СОДЕРЖАНИЕ

[Введение 4](#_Toc105623516)

[1 Анализ предметной области 5](#_Toc105623517)

[2 Постановка задачи 6](#_Toc105623518)

[3 Проектирование программного модуля 7](#_Toc105623519)

[3.1 Классификаторы 7](#_Toc105623520)

[3.2 Инфологическая модель 8](#_Toc105623521)

[3.3 Описание входных и выходных данных 9](#_Toc105623522)

[3.4 Алгоритм решения задачи 9](#_Toc105623523)

[4 Технологии и инструменты разработки 11](#_Toc105623524)

[4.1 Технологии для разработки 11](#_Toc105623525)

[4.2 Инструменты разработки 12](#_Toc105623526)

[5 Разработка программного модуля 14](#_Toc105623527)

[5.1 Пользовательский интерфейс программного модуля 14](#_Toc105623528)

[5.2 Описание методов разработки 16](#_Toc105623529)

[Заключение 18](#_Toc105623530)

[Список использованных источников 19](#_Toc105623531)

[Приложение А 20](#_Toc105623532)

[Приложение Б 21](#_Toc105623533)

# Введение

В рамках распоряжения Правительства Российской Федерации от 17 декабря 2009 года № 1993-р утверждён сводный перечень первоочередных государственных и муниципальных услуг, предоставляемых органами исполнительной власти субъектов Российской Федерации и органами местного самоуправления в электронном виде, а также услуг, предоставляемых в электронном виде учреждениями субъектов Российской Федерации и муниципальными учреждениями согласно приложению № 1 этого распоряжения.

Каждое образовательное учреждение имеет какой — либо оборот информации. В каждом образовательном учреждении есть какое — либо количество работников, которые обязаны работать с этой информацией. Чтобы упростить работу со этой информацией её хранят в цифровом формате. Как раз поэтому и необходимо приложение управляющее базами данных с информацией о занятости кабинетов.

Актуальность темы обусловлена тем, что она позволяет без проблем осуществлять некоторые действия над кабинетами.

Цель курсового проекта заключается в создании программного модуля занятости кабинетов в учебных учреждениях на базе ОС Windows.

В курсовом проекте решаются задачи:

* формирование базы данных;
* просмотра кабинетов;
* просмотра учителей;
* просмотра групп;
* изменение групп, учителей и кабинетов с помощью приложения на ПК.

# 1 Анализ предметной области

Государственное бюджетное образовательное учреждение, как профессионально-техническое училище.

Информатизация образовательного процесса и управления образовательным учреждением реализуется по следующим направлениям:

* информатизация процесса обучения;
* информатизация профориентационной работы;
* информатизация методической работы;
* информатизация управленческой работы администрации и сотрудников.

Информационно-технологическая инфраструктура техникума соответствует современному уровню и включает в себя локальную сеть, объединяющую персональные компьютеры, копировально-множительную технику, презентационное оборудование.

Для подготовки обучающихся к занятиям с использованием сетевых учебных ресурсов и информационных интернет-ресурсов в техникуме имеется 16 компьютерных классов и читальный зал.

Парк компьютерной техники постоянно развивается и качественно, и количественно. В настоящее время общий парк вычислительной техники в техникуме насчитывает более 100 единиц IBM-совместимой вычислительной техники, из них стационарных персональных компьютеров – более 100 ед., ноутбуков – более 150 ед.

# 2 Постановка задачи

Задача курсового проекта – создание программного модуля занятости кабинетов в учебных учреждениях на базе ОС Windows для образовательных учреждений.

Разработанный модуль будет позволять производить различные действия над базой данных, содержащей в себе информацию о кабинетах, группах и учителях с помощью программного модуля установленного на персональный компьютер.

Программный модуль должен быть создан на языках С# и XAML при помощи системы для построения клиентских приложений Windows с возможностями взаимодействия с пользователем. Графический интерфейс программы дружественный и понятный, сама программа проста в использовании и не требует специального образования или прохождение курсов.

В программе должен быть использован шаблон Model-View-ViewModel, Command и дополнительные пакеты Microsoft.EntityFrameworkCore.

# 3 Проектирование программного модуля

## 3.1 Классификаторы

Классификатор представляет собой набор таблиц (справочников).

