Министерство общего и профессионального образования

Ростовской области

государственное бюджетное профессиональное образовательное учреждение

Ростовской области

«Волгодонский техникум информационных технологий,

бизнеса и дизайна имени В. В. Самарского»

**ОТЧЕТ**

о производственной практике

**09.02.07 Информационные системы и программирование**

**Квалификация: Программист**

Выполнил:

студентгруппы ***ИСП-19,***

***Аксёнов А.И.***

Принял:

преподаватель ***Ромашов Р.В.***

Оценка:

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Волгодонск

2022

СОДЕРЖАНИЕ

[Введение 3](#_Toc120901113)

[1 Анализ предметной области 4](#_Toc120901114)

[2 Постановка задачи 5](#_Toc120901115)

[3 Проектирование программного модуля 6](#_Toc120901116)

[3.1 Инфологическая модель 6](#_Toc120901117)

[3.2 Описание входных и выходных данных 7](#_Toc120901118)

[4 Технологии и инструменты разработки 8](#_Toc120901119)

[4.1 Технологии для разработки 8](#_Toc120901120)

[4.2 Инструменты разработки 9](#_Toc120901121)

[5 Разработка программного модуля 11](#_Toc120901122)

[5.1 Пользовательский интерфейс программного модуля 11](#_Toc120901123)

[5.2 Описание методов разработки 12](#_Toc120901124)

[Заключение 15](#_Toc120901125)

[Список использованных источников 16](#_Toc120901126)

# Введение

В настоящее время достаточно большое количество предприятий работает в программе «1С». Программа «1С» — это продукт для автоматизации предприятий. Перед пользователями открыт широкий ассортимент типовых решений и отраслевых конфигураций. С их помощью возможно автоматизировать бизнес-процессы в любой компании, независимо от масштабов и направлений.

Программы 1С:Предприятие созданы специально для решения всех известных задач по управлению компанией и учету. Конфигурация может содержать в себе не только стандартный набор общих функций, но и возможности под конкретную отрасль с учетом индивидуальных задач отдельной компании. Любую программу «1С:Предприятие 8» можно адаптировать для конкретной отрасли.

Целью данной производственной практики будет получение навыка разработки приложения для предприятия ООО «БИЗНЕС РЕШЕНИЯ», приложение будет создано с помощью программного обеспечения установленного на ПК практиканта. Приложение будет сделано по техническому заданию которое было выдано руководителем практики.

# 1 Анализ предметной области

ООО «Бизнес решения» официальный партнер фирмы «1С» со статусом 1С: Франчайзи уже более 13 лет. Осуществляет продажу, консультацию, настройку и поддержку продуктов 1С на предприятиях разного уровня. Это как малые филиалы, так и большие фирмы. Также активно развивают партнерство с учебными учреждениями, поддерживают партнерские отношения с другими филиалами 1С в других городах. Принимают на своей платформе студентов для прохождения производственной практики. Осуществляют обучение заинтересованных молодых специалистов за счёт предприятия с дальнейшей отработкой через трудовой договор.

Основным видом деятельности компании ООО «БИЗНЕС РЕШЕНИЯ» является разработка компьютерного программного обеспечения на платформе 1С, консультационные услуги в области разработки компьютерного программного обеспечения на платформе 1С и другие сопутствующие услуги.

В «1С» есть несколько конфигураций. Это универсальные программы, подходящие для большинства организаций и предпринимателей. Они позволяют вести учет для разных систем налогообложения и для разных форм собственности.

Наиболее востребованные виды «1С»

«1С:Бухгалтерия»

««1С:Зарплата и управление персоналом»

«1С:Управление нашей фирмой»

«1С:Управление торговлей»

«1С:Розница»

# 2 Постановка задачи

Задача производственной практики – создание программного модуля работы с товарами и услугами на базе ОС Windows для ООО «БИЗНЕС РЕШЕНИЯ».

