**Содержание**

Введение 6

1 Характеристика места практики 8

2 Сведения об имеющейся вычислительной технике и программном обеспечении 10

3 Сведения о прохождении практики в качестве дублера инженерно-технического работника 13

4 Индивидуальное задание 15

4.1 Постановка задачи 15

4.2 Проектирование структуры программы 16

4.2.1 Desktop версия приложения 16

4.3 Проектирование базы данных 17

4.4 Функциональное взаимодействие модулей программы 19

4.4.1 Desktop версия приложения 19

4.5 Работа с программой 24

4.5.1 Desktop версия приложения 24

Заключение 31

Список использованных источников 33

Приложение А (справочное) Основные процедуры desktop приложения 34

**ВВЕДЕНИЕ**

Целью производственной практики по ПМ.02 Осуществление интеграции программных модулей является закрепление и совершенствование приобретенных в процессе обучения профессиональных умений обучающихся по специальности, развитие общих и профессиональных компетенций, адаптация к конкретным условиям деятельности предприятий и организаций, освоение современных производственных процессов, обучение трудовым приемам, операциям и способам выполнения трудовых процессов.

В процессе прохождения производственной практики необходимо будет решить следующие задачи:

* приобрести практический опыт разработки требований к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент;
* приобрести практический опыт выполнения интеграции модулей в программное обеспечение;
* приобрести практический опыт отладки программного модуля с использованием специализированных программных средств, разработки тестовых наборов и тестовых сценариев для программного обеспечения;
* приобрести практический опыт инспектирования компонент программного обеспечения на предмет соответствия стандартам кодирования;
* сформировать профессиональные умения и навыки;
* приобрести практические навыки в будущей профессиональной деятельности при выполнении функций дублера инженерно-технических работников;
* выполнить индивидуальное задание.

В результате прохождения производственной практики необходимо будет подтвердить уровень овладения видом деятельности «Осуществление интеграции программных модулей» и следующими профессиональными компетенциями:

* ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент;
* ПК 2.2. Выполнять интеграцию модулей в программное обеспечение;
* ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств;
* ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения;
* ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

Производственная практика проводится на базе ИП: Ваньшев Алексей Александрович

**1 ХАРАКТЕРИСТИКА МЕСТА ПРАКТИКИ**

* 1. Общая характеристика места прохождения практики:

Название: ИП – Ваньшев Алексей Александрович;

Адрес: г. Майкоп , ул. Кужорская 97, кв. 1;

Руководитель предприятия: Ваньшев Алексей Александрович;

Руководитель отдела: Ваньшев Алексей Александрович, Начальник;

Сфера деятельности предприятия: Ремонт техники, настройка и разработка ПО, тех. поддержка клиентов;

Правила охраны труда и техники безопасности предназначены для обеспечения безопасных условий труда и защиты жизни и здоровья работников на производстве. Основные правила:

* Работники должны быть обучены правилам охраны труда и техники безопасности и ознакомлены с инструкциями по эксплуатации оборудования.
* В настройках монитора необходимо установить комфортный для глаз режим мерцания.
* На мониторе не должны появляться блики во время использования.
* Расстояние от глаз до монитора должно быть 40-75 см.
* В темное время суток нужно работать за компьютером при включенном свете.
* Если ноги не достают до пола, необходимо установить под них комфортную подставку.
* Для разгрузки кистей рук, предусмотрены специализированные подставки под клавиатуру. Чтобы руки не уставали, их необходимо держать над клавиатурой.
* Каждые 45 минут необходимо делать перерыв.
* Для дополнительного комфорта глаз можно использовать специальные очки для работы за компьютером.

Структура отдела:

Отдел разработки, разработка и настройка ПО

Структура предприятия:

* Специалист-стажер,
* Менеджер,
* Начальник.

Распорядок дня:

* С 9:00 до 13:00 – рабочее время;
* С 13:00 до 14:00 – обеденное время;
* С 14:00 до 18:00 – рабочее время;

Обязанности IT-Специалиста:

* Установка, настройка программного обеспечения;
* Мониторинг и управление сетевой инфраструктурой, включая маршрутизаторы, коммутаторы и другие сетевые устройства;
* Оказание консультационных услуг для пользователей;
* Организацию резервного копирования и восстановления данных, а также обеспечение их целостности, конфиденциальности и доступности;
* Обеспечение безопасности компьютерных систем и сетей, установку и настройку антивирусного и брандмауэрного ПО;
* Обучение пользователей использованию компьютерных систем и приложений, предоставление технической поддержки и решение проблем, связанных с компьютерами и программным обеспечением;

**2 СВЕДЕНИЯ ОБ ИМЕЮЩЕЙСЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ И ПРОГРАММНОМ ОБЕСПЕЧЕНИИ**

В отделе установлены 3 рабочих компьютера, 3 рабочих ноутбука, а так же принтер, 3D-принтер и маршрутизатор:

1. Характеристики первого компьютера:
   1. Материнская плата: GIGABYTE B360M HD3;
   2. Процессор: Intel Core i3-9100F CPU 3.60 GHz;
   3. Видео карта: NVIDIA GeForce GTX 1650;
   4. Оперативная память (RAM): 16 ГБ;
   5. Жесткий диск (HDD): 250 ГБ;
   6. Жесткий диск (HDD): 1 ТБ;
   7. Жесткий диск (HDD): 1 ТБ;
2. Характеристики второго компьютера:
   1. Материнская плата GIGABYTE H310M-A;
   2. Процессор: Intel Core i3-8100 CPU 3.60 GHz
   3. Видео карта: NVIDIA GeForce GT 710;
   4. Оперативная память (RAM): 8 ГБ;
   5. Жесткий диск (HDD): 500ГБ;
3. Характеристики третьего компьютера:
   1. Материнская плата: MSI B460M-A PRO;
   2. Процессор: Intel Core i3-10100F CPU 3.60 GHz;
   3. Видео карта: Power Color AX5670 512MD5-HV2;
   4. Оперативная память (RAM): 16 ГБ;
   5. Жесткий диск (HDD): 1 ТБ;
   6. Жесткий диск (HDD): 1 ТБ;
   7. Твердотельный накопитель (SSD): 512 ГБ;
4. Характеристики первого ноутбука:
   1. Материнская плата: ASUS N56VB;
   2. Процессор: Intel Core i5-3230M CPU 2.60 GHz;
   3. Видео карта: NVIDIA GeForce GT740M;
   4. Оперативная память (RAM): 6 ГБ;
   5. Жёсткий диск (HDD): 500 ГБ;
5. Характеристики второго ноутбука:
   1. Материнская плата: AMD M880G;
   2. Процессор: AMD Athlon II M320 2.1 GHz;
   3. Видео карта: Mobility Radeon HD 4330;
   4. Оперативная память (RAM): 3 ГБ;
   5. Жёсткий диск (HDD): 320 ГБ;
6. Характеристика третьего ноутбука:
   1. Материнская плата:ASUS X401A;
   2. Процессор: Intel Pentium B970;
   3. Видео карта: NVIDIA GeForcee GT630M;
   4. Оперативная память (RAM): 8 ГБ;
   5. Жёсткий диск (HDD): 320 ГБ;
   6. Твердотельный накопитель (SSD): 230 ГБ;
7. Принтер – HP Deskjet 3070A
8. 3D-Принтер – WANHAO Duplicator I3
9. Роутер – TP-link AC1900

Компьютеры объединены в локальную сеть и имеют выход в Internet.

Программное обеспечение рабочего места:

* ОС Windows 10 Home;
* Microsoft Office 2019;
* Git 2.32;
* .NET Framework Developer Pack 4.8;
* Microsoft Visual Studio 2019 Community;
* Yandex браузер;
* Any desk;
* Virtual Box;

**3 СВЕДЕНИЯ О ПРОХОЖДЕНИИ ПРАКТИКИ В КАЧЕСТВЕ ДУБЛЕРА ИНЖЕНЕРНО-ТЕХНИЧЕСКОГО РАБОТНИКА**

Во время прохождения практики в качестве дублера программиста я выполнял следующие действия:

* ознакомление с системным и прикладным программным обеспечением на предприятии;
* работа с документацией;
* сбор и структурирование материала по теме индивидуального задания «Учёт ремонтных комплектующих»;
* разработка интерфейса приложения;
* анализ алгоритмов, в том числе с применением инструментальных средств, осуществление рефакторинга и оптимизации программного кода;
* анализ проектной и технической документации на уровне взаимодействия компонент программного обеспечения;
* интеграция модулей в программную систему;
* отладка программного продукта с использованием спе­циализированных программных средств;
* разработка тестовых наборов и тестовых сценариев;
* инспектирование компонент программного продукта на предмет соответствия стандартам кодирования;
* тестирование программного продукта;
* разработка технической документации к программному продукту;
* написание программного кода, реализующего функциональность разрабатываемого приложения;
* установка и настройка необходимого программного обеспечения;
* разработка кода по заданию руководителя практики.

Выполненная работа соответствует профессиональным компетенциям, предусмотренным программой производственной практики ПМ.02 специальности 09.02.07 Информационные системы и программирование.

**4 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**4.1 Постановка задачи**

В период практики необходимо написать desktop-приложение реализующее учёт ремонтных комплектующих для ввода и вывода информации о складе и ремонтах для компьютерных платформ. Программа должна содержать текстовую и графическую информацию, достаточную для исследования особенностей данной функции, иметь удобный и понятный пользователю графический интерфейс, быть проста в использовании.