Объект – некоторая сущность в цифровом пространстве, обладающая определённым состоянием и поведением в данной предметной области.

Атрибут – элемент, описывающий любую характеристику объекта, важную для данной предметной области. Объектом в данной базе данных будет: информация о кабинетах приведенная в таблице 3.1, учителях показанных на таблице 3.2 и группах на таблице 3.3.

Таблица 3.1 – Атрибуты группы

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Кабинет | Номер  Название  статус |

Таблица 3.2 – Атрибуты студента

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Группа | Название |

Таблица 3.3 – Атрибуты оценки

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Учитель | Фамилия  Имя  Отчество |

## 3.2 Инфологическая модель

Инфологическая модель – это набор сущностей и связей между ними. Она содержит информацию о сущностях системы и способах их взаимодействия, включает идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

Для выполнения проекта была выбрана СУБД SQL Server. Инфологическая модель представлена на рисунке 3.1.

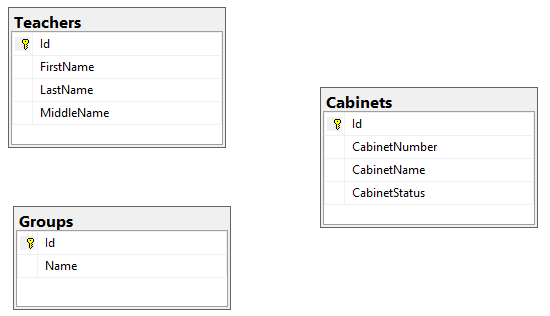


Рисунок 3.1 – Инфологическая модель

Структура таблиц «Кабинеты», «Учителя» и «Группы» базы данных представлена на рисунках 3.2 – 3.4 соответственно.

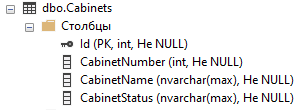


Рисунок 3.2 – Таблица «Кабинеты»

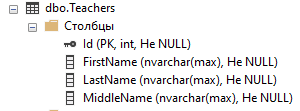


Рисунок 3.3 – Таблица «Учителя»

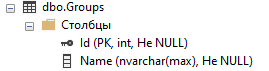


Рисунок 3.4 – Таблица «Группы»

## 3.3 Описание входных и выходных данных

Входная информация представляет собой данные об кабинетах, учителях и группах, которые заносятся в базу данных.

Выходной информацией являются уже обновленные данные об кабинетах, учителях и группах. Данные могут быть добавлены, отредактированы и удалены.

## 3.4 Алгоритм решения задачи

Для проведения каких-либо действий с кабинетами, учителями и группами необходимо выбрать действие: удалить, редактировать или добавить.

Последовательное описание блок – схемы приложения приведено ниже:

Шаг 1. Запуск программы.

Шаг 2. Вопрос: есть ли у вас аккаунт? Если нет, то переходим к шагу 3. Если есть, то переходим к шагу 4.

Шаг 3. Регистрируемся. Возвращаемся к авторизации.

Шаг 4. Вход в приложение под своей учетной записью.

Шаг 5. Вопрос: Выбрано ли действие? Если да, то переходим к шагу 6, если нет, то переходим к 8 шагу.

Шаг 6. Выполнение

Шаг 7. Возвращаемся к 4 шагу.

Шаг 8. Закрытие приложения.

Блок-схема приведена в приложении А.

Листинг программы приведен в приложении Б.

# 4 Технологии и инструменты разработки

## 4.1 Технологии для разработки

Для разработки данного приложения был выбран язык C# – объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

Windows Presentation Foundation (WPF) — это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность [1].

WPF является частью платформы .NET, и если ранее вы создавали приложения в .NET с помощью ASP.NET или Windows Forms, то общий процесс программирования вам уже знаком. WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларативную модель для программирования приложений [4].

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [5].

## 4.2 Инструменты разработки

В качестве среды для разработки приложения была выбрана программа Microsoft Visual Studio, линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight. Существует множество сред разработки, но данная была выбрана ввиду ее удобного графического интерфейса и средств отладки. Microsoft Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

Microsoft Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server) [6].

