

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
Московский техникум космического приборостроения

Код, специальности 09.02.07 Информационные системы и программирование

### Пояснительная записка

по учебной практике по профессиональному модулю

ПМ. 02 Осуществление интеграции программных модулей

Тема: Учёт заявок на ремонт оборудования

Выполнил студент Владимир Иван Сергеевич  
Курс 3 Группа ИКИТ-62  
Подпись студента (подпись)

Руководитель практики Б.И. Башкова О.В.  
(подпись) (фамилия, имя, отчество)

Москва, 2024 г.

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

Московский техникум космического приборостроения МГТУ имени Н.Э. Баумана

## ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ на прохождение учебной практики

по: ПМ.02 Осуществление интеграции программных модулей в объеме 90 часов

Студент Владимиров Иван Сергеевич, 09.02.07, ТИП-62  
(фамилия, имя, отчество; индекс специальности, группа)

Студент во время прохождения учебной практики с «30» января 2024 года по «15» февраля 2024 года должен:

### Ознакомиться:

1. с моделями процесса разработки программного обеспечения;
2. основными принципами процесса разработки программного обеспечения;
3. основными подходами к интегрированию программных модулей;
4. основами верификации и аттестации программного обеспечения.

### Уметь:

1. использовать выбранную систему контроля версий;
2. использовать методы для получения кода с заданной функциональностью и степенью качества.

### Получить практический опыт:

1. в интеграции модулей в программное обеспечение;
2. в отладке программных модулей.

### По итогам производственной практики студент обязан представить:

1. Отчет по учебной практике (Титульный лист, Дневник практики, Аттестационный лист, Письменный отчет)

Дата выдачи задания «30» января 2024г.

Руководитель практики от техникума

Студент

Башкова О.В.  
(подпись, дата) Владимиров И.С.  
(подпись, дата) (Фамилия И.О.)

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Постановка задачи.....	6
2.1 Описание предметной области .....	7
2.2 Проектирование системы с помощью UML-диаграмм .....	10
2.3 Функциональные требования к системе .....	16
2.4 Пользователи системы и их привилегии.....	17
2.5 Назначение программы.....	18
2.6 Требования к функциональным характеристикам .....	19
2.7 Требования к надежности.....	19
2.8 Требования к составу и параметрам технических средств .....	20
2.9 Требования к программной совместимости .....	21
2.10 Требования к программной документации.....	21
2 Проектирование программной системы .....	22
3.1 Структура программы.....	22
3.2 Схема алгоритма хеш-функции .....	23
3.3 Схема алгоритма регистрации клиента.....	24
3.4 Схема данных .....	25
3.5 Тестирование программы .....	29
3.6 Руководство пользователя .....	36
ЗАКЛЮЧЕНИЕ .....	43
СПИСОК ЛИТЕРАТУРЫ.....	44
ПРИЛОЖЕНИЯ.....	45
ПРИЛОЖЕНИЕ А .....	46
A.1 Листинг программы .....	47
A.2 Листинг файла Authorization.cs .....	50
A.3 Листинг файла DB.cs .....	56
A.4 Листинг файла AdminMenuForm.cs .....	58
A.5 Листинг файла AdminsProfileForm.cs.....	62
A.6 Листинг файла ApplicationForm.cs.....	73
A.7 Листинг файла ApplicationListForm.cs .....	80
A.8 Листинг файла EditApplicationAdminForm.cs.....	85
A.9 Листинг файла EditApplicationForm.cs.....	105
A.10 Листинг файла EmployeeMenuForm.cs.....	113
A.11 Листинг файла EmployeeProfileAdminForm.cs .....	117
A.12 Листинг файла EmployeeProfileForm.cs .....	131
A.13 Листинг файла LoginForm.cs .....	141
A.14 Листинг файла RegisterForm.cs .....	151
A.15 Листинг файла StatisticsForm.cs .....	158
A.16 Листинг файла StatusApplicationForm.cs.....	166
A.17 Листинг файла StatusAppplicationEmployeeForm.cs.....	176
A.18 Листинг файла UsersMenuForm.cs .....	192
A.19 Листинг файла UsersProfileAdminForm.cs .....	196
A.20 Листинг файла UsersProfileForm.cs .....	210
ПРИЛОЖЕНИЕ Б.....	222

# ВВЕДЕНИЕ

В современной жизни учет и контроль ресурсов являются одними из важнейших задач. Контроль расходов, хранение информации, а также анализ полученных данных становятся неотъемлемой частью работы в различных организациях. Учитывая значимость вопросов учета и контроля ресурсов, актуальна разработка программного продукта, который позволит проводить автоматизированный учет и контроль этих ресурсов.

Разработка модулей программного обеспечения для компьютерных систем позволяет создать автоматизированную систему учета и контроля ресурсов, которая оптимизирует работу организации и обеспечивает доступ к нужной информации.

Создание программного обеспечения для автоматизации задач в данной области позволяет сократить время, требуемое на заполнение данных пользователей и обработку информации. Приложение, разработанное для автоматизации процесса, значительно упрощает работу и увеличивает эффективность использования ресурсов.

Использование информационных систем со стороны работы приложения позволит избежать большинство проблем, которые были обусловлены, использованием бумажных носителей, например, таких как:

- повысится оперативность учета;
- уменьшится вероятность потери информации;
- на расчеты больше не будет влиять человечески фактор, потому что их будет выполнять машина по строго определенному алгоритму.
- информация будет выдаваться и храниться в структурированном виде;
- поиск информации будет занимать считанные секунды;

Предметом исследования является процесс разработки модулей программного обеспечения для компьютерных систем посредством создания базы данных MySQL, а также выполнение запросов с использованием СУБД MySQL Workbench.

Целью работы является создание приложения, которое будет предоставлять клиентам возможность создания заявки по ремонту компьютерных средств, а также базы данных, которая будет содержать информацию о предметной области.

# 1 Постановка задачи

Предметная область предполагает онлайн – регистрацию заявки, хранение данных о ней, о выполненных работах, а также информацию о предоставляемых данных самим пользователем.

Целью данной работы является разработка автоматизированного программного обеспечения для учета заявок в области компьютерных систем. Программное обеспечение предназначено для онлайн-регистрации заявок, хранения информации о заявках, выполненных работах, а также информации, предоставляемой пользователями.

Основной задачей программного обеспечения является обеспечение пользователя возможностью быстрой и удобной регистрации заявок, а также полного контроля за данными, получаемыми от клиентов. В итоге работы системы клиентам будет предоставлен документ, содержащий полную информацию о заявке и ее деталях, который заносится в базу данных для последующего использования.

Сотрудники сайта обеспечивают обратную связь и полный контроль поступаемых данных от клиента, затем формируют различные отчеты.

Для достижения и решения поставленной цели необходимо решить следующие задачи:

- описать предметную область;
- спроектировать систему с помощью UML-диаграмм;
- описать функциональные требования к системе;
- описать основных пользователей системы и их привилегий;
- создать структуру программы;
- создать схему данных;
- создать приложение;
- провести тестирование программы;
- составить руководство пользователя.

## 2 Проектирование программной системы

### 2.1 Описание предметной области

Основная цель учёта заявок на ремонт оборудования – эффективное и оперативное осуществление ремонтных работ с минимизацией простоев и удовлетворением запросов клиентов или сотрудников. Эта предметная область широко используется в различных сферах деятельности, таких как сервисные услуги, производство, информационные технологии и другие.

Предметная область учёта заявок на ремонт оборудования касается процесса подачи, обработки и учёта заявок на ремонт различного оборудования.

В данной области включены следующие основные составляющие:

1. Заявка на ремонт: это информация, предоставленная клиентом или сотрудником о неисправности оборудования, которое требует ремонта. Заявка может содержать данные о типе оборудования, его серийном номере, описании проблемы и другой важной информации.

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке и её приоритете.

3. Обработка заявки: процесс, включающий анализ заявки, определение её приоритетности и назначение исполнителя (ремонтного специалиста) для задачи. В процессе обработки может потребоваться дополнительная информация или уточнение деталей проблемы у клиента или сотрудника.

4. Исполнение заявки: фактическое выполнение ремонта оборудования. В этом этапе назначенный исполнитель ремонтирует оборудование, вносит необходимые изменения или заменяет неисправные компоненты. Важно отметить, что на этом этапе

могут возникать необходимость заказа запчастей или координации работ с другими специалистами.

5. Отчётность и информирование: важной составляющей учёта заявок на ремонт является фиксация и отчёт о выполненной работе. После завершения ремонта, исполнитель должен предоставить отчёт о проделанной работе, включая информацию о затраченных ресурсах (время, материалы, стоимость), причине неисправности и оказанной помощи.

6. Мониторинг и анализ: этот этап предполагает контроль и анализ процесса учёта заявок на ремонт. Важно отслеживать и анализировать время обработки заявок, качество выполненных работ, расходы и прочие параметры, которые могут помочь в оптимизации и улучшении процесса.

