

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	2
1    Общая часть.....	3
1.1  Проектирование системы.....	3
1.2  Создание базы данных и заполнение таблиц данными .....	14
1.3  Разработка библиотеки и подключение её к проекту .....	19
1.4  Разработка приложения.....	19
1.5  Тестирование приложения.....	26
1.6  Выгрузка готового проекта в репозиторий Git.....	27
ЗАКЛЮЧЕНИЕ .....	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	29
ПРИЛОЖЕНИЕ А (справочное) Таблицы тестирования .....	30
ПРИЛОЖЕНИЕ Б (справочное) Исходный код.....	34

## ВВЕДЕНИЕ

Была разработана информационная система под названием «Информационная система обслуживания оргтехники». Основное назначение данного программного обеспечения – автоматизировать процессы сбора и хранения данных, связанных с заявками на ремонт офисного оборудования, а также учет информации о клиентах и сотрудниках.

Цель создания системы заключается в упрощении и оптимизации процесса управления заявками, что позволит более эффективно взаимодействовать с клиентами и мастерами по обслуживанию. Система предоставляет возможность фиксировать все этапы выполнения заявок на ремонт, отслеживать их статус и хранить информацию о каждом устройстве.

В ходе учебной практики предстоит проанализировать предметную область сервиса для обслуживания оргтехники. В рамках работы будет осуществлена разработка технического задания и прочей документации. Кроме этого, будет уделено время проектированию UML-диаграмм. Особое внимание будет уделено проектированию пользовательского интерфейса и разработке дизайна программы в соответствии с руководством по стилю. В процессе работы будет разработана база данных. Предстоит создать приложение на языке программирования C#, включая формы авторизации, основные формы и библиотеку классов. Помимо этого, предстоит создать тест-кейсы для проверки функциональности приложения. Завершится работа отладкой программных модулей для дальнейшей выгрузки программных модулей на Git.

# **1      Общая часть**

## **1.1   Проектирование системы**

### **1.1.1 Краткая спецификация**

Программный модуль для учета заявок на ремонт оргтехники предназначен для автоматизации процесса приема и обработки заявок на ремонт от сотрудников офиса или других пользователей. Он позволяет упростить и ускорить процесс решения проблем с оргтехникой.

Основные функции и возможности программного изделия:

- 1)     заявка на ремонт;
- 2)     регистрация заявки;
- 3)     обработка заявки;
- 4)     исполнение заявки;
- 5)     отчётность и информирование;
- 6)     мониторинг и анализ.

Нефункциональные требования к системе:

- 1)     безопасность;
- 2)     надежность;
- 3)     производительность;
- 4)     пользовательский интерфейс;
- 5)     масштабируемость.

Требования к документации:

- 1)     техническое задание (ТЗ);
- 2)     руководство системного программиста.

### 1.1.2 Диаграмма вариантов использования

Диаграмма прецедентов используется для визуального отображения взаимодействия пользователей (актеров) с системой. (Рисунок 1).

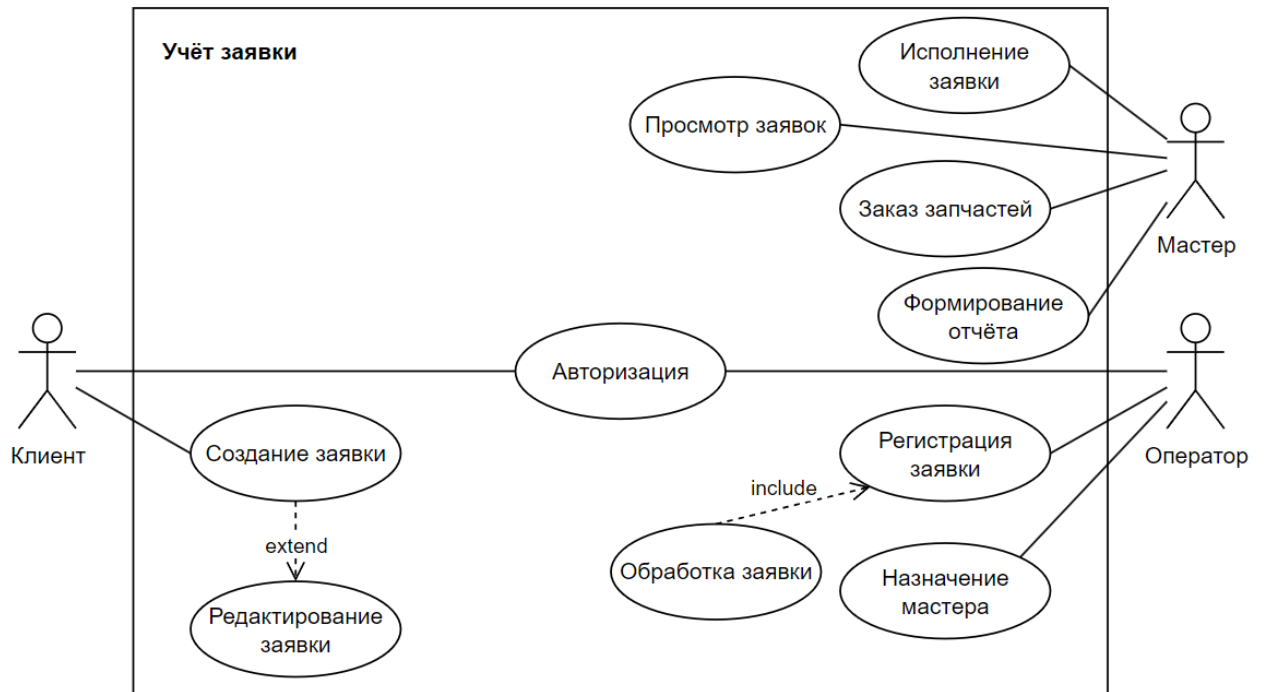


Рисунок 1 – Диаграмма прецедентов

### 1.1.3 Диаграмма активности

Диаграмма активности моделирует последовательность действий или процессов. Данная диаграмма показывает процесс создания новой заявки (Рисунок 2).

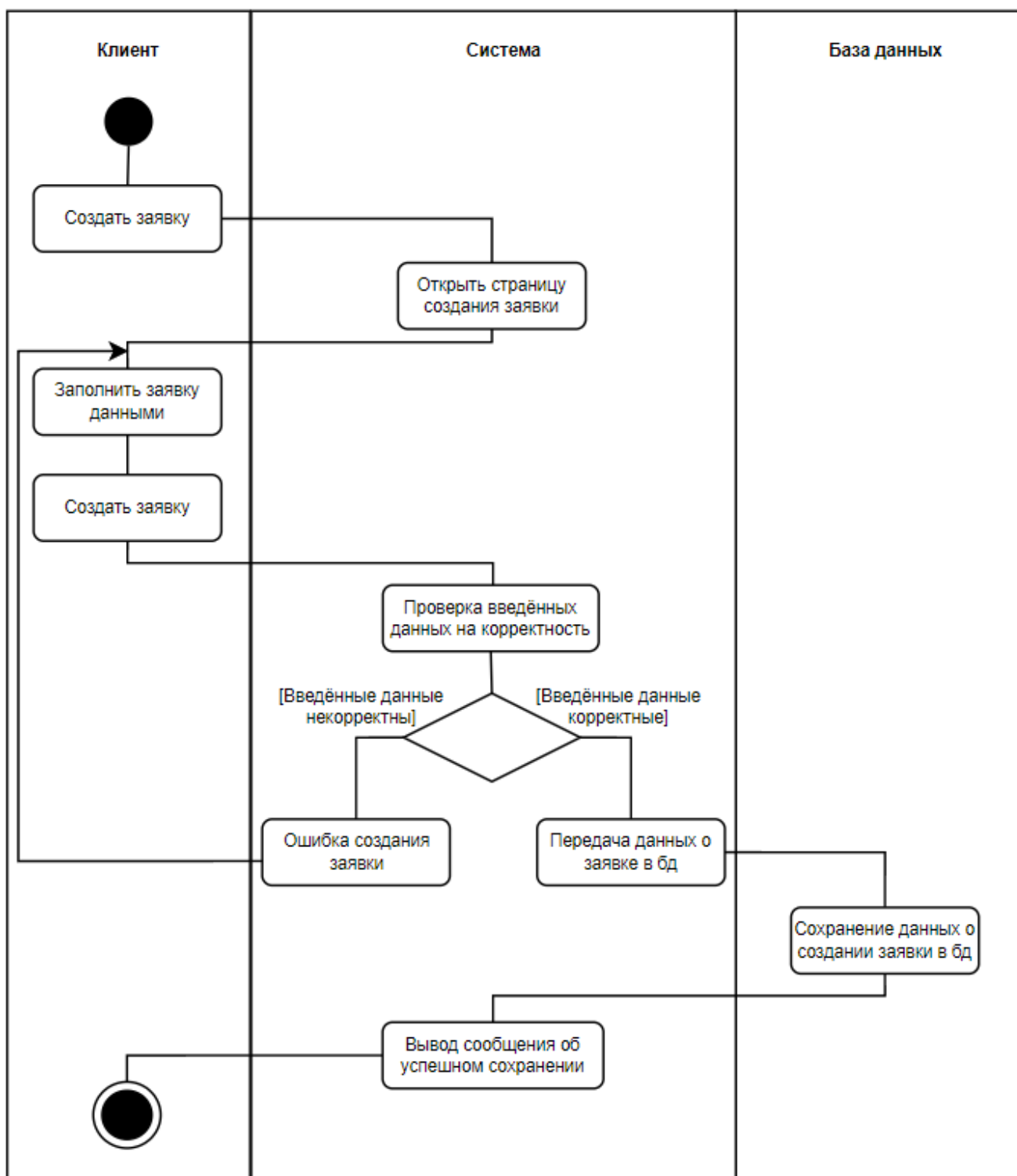


Рисунок 2 – Диаграмма активности

### 1.1.4 Диаграмма последовательности

Диаграмма последовательности иллюстрирует, как оператор взаимодействует с системой и базой данных в процессе обработки заявки на ремонт (Рисунок 3).

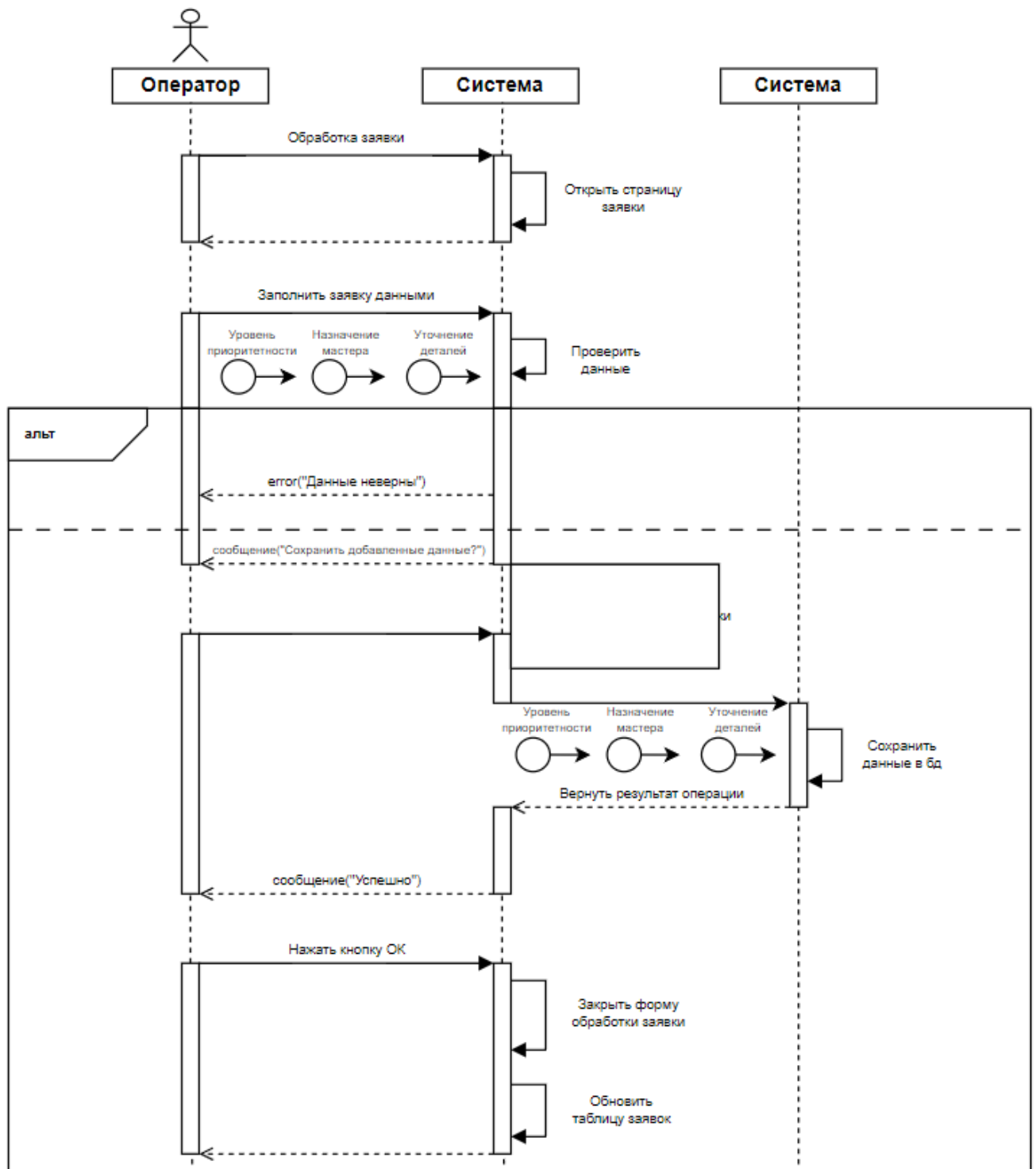


Рисунок 3 – Диаграмма последовательности

### 1.1.5 Общий алгоритм

Основной алгоритм системы учета заявок на ремонт в виде блок-схемы описывает последовательность действий пользователя и его взаимодействия с системой, а также доступные им функции (Рисунок 4).

