|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**  **РОССИЙСКОЙ ФЕДЕРАЦИИ**  **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**  **ВЫСШЕГО ОБРАЗОВАНИЯ**  **«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  **ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ имени К.Г. РАЗУМОВСКОГО**  **(Первый казачий университет)»**  **(ФГБОУ ВО «МГУТУ им. К.Г. Разумовского (ПКУ)»)**  **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**  **О Т Ч Е Т**  **по прохождению учебной практики**  **профессионального модуля**  **ПМ.02 Разработка и администрирование баз данных**  **09.02.03 Программирование в компьютерных системах**  **Выполнил студент группы 090203-9o-19/2 специальность 09.02.03 «Программирование в компьютерных системах»**  Алексеев Артём Анатольевич фамилия, имя, отчество студента  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись студента  **Москва**  **2022 год** |

**СОДЕРЖАНИЕ**

[1. ЦЕЛИ И ЗАДАЧИ ПРАКТИКИ 3](#_Toc107276191)

[2. ОСВОЕННЫЕ ОБЩИЕ И ПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ 4](#_Toc107276192)

[3. ОПИСАНИЕ ВЫПОЛНЕННОЙ РАБОТЫ 6](#_Toc107276193)

[4. ТРУДНОСТИ, ВОЗНИКШИЕ ПРИ ВЫПОЛНЕНИИ ЗАДАНИЙ, И ПУТИ ИХ РАЗРЕШЕНИЯ 8](#_Toc107276194)

[5. ВЫВОДЫ О ПОЛУЧЕННЫХ ПРАКТИЧЕСКИХ НАВЫКАХ 9](#_Toc107276195)

[6. ПРИЛОЖЕНИЕ 10](#_Toc107276196)

# ЦЕЛИ И ЗАДАЧИ ПРАКТИКИ

Цели:

* Закрепление навыков разработки баз данных в среде MS SQL;
* Закрепление навыков работы с Entity Framework;
* Написание прикладного приложения к разработанной базе данных.

Задачи:

* Разработка логической модели;
* Проработка объектов базы данных;
* Определение отношений;
* Импорт данных из Excel в базу данных;
* Создание скрипта на заполнение таблиц;
* Написание запросов на языке SQL;
* Работа с MS SQL Server в С# через Entity Framework;
* Изучение новых методов работы с SQL по средствам Entity Framework;
* Создание триггеров и процедур.

# **ОСВОЕННЫЕ ОБЩИЕ И ПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **УМЕНИЯ** | | | | | |
| **Код** | **Наименование результата обучения** | | | **% выполнения части заданий, отвечающие основным показателям оценки результата** | |
| У 1 | Создаёт объекты баз данных в современных СУБД и управляет доступом к этим объектам; | | | 97% | |
| У 2 | Работает с современными Case-средствами проектирования баз данных; | | | 92% | |
| У 3 | Формирует и настраивает схему базы данных; | | | 91% | |
| У 4 | Разрабатывает прикладные программы с использованием языка SQL; | | | 86% | |
| **ОБЩИЕ КОМПЕТЕНЦИИ** | | | | | |
| **Код** | | | **Наименование результата обучения** | **Сформирована**  **/не сформирована** | |
| ОК 1 | | | Понимает сущность и социальную значимость своей будущей профессии, проявляет к ней устойчивый интерес | Сформирована | |
| ОК 2 | | | Организовывает собственную деятельность, выбирает типовые методы и способы выполнения профессиональных задач, оценивает их эффективность и качество | Сформирована | |
| ОК 3 | | | Принимает решения в стандартных и нестандартных ситуациях и несёт за них ответственность. | Сформирована | |
| ОК 4 | | | Осуществляет поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития. | Сформирована | |
| ОК 5 | | | Использует информационно-коммуникационные технологии в профессиональной деятельности. | Сформирована | |
| ОК 6 | | | Работает в коллективе и в команде, эффективно общается с коллегами, руководством, потребителями. | Сформирована | |
| ОК 7 | | | Берёт на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий. | Сформирована | |
| ОК 8 | | | Самостоятельно определяет задачи профессионального и личностного развития, занимается самообразованием, осознанно планирует повышение квалификации. | Сформирована | |
| ОК 9 | | | Ориентируется в условиях частой смены технологий в профессиональной деятельности. | Сформирована | |
| **ПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ** | | | | | |
| **Код** | | **Наименование результата обучения** | | | **% выполнения части заданий, отвечающие основным показателям оценки результата** |
| ПК 2.1 | | Разрабатывает объекты базы данных | | | 100% |
| ПК 2.2 | | Реализовывает базу данных в конкретной СУБД | | | 100% |
| ПК 2.3 | | Решает вопросы администрирования базы данных | | | 85% |
| ПК 2.4 | | Реализовывает методы и технологии защиты информации в базах данных | | | 85% |

# ОПИСАНИЕ ВЫПОЛНЕННОЙ РАБОТЫ

В данной учебной практике была реализована база данных «Павильоны», и приложение использующее данную базу данных. Сначала была разработана логическая модель базы данных (объекты и их отношения) выполненная в ER Assistant.

Следующим действием была нормализация отношений, а затем уже превращение из логической модели в физическую модель базы данных с последующим импортом данных из вручную подготовленных Excel таблиц. В дополнение к импорту данных сделал полный скрип для создания базы данных. После импортирования данных была создана процедура проноса сроков аренды и павильонов.

Был создан WPF проект с NuGet пакетом Entity Framework в целях создания оконного приложения с привязкой к Git-репозиторию для сохранения результатов работы по практике. Используя средства разработки Visual Studio выполнил создание классов через создание EDM (модель данных с использованием сущностей).

Затем был создан фрагмент приложения для загрузки пользователей и торговых центров. Но он оказался неверным, так что его пришлось переделать.

Следуя предыдущему опыту, я использовал паттерн проектирования MVVM в своих проектах, и эта учебная практика не стала исключением. Данный подход очень удобен по нескольким причинам:

* Возможность итеративного произвольного стиля программирования. Изолированные изменения менее опасны и более удобны для экспериментирования;
* Упрощение модульного тестирования. Блоки кода, изолированные друг от друга, можно тестировать отдельно и вне рабочих сред;
* Повышение удобства обслуживания. Устранение ошибок в несвязанном коде снижает вероятность возникновения регрессий в другом коде.

Он имеет три основные сущности такие как:

* View – представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением;
* ViewModel – модель представления связывает модель и представление через механизм привязки данных;
* Model - модель описывает используемые в приложении данные;

Добавил файл .gitignore для игнорирования файлов сборки, а также выполняемых файлов для исключения их из коммитов в Git-репозиторий.

Сделал интерфейс входа в систему и отображение капчи при вводе неверные данных.

Для «Менеджеров С» добавил интерфейс работы с торговыми центрами (добавление/изменение/удаление). Выбирая конкретный торговый центр можно просмотреть список павильонов данного торгового центра. В списке павильонов также доступно добавление, изменение и удаление. Используя интерфейс просмотра павильонов выбранный павильон можно арендовать или забронировать на определенную дату. Данное действие реализовано через триггер и процедуру аренды павильона.

# ТРУДНОСТИ, ВОЗНИКШИЕ ПРИ ВЫПОЛНЕНИИ ЗАДАНИЙ, И ПУТИ ИХ РАЗРЕШЕНИЯ

В данной практике возникали некоторые трудности.

Первая из них - это проблема обновления модели EDM из базы данных. При изменении объектов базы данных (столбцов или типов) иногда модель EDM просто остается прежней или же добавляет лишние первичные ключи в модель EDM. Помогает обычное пересоздание модели EDM через создание элемент данных в интерфейсе Visual Studio.

Вторая проблема заключалась в том, что была использована одна ViewModel’и для двух View (представлений). Таким образом для каждого View срабатывал отдельных конструктор ViewModel’и что не давало использовать общие данные из двух View. Проблема решилась указанием ViewModel’и как статического ресурса в общих ресурсах приложения.

И третья проблема, связанная с конструктором Visual Studio, которая при создании Binding’ов на элементы управления из ViewModel’и не подгружает их статические значения. Её решить не удалось, хотя в другом проекте это работает нормально.

# ВЫВОДЫ О ПОЛУЧЕННЫХ ПРАКТИЧЕСКИХ НАВЫКАХ

В этой учебной практике я приобрел новые навыки, компетенции и драгоценный опыт разработки приложений, работающих с базой данных. Использовав Linq научился создавать SQL запросы из C# кода используя встроенные методы такие как: select, orderby, join, distinct.

Научился использовать новый вид коллекций, а именно ObservableCollection (наследующийся от интерфейсов INotifyCollectionChanged, INotifyPropertyChanged). Коллекция ObservableCollection позволяет оповещать внешние объекты о том, что коллекция была обновлена тем самым обновляя View (представления) в реальном времени.

Научился пользоваться разметкой управления XAML-документа в полной мере, который служит для описания внешнего вида, действий окон или страниц приложений, а логика приложения полностью отделена от интерфейса что делает паттерн MVVM очень полезным в больших проектах.

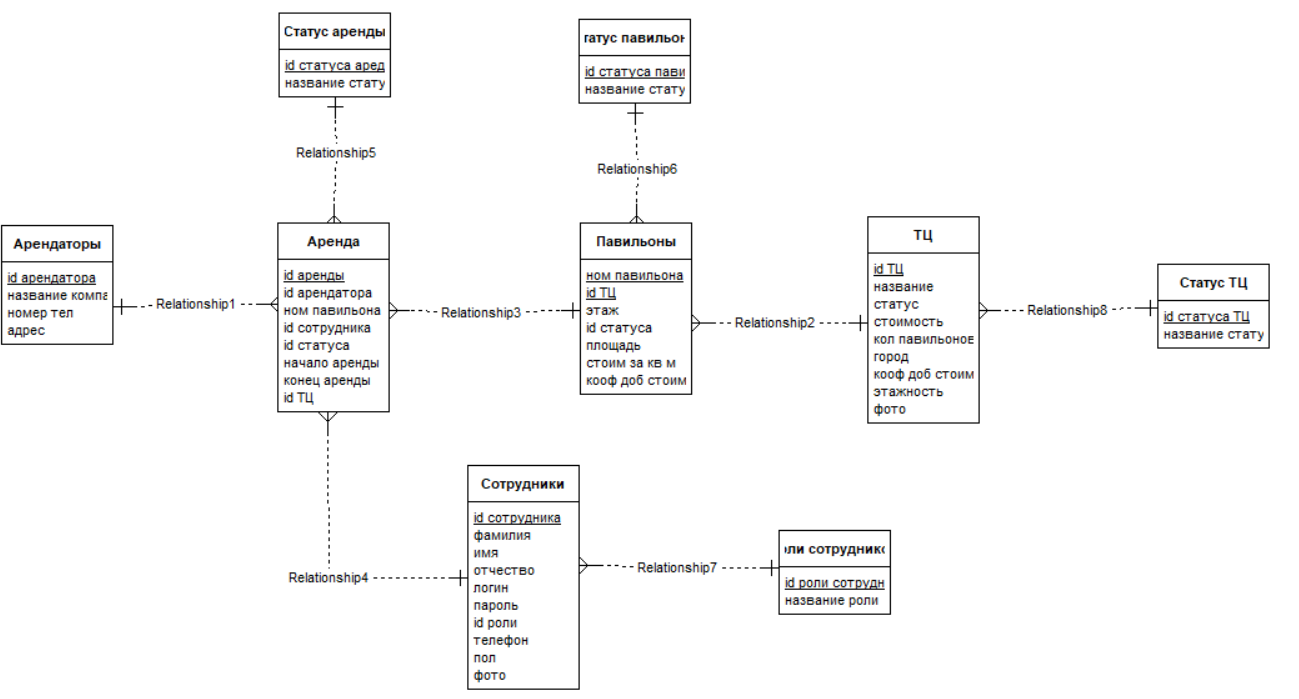
Данная практика вдохновила меня на новый проект связанный с написанием Telegram бота используя MVVM, ООП, MS SQL и многопоточность.

