**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Чаликян Арен Оганнесович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «9» декабря 2024 г. по «21» декабря 2024 г.

Руководитель практики Шаврова Л. С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Чаликян Арен Оганнесович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «9» декабря 2024 г. по «21» декабря 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения;
* Разработка модулей программного обеспечения;
* Тестирование программных модулей и их интеграции;
* Разработка программной документации и стандарты кодирования.

**Индивидуальное задание: ВАРИАНТ 1**

Задание выдал «9» декабря 2024 г. \_\_\_\_\_\_\_\_\_\_ Шаврова Л.С.

(подпись) (Ф.И.О.)

Задание получил «9» декабря 2024 г. \_\_\_\_\_\_\_\_\_\_ Чаликян А.О.(подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Чаликян Арен Оганнесович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «9» декабря 2024 г. по «21» декабря 2024 г.

Руководитель практики Шаврова Л.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 09.12 | Анализ предметной области. Выявление требований к программе. |  |
| 10.12 | Разработка технического задания. UML. Проектирование диаграммы вариантов использования. |  |
| 11.12 | UML. Проектирование диаграммы последовательности. UML. Проектирование диаграммы активности. |  |
| 12.12 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 13.12 | Проектирование интерфейса пользователя. Создание Wireframe эскизов. Разработка дизайна программы в соответствии с руководством по стилю. |  |
| 14.12 | Разработка базы данных. Разработка словаря данных. |  |
| 15.12 | Создание приложения. Форма авторизации. Создание приложения. Форма заказов. |  |
| 17.12 | Создание приложения. Основные формы приложения. Разработка библиотеки классов. |  |
| 18.12 | Подготовка отчетов и выгрузка документов для печати. Отладка программных модулей. |  |
| 19.12 | Модульное тестирование. Создание тестовых случаев.автом |  |
| 20.12 | Интеграционное тестирование. Разработка самодокументирующегося кода. |  |
| 21.12 | Инспекция кода на соответствие стандартам кодирования. Подготовка отчета и размещение результатов в репозитории контроля версий. |  |

**Содержание**

[ВВЕДЕНИЕ 7](#_Toc185373917)

[1 Общая часть 8](#_Toc185373918)

[1.1 Проектирование системы 8](#_Toc185373919)

[1.1.1 Краткая спецификация 8](#_Toc185373920)

[1.1.2 Диаграмма вариантов использования 11](#_Toc185373921)

[1.1.3 Диаграмма активности 12](#_Toc185373922)

[1.1.4 Диаграмма последовательности 13](#_Toc185373923)

[1.1.5 Общий алгоритм 14](#_Toc185373924)

[1.1.6 Алгоритм одной из функций 15](#_Toc185373925)

[1.1.7 Разработка макетов интерфейса системы 16](#_Toc185373926)

[1.2 Создание базы данных и заполнение таблиц данными 22](#_Toc185373927)

[1.2.1 ER-диаграмма 22](#_Toc185373928)

[1.2.2 Словарь данных 23](#_Toc185373929)

[1.2.3 Заполненные данными таблицы 27](#_Toc185373930)

[1.2.4 Резервное копирование 29](#_Toc185373931)

[1.3 Разработка библиотеки и подключение её к проекту 29](#_Toc185373932)

[1.4 Разработка приложения 31](#_Toc185373933)

[1.5 Тестирование приложения 50](#_Toc185373934)

[1.5.1 Создание тестовых случаев 50](#_Toc185373935)

[1.5.2 Модульное тестирование 50](#_Toc185373936)

[1.6 Выгрузка готового проекта в репозиторий GIT 52](#_Toc185373937)

[1.7 Качественные характеристики кода 53](#_Toc185373938)

[1.7.1 Полнота обработки ошибочных данных 53](#_Toc185373939)

[1.7.2 Наличие тестов для проверки допустимых значений входных данных 53](#_Toc185373940)

[1.7.3 Наличие средств контроля корректности входных данных 55](#_Toc185373941)

[1.7.4 Наличие средств восстановления при сбоях оборудования 55](#_Toc185373942)

[1.7.5 Наличие комментариев 55](#_Toc185373943)

[1.7.6 Наличие проверки корректности передаваемых данных 55](#_Toc185373944)

[1.7.7 Наличие описания основных функций 56](#_Toc185373945)

[ЗАКЛЮЧЕНИЕ 57](#_Toc185373946)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 58](#_Toc185373947)

[Приложение А (справочное) Скрипт БД 59](#_Toc185373948)

[Приложение Б (справочное) Библиотека классов 61](#_Toc185373949)

[Приложение В (справочное) Исходный код 63](#_Toc185373950)

[Приложение Г (справочное) Данные тестирования 110](#_Toc185373951)

[Приложение Д (справочное) Код unit-тестов 116](#_Toc185373952)

# **ВВЕДЕНИЕ**

Активное применение компьютерных технологий для хранения и передачи информации на станциях технического обслуживания способствует повышению эффективности учета заявок, сокращению времени их обработки, оптимизации использования ресурсов и улучшению удовлетворенности клиентов.

Разработка программного модуля для управления заявками на ремонт климатического оборудования включает создание программного обеспечения, которое позволит сервисам более эффективно отслеживать и управлять заявками.

Основная цель создания данного модуля заключается в автоматизации учета и обработки заявок на ремонт, что обеспечит сотрудникам сервисов удобное и прозрачное управление процессом, контроль статусов ремонта и улучшение организации работы.

Задачи по разработке программного модуля включают:

* Анализ предметной области и разработка технического задания.
* Проектирование диаграмм вариантов использования, последовательности и активности.
* Моделирование структуры программного обеспечения.
* Проектирование пользовательского интерфейса.
* Разработка базы данных и словаря данных.
* Создание приложения, включая формы авторизации, заказов и основные формы приложения, а также разработка библиотеки классов.
* Отладка программных модулей.
* Модульное тестирование и создание тестовых случаев.
* Инспекция кода на соответствие стандартам кодирования.
* Подготовка отчета и размещение результатов в репозитории Git.

# **Общая часть**

## **Проектирование системы**

Проектирование системы играет важную роль в создании детального плана, который помогает эффективно организовать процесс разработки, определить ключевые этапы и установить сроки выполнения. Это также позволяет учесть правовые и технические аспекты проекта. Грамотно спланированная архитектура системы дает возможность команде заранее оценить объем работ и возможные сложности, что помогает избежать лишних исправлений и длительных согласований. Это способствует улучшению координации внутри команды и повышению общей удовлетворенности как со стороны заказчика, так и исполнителей, минимизируя риски на этапе реализации.

### **Краткая спецификация**

**А) Выполнить при разработке:**

Функции включают добавление заявок (с уникальным идентификатором), редактирование заявок, анализ скорости выполнения услуг, добавление отчета, назначение исполнителя для каждой заявки.

**Б) Обеспечить:**

Использование лицензированного ПО, доступ к системе должен быть защищен авторизацией пользователей, организация резервного копирования данных, требования к составу и параметрам технических средств.

**В) Составить техническую документацию и отчетность:**

- «Техническое задание»;

- разработка руководства системного программиста.

**Определение входных и выходных данных**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Фактический результат/Выходные данные** |
| Логин “login1”,  пароль “pass1” | Открытие страницы техника | Открытие страницы техника |
| Логин “login1”,  пароль “123” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login2”,  пароль “pass2” | Открытие страницы техника | Открытие страницы техника |
| Логин “login2”,  пароль “mbh” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login3”,  пароль “pass3” | Открытие страницы клиента | Открытие страницы клиента |
| Логин “login3”,  пароль “прм” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login4”,  пароль “pass4” | Открытие страницы клиента | Открытие страницы клиента |
| Логин “login4”,  пароль “222” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login5”,  пароль “pass5” | Открытие страницы клиента | Открытие страницы клиента |
| Логин “login5”,  пароль “345” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login6”,  пароль “pass6” | Открытие страницы клиента | Открытие страницы клиента |
| Логин “login6”,  пароль “675” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login7”,  пароль “pass7” | Открытие страницы клиента | Открытие страницы клиента |
| Логин “login7”,  пароль “456” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login8”,  пароль “pass8” | Открытие страницы оператора | Открытие страницы оператора |
| Логин “login8”,  пароль “896” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |
| Логин “login9”,  пароль “pass9” | Открытие страницы оператора | Открытие страницы оператора |
| Логин “login10”,  пароль “222” | Открытие уведомления “Неверно введен пароль” | Открытие уведомления “Неверно введен пароль” |

### **Диаграмма вариантов использования**

На рисунке 1 представлена диаграмма вариантов использования, которая отражает основной функционал пользователей программного обеспечения по ремонту бытовой техники.



Рисунок 1 – Диаграмма вариантов использования

### **Диаграмма активности**

На рисунке 2 представлена диаграмма активности, которая отображает динамические аспекты поведения системы. На диаграмме представлен процесс проверки данных заявки.



Рисунок 2 – Диаграмма активности

### **Диаграмма последовательности**

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования. На рисунке 3 представлена диаграмма, которая отражает процесс работы мастера над заявкой.



Рисунок 3 – Диаграмма последовательности

### **Общий алгоритм**

Общий алгоритм описывает те действия, которые обязательно произойдут при запуске программы. Общий алгоритм данной системы представлен на рисунке 4.



Рисунок 4 – Общий алгоритм

### **Алгоритм одной из функций**

На рисунке 5 представлен подробный алгоритм одной из функций данного приложения, а именно расчета среднего времени ремонта.



Рисунок 5 – Алгоритм функции расчета среднего времени ремонта

### **Разработка макетов интерфейса системы**

Создание макетов — это процесс, в котором дизайнеры, верстальщики и разработчики формируют элементы, необходимые для работы графических интерфейсов. Работа обычно начинается с разработки макета, представляющего собой статичное изображение или набор элементов, запланированных для использования в приложении.

Хорошо продуманные макеты ускоряют разработку интерфейса, обеспечивая ясное взаимодействие между командой и заказчиком. Такой макет позволяет представить, как будет выглядеть приложение, ещё до начала его создания.

Макеты и мокапы интерфейса, выполненные в графических редакторах, представлены на рисунках 6–15.