Главные возможности среды разработки Microsoft Visual Studio:

* включены все «интеллектуальные» возможности по редактированию кода;
* есть возможность визуального просмотра будущего приложения;
* сборка проекта работает быстро;
* удобный конструктор интерфейсов;
* удобное и интуитивно понятное логирование проекта;
* указанные цвета и рисунки, использованные в layout'e отображаются на границе в виде небольших превью, которые легко помогают понять какой конкретно ресурс используется;
* среда разработки является технологиями компании Microsoft;
* при выборе ресурса, его содержимое отображается во всплывающих окнах;
* возможность создания одним кликом новых окон и страниц;
* мониторинг используемой памяти;
* возможность подключать сторонние дополнения(плагины);
* возможность добавления новых инструментов.

Рекомендуемые системные требования к компьютеру на которой будет установлена Visual Studio следующие:

* операционная система – Microsoft® Windows® 7/8/10 (32- или 64-bit);
* 2 ГБ ОЗУ. рекомендуется 8 ГБ ОЗУ (минимум 2,5 ГБ при выполнении на виртуальной машине);
* место на жестком диске: до 210 ГБ (минимум 800 МБ) свободного места в зависимости от установленных компонентов; обычно для установки требуется от 20 до 50 ГБ свободного места;
* видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение WXGA (1366 на 768 пикселей) или более высокое.

# 5 Разработка программного модуля

## 5.1 Пользовательский интерфейс программного модуля

В окне входа в приложение, представленном на рисунке 5.1, находятся два текстовых поля логин и пароль, кнопка авторизации и регистрации. Для дальнейшей работы нужно верно указать логин и пароль. Если логин и пароль верны, то откроется основное окно, представленное на рисунке 5.2 с возможностью выбора действий. Кнопки с изображением лупы используются для сортировки данных, а при нажатии на кнопку таблицы откроется окно, с редактирование данных в БД, показанное на рисунке 5.3. Чтобы добавить, удалить или редактировать данные нужно выбрать таблицу и пункт для действия, как на рисунке 5.4.