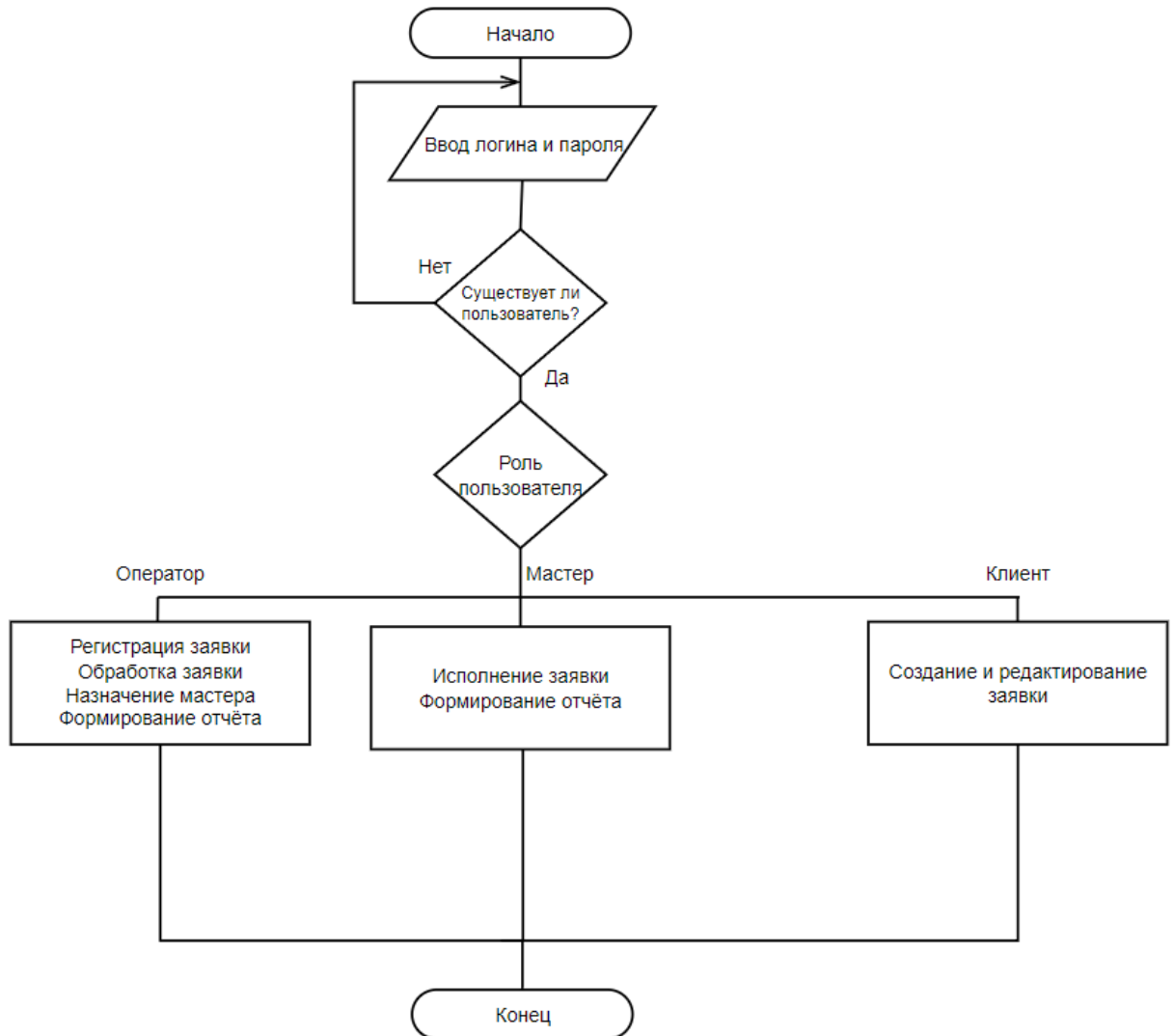


Рисунок 4 – Основной алгоритм

### 1.1.6 Алгоритм функции расчета количества заявок

Подробный алгоритм функции расчета количества заявок (Рисунок 5).

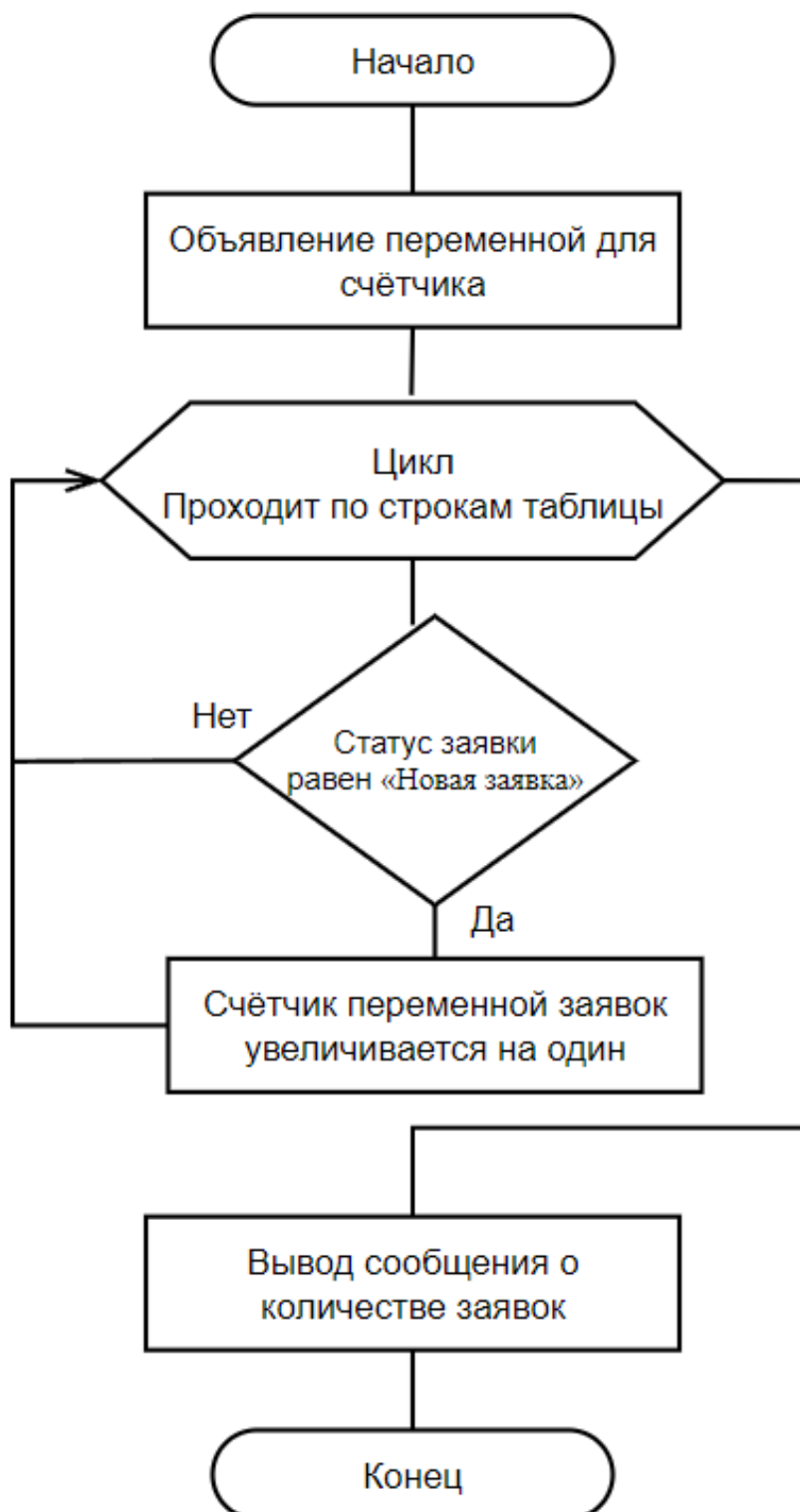
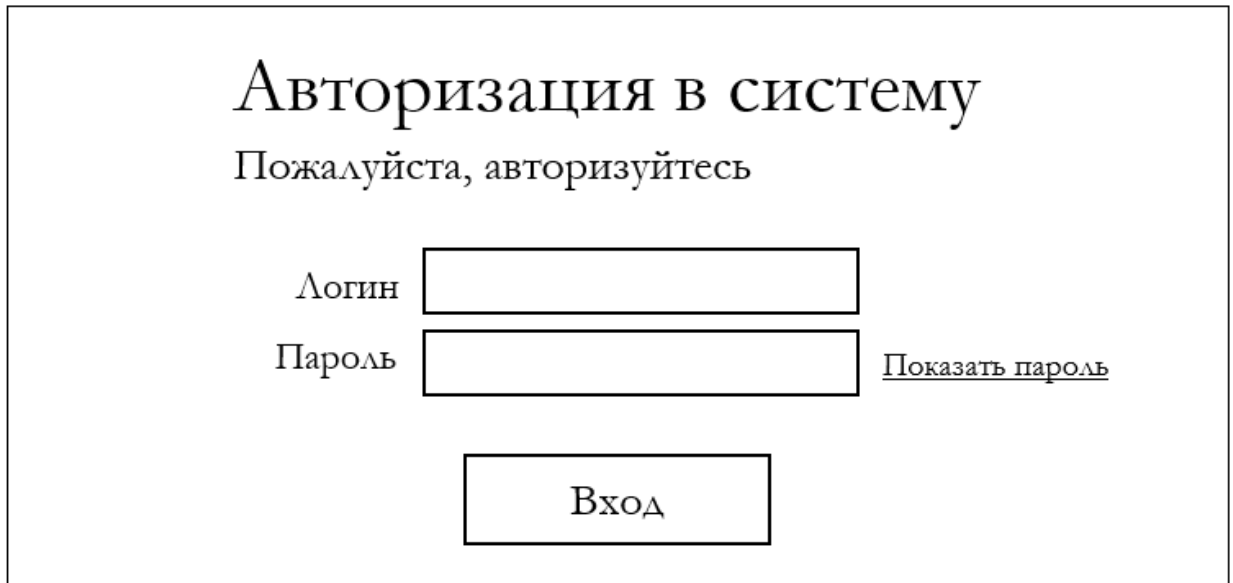


Рисунок 5 – Функция расчета количества заявок



### 1.1.7 Разработка макетов интерфейса системы

Макеты окна авторизации (Рисунки 6-7).



Авторизация в систему

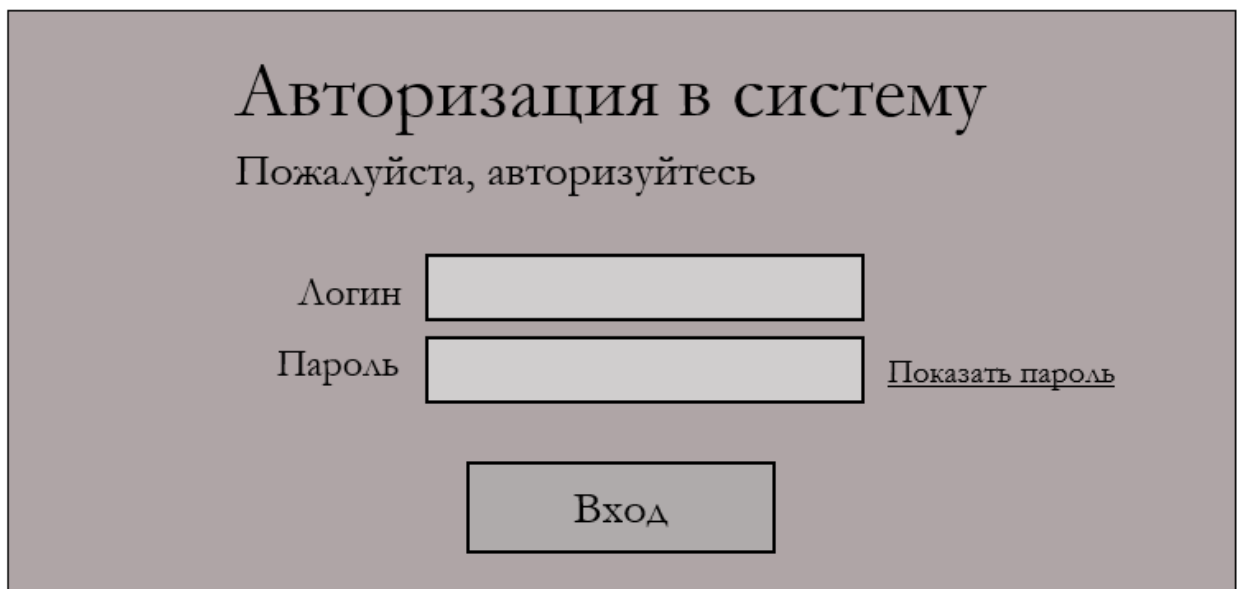
Пожалуйста, авторизуйтесь

Логин

Пароль  [Показать пароль](#)

This wireframe mockup shows a simple layout for an authorization window. It features a title 'Авторизация в систему' and a subtitle 'Пожалуйста, авторизуйтесь'. Below these are two input fields: one for 'Логин' (Login) and one for 'Пароль' (Password). The password field is accompanied by a link 'Показать пароль' (Show password). At the bottom is a button labeled 'Вход' (Login).

Рисунок 6 – Wireframes макет окна авторизации



Авторизация в систему

Пожалуйста, авторизуйтесь

Логин

Пароль  [Показать пароль](#)

This mockup is similar to the wireframe but has a solid grey background. It contains the same text and form elements: the title 'Авторизация в систему', subtitle 'Пожалуйста, авторизуйтесь', login and password input fields with the 'Показать пароль' link, and the 'Вход' button.