Программа должна предлагать следующие возможности:

* ввода и вывода информации о комплектующих для ремонта;
* добавлять информацию о комплектующих для ремонта;
* редактировать информацию о комплектующих для ремонта
* удалять информацию о комплектующих для ремонта
* выдавать справочную информацию;
* сохранять результаты в базу данных.

Входными данными будут:

* выбранная таблица
* значение из таблицы;

Выходными данными будут:

* визуальное отображение графика функции;
* информация, полученная при исследовании функции;
* пользовательская справочная система.

**4.2 Проектирование структуры программы**

**4.2.1 Desktop версия приложения**

С целью удостоверения предъявляемых к программе требований, разрабатываемое приложение будет состоять из следующих окон:

* загрузки – будет запускаться перед отображением главного окна;
* авторизация – будет содержать блоки для ввода информации и кнопку входа в приложение;
* главная – окно для просмотра/добавления информации о детях, воспитателях, расписании и родителях;
* добавление – окно, где можно добавить информацию о новом ребёнке, воспитателе, расписании или родителе;
* справка – будет выводить справочный материал для пользователя.

Переход к блоку «Добавление» будет происходить при выборе пункта «Добавить» в главном окне.

Материалы будут храниться в БД. Проектируемая структура приложения представлена на рисунке 1.

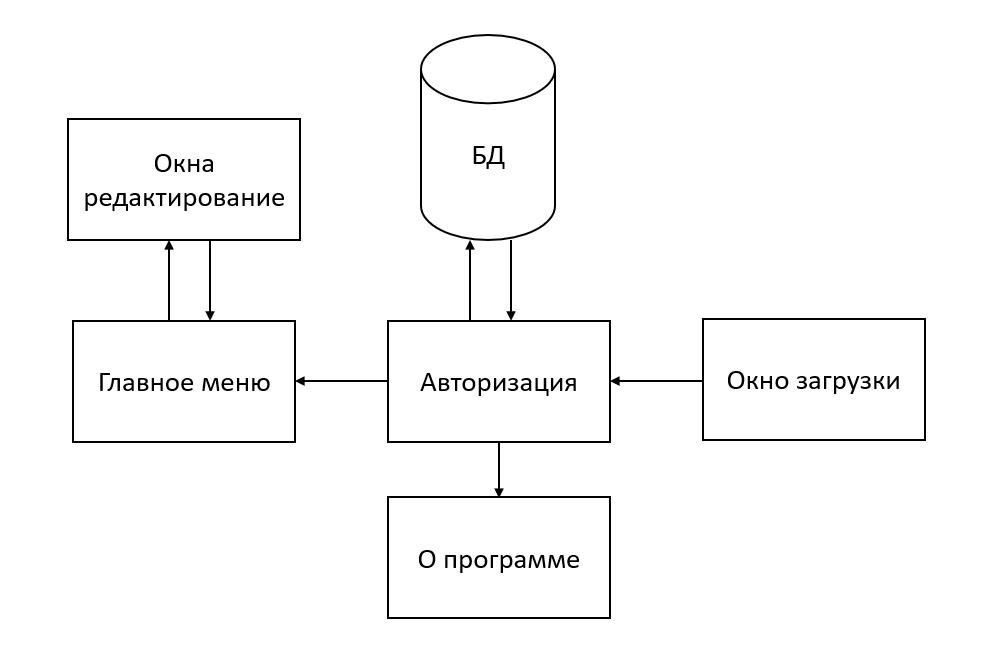


Рисунок 1 – Схема проектируемой структуры приложения

**4.3 Проектирование базы данных**

Для хранения информации была разработана БД. Логическая модель БД представлена на рисунке 2.

Физическая модель данных описывает то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Также описываются типы, идентификаторы и разрядность полей.

На основе логической модели были создана физическая модель БД, которая хранится в компьютере в файле AccountingOfResonantComponentsDBData.sql (таблицы 1-4).