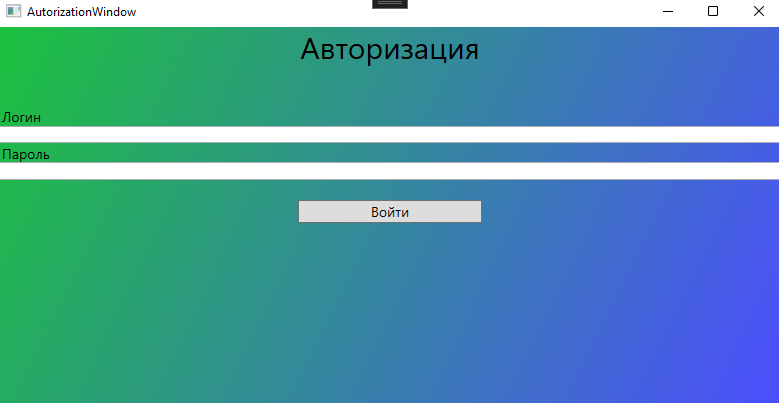


Рисунок 5.1 – Окно входа

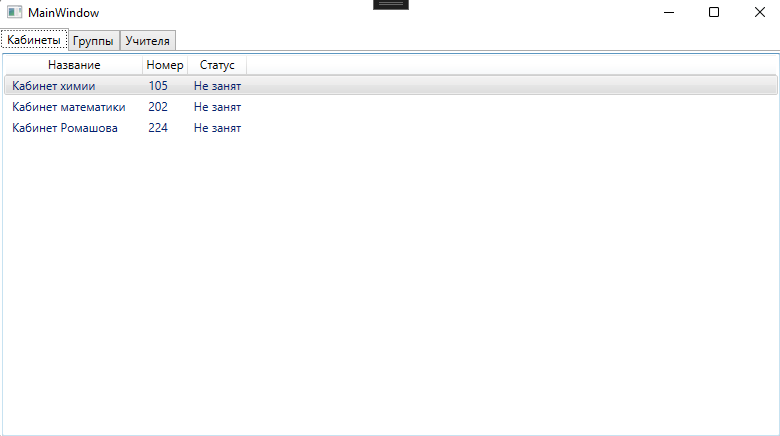


Рисунок 5.2 – Основное окно

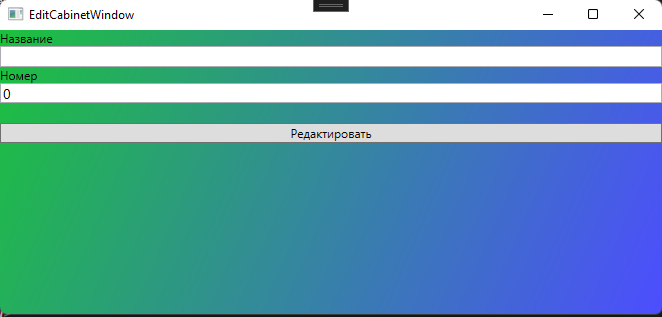


Рисунок 5.3 – Окно редактирования данных

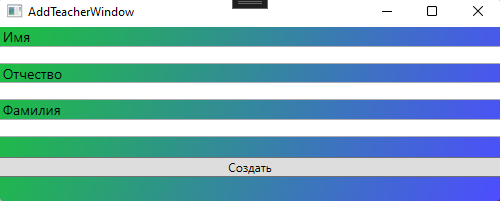


Рисунок 5.4 – Окно действия

## 5.2 Описание методов разработки

В шаблоне MVVM есть три основных компонента: модель, представление и модель представления. Каждый из них обслуживает отдельную цель. Кроме понимания обязанностей каждого компонента, важно понимать, как они взаимодействуют друг с другом. На высоком уровне в представлении «известно о модели представления и модели представления» известно о модели, но модель не знает модель представления, а модель представления не знает об этом представлении. Таким образом, модель представления изолирует представление от модели и позволяет модели развиваться независимо от представления.

Ниже приведены преимущества использования шаблона MVVM:

* если реализована существующая реализация модели, которая инкапсулирует существующую бизнес-логику, она может быть сложной или рискованной для ее изменения. В этом сценарии модель представления выступает в качестве адаптера для классов модели и позволяет избежать внесения значительных изменений в код модели;
* разработчики могут создавать модульные тесты для модели представления и модели без использования представления. Модульные тесты для модели представления могут выполнять точно те же функциональные возможности, которые используются в представлении;
* пользовательский интерфейс приложения можно переконструировать, не затрагивая код, при условии, что представление полностью реализовано в XAML. Поэтому Новая версия представления должна работать с существующей моделью представления;
* разработчики и разработчики могут одновременно работать с компонентами в процессе разработки. Дизайнеры могут сосредоточиться на представлении, тогда как разработчики могут работать над моделью представления и компонентами модели [3].

Команды являются привязанными объектами, что позволяет разделить логику и пользовательский интерфейс друг от друга.

Если рассматривать команды более подробно, то они представляют из себя следующее:

* команды представляют собой объекты, реализующие интерфейс ICommand;
* обычно команды связанны с какой-либо функцией;
* элементы пользовательского интерфейса привязываются к командам — кода интерфейс активируется пользователем, то выполняется команда — вызывается соответствующая функция;
* команды знают, включены ли они или нет;
* функции могут отключать команды – автоматическое отключение всех пользовательских элементов ассоциированных с ней;
* на самом деле существует множество различных применений команд. Например, использование команд для создания асинхронных функций, обеспечивающих логику, которая может быть проверена с/без помощи использования пользовательского интерфейса и др.

Entity Framework Core - представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, мы можем через EF Core работать с любой СУБД, если для нее имеется нужный провайдер [8].

# Заключение

Разработанный программный модуль занятости кабинетов в учебном учреждении для образовательных учреждений представляет собой простое приложение.

Программа разработана на языке С# в интегрированной среде разработки Microsoft Visual Studio с использованием СУБД SQL Server. Приложение интуитивно понятно даже неопытному пользователю и не затрудняет его работу с кабинетами, группами и учителями.

В дальнейшем в приложение будет обновляться и будут добавляться новые функции необходимые для работы с данными.

В процессе написания кода для данной программы были приобретены практические навыки в работе в СУБД SQL Server и языке программирования С#.