Разрабатываемый модуль должен позволять производить различные действия над базой данных содержащей в себе информацию о товарах, заказах, услугах и пользователях 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 должен быть дружественный и понятный, сама программа должна быть проста в использовании и не требовать специального образования или прохождения 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 пользователях приведенная в таблице 3.1, товарах показанных в таблице 3.2, услугах в таблице 3.3, запросах в таблице 3.4, заказах в таблице 3.5, категориях товаров в таблице 3.6, ролях пользователя в таблице 3.7.

Таблица 3.1 – Атрибуты пользователя

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Пользователь | Идентификационный код  Логин  Пароль  Имя  Фамилия  Отчество  Номер телефона  Email |

Таблица 3.2 – Атрибуты товара

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Товар | Идентификационный код  Название  Описание  Изображение  Стоимость  Идентификационный код категории |

Таблица 3.3 – Атрибуты услуги

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Услуга | Идентификационный код  Название  Описание |

Таблица 3.4 – Атрибуты запроса

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Запрос | Идентификационный код  Идентификационный код пользователя  Идентификационный код услуги  Статус выполнения |

Таблица 3.5 – Атрибуты заказа

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Заказ | Идентификационный код  Идентификационный код товара  Идентификационный код пользователя  Статус |

Таблица 3.6 – Атрибуты категории товара

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Категория товара | Идентификационный код  Название |

Таблица 3.7 – Атрибуты роли

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Роль | Идентификационный код  Название |

## 3.2 Инфологическая модель

Инфологическая модель - это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.

Для выполнения проекта была выбрана СУБД SQL Server. Инфологическая модель (схема данных) БД представлена на рисунке 3.1.

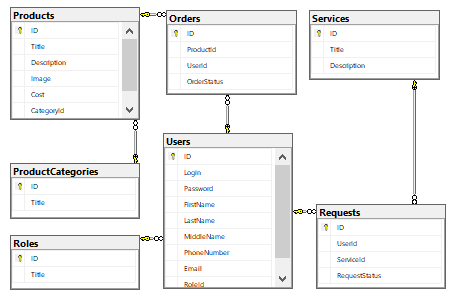


Рисунок 3.1 – Схема данных

Структура таблиц «Товары», «Категории товаров», «Услуги», «Заказы», «Запросы», «Пользователи» и «Роли» базы данных представлена на рисунках 3.2 – 3.8 соответственно.

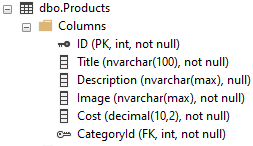


Рисунок 3.2 – Таблица «Товары»

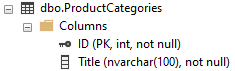


Рисунок 3.3 – Таблица «Категории товаров»

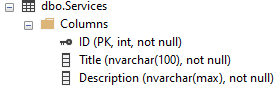


Рисунок 3.4 – Таблица «Услуги»

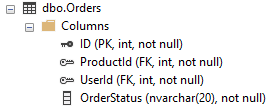


Рисунок 3.5 – Таблица «Заказы»

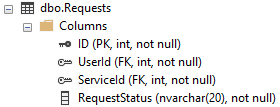


Рисунок 3.6 – Таблица «Запросы»

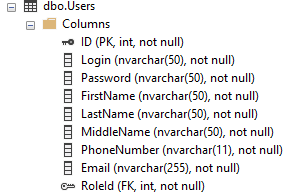


Рисунок 3.7 – Таблица «Пользователи»

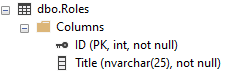


Рисунок 3.8 – Таблица «Роли»

## 3.3 Описание входных и выходных данных

Входная информация представляет собой данные об товарах, категориях товаров, услугах, заказах, запросах, пользователях и ролях пользователей, которые заносятся в базу данных.

Выходной информацией являются уже обновленные данные об товарах, категориях товаров, услугах, заказах, запросах, пользователях и ролях пользователей. Данные могут быть добавлены, отредактированы и удалены.

# 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 [2].

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.

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 [3].

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 [1]. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [4].

## 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 в 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.

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) [5].

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 Создание технического и рабочего проектирования

Основанием для разработки технического проекта системы служит техническое задание, утвержденное заказчиком.