На рисунке 2.1 представлена контекстная диаграмма IDEF0, описывающая процесс создания заявки клиентом.

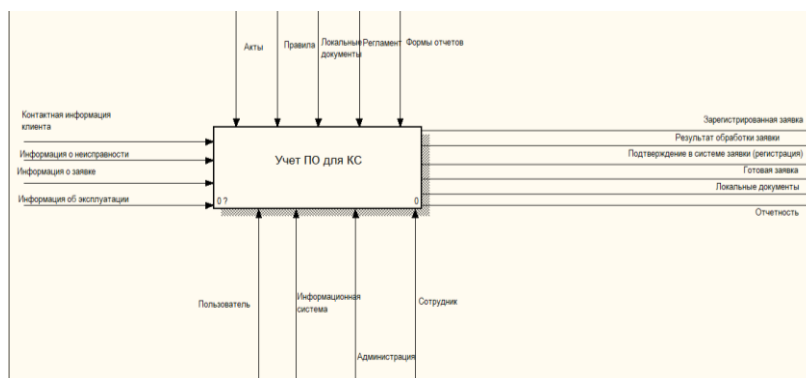


Рисунок 2.1 – Контекстная диаграмма IDEF0

На рисунке 2.2 представлена диаграмма декомпозиции IDEF0

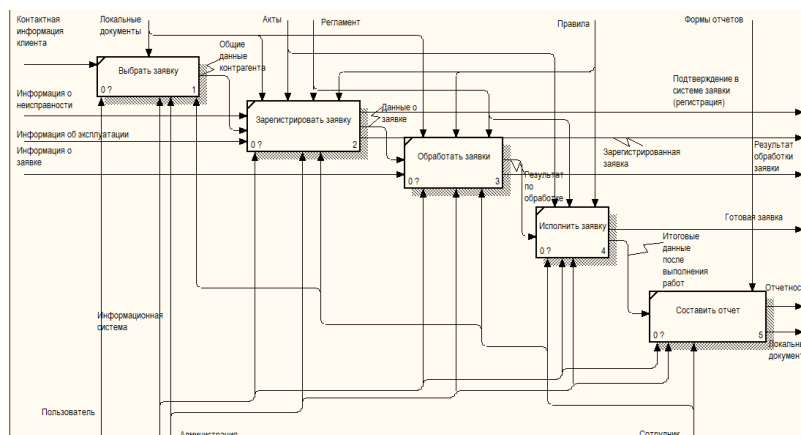


Рисунок 2.2 – Декомпозиция контекстной диаграммы



Ниже, на рисунках 2.3 и 2.4, представлена схема DFD, на ней отражен принцип работы приложения и процессы, связанные с потоками данных.

Далее, на рисунке 2.3 будет представлена диаграмма DFD, которая поможет лучше понять принцип работы приложения с точки зрения программы, связанных с потоками данных. Сотрудники, пользователи, администраторы могут добавить информацию, а также при необходимости удалить или изменить её.

На рисунке 2.3 представлена контекстная диаграмма DFD, изображающая потоки данных в программе.

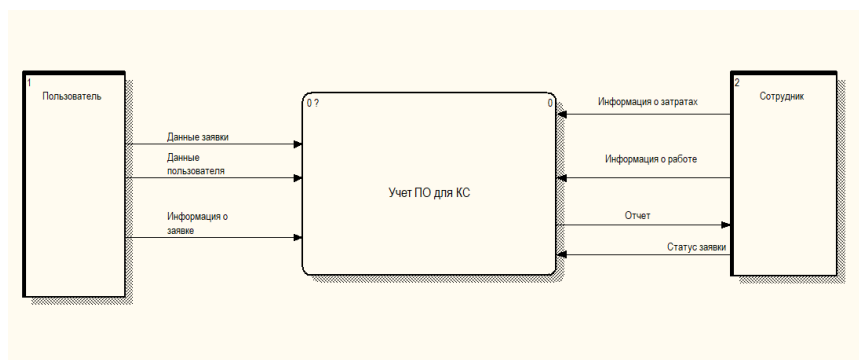


Рисунок 2.3 – Контекстная диаграмма DFD

На рисунке 2.4 представлена диаграмма декомпозиции DFD.

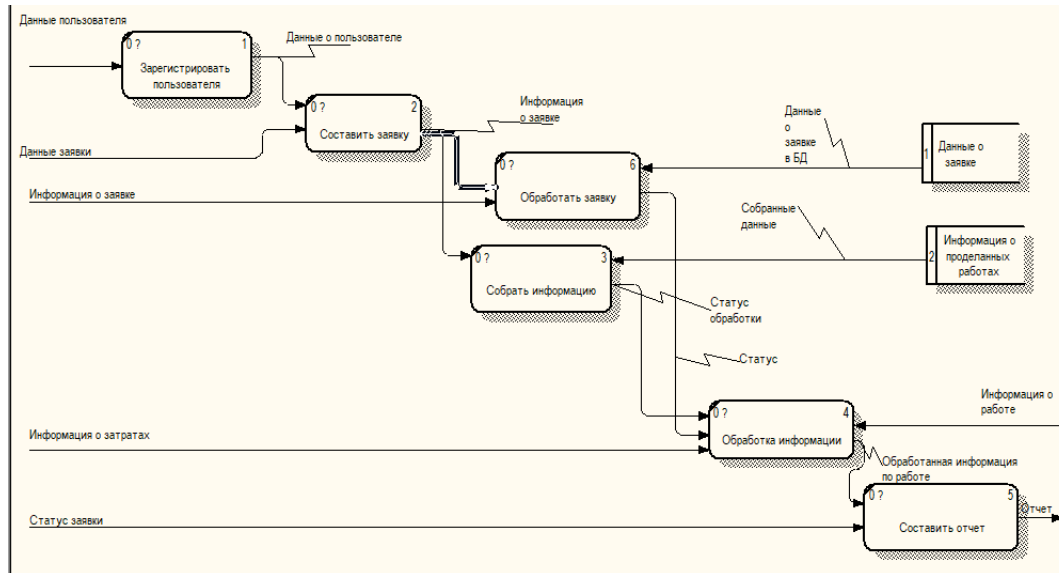


Рисунок 2.4 – Декомпозиция контекстной диаграммы DFD

После обращения пользователя происходит сбор и обработка данных. Информация о заявке и приделанной работе берется из базы данных.

Из этого мы можем выделить главную задачу приложения – это быстрая обработка заказов и обслуживание техники, а также при необходимости автоматическое создание документов.

Подводя итог, можно выделить основные функциональные требования к разрабатываемому программному продукту:

1. Понятный для пользователя интерфейс и удобная навигация по приложению.
2. Обеспечение защиты информации от несанкционированного доступа и изменений.
3. Возможность заполнения данных о заявке.
4. Функция автоматического заполнения документа по шаблону.
5. Функция проверки вводимых данных.
6. Перехват и вывод ошибок в доступной форме в интерфейс при их возникновении.
7. Занесение всех вводимых данных в БД.

## 2.2 Проектирование системы с помощью UML-диаграмм

Для начала рассмотрим диаграмму использования.

На рисунке 2.5 представлена диаграмма вариантов использования.

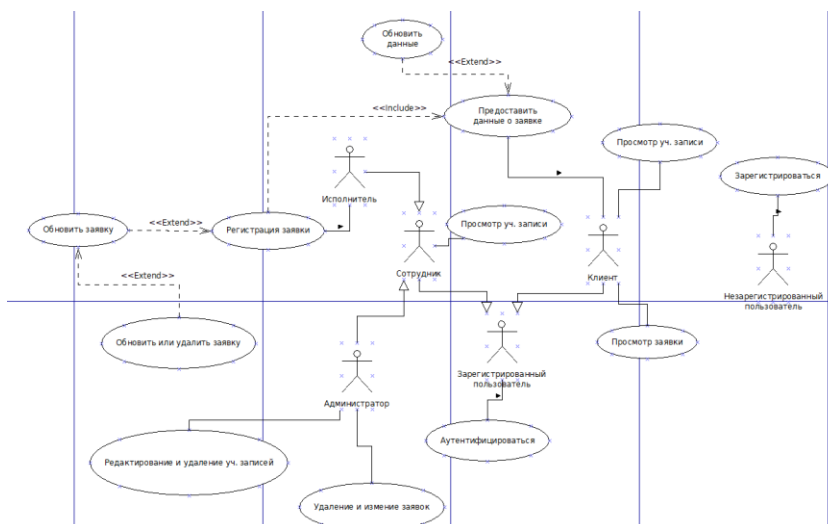


Рисунок 2.5 – Диаграмма вариантов использования

На диаграмме вариантов использования представлены актеры: сотрудник, администратор, исполнитель, клиент, зарегистрированный пользователь, незарегистрированный пользователь. Клиент проходит регистрацию (связь - ассоциация), после регистрации расширяются данные о клиенте (связь - расширение extend), регистрация включает в себя аутентификацию пациента и позволяет клиенту зарегистрировать заявку (связь – включение include), расширение для регистрации заявки является возможность ее обновления и удаления. Далее идет актер Сотрудник у него ассоциативная связь с Регистрацией заявки, именно он регистрирует заявку в БД. Рассмотрим незарегистрированного пользователя, он должен зарегистрироваться в системе (связь – ассоциация). Сотрудник и клиент связываются обобщением – зарегистрированный пользователь, данный актер должен аутентифицируется (связь – ассоциация). Администратор и сотрудник связываются обобщением – сотрудник данный актер может просмотреть учетную запись(связь – ассоциация).