Цель курсового проекта достигнута. Задачи выполнены.

# Список использованных источников

Список использованной литературы

1. Моргунов, Е.П. PostgreSQL. Основы языка SQL / Е.П. Моргунов. – Санкт-Петербург : БХВ-Петербург, 2019.

Интернет ресурсы

1. Электронный фонд правовых и нормативно-технических документов. [Электронный ресурс] – Режим доступа: https://docs.cntd.ru/document/902191383
2. Википедия. Язык программирования C# [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp
3. Windows Presentation Foundation (WPF) [Электронный ресурс] – Режим доступа: – https://docs.microsoft.com/ru-ru/visualstudio/designers/getting-started-with-wpf?view=vs-2022
4. SQL Server 2019 [Электронный ресурс] – Режим доступа: – https://www.microsoft.com/ru-ru/sql-server/sql-server-2019
5. Microsoft Visual Studio. [Электронный ресурс] – Режим доступа: – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio
6. Шаблон Model-View-ViewModel. [Электронный ресурс] – Режим доступа:https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm
7. Документация по Entity Framework. [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/ef/

# Приложение А

(обязательное)

Блок – схема программы

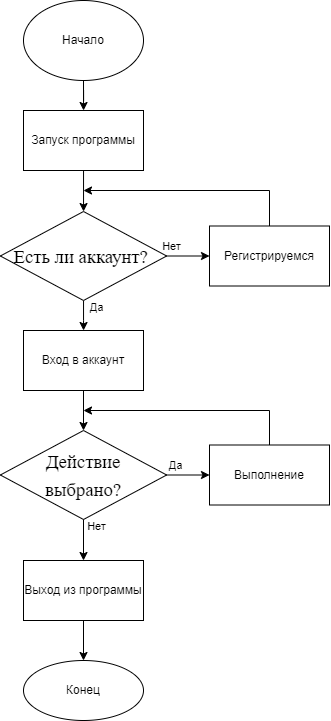


Рисунок А.1–Блок схема программы

# Приложение Б

(обязательное)

Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class AddCabinetViewModel : Command

{

#region COMMANDS TO ADD

public RelayCommand AddNewCabinet

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (CabinetName == null || CabinetName.Replace(" ", "").Length ==0

{

Command.ShowMessageToUser("Некорректное название");

}

else if (CabinetNumber == 0)

{

Command.ShowMessageToUser("Некорректный номер");

}

else

{

resultStr = DBWorker.CreateCabinet(CabinetNumber,CabinetName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class AddGroupViewModel : Command

{

#region COMMANDS TO ADD

public RelayCommand AddNewGroup

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (GroupName == null)

{

ShowMessageToUser("Не выбрана группа");

}

else

{

resultStr = DBWorker.CreateGroup(GroupName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class AddTeacherViewModel : Command

{

#region COMMANDS TO ADD

public RelayCommand AddNewTeacher

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (TeacherFirstName == null)

{

ShowMessageToUser("Не выбрано имя");

}

else if (TeacherMiddleName == null)

{

ShowMessageToUser("Не выбрано отчество");

}

else if (TeacherLastName == null)

{

ShowMessageToUser("Не выбрана фамилия");

}

else

{

resultStr = DBWorker.CreateTeacher(TeacherFirstName, TeacherLastName, TeacherMiddleName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

using WpF\_KYPCA4.MVVM.Views;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class AutorizationViewModel : Command

{

private void OpenMainWindow()

{

MainWindow mainWindow = new MainWindow();

SetCenterPositionAndOpen(mainWindow);

}

#region CheckLoginPssword

private RelayCommand checkAuthUser;

public RelayCommand CheckAuthUser

{

get

{

return checkAuthUser ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (Login == null || Login.Replace(" ", "").Length == 0)

{

ShowMessageToUser("Некорректный логин");

}

if (Password == null || Password.Replace(" ", "").Length == 0)

{

ShowMessageToUser("Некорректный пароль");

}

else

{

resultStr = "Неправильный логин или пароль";

if (DBWorker.CheckAuthUser(Login, Password))

{

resultStr = "Успешный вход";

ShowMessageToUser(resultStr);

OpenMainWindow();

SetNullValuesToProperties();

}

else

{

ShowMessageToUser(resultStr);