Технический проект системы — это техническая документация, утвержденная в установленном порядке, содержащая общесистемные проектные решения, алгоритм решения задач, а также оценку экономической эффективности автоматизированной системы управления и перечень мероприятий по подготовке объекта к внедрению.

Технический проект разрабатывается в целях определения основных проектных решений по созданию системы. На этом этапе осуществляется комплекс научно-исследовательских и экспериментальных работ для выбора наилучших вариантов решений, проводятся экспериментальная проверка основных проектных решений и расчет экономической эффективности системы. Фактически технический проект содержит комплекс экономико-математических и алгоритмических моделей.

Рабочее проектирование заключается в разработке материалов, обеспечивающих эксплуатацию автоматизированной системы обработки информации.

Рабочий проект — это техническая документация, утвержденная в установленном порядке, содержащая уточненные данные и детализированные общесистемные проектные решения, программы и инструкции по решению задач, а также уточненную оценку экономической эффективности автоматизированной системы управления и уточненный перечень мероприятий по подготовке объекта к внедрению. Рабочий проект разрабатывается на основе технического проекта, утвержденного заказчиком.

## 5.2 Пользовательский интерфейс программного модуля

В окне входа в приложение, представленном на рисунке 5.1, находятся два текстовых поля логин и пароль и кнопка авторизации. 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. В главном окне присутствует возможность сортировки, поиска, фильтрации данных о товарах, а также кнопки для управления товарами и услугами. При нажатии на кнопку заказов откроется окно со всеми заказами пользователя, представленное на рисунке 5.3. При нажатии на кнопку услуг откроется окно со всеми услугами пользователя, с возможностью запроса, представленное на рисунке 5.4. При нажатии на кнопку настройки откроется окно со всеми данными пользователя, с возможностью их изменения, функционал различен от роли пользователя, представленное на рисунке 5.5. При нажатии на кнопки администратора откроется окно с возможностью редактирования некоторых данных о пользователях, запросах и заказах, представленное на рисунке 5.6.