Далее рассмотрим диаграмму последовательности.

На рисунке 2.6 представлена диаграмма последовательности для регистрации заявки.

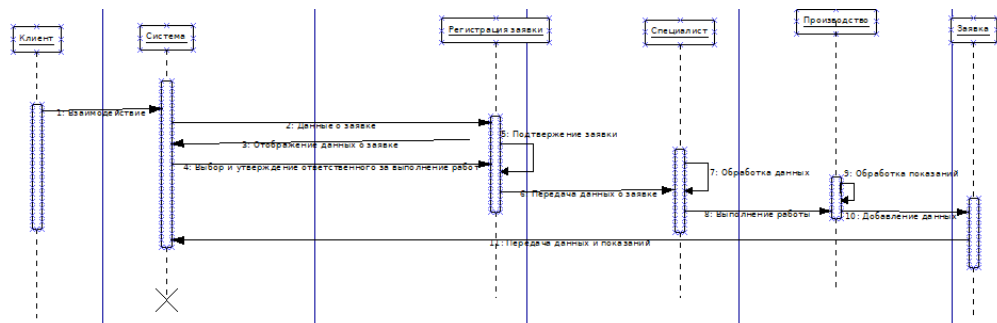


Рисунок 2.6 – Диаграмма последовательности

На данной диаграмме пациент вводит данные в поля ввода, которые в свою очередь проверяются на корректность. Далее происходит регистрация заявки в БД.

На рисунке 2.7 представлена диаграмма последовательности для регистрации в БД.

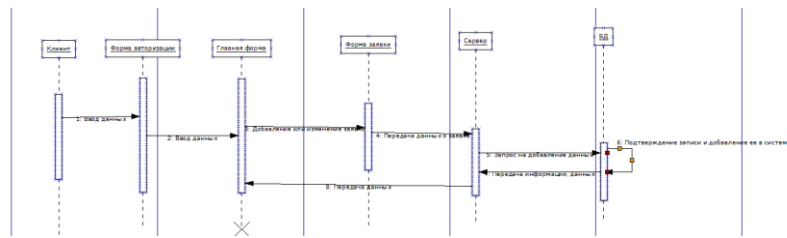


Рисунок 2.7 – Диаграмма последовательности

В данных диаграммах Клиент вводит данные в систему на форме регистрации(аутентифицируется), на форме регистрации заявки клиент вводит данные о заявке и они отображаются, система же утверждает ответственного за выполнения работы. далее данные передаются на сервер в систему, происходит подтверждение данных в бд, и при подтверждении данные и информация передается назад на сервер. После все отображается на форме регистрации заявки. На этом жизненный цикл системы заканчивается.

Рассмотрим диаграмму деятельности UML

На рисунке 2.8 представлена диаграмма деятельности UML

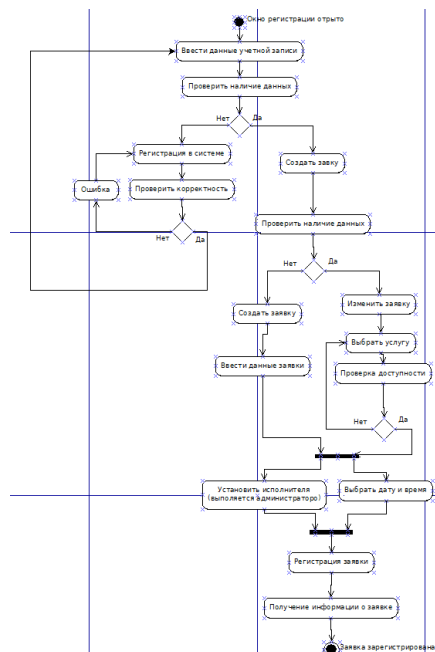


Рисунок 2.8 – Демонстрация диаграммы

На диаграмме деятельности рассмотрена регистрация в системе. В самом начале пользователь вводит данные в системе, а именно логин и пароль, далее в системе проверяется, есть ли данные о таком пользователе. Если нет, то необходимо пройти регистрацию и ввести свои данные. Если же да, то пользователь может создать заявку или изменить ее, проверяется доступность данной услуги в системе и наличие данных в системе, далее если получен доступ, то администратор устанавливает исполнителя работы, далее происходит регистрирование заявки. В результате всех действий заявка зарегистрирована.

Далее следует диаграмма классов

Классы-сущности:

- 1) Клиент
- 2) Заявка
- 3) Сервер
- 4) Окно программы
- 5) Форма регистрации заявки
- 6) Окно заявки

Таблица 2.1 - «Пациент»

Параметр	Значение
Комментарий	Класс, представляющий собой клиента
Атрибуты	FIO: String – ФИО пользователя Position: Datetime – должность Department: String – Отдел Contact_Details:String – Контактные данные Phone_Number:String – Номер телефона пользователя Address:String – Адрес пользователя
Операции	EnterData(): String – Ввести данные Save/deleteData():Void– сохранить/удалить данные AddEntries():Void – добавить заявку GetEntries (): String – Получить заявку Save/deleteEntries ():String – Сохранить/удалить заявку

Таблица 2.2 - «Заявка»

Параметр	Значение
Комментарий	Класс, представляющий запрос клиента
Атрибуты	Status:Boolean – Статус заявки Date:Date – Дата заявки Number: Integer – Номер заявки Performer: String – ФИО исполнителя
Операции	GetAppointment():Boolean – Получить заявку CancelAppointment(): Void – Отменить заявку

Таблица 2.3 - «Сервер»

Параметр	Значение
Комментарий	Класс, представляющий поле ввода
Атрибуты	-
Операции	ProcessRequest: Boolean– Обработать запрос SendRequest: String– Отправить запрос GetAnswer: String– Получить ответ AddData():String – Добавить данные о заявке GetData():String – Получить данные

Таблица 2.4 - «Окно программы»

Параметр	Значение
Комментарий	Класс, представляющий объект сервера
Атрибуты	-
Операции	SendActionToUser():Void – Отправить ответ пользователю Interact():Void – Взаимодействовать с системой

Таблица 2.5 - «Форма регистрации»

Параметр	Значение
Комментарий	Класс, представляющий объект сервера
Атрибуты	Password: String – Пароль Login: String – Логин
Операции	Registers():Void – Зарегистрироваться CompletelyEnter():Void – Войти в ученую запись EnterData():Void – Ввести данные

Таблица 2.6 - «Окно заявки»

Параметр	Значение
Комментарий	Класс, представляющий объект сервера
Атрибуты	ListApplication: String – Список заявок Schedule: String – расписание
Операции	ChangeApplications (): Void – Отменить заявку SignUpApplications ():Void – Регистрация заявки

На рисунке 2.9 представлена связь классов друг с другом.

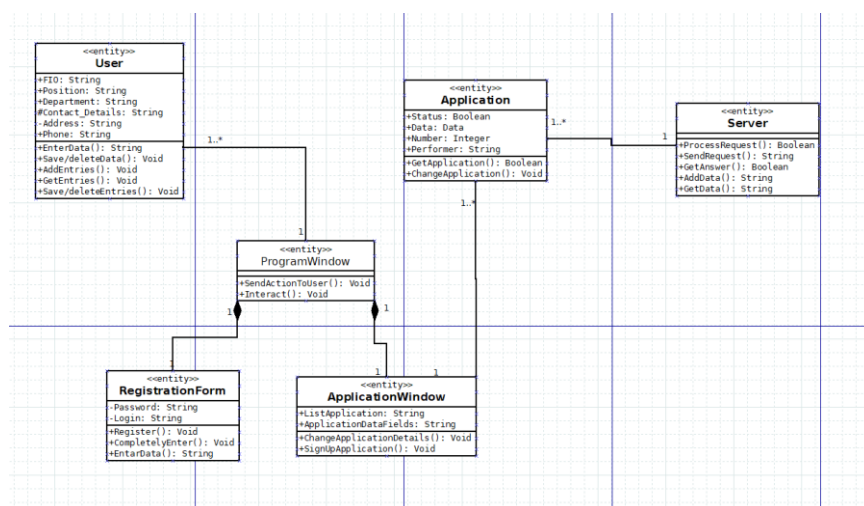


Рисунок 2.9 –Связь классов

«Клиент» и «Окно программы» связаны между собой ассоциацией, «Форма регистрации заявки» и «Окно заявки» является частью запроса, поэтому связывается композицией, «Заявка» и «Окно Заявки» имеют ассоциативную связь, как и «Заявка» и «Сервер».