}

SetNullValuesToProperties();

}

}

);

}

}

#endregion

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class EditCabinetViewModel : Command

{

public RelayCommand EditCabinet

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (SelectedCabinet == null)

{

ShowMessageToUser("Не выбран кабинет");

}

else if (CabinetNumber == null)

{

ShowMessageToUser("Не выбран номер");

}

else if (CabinetName == null)

{

ShowMessageToUser("Не выбрано название");

}

else if (CabinetStatus == null)

{

ShowMessageToUser("Не выбран статус");

}

else

{

resultStr = DBWorker.EditCabinet(SelectedCabinet, CabinetNumber, CabinetName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class EditGroupViewModel : Command

{

public RelayCommand EditGroup

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (SelectedGroup == null)

{

ShowMessageToUser("Не выбрана группа");

}

else if (CabinetName == null)

{

ShowMessageToUser("Не выбрано название");

}

else

{

resultStr = DBWorker.EditGroup(SelectedGroup,GroupName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class EditTeacherViewModel : Command

{

public RelayCommand EditTeacher

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (SelectedTeacher == null)

{

ShowMessageToUser("Не выбран учитель");

}

else if (TeacherFirstName == null)

{

ShowMessageToUser("Не выбрано имя");

}

else if (TeacherLastName == null)

{

ShowMessageToUser("Не выбрана фамилия");

}

else if (TeacherMiddleName == null)

{

ShowMessageToUser("Не выбрано отчество");

}

else

{

resultStr = DBWorker.EditTeacher(SelectedTeacher, TeacherFirstName, TeacherLastName, TeacherMiddleName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

using WpF\_KYPCA4.MVVM.Views;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class MainViewModel : Command

{

public RelayCommand OpenAddWindow

{

get

{

return new RelayCommand(obj =>

{

if (SelectedTabItem.Name == "Cabinets")

{

OpenAddCabinetWindow();

}

else if (SelectedTabItem.Name == "Groups")

{

OpenAddGroupWindow();

}

else if (SelectedTabItem.Name == "Teachers")

{

OpenAddTeacherWindow();

}

});

}

}

public RelayCommand OpenEditWindow

{

get

{

return new RelayCommand(obj =>

{

if (SelectedTabItem.Name == "Cabinets")

{

OpenEditCabinetWindow();

}

else if (SelectedTabItem.Name == "Groups")

{

OpenEditGroupWindow();

}

else if (SelectedTabItem.Name == "Teachers")

{

OpenEditTeacherWindow();

}

});

}

}

#region SwapStatus

public RelayCommand OpenTakeWindow

{

get

{

return new RelayCommand(obj =>

{

if (SelectedCabinet==null)

{

ShowMessageToUser("Выберите кабинет");

}

else

{

OpenTakeCabinetWindow(SelectedCabinet.FullName);

}

});

}

}

public RelayCommand SwapStatus

{

get

{

return new RelayCommand(obj =>

{

string resultStr;

if (SelectedCabinet==null)

{

resultStr = "Выберите кабинет";

}

else

{

resultStr = DBWorker.NoTakeCabinet(SelectedCabinet);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

});

}

}

#endregion

private void OpenTakeCabinetWindow(string cabinet)

{

TakeCabinetWindow takeCabinetWindow = new TakeCabinetWindow(cabinet);

SetCenterPositionAndOpen(takeCabinetWindow);

}

private void OpenAddCabinetWindow()

{

AddCabinetWindow addCabinetWindow = new AddCabinetWindow();

SetCenterPositionAndOpen(addCabinetWindow);

}

private void OpenAddGroupWindow()

{

AddGroupWindow addGroupWindow = new AddGroupWindow();

SetCenterPositionAndOpen(addGroupWindow);

}

private void OpenAddTeacherWindow()

{

AddTeacherWindow addTeacherWindow = new AddTeacherWindow();

SetCenterPositionAndOpen(addTeacherWindow);

}

#region COMMANDS TO DELETE

public RelayCommand DeleteCabinet

{

get

{

return new RelayCommand(obj =>

{

string resultStr = "";

if (SelectedCabinet == null)

{

ShowMessageToUser("Не выбран кабинет");