# ПРИЛОЖЕНИЕ

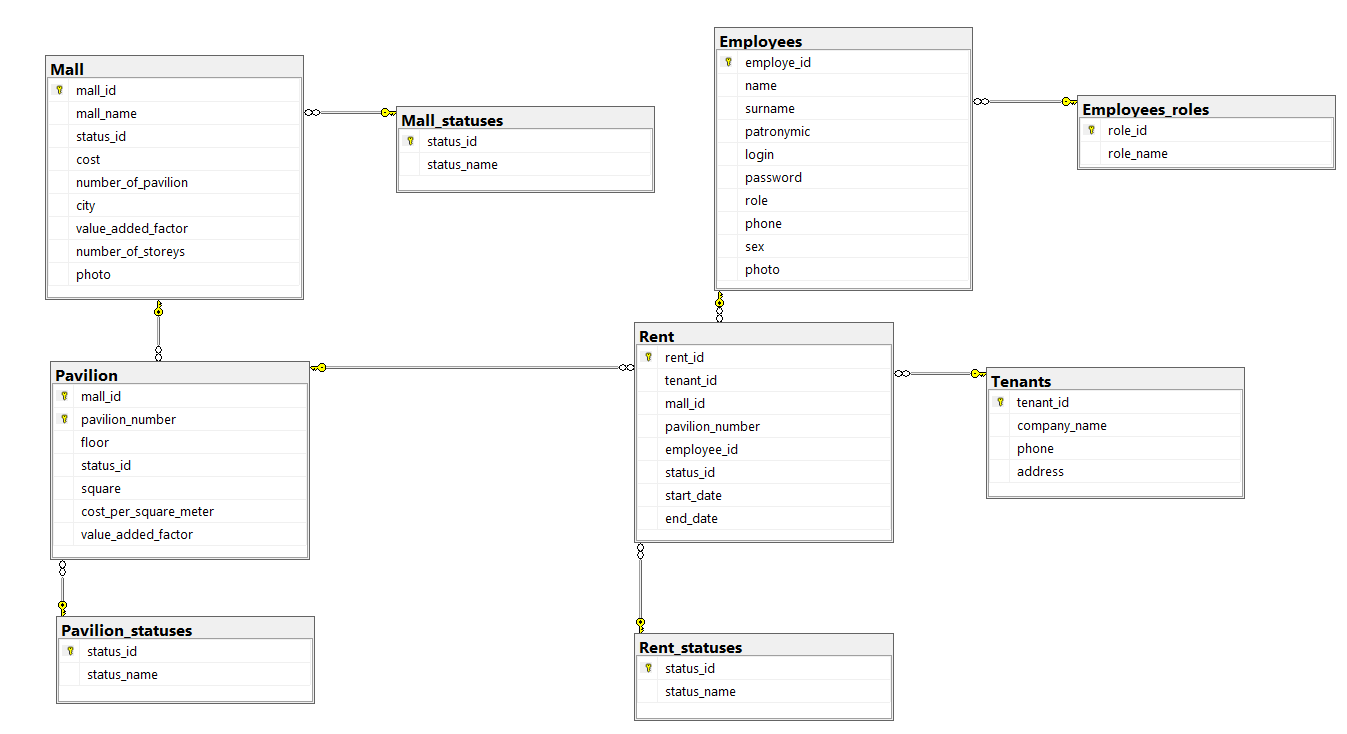
Сессия 1:

**1.1 СОЗДАНИЕ БАЗЫ ДАННЫХ**

**1 Диаграмма базы данных**

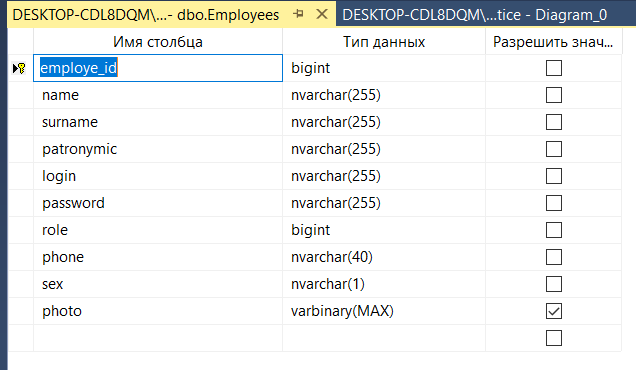


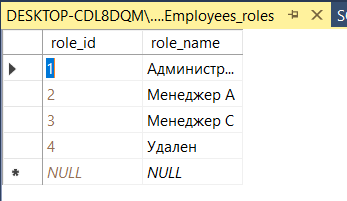
Логическая модель

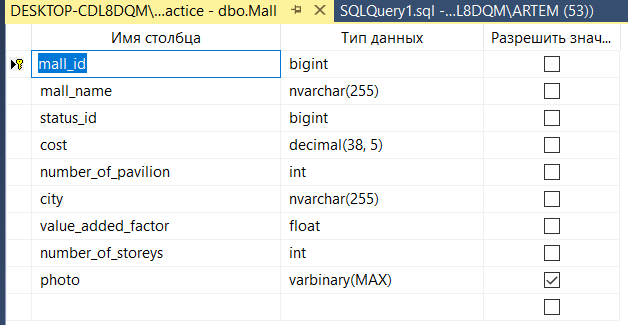


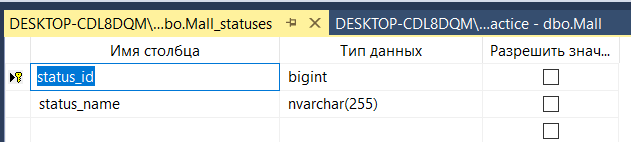
Физическая модель

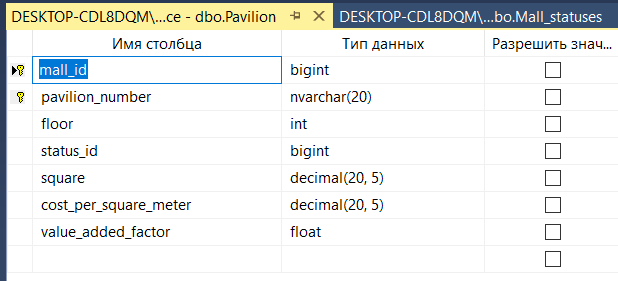
**2 Скриншот таблиц**

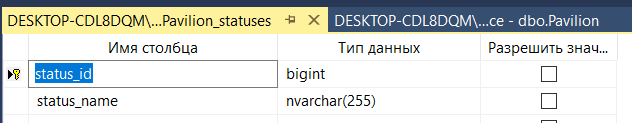


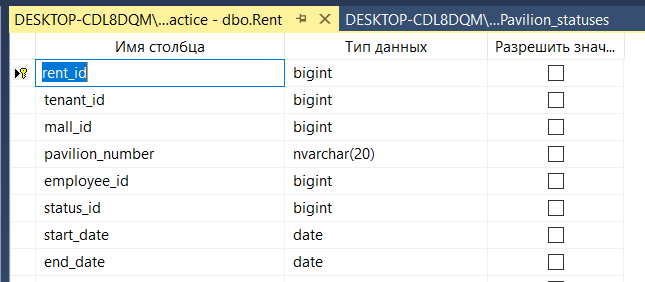
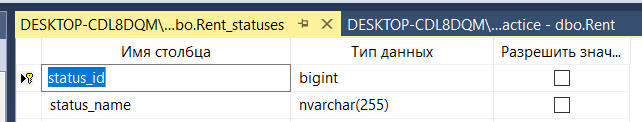


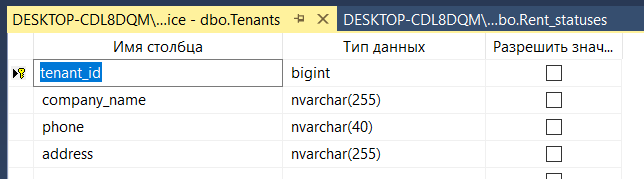




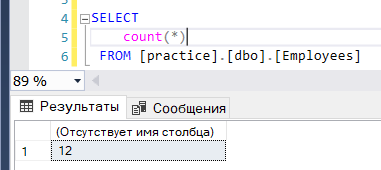
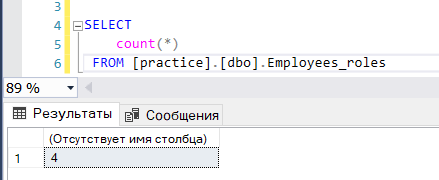
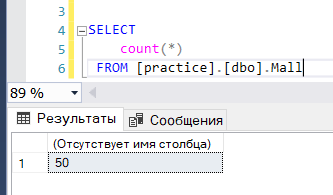
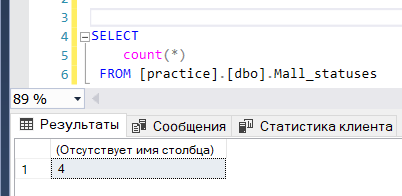
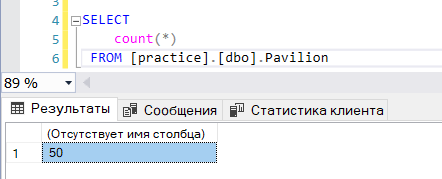
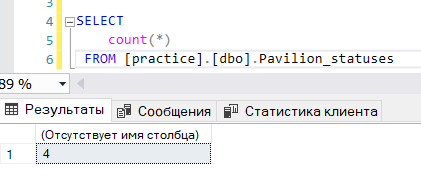
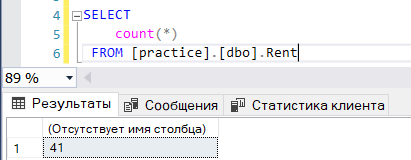
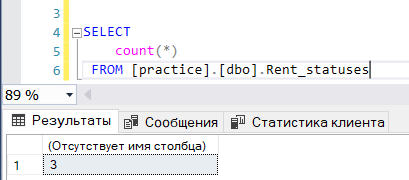
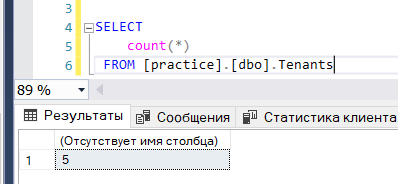






**3 Вывод количества записей в таблицах**

**ЗАГРУЗКА ФОТОГРАФИЙ В БАЗУ ДАННЫХ**

**LoadEmployeesImagesCommand:**

try

{

string pathToDirWithImages = $@"C:\Users\ARTEM\Desktop\КОРОНОВИРУС\21-22\Практика\Ресурсы\Image Сотрудники";

DirectoryInfo directoryInfo = new DirectoryInfo(pathToDirWithImages);

foreach (FileInfo fileInfo in directoryInfo.GetFiles())

{

if (fileInfo.Extension == ".jpg")

{

string fileName = fileInfo.FullName;

string[] fullName = fileName.Substring(fileName.LastIndexOf('\\') + 1).Split('.')[0].Split(' ');

string surname = fullName[0];

string name = fullName[1];

string patronymic = fullName[2];

Employees employee = (from em in Manager.Instance.Context.Employees

where em.name == name && em.surname == surname && em.patronymic == patronymic

select em).FirstOrDefault();

if (employee != null)

{

Debug.WriteLine(fileInfo);

employee.photo = Tools.GetImageBytes(fileInfo.FullName);

}

}

}

Manager.Instance.Context.SaveChanges();

Debug.WriteLine("Фото сотрудников загружены.");

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

**LoadMallImagesCommand:**

try

{

string pathToDirWithImages = $@"C:\Users\ARTEM\Desktop\КОРОНОВИРУС\21-22\Практика\Ресурсы\Image ТЦ";

DirectoryInfo directoryInfo = new DirectoryInfo(pathToDirWithImages);

foreach (FileInfo fileInfo in directoryInfo.GetFiles())

{

if (fileInfo.Extension == ".jpg")

{

string fileName = fileInfo.FullName;

string mallName = fileName.Substring(fileName.LastIndexOf('\\') + 1).Split('.')[0];

Mall mall = (from m in Manager.Instance.Context.Mall

where m.mall\_name == mallName

select m).FirstOrDefault();

if (mall != null)