Рассмотрим диаграмму компонентов UML

На рисунке 2.10 представлена диаграмма компонентов UML

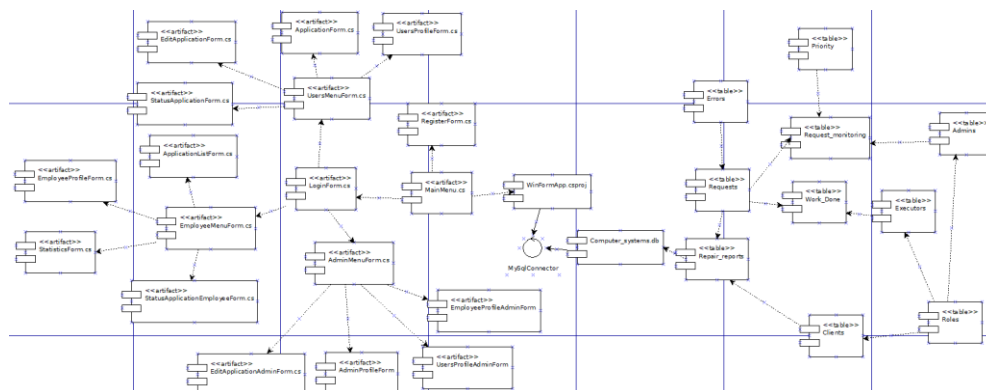


Рисунок 2.10 – Демонстрация диаграммы

Программа состоит из главного меню и 18 форм, представляющих конкретный функционал приложения. Подключение к базе данных `Computer_systems.db` происходит с помощью создания класса соединения `MySQLConnector`. В базе данных есть таблицы: заявки, исполнители, отчеты по ремонту, мониторинг заявок, ошибки, профиль.

## 2.3 Функциональные требования к системе

Функциональные требования системы включают в себя следующие основные функции:

Возможность добавления заявок в базу данных с указанием следующих параметров:

- номер заявки;
- дата добавления;
- оборудование, которое требует ремонта;
- тип неисправности;
- описание проблемы;



- клиент, который подал заявку;
- статус заявки (в ожидании, в работе, выполнено).

Возможность редактирования заявок:

- Изменение этапа выполнения (выполнено, в работе, не выполнено);
- Изменение описания проблемы;
- Изменение, ответственного за выполнение работ.

Возможность отслеживания статуса заявки:

- Отображение списка заявок;
- Получение уведомлений о смене статуса заявки;
- Поиск заявки по номеру или по параметрам.

Возможность назначения ответственных за выполнение работ:

- Добавление исполнителя к заявке;
- Отслеживание состояния работы и получение уведомлений о ее

завершении;

- Исполнитель может добавлять комментарии на форме заявки.

Расчет статистики работы отдела обслуживания:

- Количество выполненных заявок;
- Среднее время выполнения заявки;
- Статистика по типам неисправностей.

## 2.4 Пользователи системы и их привилегии

Список пользователей и их привилегии в системе:

### 1. Клиент:

- изменение личных данных в системе;
- предоставление личных данных;
- просмотр собственных данных;
- регистрация заявок;

- изменение данных заявки.

## 2. Администратор:

- регистрация клиентов в системе;
- просмотр данных клиентов;
- изменение данных клиентов;
- добавление клиентов в БД;
- назначение исполнителей работ.

## 3. Исполнитель работ:

- просмотр заявок;
- добавление информации в систему;
- получение результатов (статистики);
- оформление заявок клиентов.

## 4. Незарегистрированный пользователь:

- регистрация в системе.

## 5. Зарегистрированный пользователь:

- аутентификация в системе.

## 2.5 Назначение программы

Назначение технического задания на тему "Учёт заявок на ремонт оборудования" заключается в определении технических требований к разработке программного продукта, который будет предназначен для эффективного учета информацией о заявках, планирования деятельности исполнителей работ, управления исполнителями и обеспечения надежного хранения данных о КС. Приложение разрабатывается с учетом потребностей не только администратора, но и персонала, клиентов, обеспечивая различные возможности, такие как создание, изменение и удаление заявок и автоматизацию процессов учета и анализа.

## 2.6 Требования к функциональным характеристикам

Система должна обеспечивать возможность выполнения следующих функций:

- инициализация системы, то есть ввод списков специалистов, список уже зарегистрированных пользователей, информации о заявках;
- хранение информации о заявках;
- ввод данных и возможность коррекции текущей информации о пользователе, а также заявке;
- видимость и получение сведений о всех текущих заявках;
- изменение информации о заявке и изменение статуса выполнения заявки.

Исходные данные:

- списки исполнителей;
- списки зарегистрированных клиентов;
- списки зарегистрированных заявок.

Организация входных и выходных данных:

Входные данные поступают с клавиатуры. Выходные данные должны отображаться на мониторе.

Основной режим использования системы – ежедневная работа.

Предполагается, что на выходе должна обеспечить по требованию пользователя выход следующей информации:

- информация о заявке;
- информация о пользователях;
- список пользователей;
- информация о сотрудниках.

## 2.7 Требования к надежности

Система должна иметь стандартные и продвинутые настройки безопасности.

Для обычного пользователя обеспечить ограничение доступа к изменениям настроек системы, данная возможность есть только у администратора.

Предусматривать контроль вводимой информации и блокировку некорректных действий пользователя при работе с системой, обеспечить максимально возможную защиту от ошибок системы.

Кроме того, необходимо обеспечить возможность настройки частоты резервного копирования и восстановление – это подразумевает возможность в любой момент вернуть настройки системы и в целом базы данных к прошлому виду.

Надежное функционирование информационной системы должно быть обеспечено выполнением организационно-технических мероприятий, таких как:

- Использование лицензионного программного обеспечения;
- Организация бесперебойного питания путем использования блоков бесперебойного питания для сервера;
- Любые изменения информации в базе данных должны быть целостны и непротиворечивы.

## 2.8 Требования к составу и параметрам технических средств

Система должна работать на IBM совместимых персональных компьютерах.

Минимальная конфигурация:

- тип процессора Intel Core i3 или AMD Ryzen 3и выше;
- тактовая частота процессора 3.0 ГГц;
- объем оперативного запоминающего устройства около 8 Мб и выше;
- объем свободного места на жестком диске определяется объемом данных и базы данных системы, однако ожидается около 50 Мб и выше;
- платформа 32-х разрядная и выше;

Рекомендуемая конфигурация:

- тип процессора Intel Core i5 или AMD Ryzen 5;
- тактовая частота процессора 4.0 ГГц;

- объем оперативного запоминающего устройства около 128 Мб;
- объем свободного места на жестком диске около 60 Мб;
- платформа 32-х, 64-х разрядная.

## 2.9 Требования к программной совместимости

Программное обеспечение должно быть совместимо с операционными системами macOS, Windows 7/10/11 в соответствии с оборудованием организации.

## 2.10 Требования к программной документации

Документация должна быть разделена на описание интерфейсной части системы и низкоуровневой, а также содержать требования к конфигурации системы, руководство по подключению и развертыванию пошаговое обучение основным и продвинутым функциям программного обеспечения, объяснение нюансов работы с ним и с системой в целом (ограничения, предупреждения). Разъяснения сообщений, которые система может оставлять при журналировании, отображении ошибок и иных уведомлений.

Также необходим перечень правил использования ПО:

- руководство пользователя;
- руководство администратора.

### 3 Проектирование программной системы

#### 3.1 Структура программы

На рисунке 3.1 представлена диаграмма, описывающая структуру программы.