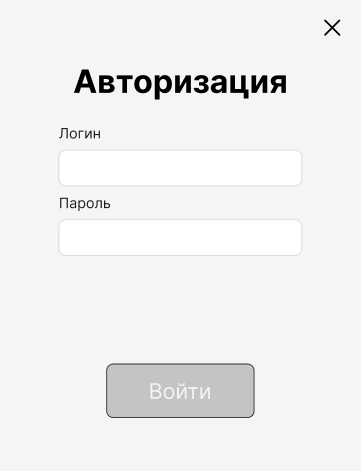


Рисунок 6 – Макет формы авторизации

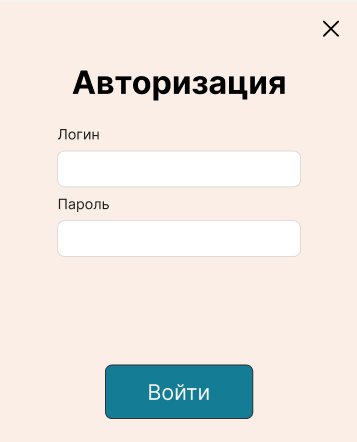


Рисунок 7 – Мокап формы авторизации

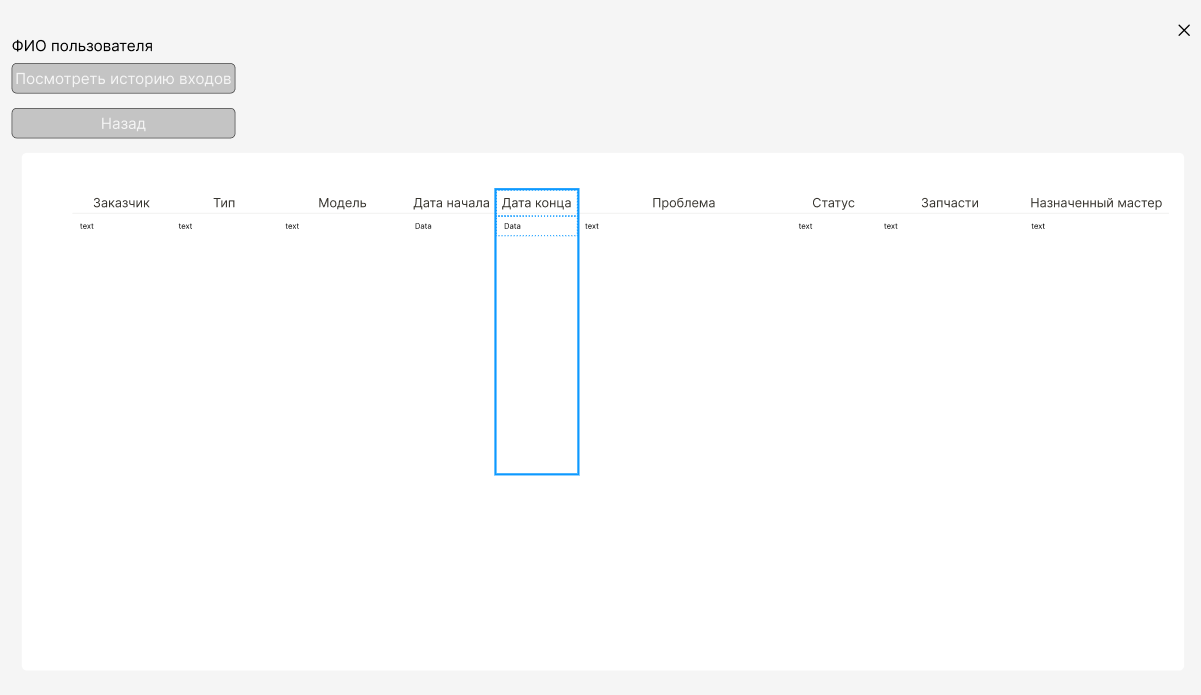


Рисунок 8 – Макет формы оператора

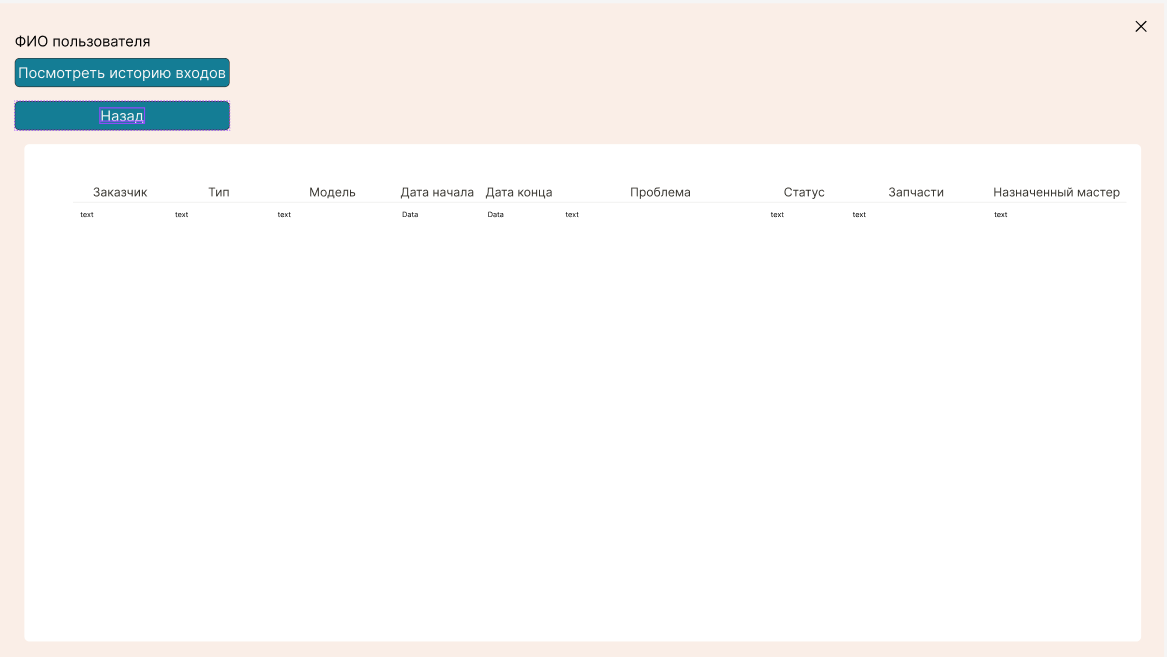


Рисунок 9 – Мокап формы оператора



Рисунок 10 – Макет формы создания заявки заказчиком

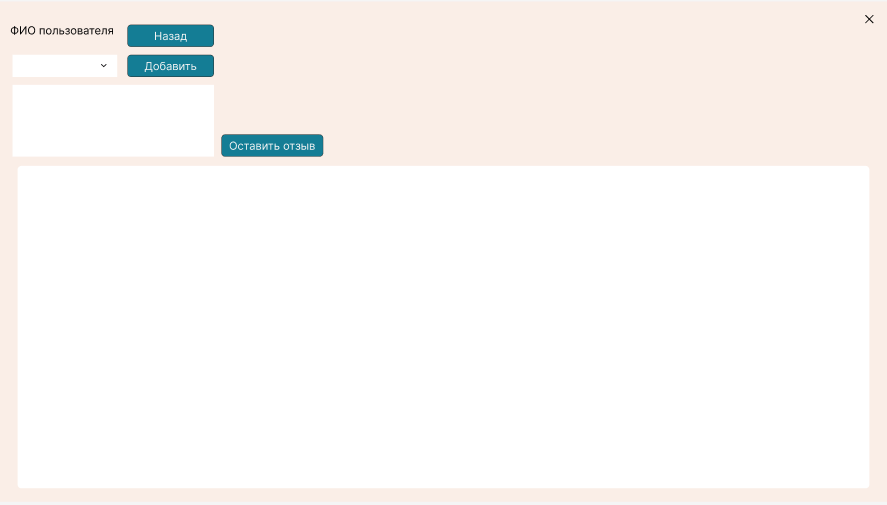


Рисунок 11 – Мокап формы создания заявки заказчиком

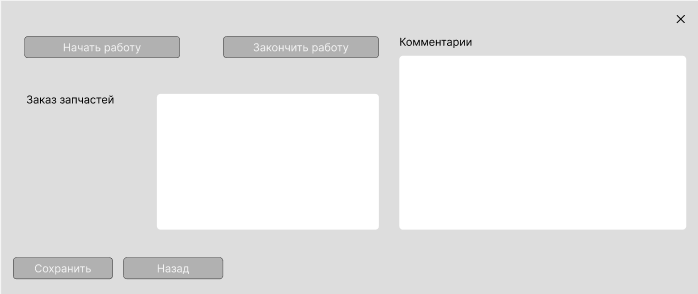


Рисунок 12 – Макет формы для работы мастера

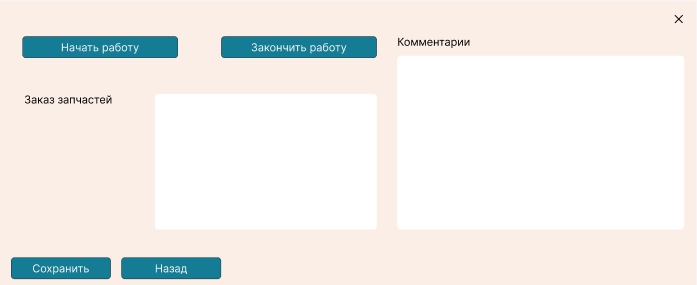


Рисунок 13 – Мокап формы для работы мастера



Рисунок 14 – Макет формы назначения мастера

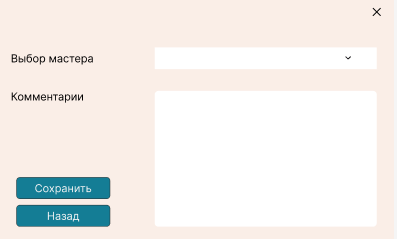


Рисунок 15 – Мокап формы назначения мастера

## **Создание базы данных и заполнение таблиц данными**

База данных – это совокупность структурированных и взаимосвязанных данных и методов, обеспечивающих добавление выборку и отображение данных.

Скрипт таблиц базы данных представлен в приложении А.

### **ER-диаграмма**

Перед разработкой ПО нужно определить, с какими данными предстоит работать и как они связаны между собой. Для этого системные аналитики строят модели данных и создают ER-диаграммы. Собрав все сущности будущего проекта, было выяснено, как они связаны между собой, и была составлена ER-модель. ER-модель данной программы представлена на рисунке 16.



Рисунок 16 – ER-диаграмма

### **Словарь данных**

Словарь данных — это набор реляционных таблиц и представлений, которые содержат информацию о таблицах или метаданные. Словарь данных содержит информацию об источниках, форматах и взаимосвязях между данными, их описания, сведения о характере использования и распределении ответственности. Словарь данных можно рассматривать как вспомогательную базу данных, в которой хранится информация об основной базе данных.

В таблице 1 приведен словарь данных для отношения «Comments».

Таблица 1 – Словарь данных таблицы «Comments»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEY** | **Имя поля** | **Тип данных / Размер** | **Необходимо?** | **Примечания** |
| PK | ID | INTEGER | Y | Auto Increment |
|  | message | NVARCHAR(255) |  |  |
| FK Staff | staffID | INTEGER |  | On delete(update)  cascade |
| FK Requst | requestID | INTEGER |  | On delete(update)  cascade |

В таблице 2 приведен словарь данных для отношения «Requests».

Таблица 2 – Словарь данных таблицы «Requests»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEY** | **Имя поля** | **Тип данных / Размер** | **Необходимо?** | **Примечания** |
| PK | ID | INTEGER | Y | Auto Increment |
|  | startDate | DATE |  |  |
|  | problemDescryption | NVARCHAR(255) | Y |  |
|  | completionDate | DATE |  |  |
|  | repairParts | NVARCHAR(255) |  |  |
| FK Staff | staffID | INTEGER |  | On delete(update)  cascade |
| FK Clients | clientID | INTEGER |  | On delete(update)  cascade |
| FK requestStatus | requestStatusID | INTEGER |  | On delete(update)  cascade |
| FK Technic | technicID | INTEGER |  | On delete(update)  cascade |

В таблице 3 приведен словарь данных для отношения «RequestStatus».

Таблица 3 – Словарь данных таблицы «RequestStatus»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEY** | **Имя поля** | **Тип данных / Размер** | **Необходимо?** | **Примечания** |
| PK | ID | INTEGER | Y | Auto Increment |
|  | requestStatus | NVARCHAR(255) |  |  |

В таблице 4 приведен словарь данных для отношения «Technic».

Таблица 4 – Словарь данных таблицы «Technic»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEY** | **Имя поля** | **Тип данных / Размер** | **Необходимо?** | **Примечания** |
| PK | ID | INTEGER | Y | Auto Increment |
|  | climateTechType | NVARCHAR(255) | Y |  |
|  | climateTechModel | NVARCHAR(255) | Y |  |

В таблице 5 приведен словарь данных для отношения «Users».

Таблица 5 – Словарь данных таблицы «Clients»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEY** | **Имя поля** | **Тип данных / Размер** | **Необходимо?** | **Примечания** |
| PK | ID | INTEGER | Y | Auto Increment |
|  | fio | NVARCHAR(255) | Y |  |
|  | phone | NVARCHAR(11) | Y |  |
|  | logins | NVARCHAR(30) | Y |  |
|  | passwords | NVARCHAR(30) | Y |  |

В таблице 6 приведен словарь данных для отношения «Staff».

Таблица 6 – Словарь данных таблицы «Staff»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KEY** | **Имя поля** | **Тип данных / Размер** | **Необходимо?** | **Примечания** |
| PK | ID | INTEGER | Y | Auto Increment |
|  | fio | NVARCHAR(255) | Y |  |
|  | phone | NVARCHAR(11) | Y |  |
|  | logins | NVARCHAR(30) | Y |  |
|  | passwords | NVARCHAR(30) | Y |  |
|  | type | NVARCHAR(20) | Y |  |

### **Заполненные данными таблицы**

Операция вставки данных в реляционных БД — одна из наиболее востребованных, без которой невозможна нормальная работа с таблицами. В данном приложении таблицы были заполнены средствами реляционных СУБД, работающих с языком SQL.

Со скриптом заполнения базы данных можно ознакомиться в приложении А.

Результаты заполнения таблиц представлены на рисунках 17-22.

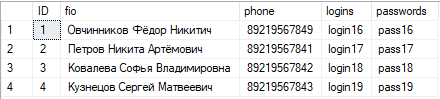


Рисунок 17 – таблица заказчиков

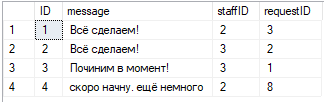


Рисунок 18 – таблица комментариев

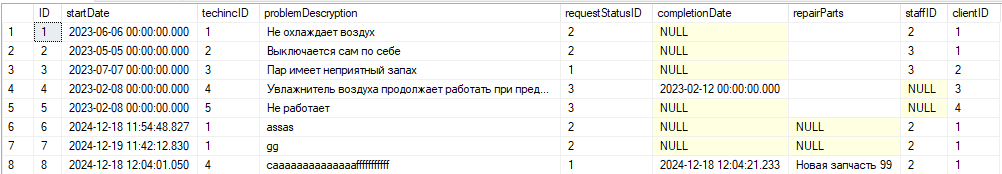


Рисунок 19 – таблица заявок

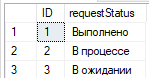


Рисунок 20 – таблица статусов заявки

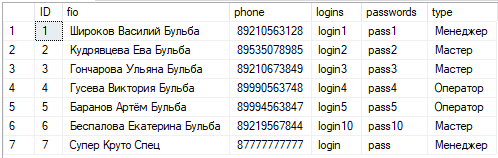


Рисунок 21 – таблица персонала

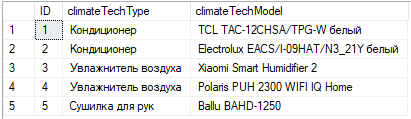


Рисунок 22 – таблица с техникой

### **Резервное копирование**

Резервным копированием называется сохранение копии данных где-то вне основного места их хранения. Главное назначение резервного копирования – восстановление данных после их потери.

Создание резервной копии базы данных и восстановление базы данных представлено на рисунках 23-24.