Рисунок 7 – Mockups макет окна авторизации

Макеты главной страницы клиента (Рисунки 8-9).

Добро пожаловать!
История ваших заявок

Вернуться назад

Код заявки	Дата создания	Техника	Описание проблемы	Статус заявки	Дата окончания работ	Комментарий

Оставить заявку

Удалить заявку

Редактировать заявку

Рисунок 8 – Wireframes макет главной страницы заказчика

Добро пожаловать!
История ваших заявок

Вернуться назад

Код заявки	Дата создания	Техника	Описание проблемы	Статус заявки	Дата окончания работ	Комментарий
1	2024-05-07	Принтер HP	Громко гудит	Новая заявка		

Оставить заявку

Удалить заявку

Редактировать заявку

Рисунок 9 – Mockups макет главной страницы заказчика

Макеты главной страницы оператора (Рисунки 10-11).

Добро пожаловать!
Заявки клиентов

Вернуться назад

Код заявки	Дата создания	Техника	Описание проблемы	ФИО клиента	Телефон клиента

Отклонить заявку

Обработать заявку

Рисунок 10 – Wireframes макет главной страницы оператора

Добро пожаловать!
Заявки клиентов

Вернуться назад

Код заявки	Дата создания	Техника	Описание проблемы	ФИО клиента	Телефон клиента
1	2024-05-07	Принтер HP	Громко гудит	Морозов Александр Михайлович	89117451205

Отклонить заявку

Обработать заявку

Рисунок 11 – Mockups макет главной страницы оператора

Макеты главной страницы мастера (Рисунки 12-13).

Добро пожаловать!
Активные заявки

Вернуться назад

Код заявки	Дата создания	Техника	Описание проблемы	ФИО клиента	Телефон клиента

Сформировать отчёт о выполняемых работ

Рисунок 12 – Wireframes макет главной страницы мастера

Добро пожаловать!
Активные заявки

Вернуться назад

Код заявки	Дата создания	Техника	Описание проблемы	ФИО клиента	Телефон клиента
2	2024-12-06	Ноутбук ASUS	Не запускается	Фараонов Глеб Геннадьевич	81111112233

Сформировать отчёт о выполняемых работ

Рисунок 13 – Mockups макет главной страницы мастера

Макеты страницы создания новой заявки (Рисунки 14-15).

Создание новой заявки

Выберите модель техники:

Опишите как можно подробнее проблему:

Оставить заявку

This wireframe shows a form for creating a new request. It has a title 'Создание новой заявки', a label 'Выберите модель техники:' followed by a text input field, a label 'Опишите как можно подробнее проблему:' followed by a large text area, and a button 'Оставить заявку' at the bottom.

Рисунок 14 – Wireframes макет создания новой заявки

Создание новой заявки

Выберите модель техники:

Моноблок игровой ASUS

Опишите как можно подробнее проблему:

Горит синий экран смерти

Оставить заявку

This annotated wireframe shows the same form as Figure 14, but with sample data. The text input field contains 'Моноблок игровой ASUS' and the text area contains 'Горит синий экран смерти'. The button 'Оставить заявку' is at the bottom.

Рисунок 15 – Mockups макет создания новой заявки

## 1.2 Создание базы данных и заполнение таблиц данными

Разработка скрипта для создания таблиц в БД (ПРИЛОЖЕНИЕ).

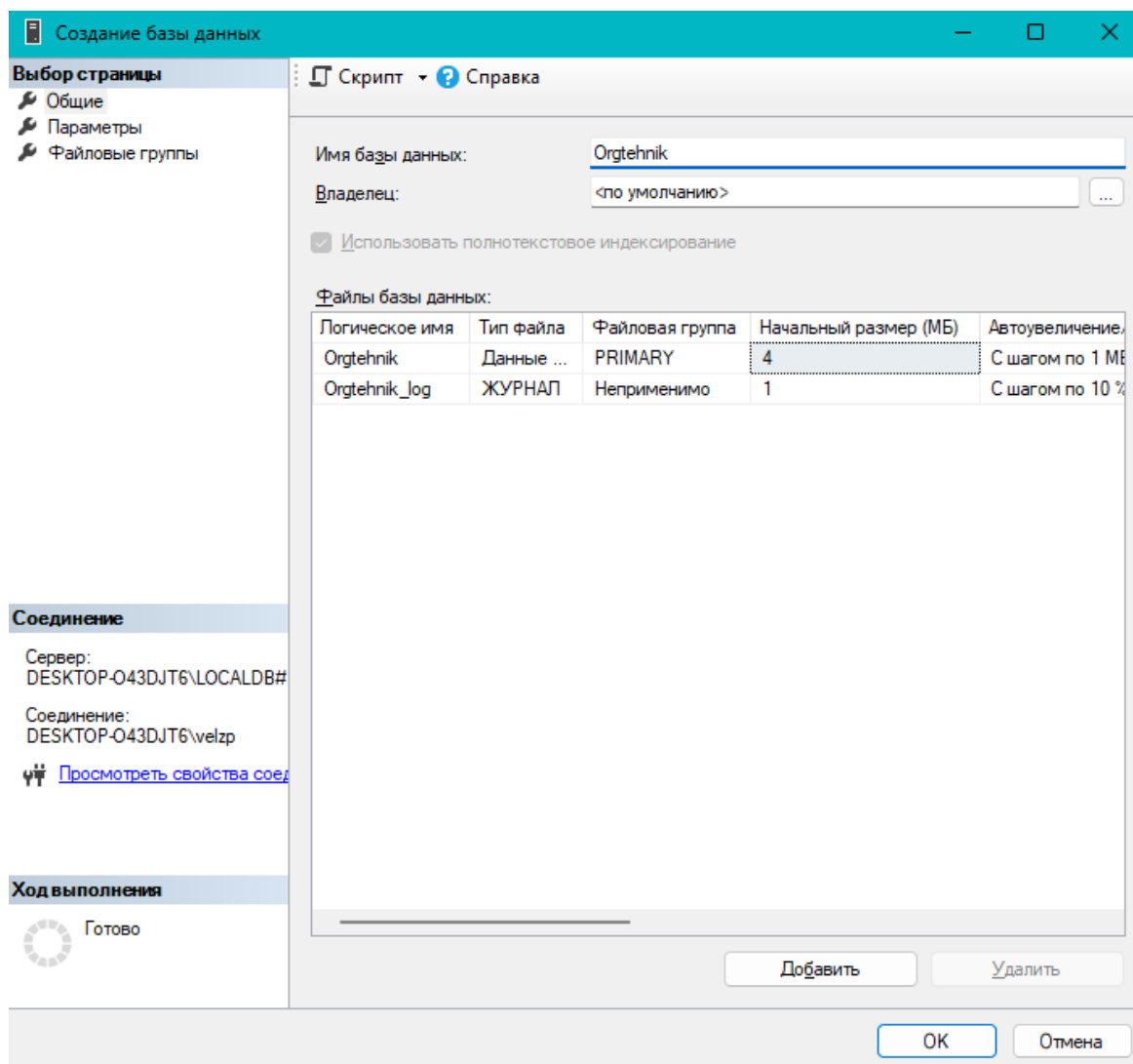


Рисунок 16 – создание базы данных в Microsoft SQL Server.

### 1.2.1 ER-диаграмма

ER-диаграмма представляет собой визуальное отображение структуры базы данных для системы учета заявок на ремонт оргтехники. Она иллюстрирует ключевые сущности, такие как пользователи, заявки, модели и типы техники, а также взаимосвязи между ними. Каждая сущность соединена

с другими через внешние ключи. Это позволяет эффективно управлять информацией и обеспечивает возможность выполнения сложных запросов к базе данных (Рисунок 17).

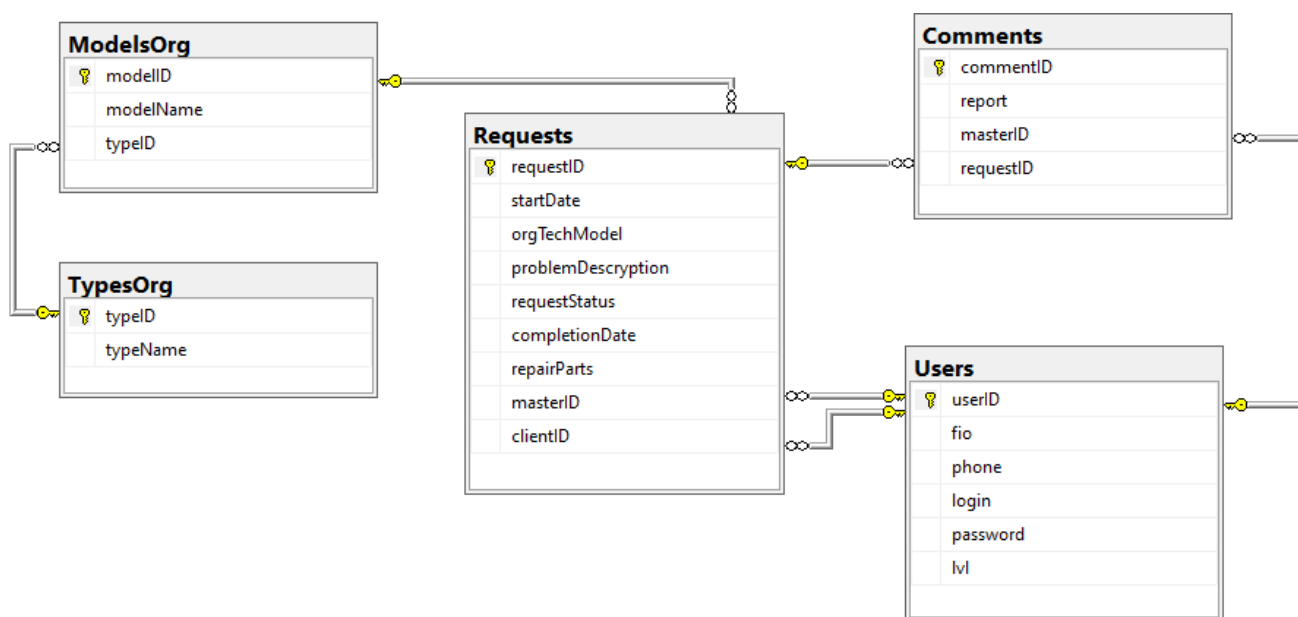


Рисунок 17 – ER-диаграмма

## 1.2.2 Словарь данных

Таблица 1 – Таблица Users

Ключ	Атрибут	Тип данных	Комментарий
Первичный	userID	INT	Код пользователя
	fio	VARCHAR(50)	Автоинкремент
	phone	VARCHAR(11)	
	login	VARCHAR(25)	
	password	VARCHAR(25)	
	lvl	VARCHAR(25)	

Таблица 2 – Таблица Requests

Ключ	Атрибут	Тип данных	Комментарий
Первичный	requestID	INT	Автоинкремент
	startDate	DATETIME	
	orgTechModel	INT	
	problemDescription	VARCHAR(100)	
	requestStatus	NVARCHAR(30)	
	completionDate	DATETIME	

Продолжение таблицы 2

	repairParts	VARCHAR(150)	
	masterID	INT	
	clientID	INT	

Таблица 3 – Таблица Comments

Ключ	Атрибут	Тип данных	Комментарий
Первичный ключ	commentID	INT	Автоинкремент
	message	VARCHAR(150)	
Внешний ключ на Users	masterID	INT	
Внешний ключ на Requests	requestID	INT	

Таблица 4 – Таблица ModelsOrg

Ключ	Атрибут	Тип данных	Комментарий
Первичный ключ	modelID	INT	Автоинкремент
	modelName	VARCHAR (150)	
Внешний ключ на TypesOrg	typeID	INT	

Таблица 5 – Таблица TypesOrg

Ключ	Атрибут	Тип данных	Комментарий
Первичный ключ	typeID	INT	Автоинкремент
	typeName	VARCHAR (50)	
FK	requestID	INT	Код заявки



### 1.2.3 Заполненные данными таблицы

Таблицы с данными (Рисунки 18-22).

SQLQuery13.sql - (...O43DJT6\velzp (58)) DESKTOP-O43DJT6\L...hnika - dbo.Users						
	userID	fio	phone	login	password	lvl
	1	Носов Иван Михайлович	89210563128	login1	pass1	Менеджер
	2	Ильин Александр Андреев...	89535078985	login2	pass2	Мастер
	3	Никифоров Иван Дмитрие...	89210673849	login3	pass3	Мастер
	4	Елисеев Артём Леонидович	89990563748	login4	pass4	Оператор
	5	Титов Сергей Кириллович	89994563847	login5	pass5	Оператор
	6	Григорьев Семён Викторо...	89219567849	login11	pass11	Клиент
	7	Сорокин Дмитрий Ильич	89219567841	login12	pass12	Клиент
▶	8	Белоусов Егор Ярославович	89219567842	login13	pass13	Клиент
	9	Суслов Михаил Александр...	89219567843	login14	pass14	Клиент
	10	Васильев Вячеслав Алекса...	89219567844	login15	pass15	Мастер
*	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 18 – Таблица Users

DESKTOP-O43DJT6\L...ka - dbo.TypesOrg		
	typeID	typeName
▶	1	Компьютер
	2	Ноутбук
	3	Принтер
*	NULL	NULL

Рисунок 19 – Таблица TypesOrg

DESKTOP-O43DJT6\...a - dbo.ModelsOrg DESKTOP-O43DJT6\L...ka - dbo.ModelsOrg			
	modelID	modelName	typeID
▶	1	DEXP Aquilon ...	1
	2	DEXP Atlas H388	1
	3	MSI GF76 Katan...	2
	4	MSI Modern 15 ...	2
	5	HP LaserJet Pro...	3
*	NULL	NULL	NULL

Рисунок 20 – Таблица ModelsOrg

requestID	startDate	orgTechModel	problemDescr...	requestStatus	completionDate	repairParts	masterID	clientID
1	2023-06-06 00:0...	1	Перестал рабо...	В процессе	NULL	NULL	2	1
2	2023-05-05 00:0...	1	Перестал рабо...	В процессе	NULL	NULL	3	1
3	2022-07-07 00:0...	2	Выключается	В процессе	NULL	NULL	2	1
4	2023-08-02 00:0...	2	Выключается	Новая заявка	NULL	NULL	NULL	3
5	2023-08-02 00:0...	3	Перестал вкл...	Новая заявка	NULL	NULL	NULL	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 21 – Таблица Requests

commentID	message	masterID	requestID
2	Интересно...	2	1
4	Будем разбираться!	3	2
5	Сделаем всё на высшем уров...	2	3
6	NULL	NULL	4
7	NULL	NULL	5
NULL	NULL	NULL	NULL

Рисунок 22 – Таблица Comments

## 1.2.4 Резервное копирование

Создание резервной копии базы данных (Рисунки 23-24).