{

Debug.WriteLine(fileInfo);

mall.photo = Tools.GetImageBytes(fileInfo.FullName);

}

}

}

Manager.Instance.Context.SaveChanges();

Debug.WriteLine("Фото ТЦ загружены.");

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

**ПЕРЕНОС СРОКА АРЕНДЫ**

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GO

IF EXISTS (SELECT name FROM sysobjects WHERE name = 'UpdateRentsAndPavilions' AND type = 'P')

BEGIN

DROP PROCEDURE UpdateRentsAndPavilions;

END;

GO

GO

CREATE PROCEDURE UpdateRentsAndPavilions AS

BEGIN

declare @years int = 3;

UPDATE rent

SET

[start\_date] = DATEADD(YEAR, @years, [start\_date]),

[end\_date] = DATEADD(YEAR, @years, [end\_date]);

UPDATE rent

SET status\_id =

CASE

WHEN [start\_date] <= GETDATE() AND [end\_date] >= GETDATE()

THEN 1 --открыт

WHEN [start\_date] > GETDATE()

THEN 2 --ожидание

WHEN [end\_date] < GETDATE()

THEN 3 --закрыт

END;

UPDATE Pavilion

SET status\_id =

CASE

WHEN EXISTS (SELECT rent\_id FROM rent WHERE [start\_date] <= GETDATE() AND [end\_date] >= GETDATE() AND pavilion\_number = p.pavilion\_number)

THEN 3 --арендован

WHEN EXISTS (SELECT rent\_id FROM rent WHERE [start\_date] > GETDATE() AND pavilion\_number = p.pavilion\_number)

THEN 2 --забронирован

ELSE 1 --свободен

END

FROM Pavilion p

JOIN rent r ON

r.pavilion\_number = p.pavilion\_number

END;

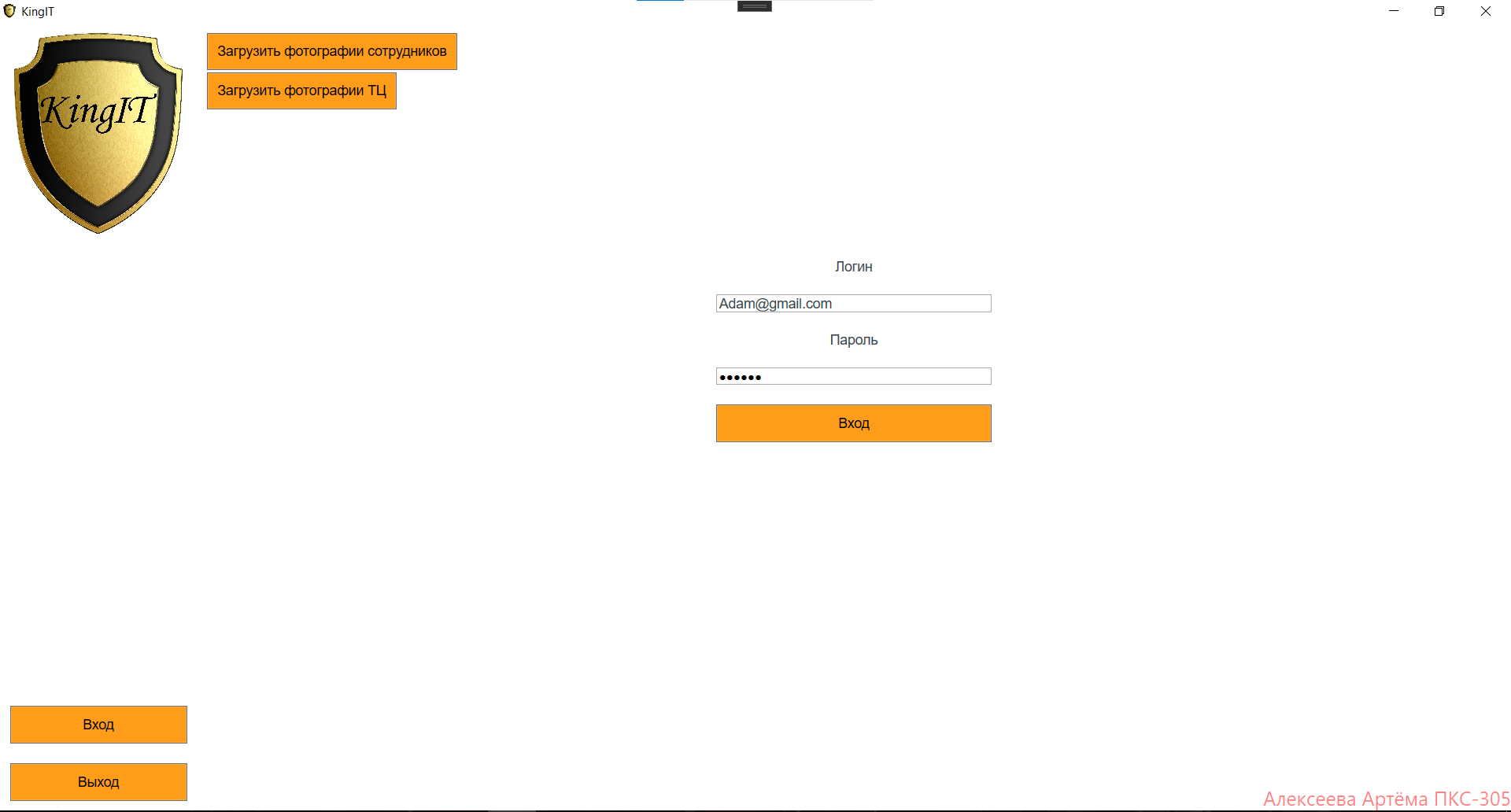
GO

**Вызов процедуры**

|  |
| --- |
| Exec UpdateRentsAndPavilions;  **1.2 СОЗДАНИЕ ПРИЛОЖЕНИЯ** |

**1.3 АВТОРИЗАЦИЯ**

Скриншоты



Код приложения

**LoginPageViewModel.cs :**

using Project.Infrastructure.Commands;

using Project.ViewModels.Base;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Windows;

using System.Windows.Input;

using System;

using System.Drawing;

using System.Linq;

using System.Windows.Interop;

using System.Windows.Controls;

using Project.Models;

using System.Data.Linq;

namespace Project.ViewModels

{

internal class LoginPageViewModel : ViewModel

{

#region Логин

private string \_loginLabelName = "Логин";

/// <summary>

/// Логин

/// </summary>

public string LoginLabelName

{

get => \_loginLabelName;

set => Set(ref \_loginLabelName, value);

}

#endregion

#region Заголовок пароля

private string \_passwordLabelName = "Пароль";

/// <summary>

/// Заголовок пароля

/// </summary>

public string PasswordLabelName

{

get => \_passwordLabelName;

set => Set(ref \_passwordLabelName, value);

}

#endregion

#region Текст кнопки входа

private string \_exitButtonText = "Вход";

/// <summary>

/// Текст кнопки входа

/// </summary>

public string EntryButtonText

{

get => \_exitButtonText;

set => Set(ref \_exitButtonText, value);

}

#endregion

#region Текст логина

private string \_loginText = "Adam@gmail.com";

/// <summary>

/// Текст логина

/// </summary>

public string LoginText

{

get => \_loginText;

set => Set(ref \_loginText, value);

}

#endregion

#region Видимость кнопки загрузки фотографий сотрудников

private Visibility \_loadEmployeesImagesVisibility = Visibility.Visible;

/// <summary>

/// Видимость кнопки загрузки фото сотрудников

/// </summary>

public Visibility LoadEmployeesImagesVisibility

{

get => \_loadEmployeesImagesVisibility;

set => Set(ref \_loadEmployeesImagesVisibility, value);

}

#endregion

#region Название кнопки загрузки фото сотрудников

private string \_employeesImagesButtonText = "Загрузить фотографии сотрудников";

/// <summary>

/// Название кнопки загрузки фото сотрудников

/// </summary>

public string EmployeesImagesButtonText

{

get => \_employeesImagesButtonText;

set => Set(ref \_employeesImagesButtonText, value);

}

#endregion

#region Видимость кнопки загрузки фотографий ТЦ

private Visibility \_loadMallImagesVisibility = Visibility.Visible;

/// <summary>

/// Видимость кнопки загрузки фото ТЦ

/// </summary>

public Visibility LoadMallImagesVisibility

{

get => \_loadMallImagesVisibility;

set => Set(ref \_loadMallImagesVisibility, value);

}

#endregion

#region Название кнопки загрузки фото ТЦ

private string \_loadMallImagesButtonText = "Загрузить фотографии ТЦ";

/// <summary>

/// Название кнопки загрузки фото ТЦ

/// </summary>

public string LoadMallImagesButtonText

{

get => \_loadMallImagesButtonText;

set => Set(ref \_loadMallImagesButtonText, value);

}

#endregion

#region Название текста изминения капчи

private string \_updateCaptchaText = "Изменить картинку";

/// <summary>

/// Название кнопки загрузки фото ТЦ

/// </summary>

public string UpdateCaptchaText

{

get => \_updateCaptchaText;

set => Set(ref \_updateCaptchaText, value);

}

#endregion

#region Видимость поля ввода капчи

private Visibility \_сaptchaTextBoxVisibility = Visibility.Hidden;

/// <summary>

/// Видимость поля ввода капчи

/// </summary>

public Visibility CaptchaTextBoxVisibility

{

get => \_сaptchaTextBoxVisibility;

set => Set(ref \_сaptchaTextBoxVisibility, value);

}

#endregion

#region Видимость текста изминения капчи

private Visibility \_updateCaptchaTextBlockVisibility = Visibility.Hidden;

/// <summary>

/// Видимость текста изминения капчи

/// </summary>

public Visibility UpdateCaptchaTextBlockVisibility

{

get => \_updateCaptchaTextBlockVisibility;

set => Set(ref \_updateCaptchaTextBlockVisibility, value);

}

#endregion

#region Видимость текста изминения капчи

private Visibility \_captchaTextVisibility = Visibility.Hidden;

/// <summary>

/// Видимость текста изминения капчи

/// </summary>

public Visibility CaptchaTextVisibility

{

get => \_captchaTextVisibility;

set => Set(ref \_captchaTextVisibility, value);

}

#endregion

#region Видимость капчи

private Visibility \_captchaImageVisibility = Visibility.Hidden;

/// <summary>

/// Видимость капчи

/// </summary>

public Visibility CaptchaImageVisibility

{

get => \_captchaImageVisibility;

set => Set(ref \_captchaImageVisibility, value);

}

#endregion

#region Капча

private System.Windows.Media.ImageSource \_captchaImage = null;

/// <summary>

/// Капча

/// </summary>

public System.Windows.Media.ImageSource CaptchaImage

{

get => \_captchaImage;

set => Set(ref \_captchaImage, value);

}

#endregion

#region Текст капчи

private string \_captchaText = string.Empty;

/// <summary>

/// Текст капчи

/// </summary>

public string CaptchaText

{

get => \_captchaText;

set => Set(ref \_captchaText, value);

}

#endregion

#region Текст сгенерированной капчи

private string \_generatedCaptchaText = string.Empty;

/// <summary>

/// Текст сгенерированной капчи

/// </summary>

public string GeneratedCaptchaText

{

get => \_generatedCaptchaText;

set => Set(ref \_generatedCaptchaText, value);

}

#endregion

#region Счетчик неправильных попыток

private int \_count = 0;

/// <summary>

/// Счетчик неправильных попыток

/// </summary>

public int Count

{

get => \_count;

set => Set(ref \_count, value);

}

#endregion

private bool EntryClick(string Password)

{

bool result = false;

List<Employees> Employees\_ = (from em in Manager.Instance.Context.Employees

where em.login.ToLower() == LoginText.ToLower() &&

em.password == Password

select em).ToList();

if (Employees\_.Count == 0)

{

MessageBox.Show("Пользователь не найден.");

Count++;

}

else

{

result = true;

Debug.WriteLine("Пользователь найден.");

Employees employee = Employees\_[0];

Manager.Instance.LoadEmployeeInterface(employee);

}

if (Count == 3)

{

CaptchaImageVisibility = Visibility.Visible;