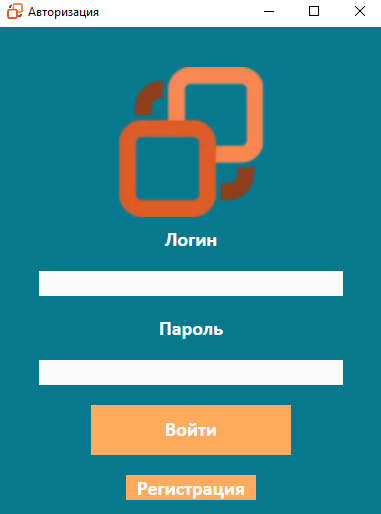


Рисунок 5.1 – Окно входа

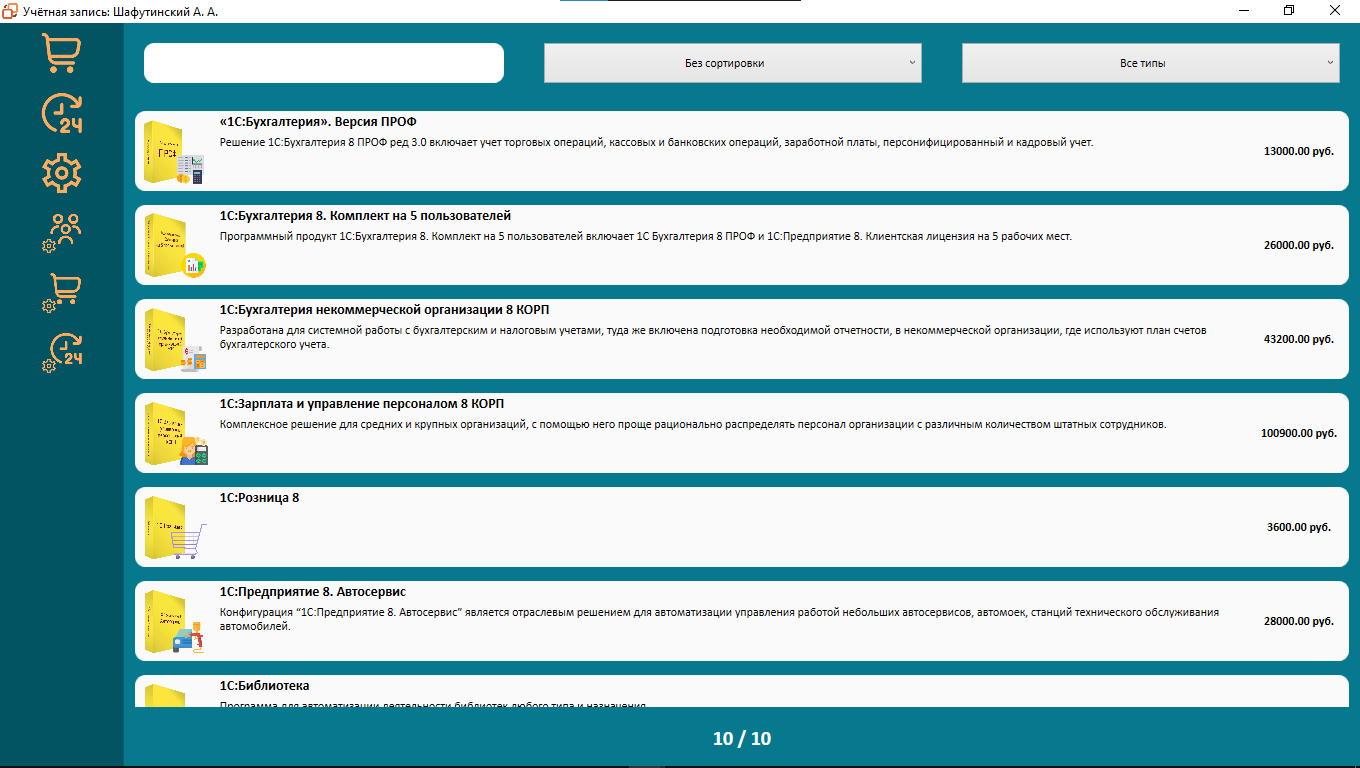


Рисунок 5.2 – Основное окно

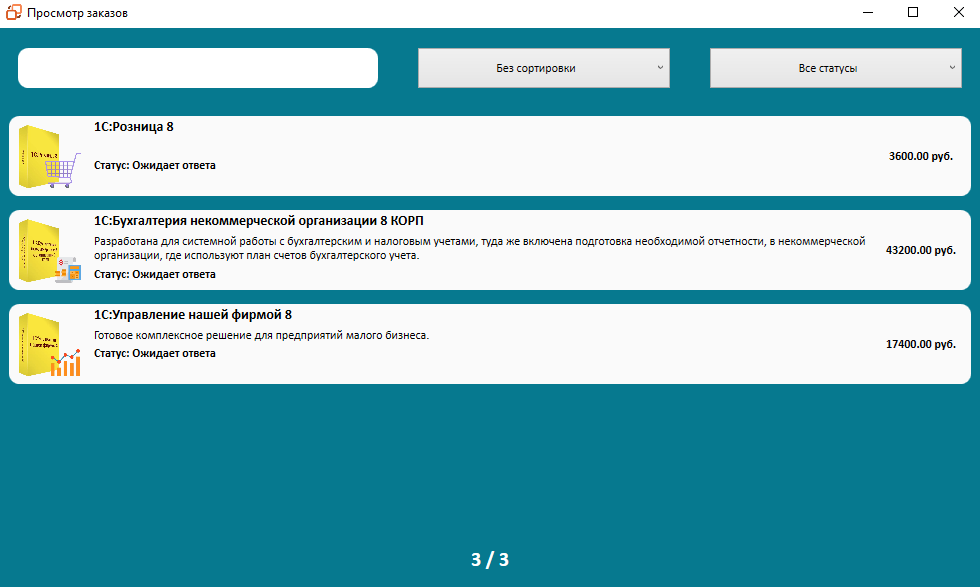


Рисунок 5.3 – Окно просмотра заказов

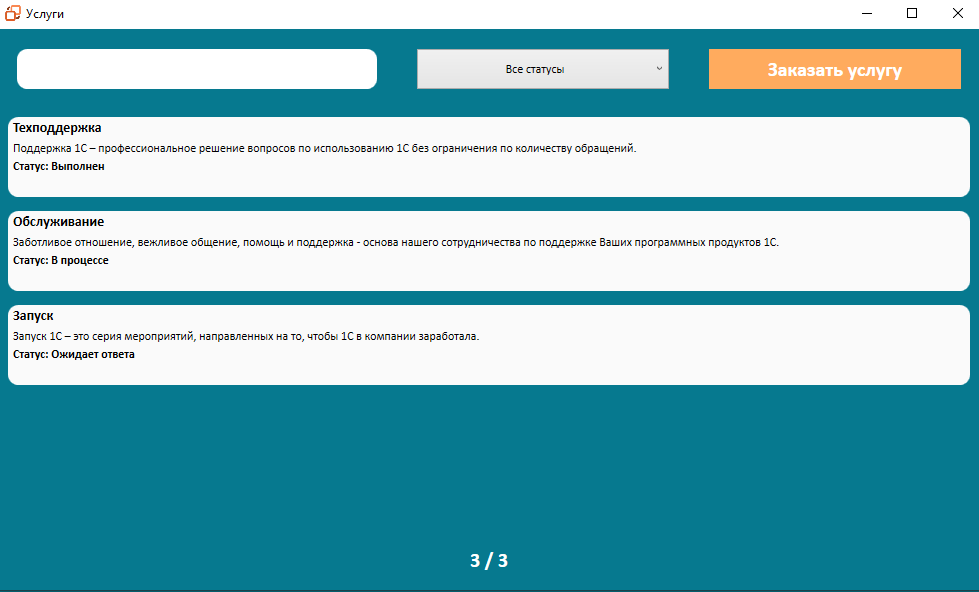


Рисунок 5.4 – Окно просмотра услуг



Рисунок 5.5 – Окно данных пользователя

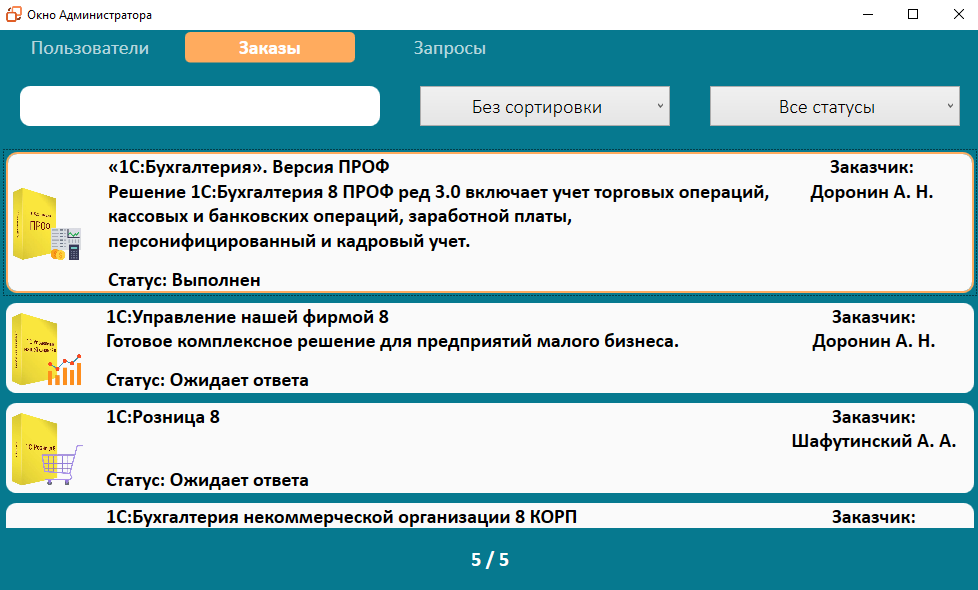


Рисунок 5.6 – Окно администратора

## 5.3 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:

* eсли 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 могут одновременно работать с компонентами в процессе разработки. Дизайнеры могут сосредоточиться на представлении, тогда как разработчики могут работать над моделью представления и компонентами модели [6].