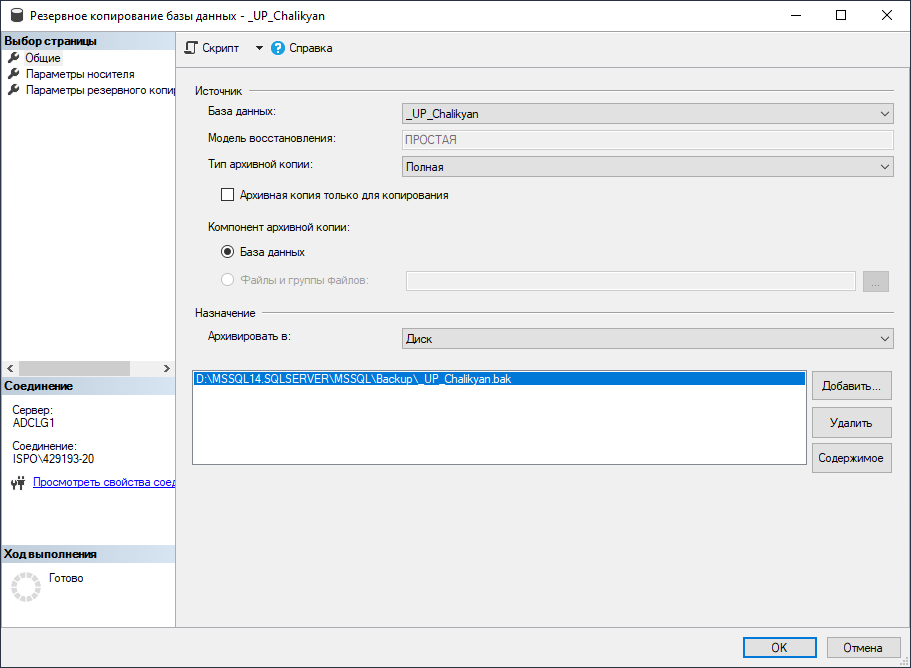
****

Рисунок 23 – Создание резервной копии

****

Рисунок 24 – Файл резервной копии

## **Разработка библиотеки и подключение её к проекту**

Динамическая библиотека — это отдельный исполняемый файл на машинных кодах. Полностью термин звучит как «библиотека подпрограмм». Функции, объекты и другие сущности, которые входят в библиотеку, можно использовать в коде и тем самым облегчать задачу разработки. Например, не писать с нуля сложный алгоритм, а вызвать функцию из библиотеки, где он уже реализован.

Код библиотеки представлен в приложении Б.

## **Разработка приложения**

После установки программы на компьютер и ее запуска на экране появляется окно авторизации пользователя.

При успешном запуске программы появляется экранная форма авторизации – окно с предложением ввести имя пользователя и его пароль. Внешний вид экранной формы авторизации показан на рисунке 1.

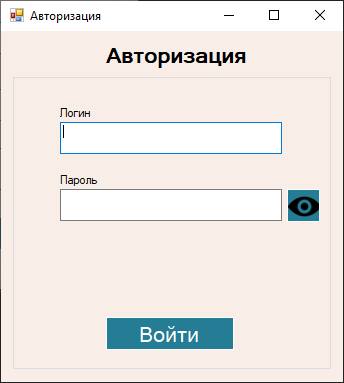


Рисунок 25 – Окна авторизации

На окне есть два поля для ввода данных (логина и пароля). Также представлены 2 кнопки:

1. Кнопка “Войти” – при правильно введенных данных после нажатия нас перенесет в соответствующее рабочее окно
2. Кнопка с иконкой глаза – раскрывает поле с паролем

Открытие формы оператора

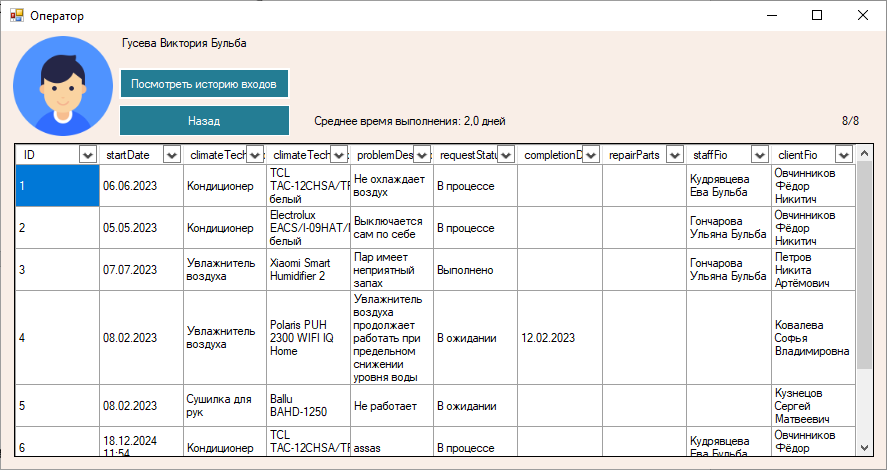


Рисунок 26 – форма оператора

На окне есть возможность фильтрации таблицы и расчёт среднего времени работы. Также имеются 2 кнопки и возможность редактирования заявки, путём клика на неё.

1. Кнопка «Посмотреть историю входов» – открывает историю удачных и неудачных входов всех логинов.

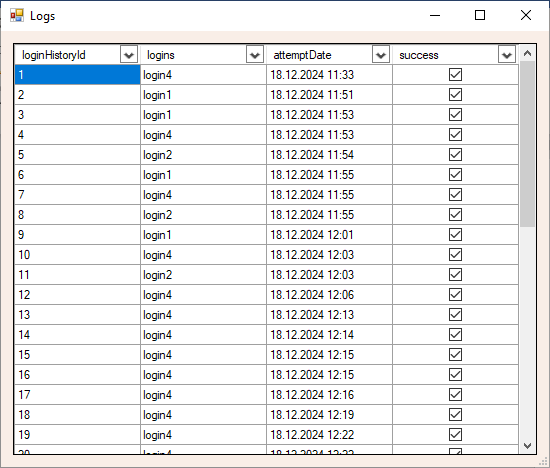


Рисунок 27 – форма с попытками авторизации

1. Кнопка «назад»– отправляет пользователя обратно на форму авторизации.
2. Нажатие на строчку – открывает форму для редактирования заявки, где оператор может назначить мастера и посмотреть комментарии. Нажав на кнопку «Сохранить» - изменения сохраняются, «Назад» же не сохраняет:

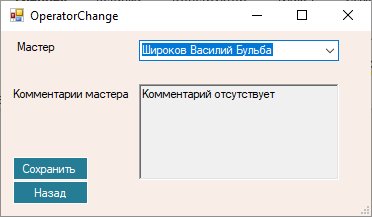


Рисунок 28 – форма редактирования заявки у оператора

Открытие формы клиента

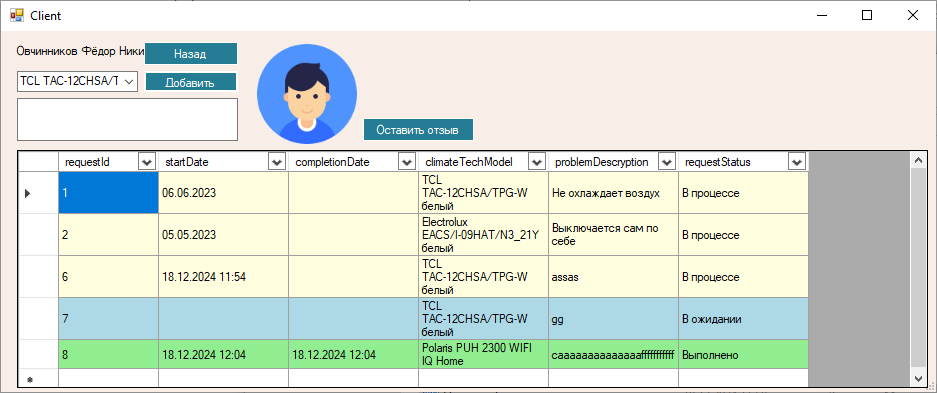


Рисунок 29 – форма клиента

На окне есть возможность фильтрации таблицы, в списке можно выбрать модель техники и описание поломки. Также имеются 3 кнопки и возможность редактирования заявки, путём клика на неё.

1. Кнопка «Назад» – отправляет пользователя обратно на окно авторизации.
2. Кнопка «Добавить»– добавляет в таблицу заявок новую заявку со статусом «В ожидании» (пример – строка под id 7).
3. Нажатие на строчку – открывает форму для редактирования заявки только в случае, если статус «В ожидании», где клиент может изменить модель техники и описание проблемы. Нажав на кнопку «Сохранить» - изменения сохраняются, «Назад» же не сохраняет:

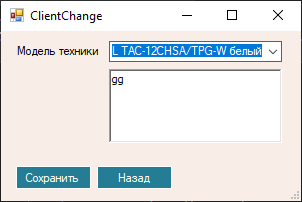


Рисунок 30 – форма редактирования заявки у клиента

1. Нажатие кнопки «Оставить отзыв» - открывает форму с QR-кодом, где заказчик может, перейдя по ссылке, пройти опрос.

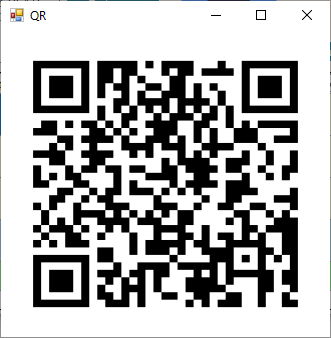


Рисунок 31 – форма с QR-кодом

Открытие формы мастера

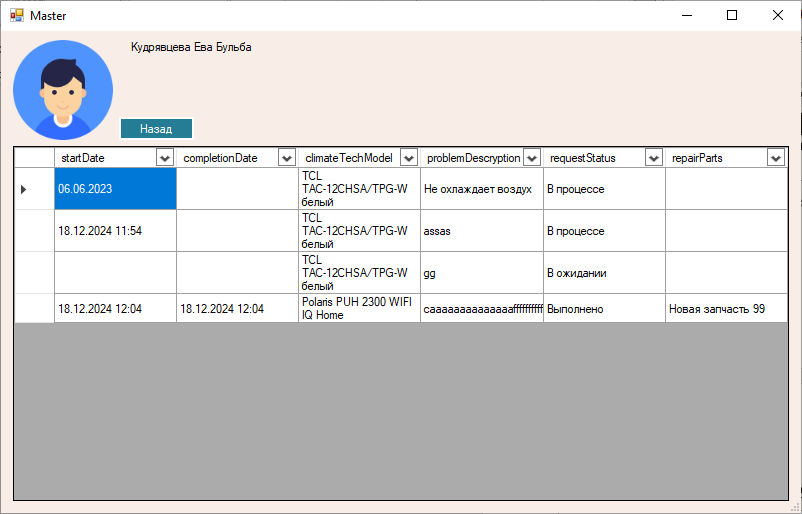


Рисунок 32 – форма мастера

На окне есть возможность фильтрации таблицы. Также имеется 1 кнопка и возможность редактирования заявки, путём клика на неё.

1. Кнопка «Назад» – отправляет пользователя обратно на окно авторизации.
2. Нажатие на строчку – открывает форму для редактирования заявки только в случае, если статус «В процессе» или «В ожидании», где мастер может начать/закончить, написать комментарии и заказать запчасти. Нажав на кнопку «Сохранить» - изменения сохраняются, «Назад» же не сохраняет:

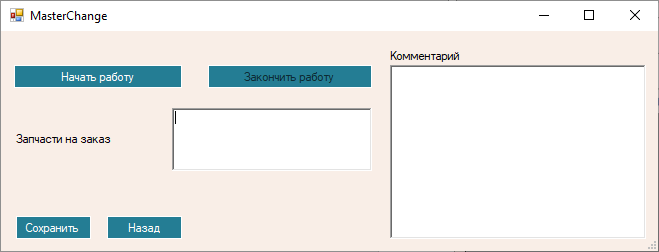


Рисунок 33 – форма редактирования заявки у мастера

Открытие формы менеджера

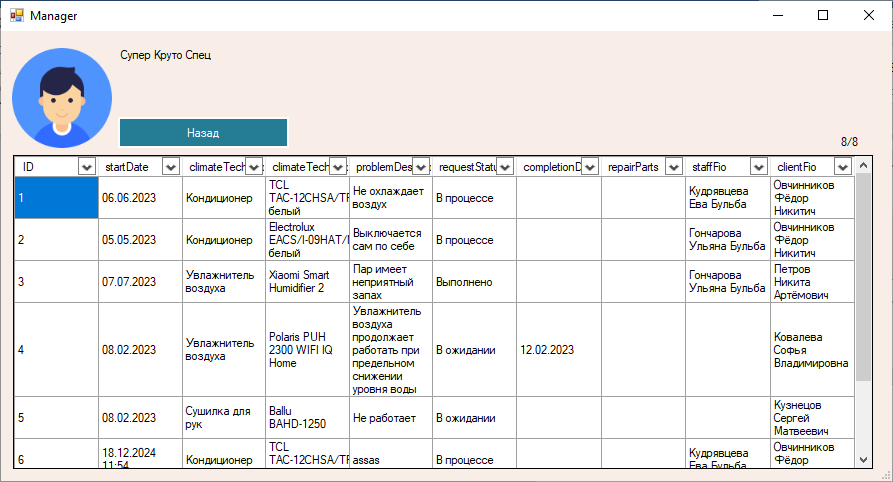


Рисунок 34 – форма менеджера

На окне есть возможность фильтрации таблицы. Также имеется 1 кнопка и возможность редактирования заявки, путём клика на неё.