CaptchaTextBoxVisibility = Visibility.Visible;

UpdateCaptchaTextBlockVisibility = Visibility.Visible;

}

return result;

}

private void UpdateImage(int Width = 250, int Height = 60)

{

CaptchaImage = BitmapToImageSource(CreateImage(Width, Height));

}

public System.Windows.Media.ImageSource BitmapToImageSource(Bitmap bmp)

{

try

{

IntPtr handle;

handle = bmp.GetHbitmap();

return Imaging.CreateBitmapSourceFromHBitmap(handle, IntPtr.Zero, Int32Rect.Empty, System.Windows.Media.Imaging.BitmapSizeOptions.FromEmptyOptions());

}

catch

{

return null;

}

}

private Bitmap CreateImage(int Width, int Height)

{

Random rnd = new Random();

//Создадим изображение

Bitmap result = new Bitmap(Width, Height);

//Вычислим позицию текста

int Xpos = 10;

int Ypos = 10;

//Добавим различные цвета ддя текста

Brush[] colors = {

Brushes.Black,

Brushes.Red,

Brushes.RoyalBlue,

Brushes.Green,

Brushes.Yellow,

Brushes.White,

Brushes.Tomato,

Brushes.Sienna,

Brushes.Pink };

//Добавим различные цвета линий

Pen[] colorpens = {

Pens.Black,

Pens.Red,

Pens.RoyalBlue,

Pens.Green,

Pens.Yellow,

Pens.White,

Pens.Tomato,

Pens.Sienna,

Pens.Pink };

//Делаем случайный стиль текста

System.Drawing.FontStyle[] fontstyle = {

System.Drawing.FontStyle.Bold,

System.Drawing.FontStyle.Italic,

System.Drawing.FontStyle.Regular,

System.Drawing.FontStyle.Strikeout,

System.Drawing.FontStyle.Underline};

//Добавим различные углы поворота текста

short[] rotate = { 1, -1, 2, -2, 3, -3, 4, -4, 5, -5, 6, -6 };

//Укажем где рисовать

Graphics g = Graphics.FromImage(result);

//Пусть фон картинки будет серым

g.Clear(Color.Gray);

//Делаем случайный угол поворота текста

g.RotateTransform(rnd.Next(rotate.Length));

//Генерируем текст

GeneratedCaptchaText = string.Empty;

string symbols = string.Empty;

string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

symbols += alphabet;

symbols += alphabet.ToLower();

symbols += "0123456789";

for (int i = 0; i < 6; ++i)

{

GeneratedCaptchaText += symbols[rnd.Next(symbols.Length)];

}

//Нарисуем сгенирируемый текст

g.DrawString(GeneratedCaptchaText,

new Font("Arial", 25, fontstyle[rnd.Next(fontstyle.Length)]),

colors[rnd.Next(colors.Length)],

new PointF(Xpos, Ypos));

//Добавим немного помех

//Линии из углов

g.DrawLine(colorpens[rnd.Next(colorpens.Length)],

new System.Drawing.Point(0, 0),

new System.Drawing.Point(Width - 1, Height - 1));

g.DrawLine(colorpens[rnd.Next(colorpens.Length)],

new System.Drawing.Point(0, Height - 1),

new System.Drawing.Point(Width - 1, 0));

//Белые точки

for (int i = 0; i < Width; ++i)

{

for (int j = 0; j < Height; ++j)

{

if (rnd.Next() % 20 == 0)

{

result.SetPixel(i, j, Color.White);

}

}

}

return result;

}

#region Команды

public ICommand LoadEmployeesImagesCommand { get; }

private bool CanLoadEmployeesImagesCommandExecute(object parameters) => true;

private void OnLoadEmployeesImagesCommandExecuted(object parameters)

{

try

{

string pathToDirWithImages = $@"C:\Users\ARTEM\Desktop\КОРОНОВИРУС\21-22\Практика\Ресурсы\Image Сотрудники";

DirectoryInfo directoryInfo = new DirectoryInfo(pathToDirWithImages);

foreach (FileInfo fileInfo in directoryInfo.GetFiles())

{

if (fileInfo.Extension == ".jpg")

{

string fileName = fileInfo.FullName;

string[] fullName = fileName.Substring(fileName.LastIndexOf('\\') + 1).Split('.')[0].Split(' ');

string surname = fullName[0];

string name = fullName[1];

string patronymic = fullName[2];

Employees employee = (from em in Manager.Instance.Context.Employees

where em.name == name && em.surname == surname && em.patronymic == patronymic

select em).FirstOrDefault();

if (employee != null)

{

Debug.WriteLine(fileInfo);

employee.photo = Tools.GetImageBytes(fileInfo.FullName);

}

}

}

Manager.Instance.Context.SaveChanges();

Debug.WriteLine("Фото сотрудников загружены.");

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

public ICommand LoadMallImagesCommand { get; }

private bool CanLoadMallImagesCommandExecute(object parameters) => true;

private void OnLoadMallImagesCommandExecuted(object parameters)

{

try

{

string pathToDirWithImages = $@"C:\Users\ARTEM\Desktop\КОРОНОВИРУС\21-22\Практика\Ресурсы\Image ТЦ";

DirectoryInfo directoryInfo = new DirectoryInfo(pathToDirWithImages);

foreach (FileInfo fileInfo in directoryInfo.GetFiles())

{

if (fileInfo.Extension == ".jpg")

{

string fileName = fileInfo.FullName;

string mallName = fileName.Substring(fileName.LastIndexOf('\\') + 1).Split('.')[0];

Mall mall = (from m in Manager.Instance.Context.Mall

where m.mall\_name == mallName

select m).FirstOrDefault();

if (mall != null)

{

Debug.WriteLine(fileInfo);

mall.photo = Tools.GetImageBytes(fileInfo.FullName);

}

}

}

Manager.Instance.Context.SaveChanges();

Debug.WriteLine("Фото ТЦ загружены.");

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

public ICommand EntryCommand { get; }

private bool CanEntryCommandExecute(object p) => true;

private void OnEntryCommandExecuted(object p)

{

PasswordBox passwordBox = p as PasswordBox;

string password = passwordBox.Password;

if (Count != 3)

{

EntryClick(password);

}

else if (CaptchaText == GeneratedCaptchaText)

{

if (EntryClick(password))

{

CaptchaImageVisibility = Visibility.Hidden;

CaptchaTextBoxVisibility = Visibility.Hidden;

UpdateCaptchaTextBlockVisibility = Visibility.Hidden;

Count = 0;

CaptchaText = string.Empty;

}

}

else

{

MessageBox.Show("Неверная капча.");

}

UpdateImage();

}

public ICommand UpdateCaptchaLinkCommand { get; }

private bool CanUpdateCaptchaLinkCommandExecute(object p) => true;

private void OnUpdateCaptchaLinkCommandExecuted(object p)

{

UpdateImage();

}

#endregion

#region Конструктор

public LoginPageViewModel()

{

LoadEmployeesImagesCommand = new LambdaCommand(OnLoadEmployeesImagesCommandExecuted, CanLoadEmployeesImagesCommandExecute);

LoadMallImagesCommand = new LambdaCommand(OnLoadMallImagesCommandExecuted, CanLoadMallImagesCommandExecute);

EntryCommand = new LambdaCommand(OnEntryCommandExecuted, CanEntryCommandExecute);

UpdateCaptchaLinkCommand = new LambdaCommand(OnUpdateCaptchaLinkCommandExecuted, CanUpdateCaptchaLinkCommandExecute);

UpdateImage();

}

#endregion

}

}

**LoginPage.xaml :**

<Page x:Class="Project.LoginPage"

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

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

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

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

xmlns:local="clr-namespace:Project"

xmlns:vm="clr-namespace:Project.ViewModels"

mc:Ignorable="d"

d:DesignHeight="450"

d:DesignWidth="800"

Title="LoginPage">

<Page.DataContext>

<vm:LoginPageViewModel/>

</Page.DataContext>

<Grid Background="White">

<TextBlock Style="{StaticResource AuthorTemplate}"/>

<Button Content="{Binding EmployeesImagesButtonText}"

HorizontalAlignment="Left"

VerticalAlignment="Top"

Visibility="{Binding LoadEmployeesImagesVisibility}"

Style="{StaticResource ButtonTemplate}"

Command="{Binding LoadEmployeesImagesCommand}"/>

<Button Content="{Binding LoadMallImagesButtonText}"

HorizontalAlignment="Left"

VerticalAlignment="Top"

Margin="10,50,0,0"

Style="{StaticResource ButtonTemplate}"

Visibility="{Binding LoadMallImagesVisibility}"

Command="{Binding LoadMallImagesCommand}"/>

<StackPanel Orientation="Vertical"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Width="300">

<TextBlock Text="{Binding LoginLabelName}"

Style="{StaticResource TextBlockTemplate}"

Margin="10"

HorizontalAlignment="Center"/>

<TextBox Text="{Binding LoginText}"

Margin="10"

Style="{StaticResource TextBoxTemplate}"/>

<TextBlock Text="{Binding PasswordLabelName}"

Margin="10"

Style="{StaticResource TextBlockTemplate}"

HorizontalAlignment="Center"/>

<PasswordBox x:Name="Password"

Password="7SP9CV"

Margin="10"/>

<Button Content="{Binding EntryButtonText}"

Style="{StaticResource ButtonTemplate}"

Margin="10"

Command="{Binding EntryCommand}"

CommandParameter="{Binding ElementName=Password}"/>

<Image Source="{Binding CaptchaImage}"

Height="60"

Width="250"

Margin="10"

Visibility="{Binding CaptchaImageVisibility}"/>

<TextBlock Text="{Binding UpdateCaptchaText}"

HorizontalAlignment="Center"

TextDecorations="Underline"

Style="{StaticResource TextBlockTemplate}"

Visibility="{Binding UpdateCaptchaTextBlockVisibility}">

<TextBlock.InputBindings>

<MouseBinding Command="{Binding UpdateCaptchaLinkCommand}"

MouseAction="LeftClick"/>

</TextBlock.InputBindings>

</TextBlock>

<TextBox Text="{Binding CaptchaText}"

Margin="10"

Visibility="{Binding CaptchaTextBoxVisibility}"