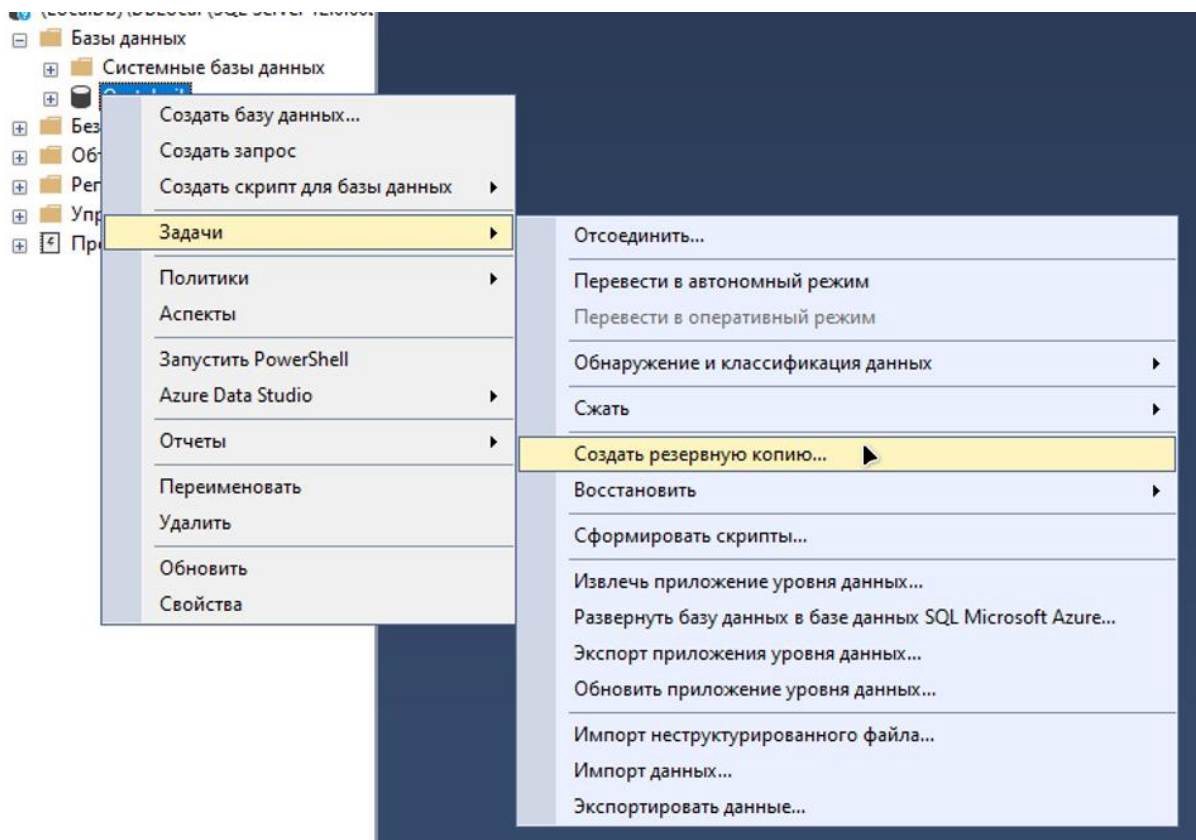


Рисунок 23 – Создание резервной копии базы данных

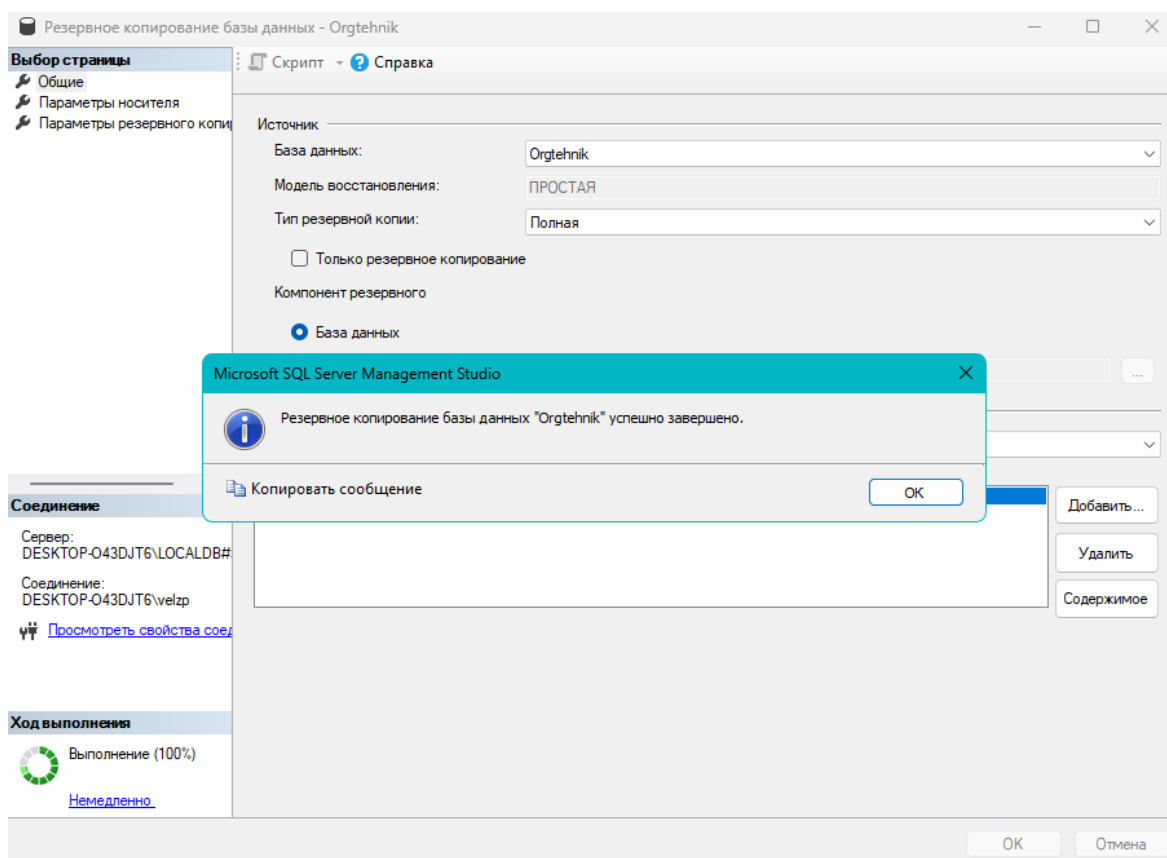


Рисунок 24 – Результат резервного копирования базы данных

### 1.3 Разработка библиотеки и подключение её к проекту

Для обеспечения тестирования базы данных была разработана отдельная библиотека «ControllorDB». Она содержит необходимые классы и методы для взаимодействия с базой данных, что позволяет проверять её функциональность и проводить модульные тесты (ПРИЛОЖЕНИЕ Б).

### 1.4 Разработка приложения

Приложение разработано для учета заявок на обслуживание и ремонт оргтехники (ПРИЛОЖЕНИЕ Б).

При успешном запуске программы появляется экранная форма авторизации – окно с предложением ввести имя пользователя и его пароль. Внешний вид экранной формы авторизации представлен на рисунке 25.

Авторизация

Авторизация в систему  
Пожалуйста, авторизуйтесь

Логин

Пароль  [Показать пароль](#)

Вход

Рисунок 25 – Окно авторизации

На форме имеется два поля для ввода логина и пароля. Также представлена кнопка для авторизации. Если её нажать и данные указаны корректно – откроется главное окно пользователя (Рисунок 26)

Форма клиентов

Добро пожаловать!

История ваших заявок [Вернуться назад](#)

Код заявки	Дата создания	Техника	Описание проблемы	Статус заявки	Дата окончания работ	Комментарий
4	02.08.2023 0:00:00	DEXP Atlas H388	Выключается	Завершено	19.09.2024 12:32:30	
4	02.08.2023 0:00:00	DEXP Atlas H388	Выключается	Завершено	19.09.2024 12:32:30	Будем смотреть
2008	19.09.2024 12:36:16	MSI GF76 Katana 11UC-879XRU черный	Не работает экран	Завершено	19.09.2024 12:38:19	Будем смотреть

Оставить заявку Удалить заявку Редактировать заявку

Найти запись

Количество записей: 3

Рисунок 26 – Главная страница клиента

У клиента имеется несколько кнопок и функций, которые он может использовать в системе.

- 1) Кнопка «Составить заявку» - открывается форма с новой заявкой, где клиент может выбрать технику для ремонта и оставить описание проблемы.
- 2) Кнопка «Удалить заявку» - удаляется новая выбранная заявка клиентом
- 3) Кнопка «Редактировать заявку» - открывается форма редактирования заявки, где клиент может редактировать уже созданную им заявку.
- 4) Кнопка и поле «Поиск» - позволяет искать созданные заявки по ключевым словам.
- 5) Кнопка «Вернуться назад» - позволяет пользователю вернуться на страницу авторизации.

При входе в систему с данными оператора отображается окно работника. Внешний вид окна представлен на рисунке 27.

Форма оператора

Добро пожаловать!

Заявки клиентов

Вернуться назад

	Код заявки	Дата создания	Техника	Статус заявки	Описание проблемы	ФИО клиента	Телефон клиента
▶	3009	19.09.2024 19:57:55	DEXP Aquilon O286	Новая заявка	ТестОписание	Григорьев Семён Викторович	89219567849
	3010	19.09.2024 21:44:55	DEXP Aquilon O286	Новая заявка	ТестОписание	Григорьев Семён Викторович	89219567849
	3012	19.09.2024 22:14:53	DEXP Aquilon O286	Новая заявка	ТестОписание	Григорьев Семён Викторович	89219567849
	4010	20.09.2024 0:37:06	DEXP Aquilon O286	Новая заявка	ТестОписание	Григорьев Семён Викторович	89219567849

Отклонить заявку

Обработать заявку

Найти записи

Количество записей: 4

Рисунок 27 – Главная страница оператора

У оператора есть несколько кнопок и функций, которые он может использовать в системе.

- 1) Кнопка «Отклонить заявку» - позволяет оператору отклонять оставленные клиентами заявки по какой-либо причине.
- 2) Кнопка «Обработать» - открывается форма с выбранной заявкой, где оператор может назначить мастера и оставить комментарий клиенту.
- 3) Кнопка и поле «Поиск» - позволяет искать созданные заявки по ключевым словам.
- 4) Кнопка «Вернуться назад» - позволяет пользователю вернуться на страницу авторизации.

При входе в систему с данными мастера отображается окно мастера. Внешний вид окна представлен на рисунке 28.

Код заявки	Дата создания	Техника	Статус заявки	Описание проблемы	ФИО клиента	Телефон клиента
2	05.05.2023 0:00:00	DEXP Aquilon O286	В работе	Перестал работать	Носов Иван Михайлович	89210563128

Сформировать отчет о выполняемых работ

Найти запись

Количество записей: 1

Рисунок 28 – Главная страница мастера

У мастера есть несколько кнопок и функций, которые он может использовать в системе.

1) Кнопка «Сформировать отчёт о выполняемых работ» - открывается форма с выбранной заявкой, мастер может указать запчасти, которые были использованы в ходе ремонта.

2) Кнопка и поле «Поиск» - позволяет искать созданные заявки по ключевым словам.

3) Кнопка «Вернуться назад» - позволяет пользователю вернуться на страницу авторизации.