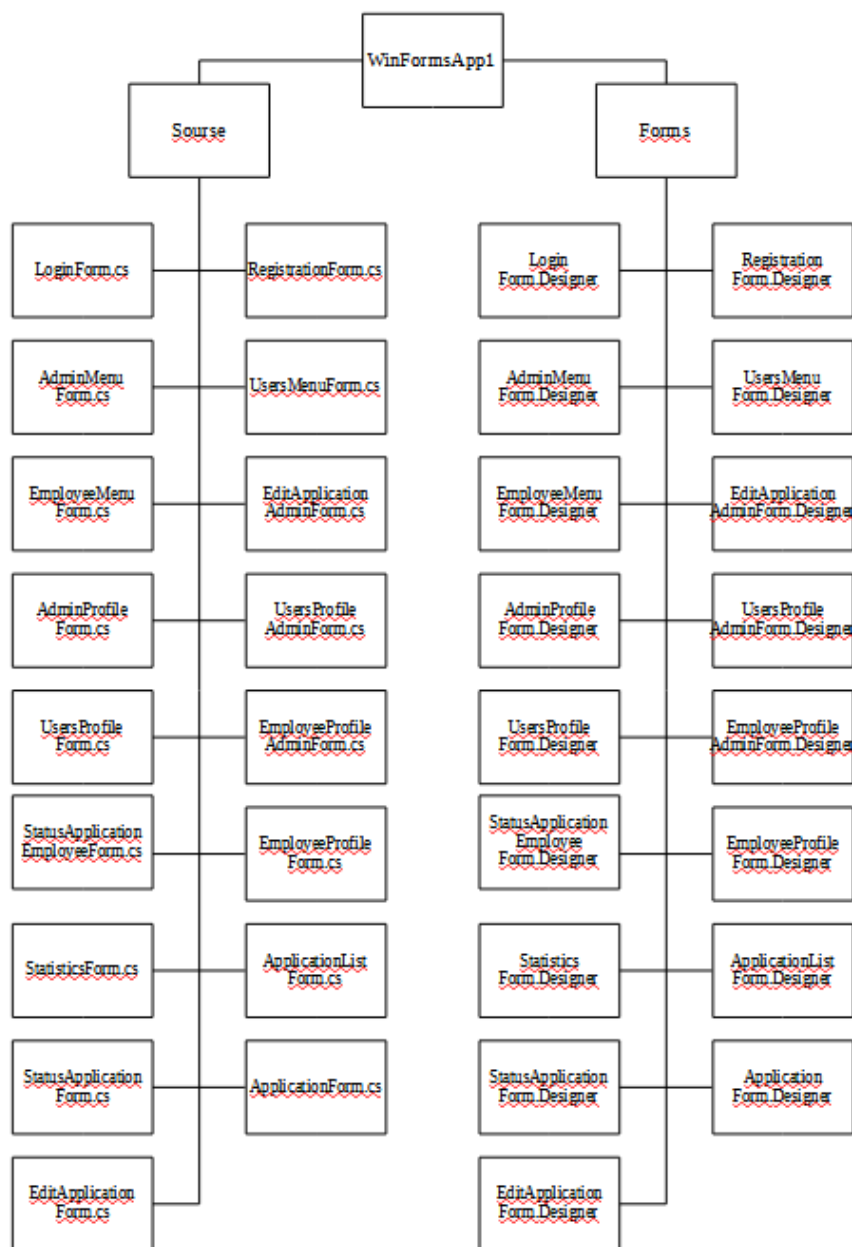
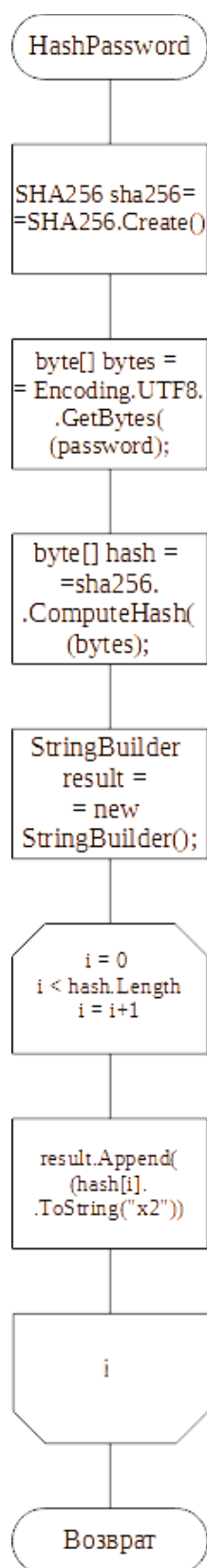
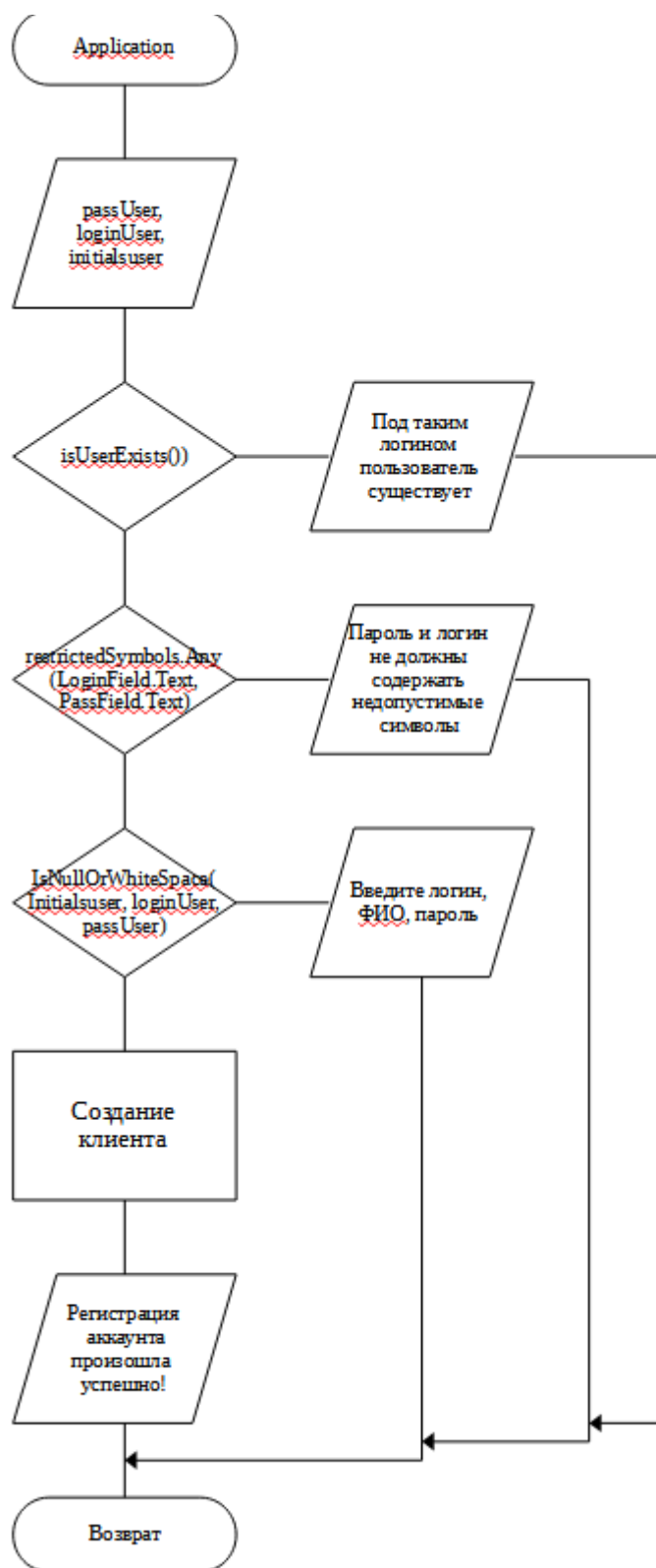


Рисунок 3.1 – Структура программы

### 3.2 Схема алгоритма хеш-функции



### 3.3 Схема алгоритма регистрации клиента





### 3.4 Схема данных

Основной задачей проектирования базы данных является обоснованный выбор такой структуры, которая обеспечит согласованное взаимодействие всех компонентов информационной системы согласно заданным требованиям и ограничениям по ряду интегрированных параметров.

Проектирование базы данных – процесс создания необходимых схем (концептуальной и логической). Для их создания необходимо выделить главные сущности:

1. Errors – данные об ошибках которые могут возникнуть в КС.
2. Requests – заявки, которые оставляет пользователь.
3. Work\_done – выполненная работа.
4. Executors– информация об исполнителе.
5. Admins– информация об админе.
6. Clients– информация о клиенте.
7. Roles – информация о ролях, которые есть в системе.
8. Request\_monitoring– информация о поступающих заявках.
9. Priority – информация о приоритете выполнения заявок.
10. Repair\_reports– итоговый отчет о заявке.

Все связи являются бинарными и представляются линиями, соединяющими родительскую сущность с одной или несколькими сущностями потомками.

На рисунке 3.2 представлена диаграмма IDEF1X базы данных, присутствующей в системе.