Style="{StaticResource TextBoxTemplate}"/>

</StackPanel>

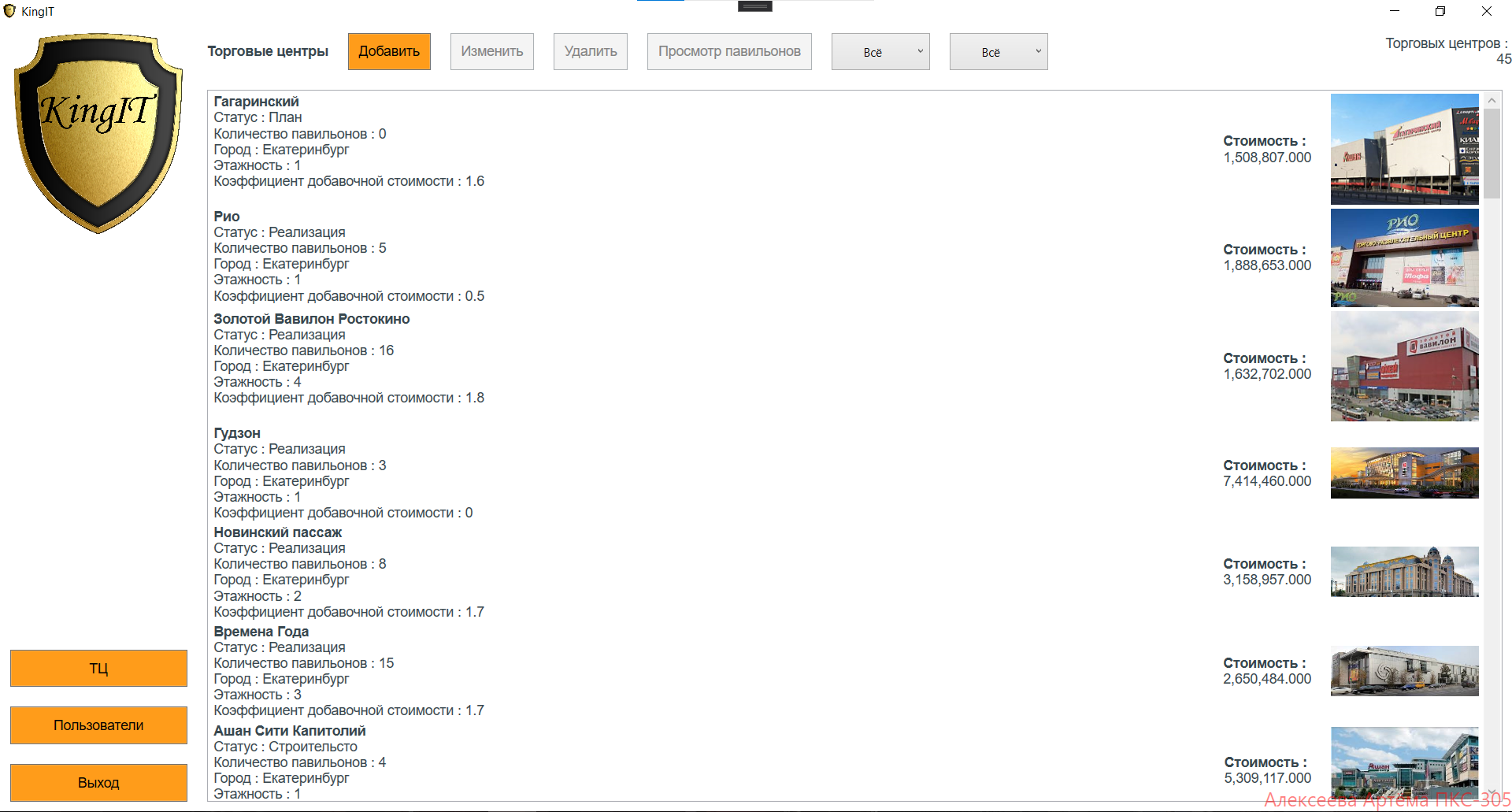
</Grid>

</Page>

1.4 **ИНТЕРФЕЙС «МЕНЕДЖЕРА C»**

**СПИСОК ТЦ**

Скриншот



Код приложения на XAML, C#

**ViewingMallPage.xaml :**

<Page x:Class="Project.Views.Pages.ViewingMallPage"

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

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

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

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

mc:Ignorable="d"

d:DesignHeight="450"

d:DesignWidth="800"

Title="ViewingMallPage">

<Grid Background="White"

DataContext="{StaticResource ViewingMallPageViewModel}">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<TextBlock Style="{StaticResource AuthorTemplate}"

Grid.Row="2"

Panel.ZIndex="1"/>

<DockPanel Grid.Row="0">

<TextBlock Text="{Binding MallTitleName}"

Style="{StaticResource TextBlockTemplate}"

FontWeight="Bold"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Margin="10"/>

<Button Content="Добавить"

Style="{StaticResource ButtonTemplate}"

Command="{Binding AddMallCommand}"/>

<Button Content="Изменить"

Style="{StaticResource ButtonTemplate}"

Command="{Binding ChangeMallCommand}"/>

<Button Content="Удалить"

Style="{StaticResource ButtonTemplate}"

Command="{Binding DeleteMallCommand}"/>

<Button Content="Просмотр павильонов"

Style="{StaticResource ButtonTemplate}"

Command="{Binding ViewMallCommand}"/>

<ComboBox ItemsSource="{Binding MallStatusesSorting}"

SelectedItem="{Binding SelectedMallStatusSorting}"

Margin="10"

Width="100"

HorizontalContentAlignment="Center"

VerticalContentAlignment="Center"/>

<ComboBox ItemsSource="{Binding Cities}"

SelectedItem="{Binding SelectedCity}"

Margin="10"

Width="100"

HorizontalContentAlignment="Center"

VerticalContentAlignment="Center"/>

<StackPanel VerticalAlignment="Center"

HorizontalAlignment="Right">

<TextBlock Text="Торговых центров : "

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=Malls.Count}"

Style="{StaticResource TextBlockTemplate}"

TextAlignment="Right"/>

</StackPanel>

</DockPanel>

<ListView Margin="10,10,10,10"

ItemsSource="{Binding Malls}"

SelectedItem="{Binding SelectedMall}"

SelectionMode="Single"

HorizontalContentAlignment="Stretch"

Grid.Row="1">

<ListView.ItemTemplate>

<DataTemplate>

<Grid HorizontalAlignment="Stretch">

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Path=mall\_name}"

Style="{StaticResource TextBlockTemplate}"

FontWeight="Bold"/>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Статус : "

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=status\_name}"

Style="{StaticResource TextBlockTemplate}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Количество павильонов : "

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=number\_of\_pavilion}"

Style="{StaticResource TextBlockTemplate}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Город : "

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=city}"

Style="{StaticResource TextBlockTemplate}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Этажность : "

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=number\_of\_storeys}"

Style="{StaticResource TextBlockTemplate}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Коэффициент добавочной стоимости : "

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=value\_added\_factor}"

Style="{StaticResource TextBlockTemplate}"/>

</StackPanel>

</StackPanel>

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Right"

VerticalAlignment="Center">

<StackPanel VerticalAlignment="Center"

Margin="0,0,20,0">

<TextBlock Text="Стоимость :"

FontWeight="Bold"

Style="{StaticResource TextBlockTemplate}"/>

<TextBlock Text="{Binding Path=cost,StringFormat=N3}"

Style="{StaticResource TextBlockTemplate}"

TextAlignment="Right"/>

</StackPanel>

<Image Source="{Binding Path=photo}"

MaxHeight="150"

MaxWidth="150"/>

</StackPanel>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

</Page>

**ViewingMallPageViewModel.cs :**

using Microsoft.Win32;

using Project.Infrastructure.Commands;

using Project.Models;

using Project.ViewModels.Base;

using Project.Views.Pages;

using System;

using System.Collections.ObjectModel;

using System.Data.SqlClient;

using System.Linq;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media.Imaging;

namespace Project.ViewModels

{

internal class ViewingMallPageViewModel : ViewModel

{

#region ТЦ

private ObservableCollection<MallItem> \_malls;

/// <summary>

/// ТЦ

/// </summary>

public ObservableCollection<MallItem> Malls

{

get => \_malls;

set => Set(ref \_malls, value);

}

#endregion

#region Выбранный ТЦ

private MallItem \_selectedMall;

/// <summary>

/// Выбранный ТЦ

/// </summary>

public MallItem SelectedMall

{

get => \_selectedMall;

set

{

IsSelectedMall = value != null;

Set(ref \_selectedMall, value);

}

}

#endregion

#region ТЦ выбран

private bool \_isSelectedMall = false;

/// <summary>

/// ТЦ выбран

/// </summary>

public bool IsSelectedMall

{

get => \_isSelectedMall;

set => Set(ref \_isSelectedMall, value);

}

#endregion

#region ТЦ название

private string \_mallTitleName = "Торговые центры";

/// <summary>

/// ТЦ

/// </summary>

public string MallTitleName

{

get => \_mallTitleName;

set => Set(ref \_mallTitleName, value);

}

#endregion

#region Добавить ТЦ

public ICommand AddMallCommand { get; }

private bool CanAddMallCommandExecute(object parameters) => true;

private void OnAddMallCommandExecuted(object parameters)

{

CurrentMallActionEntities = ActionEntities.Add;

Manager.Instance.MainFrameNavigate(new MallPage());

}

#endregion

#region Изменить ТЦ

public ICommand ChangeMallCommand { get; }

private bool CanChangeMallCommandExecute(object parameters) => IsSelectedMall;

private void OnChangeMallCommandExecuted(object parameters)

{

CurrentMallActionEntities = ActionEntities.Change;

Manager.Instance.MainFrameNavigate(new MallPage());

}

#endregion

#region Удалить ТЦ

public ICommand DeleteMallCommand { get; }

private bool CanDeleteMallCommandExecute(object parameters) => IsSelectedMall;

private void OnDeleteMallCommandExecuted(object parameters)

{

try

{

Mall mall = (

from m in Manager.Instance.Context.Mall

where m.mall\_id == SelectedMall.mall\_id

select m

).FirstOrDefault();

mall.status\_id = (

from m in Manager.Instance.Context.Mall\_statuses

where m.status\_name == DeleteNameSorting

select m.status\_id

).FirstOrDefault();

Manager.Instance.Context.SaveChanges();

Malls.Remove(SelectedMall); // или UpdateMalls();

MessageBox.Show($"Торговый центр удалён.");

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

#endregion

#region Просмотр ТЦ

public ICommand ViewMallCommand { get; }

private bool CanViewMallCommandExecute(object parameters) => IsSelectedMall;

private void OnViewMallCommandExecuted(object parameters)

{

UpdatePavilions();

Manager.Instance.MainFrameNavigate(new ViewingPavilionPage());

}

#endregion

#region Статусы

private ObservableCollection<string> \_mallStatuses;

/// <summary>

/// Статусы

/// </summary>

public ObservableCollection<string> MallStatuses

{

get => \_mallStatuses;

set => Set(ref \_mallStatuses, value);

}

#endregion

#region Текущий ТЦ

private Mall \_currentMall = new Mall();

/// <summary>

/// ТЦ

/// </summary>

public Mall CurrentMall

{

get => \_currentMall;

set => Set(ref \_currentMall, value);

}

#endregion

#region ActionEntities

private ActionEntities \_currentMallActionEntities = ActionEntities.None;

/// <summary>

/// ActionEntities

/// </summary>

public ActionEntities CurrentMallActionEntities

{

get => \_currentMallActionEntities;

set

{

Set(ref \_currentMallActionEntities, value);

UpdateMallStatuses();

if (CurrentMallActionEntities == ActionEntities.Add)

{

MallButtonName = "Добавить";

LoadedImage = null;

CurrentMall = new Mall();

}

else if (CurrentMallActionEntities == ActionEntities.Change)

{

MallButtonName = "Изменить";

CurrentMall = (

from m in Manager.Instance.Context.Mall

where m.mall\_id == SelectedMall.mall\_id

select m

).FirstOrDefault();

SelectedMallStatus = SelectedMall.status\_name;

if (CurrentMall.photo != null)

{

LoadedImage = Tools.BytesToImage(CurrentMall.photo);

}

}

}

}

#endregion

#region Img

private BitmapImage \_loadedImage;

/// <summary>

/// Img

/// </summary>

public BitmapImage LoadedImage

{

get => \_loadedImage;

set => Set(ref \_loadedImage, value);

}

#endregion

#region MallButtonName

private string \_mallButtonName;

/// <summary>

/// MallButtonName

/// </summary>

public string MallButtonName

{

get => \_mallButtonName;

set => Set(ref \_mallButtonName, value);

}

#endregion

#region Команда

public ICommand MallExecuteCommand { get; }

private bool CanMallExecuteCommandExecute(object parameters) => true;

private void OnMallExecuteCommandExecuted(object parameters)

{

if (CurrentMall.cost < 0 ||

CurrentMall.number\_of\_pavilion < 0 ||

CurrentMall.value\_added\_factor < 0.1 ||

CurrentMall.number\_of\_storeys < 0)

{

MessageBox.Show($"Числовые поля должны быть положительными.");

}

else if (string.IsNullOrEmpty(CurrentMall.mall\_name.Trim()) ||

SelectedMallStatus == null ||

string.IsNullOrEmpty(CurrentMall.city.Trim()) ||

CurrentMall.photo == null)

{

MessageBox.Show($"Заполните все поля.");

}

else

{

try

{

CurrentMall.status\_id = (

from ms in Manager.Instance.Context.Mall\_statuses

where ms.status\_name == SelectedMallStatus

select ms.status\_id

).FirstOrDefault();

CurrentMall.mall\_name = CurrentMall.mall\_name.Trim();

CurrentMall.city = CurrentMall.city.Trim();

switch (CurrentMallActionEntities)

{

case ActionEntities.Add:

Manager.Instance.Context.Mall.Add(CurrentMall);

MessageBox.Show($"Торговый центр добавлен.");

break;

case ActionEntities.Change:

MessageBox.Show($"Торговый центр изменён.");

break;

}

Manager.Instance.Context.SaveChanges();

Manager.Instance.MainFrameNavigate(new ViewingMallPage());

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

SelectedMallStatusSorting = AllNameSorting;

SelectedCity = AllNameSorting;

UpdateMalls();

UpdateCities();

}

#endregion

#region Загрузка фотографии

public ICommand LoadPhotoCommand { get; }

private bool CanLoadPhotoCommandExecute(object parameters) => true;

private void OnLoadPhotoCommandExecuted(object parameters)

{

try

{

OpenFileDialog fileDialog = new OpenFileDialog

{

Filter = "Image Files|\*.jpg;\*png;"

};

if ((bool)fileDialog.ShowDialog())

{

if (fileDialog.FileName.EndsWith(".jpg") ||

fileDialog.FileName.EndsWith(".png"))

{

CurrentMall.photo = Tools.GetImageBytes(fileDialog.FileName);

LoadedImage = new BitmapImage(new Uri(fileDialog.FileName));

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

#endregion

#region Сортировка статусов торговых центров

private ObservableCollection<string> \_mallStatusesSorting;

/// <summary>

/// Сортировка статусов торговых центров

/// </summary>

public ObservableCollection<string> MallStatusesSorting

{

get => \_mallStatusesSorting;

set => Set(ref \_mallStatusesSorting, value);

}

#endregion

#region Выбранная сортировка статусов

private string \_selectedMallStatusSorting;

/// <summary>

/// Выбранная сортировка статусов

/// </summary>

public string SelectedMallStatusSorting

{

get => \_selectedMallStatusSorting;

set

{

Set(ref \_selectedMallStatusSorting, value);

UpdateMalls();

}

}

#endregion

#region Города

private ObservableCollection<string> \_cities;

/// <summary>

/// Города

/// </summary>

public ObservableCollection<string> Cities

{

get => \_cities;

set => Set(ref \_cities, value);

}

#endregion

#region Выбранный город

private string \_selectedCity;

/// <summary>

/// Выбранный город

/// </summary>

public string SelectedCity

{

get => \_selectedCity;

set

{

Set(ref \_selectedCity, value);

UpdateMalls();

}

}

#endregion

#region Статус

private string \_selectedMallStatus;

/// <summary>

/// Статус

/// </summary>

public string SelectedMallStatus

{

get => \_selectedMallStatus;

set => Set(ref \_selectedMallStatus, value);

}

#endregion

#region GetMalls

private ObservableCollection<MallItem> GetMalls()

{

ObservableCollection<MallItem> data = new ObservableCollection<MallItem>();

if (SelectedMallStatusSorting == AllNameSorting)

{

if (SelectedCity == AllNameSorting)

{

data = new ObservableCollection<MallItem>(

from mall in Manager.Instance.Context.Mall

join ms in Manager.Instance.Context.Mall\_statuses on mall.status\_id equals ms.status\_id

orderby mall.city, ms.status\_name

where ms.status\_name != DeleteNameSorting

select new MallItem

{

mall\_id = mall.mall\_id,

mall\_name = mall.mall\_name,

status\_name = ms.status\_name,

status\_id = ms.status\_id,

number\_of\_pavilion = mall.number\_of\_pavilion,

city = mall.city,

cost = mall.cost,

number\_of\_storeys = mall.number\_of\_storeys,

value\_added\_factor = mall.value\_added\_factor,

photo = mall.photo

});

}

else

{

data = new ObservableCollection<MallItem>(

from mall in Manager.Instance.Context.Mall

join ms in Manager.Instance.Context.Mall\_statuses on mall.status\_id equals ms.status\_id

orderby mall.city, ms.status\_name

where ms.status\_name != DeleteNameSorting &&

mall.city == SelectedCity

select new MallItem

{

mall\_id = mall.mall\_id,

mall\_name = mall.mall\_name,

status\_name = ms.status\_name,

status\_id = ms.status\_id,

number\_of\_pavilion = mall.number\_of\_pavilion,

city = mall.city,

cost = mall.cost,

number\_of\_storeys = mall.number\_of\_storeys,

value\_added\_factor = mall.value\_added\_factor,

photo = mall.photo

});

}

}

else

{

// SelectedMallStatusSorting != AllNameSorting

if (SelectedCity == AllNameSorting)

{

data = new ObservableCollection<MallItem>(

from mall in Manager.Instance.Context.Mall

join ms in Manager.Instance.Context.Mall\_statuses on mall.status\_id equals ms.status\_id

orderby mall.city, ms.status\_name

where ms.status\_name == SelectedMallStatusSorting

select new MallItem

{

mall\_id = mall.mall\_id,

mall\_name = mall.mall\_name,

status\_name = ms.status\_name,

status\_id = ms.status\_id,

number\_of\_pavilion = mall.number\_of\_pavilion,

city = mall.city,

cost = mall.cost,

number\_of\_storeys = mall.number\_of\_storeys,

value\_added\_factor = mall.value\_added\_factor,

photo = mall.photo

}

);

}

else

{

data = new ObservableCollection<MallItem>(

from mall in Manager.Instance.Context.Mall

join ms in Manager.Instance.Context.Mall\_statuses on mall.status\_id equals ms.status\_id

orderby mall.city, ms.status\_name

where ms.status\_name == SelectedMallStatusSorting &&

mall.city == SelectedCity

select new MallItem

{

mall\_id = mall.mall\_id,

mall\_name = mall.mall\_name,

status\_name = ms.status\_name,

status\_id = ms.status\_id,

number\_of\_pavilion = mall.number\_of\_pavilion,

city = mall.city,

cost = mall.cost,

number\_of\_storeys = mall.number\_of\_storeys,

value\_added\_factor = mall.value\_added\_factor,

photo = mall.photo

}

);

}

}

return data;

}

#endregion

#region UpdateMalls

private void UpdateMalls()

{

Malls = GetMalls();

}

#endregion

#region GetMallStatuses

private ObservableCollection<string> GetMallStatuses()

{

return new ObservableCollection<string>(

from ms in Manager.Instance.Context.Mall\_statuses

orderby ms.status\_name

select ms.status\_name

);

}

#endregion

#region UpdateMallStatuses

private void UpdateMallStatuses()

{

MallStatuses = GetMallStatuses();

}

#endregion

#region GetCities

private void UpdateCities()

{

Cities = GetCities();

}

#endregion

#region GetCities

private ObservableCollection<string> GetCities()

{

return new ObservableCollection<string>(

(

from m in Manager.Instance.Context.Mall

orderby m.city

where m.status\_id != (from ms in Manager.Instance.Context.Mall\_statuses

where ms.status\_name == DeleteNameSorting

select ms.status\_id).FirstOrDefault()

select m.city

).Distinct().ToList()

)

{

AllNameSorting

};

}

#endregion

#region Статус сортировки Всё

private string \_allNameSorting = "Всё";

/// <summary>

/// Статус сортировки Всё

/// </summary>

public string AllNameSorting

{

get => \_allNameSorting;

set => Set(ref \_allNameSorting, value);

}

#endregion

#region Статус сортировки Удалён

private string \_deleteNameSorting = "Удалён";

/// <summary>

/// Статус сортировки Удалён

/// </summary>

public string DeleteNameSorting

{

get => \_deleteNameSorting;

set => Set(ref \_deleteNameSorting, value);

}

#endregion

#region ViewingPavilionPage

#region Павильоны название

private string \_pavilionTitleName = "Павильоны";

/// <summary>

/// Павильоны название

/// </summary>

public string PavilionTitleName

{

get => \_pavilionTitleName;

set => Set(ref \_pavilionTitleName, value);

}

#endregion

#region Павильоны

private ObservableCollection<PavilionItem> \_pavilions;

/// <summary>

/// Павильоны

/// </summary>

public ObservableCollection<PavilionItem> Pavilions

{

get => \_pavilions;

set => Set(ref \_pavilions, value);

}

#endregion

#region Выбранный Павильон

private PavilionItem \_selectedPavilion;

/// <summary>

/// Выбранный Павильон

/// </summary>

public PavilionItem SelectedPavilion

{

get => \_selectedPavilion;

set

{

IsSelectedPavilion = value != null;

Set(ref \_selectedPavilion, value);

}

}

#endregion

#region Павильон выбран

private bool \_isSelectedPavilion = false;

/// <summary>

/// Павильон выбран

/// </summary>

public bool IsSelectedPavilion

{

get => \_isSelectedPavilion;

set => Set(ref \_isSelectedPavilion, value);

}

#endregion

#region UpdatePavilions

private void UpdatePavilions()

{

Pavilions = GetPavilions();

}

#endregion

#region GetPavilions

private ObservableCollection<PavilionItem> GetPavilions()

{

ObservableCollection<PavilionItem> data = new ObservableCollection<PavilionItem>();

if (SelectedMall == null)

{

return null;

}

else if (SelectedPavilionStatusSorting == AllNameSorting)

{

data = new ObservableCollection<PavilionItem>(

from p in Manager.Instance.Context.Pavilion

join ps in Manager.Instance.Context.Pavilion\_statuses on p.status\_id equals ps.status\_id

join m in Manager.Instance.Context.Mall on p.mall\_id equals m.mall\_id

join ms in Manager.Instance.Context.Mall\_statuses on m.status\_id equals ms.status\_id

where ps.status\_name != DeleteNameSorting &&

p.floor == Floor &&

p.square >= MinSquare &&

p.square <= MaxSquare &&

m.mall\_name == SelectedMall.mall\_name

select new PavilionItem

{

mall\_id = m.mall\_id,

mall\_status\_id = m.status\_id,

mall\_status\_name = ms.status\_name,

mall\_name = m.mall\_name,

floor = p.floor,

pavilion\_number = p.pavilion\_number,

square = p.square,

pavilion\_status\_id = p.status\_id,

pavilion\_status\_name = ps.status\_name,

cost\_per\_square\_meter = p.cost\_per\_square\_meter,

value\_added\_factor = p.value\_added\_factor

});

}

else

{

data = new ObservableCollection<PavilionItem>(

from p in Manager.Instance.Context.Pavilion

join ps in Manager.Instance.Context.Pavilion\_statuses on p.status\_id equals ps.status\_id

join m in Manager.Instance.Context.Mall on p.mall\_id equals m.mall\_id

join ms in Manager.Instance.Context.Mall\_statuses on m.status\_id equals ms.status\_id

where ps.status\_name != DeleteNameSorting &&

ps.status\_name == SelectedPavilionStatusSorting &&

p.floor == Floor &&

p.square >= MinSquare &&

p.square <= MaxSquare &&

m.mall\_name == SelectedMall.mall\_name

select new PavilionItem

{

mall\_id = m.mall\_id,

mall\_status\_id = m.status\_id,

mall\_status\_name = ms.status\_name,

mall\_name = m.mall\_name,

floor = p.floor,

pavilion\_number = p.pavilion\_number,

square = p.square,

pavilion\_status\_id = p.status\_id,

pavilion\_status\_name = ps.status\_name,

cost\_per\_square\_meter = p.cost\_per\_square\_meter,

value\_added\_factor = p.value\_added\_factor

});

}

return data;

}

#endregion

#region ButtonName

private string \_ButtonName;

/// <summary>

/// ButtonName

/// </summary>

public string ButtonName

{

get => \_ButtonName;

set => Set(ref \_ButtonName, value);

}

#endregion

#region GetNewPavilion

private Pavilion GetNewPavilion()

{

return new Pavilion()

{

pavilion\_number = ""

};

}

#endregion

#region Добавить павильон

public ICommand AddPavilionCommand { get; }

private bool CanAddPavilionCommandExecute(object parameters) => true;

private void OnAddPavilionCommandExecuted(object parameters)

{

CurrentActionEntities = ActionEntities.Add;

Manager.Instance.MainFrameNavigate(new PavilionPage());

}

#endregion

#region Изменить павильон

public ICommand ChangePavilionCommand { get; }

private bool CanChangePavilionCommandExecute(object parameters) => IsSelectedPavilion;

private void OnChangePavilionCommandExecuted(object parameters)

{

CurrentActionEntities = ActionEntities.Change;