## Выполнение отладки программного модуля (Рисунок 29).

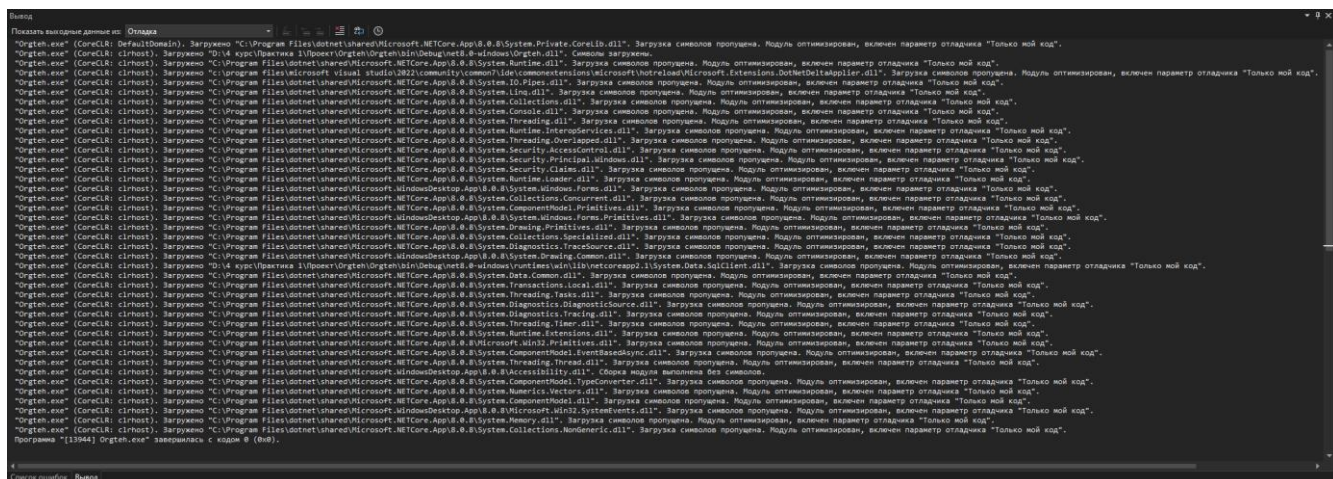


Рисунок 29 – Отладка программного модуля

Выполнение отладки программного обеспечения с использованием инструментальных средств (Рисунки 30-32).

```

81
82
83     Savelog(login, true);
84     string[] arrFio = dt.Rows[0]["fio"].ToString().Split();
85     name = arrFio[1];
86     surname = arrFio[0];
87     lvl = dt.Rows[0]["lvl"].ToString();
88     userID = Convert.ToInt32(dt.Rows[0]["userID"]);
89     MessageBox.Show($"Добро пожаловать {name} {surname}!\nВы зашли как {lvl}");
90     Form MainForm = new Form();
91     switch (lvl)
92     {
93         case "Клиент": MainForm = new ClientForm(); break;
94         case "Мастер": MainForm = new MasterForm(); break;
95         case "Оператор": MainForm = new OperatorForm(); break;
96     }
97     this.Hide();
98     MainForm.ShowDialog();
99 }
100
101 else
102 {
103     failenters = failenters + 1;

```

Рисунок 30 – Получение данных о пользователе

The screenshot shows the Visual Studio IDE with the same C# code as in Figure 30. The Local Variables window is open, displaying the following information:

Имя	Значение	Тип
System.Data.DataTable.Rows.get возвращено	(System.Data.DataRowCollection)	System.Data.DataRowCollection
System.Data.DataRowCollection.this[int].get возвращено	(System.Data.DataRow)	System.Data.DataRow
System.Data.DataRow.this[string].get возвращено	"Клиент"	string
object.ToString возвращено	"Клиент"	string
this	(Orgteh.Authorization, Text: Учёт заявок)	Orgteh.Authorization
sender	(Text = "Вход")	object (System.Windows.Forms.But...
e	(X = 125 Y = 14 Button = Left)	System.EventArgs (System.Window...
login	"1"	string
password	"1"	string
adapter	(System.Data.SqlClient.SqlDataAdapter)	System.Data.SqlClient.SqlDataAdapt...
dt	()	System.Data.DataTable
query	"SELECT userID, fio, lvl FROM Users WHERE login = @login AND password = @password"	string
sqlCommand	(System.Data.SqlClient.SqlCommand)	System.Data.SqlClient.SqlCommand
arrFio	{string[3]}	string[]
MainForm	null	System.Windows.Forms.Form

Рисунок 31 – Локальные переменные



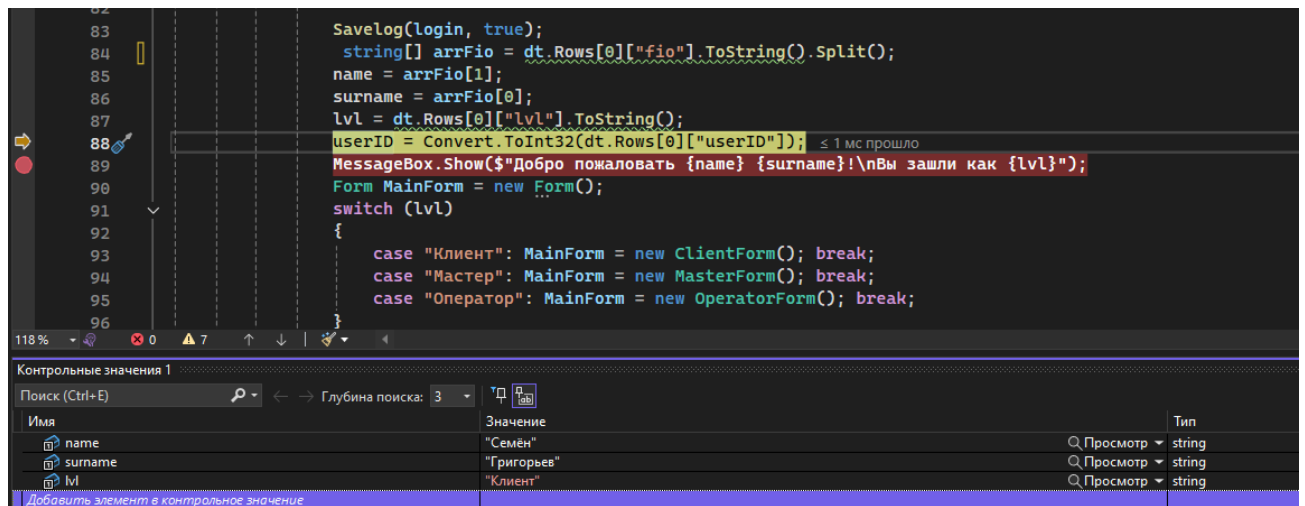


Рисунок 32 – Контрольные значения полученные из БД

## 1.5 Тестирование приложения

### 1.5.1 Создание тестовых случаев

Для тестирования приложения было подготовлено 5 тестовых случаев, с помощью которых можно тестировать функционал приложения. Были протестированы такие функции как: успешная авторизация пользователя, неуспешная авторизация пользователя, создание новой заявки клиентом, проверка удаления заявки, изменение статуса заявки оператором (ПРИЛОЖЕНИЕ А).

### 1.5.2 Модульное тестирование

Были разработаны модульные тесты, которые позволяют проверять функциональность основных методов программного обеспечения. Тестируется корректность выполнения запросов на получение данных из базы данных, а также обработка исключений при выполнении неправильных запросов. Оценивается правильность выполнения запросов на получение конкретных записей, анализируется обработка исключений при неверных запросах на получение записей, и тестируется обработка исключений при попытках изменить данные в базе с неправильными запросами.

Для модульного тестирования были разработаны юнит-тесты (ПРИЛОЖЕНИЕ Б).

Результаты тестирования представлены на рисунке 33.

Тестирование	Длительн...	Признаки	Сообщение об ош...
▲ ✓ TestProject2 (5)	2,8 с		
▲ ✓ TestProject2 (5)	2,8 с		
▲ ✓ UnitTest1 (5)	2,8 с		
✓ TestChangeRequestStatus	18 мс		
✓ TestDeleteRequest	3 мс		
✓ TestGetDataTable	9 мс		
✓ TestGetSqlDataReader	1,1 с		
✓ TestInvalidLogin	1,6 с		

Рисунок 33 – Результаты тестирования

## 1.6 Выгрузка готового проекта в репозиторий Git

Выгрузка проекта в репозиторий Git представлена на рисунке 34.

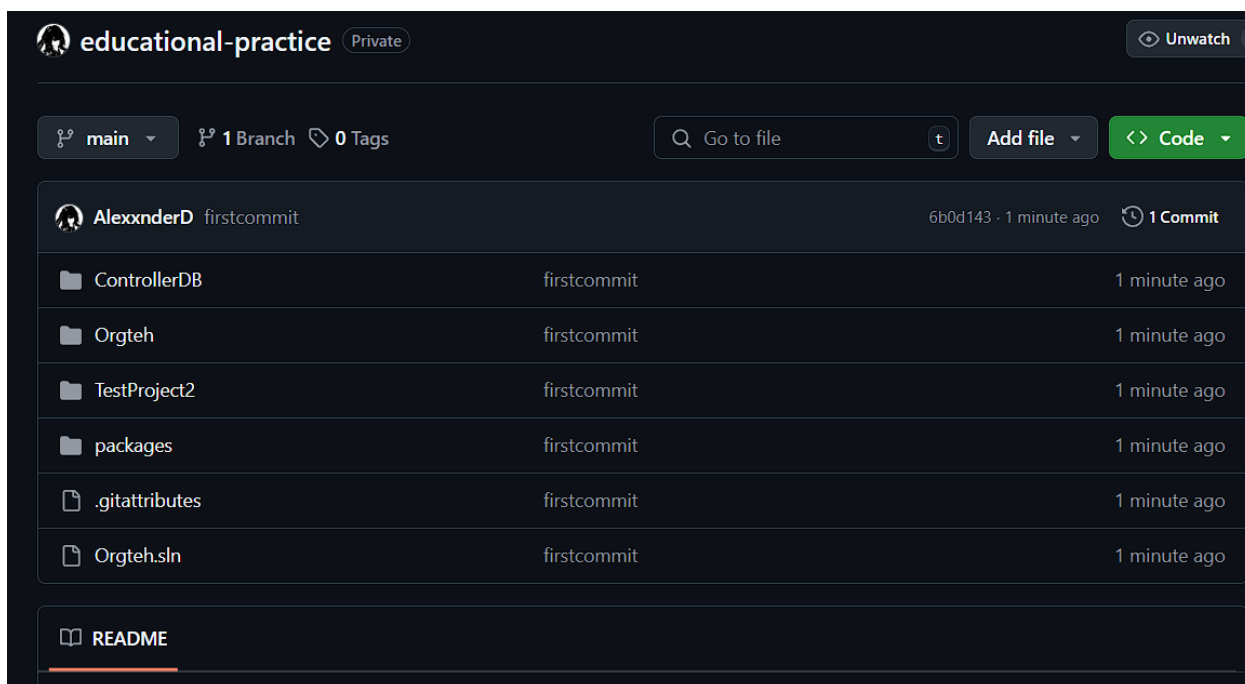


Рисунок 34 – Проект в GitHub

## **ЗАКЛЮЧЕНИЕ**

В рамках работы была осуществлена разработка технического задания и прочей документации. Кроме этого, было уделено время проектированию UML-диаграмм. Особое внимание было уделено проектированию пользовательского интерфейса и разработке дизайна программы. Была разработана база данных. Также было реализовано приложение на языке программирования C#, включая формы авторизации, основные формы и библиотеку классов. Помимо этого, были созданы тест-кейсы для проверки функциональности приложения

Практика позволила закрепить теоретические знания и получить ценный опыт работы с современными инструментами разработки программного обеспечения. Особое внимание было уделено интеграции модулей и тестированию приложений, что способствовало улучшению навыков проектирования и разработки программного обеспечения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Горбач, И. Microsoft SQL Server 2005 Analysis Services. OLAP и многомерный анализ данных / И. Горбач, А. Бергер. - М.: БХВ-Петербург, Год издания: 2019.
- 2) Жилинский, А. Самоучитель Microsoft SQL Server 2008 / А. Жилинский. - М.: БХВ-Петербург, Год издания: 2018.
- 3) Кригель, А. SQL. Библия пользователя / А. Кригель. - М.: Диалектика / Вильямс, Год издания: 2019.
- 4) Ицик, Бен-Ган Microsoft SQL Server 2012. Высокопроизводительный код T-SQL. Оконные функции / Бен-Ган Ицик. - М.: Русская Редакция, Год издания: 2019.
- 5) Сомов, Н. И. UML и проектирование программных систем: учебное пособие. — СПб.: БХВ-Петербург, 2018. — 416 с.
- 6) Перси, Р. Практическое руководство по SQL и базам данных: учебник. — СПб.: БХВ-Петербург, 2021. — 448 с.
- 7) Максвелл, Дж. Программирование на C#: практическое руководство. — М.: Эксмо, 2020. — 368 с.
- 8) Скотт, М. ASP.NET Core для разработчиков: учебное пособие. — СПб.: Питер, 2021. — 560 с.
- 9) Пост, Г. SQL: основы программирования баз данных: учебное пособие. — СПб.: Питер, 2019. — 512 с.
- 10) Долгих, А. Microsoft SQL Server 2005. Практические методы работы (+ CD-ROM) / А. Долгих. - М.: Эком, Год издания: 2015.