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 [7].

Разработанный программный модуль работы с товарами и услугами работает по следующему простому алгоритму приведённому 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. Возвращаемя 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 5 шагу.

Шаг 8. Закрытие приложения.

Блок-схема приведена в приложении А.

Листинг программы приведен в приложении Б.

5.4 Охрана труда и техника безопасности при работе с ПК и оргтехникой

К работе на персональном компьютере допускаются лица, прошедшие обучение безопасным методам труда, вводный инструктаж, первичный инструктаж на рабочем месте. При эксплуатации персонального компьютера на работника могут оказывать действие следующие опасные и вредные производственные факторы:

* повышенный уровень электромагнитных излучений;
* повышенный уровень статического электричества;
* пониженная ионизация воздуха;
* статические физические перегрузки;
* перенапряжение зрительных анализаторов.

Требования безопасности перед началом работы:

* Подготовить рабочее место.
* Отрегулировать освещение на рабочем месте, убедиться в отсутствии бликов на экране.
* Проверить правильность подключения оборудования к электросети.
* Проверить исправность проводов питания и отсутствие оголенных участков проводов.
* Убедиться в наличии заземления системного блока, монитора и защитного экрана.
* Протереть антистатической салфеткой поверхность экрана монитора и защитного экрана.
* Проверить правильность установки стола, стула, подставки для ног, пюпитра, угла наклона экрана, положение клавиатуры, положение "мыши" на специальном коврике, при необходимости произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

# 6 Задачи производственной практики

# Заключение

Разработанный программный модуль работы с заказами для ООО «БИЗНЕС РЕШЕНИЯ» представляет собой простое приложение.

Программа разработана на языке С# в интегрированной среде разработки Microsoft Visual Studio с использованием СУБД SQL Server и шаблона Model-View-ViewModel. Приложение интуитивно понятно даже неопытному пользователю и не затрудняет его работу с оценками, группами, дисциплинами и студентами.

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

В процессе написания кода для данной программы были приобретены практические навыки в работе в СУБД SQL Server, шаблоном Model-View-ViewModel и языком программирования С#.

# Список использованных источников

Список использованной литературы

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

Интернет ресурсы

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

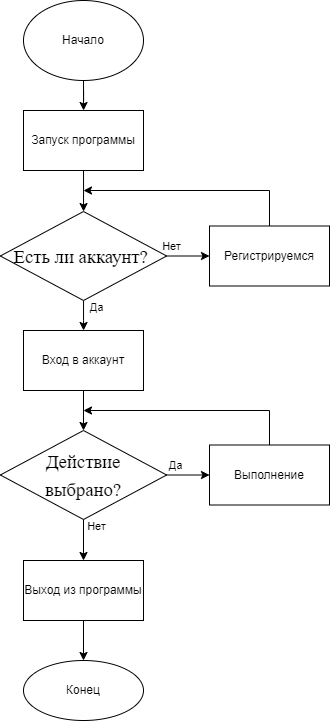


Рисунок А.1–Блок схема программы

# Приложение Б

(обязательное)

Листинг программы

namespace GraduateWork.ViewModels

{

internal class LoginWindowViewModel:ViewModelBase

{

private User \_currentUser=new();

public User CurrentUser

{

get => \_currentUser;

set => Set(ref \_currentUser, value, nameof(CurrentUser));

}

internal void Login()

{

using(ApplicationDbContext context = new())

{

if((CurrentUser.Login==string.Empty || CurrentUser.Login==null)

||(CurrentUser.Password==null || CurrentUser.Password==string.Empty))

MessageBox.Show("Введите данные!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

else if (context.Users.Where(u => u.Login == CurrentUser.Login && u.Password == CurrentUser.Password).Count() == 0)

MessageBox.Show("Неправильный логин или пароль!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

else

new MainWindow(context.Users.Include(u=>u.Role)