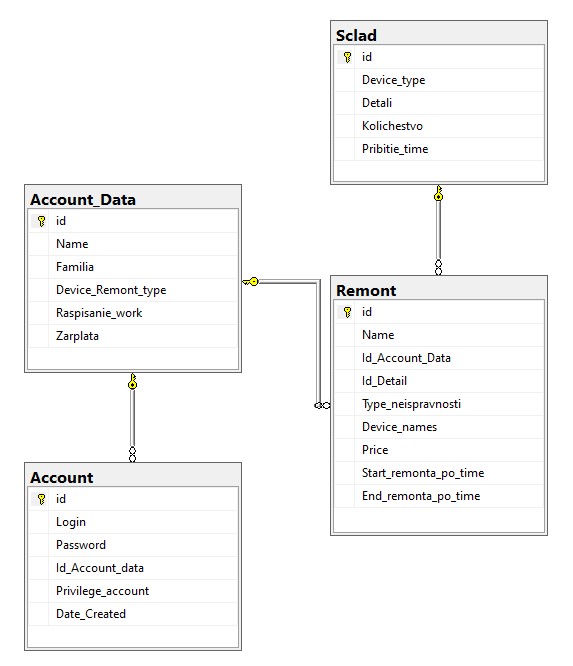


Рисунок 2 – Логическая модель БД

Таблица 1 – Физическая модель таблицы «Account»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| Id | Уникальный номер | int | Первичный |
| Login | Логин | nvarchar(255) | - |
| Password | Пароль | nvarchar(255) | - |
| Id\_Account\_data | Идентификатор данных  Аккаунта(Работник) | int | Внешний |
| Privilege\_account | Должность | int | - |
| Date\_Created | Дата Создания | int | - |

Таблица 2 – Физическая модель таблицы «Account\_Data»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| Id | Уникальный номер | int | Первичный |
| Name | Имя | nvarchar(255) | - |
| Familia | Фамилия | nvarchar(255) | - |
| Device\_Remont\_type | Тип устройства для ремонта | nvarchar(255) | - |
| Raspisanie\_work | Расписания работы | nvarchar(255) | - |
| Zarplata | Зарплата | int | - |

Таблица 3 – Физическая модель таблицы «Remont»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| Id | Уникальный номер | int | Первичный |
| Name | Имя | nvarchar(255) | - |
| Id\_Account\_Data | Идентификатор данных  Аккаунта(Работник) | int | Внешний |
| Id\_Detail | Идентификатор детали на складе | int | Внешний |
| Type\_neispravnosti | Расписания работы | nvarchar(255) | - |
| Device\_names | Название устройство | nvarchar(255) | - |
| Price | Стоимость | int | - |
| Start\_remonta\_po\_time | Начало ремонта | Data type | - |
| End\_remonta\_po\_time | Конец ремонта | Data type | - |

Таблица 4 – Физическая модель таблицы «Sclad»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| Id | Уникальный номер | int | Первичный |
| Device\_type | Тип устройства | nvarchar(255) | - |
| Detali | Название детали | nvarchar(255) | - |
| Kolichestvo | Количкство | int | - |
| Pribitie\_time | Время прибытия | Data type | - |

**4.4 Функциональное взаимодействие модулей программы**

**4.4.1 Desktop версия приложения**

На основе разработанного технического проекта был написан программный продукт «Учёт ремонтных комплектующих». Для создания приложения была использована среда программирования Visual Studio 2022 и язык программирования С#. В программе используются следующие языковые модули:

* System.Windows.Controls − предоставляет классы для создания элементов, известных как элементы управления, позволяющих пользователю взаимодействовать с приложением;
* System − содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Windows − содержит типы, используемые в приложениях WindowsPresentationFoundation (WPF), включая клиенты анимации, элементы управления пользовательского интерфейса, привязку данных и преобразование типов;
* System.Data − содержит типы для доступа к данным из различных источников и для управления этими данными. Пространство имен верхнего уровня и несколько дочерних пространств имен образуют архитектуру ADO.NET и поставщиков данных ADO.NET [9].
* System.Text - Содержит классы, которые представляют кодировки ASCII и Юникода; абстрактные базовые классы для преобразования блоков знаков в блоки байтов и обратно; вспомогательный класс, который обрабатывает и форматирует объекты [String](https://learn.microsoft.com/ru-ru/dotnet/api/system.string?view=net-8.0), не создавая промежуточные экземпляры [String](https://learn.microsoft.com/ru-ru/dotnet/api/system.string?view=net-8.0).
* System.Collections.Generic – работа с обобщёнными коллекциями (List<string>) [5].

В программе были созданы следующие пользовательские модули:

* SplashScreen.cs (окно заставки) – заставка приложения;
* MainMenu.cs (главное окно программы) − обеспечивает взаимодействовать с программой, и позволяет вам редактировать данные.
* Autorization.cs – система авторизации, которая поможет обеспечить безопасность приложению.
* Handbook.cs (окно о программе) – этот модуль содержит сведения о разработчике программы.

В модуле MainMenu.cs описаны и используются следующие процедуры:

* private void adminComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e) − переключает страницы в главном окне
* private void Drag(object sender, RoutedEventArgs e) – позволяет передвигать окно по экрану
* private void DeleteButton\_Click(object sender, RoutedEventArgs e) − удаляет выбранную строку в выбранную таблицы
* private void InsertButton\_Click(object sender, RoutedEventArgs e) – добавляет строку в выбранную таблицу
* private void EditButton\_Click(object sender, RoutedEventArgs e) – редактирует выбранную строчку в выбранную таблицы

Код данных процедур представлен в приложении А.