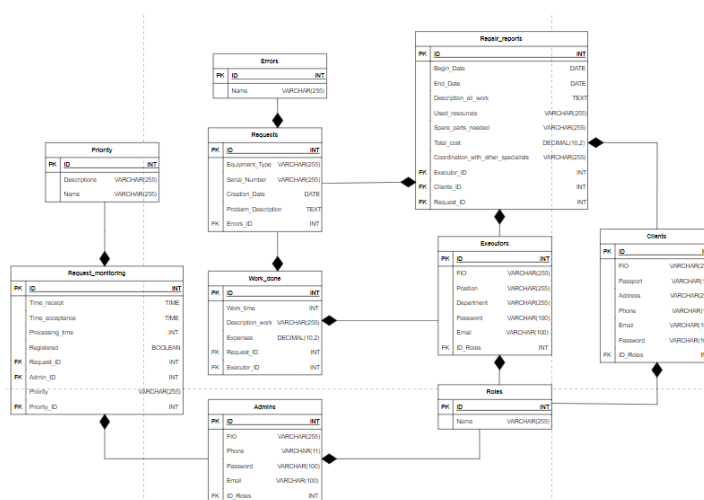


Рисунок 3.2 – Диаграмма IDEF1X

Ниже представлено структурное описание таблиц базы данных в таблицах 3.1–3.10

Таблица 3.1 - Описание таблицы Repair\_Reports

Идентификатор	Описание	Тип данных	Размер
ID	ID отчета работы	Числовой	Целый
Begin_Date	Начальная дата	Дата	
End_Date	Конечная дата	Дата	
Description_all_work	Описание всех проведенных работ	Текстовый	
Used_resources	Затраченные ресурсы	Текстовый	255
Spare_parts_needed	Потребность в запчастях	Текстовый	255
Total_cost	Итоговая цена	Числовой	С фиксированной точностью
Coordination_with_other_specialists	Координация с другими специалистами	Текстовый	255
Request_ID	ID заявки	Числовой	Целый
Executor_ID	ID исполнителя	Числовой	Целый
Clients_ID	ID клиента	Числовой	Целый

Таблица 3.2 - Описание таблицы Work\_done

Идентификатор	Описание	Тип данных	Размер
ID	ID выполненной работы	Числовой	Целый
Work_time	Затраченное на работу время	Числовой	Целый
Description_work	Описание работы	Текстовый	255
Expenses	Тип квартиры	Числовой	С фиксированной точностью
Request_ID	ID заявки	Числовой	Целый
Executor_ID	ID исполнителя	Числовой	Целый

Таблица 3.3 - Описание таблицы Admins

Идентификатор	Описание	Тип данных	Размер
ID	ID клиента	Числовой	Целый
FIO	ФИО клиента	Текстовый	100
Phone	Номер телефона	Текстовый	11
Password	Пароль	Текстовый	100
Email	Логин	Текстовый	100
ID_Roles	ID роли	Числовой	Целый

Таблица 3.4 - Описание таблицы Requests

Идентификатор	Описание	Тип данных	Размер
ID	ID заявки	Числовой	Целый
Equipment_Type	Тип оборудования	Текстовый	255
Serial_Number	Серийный номер	Текстовый	255
Creation_Date	Дата создания	Дата	
Problem_Description	Описание проблемы	Текстовый	
Errors_ID	ID неисправности	Числовой	Целый

Таблица 3.5 - Описание таблицы Errors

Идентификатор	Описание	Тип данных	Размер
ID	ID неисправности	Числовой	Целый
Name	Название	Текстовый	50

Таблица 3.6 - Описание таблицы Roles

Идентификатор	Описание	Тип данных	Размер
ID	ID роли	Числовой	Целый
Name	Название	Текстовый	100

Таблица 3.7 - Описание таблицы Executors

Идентификатор	Описание	Тип данных	Размер
ID	ID исполнителя	Числовой	Целый
FIO	ФИО исполнителя	Текстовый	255
Position	Должность	Текстовый	255
Department	Отдел	Текстовый	255
Email	Логин	Текстовый	100
Password	Пароль	Текстовый	100
ID_Roles	ID роли	Числовой	Целый
Phone	Номер телефона	Текстовый	11

Таблица 3.8 - Описание таблицы Clients

Идентификатор	Описание	Тип данных	Размер
ID	ID клиента	Числовой	Целый
FIO	ФИО клиента	Текстовый	255
Passport	Паспорт	Текстовый	10
Address	Адрес	Текстовый	255
Email	Логин	Текстовый	100
Password	Пароль	Текстовый	100
ID_Roles	ID роли	Числовой	Целый
Phone	Номер телефона	Текстовый	11

Таблица 3.9 - Описание таблицы Priority

Идентификатор	Описание	Тип данных	Размер
ID	ID приоритета	Числовой	Целый
Descriptions	Описание	Текстовый	255
Name	Итоговая цена оплаты	Текстовый	255

Таблица 3.10 - Описание таблицы Request\_Monitoring

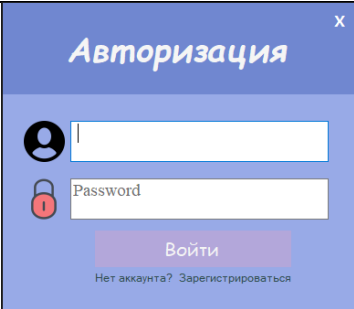
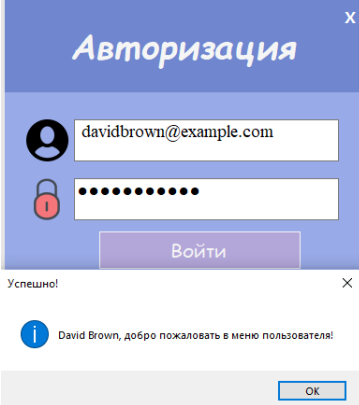
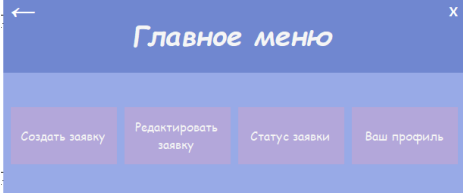
Идентификатор	Описание	Тип данных	Размер
ID	ID мониторинга заявок	Числовой	Целый
Time_receipt	Время регистрации заявки	Время	
Time_acceptance	Время регистрирования заявки	Время	
Processing_time	Время обработки	Числовой	Целый
Registered	Зарегистрирована ли заявка	Бинарный	Переменный
Priority_ID	ID приоритета	Числовой	Целый
Admin_ID	ID администратора	Числовой	Целый
Request_ID	ID заявки	Числовой	Целый

### 3.5 Тестирование программы

Проведем тестирование авторизации в программе.

В таблице 3.11 представлен пример позитивного тест-кейса авторизации

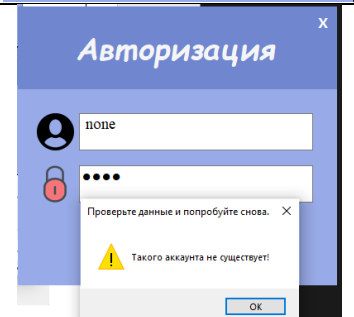
Таблица 3.11 – Пример позитивного тест-кейса авторизации

Действие	Ожидаемый результат	Результат теста
1. Запустить программу	<ul style="list-style-type: none"> <li>– программа запущена</li> <li>– открыта форма авторизации</li> </ul>	
2. Заполнить поля формы: Логин = davidbrow@example.com Пароль = Password345	<ul style="list-style-type: none"> <li>– поля ввода заполнены</li> <li>– пароль скрыт системными символами</li> </ul>	
3. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> <li>– авторизация проходит проверку</li> <li>– открывается главная форма</li> </ul>	

Далее рассмотрим негативный тест-кейс.

В таблице 3.12 приведен пример негативного тест-кейса

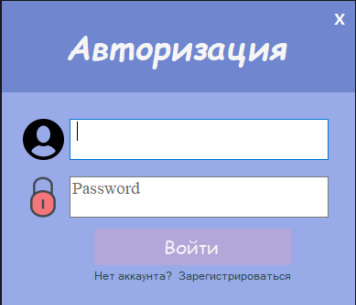
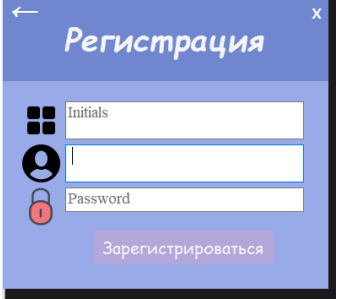
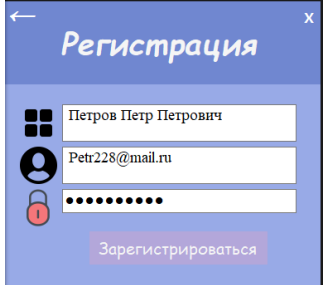
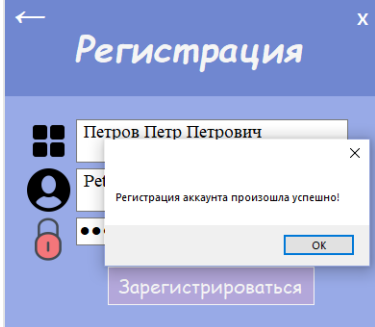
Таблица 3.12 – Пример негативного тест-кейса авторизации

Действие	Ожидаемый результат	Результат теста
1. Запустить программу	<ul style="list-style-type: none"> <li>– программа запущена</li> <li>– открыта форма авторизации</li> </ul>	
2. Заполнить поля формы: Логин = none Пароль = none	<ul style="list-style-type: none"> <li>– поля ввода заполнены</li> <li>– пароль скрыт системными символами</li> </ul>	
3. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> <li>– авторизация не проходит проверку</li> <li>– появляется сообщение с ошибкой</li> </ul>	

Проведем тестирование регистрации пациента в системе.