ПРИЛОЖЕНИЕ А

(справочное)

**Таблицы тестирования**

Таблица А.1 – Аннотация теста

<b>Название проекта</b>	Система учета заявок на ремонт офисной техники
<b>Рабочая версия</b>	1.0
<b>Имя тестирующего</b>	Морозов Александр Михайлович
<b>Дата тестирования</b>	19.09.2024

Таблица А.2 – Тестовый пример #1

<b>Тестовый пример #</b>	1.0
<b>Приоритет тестирования</b>	Высокий
<b>Заголовок/название теста</b>	Проверка авторизации с действительными данными
<b>Краткое изложение теста</b>	Тест проверяет успешный вход мастера в систему с корректным логином и паролем
<b>Этапы теста</b>	<ol style="list-style-type: none"> <li>1. Открыть форму авторизации</li> <li>2. Ввод логина и пароля в текстовые поля «Логин» и «Пароль»</li> <li>3. Нажать кнопку «Войти»</li> </ol>
<b>Тестовые данные</b>	Логин: login2 Пароль: pass2
<b>Ожидаемый результат</b>	Пользователь успешно авторизован
<b>Фактический результат</b>	Пользователь успешно авторизован
<b>Статус</b>	Зачет
<b>Предварительное условие</b>	В базе данных должен существовать мастер с логином login2 и паролем pass2
<b>Постусловие</b>	Система открывает главную форму мастера
<b>Примечания/комментарии</b>	

Таблица А.3 – Тестовый пример #2

<b>Тестовый пример #</b>	2.0
<b>Приоритет тестирования</b>	Высокий
<b>Заголовок/название теста</b>	Проверка авторизации с неверными данными
<b>Краткое изложение теста</b>	Тест проверяет отклонение попытки входа при вводе неверных данных
<b>Этапы теста</b>	<ol style="list-style-type: none"> <li>1. Открыть форму авторизации</li> <li>2. Ввод логина и пароля в текстовые поля «Логин» и «Пароль»</li> <li>3. Нажать кнопку «Войти»</li> </ol>
<b>Тестовые данные</b>	Логин: login12 Пароль: 123
<b>Ожидаемый результат</b>	Вывод сообщения об ошибке авторизации: "Неправильно введен логин или пароль!"
<b>Фактический результат</b>	Вывод сообщения об ошибке авторизации: "Неправильно введен логин или пароль!"
<b>Статус</b>	Зачет
<b>Предварительное условие</b>	Пользователь в базе данных с такими данными не должен существовать
<b>Постусловие</b>	Авторизация не пройдена, доступ отклонен
<b>Примечания/комментарии</b>	

Таблица А.4 – Тестовый пример #3

<b>Тестовый пример #</b>	3.0
<b>Приоритет тестирования</b>	Высокий
<b>Заголовок/название теста</b>	Проверка создания новой заявки на ремонт с корректными данными
<b>Краткое изложение теста</b>	Тест проверяет успешное создание новой заявки клиентом
<b>Этапы теста</b>	<ol style="list-style-type: none"> <li>1. Открыть форму клиента</li> <li>2. Нажать кнопку «Оставить заявку»</li> <li>3. Выбрать технику в поле «Модель техники»</li> <li>4. Описать проблему в поле «Описание проблемы»</li> </ol>
<b>Тестовые данные</b>	Модель техники: «DEXP Aquilon O286» Описание проблемы: «ТестОписание»
<b>Ожидаемый результат</b>	Заявка успешно создана

Продолжение таблицы А.4

<b>Фактический результат</b>	Заявка успешно создана
<b>Статус</b>	Зачет
<b>Предварительное условие</b>	Необходимо быть авторизованным в систему с правами клиента
<b>Постусловие</b>	Заявка успешно добавлена в таблицу и готова к дальнейшей обработке
<b>Примечания/комментарии</b>	

Таблица А.5 – Тестовый пример #4

<b>Тестовый пример #</b>	4.0
<b>Приоритет тестирования</b>	Высокий
<b>Заголовок/название теста</b>	Проверка удаления заявки со статусом "Новая заявка"
<b>Краткое изложение теста</b>	Тест проверяет, удаляется ли заявка со статусом "Новая заявка" при выполнении действия "Удалить заявку".
<b>Этапы теста</b>	<ol style="list-style-type: none"> <li>1. Открыть форму клиента</li> <li>2. Выбрать заявку со статусом "Новая заявка" в таблице заявок</li> <li>3. Нажать на кнопку «Удалить заявку»</li> </ol>
<b>Тестовые данные</b>	Статус заявки: "Новая заявка" ID заявки: 4009 (или любая доступная заявка со статусом "Новая заявка")
<b>Ожидаемый результат</b>	Заявка со статусом "Новая заявка" удаляется из таблицы заявок.
<b>Фактический результат</b>	Заявка со статусом "Новая заявка" удаляется из таблицы заявок.
<b>Статус</b>	Зачет
<b>Предварительное условие</b>	Необходимо быть авторизованным в систему с правами клиента. Должна быть хотя бы одна заявка со статусом "Новая заявка"
<b>Постусловие</b>	Заявка с указанным id удалена из таблицы заявок.
<b>Примечания/комментарии</b>	



Таблица А.6 – Тестовый пример #5

<b>Тестовый пример #</b>	5.0
<b>Приоритет тестирования</b>	Высокий
<b>Заголовок/название теста</b>	Проверка изменения статуса заявки
<b>Краткое изложение теста</b>	Тест проверяет обновление статуса заявки и его запись в базе данных
<b>Этапы теста</b>	<ol style="list-style-type: none"> <li>1. Открыть список заявок</li> <li>2. Выбрать заявку со статусом "Новая заявка"</li> <li>3. Установить статус заявки на "В процессе"</li> </ol>
<b>Тестовые данные</b>	Статус заявки: "Новая заявка"
<b>Ожидаемый результат</b>	Статус заявки изменен, данные обновлены в базе данных
<b>Фактический результат</b>	Статус заявки изменен, данные обновлены в базе данных
<b>Статус</b>	Зачет
<b>Предварительное условие</b>	Необходимо быть авторизованным в систему с правами оператора, должна существовать заявка со статусом «Новая заявка»
<b>Постусловие</b>	В таблице Requests статус заявки изменён на «В процессе»
<b>Примечания/комментарии</b>	

## ПРИЛОЖЕНИЕ Б

(справочное)

### Исходный код

#### Скрипт базы данных:

```
CREATE TABLE Users (  
    userID INT IDENTITY(1,1) PRIMARY KEY,  
    fio VARCHAR(50),  
    phone varchar(11),  
    login varchar(25),  
    password varchar(25),  
    lvl varchar(25),  
);  
CREATE TABLE TypesOrg (  
    typeID INT IDENTITY(1,1) PRIMARY KEY,  
    typeName varchar(50),  
);  
CREATE TABLE ModelsOrg(  
    modelID INT IDENTITY(1,1) PRIMARY KEY,  
    modelName varchar(150),  
    typeID INT,  
    CONSTRAINT FK_TechTypes_TypesOrg FOREIGN KEY (typeID) REFERENCES TypesOrg (typeID),  
);  
CREATE TABLE Requests (  
    requestID INT IDENTITY(1,1) PRIMARY KEY,  
    startDate DATETIME,  
    orgTechModel INT,  
    problemDescription VARCHAR(100),  
    requestStatus varchar(30),  
    completionDate DATETIME,  
    repairParts VARCHAR(150),  
    masterID INT,  
    clientID INT,  
    CONSTRAINT FK_Requests_Client FOREIGN KEY (clientID) REFERENCES Users(userID),  
    CONSTRAINT FK_Requests_Master FOREIGN KEY (masterID) REFERENCES Users(userID),  
    CONSTRAINT FK_Requests_Tech FOREIGN KEY (orgTechModel) REFERENCES ModelsOrg  
(ModelID),  
);  
CREATE TABLE Comments (  
    commentID INT IDENTITY(1,1) PRIMARY KEY,  
    message varchar(150),  
    masterID INT,  
    requestID INT,  
    CONSTRAINT FK_comments_Requests FOREIGN KEY (requestID) REFERENCES Requests(requestID),  
    CONSTRAINT FK_Comments_Users FOREIGN KEY (masterID) REFERENCES Users(userID),  
);
```

#### Код библиотеки ControllerDB:

```
using System;  
using System.Collections.Generic;  
using System.Data.SqlClient;  
using System.Data;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```

namespace ControllerDB
{
    public class Database
    {
        private SqlConnection sqlConnection = new SqlConnection(@"Data Source = (LocalDb)\DBLocal;
Initial Catalog=Orgtehnika; Integrated Security=True");

        private void OpenConnection()
        {
            if (sqlConnection.State == System.Data.ConnectionState.Closed)
            {
                sqlConnection.Open();
            }
        }

        private void CloseConnection()
        {
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }

        private SqlConnection GetConnection()
        {
            return sqlConnection;
        }

        public DataTable LoadTable(string query)
        {
            DataTable dt = new DataTable();
            try
            {
                OpenConnection();
                SqlDataAdapter adapter = new SqlDataAdapter();
                SqlCommand sqlCommand = new SqlCommand(query, GetConnection());
                adapter.SelectCommand = sqlCommand;
                adapter.Fill(dt);
                return dt;
            }
            catch (Exception)
            {
                throw new Exception("Данные не были получены!");
            }
        }

        public void ExecuteQuery(string query)
        {
            try
            {
                OpenConnection();
                using (SqlCommand command = new SqlCommand(query, GetConnection()))
                {
                    command.ExecuteNonQuery();
                }
            }
            catch (Exception)
            {
                throw new Exception("Ошибка запроса!");
            }
            finally
            {
                CloseConnection();
            }
        }
    }
}

```

```

    }

    public SqlDataReader GetSqlDataReader(string query)
    {
        SqlDataReader reader = null;
        try
        {
            OpenConnection();
            SqlCommand command = new SqlCommand(query, GetConnection());
            reader = command.ExecuteReader();
            return reader;
        }
        catch (Exception)
        {
            throw new Exception("Данные не были получены!");
        }
    }
}
}

```

### Код приложения:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Net.NetworkInformation;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Schema;

namespace Orgteh
{
    public partial class Authorization : Form
    {
        DB dataBase = new DB();

        public static int userID;
        public static string lvl = null;

        private bool showpass = true;
        private bool reloadapp = false;
        private string captchat = null;
        private int failenters = 0;
        private int counttimer = 3;
        public static string name = null;
        public static string surname = null;
        public Authorization()
        {
            InitializeComponent();
            textBoxpass.PasswordChar = '*';
            textBoxLogin.MaxLength = 25;
            textBoxpass.MaxLength = 25;
            this.StartPosition = FormStartPosition.CenterScreen;
            timerEnter.Interval = 1000;
        }
    }
}

```

```

private void buttonLogin_Click(object sender, EventArgs e)
{
    if (reloadapp)
    {
        Application.Restart();
    }
    string login = textBoxLogin.Text;
    string password = textBoxpass.Text;
    if (login == "" || password == "")
    {
        MessageBox.Show("Укажите логин и пароль для входа.");
        return;
    }
    SqlDataAdapter adapter = new SqlDataAdapter();
    DataTable dt = new DataTable();
    string query = "SELECT userID, fio, lvl FROM Users WHERE login = @login AND password = @password";

    SqlCommand sqlCommand = new SqlCommand(query, dataBase.GetConnection());
    sqlCommand.Parameters.AddWithValue("@login", login);
    sqlCommand.Parameters.AddWithValue("@password", password);

    adapter.SelectCommand = sqlCommand;
    try
    {
        adapter.Fill(dt);
    }
    catch (Exception)
    {
        MessageBox.Show("Не удалось подключиться к базе данных!");
        return;
    }

    if (panelcaptch.Visible == true)
    {
        if (textBoxCaptcha.Text != captchat)
        {
            MessageBox.Show("Неверный код капчи! Попробуйте ещё раз.");
            return;
        }
    }
    if (dt.Rows.Count == 1)
    {
        Savelog(login, true);
        string[] arrFio = dt.Rows[0]["fio"].ToString().Split();
        name = arrFio[1];
        surname = arrFio[0];
        lvl = dt.Rows[0]["lvl"].ToString();
        userID = Convert.ToInt32(dt.Rows[0]["userID"]);
        MessageBox.Show($"Добро пожаловать {name} {surname}!\nВы зашли как {lvl}");
        Form MainForm = new Form();
        switch (lvl)
        {
            case "Клиент": MainForm = new ClientForm(); break;
            case "Мастер": MainForm = new MasterForm(); break;
            case "Оператор": MainForm = new OperatorForm(); break;
        }
        this.Hide();
        MainForm.ShowDialog();
    }
}

```

```

else
{
    failenters = failenters + 1;
    Savelog(login, false);
    FailEnter();
}
}

private void Savelog(string login, bool enterstatus)
{
    dataBase.OpenConnection();
    string query = "INSERT INTO Logs (entertime, userlogin, enterstatus) " +
        "VALUES (@entertime, @userlogin, @enterstatus)";
    SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
    command.Parameters.AddWithValue("@entertime", DateTime.Now);
    command.Parameters.AddWithValue("@userlogin", login);
    command.Parameters.AddWithValue("@enterstatus", enterstatus ? 1 : 0);

    command.ExecuteNonQuery();
    dataBase.CloseConnection();
}

private void label2_Click(object sender, EventArgs e)
{
    if (showpass)
    {
        textBoxpass.PasswordChar = '*';
    }

    else
    {
        textBoxpass.PasswordChar = '\0';
    }
    showpass = !showpass;
}

private void FailEnter()
{
    if (failenters == 2)
    {
        timerEnter.Start();
        buttonEnter.Enabled = false;
    }
    else if (failenters == 3)
    {
        reloadapp = true;
        buttonEnter.Text = "Требуется перезапуск";
    }
}

MessageBox.Show("Неправильно введён логин или пароль!");
textBoxLogin.Text = null;
textBoxpass.Text = null;
panelcaptch.Visible = true;
pictureBox1.Image = CreateImage(pictureBox1.Width, pictureBox1.Height);
}

private Bitmap CreateImage(int Width, int Height)
{
    Random rnd = new Random();

    Bitmap result = new Bitmap(Width, Height);

    int Xpos = rnd.Next(0, Width - 70);

```

```

        int Ypos = rnd.Next(0, Height - 15);

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

        Graphics g = Graphics.FromImage(result);

        g.Clear(Color.Gray);

        captchat = null;
        string ALF = "1234567890QWERTYUIOPASDFGHJKLZXCVBNM";
        for (int i = 0; i < 5; ++i)
            captchat += ALF[rnd.Next(ALF.Length)];

        g.DrawString(captchat, 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));

        for (int i = 0; i < Width; ++i)
            for (int j = 0; j < Height; ++j)
                if (rnd.Next() % 20 == 0)
                    result.SetPixel(i, j, Color.White);

        return result;
    }

    private void buttonNewCaptcha_Click(object sender, EventArgs e)
    {
        pictureBox1.Image = CreateImage(pictureBox1.Width, pictureBox1.Height);
    }

    private void timerEnter_Tick(object sender, EventArgs e)
    {
        if (counttimer >= 0)
        {
            buttonEnter.Text = counttimer.ToString();
            counttimer--;
        }

        if (counttimer < 0)
        {
            timerEnter.Stop();
            buttonEnter.Enabled = true;
            buttonEnter.Text = "Вход";
        }
    }

    private void Authorization_FormClosed(object sender, FormClosedEventArgs e)
    {
        Application.Exit();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;

```

