Add" HorizontalAlignment="Left" Margin="20,0,0,0" VerticalAlignment="Top" Width="140" Height="40" Foreground="White" Background="#FF7B8893" Click="Button\_Click\_1"/>

</StackPanel>

</StackPanel>

</Grid>

</Window>

TeacherChange.cs

using Npgsql;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

namespace Timetable\_of\_classes

{

/// <summary>

/// +

/// Логика взаимодействия для TeacherChange.xaml

/// </summary>

public partial class TeacherChange : Window

{

Dictionary<int, string> keyValueSubject;

string currentSubject = null;

private string sqlExpression;

public TeacherChange()

{

InitializeComponent();

this.keyValueSubject = new Dictionary<int, string>();

string sqlExpression = "SELECT \* FROM Subject";

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int id = (int)reader.GetValue(0);

string nameSubject = (string)reader.GetValue(1);

this.keyValueSubject.Add(id, nameSubject);

}

reader.Close();

this.comboBoxSubjects.ItemsSource = this.keyValueSubject.Values;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

UpdateOdAddTeacher("update\_teacher");

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

UpdateOdAddTeacher("add\_teacher");

}

private async void UpdateOdAddTeacher(string EXEC)

{

if (string.IsNullOrEmpty(currentSubject) || string.IsNullOrEmpty(this.teacherFullName.Text))

{

MessageBox.Show("Subject is empty");

}

else

{

if (EXEC == "update\_teacher")

{

int subjectId = keyValueSubject.FirstOrDefault(x => x.Value == this.currentSubject).Key;

sqlExpression = $"select {EXEC} ({Connection.currentId},'{this.teacherFullName.Text}',{subjectId})";

}

else if (EXEC == "add\_teacher")

{

int subjectId = keyValueSubject.FirstOrDefault(x => x.Value == this.currentSubject).Key;

sqlExpression = $"select {EXEC} ('{this.teacherFullName.Text}',{subjectId})";

}

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, Connection.sqlConnection);

await command.ExecuteNonQueryAsync();

Connection.GroupGrid.ItemsSource = Connection.SelectFromTeacher();

this.Close();

}

}

private void comboBoxSubjects\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

this.currentSubject = comboBoxSubjects.SelectedItem.ToString();

}

}

}

# Вывод

Написав данную курсовую работу я получил основные навыки по созданию баз данных и написанию sql запросов. Эти навыки безусловно помогут мне в будущем, т.к. практически в каждой IT сфере навыки работы с базами данных необходимы.

# Источники

1. <https://metanit.com/sharp/>
2. <https://info-comp.ru/>
3. <https://www.postgresql.org/docs/>