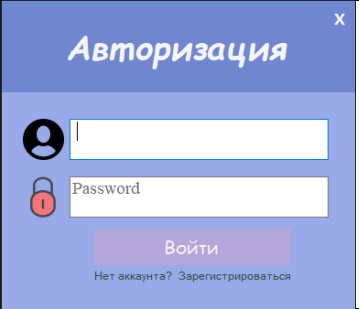
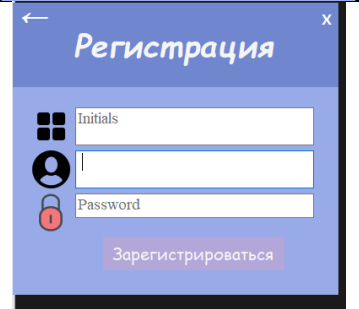
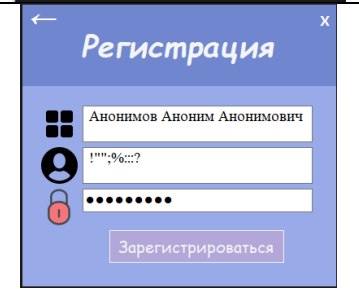
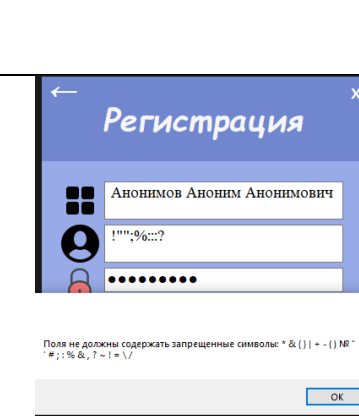
В таблице 3.13 представлен пример позитивного тест-кейса регистрации

Таблица 3.13 – Пример позитивного тест-кейса регистрации

Действие	Ожидаемый результат	Результат теста
1. Запустить программу	<ul style="list-style-type: none"> <li>– программа запущена</li> <li>– открыта форма авторизации</li> </ul>	
2. Нажать на ссылку «Нет аккаунта? Зарегистрироваться»	<ul style="list-style-type: none"> <li>– открывается форма регистрации</li> </ul>	
3. Заполнить поля формы: Логин = Petr228@mail.ru Пароль = Petr228POP ФИО = Петров Петр Петрович	<ul style="list-style-type: none"> <li>– все поля заполнены</li> <li>– пароль скрыт системными символами</li> </ul>	
4. Нажать на кнопку «Зарегистрироваться»	<ul style="list-style-type: none"> <li>– регистрация проходит успешно</li> <li>– появляется сообщение «Вы зарегистрированы»</li> <li>– форма регистрации закрывается</li> </ul>	

В таблице 3.14 приведен пример негативного тест-кейса регистрации

Таблица 3.14 – Пример негативного тест-кейса регистрации

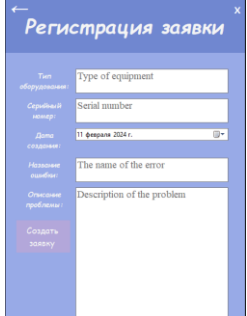
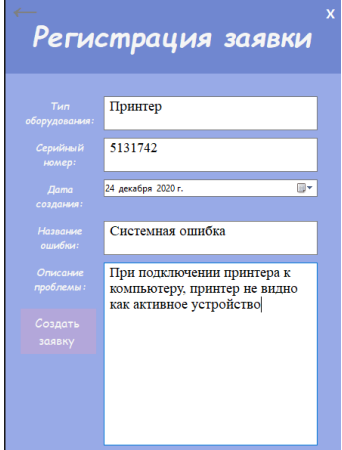
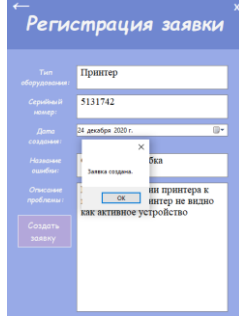
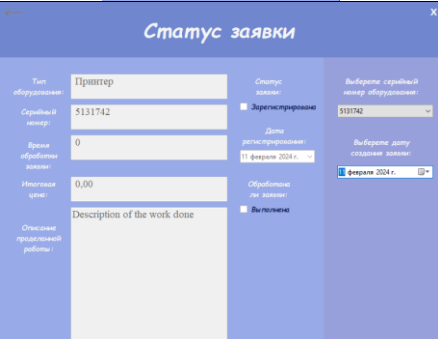
Действие	Ожидаемый результат	Результат теста
1. Запустить программу	<ul style="list-style-type: none"> <li>– программа запущена</li> <li>– открыта форма авторизации</li> </ul>	
2. Нажать на ссылку «Нет аккаунта? Зарегистрироваться»	<ul style="list-style-type: none"> <li>– открывается форма регистрации пациента</li> </ul>	
3. Заполнить поля формы: Логин = !"";%::? Пароль = !"";%::? ФИО = Анонимов Аноним Аноимович	<ul style="list-style-type: none"> <li>– все поля заполнены</li> <li>– пароль скрыт системными символами</li> </ul>	
4. Нажать на кнопку «Зарегистрироваться»	<ul style="list-style-type: none"> <li>– регистрация не проходит успешно</li> <li>– появляется сообщение «Поля не должны содержать запрещенные символы ...»</li> </ul>	



Проведем тестирование формы регистрации заявки.


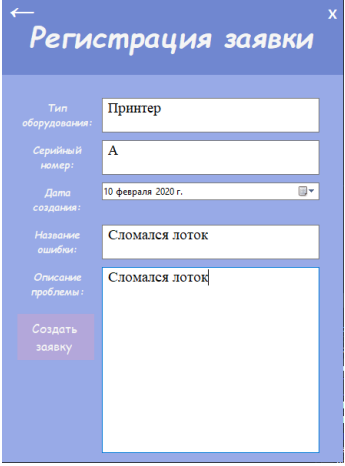
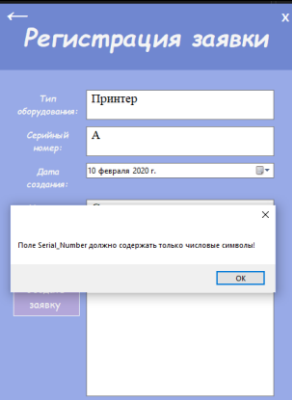
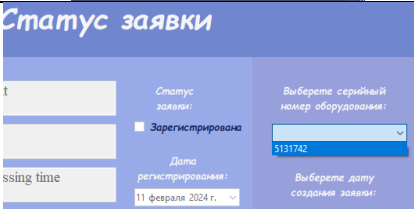
В таблице 3.15 представлен позитивный тест-кейс регистрации заявки.

Таблица 3.15 – Пример позитивного тест-кейса регистрации

Действие	Ожидаемый результат	Результат теста
1. Открыть форму регистрации заявки	<ul style="list-style-type: none"> <li>— форма записи открыта</li> <li>— поля не заполнены</li> </ul>	
2. Заполнить поля: Тип оборудования = Принтер Серийный номер = 5131742 Дата создания = 24 декабря 2020 Название ошибки = Системная ошибка Описание проблемы = При подключении ..... .....	<ul style="list-style-type: none"> <li>— поля формы заполнены</li> </ul>	
3. Нажать кнопку «Создать заявку»	<ul style="list-style-type: none"> <li>— появляется сообщение «Запись добавлена»</li> </ul>	
4. Просмотреть статус заявки	<ul style="list-style-type: none"> <li>— Заявка добавлена в систему</li> </ul>	

В таблице 3.16 представлен негативный тест-кейс записи на прием

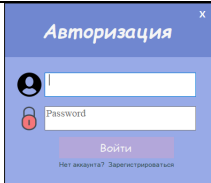
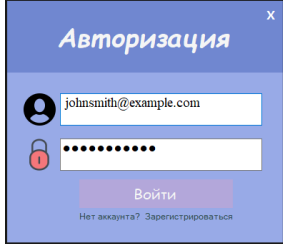
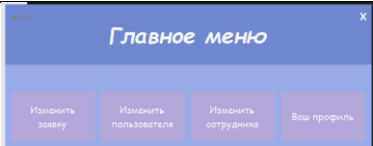
Таблица 3.16 – Пример негативного тест-кейса записи

Действие	Ожидаемый результат	Результат теста
1. Открыть форму записи	<ul style="list-style-type: none"> <li>– форма записи открыта</li> <li>– поля не заполнены</li> </ul>	
1. Заполнить поля: Тип оборудования = Принтер Серийный номер = А Дата создания = 10 февраля 2020 Название ошибки = Сломался лоток Описание проблемы = Сломался лоток	<ul style="list-style-type: none"> <li>– все поля, заполнены, но в серийном номере указана буква</li> </ul>	
2. Нажать кнопку «Создать заявку»	<ul style="list-style-type: none"> <li>– появляется сообщение «Поле серийный номер должно содержать только числовые символы»</li> </ul>	
3. Просмотреть запись	<ul style="list-style-type: none"> <li>– Заявка не добавлена (не прошла регистрацию)</li> </ul>	

Проведем тестирование работоспособности администратора.