1. Кнопка «Назад» – отправляет пользователя обратно на окно авторизации.
2. Нажатие на строчку – открывает форму для редактирования заявки, где менеджер может назначить мастера и продлить дату окончания с согласия пользователя. Нажав на кнопку «Сохранить» - изменения сохраняются, «Назад» же не сохраняет:

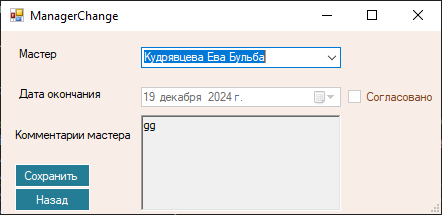


Рисунок 35 – форма редактирования заявки у менеджера

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится узнавать текущие значения переменных и выяснять, по какому пути выполнялась программа. Точки останова — это один из важнейших инструментов разработчика для выполнения отладки. Точки останова устанавливаются везде, где нужно приостановить выполнение отладчика. Например, вы хотите просмотреть состояние переменных кода или стек вызовов в определенной точке останова.

Код программы представлен в приложении В.

Отладка представлена на рисунках 36–41.

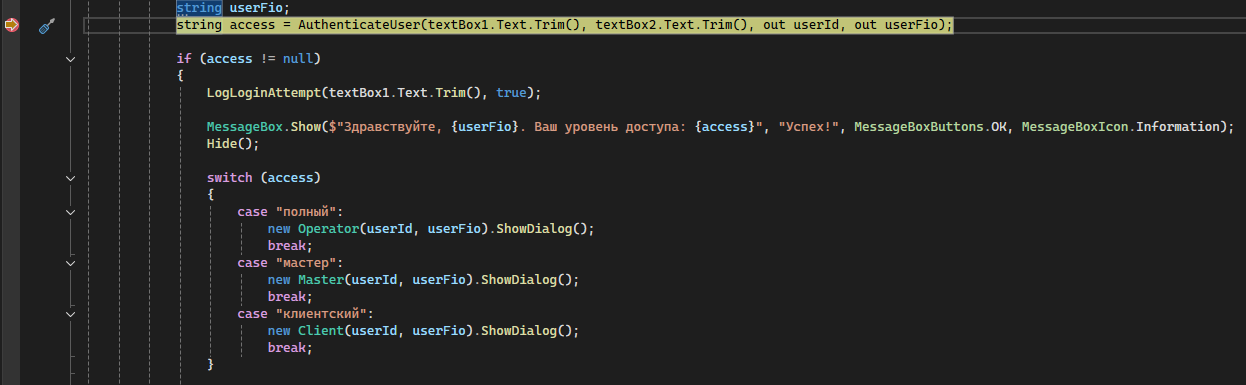


Рисунок 36 – отладка аутентификации пользователя

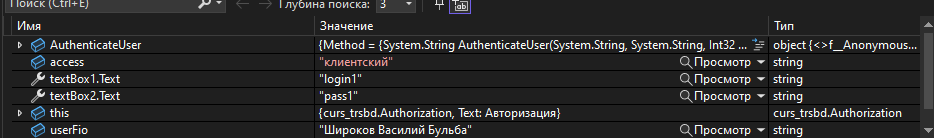


Рисунок 37 – вывод фамилии и имени

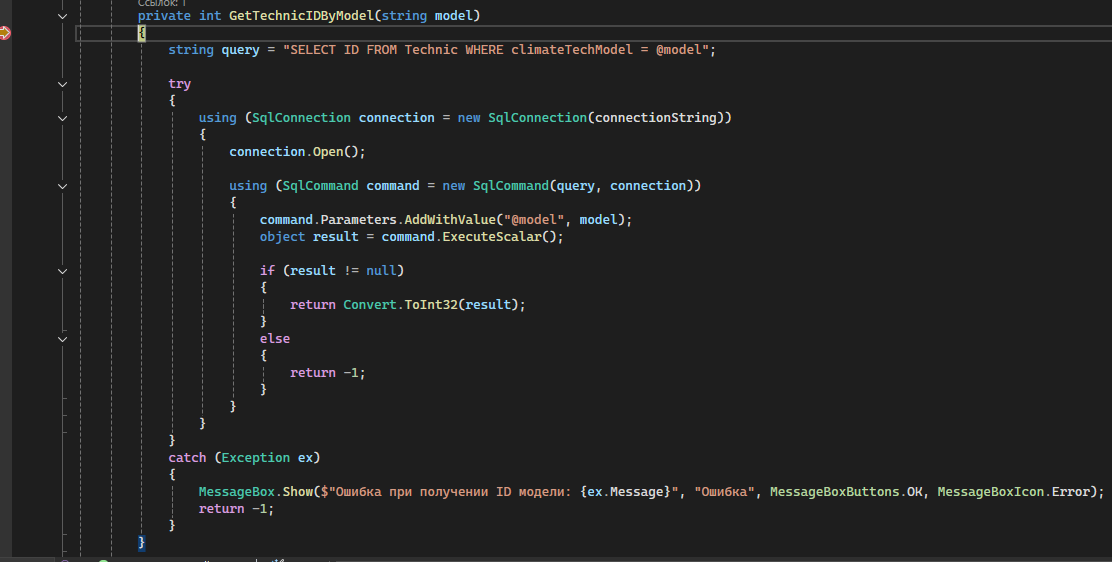


Рисунок 38 – отладка получения id модели

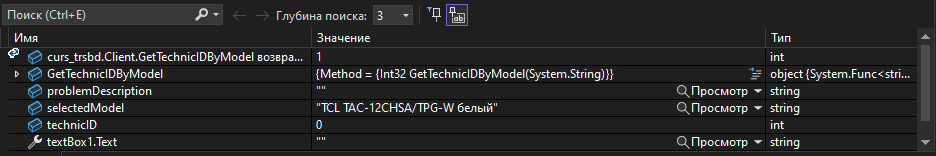


Рисунок 39 – id получено

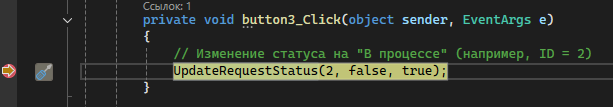


Рисунок 40 – отладка начала работы

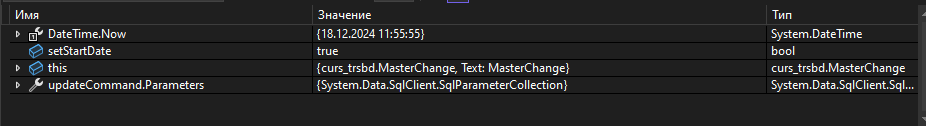


Рисунок 41 – изменение даты начала на текущую

## **Тестирование приложения**

Тестирование программного обеспечения (Software Testing) — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

Цель тестирования — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

### **Создание тестовых случаев**

Тестовый случай (тест-кейс) — это профессиональная документация тестировщика, последовательность действий, направленная на проверку какого-либо функционала, описывающая, как прийти к фактическому результату. Тест-кейсы помогают провести проверку продукта без ознакомления со всей документацией.

Тестовые случаи представлены в приложении Г.

### **Модульное тестирование**

Модульное тестирование, иногда блочное тестирование или юнит-тестирование (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки. Цель модульного тестирования — изолировать отдельные части программы и показать, что по отдельности эти части работоспособны. Также, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность.

Код модульного тестирования представлен в приложении Д.

Результат модульного тестирования библиотеки проекта представлен на рисунке 42.

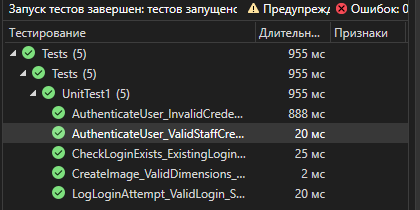


Рисунок 42 – Успешно пройденные юнит-тесты

## **Выгрузка готового проекта в репозиторий GIT**

Git — распределённая система управления версиями. Репозиторий в Git — это место, в котором хранится весь код и вся история его изменений. Он может быть локальным (на компьютере пользователя) или удалённым (на сервере или в облачном хранилище).

Для данного проекта был создан удаленный репозиторий на сайте Git Hub. Вид проекта, загруженного на сайт, предложен на рисунке 44.

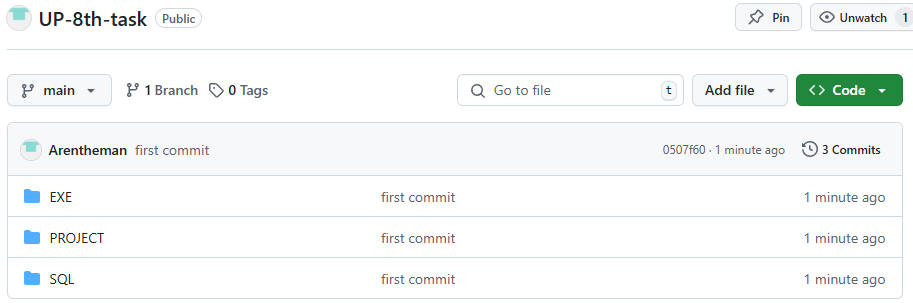


Рисунок 43 – Создание репозитория на Git Hub

## **Качественные характеристики кода**

Качественные характеристики кода — это набор признаков и аспектов, которые определяют, насколько хорошо написан программный код с точки зрения его эффективности, читаемости, удобства сопровождения и других факторов. Качественный код облегчает его поддержку, улучшает производительность и снижает количество ошибок в дальнейшем.

### **Полнота обработки ошибочных данных**

В коде программы предусмотрена обработка возможных ошибок при взаимодействии с базой данных. Используются блоки try-catch, которые перехватывают исключения, возникающие при работе с SqlConnection и SqlCommand. Этот подход позволяет избежать некорректной работы программы при ошибках, связанных с подключением к базе данных или некорректным SQL-запросом. В случае возникновения ошибки выводится сообщение с описанием ошибки:

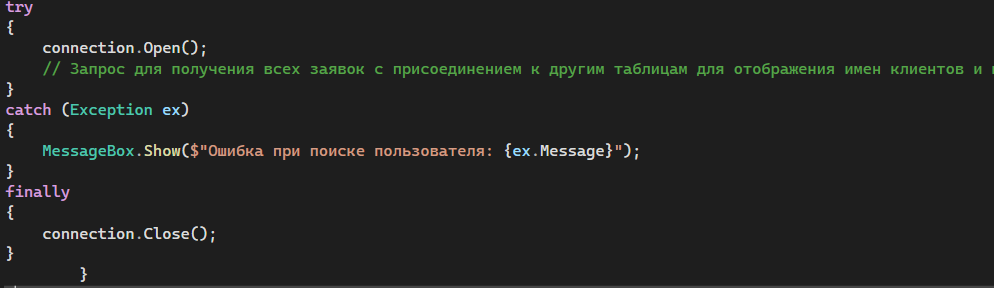


Рисунок 44 – пример конструкции try-catch

### **Наличие тестов для проверки допустимых значений входных данных**

В коде реализована проверка допустимых значений при вводе данных в форме авторизации. Например, при вводе логина и пароля проверяется их наличие в бд через метод AuthenticateUser:

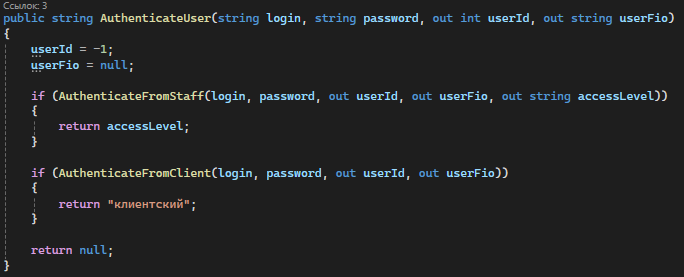


Рисунок 45 – метод AuthenticateUser



Рисунок 46 – метод поиска в одной из таблиц

Также, в форме авторизации при множественных неудачных попытках входа (3 раза) система блокирует доступ и сообщает об этом пользователю:

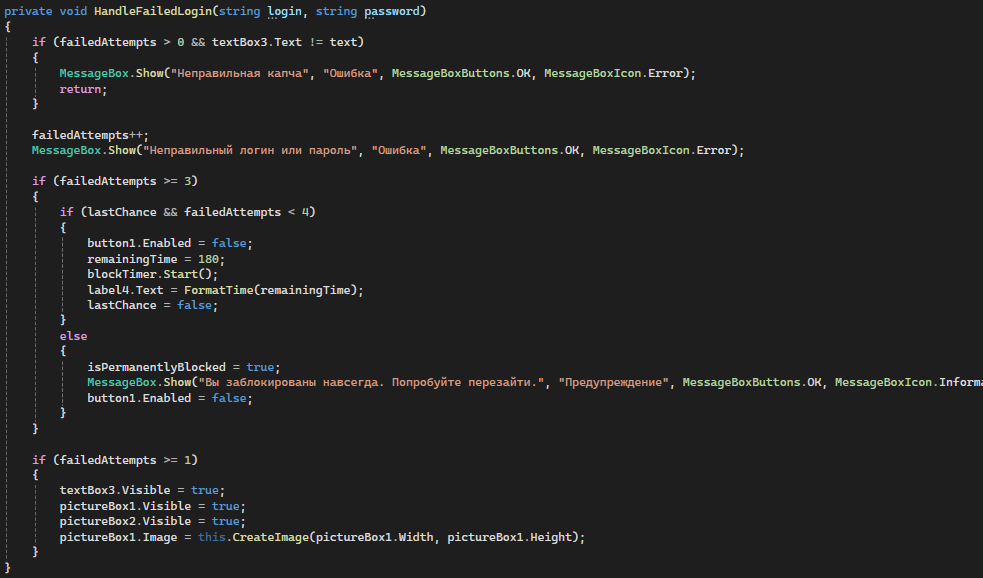


Рисунок 47 – метод обработки ошибочных входов

### **Наличие средств контроля корректности входных данных**

В коде присутствует множество проверок корректности входных данных, например, пустых полей при создании пользователем заявки и вывода сообщения об этом.