**4.5 Работа с приложением**

**4.5.1 Desktop версия приложения**

При запуске программы открывается заставка (рисунок 3).

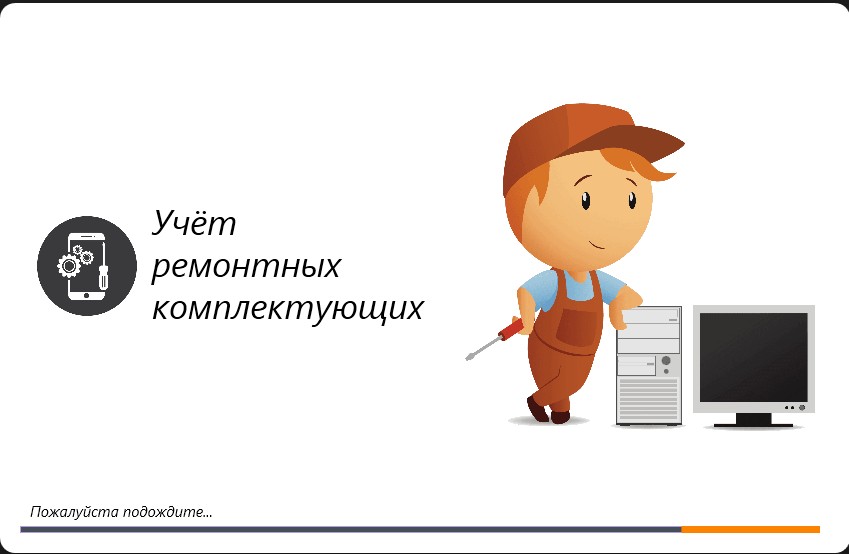


Рисунок 3 – Заставка

После завершения загрузки, откроется окно авторизации программы.

Интерфейс авторизации имеет следующие элементы (рисунок 4):

* область ввода логина и пароля (1);
* кнопка вызова справки (2);
* кнопка, закрывающая приложение (3);
* кнопка, сворачивания приложение (4);

Если у пользователя возникнуть вопросы по работе с программой, он может вызвать справочное окно, нажав на кнопку «Вопрос»(рисунок 4).



Рисунок 4 – Основное окно программы

При нажатии на кнопку «Вопрос» в окне авторизации, откроется окно просмотра информации о создателе программы (рисунок 6).

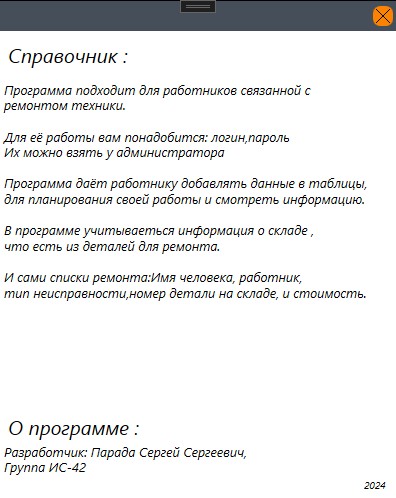


Рисунок 5 – Окно справки и о программе

Интерфейс программы при входе администратора имеет следующие элементы (рисунок 6):

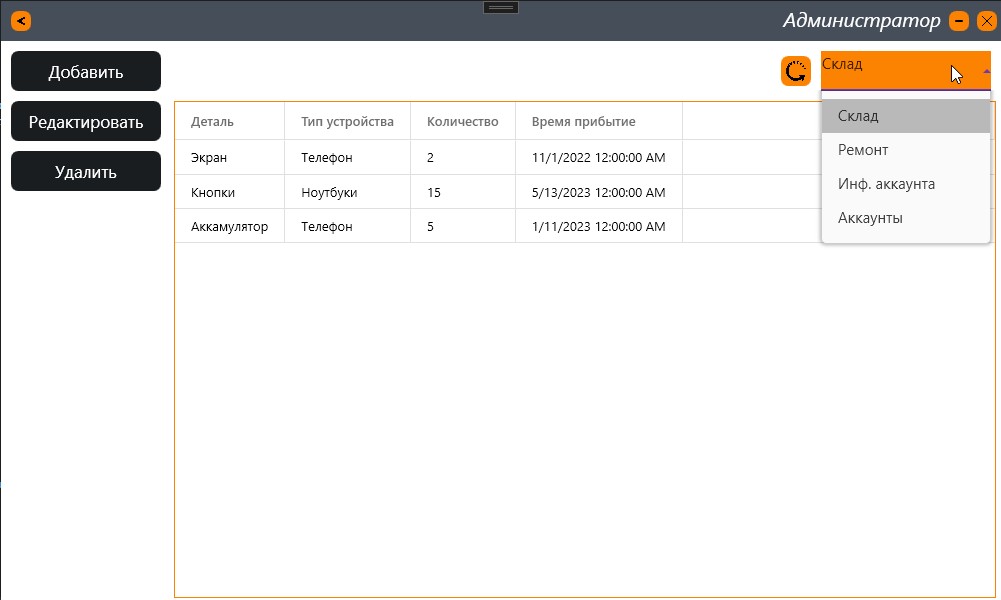
* область вывода таблиц (1);
* кнопка, перехода на авторизацию (2);
* кнопка, сворачивания окна (3)
* кнопка, закрывающая приложение (4);
* сворачиваемое меню для выбора таблицы(5);
* кнопка, добавить строчку в таблицу(6);
* кнопка, редактировать строчку в таблицу(7);
* кнопка, удаления строчку в таблицу(8);