.Where(u => u.Login == CurrentUser.Login && u.Password == CurrentUser.Password)

.Single()).ShowDialog();

}

}

}

}

namespace GraduateWork.ViewModels

{

internal class MainWindowViewModel:ViewModelBase

{

private User \_currentUser;

private List<Product> \_displayingProducts;

private string \_elementsCount;

private Product \_selectedProduct;

private string \_searchValue;

private string \_filtherValue;

private string \_sortValue;

public List<string> FiltherValues { get; } = new List<string>()

{

"Все типы"

};

public List<string> SortValues { get; } = new List<string>()

{

"Без сортировки",

"По цене (возр.)",

"По цене (убыв.)",

};

public User CurrentUser

{

get => \_currentUser;

set => Set(ref \_currentUser, value, nameof(CurrentUser));

}

public List<Product> DisplayingProducts

{

get => \_displayingProducts;

set

{

Set(ref \_displayingProducts, value, nameof(DisplayingProducts));

ElementsCount = $"{value.Count} / {GetProducts().Count}";

}

}

public string ElementsCount

{

get => \_elementsCount;

set => Set(ref \_elementsCount, value, nameof(ElementsCount));

}

public string SearchValue

{

get => \_searchValue;

set

{

Set(ref \_searchValue, value, nameof(SearchValue));

DisplayProducts();

}

}

public string FiltherValue

{

get => \_filtherValue;

set

{

Set(ref \_filtherValue, value, nameof(FiltherValue));

DisplayProducts();

}

}

public string SortValue

{

get => \_sortValue;

set

{

Set(ref \_sortValue, value, nameof(SortValue));

DisplayProducts();

}

}

public Product SelectedProduct

{

get => \_selectedProduct;

set => Set(ref \_selectedProduct, value, nameof(SelectedProduct));

}

public MainWindowViewModel(User currentUser)

{

CurrentUser = currentUser;

using (ApplicationDbContext context = new())

{

DisplayingProducts= new List<Product>(GetProducts());

FiltherValues.AddRange(context.ProductCategories.Select(pc=>pc.Title));

}

SearchValue = null;

FiltherValue = FiltherValues[0];

SortValue = SortValues[0];

}

private List<Product> GetProducts()

{

using (ApplicationDbContext context = new())

{

return context.Products

.Include(pc => pc.Category)

.OrderBy(p=>p.Id)

.ToList();

}

}

private void DisplayProducts()

{

DisplayingProducts = Sort(Search(Filther(GetProducts())));

}

private List<Product> Search(List<Product> products)

{

if (SearchValue == null || SearchValue == string.Empty)

return products;

else

return products.Where(p => p.Title.ToLower().Contains(SearchValue.ToLower())).ToList();

}

private List<Product> Filther(List<Product> products)

{

if (FiltherValue == FiltherValues[0])

return products;

else

return products.Where(p=>p.Category.Title==FiltherValue).ToList();

}

private List<Product> Sort(List<Product> products)

{

if (SortValue == SortValues[1])

return products.OrderBy(p => p.Cost).ToList();

else if (SortValue == SortValues[2])

return products.OrderByDescending(p => p.Cost).ToList();

else

return products;

}

internal void AddOrder()

{

using (ApplicationDbContext context = new())

{

Order newOrder = new Order()

{

UserId=CurrentUser.Id,

ProductId=SelectedProduct.Id,

OrderStatus = "Ожидает ответа",

};

context.Orders.Add(newOrder);

context.SaveChanges();

MessageBox.Show("Продукт успешно заказан!","Уведомление",MessageBoxButton.OK,MessageBoxImage.Information);

}

}

internal void OpenUserOrders()

{

new UserOrdersWindow(CurrentUser).ShowDialog();

}

internal void OpenUserRequests()

{

new UserRequestsWindow(CurrentUser).ShowDialog();

}

internal void OpenUserSetings()

{

new UserSettingsWindow(CurrentUser).ShowDialog();

}

internal void OpenAdminControl(int tabIndex)