```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Orgteh
{
    public partial class ClientForm : Form
    {
        DB dataBase = new DB();
        private string userID = Authorization.userID.ToString();
        public ClientForm()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
        }

        private void LoadTable(string filter = "")
        {
            dataBase.OpenConnection();
            string query = @"SELECT Requests.requestID, startDate, ModelsOrg.modelName,
problemDescription, requestStatus, completionDate, message
FROM Requests
INNER JOIN ModelsOrg ON Requests.orgTechModel = ModelsOrg.modelID
INNER JOIN TypesOrg ON ModelsOrg.typeID = TypesOrg.typeID
LEFT JOIN Comments ON Comments.requestID = Requests.requestID
WHERE Requests.clientID = @clientID";

            if (!string.IsNullOrEmpty(filter))
            {
                query += " AND (ModelsOrg.modelName LIKE @filter OR problemDescription LIKE @filter
OR message LIKE @filter)";
            }

            SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
            command.Parameters.AddWithValue("@clientID", userID);
            if (!string.IsNullOrEmpty(filter))
            {
                command.Parameters.AddWithValue("@filter", "%" + filter + "%");
            }

            SqlDataAdapter adapter = new SqlDataAdapter(command);
            DataTable dt = new DataTable();
            adapter.Fill(dt);
            dataGridViewRequest.Columns.Clear();
            dataGridViewRequest.Rows.Clear();
            foreach (DataColumn column in dt.Columns)
            {
                DataGridViewColumn dataGridViewColumn = new DataGridViewTextBoxColumn
                {
                    Name = column.ColumnName,
                    HeaderText = HeadertableText(column.ColumnName)
                };
                dataGridViewRequest.Columns.Add(dataGridViewColumn);
            }
            foreach (DataRow row in dt.Rows)
            {
                dataGridViewRequest.Rows.Add(row.ItemArray);
            }
            dataBase.CloseConnection();
        }
    }
}

```



```

        labelRecordCount.Text = $"Количество записей: {dt.Rows.Count}";
    }

private string Headertext(string columnName)
{
    switch (columnName)
    {
        case "requestID":
            return "Код заявки";
        case "startDate":
            return "Дата создания";
        case "modelName":
            return "Техника";
        case "problemDescription":
            return "Описание проблемы";
        case "requestStatus":
            return "Статус заявки";
        case "completionDate":
            return "Дата окончания работ";
        case "message":
            return "Комментарий";
        default:
            return columnName;
    }
}

private void buttoncreateRequest_Click(object sender, EventArgs e)
{
    CreateRequest newRequest = new CreateRequest();
    if (newRequest.ShowDialog() == DialogResult.Cancel)
    {
        LoadTable();
    }
}

private void buttonEditRequest_Click(object sender, EventArgs e)
{
    if (dataGridViewRequest.SelectedCells.Count > 0)
    {
        int selectedRowIndex = dataGridViewRequest.SelectedCells[0].RowIndex;
        DataGridViewRow selectedRow = dataGridViewRequest.Rows[selectedRowIndex];
        if (selectedRow.Cells[4].Value.ToString() != "Новая заявка")
        {
            MessageBox.Show("Вы можете изменять только заявки, которые имеют статус Новая заявка.");
            return;
        }
        string requestID = selectedRow.Cells[0].Value.ToString();
        EditRequest editRequest = new EditRequest(requestID);
        if (editRequest.ShowDialog() == DialogResult.Cancel)
        {
            LoadTable();
        }
    }
    else
    {
        MessageBox.Show("Выберите заявку из списка в таблице.");
    }
}

```

```

private void buttonDelRequest_Click(object sender, EventArgs e)
{
    if (dataGridViewRequest.SelectedCells.Count > 0)
    {
        int selectedRowIndex = dataGridViewRequest.SelectedCells[0].RowIndex;
        DataGridViewRow selectedRow = dataGridViewRequest.Rows[selectedRowIndex];
        if (selectedRow.Cells[4].Value.ToString() != "Новая заявка")
        {
            MessageBox.Show("Вы можете удалять только заявки, которые имеют статус Новая заявка!");
            return;
        }
        string requestID = selectedRow.Cells[0].Value.ToString();
        dataBase.OpenConnection();
        string query = $"DELETE FROM Requests WHERE requestID = {requestID}";
        SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
        command.ExecuteNonQuery();
        dataBase.CloseConnection();
        MessageBox.Show("Заявка была успешно удалена.");
        LoadTable();
    }
    else
    {
        MessageBox.Show("Выберите заявку из списка в таблице.");
    }
}

private void ExitBtn_Click(object sender, EventArgs e)
{
    Authorization logIn = new Authorization();
    foreach (Form form in Application.OpenForms)
    {
        form.Hide();
    }
    logIn.ShowDialog();
}

private void ClientForm_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void ClientForm_Load(object sender, EventArgs e)
{
    LoadTable();
}

private void buttonSearch_Click(object sender, EventArgs e)
{
    string searchText = textBoxSearch.Text.Trim();
    LoadTable(searchText);
}
}

```

```

using System;

```

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;

```

```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Orgteh
{
    public partial class MasterForm : Form
    {
        DB dataBase = new DB();

        public MasterForm()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
            LoadTable("B pa6ore", dataGridViewRequest);
        }

        private void MasterForm_FormClosed(object sender, FormClosedEventArgs e)
        {
            Application.Exit();
        }

        private void LoadTable(string searchText, DataGridView dataGridView)
        {
            dataBase.OpenConnection();
            string query = @"SELECT Requests.requestID, startDate, modelName, requestStatus,
problemDescription, fio, phone
FROM Requests
INNER JOIN ModelsOrg ON Requests.orgTechModel = ModelsOrg.modelID
INNER JOIN Users ON Users.userID = Requests.clientID
WHERE requestStatus = @statusDescription AND masterID = @masterID";

            if (!string.IsNullOrEmpty(searchText))
            {
                query += " AND (modelName LIKE @searchText OR problemDescription LIKE @searchText
OR fio LIKE @searchText)";
            }

            using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
            {
                command.Parameters.AddWithValue("@statusDescription", "B pa6ore");
                command.Parameters.AddWithValue("@masterID", Authorization.userID.ToString());
                if (!string.IsNullOrEmpty(searchText))
                {
                    command.Parameters.AddWithValue("@searchText", "%" + searchText + "%");
                }

                SqlDataAdapter adapter = new SqlDataAdapter(command);
                DataTable dt = new DataTable();
                adapter.Fill(dt);

                dataGridView.Columns.Clear();
                dataGridView.Rows.Clear();

                foreach (DataColumn column in dt.Columns)
                {
                    DataGridViewColumn dataGridViewColumn = new DataGridViewTextBoxColumn
                    {
                        Name = column.ColumnName,
                        HeaderText = HeadertableText(column.ColumnName)
                    }
                }
            }
        }
    }
}

```

```

        };
        dataGridView.Columns.Add(dataGridViewColumn);
    }

    foreach (DataRow row in dt.Rows)
    {
        dataGridView.Rows.Add(row.ItemArray);
    }
    labelRecordCount.Text = $"Количество записей: {dt.Rows.Count}";
}
dataBase.CloseConnection();
}

private string HeadertableText(string columnName)
{
    switch (columnName)
    {
        case "requestID":
            return "Код заявки";
        case "startDate":
            return "Дата создания";
        case "modelName":
            return "Техника";
        case "problemDescription":
            return "Описание проблемы";
        case "fio":
            return "ФИО клиента";
        case "phone":
            return "Телефон клиента";
        default:
            return columnName;
    }
}

private void buttonReport_Click(object sender, EventArgs e)
{
    if (dataGridViewRequest.SelectedCells.Count > 0)
    {
        int selectedRowIndex = dataGridViewRequest.SelectedCells[0].RowIndex;
        DataGridViewRow selectedRow = dataGridViewRequest.Rows[selectedRowIndex];
        string requestID = selectedRow.Cells[0].Value.ToString();
        ReportRequest reportRequest = new ReportRequest(requestID);
        if (reportRequest.ShowDialog() == DialogResult.Cancel)
        {
            LoadTable("В работе", dataGridViewRequest);
        }
        LoadTable("", dataGridViewRequest);
    }
    else
    {
        MessageBox.Show("Пожалуйста, выберите заявку из таблицы для формирования отчета.");
    }
}

private void ExitBtn_Click(object sender, EventArgs e)
{
    Authorization logIn = new Authorization();

    foreach (Form form in Application.OpenForms)
    {
        form.Hide();
    }
}

```

```

    }
    logIn.ShowDialog();
}

private void buttonSearch_Click(object sender, EventArgs e)
{
    string searchText = textBoxSearch.Text.Trim();
    LoadTable(searchText, dataGridViewRequest);
}

private void MasterForm_Load(object sender, EventArgs e)
{
    LoadTable("", dataGridViewRequest);
}
}
}

```

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

```

```

namespace Orgteh
{

```

```

    public partial class OperatorForm : Form
    {
        DB dataBase = new DB();

```

```

        public OperatorForm()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
            LoadTable("Новая заявка", string.Empty);
        }

```

```

        private void LoadTable(string status, string searchText)
        {
            dataBase.OpenConnection();

```

```

            string query = @"SELECT Requests.requestID, startDate, modelName, requestStatus,
problemDescription, fio, phone
FROM Requests
INNER JOIN ModelsOrg ON Requests.orgTechModel = ModelsOrg.modelID
INNER JOIN Users ON Users.userID = Requests.clientID
WHERE requestStatus = @status";

```

```

            if (!string.IsNullOrEmpty(searchText))
            {
                query += " AND (modelName LIKE @searchText OR problemDescription LIKE @searchText
OR fio LIKE @searchText)";
            }

```

```

            using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
            {
                command.Parameters.AddWithValue("@status", status);
                if (!string.IsNullOrEmpty(searchText))
                {
                    command.Parameters.AddWithValue("@searchText", "%" + searchText + "%");
                }

```

```

                SqlDataAdapter adapter = new SqlDataAdapter(command);

```

```

        DataTable dt = new DataTable();
        adapter.Fill(dt);

        dataGridViewRequest.Columns.Clear();
        dataGridViewRequest.Rows.Clear();

        foreach (DataColumn column in dt.Columns)
        {
            DataGridViewColumn dataGridViewColumn = new DataGridViewTextBoxColumn
            {
                Name = column.ColumnName,
                HeaderText = HeadertableText(column.ColumnName)
            };
            dataGridViewRequest.Columns.Add(dataGridViewColumn);
        }

        foreach (DataRow row in dt.Rows)
        {
            dataGridViewRequest.Rows.Add(row.ItemArray);
        }
        labelRecordCount.Text = $"Количество записей: {dt.Rows.Count}";
    }
    dataBase.CloseConnection();
}

private string HeadertableText(string columnName)
{
    switch (columnName)
    {
        case "requestID":
            return "Код заявки";
        case "startDate":
            return "Дата создания";
        case "modelName":
            return "Техника";
        case "problemDescription":
            return "Описание проблемы";
        case "fio":
            return "ФИО клиента";
        case "phone":
            return "Телефон клиента";
        default:
            return columnName;
    }
}

private void OperatorForm_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void buttondismiss_Click(object sender, EventArgs e)
{
    if (dataGridViewRequest.SelectedCells.Count > 0)
    {
        ModerateRequest("Отказано", dataGridViewRequest);
    }
    else
    {
        MessageBox.Show("Пожалуйста, выберите заявку для отказа или обработки заявки.");
    }
}

```

```

private void ModerateRequest(string requestStatus, DataGridView dataGridView)
{
    if (dataGridView.CurrentRow == null)
    {
        MessageBox.Show("Выберите заявку из списка в таблице.");
        return;
    }

    string requestID = dataGridView.CurrentRow.Cells[0].Value?.ToString();
    if (string.IsNullOrEmpty(requestID))
    {
        MessageBox.Show("Невозможно получить ID заявки.");
        return;
    }

    try
    {
        string query = "UPDATE Requests SET requestStatus = @status WHERE requestID = @requestID";
        using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@status", requestStatus);
            command.Parameters.AddWithValue("@requestID", requestID);
            dataBase.OpenConnection();
            command.ExecuteNonQuery();
        }
        LoadTable("Новая заявка", textBoxSearch.Text);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
    finally
    {
        dataBase.CloseConnection();
    }
}

private void buttonaccept_Click(object sender, EventArgs e)
{
    if (dataGridViewRequest.SelectedCells.Count > 0)
    {
        int selectedRowIndex = dataGridViewRequest.SelectedCells[0].RowIndex;
        DataGridViewRow selectedRow = dataGridViewRequest.Rows[selectedRowIndex];
        string requestID = selectedRow.Cells[0].Value.ToString();
        Moderate processRequest = new Moderate(requestID);
        if (processRequest.ShowDialog() == DialogResult.Cancel)
        {
            LoadTable("В процессе", textBoxSearch.Text);
        }
    }
    else
    {
        MessageBox.Show("Выберите заявку из списка в таблице.");
    }
}

private void ExitBtnn_Click(object sender, EventArgs e)
{
    Authorization logIn = new Authorization();
    foreach (Form form in Application.OpenForms)

```

```

        {
            form.Hide();
        }
        logIn.ShowDialog();
    }

    private void buttonSearch_Click(object sender, EventArgs e)
    {
        LoadTable("Новая заявка", textBoxSearch.Text);
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Orgteh
{
    public partial class CreateRequest : Form
    {
        DB dataBase = new DB();

        public CreateRequest()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
            try
            {
                dataBase.OpenConnection();

                string query = "SELECT modelName FROM ModelsOrg";
                SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    comboBoxModel.Items.Add(reader[0].ToString());
                }
                reader.Close();
                dataBase.CloseConnection();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message);
            }
        }

        private void buttonNewRequest_Click(object sender, EventArgs e)
        {
            if (comboBoxModel.Text == "" || textBoxproblemdesk.Text == "")
            {
                MessageBox.Show("Заполните все поля!");
                return;
            }
        }
    }
}

```