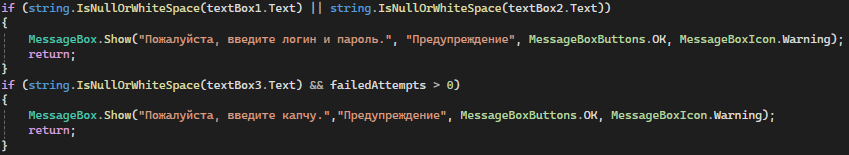


Рисунок 48 – метод обработки пустых полей

### **Наличие средств восстановления при сбоях оборудования**

Код не содержит явных механизмов восстановления после сбоев оборудования. Тем не менее, использование обработки исключений позволяет в определенной степени реагировать на проблемы с оборудованием (например, сбои при подключении к базе данных). В случае сбоя система выводит сообщение об ошибке и даёт пользователю информацию о том, что необходимо предпринять.

### **Наличие комментариев**

В коде присутствуют комментарии для объяснения неоднозначных моментов, например некоторых методов и функций.

### **Наличие проверки корректности передаваемых данных**

Код тщательно проверяет корректность вводимых пользователем данных, таких как логин, пароль и капча. Проверка происходит на уровне ввода данных с последующей валидацией с использованием запросов к базе данных.

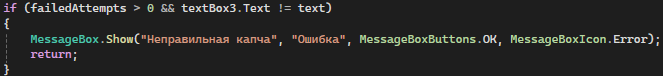


Рисунок 49 – метод обработки ошибочной капчи

### **Наличие описания основных функций**

Каждый метод выполняет свою отдельную задачу, что соответствует принципам структурированного и читаемого кода.

* button1\_Click(object sender, EventArgs e) - Обработчик события для кнопки "Отправить", выполняет валидацию данных, добавляет информацию об оргтехнике и запросе в базу данных, и отображает сообщения об успехе или ошибках.
* button1\_Click - Обрабатывает нажатие кнопки для входа в систему, проверяет, заблокирован ли пользователь, валидирует введенные данные, управляет попытками входа и отображает CAPTCHA при необходимости.

# **ЗАКЛЮЧЕНИЕ**

В результате учебной практики был создан программный продукта, который позволяет сервисам по ремонту климатического оборудования эффективно отслеживать учет заявок на ремонт.

Была достигнута основная цель разработки программного продукта.

Были выполнены задачи:

* Анализ предметной области. Разработка технического задания.
* Проектирование диаграмм вариантов использования, последовательности, активности.
* Моделирование структуры ПО. Проектирование интерфейса пользователя.
* Разработка базы данных и словаря данных.
* Создание приложения: форма авторизации, истории входов, основных форм приложения с заявками. Разработка библиотеки классов.
* Отладка программных модулей.
* Модульное тестирование. Создание тестовых случаев.
* Инспекция кода на соответствие стандартам кодирования.
* Подготовка отчета и размещение результатов в репозитории контроля версий.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Стандарты Единой Системы Программной Документации:

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению

ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

1. Культин Н., C++ в задачах и примерах, 5-е издание, СПб: БХВ, 2020 — 656 с.
2. Моллер Й., Уэллингер С., C++: Полное руководство, 2-е издание, М: Вильямс, 2019 — 880 с.
3. Харрисон Л., Qt 5: Графика и создание интерфейсов, М: ДМК Пресс, 2018 — 560 с.
4. Документация по Qt [Электронный ресурс] — URL: https://doc.qt.io/qt-5/gettingstarted.html (дата обращения: 16.12.2024)

# Приложение А

(справочное)

**Скрипт БД**

Create table Client

(

ID INT PRIMARY KEY IDENTITY,

fio NVARCHAR(255) NOT NULL,

phone NVARCHAR(11) NOT NULL,

logins NVARCHAR(30) NOT NULL,

passwords NVARCHAR(30) NOT NULL,

CONSTRAINT check\_phone\_format CHECK (phone LIKE '8[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

)

Create table Staff

(

ID INT PRIMARY KEY IDENTITY,

fio NVARCHAR(255) NOT NULL,

phone NVARCHAR(11) NOT NULL,

logins NVARCHAR(30) NOT NULL,

passwords NVARCHAR(30) NOT NULL,

[type] NVARCHAR(20) NOT NULL,

CONSTRAINT check\_phone\_format1 CHECK (phone LIKE '8[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

)

Create table RequestStatus

(

ID INT PRIMARY KEY IDENTITY,

requestStatus NVARCHAR(255) NOT NULL

)

Create table Technic

(

ID INT PRIMARY KEY IDENTITY,

climateTechType NVARCHAR(255) NOT NULL,

climateTechModel NVARCHAR(255) NOT NULL

)

Create table Requests

(

ID INT PRIMARY KEY IDENTITY,

startDate DATETIME,

techincID INT,

problemDescryption NVARCHAR(255) NOT NULL,

requestStatusID INT,

completionDate DATETIME,

repairParts NVARCHAR(255),

staffID INT,

clientID INT,

FOREIGN KEY (staffID) REFERENCES Staff(ID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (clientID) REFERENCES Client(ID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (requestStatusID) REFERENCES RequestStatus(ID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (techincID) REFERENCES Technic(ID) ON DELETE CASCADE ON UPDATE CASCADE,

)

CREATE TABLE Comments

(

ID INT PRIMARY KEY IDENTITY,

[message] NVARCHAR(255) NOT NULL,

staffID INT,

requestID INT,

FOREIGN KEY (staffID) REFERENCES Staff(ID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (requestID) REFERENCES Requests(ID) ON DELETE NO ACTION ON UPDATE NO ACTION

)

CREATE TABLE loginHistory (

loginHistoryId INT PRIMARY KEY IDENTITY,

logins NVARCHAR(30) NOT NULL,

attemptDate DATETIME NOT NULL,

success BIT NOT NULL

);

INSERT INTO Staff

VALUES

('Широков Василий Бульба','89210563128','login1','pass1','Менеджер'),

('Кудрявцева Ева Бульба','89535078985','login2','pass2','Мастер'),

('Гончарова Ульяна Бульба','89210673849','login3','pass3','Мастер'),

('Гусева Виктория Бульба','89990563748','login4','pass4','Оператор'),

('Баранов Артём Бульба','89994563847','login5','pass5','Оператор'),

('Беспалова Екатерина Бульба','89219567844','login10','pass10','Мастер');

INSERT INTO Client

VALUES

('Овчинников Фёдор Никитич','89219567849','login16','pass16'),

('Петров Никита Артёмович','89219567841','login17','pass17'),

('Ковалева Софья Владимировна','89219567842','login18','pass18'),

('Кузнецов Сергей Матвеевич','89219567843','login19','pass19');

INSERT INTO Comments

Values

('Всё сделаем!', 2, 3),

('Всё сделаем!', 3, 2),

('Починим в момент!', 3, 1);

INSERT INTO RequestStatus

Values

('Выполнено'),

('В процессе'),

('В ожидании');

INSERT INTO Technic

VALUES

('Кондиционер', 'TCL TAC-12CHSA/TPG-W белый'),

('Кондиционер', 'Electrolux EACS/I-09HAT/N3\_21Y белый'),

('Увлажнитель воздуха', 'Xiaomi Smart Humidifier 2'),

('Увлажнитель воздуха', 'Polaris PUH 2300 WIFI IQ Home'),

('Сушилка для рук', 'Ballu BAHD-1250');

INSERT INTO Requests

VALUES

('2023-06-06',1,'Не охлаждает воздух',2,null,'',2,1),

('2023-05-05',2,'Выключается сам по себе',2,null,'',3,1),

('2023-07-07',3,'Пар имеет неприятный запах',1,null,'',3,2),

('2023-08-02',4,'Увлажнитель воздуха продолжает работать при предельном снижении уровня воды',3,'2023-12-02','',null,3),

('2023-08-02',5,'Не работает',3,null,'',null,4);

# Приложение Б

(справочное)

**Библиотека классов**

Файл «AvarageTime.cs»:

using System;

using System.Data;

namespace curs\_trsbd

{

public static class AvarageTime

{

static string result;

public static string UpdateAverageCompletionTime(DataTable dataTable)

{

try

{

TimeSpan totalDuration = TimeSpan.Zero;

int completedCount = 0;

foreach (DataRow row in dataTable.Rows)

{

if (row["completionDate"] != DBNull.Value && row["startDate"] != DBNull.Value)

{

DateTime startDate = Convert.ToDateTime(row["startDate"]);

DateTime completionDate = Convert.ToDateTime(row["completionDate"]);

totalDuration += (completionDate - startDate);

completedCount++;

}

}

if (completedCount > 0)

{

double avgDuration = totalDuration.TotalDays / completedCount;

return $"Среднее время выполнения: {avgDuration:F1} дней";

}

else

{

return "Среднее время выполнения: нет завершенных заявок";

}

}

catch (Exception ex)

{

return $"Ошибка при вычислении среднего времени выполнения: {ex.Message}";

}

}

}

}

# Приложение В

(справочное)

**Исходный код**

Файл «Authorization.cs»:

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class Authorization : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

private string text = String.Empty;

private int failedAttempts = 0;

private Timer blockTimer;

private int remainingTime;

private bool isPermanentlyBlocked = false;

private bool lastChance = true;

public Authorization()

{

InitializeComponent();

pictureBox1.Image = this.CreateImage(pictureBox1.Width, pictureBox1.Height);

blockTimer = new Timer();

blockTimer.Interval = 1;

blockTimer.Tick += BlockTimer\_Tick;

textBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox2.Visible = false;

}

private void TextBoxToLower(object sender, EventArgs e)

{

if (sender is TextBox textBox)

{

int cursorPosition = textBox.SelectionStart;

textBox.Text = textBox.Text.ToLower();

textBox.SelectionStart = cursorPosition;

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(textBox1.Text) || string.IsNullOrWhiteSpace(textBox2.Text))

{

MessageBox.Show("Пожалуйста, введите логин и пароль.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (string.IsNullOrWhiteSpace(textBox3.Text) && failedAttempts > 0)

{

MessageBox.Show("Пожалуйста, введите капчу.","Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

int userId;

string userFio;

string access = AuthenticateUser(textBox1.Text.Trim(), textBox2.Text.Trim(), out userId, out userFio);

if (access != null)

{

LogLoginAttempt(textBox1.Text.Trim(), true);

MessageBox.Show($"Здравствуйте, {userFio}. Ваш уровень доступа: {access}", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

Hide();

switch (access)

{

case "полный":

new Operator(userId, userFio).ShowDialog();

break;

case "мастер":

new Master(userId, userFio).ShowDialog();

break;

case "менеджер":

new Manager(userId, userFio).ShowDialog();

break;

case "клиентский":

new Client(userId, userFio).ShowDialog();

break;

}

Close();

}

else

{

bool isLoginValid = CheckLoginExists(textBox1.Text.Trim());

if (isLoginValid)

{

LogLoginAttempt(textBox1.Text.Trim(), false);

}

HandleFailedLogin(textBox1.Text.Trim(), textBox2.Text.Trim());

}

}

public string AuthenticateUser(string login, string password, out int userId, out string userFio)

{

userId = -1;

userFio = null;

if (AuthenticateFromStaff(login, password, out userId, out userFio, out string accessLevel))

{

return accessLevel;

}

if (AuthenticateFromClient(login, password, out userId, out userFio))

{

return "клиентский";

}

return null;

}

private bool AuthenticateFromStaff(string login, string password, out int userId, out string userFio, out string accessLevel)

{

userId = -1;

userFio = null;

accessLevel = null;

string query = @"

SELECT ID, fio,

CASE

WHEN [type] = 'Оператор' THEN 'полный'

WHEN [type] = 'Мастер' THEN 'мастер'

WHEN [type] = 'Менеджер' THEN 'менеджер'

ELSE 'клиентский'

END AS AccessLevel

FROM Staff

WHERE logins = @login AND passwords = @password";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@login", login);

command.Parameters.AddWithValue("@password", password);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

userId = Convert.ToInt32(reader["ID"]);

userFio = reader["fio"].ToString();

accessLevel = reader["AccessLevel"].ToString();

return true;