Рисунок 6 – Главное окно администратора

Интерфейс программы при входе сотрудника имеет следующие элементы (рисунок 7):

* область вывода таблиц (1);
* кнопка, перехода на авторизацию (2);
* кнопка, сворачивания окна (3)
* кнопка, закрывающая приложение (4);
* сворачиваемое меню для выбора таблицы(5);
* кнопка, добавить строчку в таблицу(6);

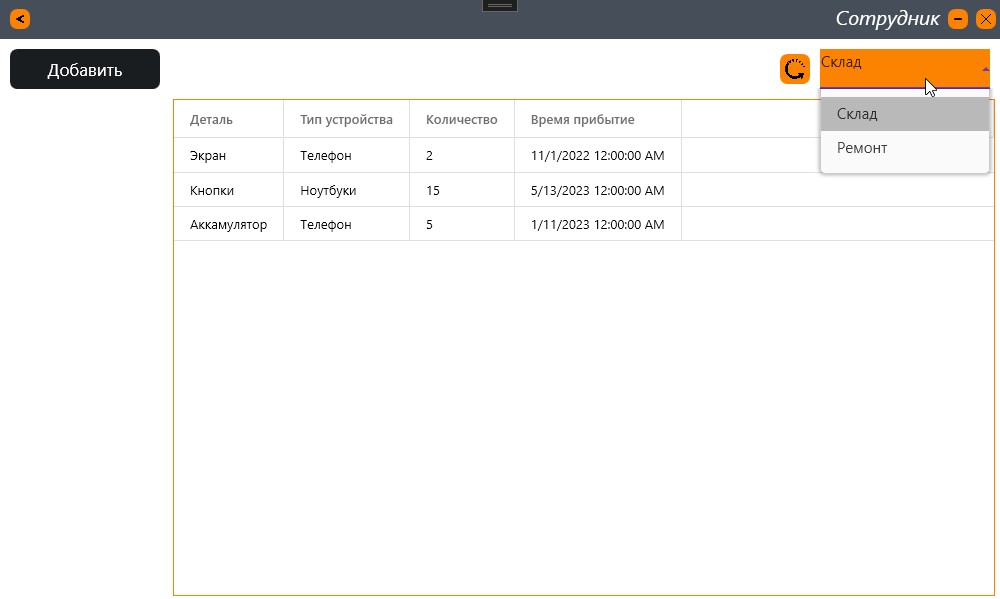


Рисунок 7 – Главное окно сотрудника

Интерфейс программы при добавлении или редактировании для таблицы склад имеет следующие элементы (рисунок 8):

* кнопка, сворачивания окна (1)
* кнопка, закрывающая приложение (2);
* поля ввода информации(3);
* кнопка, сохранения информации в таблицу(4)

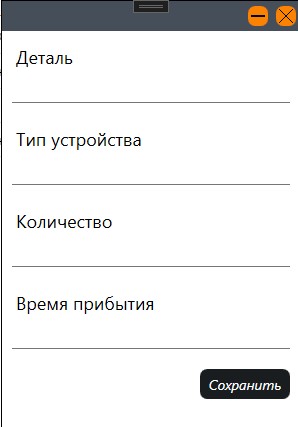


Рисунок 8 – Добавления или редактирования таблицы склад

Интерфейс программы при добавлении или редактировании для таблицы ремонт имеет следующие элементы (рисунок 9):

* кнопка, сворачивания окна (1)
* кнопка, закрывающая приложение (2);
* поля ввода информации(3);
* кнопка, сохранения информации в таблицу(4)

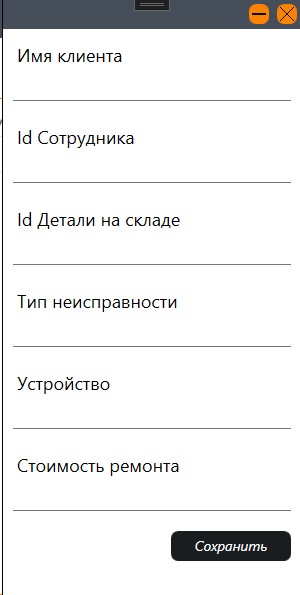


Рисунок 9 – Добавления или редактирования таблицы ремонт

Интерфейс программы при добавлении или редактировании для таблицы Склад имеет следующие элементы (рисунок 10):