}

else

{

resultStr = DBWorker.DeleteCabinet(SelectedCabinet);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

public RelayCommand DeleteGroup

{

get

{

return new RelayCommand(obj =>

{

string resultStr = "";

if (SelectedGroup == null)

{

ShowMessageToUser("Не выбрана группа");

}

else

{

resultStr = DBWorker.DeleteGroup(SelectedGroup);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

public RelayCommand DeleteTeacher

{

get

{

return new RelayCommand(obj =>

{

string resultStr = "";

if (SelectedTeacher == null)

{

ShowMessageToUser("Не выбран учитель");

}

else

{

resultStr = DBWorker.DeleteTeacher(SelectedTeacher);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

}

}

);

}

}

#endregion

private void OpenEditCabinetWindow()

{

EditCabinetWindow editCabinetWindow = new EditCabinetWindow();

SetCenterPositionAndOpen(editCabinetWindow);

}

private void OpenEditGroupWindow()

{

EditGroupWindow editGroupWindow = new EditGroupWindow();

SetCenterPositionAndOpen(editGroupWindow);

}

private void OpenEditTeacherWindow()

{

EditTeacherWindow editTeacherWindow = new EditTeacherWindow();

SetCenterPositionAndOpen(editTeacherWindow);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using WpF\_KYPCA4.Commands;

using WpF\_KYPCA4.MVVM.Models;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

internal class TakeCabinetViewModel:Command

{

#region COMMANDS TO TAKE

public RelayCommand TakeThisCabinet

{

get

{

return new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (SelectedTeacher == null)

{

ShowMessageToUser("Не выбран преподаватель");

}

else if(SelectedGroup==null)

{

ShowMessageToUser("Не выбрана группа");

}

else

{

resultStr = DBWorker.TakeCabinet(SelectedCabinet,SelectedTeacher,SelectedGroup);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

wnd.Close();

}

}

);

}

}

#endregion

}

}

using Microsoft.EntityFrameworkCore;

namespace WpF\_KYPCA4.MVVM.Models.Data

{

public class ApplicationContext : DbContext

{

public DbSet<Cabinet> Cabinets { get; set; }

public DbSet<Teacher> Teachers { get; set; }

public DbSet<Register> Registers { get; set; }

public DbSet<Group> Groups { get; set; }

public ApplicationContext() => Database.EnsureCreated();

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

//optionsBuilder.UseSqlServer("Data Source=DESKTOP-I8L1GP6;Initial Catalog=Cabinets;Integrated Security = True");

//optionsBuilder.UseSqlServer(@"Data Source=localhost\SQLEXPRESS;Initial Catalog=Cabinets;User ID=sa; Password=4321193");

optionsBuilder.UseSqlServer(@"Data Source=MSI-AEGIS-TI3;Initial Catalog=Cabinets;Integrated Security = True");

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WpF\_KYPCA4.Commands;

namespace WpF\_KYPCA4.MVVM.Models

{

public class Cabinet

{

public int Id { get; set; }

public int CabinetNumber { get; set; }

public string CabinetName { get; set; } = null!;

public string CabinetStatus { get; set;} = "Не занят";

[NotMapped]

public string FullName => $"{CabinetName} - {CabinetNumber}";

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpF\_KYPCA4.MVVM.Models

{

public class Group

{

public int Id { get; set; }

public string Name { get; set; } = null!;

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpF\_KYPCA4.MVVM.Models

{

public class Register

{

public int Id { get; set; }

public string Login { get; set; } = null!;

public string Password { get; set; } = null!;

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpF\_KYPCA4.MVVM.Models

{

public class Teacher

{

public int Id { get; set; }

public string FirstName { get; set; } = null!;

public string LastName { get; set; } = null!;

public string MiddleName { get; set; } = null!;

[NotMapped]

public string FullName => $"{LastName} {FirstName[0]}. {MiddleName[0]}.";

}

}

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace WpF\_KYPCA4.Migrations

{

public partial class Sheeeshki1 : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Cabinets",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

CabinetNumber = table.Column<int>(type: "int", nullable: false),

CabinetName = table.Column<string>(type: "nvarchar(max)", nullable: false),

CabinetStatus = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Cabinets", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Groups",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Groups", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Registers",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Login = table.Column<string>(type: "nvarchar(max)", nullable: false),