}

}

}

}

return false;

}

private bool AuthenticateFromClient(string login, string password, out int userId, out string userFio)

{

userId = -1;

userFio = null;

string query = @"

SELECT ID, fio

FROM Client

WHERE logins = @login AND passwords = @password";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@login", login);

command.Parameters.AddWithValue("@password", password);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

userId = Convert.ToInt32(reader["ID"]);

userFio = reader["fio"].ToString();

return true;

}

}

}

}

return false;

}

public void LogLoginAttempt(string login, bool success)

{

string query = @"

INSERT INTO loginHistory (logins, attemptDate, success)

VALUES (@login, @attemptDate, @success)";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@login", login);

command.Parameters.AddWithValue("@attemptDate", DateTime.Now);

command.Parameters.AddWithValue("@success", success);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка логирования: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public bool CheckLoginExists(string login)

{

string query = @"

SELECT COUNT(1)

FROM Staff

WHERE logins = @login

UNION ALL

SELECT COUNT(1)

FROM Client

WHERE logins = @login";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@login", login);

int result = (int)command.ExecuteScalar();

return result > 0;

}

}

}

private void HandleFailedLogin(string login, string password)

{

if (failedAttempts > 0 && textBox3.Text != text)

{

MessageBox.Show("Неправильная капча", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

failedAttempts++;

MessageBox.Show("Неправильный логин или пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

if (failedAttempts >= 3)

{

if (lastChance && failedAttempts < 4)

{

button1.Enabled = false;

remainingTime = 180;

blockTimer.Start();

label4.Text = FormatTime(remainingTime);

lastChance = false;

}

else

{

isPermanentlyBlocked = true;

MessageBox.Show("Вы заблокированы навсегда. Попробуйте перезайти.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Information);

button1.Enabled = false;

}

}

if (failedAttempts >= 1)

{

textBox3.Visible = true;

pictureBox1.Visible = true;

pictureBox2.Visible = true;

pictureBox1.Image = this.CreateImage(pictureBox1.Width, pictureBox1.Height);

}

}

private void BlockTimer\_Tick(object sender, EventArgs e)

{

remainingTime--;

label4.Text = FormatTime(remainingTime);

if (remainingTime <= 0)

{

blockTimer.Stop();

button1.Enabled = true;

label4.Text = string.Empty;

}

}

private string FormatTime(int seconds)

{

int minutes = seconds / 60;

seconds %= 60;

return $"{minutes}:{seconds:D2}";

}

public Bitmap CreateImage(int Width, int Height)

{

Random rnd = new Random();

Bitmap result = new Bitmap(Width, Height);

int Xpos = rnd.Next(0, Width - 50);

int Ypos = rnd.Next(15, Height - 15);

Brush[] colors = { Brushes.Black, Brushes.Red, Brushes.White };

Graphics g = Graphics.FromImage((Image)result);

g.Clear(Color.Gray);

text = String.Empty;

string ALF = "1234567890QWERTYUIOPASDFGHJKLZXCVBNM";

for (int i = 0; i < 5; ++i)

text += ALF[rnd.Next(ALF.Length)];

g.DrawString(text, new Font("Arial", 15), colors[rnd.Next(colors.Length)], new PointF(Xpos, Ypos));

g.DrawLine(Pens.Black, new Point(0, 0), new Point(Width - 1, Height - 1));

g.DrawLine(Pens.Black, new Point(0, Height - 1), new Point(Width - 1, 0));

return result;

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

pictureBox1.Image = this.CreateImage(pictureBox1.Width, pictureBox1.Height);

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBox2.PasswordChar == '\*')

{

textBox2.PasswordChar = '\0';

}

else

{

textBox2.PasswordChar = '\*';

}

}

}

}

Файл «Client.cs»:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class Client : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

private int userId;

private string userFio;

public Client(int userId, string userFio)

{

InitializeComponent();

pictureBox1.Image = Image.FromFile(@"D:\4 курс\УП 02.01\curs\_trsbd\curs trsbd\curs trsbd\Resources\progile.jpg");

this.userId = userId;

FillDataGridView();

dataGridView1.DataBindingComplete += DataGridView1\_DataBindingComplete;

dataGridView1.RowPrePaint += dataGridView1\_RowPrePaint;

FillComboBoxWithModels();

comboBox1.SelectedIndex = 0;

dataGridView1.CellClick += DataGridView1\_CellClick;

this.userFio = userFio;

label1 .Text = userFio;

}

public void FillDataGridView()

{

string query = @"

SELECT r.ID as requestId, r.startDate, r.completionDate, t.climateTechModel,

r.problemDescryption, rs.requestStatus

FROM Requests r

LEFT JOIN Technic t ON r.techincID = t.ID

LEFT JOIN RequestStatus rs ON r.requestStatusID = rs.ID

WHERE r.clientID = @userId;";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@userId", userId);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

// Добавляем скрытую колонку для requestId

DataGridViewColumn requestIdColumn = new DataGridViewTextBoxColumn

{

Name = "requestId",

DataPropertyName = "requestId",

Visible = false // Скрываем колонку

};

dataGridView1.Columns.Add(requestIdColumn);

// Устанавливаем ширину для остальных колонок

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

if (column.Name != "requestId") // Не меняем ширину для скрытой колонки

{

column.Width = 130;

column.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

}

}

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;

dataGridView1.RowHeadersVisible = true;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void DataGridView1\_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs e)

{

int totalWidth = 0;

foreach (DataGridViewColumn column in dataGridView1.Columns)

totalWidth += column.Width;

totalWidth += dataGridView1.RowHeadersWidth;

dataGridView1.Width = totalWidth + 20;

int totalHeight = dataGridView1.Rows.Cast<DataGridViewRow>().Sum(r => r.Height) + dataGridView1.ColumnHeadersHeight;

dataGridView1.Height = Math.Min(totalHeight, 600);

this.Width = dataGridView1.Width + 40;

this.Height = Math.Max(dataGridView1.Height + 60, 400);

}

private void dataGridView1\_RowPrePaint(object sender, DataGridViewRowPrePaintEventArgs e)

{

try

{

if (!dataGridView1.Rows[e.RowIndex].IsNewRow)

{

var status = dataGridView1.Rows[e.RowIndex].Cells["requestStatus"].Value?.ToString();

if (status == "Выполнено")

dataGridView1.Rows[e.RowIndex].DefaultCellStyle.BackColor = Color.LightGreen;

else if (status == "В процессе")

dataGridView1.Rows[e.RowIndex].DefaultCellStyle.BackColor = Color.LightYellow;

else if (status == "В ожидании")

dataGridView1.Rows[e.RowIndex].DefaultCellStyle.BackColor = Color.LightBlue;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка форматирования строки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void FillComboBoxWithModels()

{

string query = "SELECT climateTechModel FROM Technic";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader["climateTechModel"].ToString());

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button1\_Click(object sender, EventArgs e)

{

string selectedModel = comboBox1.SelectedItem.ToString();

string problemDescription = textBox1.Text.Trim();

int technicID = GetTechnicIDByModel(selectedModel);

if (textBox1.Text.Trim() == string.Empty)

{

MessageBox.Show("Напишите описание поломки", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (technicID == -1)

{

MessageBox.Show("Выберите допустимую модель техники!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string query = @"

INSERT INTO Requests

(clientID, techincID, problemDescryption, requestStatusID)

VALUES

(@clientID, @techincID, @problemDescryption, @requestStatusID)";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@clientID", userId);

command.Parameters.AddWithValue("@techincID", technicID);

command.Parameters.AddWithValue("@problemDescryption", problemDescription);

command.Parameters.AddWithValue("@requestStatusID", 3);

command.ExecuteNonQuery();

}

}

MessageBox.Show("Запрос успешно добавлен!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

FillDataGridView();

dataGridView1.RowPrePaint += dataGridView1\_RowPrePaint;

}

private int GetTechnicIDByModel(string model)

{

string query = "SELECT ID FROM Technic WHERE climateTechModel = @model";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@model", model);

object result = command.ExecuteScalar();

if (result != null)

{

return Convert.ToInt32(result);

}

else

{

return -1;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении ID модели: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return -1;

}

}

private void DataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

var requestId = dataGridView1.Rows[e.RowIndex].Cells["requestId"].Value?.ToString();

var status = dataGridView1.Rows[e.RowIndex].Cells["requestStatus"].Value?.ToString();

if (status == "В ожидании")

{

if (requestId != null)

{

ClientChange editForm = new ClientChange(Convert.ToInt32(requestId), userId, userFio);

this.Hide();

editForm.ShowDialog();

this.Close();

}

else

{

MessageBox.Show("Ошибка: не найдено ID заявки.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Редактирование доступно только для заявок со статусом 'В ожидании'.", "Редактирование недоступно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

Authorization authorization = new Authorization();

this.Hide();

authorization.ShowDialog();

this.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

QR code = new QR();

code.ShowDialog();

}

}

}

Файл «ClientChange.cs»:

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class ClientChange : Form

{

private int requestId;

private int userid;

private string userFio;

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

public ClientChange(int requestId, int userid, string userFio)

{

InitializeComponent();

this.requestId = requestId;

LoadData();

FillComboBoxWithModels();

this.userid = userid;

comboBox1.SelectedIndex = 0;

this.userFio = userFio;

label2.Text = "";

}

private void LoadData()

{

string query = "SELECT techincID, problemDescryption FROM Requests WHERE ID = @requestId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestId", requestId);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

// Заполняем форму данными заявки

int technicId = reader.GetInt32(reader.GetOrdinal("techincID"));

string problemDescription = reader["problemDescryption"].ToString();

// Ищем выбранную модель по технике

comboBox1.SelectedItem = GetModelByTechnicId(technicId);

richTextBox1.Text = problemDescription;

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void FillComboBoxWithModels()

{

string query = "SELECT climateTechModel FROM Technic";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader["climateTechModel"].ToString());

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при заполнении ComboBox: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private string GetModelByTechnicId(int technicId)

{

string query = "SELECT climateTechModel FROM Technic WHERE ID = @technicId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@technicId", technicId);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

else

{

return string.Empty;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении модели: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return string.Empty;

}

}

private void button1\_Click(object sender, EventArgs e)

{

string selectedModel = comboBox1.SelectedItem?.ToString();

string problemDescription = richTextBox1.Text.Trim();

if (string.IsNullOrEmpty(selectedModel))

{

MessageBox.Show("Выберите модель техники.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (string.IsNullOrEmpty(problemDescription))

{

MessageBox.Show("Опишите проблему.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int technicID = GetTechnicIDByModel(selectedModel);

if (technicID == -1)

{

MessageBox.Show("Выберите допустимую модель техники.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string query = @"

UPDATE Requests

SET techincID = @techincID, problemDescryption = @problemDescryption

WHERE ID = @requestId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestId", requestId);

command.Parameters.AddWithValue("@techincID", technicID);

command.Parameters.AddWithValue("@problemDescryption", problemDescription);

command.ExecuteNonQuery();

MessageBox.Show("Данные успешно обновлены!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Hide();

Client clientForm = new Client(userid, userFio);

clientForm.ShowDialog();

this.Close();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private int GetTechnicIDByModel(string model)

{

string query = "SELECT ID FROM Technic WHERE climateTechModel = @model";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@model", model);

object result = command.ExecuteScalar();

if (result != null)

{

return Convert.ToInt32(result);

}

else

{

return -1;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении ID модели: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return -1;

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

Client clientForm = new Client(userid, userFio);

clientForm.ShowDialog();

this.Close();

}

}

}

Файл «Logs.cs»:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class Logs : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

public Logs()

{

InitializeComponent();

InitializeDataGridView();

FillLoginHistoryDataGridView();

}

private void InitializeDataGridView()

{

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

dataGridView1.ReadOnly = true;

dataGridView1.AllowUserToAddRows = false;

dataGridView1.RowHeadersVisible = false;

dataGridView1.AllowUserToResizeColumns = false;

}

private void FillLoginHistoryDataGridView()

{

string query = "SELECT loginHistoryId, logins, attemptDate, success FROM loginHistory";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Файл «Manager.cs»:

using System;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class Manager : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

private int userId;

private string userFio;

private int totalRequests;

private int currentRequests;

public Manager(int userId, string userFio)

{

InitializeComponent();

LoadTotalRequests();

this.userId = userId;

FillDataGridView();

InitializeDataGridView();

dataGridView3.CellClick += dataGridView3\_CellDoubleClick;

this.userFio = userFio;

pictureBox1.Image = Image.FromFile(@"D:\4 курс\УП 02.01\curs\_trsbd\curs trsbd\curs trsbd\Resources\progile.jpg");

label1.Text = userFio;

currentRequests = dataGridView3.Rows.Count;

label3.Text = $"{currentRequests}/{totalRequests}";

}

private void LoadTotalRequests()

{

string query = "SELECT COUNT(\*) FROM Requests";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

totalRequests = (int)command.ExecuteScalar();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении общего количества заявок: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void InitializeDataGridView()

{

dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView3.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView3.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

foreach (DataGridViewColumn column in dataGridView3.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

dataGridView3.ReadOnly = true;

dataGridView3.AllowUserToAddRows = false;

dataGridView3.RowHeadersVisible = false;

dataGridView3.AllowUserToResizeColumns = false;

}

public void FillDataGridView()