* кнопка, сворачивания окна (1)
* кнопка, закрывающая приложение (2);
* поля ввода информации(3);
* кнопка, сохранения информации в таблицу(4)

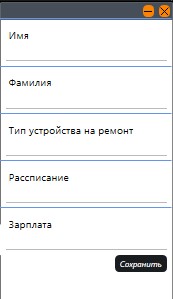


Рисунок 10 – Добавления или редактирования таблицы информация сотрудника

Интерфейс программы при добавлении или редактировании для таблицы Склад имеет следующие элементы (рисунок 11):

* кнопка, сворачивания окна (1)
* кнопка, закрывающая приложение (2);
* поля ввода информации(3);
* кнопка, сохранения информации в таблицу(4)

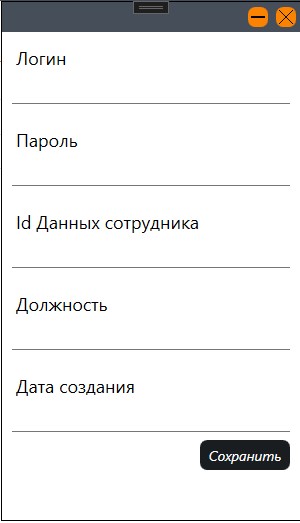


Рисунок 11 – Добавления или редактирования таблицы информация аккаунты

**ЗАКЛЮЧЕНИЕ**

В результате прохождения производственной практики мною были закреплены теоретические знания, полученные при изучении ПМ.02; отработаны профессиональные компетенции по получаемой специальности; приобретены практические навыки в будущей профессиональной деятельности; выполнено индивидуальное задание.

Мною было разработано приложение для учёта ремонтных комплектующих Программа имеет удобный и понятный пользователю графический интерфейс, проста в использовании и обеспечивает следующие возможности:

* ввода и вывода информации о комплектующих для ремонта;
* добавлять информацию о комплектующих для ремонта;
* редактировать информацию о комплектующих для ремонта
* удалять информацию о комплектующих для ремонта
* выдавать справочную информацию;
* сохранять результаты в базу данных.

В ходе выполнения разработки данного приложения были:

* проанализированы похожие приложения;
* спроектирована структура приложения и программного кода;
* разработаны справочные материалы;
* выбраны средства реализации приложения;
* разработан программный код приложения;

Основными достоинствами программы являются:

* программа проста в установке, настройке и эксплуатации;
* данный программный продукт предназначен для широкого круга пользователей;
* наличие справочной системы.

К недостаткам данной программы можно отнести то, что она предназначена для предприятия.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Хокинг, Д.«Unity в действии: разработка мультиплатформенных игр на C#», Джо Хокинг(дата посищение:16.01.2024)
2. Ферроне ,Х.«Изучение C# путем разработки игр на Unity», Харрисон Ферроне(дата посищение:21.01.2024)
3. «Освоение разработки 2D-игр на Unity», Саймон Джексон. (дата посищение:18.01.2024)
4. Шарп, Д. «Microsoft Visual C# шаг за шагом», Джон Шарп. (дата посищение:17.01.2024)
5. Гейг, М. «Разработка игр на Unity за 24 часа», Майк Гейг. (дата посищение:19.12.2023)
6. Ферроне, Х. «Разработка игр на C# для профессионалов Unity», Харрисон Ферроне(дата посищение:14.01.2024)
7. Microsoft Developer Network (MSDN): Официальный сайт Microsoft для разработчиков, содержащий документацию по C# и .NET Framework. (дата посищение:19.12.2023). Ссылка: <https://docs.microsoft.com/en-us/dotnet/csharp/>МэйнС. Основы Windows Communication Foundation для .NET Framework 3.5. Санкт-Петербург.: БХВ-Петербург, 2015. (дата посищение:22.01.2024)
8. Stack Overflow: Онлайн-сообщество, где разработчики могут задавать вопросы и получать ответы от других опытных программистов. Здесь вы можете найти различные вопросы и ответы на темы, связанные с C# и Unity.  
   Ссылка: <https://stackoverflow.com/> (дата посищение:18.01.2024)