{

new AdminControlWindow().ShowDialog();

}

}

}

namespace GraduateWork.ViewModels

{

abstract internal class ViewModelBase : INotifyPropertyChanged

{

public event PropertyChangedEventHandler? PropertyChanged;

protected bool Set<T>(ref T field,T value,string propertyName)

{

if(EqualityComparer<T>.Default.Equals(field,value))

return false;

field = value;

OnProperyChnged(propertyName);

return true;

}

private void OnProperyChnged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

namespace GraduateWork.ViewModels

{

internal class UserOrdersWindowViewModels : ViewModelBase

{

private User \_currentUser;

private Order \_selectedOrder;

private string \_elementsCount;

private string \_searchValue;

private string \_filtherValue;

private string \_sortValue;

private List<Order> \_displayingOrders;

public List<string> FiltherValues { get; } = new List<string>()

{

"Все статусы",

"Ожидает ответа",

"В процессе",

"Выполнен",

};

public List<string> SortValues { get; } = new List<string>()

{

"Без сортировки",

"По цене (возр.)",

"По цене (убыв.)",

};

public User CurrentUser

{

get => \_currentUser;

set => Set(ref \_currentUser, value, nameof(CurrentUser));

}

public List<Order> DisplayingOrders

{

get => \_displayingOrders;

set

{

Set(ref \_displayingOrders, value, nameof(DisplayingOrders));

ElementsCount = $"{value.Count} / {GetOrders().Count}";

}

}

public Order SelectedOrder

{

get => \_selectedOrder;

set => Set(ref \_selectedOrder, value, nameof(SelectedOrder));

}

public string ElementsCount

{

get => \_elementsCount;

set => Set(ref \_elementsCount, value, nameof(ElementsCount));

}

public string SearchValue

{

get => \_searchValue;

set

{

Set(ref \_searchValue, value, nameof(SearchValue));

DisplayOrders();

}

}

public string FiltherValue

{

get => \_filtherValue;

set

{

Set(ref \_filtherValue, value, nameof(FiltherValue));

DisplayOrders();

}

}

public string SortValue

{

get => \_sortValue;

set

{

Set(ref \_sortValue, value, nameof(SortValue));

DisplayOrders();

}

}

public UserOrdersWindowViewModels(User currentUser)

{

CurrentUser = currentUser;

using (ApplicationDbContext context = new())

{

DisplayingOrders= new List<Order>(GetOrders());

}

SearchValue = null;

SortValue = SortValues[0];

FiltherValue = FiltherValues[0];

}

private List<Order> GetOrders()

{

using (ApplicationDbContext context = new())

{

return context.Orders

.Include(p=>p.Product)

.Include(u=>u.User)

.OrderBy(o => o.Id)

.Where(o => o.UserId == CurrentUser.Id)

.ToList();

}

}

private void DisplayOrders()

{

DisplayingOrders = Sort(Search(Filther(GetOrders())));

}

private List<Order> Search(List<Order> orders)

{

if (SearchValue == null || SearchValue == string.Empty)

return orders;

else

return orders.Where(o => o.Product.Title.ToLower().Contains(SearchValue.ToLower())).ToList();

}

private List<Order> Sort(List<Order> orders)

{

if (SortValue == SortValues[1])

return orders.OrderBy(o => o.Product.Cost).ToList();

else if (SortValue == SortValues[2])

return orders.OrderByDescending(o => o.Product.Cost).ToList();

else

return orders;

}

private List<Order> Filther(List<Order> orders)

{

if (FiltherValue == FiltherValues[0])

return orders;

else

return orders.Where(o => o.OrderStatus == FiltherValue).ToList();

}

internal void DeleteUserOrder()

{

using (ApplicationDbContext context = new())

{

var result = MessageBox.Show("Вы точно хотите отменить этот заказ?","Внимание!",MessageBoxButton.YesNo, MessageBoxImage.Question);

if(result == MessageBoxResult.Yes)

{

context.Orders.Remove(SelectedOrder);

context.SaveChanges();

DisplayOrders();

}

}

}

}

}