```

try
{
    dataBase.OpenConnection();

    string query = @"INSERT INTO Requests (startDate, orgTechModel, problemDescription,
requestStatus, clientID)
    VALUES (@startDate, @techModelID, @problemDescription, @status, @clientID)";

    using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
    {
        command.Parameters.AddWithValue("@startDate", DateTime.Now);
        command.Parameters.AddWithValue("@techModelID", OrgModelid());
        command.Parameters.AddWithValue("@problemDescription", textBoxproblemdesk.Text);
        command.Parameters.AddWithValue("@status", "Новая заявка");
        command.Parameters.AddWithValue("@clientID", Authorization.userID);

        int result = command.ExecuteNonQuery();
        dataBase.CloseConnection();

        if (result > 0)
        {
            MessageBox.Show("Запись успешно создана!");
        }
        else
        {
            MessageBox.Show("Ошибка при добавлении записи!");
        }
        this.Close();
    }
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка: " + ex.Message);
}
}

private string OrgModelid()
{
    try
    {
        string query = "SELECT modelID FROM ModelsOrg WHERE modelName = @modelName";

        using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@modelName", comboBoxModel.Text);

            using (SqlDataReader reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    return reader["modelID"].ToString();
                }
            }
        }
        return null;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
        return null;
    }
}

```

```

    }

    private void buttonCreate_Click(object sender, EventArgs e)
    {
        if (comboBoxModel.Text == "" || textBoxproblemdesk.Text == "")
        {
            MessageBox.Show("Заполните все поля!");
            return;
        }
        try
        {
            dataBase.OpenConnection();

            string query = @"INSERT INTO Requests (startDate, orgTechModel, problemDescription,
requestStatus, clientID)
                VALUES (@startDate, @techModelID, @problemDescription, @status, @clientID)";

            using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
            {
                command.Parameters.AddWithValue("@startDate", DateTime.Now);
                command.Parameters.AddWithValue("@techModelID", OrgModelid());
                command.Parameters.AddWithValue("@problemDescription", textBoxproblemdesk.Text);
                command.Parameters.AddWithValue("@status", "Новая заявка");
                command.Parameters.AddWithValue("@clientID", Authorization.userID);

                int result = command.ExecuteNonQuery();
                dataBase.CloseConnection();

                if (result > 0)
                {
                    MessageBox.Show("Запись успешно создана!");
                }
                else
                {
                    MessageBox.Show("Ошибка при добавлении записи!");
                }
                this.Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка: " + ex.Message);
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace Orgteh
{
    public partial class EditRequest : Form

```

```

{
    DB dataBase = new DB();
    private string requestID;
    public EditRequest(string requestID)
    {
        InitializeComponent();
        this.StartPosition = FormStartPosition.CenterScreen;
        this.requestID = requestID;
        LoadTable();
        try
        {
            dataBase.OpenConnection();
            string query = "SELECT modelName FROM ModelsOrg";

            SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                comboBoxModel.Items.Add(reader[0].ToString());
            }
            reader.Close();
            dataBase.CloseConnection();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка: " + ex.Message);
        }
    }
}
private void LoadTable()
{
    try
    {
        dataBase.OpenConnection();
        string query = $"SELECT modelName, problemDescription FROM Requests " +
            $"INNER JOIN ModelsOrg ON orgTechModel = modelID" +
            $" WHERE requestID = {requestID}";
        SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
        SqlDataReader reader = command.ExecuteReader();
        if (reader.Read())
        {
            comboBoxModel.Text = reader[0].ToString();
            textBoxproblemdesc.Text = reader[1].ToString();
        }
        reader.Close();
        dataBase.CloseConnection();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}
private string OrgModelid()
{
    try
    {
        string query = "SELECT modelID FROM ModelsOrg WHERE modelName = @modelName";

        using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@modelName", comboBoxModel.Text);

            using (SqlDataReader reader = command.ExecuteReader())

```

```

        {
            if (reader.Read())
            {
                return reader["modelID"].ToString();
            }
        }
    }
    return null;
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка: " + ex.Message);
    return null;
}
}

private void buttonSave_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(comboBoxModel.Text) ||
        string.IsNullOrEmpty(textBoxproblemdesc.Text))
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    try
    {
        dataBase.OpenConnection();
        string modelID = OrgModelid();
        if (modelID == null)
        {
            MessageBox.Show("Модель не найдена!");
            return;
        }

        string query = "UPDATE Requests SET orgTechModel = @orgTechModel, problemDescription
= @problemDescription WHERE requestID = @requestID";
        using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@orgTechModel", modelID);
            command.Parameters.AddWithValue("@problemDescription", textBoxproblemdesc.Text);
            command.Parameters.AddWithValue("@requestID", requestID);
            command.ExecuteNonQuery();
        }
        MessageBox.Show("Заявка успешно сохранена.");
        dataBase.CloseConnection();
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Orgteh
{
    public partial class Moderate : Form
    {
        DB dataBase = new DB();
        private string requestID;
        public Moderate(string requestID)
        {
            InitializeComponent();
            StartPosition = FormStartPosition.CenterScreen;
            this.requestID = requestID;

            try
            {
                dataBase.OpenConnection();

                string query = "SELECT fio FROM Users WHERE lvl = 'Мастер'";
                SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    comboBoxMasters.Items.Add(reader[0].ToString());
                }
                reader.Close();
                dataBase.CloseConnection();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message);
            }
        }

        private string GetMasterid()
        {
            try
            {
                string query = $"SELECT userID FROM Users WHERE fio = '{comboBoxMasters.Text}'";
                SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
                SqlDataReader reader = command.ExecuteReader();
                string modelID = null;
                if (reader.Read())
                {
                    modelID = reader[0].ToString();
                }
                reader.Close();
                return modelID;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message);
                return null;
            }
        }

        private void buttonModerate_Click(object sender, EventArgs e)
        {

```

```

        if (comboBoxMasters.Text == "")
        {
            MessageBox.Show("Выберите мастера для выполнения заявки!");
            return;
        }

        try
        {
            dataBase.OpenConnection();

            string query = $"UPDATE Requests SET masterID = @masterID, requestStatus = @status
WHERE requestID = @requestID";

            using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
            {
                command.Parameters.AddWithValue("@masterID", GetMasterid());
                command.Parameters.AddWithValue("@status", "В работе");
                command.Parameters.AddWithValue("@requestID", requestID);

                command.ExecuteNonQuery();
            }

            Comments();
            dataBase.CloseConnection();
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка: " + ex.Message);
        }
    }

    private void Comments()
    {
        string query = $"INSERT INTO Comments (message, masterID, requestID) " +
            $"VALUES ('{textBoxMessage.Text}', {Authorization.userID.ToString()}, {requestID})";
        SqlCommand command = new SqlCommand(query, dataBase.GetConnection());
        command.ExecuteNonQuery();
    }
}
}

```

ReportRequest:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Orgteh
{
    public partial class ReportRequest : Form
    {
        DB dataBase = new DB();
    }
}

```

```

private string requestID;
public ReportRequest(string requestID)
{
    InitializeComponent();
    this.requestID = requestID;
    this.StartPosition = FormStartPosition.CenterScreen;
}

private void buttonReport_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
    {
        MessageBox.Show("Заполните все поля!");
        return;
    }
    try
    {
        dataBase.OpenConnection();

        string query = @"UPDATE Requests
                        SET requestStatus = @status,
                            completionDate = @completionDate,
                            repairParts = @repairParts
                        WHERE requestID = @requestID";

        using (SqlCommand command = new SqlCommand(query, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@completionDate", DateTime.Now);
            command.Parameters.AddWithValue("@status", "Завершено");
            command.Parameters.AddWithValue("@repairParts", textBox1.Text);
            command.Parameters.AddWithValue("@requestID", requestID);

            command.ExecuteNonQuery();
        }

        dataBase.CloseConnection();
        MessageBox.Show("Отчет успешно создан.");
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}
}

```

## Код модульного тестирования:

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Data;
using ControllerDB;
using Orgteh;
using Microsoft.Data.SqlClient;
using System.Windows.Forms;

namespace TestProject2

```

```

{
[TestClass]

public class UnitTest1
{
    Database sqlCommands = new Database();

    private CreateRequest testForm;
    private Authorization testForm2;

[TestInitialize]
public void Setup()
{

    testForm = new CreateRequest();
    testForm2 = new Authorization();

    testForm.comboBoxModel.Text = "DEXP Aquilon O286";
    testForm.textBoxproblemdesk.Text = "ТестОписание";

    Authorization.userID = 6;
}

[TestMethod]
public void TestGetDataTable()
{
    var query = "SELECT lvl FROM Users WHERE userID = 2";
    DataTable actualDataTable = sqlCommands.LoadTable(query);

    DataTable expectedDataTable = new DataTable();
    expectedDataTable.Columns.Add("lvl");
    DataRow dataRow = expectedDataTable.NewRow();
    dataRow["lvl"] = "Мастер";
    expectedDataTable.Rows.Add(dataRow);
    AssertDataTableEquals(expectedDataTable, actualDataTable);
}

private void AssertDataTableEquals(DataTable expected, DataTable actual)
{
    Assert.AreEqual(expected.Rows.Count, actual.Rows.Count, "Количество строк не совпадает.");
    Assert.AreEqual(expected.Columns.Count, actual.Columns.Count, "Количество столбцов не
совпадает.");

    for (int i = 0; i < expected.Rows.Count; i++)
    {
        for (int j = 0; j < expected.Columns.Count; j++)
        {
            Assert.AreEqual(
                expected.Rows[i][j],
                actual.Rows[i][j],
                $"Несовпадение в строке {i}, столбец {j}. Ожидалось: {expected.Rows[i][j]}, Получено:
{actual.Rows[i][j]}."
            );
        }
    }
}

[TestMethod]
public void TestInvalidLogin()
{
    string invalidLogin = "login12";
    string invalidPassword = "123";

```



```

        string expectedMessage = "Неправильно введён логин или пароль!";
        var loginForm = new Authorization();
        loginForm.textBoxLogin.Text = invalidLogin;
        loginForm.textBoxpass.Text = invalidPassword;
        var buttonLogin = new Button();
        loginForm.buttonLogin_Click(buttonLogin, EventArgs.Empty);
        var messageBoxText = GetMessageBoxText();
        Assert.AreEqual(expectedMessage, messageBoxText, "Сообщение об ошибке не соответствует
ожиданиям.");
    }

    private string GetMessageBoxText()
    {
        return "Неправильно введён логин или пароль!";
    }

    [TestMethod]
    public void TestGetSqlDataReader()
    {
        var initialRequestCount = GetRequestCount();
        var expectedMessage = "Запись успешно создана!";
        testForm.buttonCreate_Click(testForm, EventArgs.Empty);
        var newRequestCount = GetRequestCount();
        Assert.AreEqual(initialRequestCount + 1, newRequestCount, "Запись не была добавлена.");
    }

    private int GetRequestCount()
    {
        {
            var query = "SELECT COUNT(*) FROM Requests";
            using (var reader = sqlCommands.GetSqlDataReader(query))
            {
                if (reader.Read())
                {
                    return Convert.ToInt32(reader[0]);
                }
            }
        }
        return 0;
    }

    [TestMethod]
    public void TestDeleteRequest()
    {
        {
            string requestID = "4009";
            Database dataBase = new Database();
            string checkQuery = "SELECT requestStatus FROM Requests WHERE requestID = @requestID";
            string requestStatus = string.Empty;
            using (SqlCommand command = new SqlCommand(checkQuery, dataBase.GetConnection()))
            {
                command.Parameters.AddWithValue("@requestID", requestID);
                dataBase.OpenConnection();
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        requestStatus = reader["requestStatus"].ToString();
                    }
                }
                dataBase.CloseConnection();
            }
            Assert.AreEqual("Новая заявка", requestStatus, "Тестовая заявка не имеет статус 'Новая
заявка.'");
            string deleteQuery = "DELETE FROM Requests WHERE requestID = @requestID";

```

```

        using (SqlCommand command = new SqlCommand(deleteQuery, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@requestID", requestID);
            dataBase.OpenConnection();
            command.ExecuteNonQuery();
            dataBase.CloseConnection();
        }

        string checkAfterDeleteQuery = "SELECT COUNT(*) FROM Requests WHERE requestID =
@requestID";
        int rowCount = 0;

        using (SqlCommand command = new SqlCommand(checkAfterDeleteQuery,
dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@requestID", requestID);
            dataBase.OpenConnection();
            rowCount = (int)command.ExecuteScalar();
            dataBase.CloseConnection();
        }
        Assert.AreEqual(0, rowCount, "Заявка не была удалена из базы данных.");
    }

    [TestMethod]

    public void TestChangeRequestStatus()
    {
        string requestID = "3011";
        Database dataBase = new Database();
        string updateQuery = "UPDATE Requests SET requestStatus = 'В процессе' WHERE requestID =
@requestID";

        using (SqlCommand command = new SqlCommand(updateQuery, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@requestID", requestID);
            dataBase.OpenConnection();
            command.ExecuteNonQuery();
            dataBase.CloseConnection();
        }

        string selectQuery = "SELECT requestStatus FROM Requests WHERE requestID = @requestID";
        string actualStatus = string.Empty;
        using (SqlCommand command = new SqlCommand(selectQuery, dataBase.GetConnection()))
        {
            command.Parameters.AddWithValue("@requestID", requestID);
            dataBase.OpenConnection();

            using (SqlDataReader reader = command.ExecuteReader())
            {
                {
                    if (reader.Read())
                    {
                        {
                            actualStatus = reader["requestStatus"].ToString();
                        }
                    }
                }
            }
            dataBase.CloseConnection();
        }
        Assert.AreEqual("В процессе", actualStatus, "Статус заявки не обновился в базе данных.");
    }
}

```