{

try

{

string query = @"SELECT

r.ID,

r.startDate,

t.climateTechType,

t.climateTechModel,

r.problemDescryption,

rs.requestStatus,

r.completionDate,

r.repairParts,

s.fio AS staffFio,

c.fio AS clientFio

FROM

Requests r

LEFT JOIN

Technic t ON r.techincID = t.ID

LEFT JOIN

RequestStatus rs ON r.requestStatusID = rs.ID

LEFT JOIN

Staff s ON r.staffID = s.ID

LEFT JOIN

Client c ON r.clientID = c.ID

ORDER BY

r.ID;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

dataAdapter.Fill(dataTable);

bindingSource.DataSource = dataTable;

dataGridView3.DataSource = bindingSource;

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView3\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

int requestId = Convert.ToInt32(dataGridView3.Rows[e.RowIndex].Cells["ID"].Value);

this.Hide();

ManagerChange operatorChange = new ManagerChange(requestId, userId, userFio);

operatorChange.ShowDialog();

this.Close();

FillDataGridView();

}

}

private void button1\_Click(object sender, EventArgs e)

{

Authorization authorization = new Authorization();

this.Hide();

authorization.ShowDialog();

this.Close();

}

private void bindingSource\_ListChanged(object sender, ListChangedEventArgs e)

{

try

{

if (dataGridView3.DataSource is BindingSource dataTable)

{

label3.Text = $"{bindingSource.Count}/{totalRequests}";

}

else

{

currentRequests = 0;

label3.Text = $"0/{totalRequests}";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении строк после фильтрации: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Файл «ManagerChange.cs»:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class ManagerChange : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

private int requestId;

private int userId;

private string userFio;

private DateTime startDate;

public ManagerChange(int requestId, int userId, string userFio)

{

InitializeComponent();

this.requestId = requestId;

this.userId = userId;

this.userFio = userFio;

LoadRequestDetails();

LoadStaffToComboBox();

}

private void LoadRequestDetails()

{

string query = @"SELECT

r.startDate,

r.completionDate,

r.problemDescryption,

s.fio AS staffFio

FROM

Requests r

LEFT JOIN

Staff s ON r.staffID = s.ID

WHERE

r.ID = @requestId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestId", requestId);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

startDate = reader["startDate"] != DBNull.Value

? Convert.ToDateTime(reader["startDate"])

: DateTime.MinValue;

if (startDate != DateTime.MinValue)

{

dateTimePicker1.MinDate = startDate;

dateTimePicker1.Enabled = checkBox1.Checked;

checkBox1.Enabled = true;

if (reader["completionDate"] != DBNull.Value)

{

dateTimePicker1.Value = Convert.ToDateTime(reader["completionDate"]);

}

}

else

{

dateTimePicker1.Enabled = false;

checkBox1.Enabled = false;

}

richTextBox1.Text = reader["problemDescryption"].ToString();

richTextBox1.ReadOnly = true;

if (reader["staffFio"] != DBNull.Value)

{

comboBox1.SelectedItem = reader["staffFio"].ToString();

}

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных заявки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadStaffToComboBox()

{

string query = "SELECT ID, fio FROM Staff WHERE [type] = 'Мастер'";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

DataTable staffTable = new DataTable();

staffTable.Load(reader);

comboBox1.DataSource = staffTable;

comboBox1.DisplayMember = "fio";

comboBox1.ValueMember = "ID";

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке списка мастеров: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedValue == null || dateTimePicker1.Value < startDate)

{

MessageBox.Show("Пожалуйста, выберите корректного мастера и дату завершения.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

UpdateRequest();

}

private void UpdateRequest()

{

string query = @"

UPDATE Requests

SET

completionDate = @completionDate,

staffID = @staffId

WHERE

ID = @requestId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

if (dateTimePicker1.Enabled)

{

command.Parameters.AddWithValue("@completionDate", dateTimePicker1.Value);

}

else

{

command.Parameters.AddWithValue("@completionDate", DBNull.Value);

}

command.Parameters.AddWithValue("@staffId", comboBox1.SelectedValue ?? (object)DBNull.Value);

command.Parameters.AddWithValue("@requestId", requestId);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Заявка успешно обновлена.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Hide();

Manager clientForm = new Manager(userId, userFio);

clientForm.ShowDialog();

this.Close();

}

else

{

MessageBox.Show("Не удалось обновить заявку. Попробуйте еще раз.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении заявки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

Manager clientForm = new Manager(userId, userFio);

clientForm.ShowDialog();

this.Close();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

dateTimePicker1.Enabled = checkBox1.Checked;

}

}

}

Файл «Master.cs»:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class Master : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

private int userid;

private string userFio;

public Master(int userid, string userFio)

{

InitializeComponent();

pictureBox1.Image = Image.FromFile(@"D:\4 курс\УП 02.01\curs\_trsbd\curs trsbd\curs trsbd\Resources\progile.jpg");

this.userid = userid;

FillDataGridView();

InitializeDataGridView();

dataGridView1.CellClick += DataGridView1\_CellClick;

this.userFio = userFio;

label1 .Text = userFio;

}

private void InitializeDataGridView()

{

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

dataGridView1.ReadOnly = true;

dataGridView1.AllowUserToAddRows = false;

dataGridView1.AllowUserToResizeColumns = false;

dataGridView1.SelectionMode = DataGridViewSelectionMode.CellSelect;

dataGridView1.MultiSelect = false;

}

public void FillDataGridView()

{

string query = @"

SELECT r.ID, r.startDate, r.completionDate, t.climateTechModel,

r.problemDescryption, rs.requestStatus, r.repairParts

FROM Requests r

LEFT JOIN Technic t ON r.techincID = t.ID

LEFT JOIN RequestStatus rs ON r.requestStatusID = rs.ID

WHERE r.staffID = @userId;";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@userId", userid);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

dataGridView1.Columns["ID"].Width = 0;

dataGridView1.Columns["ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void DataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex >= 0)

{

int requestId = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells["ID"].Value);

this.Hide();

MasterChange requestDetailsForm = new MasterChange(requestId, userid, userFio);

requestDetailsForm.ShowDialog();

this.Close();

}

}

private void button1\_Click(object sender, EventArgs e)

{

Authorization authorization = new Authorization();

this.Hide();

authorization.ShowDialog();

this.Close();

}

}

}

Файл «MasterChange.cs»:

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class MasterChange : Form

{

private int requestId;

private int userid;

private string userFio;

private DateTime? startDate;

private DateTime? completionDate;

private string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

public MasterChange(int requestId, int userid, string userFio)

{

InitializeComponent();

this.requestId = requestId;

this.userid = userid;

LoadComment();

LoadRequestDetails();

UpdateControlStates();

this.userFio = userFio;

}

private void LoadComment()

{

string query = @"

SELECT [message]

FROM Comments

WHERE requestID = @requestID AND staffID = @staffID";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestID", requestId);

command.Parameters.AddWithValue("@staffID", userid);

object result = command.ExecuteScalar();

if (result != null)

{

richTextBox2.Text = result.ToString();

}

else

{

richTextBox2.Text = "";

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки комментария: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UpdateControlStates()

{

bool hasStartDate = startDate.HasValue;

bool isInProgress = GetRequestStatus()==2;

button3.Enabled = !hasStartDate;

button4.Enabled = isInProgress;

richTextBox1.ReadOnly = isInProgress;

}

private int GetRequestStatus()

{

string query = "SELECT requestStatusID FROM Requests WHERE ID = @requestId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestId", requestId);

object result = command.ExecuteScalar();

if (result != null && int.TryParse(result.ToString(), out int statusId))

{

return statusId;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка получения статуса заявки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return -1;

}

private void UpdateRequestStatus(int statusId, bool setCompletionDate, bool setStartDate = false)

{

string updateRequestQuery = @"

UPDATE Requests

SET

requestStatusID = @statusID,

completionDate = @completionDate" +

(setStartDate ? ", startDate = @startDate" : "") + @"

WHERE ID = @requestID";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand updateCommand = new SqlCommand(updateRequestQuery, connection))

{

updateCommand.Parameters.AddWithValue("@statusID", statusId);

updateCommand.Parameters.AddWithValue("@requestID", requestId);

if (setCompletionDate)

{

updateCommand.Parameters.AddWithValue("@completionDate", DateTime.Now);

}

else

{

updateCommand.Parameters.AddWithValue("@completionDate", DBNull.Value);

}

if (setStartDate)

{

updateCommand.Parameters.AddWithValue("@startDate", DateTime.Now);

}

updateCommand.ExecuteNonQuery();

}

MessageBox.Show("Статус заявки успешно обновлён!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка обновления статуса: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadRequestDetails()

{

string query = "SELECT startDate, completionDate, repairParts FROM Requests WHERE ID = @requestId";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestId", requestId);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

richTextBox1.Text = reader["repairParts"].ToString();

completionDate = reader["completionDate"] as DateTime?;

startDate = reader["startDate"] as DateTime?;

UpdateControlStates();

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonSave\_Click(object sender, EventArgs e)

{

SaveChanges();

this.Hide();

Master masterForm = new Master(userid, userFio);

masterForm.ShowDialog();

this.Close();

}

private void SaveChanges()

{

string selectCommentQuery = @"

SELECT COUNT(1)

FROM Comments

WHERE requestID = @requestID AND staffID = @staffID";

string insertCommentQuery = @"

INSERT INTO Comments ([message], staffID, requestID)

VALUES (@message, @staffID, @requestID)";

string updateCommentQuery = @"

UPDATE Comments

SET [message] = @message

WHERE requestID = @requestID AND staffID = @staffID";

string updateRepairPartsQuery = @"

UPDATE Requests

SET repairParts = @repairParts

WHERE ID = @requestID";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand selectCommand = new SqlCommand(selectCommentQuery, connection))

{

selectCommand.Parameters.AddWithValue("@requestID", requestId);

selectCommand.Parameters.AddWithValue("@staffID", userid);

int commentExists = (int)selectCommand.ExecuteScalar();

if (!string.IsNullOrWhiteSpace(richTextBox2.Text))

{

if (commentExists > 0)

{

using (SqlCommand updateCommentCommand = new SqlCommand(updateCommentQuery, connection))

{

updateCommentCommand.Parameters.AddWithValue("@message", richTextBox2.Text);

updateCommentCommand.Parameters.AddWithValue("@staffID", userid);

updateCommentCommand.Parameters.AddWithValue("@requestID", requestId);

updateCommentCommand.ExecuteNonQuery();

}

}

else

{

using (SqlCommand insertCommentCommand = new SqlCommand(insertCommentQuery, connection))

{

insertCommentCommand.Parameters.AddWithValue("@message", richTextBox2.Text);

insertCommentCommand.Parameters.AddWithValue("@staffID", userid);

insertCommentCommand.Parameters.AddWithValue("@requestID", requestId);

insertCommentCommand.ExecuteNonQuery();

}

}

}

}

if (!string.IsNullOrWhiteSpace(richTextBox1.Text))

{

using (SqlCommand updateRepairPartsCommand = new SqlCommand(updateRepairPartsQuery, connection))

{

updateRepairPartsCommand.Parameters.AddWithValue("@repairParts", richTextBox1.Text);

updateRepairPartsCommand.Parameters.AddWithValue("@requestID", requestId);

updateRepairPartsCommand.ExecuteNonQuery();

}

}

}

MessageBox.Show("Данные успешно сохранены!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3\_Click(object sender, EventArgs e)

{

UpdateRequestStatus(2, false, true);

}

private void button4\_Click(object sender, EventArgs e)

{

UpdateRequestStatus(1, true);

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

Master masterForm = new Master(userid, userFio);

masterForm.ShowDialog();

this.Close();

}

}

}

Файл «Operator.cs»:

using System;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class Operator : Form

{

string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

private int userId;

private string userFio;

private int totalRequests;

private int currentRequests;

public Operator(int userId, string userFio)

{

InitializeComponent();

LoadTotalRequests();

this.userId = userId;

FillDataGridView();

InitializeDataGridView();

dataGridView3.CellClick += dataGridView3\_CellDoubleClick;

this.userFio = userFio;

pictureBox1.Image = Image.FromFile(@"D:\4 курс\УП 02.01\curs\_trsbd\curs trsbd\curs trsbd\Resources\progile.jpg");

label1.Text = userFio;

currentRequests = dataGridView3.Rows.Count;

label3.Text = $"{currentRequests}/{totalRequests}";

}

private void LoadTotalRequests()

{

string query = "SELECT COUNT(\*) FROM Requests";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

totalRequests = (int)command.ExecuteScalar();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении общего количества заявок: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UpdateAverageCompletionTime()

{

try

{

if (dataGridView3.Rows.Count > 0)

{

TimeSpan totalDuration = TimeSpan.Zero;

int completedCount = 0;

foreach (DataGridViewRow row in dataGridView3.Rows)

{

if (row.Cells["completionDate"].Value != DBNull.Value && row.Cells["startDate"].Value != DBNull.Value)

{

DateTime startDate = Convert.ToDateTime(row.Cells["startDate"].Value);

DateTime completionDate = Convert.ToDateTime(row.Cells["completionDate"].Value);

totalDuration += (completionDate - startDate);

completedCount++;

}

}

if (completedCount > 0)