Manager.Instance.MainFrameNavigate(new PavilionPage());

}

#endregion

#region Обновить павильоны

public ICommand UpdatePavilionsCommand { get; }

private bool CanUpdatePavilionsCommandExecute(object parameters) => true;

private void OnUpdatePavilionsCommandExecuted(object parameters)

{

UpdatePavilions();

}

#endregion

#region Удалить павильон

public ICommand DeletePavilionCommand { get; }

private bool CanDeletePavilionCommandExecute(object parameters) => IsSelectedPavilion;

private void OnDeletePavilionCommandExecuted(object parameters)

{

try

{

Pavilion pavilion = (

from p in Manager.Instance.Context.Pavilion

where p.mall\_id == SelectedPavilion.mall\_id &&

p.pavilion\_number == SelectedPavilion.pavilion\_number

select p

).FirstOrDefault();

pavilion.status\_id = (

from m in Manager.Instance.Context.Pavilion\_statuses

where m.status\_name == DeleteNameSorting

select m.status\_id

).FirstOrDefault();

Manager.Instance.Context.SaveChanges();

Pavilions.Remove(SelectedPavilion); // или UpdatePavilions();

MessageBox.Show($"Павильон удалён.");

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

#endregion

#region Текущий павильон

private Pavilion \_currentPavilion;

/// <summary>

/// Текущий павильон

/// </summary>

public Pavilion CurrentPavilion

{

get => \_currentPavilion;

set => Set(ref \_currentPavilion, value);

}

#endregion

#region Статусы павильонов

private ObservableCollection<string> \_PavilionStatuses;

/// <summary>

/// Статусы павильонов

/// </summary>

public ObservableCollection<string> PavilionStatuses

{

get => \_PavilionStatuses;

set => Set(ref \_PavilionStatuses, value);

}

#endregion

#region Выбранный статус павильона

private string \_selectedPavilionStatus;

/// <summary>

/// Выбранный статус павильона

/// </summary>

public string SelectedPavilionStatus

{

get => \_selectedPavilionStatus;

set => Set(ref \_selectedPavilionStatus, value);

}

#endregion

#region UpdatePavilionStatuses

private void UpdatePavilionStatuses()

{

PavilionStatuses = GetPavilionStatuses();

}

#endregion

#region GetPavilionStatuses

private ObservableCollection<string> GetPavilionStatuses()

{

return new ObservableCollection<string>(

from p in Manager.Instance.Context.Pavilion\_statuses

select p.status\_name

);

}

#endregion

#region Названия павильонов

private ObservableCollection<string> \_pavilionNames;

/// <summary>

/// Названия павильонов

/// </summary>

public ObservableCollection<string> PavilionNames

{

get => \_pavilionNames;

set => Set(ref \_pavilionNames, value);

}

#endregion

#region Выбранное название павильона

private string \_selectedMallPavilionName;

/// <summary>

/// Выбранное название павильона

/// </summary>

public string SelectedMallPavilionName

{

get => \_selectedMallPavilionName;

set => Set(ref \_selectedMallPavilionName, value);

}

#endregion

#region UpdatePavilionNames

private void UpdatePavilionNames()

{

PavilionNames = GetPavilionNames();

}

#endregion

#region GetPavilionNames

private ObservableCollection<string> GetPavilionNames()

{

return new ObservableCollection<string>(

from m in Manager.Instance.Context.Mall

select m.mall\_name

);

}

#endregion

#region Сортировка статусов павильонов

private ObservableCollection<string> \_pavilionStatusesSorting;

/// <summary>

/// Сортировка статусов павильонов

/// </summary>

public ObservableCollection<string> PavilionStatusesSorting

{

get => \_pavilionStatusesSorting;

set => Set(ref \_pavilionStatusesSorting, value);

}

#endregion

#region Выбранная сортировка статуса

private string \_selectedPavilionStatusSorting;

/// <summary>

/// Выбранная сортировка статуса

/// </summary>

public string SelectedPavilionStatusSorting

{

get => \_selectedPavilionStatusSorting;

set

{

Set(ref \_selectedPavilionStatusSorting, value);

UpdatePavilions();

}

}

#endregion

#region GetPavilionStatusSorting

private ObservableCollection<string> GetPavilionStatusSorting()

{

return new ObservableCollection<string>(

from p in Manager.Instance.Context.Pavilion\_statuses

orderby p.status\_name

select p.status\_name

);

}

#endregion

#region UpdatePavilionStatusSorting

private void UpdatePavilionStatusSorting()

{

PavilionStatusesSorting = GetPavilionStatusSorting();

}

#endregion

#region ActionEntities

private ActionEntities \_currentActionEntities = ActionEntities.None;

/// <summary>

/// ActionEntities

/// </summary>

public ActionEntities CurrentActionEntities

{

get => \_currentActionEntities;

set

{

Set(ref \_currentActionEntities, value);

if (\_currentActionEntities == ActionEntities.Add)

{

CurrentPavilion = GetNewPavilion();

ButtonName = "Добавить";

}

else if (\_currentActionEntities == ActionEntities.Change)

{

ButtonName = "Изменить";

CurrentPavilion = (

from p in Manager.Instance.Context.Pavilion

where p.mall\_id == SelectedPavilion.mall\_id &&

p.pavilion\_number == SelectedPavilion.pavilion\_number

select p

).FirstOrDefault();

SelectedPavilionStatus = SelectedPavilion.pavilion\_status\_name;

SelectedMallPavilionName = SelectedPavilion.mall\_name;

}

}

}

#endregion

#region Команда

public ICommand ExecuteCommand { get; }

private bool CanExecuteCommandExecute(object parameters) => true;

private void OnExecuteCommandExecuted(object parameters)

{

if (CurrentPavilion.floor < 0 ||

CurrentPavilion.square < 0 ||

CurrentPavilion.value\_added\_factor < 0 ||

CurrentPavilion.cost\_per\_square\_meter < (decimal)0.1)

{

MessageBox.Show($"Числовые поля должны быть положительными.");

}

else if (CurrentPavilion.pavilion\_number != null &&

string.IsNullOrEmpty(CurrentPavilion.pavilion\_number.Trim()) ||

SelectedPavilionStatus == null)

{

MessageBox.Show($"Заполните все поля.");

}

else

{

try

{

CurrentPavilion.status\_id = (

from ps in Manager.Instance.Context.Pavilion\_statuses

where ps.status\_name == SelectedPavilionStatus

select ps.status\_id

).FirstOrDefault();

CurrentPavilion.mall\_id = (

from m in Manager.Instance.Context.Mall

where m.mall\_id == SelectedMall.mall\_id

select m.mall\_id

).FirstOrDefault();

CurrentPavilion.pavilion\_number = CurrentPavilion.pavilion\_number.Trim();

switch (CurrentActionEntities)

{

case ActionEntities.Add:

Manager.Instance.Context.Pavilion.Add(CurrentPavilion);

break;

case ActionEntities.Change:

break;

}

Manager.Instance.Context.SaveChanges();

switch (CurrentActionEntities)

{

case ActionEntities.Add:

MessageBox.Show($"Павильон добавлен.");

break;

case ActionEntities.Change:

MessageBox.Show($"Павильон изменён.");

break;

}

Manager.Instance.MainFrameNavigate(new ViewingPavilionPage());

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

SelectedPavilionStatusSorting = AllNameSorting;

UpdatePavilions();

}

#endregion

#region Этаж

private int \_floor = 2;

/// <summary>

/// Этаж

/// </summary>

public int Floor

{

get => \_floor;

set

{

if (value >= 0)

{

Set(ref \_floor, value);

UpdatePavilions();

}

}

}

#endregion

#region Мин площадь

private decimal \_minSquare = 0;

/// <summary>

/// Min площадь

/// </summary>

public decimal MinSquare

{

get => \_minSquare;

set

{

if (value >= 0)

{

if (value <= MaxSquare)

{

Set(ref \_minSquare, value);

}

else

{

Set(ref \_minSquare, MaxSquare);

}

UpdatePavilions();

}

}

}

#endregion

#region Мин площадь

private decimal \_maxSquare = 1000;

/// <summary>

/// Max площадь

/// </summary>

public decimal MaxSquare

{

get => \_maxSquare;

set

{

if (value >= 0)

{

if (value >= MinSquare)

{

Set(ref \_maxSquare, value);

}

else

{

Set(ref \_maxSquare, MinSquare);

}

UpdatePavilions();

}

}

}

#endregion

#endregion

#region PavilionRentalPage

#region Просмотр ТЦ

public ICommand RentPavilionCommand { get; }

private bool CanRentPavilionCommandExecute(object parameters) => IsSelectedPavilion;

private void OnRentPavilionCommandExecuted(object parameters)

{

Manager.Instance.MainFrameNavigate(new PavilionRentalPage());

}

#endregion

#region Действие аренда

private string \_rentActionName = "Аренда";

/// <summary>

/// Действие аренда

/// </summary>

public string RentActionName

{

get => \_rentActionName;

set => Set(ref \_rentActionName, value);

}

#endregion

#region Действие забронировать

private string \_bookActionName = "Бронь";

/// <summary>

/// Действие забронировать

/// </summary>

public string BookActionName

{

get => \_bookActionName;

set => Set(ref \_bookActionName, value);

}

#endregion

#region Действия аренда/бронь

private ObservableCollection<string> \_rentActions;

/// <summary>

/// Действия аренда/бронь

/// </summary>

public ObservableCollection<string> RentActions

{

get => \_rentActions;

set => Set(ref \_rentActions, value);

}

#endregion

#region Выбранное действие аренда/бронь

private string \_selectedRentAction;

/// <summary>

/// Выбранное действие аренда/бронь

/// </summary>

public string SelectedRentAction

{

get => \_selectedRentAction;

set

{

Set(ref \_selectedRentAction, value);

if (\_selectedRentAction == RentActionName)

{

PavilionRentalButtonName = "Арендовать";

}

else if (\_selectedRentAction == BookActionName)

{

PavilionRentalButtonName = "Забронировать";

}

}

}

#endregion

#region GetRentActions

private ObservableCollection<string> GetRentActions()

{

return new ObservableCollection<string>()

{

RentActionName,

BookActionName

};

}

#endregion

#region UpdateRentActions

private void UpdateRentActions()

{

RentActions = GetRentActions();

}

#endregion

#region Начало аренды

private DateTime? \_rentalStartDate = DateTime.Today;

/// <summary>

/// Начало аренды

/// </summary>

public DateTime? RentalStartDate

{

get => \_rentalStartDate;

set

{

Set(ref \_rentalStartDate, value);

MinimumRentalEndDate = \_rentalStartDate;

}

}

#endregion

#region Минимальная выбираемая дата для начала аренды

private DateTime? \_minimumRentalStartDate = DateTime.Today;

/// <summary>

/// Минимальная выбираемая дата для начала аренды

/// </summary>

public DateTime? MinimumRentalStartDate

{

get => \_minimumRentalStartDate;

set => Set(ref \_minimumRentalStartDate, value);

}

#endregion

#region Максимальная выбираемая дата для начала аренды

private DateTime? \_maximumRentalStartDate;

/// <summary>

/// Максимальная выбираемая дата для начала аренды

/// </summary>

public DateTime? MaximumRentalStartDate

{

get => \_maximumRentalStartDate;

set => Set(ref \_maximumRentalStartDate, value);

}

#endregion

#region Конец аренды

private DateTime? \_rentalEndDate;

/// <summary>

/// Конец аренды

/// </summary>

public DateTime? RentalEndDate

{

get => \_rentalEndDate;

set

{

Set(ref \_rentalEndDate, value);

MaximumRentalStartDate = \_rentalEndDate;

}

}

#endregion

#region Минимальная выбираемая дата для конца аренды

private DateTime? \_minimumRentalEndDate = DateTime.Today;

/// <summary>

/// Минимальная выбираемая дата для конца аренды

/// </summary>

public DateTime? MinimumRentalEndDate

{

get => \_minimumRentalEndDate;

set => Set(ref \_minimumRentalEndDate, value);

}

#endregion

#region Максимальная выбираемая дата для начала аренды

private DateTime? \_maximumRentalEndDate;