**ПРИЛОЖЕНИЕ А  
(справочное)**

**Основные процедуры desktop приложения**

**Класс MainMenu.cs**

using AccountingOfResonantComponent.ViewModel;

using AccountingOfResonantComponents.DbEntity;

using AccountingOfResonantComponents.View.Page;

using AccountingOfResonantComponents.View.Window;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Navigation;

namespace AccountingOfResonantComponent.View.Window

{

public partial class MainMenu

{

public int role = 1;

private Account\_Data \_accountEmployees;

public static MainMenu mainMenu;

private readonly ScladMV \_scladPageVM;

private readonly RemontMV \_remontMV;

private readonly AccountEmployeesMV \_accountEmployeesVM;

private readonly AccountDataMV \_accountDataMV;

public Account\_Data AccountEmployees

{

get => \_accountEmployees;

set

{

\_accountEmployees = value;

}

}

public MainMenu(Account accountEmployees, int role)

{

\_scladPageVM = new ScladMV();

\_remontMV = new RemontMV();

\_accountEmployeesVM = new AccountEmployeesMV();

\_accountDataMV = new AccountDataMV();

InitializeComponent();

FramePageAdmin.NavigationUIVisibility = NavigationUIVisibility.Hidden;

mainMenu = this;

TableComboAdmin.SelectedIndex = 0;

if (role == 1)

{

TableComboAdmin.ItemsSource = new Table[]

{

new Table { Name = "Склад"},

new Table { Name = "Ремонт"},

new Table { Name = "Инф. аккаунта"},

new Table { Name = "Аккаунты"},

};

DeleteButton.Visibility = Visibility.Visible;

EditButton.Visibility = Visibility.Visible;

InsertButton.Visibility = Visibility.Visible;

Position.Content = "Администратор";

}

if (role == 2)

{

TableComboAdmin.ItemsSource = new Table[]

{

new Table { Name = "Склад"},

new Table { Name = "Ремонт"},

};

InsertButton.Visibility = Visibility.Visible;

Position.Content = "Сотрудник";

}

}

private void Drag(object sender, RoutedEventArgs e)

{

if (Mouse.LeftButton == MouseButtonState.Pressed)

{

MainMenu.mainMenu.DragMove();

}

}

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

{

this.Close();

}

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

{

this.WindowState = WindowState.Minimized;

}

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

{

Autorization autorization = new Autorization();

autorization.Show();

this.Close();

}

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

{

switch (TableComboAdmin.SelectedIndex)

{

case 0:

FramePageAdmin.Navigate(new ScladPage { DataContext = \_scladPageVM });

break;

case 1:

FramePageAdmin.Navigate(new RemontPage { DataContext = \_remontMV });

break;

case 2:

FramePageAdmin.Navigate(new AccountDataPage { DataContext = \_accountDataMV });

break;

case 3:

FramePageAdmin.Navigate(new AccountEmployeesPage { DataContext = \_accountEmployeesVM });

break;

default:

break;

}

}

public void RefrashDataTable()

{

(DataContext as ScladMV).LoadData();

}

public class Table

{

public string Name { get; set; } = "";

public override string ToString() => $"{Name}";

}

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

{

switch (TableComboAdmin.SelectedIndex)

{

case 0:

\_scladPageVM.DeleteData();

break;

case 1:

\_remontMV.DeleteData();

break;

case 2:

\_accountDataMV.DeleteData();

break;

case 3:

\_accountEmployeesVM.DeleteData();

break;

default:

break;

}

}

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

{

switch (TableComboAdmin.SelectedIndex)

{

case 0:

{

var changeTableSclad = new ChangeTableSclad(null);

changeTableSclad.Show();

}

break;

case 1:

{

var changeTableRemont = new ChangeTableRemont(null);

changeTableRemont.Show();

}

break;

case 2:

{

var changeInfoEmployees = new ChangeAccountData(null);

changeInfoEmployees.Show();

}

break;

case 3:

{

var changeAccount = new ChangeAccount(null);

changeAccount.Show();

}

break;

default:

break;

}

}

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

{

switch (TableComboAdmin.SelectedIndex)

{

case 0:

{

var changeTableSclad = new ChangeTableSclad(\_scladPageVM.SelectSclad);

changeTableSclad.Show();

}

break;

case 1:

{

var changeTableRemont = new ChangeTableRemont(\_remontMV.SelectRemont);

changeTableRemont.Show();

}

break;

case 2:

{

var changeInfoEmployees = new ChangeAccountData(\_accountDataMV.SelectAccountData);

changeInfoEmployees.Show();

}

break;

case 3:

{

var changeAccount = new ChangeAccount(\_accountEmployeesVM.SelectAccountEmployees);

changeAccount.Show();

}

break;

default:

break;

}

}

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

{

switch (TableComboAdmin.SelectedIndex)

{

case 0:

\_scladPageVM.LoadData();

break;

case 1:

\_remontMV.LoadData();

break;

case 2:

\_accountDataMV.LoadData();

break;

case 3:

\_accountEmployeesVM.LoadData();

break;

default:

break;

}

}

}

}