Password = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Registers", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Teachers",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

FirstName = table.Column<string>(type: "nvarchar(max)", nullable: false),

LastName = table.Column<string>(type: "nvarchar(max)", nullable: false),

MiddleName = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Teachers", x => x.Id);

});

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Cabinets");

migrationBuilder.DropTable(

name: "Groups");

migrationBuilder.DropTable(

name: "Registers");

migrationBuilder.DropTable(

name: "Teachers");

}

}

}

using WpF\_KYPCA4.MVVM.Models;

using WpF\_KYPCA4.MVVM.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using WpF\_KYPCA4.Commands;

namespace WpF\_KYPCA4.MVVM.ViewModels

{

public class Command : INotifyPropertyChanged

{

private List<Teacher> allTeachers = DBWorker.GetAllTeachers();

public List<Teacher> AllTeachers

{

get { return allTeachers; }

set

{

allTeachers = value;

NotifyPropertyChanged("AllTeachers");

}

}

private List<Cabinet> allCabinets = DBWorker.GetAllCabinets();

public List<Cabinet> AllCabinets

{

get { return allCabinets; }

set

{

allCabinets = value;

NotifyPropertyChanged("AllCabinets");

}

}

private List<Group> allGroups = DBWorker.GetAllGroups();

public List<Group> AllGroups

{

get { return allGroups; }

set

{

allGroups = value;

NotifyPropertyChanged("AllGroups");

}

}

public static Cabinet CabinetInf { get; set; }

public static string CabinetName { get; set; }

public static int CabinetNumber { get; set; }

public static string CabinetStatus { get; set; }

public static string Login { get; set; }

public static string Password { get; set; }

public static Teacher TeacherInf { get; set; }

public static string TeacherFirstName { get; set; }

public static string TeacherLastName { get; set; }

public static string TeacherMiddleName { get; set; }

public Group GroupInf { get; set; }

public static string GroupName { get; set; }

public TabItem SelectedTabItem { get; set; }

public static Teacher SelectedTeacher { get; set; }

public static Cabinet SelectedCabinet { get; set; }

public static Group SelectedGroup { get; set; }

#region METHODS TO OPEN WINDOW

protected void SetCenterPositionAndOpen(Window window)

{

window.WindowStartupLocation = WindowStartupLocation.CenterScreen;

window.ShowDialog();

}

#endregion

#region UPDATE VIEWS

public static void SetNullValuesToProperties()

{

TeacherFirstName = null;

TeacherLastName = null;

TeacherMiddleName = null;

CabinetName = null;

CabinetNumber = 0;

TeacherFirstName = null;

TeacherLastName = null;

TeacherMiddleName = null;

GroupName = null;

CabinetStatus = null;

Login = null;

Password = null;

}

protected void UpdateInfoView()

{

UpdateCabinetsMWInfo();

UpdateGroupsMWInfo();

UpdateTeachersMWInfo();

}

protected void UpdateCabinetsMWInfo()

{

MainWindow.LWCabinets.ItemsSource = null;

MainWindow.LWCabinets.Items.Clear();

MainWindow.LWCabinets.ItemsSource = DBWorker.GetAllCabinets();

MainWindow.LWCabinets.Items.Refresh();

}

protected void UpdateGroupsMWInfo()

{

MainWindow.LWGroups.ItemsSource = null;

MainWindow.LWGroups.Items.Clear();

MainWindow.LWGroups.ItemsSource = DBWorker.GetAllGroups();

MainWindow.LWGroups.Items.Refresh();

}

protected void UpdateTeachersMWInfo()

{

MainWindow.LWTeachers.ItemsSource = null;

MainWindow.LWTeachers.Items.Clear();

MainWindow.LWTeachers.ItemsSource = DBWorker.GetAllTeachers();

MainWindow.LWTeachers.Items.Refresh();

}

#endregion

public static void ShowMessageToUser(string message)

{

MessageBox.Show(message,"Информация",MessageBoxButton.OK,MessageBoxImage.Information);

}

public event PropertyChangedEventHandler PropertyChanged;

private void NotifyPropertyChanged(String propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

}