{

double avgDuration = totalDuration.TotalDays / completedCount;

label2.Text = $"Среднее время выполнения: {avgDuration:F1} дней";

}

else

{

label2.Text = "Среднее время выполнения: нет завершенных заявок";

}

}

else

{

label2.Text = "Среднее время выполнения: нет данных";

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при вычислении среднего времени выполнения: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void InitializeDataGridView()

{

dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView3.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

dataGridView3.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

foreach (DataGridViewColumn column in dataGridView3.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

dataGridView3.ReadOnly = true;

dataGridView3.AllowUserToAddRows = false;

dataGridView3.RowHeadersVisible = false;

dataGridView3.AllowUserToResizeColumns = false;

}

public void FillDataGridView()

{

try

{

string query = @"SELECT

r.ID,

r.startDate,

t.climateTechType,

t.climateTechModel,

r.problemDescryption,

rs.requestStatus,

r.completionDate,

r.repairParts,

s.fio AS staffFio,

c.fio AS clientFio

FROM

Requests r

LEFT JOIN

Technic t ON r.techincID = t.ID

LEFT JOIN

RequestStatus rs ON r.requestStatusID = rs.ID

LEFT JOIN

Staff s ON r.staffID = s.ID

LEFT JOIN

Client c ON r.clientID = c.ID

ORDER BY

r.ID;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

dataAdapter.Fill(dataTable);

label2.Text = AvarageTime.UpdateAverageCompletionTime(dataTable);

bindingSource.DataSource = dataTable;

dataGridView3.DataSource = bindingSource;

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView3\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

int requestId = Convert.ToInt32(dataGridView3.Rows[e.RowIndex].Cells["ID"].Value);

this.Hide();

OperatorChange operatorChange = new OperatorChange(requestId, userId, userFio);

operatorChange.ShowDialog();

this.Close();

FillDataGridView();

}

}

private void button3\_Click(object sender, EventArgs e)

{

Logs logs = new Logs();

logs.ShowDialog();

}

private void button1\_Click(object sender, EventArgs e)

{

Authorization authorization = new Authorization();

this.Hide();

authorization.ShowDialog();

this.Close();

}

private void bindingSource\_ListChanged(object sender, ListChangedEventArgs e)

{

try

{

if (dataGridView3.DataSource is BindingSource dataTable)

{

label3.Text = $"{bindingSource.Count}/{totalRequests}";

}

else

{

currentRequests = 0;

label3.Text = $"0/{totalRequests}";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении строк после фильтрации: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Файл «OperatorChange.cs»:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class OperatorChange : Form

{

private int requestId;

private int userId;

private string userFio;

private string connectionString = $"Server=ADCLG1;Database=\_UP\_Chalikyan;Trusted\_connection=true;Encrypt=false;Integrated Security=true;";

public OperatorChange(int requestId, int userId, string userFio)

{

InitializeComponent();

this.requestId = requestId;

this.userId = userId;

LoadStaffComboBox();

LoadRequestDetails();

this.userFio = userFio;

}

private void LoadStaffComboBox()

{

string query = "SELECT ID, fio FROM Staff";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable staffTable = new DataTable();

dataAdapter.Fill(staffTable);

comboBox1.DisplayMember = "fio";

comboBox1.ValueMember = "ID";

comboBox1.DataSource = staffTable;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки списка сотрудников: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void LoadRequestDetails()

{

string query = @"

SELECT

r.completionDate,

c.[message]

FROM

Requests r

LEFT JOIN

Comments c ON r.ID = c.requestID

WHERE

r.ID = @requestID";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@requestID", requestId);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

richTextBox1.Text = reader.IsDBNull(1) ? "Комментарий отсутствует" : reader.GetString(1);

}

}

}

}

richTextBox1.ReadOnly = true;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonSave\_Click(object sender, EventArgs e)

{

string updateQuery = @"

UPDATE Requests

SET

staffID = @staffID

WHERE ID = @requestID";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(updateQuery, connection))

{

command.Parameters.AddWithValue("@staffID", comboBox1.SelectedValue);

command.Parameters.AddWithValue("@requestID", requestId);

command.ExecuteNonQuery();

}

}

MessageBox.Show("Изменения успешно сохранены.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Hide();

Operator operatorForm = new Operator(userId, userFio);

operatorForm.ShowDialog();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка сохранения данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

Operator operatorForm = new Operator(userId, userFio);

operatorForm.ShowDialog();

this.Close();

}

}

}

Файл «QR.cs»:

using QRCoder;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace curs\_trsbd

{

public partial class QR : Form

{

private string url = "https://code-qr.ru/blog/qr-code-survey";

public QR()

{

InitializeComponent();

GenerateQRCode(url);

}

private void GenerateQRCode(string url)

{

try

{

using (QRCodeGenerator qrGenerator = new QRCodeGenerator())

{

QRCodeData qrCodeData = qrGenerator.CreateQrCode(url, QRCodeGenerator.ECCLevel.Q);

using (QRCode qrCode = new QRCode(qrCodeData))

{

int pictureBoxWidth = pictureBox1.Width;

int pictureBoxHeight = pictureBox1.Height;

Bitmap qrCodeImage = qrCode.GetGraphic(20, Color.Black, Color.White, true);

Bitmap resizedQRCodeImage = new Bitmap(qrCodeImage, new Size(pictureBoxWidth, pictureBoxHeight));

pictureBox1.Image = resizedQRCodeImage;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при генерации QR-кода: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

# Приложение Г

(справочное)

**Данные тестирования**

В таблице Г.1 представлен заголовок тестовых сценариев.

Таблица Г.1 – Заголовок тестовых сценариев

|  |  |
| --- | --- |
| **Название проекта** | Программа для учёта заявок на ремонт климатического оборудования |
| **Рабочая версия** | 0.1 |
| **Имя тестирующего** | Чаликян Арен Оганнесович |
| **Дата(ы) теста** | 18.12.2024 |

В таблицах Г.2 – Г.6 представлены тестовые сценарии.

Таблица Г.2 – Тест кейс 1

|  |  |
| --- | --- |
| **Тест кейс #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации с корректными данными |
| **Краткое изложение теста** | Тест проверяет успешный вход пользователя в систему с корректными логином и паролем |
| **Этапы теста** | 1. Запустить приложение 2. Ввести корректные логин и пароль 3. Нажать кнопку "Войти" |
| **Тестовые данные** | Логин: login1  Пароль: pass1 |

Продолжение таблицы Г.2

|  |  |
| --- | --- |
| **Ожидаемый результат** | Появляется сообщение об успешной авторизации и открывается окно клиента. |
| **Фактический результат** | Появляется сообщение об успешной авторизации и открывается окно клиента. |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль пользователя внесены в базу данных |
| **Постусловие** | Пользователю доступен функционал роли клиента |
| **Примечания/комментарии** | Нет |

Таблица Г.3 – Тест кейс 2

|  |  |
| --- | --- |
| **Тест кейс #** | 2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации с некорректными данными |
| **Краткое изложение теста** | Тест проверяет реакцию системы при вводе неверного логина или пароля |
| **Этапы теста** | 1. Запустить приложение 2. Ввести некорректный логин или пароль 3. Нажать кнопку "Войти" |
| **Тестовые данные** | Логин: login99  Пароль: password99 |
| **Ожидаемый результат** | Появляется сообщение об ошибке "Неверный логин или пароль". |

Продолжение таблицы Г.3

|  |  |
| --- | --- |
| **Фактический результат** | Появляется сообщение об ошибке "Неверный логин или пароль". |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль пользователя не внесены в базу данных |
| **Постусловие** | При последующих попытках ввода необходимо ввести капчу |
| **Примечания/комментарии** | Нет |

Таблица Г.4 – Тест кейс 3

|  |  |
| --- | --- |
| **Тест кейс #** | 3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка авторизации после блокировки аккаунта |
| **Краткое изложение теста** | Тест проверяет возможность входа после блокировки пользователя из-за многократных неудачных попыток авторизации |
| **Этапы теста** | 1. Ввести данные 2. Нажать кнопку "Войти"   Проверить блокировку аккаунта |
| **Тестовые данные** | Логин: login99  Пароль: pass99 |

Продолжение таблицы Г.4

|  |  |
| --- | --- |
| **Ожидаемый результат** | Появляется сообщение об ошибке "Неверный логин или пароль" и сообщение о блокировки системы на 3 минуты. |
| **Фактический результат** | Появляется сообщение об ошибке "Неверный логин или пароль" и сообщение о блокировки системы на 3 минуты. |
| **Статус** | Зачет |
| **Предварительное условие** | Логин и пароль пользователя не внесены в базу данных и было выполнено 3 неправильные попытки входа |
| **Постусловие** | Вход заблокирован |
| **Примечания/комментарии** | Нет |

Таблица Г.5 – Тест кейс 4

|  |  |
| --- | --- |
| **Тест кейс #** | 4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка функциональности создания заявки |
| **Краткое изложение теста** | Тест проверяет корректность работы формы создания заявки |

Продолжение таблицы Г.5

|  |  |
| --- | --- |
| **Этапы теста** | 1. Заполнить данные заявки 2. Нажать на кнопку «Добавить» |
| **Тестовые данные** | Модель техники: TCL TAC-12CHSA/TPG-W белый  Проблема: не работает |
| **Ожидаемый результат** | Заявка успешно создана, сообщение о добавлении заявки, обновление списка заявок |
| **Фактический результат** | Заявка успешно создана, сообщение о добавлении заявки, обновление списка заявок |
| **Статус** | Зачет |
| **Предварительное условие** | Вход в систему под ролью заказчика. |
| **Постусловие** | Другие пользователи могут работать с созданной заявкой |
| **Примечания/комментарии** | Нет |

Таблица Г.6 – Тест кейс 5

|  |  |
| --- | --- |
| **Тест кейс #** | 5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка редактирования заявки заказчиком |

Продолжение таблицы Г.6

|  |  |
| --- | --- |
| **Краткое изложение теста** | Тест проверяет возможность редактирования заявки заказчиком |
| **Этапы теста** | 1. Нажать на строчку с заявкой 2. Изменить данные 3. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Статус «В ожидании» |
| **Ожидаемый результат** | Заявка успешно изменена, сообщение о завершении удаления, обновление списка заявок |
| **Фактический результат** | Заявка успешно изменена, сообщение о завершении удаления, обновление списка заявок |
| **Статус** | Зачет |
| **Предварительное условие** | Вход в систему под ролью заказчика, в системе есть заявки со статусом «В ожидании» |
| **Постусловие** | Другие пользователи могут работать с изменённой заявкой |
| **Примечания/комментарии** | Заявка изменена |

# Приложение Д

(справочное)

**Код unit-тестов**

Файл «UnitTest1»:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using curs\_trsbd;

using System;

using System.Drawing;

namespace Tests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void AuthenticateUser\_ValidStaffCredentials\_ReturnsAccessLevel()

{

var auth = new Authorization();

int userId;

string userFio;

string login = "login2";

string password = "pass2";

string result = auth.AuthenticateUser(login, password, out userId, out userFio);

Assert.IsNotNull(result, "Access level should not be null for valid staff credentials.");

Assert.AreEqual("мастер", result, "Expected access level for valid staff is 'полный'.");

}

[TestMethod]

public void AuthenticateUser\_InvalidCredentials\_ReturnsNull()

{

var auth = new Authorization();

int userId;

string userFio;

string login = "invalid\_login";

string password = "invalid\_password";

string result = auth.AuthenticateUser(login, password, out userId, out userFio);

Assert.IsNull(result, "Access level should be null for invalid credentials.");

}

[TestMethod]

public void LogLoginAttempt\_ValidLogin\_SuccessLogged()

{

var auth = new Authorization();

string login = "test\_user";

bool success = true;

try

{

auth.LogLoginAttempt(login, success);

Assert.IsTrue(true, "LogLoginAttempt executed without exceptions.");

}

catch (Exception ex)

{

Assert.Fail($"LogLoginAttempt threw an exception: {ex.Message}");

}

}

[TestMethod]

public void CheckLoginExists\_ExistingLogin\_ReturnsTrue()

{

var auth = new Authorization();

string existingLogin = "login1";

bool exists = auth.CheckLoginExists(existingLogin);

Assert.IsTrue(exists, "CheckLoginExists should return true for existing login.");

}

[TestMethod]

public void CreateImage\_ValidDimensions\_ReturnsBitmap()

{

var auth = new Authorization();

int width = 100;

int height = 50;

Bitmap image = auth.CreateImage(width, height);

Assert.IsNotNull(image, "CreateImage should return a non-null Bitmap.");

Assert.AreEqual(width, image.Width, "Image width does not match input width.");

Assert.AreEqual(height, image.Height, "Image height does not match input height.");

}

}

}

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/3 группы

Чаликян Арен Оганнесович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, Приморский пр. 63

(наименование и адрес организации)

Период прохождения практики

с «9» декабря 2024 г. по «21» декабря 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| Участие в выработке требований к программному  обеспечению | 9 |  |
| Стадии проектирования программного обеспечения | 21 |  |
| Разработка модулей программного обеспечения | 24 |  |
| Тестирование программных модулей и их интеграции | 9 |  |
| Разработка программной документации и стандарты кодирования | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Шаврова Л.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «21» декабря 2024 г.