/// <summary>

/// Максимальная выбираемая дата для конца аренды

/// </summary>

public DateTime? MaximumRentalEndDate

{

get => \_maximumRentalEndDate;

set => Set(ref \_maximumRentalEndDate, value);

}

#endregion

#region Арендаторы

private ObservableCollection<string> \_tenants;

/// <summary>

/// Арендаторы

/// </summary>

public ObservableCollection<string> Tenants

{

get => \_tenants;

set => Set(ref \_tenants, value);

}

#endregion

#region Выбранный арендатор

private string \_selectedTenant;

/// <summary>

/// Выбранный арендатор

/// </summary>

public string SelectedTenant

{

get => \_selectedTenant;

set => Set(ref \_selectedTenant, value);

}

#endregion

#region GetTenants

private ObservableCollection<string> GetTenants()

{

return new ObservableCollection<string>(

from t in Manager.Instance.Context.Tenants

select t.company\_name

);

}

#endregion

#region UpdateTenants

private void UpdateTenants()

{

Tenants = GetTenants();

}

#endregion

#region Сотрудники

private ObservableCollection<string> \_employees;

/// <summary>

/// Сотрудники

/// </summary>

public ObservableCollection<string> Employees

{

get => \_employees;

set => Set(ref \_employees, value);

}

#endregion

#region Выбранный сотрудник

private string \_selectedEmployee;

/// <summary>

/// Выбранный сотрудник

/// </summary>

public string SelectedEmployee

{

get => \_selectedEmployee;

set => Set(ref \_selectedEmployee, value);

}

#endregion

#region GetEmployees

private ObservableCollection<string> GetEmployees()

{

ObservableCollection<string> result = new ObservableCollection<string>();

foreach (Employees em in from t in Manager.Instance.Context.Employees

select t)

{

result.Add($"{em.employe\_id} {em.surname} {em.name} {em.patronymic}");

}

return result;

}

#endregion

#region UpdateEmployees

private void UpdateEmployees()

{

Employees = GetEmployees();

}

#endregion

#region PavilionRentalButtonName

private string \_pavilionRentalButtonName;

/// <summary>

/// PavilionRentalButtonName

/// </summary>

public string PavilionRentalButtonName

{

get => \_pavilionRentalButtonName;

set => Set(ref \_pavilionRentalButtonName, value);

}

#endregion

#region Удалить павильон

public ICommand PavilionRentalCommand { get; }

private bool CanPavilionRentalCommandExecute(object parameters) => true;

private void OnPavilionRentalCommandExecuted(object parameters)

{

if (SelectedRentAction != null &&

RentalStartDate != null &&

SelectedPavilion != null &&

RentalEndDate != null &&

SelectedTenant != null &&

SelectedEmployee != null)

{

try

{

int status\_action = (SelectedRentAction == RentActionName) ? 0 : 1;

string pavilion\_number = SelectedPavilion.pavilion\_number;

long mall\_id = SelectedPavilion.mall\_id;

long tenant\_id = (from t in Manager.Instance.Context.Tenants

where t.company\_name == SelectedTenant

select t.tenant\_id).FirstOrDefault();

long employee\_id = long.Parse(SelectedEmployee.Split(' ')[0]);

Manager.Instance.Context.Database.ExecuteSqlCommand

($@"exec RentOrBookPavilionInMall @status\_action,

@pavilion\_number,

@mall\_id,

@start\_date,

@end\_date,

@tenant\_id,

@employee\_id",

new SqlParameter("@status\_action", status\_action),

new SqlParameter("@pavilion\_number", pavilion\_number),

new SqlParameter("@mall\_id", mall\_id),

new SqlParameter("@start\_date", RentalStartDate),

new SqlParameter("@end\_date", RentalEndDate),

new SqlParameter("@tenant\_id", tenant\_id),

new SqlParameter("@employee\_id", employee\_id));

MessageBox.Show($"Павильон {((SelectedRentAction == RentActionName) ? "арендован" : "забронирован")}.");

Manager.Instance.MainFrameNavigate(new ViewingPavilionPage());

UpdatePavilions();

}

catch (Exception e)

{

MessageBox.Show($"Ошибка :\n{e}");

}

}

else

{

MessageBox.Show($"Заполните все поля.");

}

}

#endregion

#endregion

#region Конструктор

public ViewingMallPageViewModel()

{

DeleteMallCommand = new LambdaCommand(OnDeleteMallCommandExecuted, CanDeleteMallCommandExecute);

AddMallCommand = new LambdaCommand(OnAddMallCommandExecuted, CanAddMallCommandExecute);

ChangeMallCommand = new LambdaCommand(OnChangeMallCommandExecuted, CanChangeMallCommandExecute);

ViewMallCommand = new LambdaCommand(OnViewMallCommandExecuted, CanViewMallCommandExecute);

UpdateMalls();

MallExecuteCommand = new LambdaCommand(OnMallExecuteCommandExecuted, CanMallExecuteCommandExecute);

LoadPhotoCommand = new LambdaCommand(OnLoadPhotoCommandExecuted, CanLoadPhotoCommandExecute);

UpdateMallStatuses();

MallStatusesSorting = GetMallStatuses();

// убираю сортировку по удаленным ТЦ

MallStatusesSorting.Remove(DeleteNameSorting);

// добавляю общую сортировку

MallStatusesSorting.Add(AllNameSorting);

// текущим выбранным ставим все

SelectedMallStatusSorting = AllNameSorting;

UpdateCities();

//// текущим выбранным ставим все

SelectedCity = AllNameSorting;

#region ViewingPavilionPageViewModel

AddPavilionCommand = new LambdaCommand(OnAddPavilionCommandExecuted, CanAddPavilionCommandExecute);

ChangePavilionCommand = new LambdaCommand(OnChangePavilionCommandExecuted, CanChangePavilionCommandExecute);

DeletePavilionCommand = new LambdaCommand(OnDeletePavilionCommandExecuted, CanDeletePavilionCommandExecute);

ExecuteCommand = new LambdaCommand(OnExecuteCommandExecuted, CanExecuteCommandExecute);

UpdatePavilionsCommand = new LambdaCommand(OnUpdatePavilionsCommandExecuted, CanUpdatePavilionsCommandExecute);

UpdatePavilionStatuses();

UpdatePavilionStatusSorting();

// убираю сортировку по удаленным ТЦ

PavilionStatusesSorting.Remove(DeleteNameSorting);

// добавляю общую сортировку

PavilionStatusesSorting.Add(AllNameSorting);

// текущим выбранным ставим все

SelectedPavilionStatusSorting = AllNameSorting;

UpdatePavilions();

UpdatePavilionNames();

CurrentPavilion = GetNewPavilion();

#endregion

#region PavilionRentalPage

RentPavilionCommand = new LambdaCommand(OnRentPavilionCommandExecuted, CanRentPavilionCommandExecute);

PavilionRentalCommand = new LambdaCommand(OnPavilionRentalCommandExecuted, CanPavilionRentalCommandExecute);

UpdateRentActions();

UpdateTenants();

UpdateEmployees();

SelectedRentAction = RentActionName;

RentalEndDate = DateTime.Today.AddDays(1);

#endregion

}

#endregion

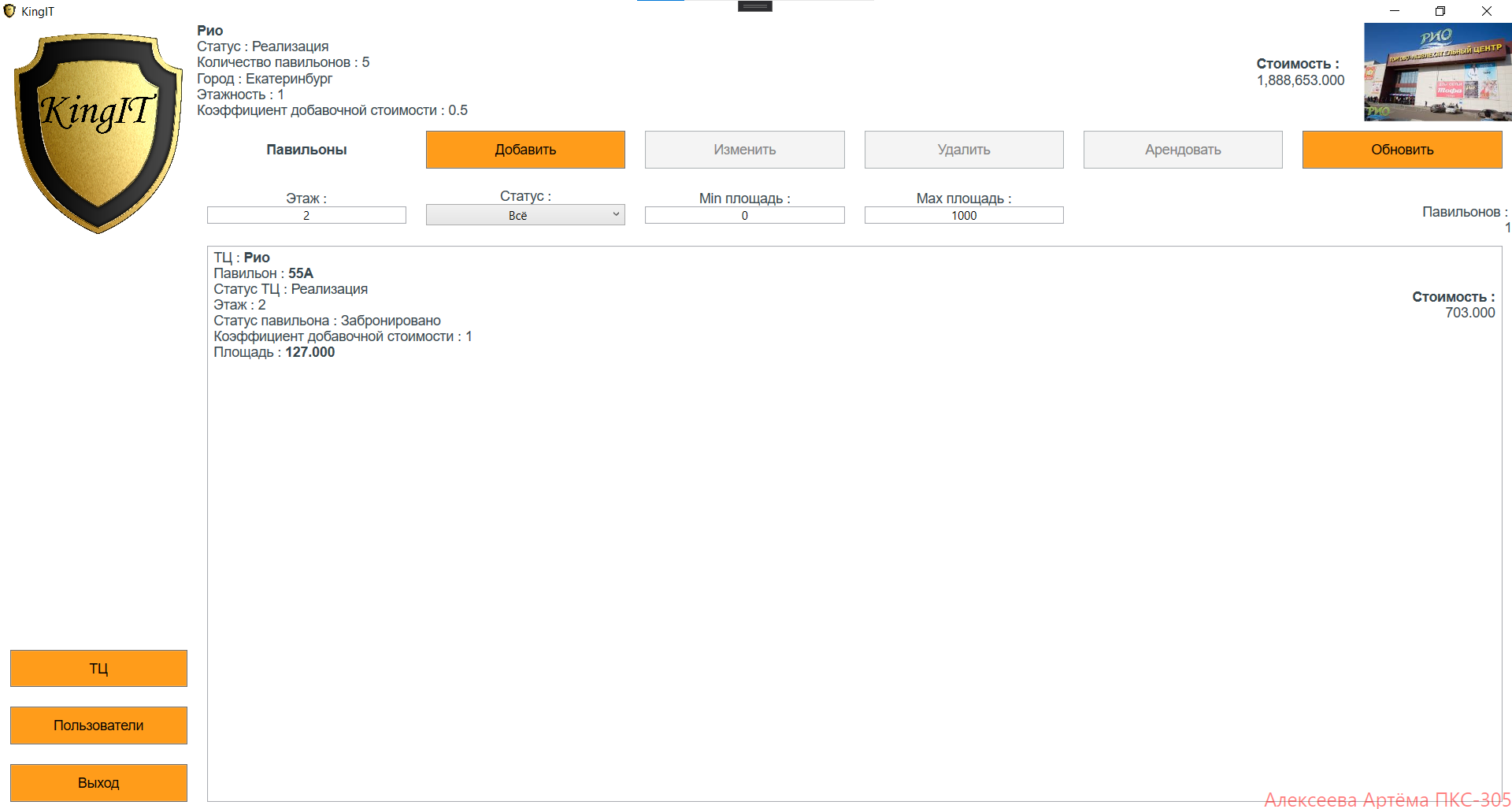
}

}

1.5 **ИНТЕРФЕЙС «МЕНЕДЖЕРА C»**

**СПИСОК ПАВИЛЬОНОВ**

Скриншот



Код приложения на XAML, C#

1.6 **ИНТЕРФЕЙС «МЕНЕДЖЕРА C»**

**ИНТЕРФЕЙС ТЦ**

Скриншот

Код приложения на XAML, C#

1.7 Создайте триггер, в котором запретите пользователю изменить статус ТЦ на “план”, если в редактируемом торговом центре есть павильоны со статусом «забронирован».

Проверьте работу триггера

Код триггера

Проверка работы триггера

**1.8 ИНТЕРФЕЙС «МЕНЕДЖЕРА C»**

**АРЕНДА ПАВИЛЬОНА**

Создайте хранимую процедуру, которая будет позволять арендовать или забронировать павильон в выбранном ТЦ.

Код процедуры

Вызов процедуры

Реализуйте триггер, который изменит статус арендованного или забронированного павильона.

Код триггера

Проверка работы триггера

Код приложения

Ссылка на проект в хранилище на GitHub

Выводы о полученных практических навыках