В таблице 3.17 представлен позитивный тест-кейс удаления клиента

Таблица 3.17 – Пример позитивного тест-кейса удаления клиента

Действие	Ожидаемый результат	Результат теста
1. Запустить приложение	– форма авторизации открыта	
2. Заполнить поля формы: Логин = johnsmith@example.com Пароль = password123	– поля формы заполнены	
3. Нажать кнопку «Войти»	– появляется форма админ-панели	
4. Нажать кнопку «изменить пользователя» далее выберем ФИО «David Brown» и нажмем кнопку удалить	– форма обновлена – учетная запись клиента удалена	

## 3.6 Руководство пользователя

### Назначение программы

Программа предназначена для предоставления клиентам системы интерфейса для создания заявки неполадок КС. Клиент, авторизовавшись в системе, имеет возможность оставить заявку, просмотреть ее статус, посмотреть свои существующие записи, а также изменить свою учетную запись. Исполнитель, авторизовавшись в системе, имеет возможность добавлять данные о работе с заявкой. Администратор, авторизовавшись в системе, контролирует работу системы, принимает и регистрирует поступающие заявки, он может изменять и удалять учетные записи клиентов и исполнителей, а также может редактировать и удалять заявки.

### Условия выполнения программы

#### Минимальные характеристики ПК:

1. Процессор intel core i3 с частотой 2.3 GHz
2. Объем оперативной памяти – 4 GB
3. Объем свободного места на жестком диске компьютера – 900 MB
4. Платформа 32-разрядная.

#### Рекомендуемые характеристики ПК:

1. Процессор intel core i5 с частотой 3.0 GHz
2. Объем оперативной памяти – 8 GB
3. Объем свободного места на жестком диске компьютера – 3.0 GB
4. Платформа 64-разрядная

### Входные данные

Авторизация в системе происходит при помощи ввода логина и пароля. Пароль хранится в базе данных в хешированном виде. Хеширование происходит с помощью хеш-функции SHA-256. Структура заявки состоит из данных заявки, исполнителя, клиента, даты и времени зарегистрировано заявки и итоговой цены.

## Выполнение программы

Программа запускается исполняемым файлом WinFormsApp1.exe. Завершить программу можно, нажав кнопку, где нарисован крестик на навигационной форме. При входе в роли клиента, пользователю предоставляется навигационная форма, форма регистрации заявки, форма редактирования заявки, статус заявки и профиль. На рисунках 3.6-3.10 представлены доступные пациенту формы

На рисунке 3.6 представлена навигационная форма.

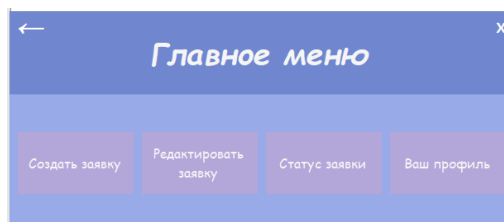


Рисунок 3.6 – Навигационная форма

На рисунке 3.7 представлена форма регистрации заявки.

Рисунок 3.7 – Форма регистрации заявки

На рисунке 3.8 представлена форма редактирования заявки.

Рисунок 3.8 – Форма редактирования заявки

На рисунке 3.9 представлена форма статуса заявки.

Рисунок 3.9 – Форма статуса заявки

На рисунке 3.10 представлена форма профиля.

Рисунок 3.10 – Форма профиля

При входе как исполнитель, пользователю доступна навигационная форма, форма списка всех заявок, форма статуса заявки, форма статистика и профиль. На рисунке 3.11-3.15 представлены формы исполнителя.

На рисунке 3.11 представлена навигационная форма.

Рисунок 3.11 – Навигационная форма

На рисунке 3.12 представлена форма списка всех заявок.



Begin_Date	Registered	Name	Equipment_Type	Serial_Number	Creation_Date	Problem_Descr	Name1
10.03.2021	<input checked="" type="checkbox"/>	High	Server	54321678	10.03.2021	Slow response t...	Slow per...
10.01.2021	<input checked="" type="checkbox"/>	Medium	Desktop MAMA	12345678	01.01.2021	Computer cras...	Blue sce...
10.01.2021	<input checked="" type="checkbox"/>	Medium	Desktop MAMA	12345678	01.01.2021	Computer cras...	Blue sce...

Рисунок 3.12 – Форма списка всех заявок

На рисунке 3.13 показана форма статуса заявки.

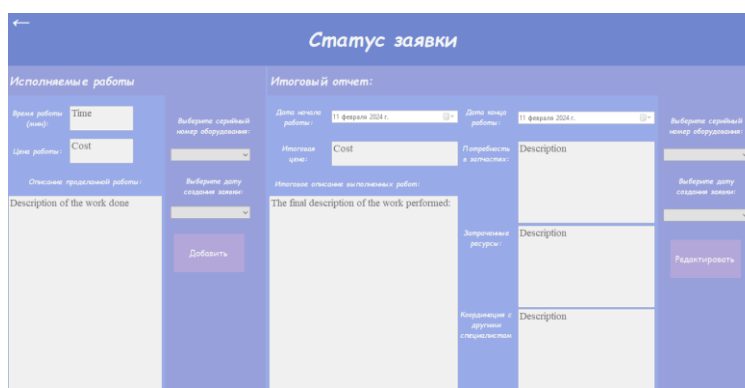


Рисунок 3.13 – Форма статуса заявки

На рисунке 3.14 показана форма статистики.

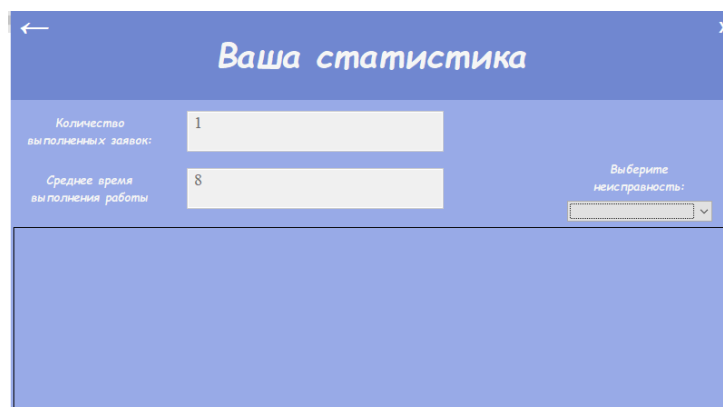


Рисунок 3.14 – Форма статистики

На рисунке 3.15 показана форма профиля.

Рисунок 3.15 – Форма профиля

При входе как администратор, пользователю дается возможность просматривать учетные записи клиентов, исполнителей фильтровать их по ФИО, возможность удалять и изменять данные. Администратору доступна форма изменения заявки, изменения профиля клиента и сотрудника, а также свой профиль. На рисунке 3.16-3.20 представлены формы администратора.

На рисунке 3.16 представлена навигационная форма.

Рисунок 3.16 – Навигационная форма

На рисунке 3.17 представлена форма изменения заявки.

Рисунок 3.17 – Форма изменения заявки



На рисунке 3.18 представлена форма изменения профиля клиента.

Рисунок 3.18 – Форма изменения профиля клиента

На рисунке 3.19 представлена форма изменения профиля исполнителя.

Рисунок 3.19 – Форма изменения профиля исполнителя

На рисунке 3.20 представлена форма профиля.

Рисунок 3.20 – Форма профиля

## Сообщения

В процессе выполнения программы пользователю выдаются сообщения, уведомляющие о каком-либо действии. На форме регистрации клиента пользователю выдается сообщение о правилах пароля, который нужно ввести.

На рисунке 3.21 представлен вывод сообщения при открытии формы регистрации и вводе неправильного пароля

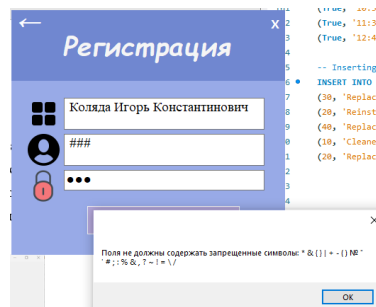


Рисунок 3.21 – Вывод сообщения

В процессе создания редактирования статуса заявки исполнитель может выбрать прошедшую дату, в случае чего ему представится сообщение о вводе неправильной даты.

На рисунке 3.22 представлен вывод сообщения при выборе прошедшей даты.

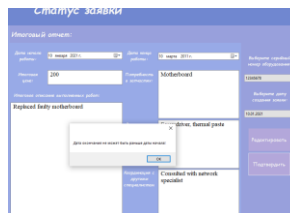


Рисунок 3.22 – Вывод сообщения

## Выходные данные

В процессе выполнения программы формируется отчет в БД по заявке.

На рисунке 3.23 представлена таблица с итоговым отчетом.

Итоговый отчет						
Дата начала работ	Дата-конца работ	Описание проделанной работы	Затраченные ресурсы	Потребность в запчастях	Итоговая цена	Координация с другими специалистами
10.01.2021 0:00:00	15.01.2021 0:00:00	Replaced faulty motherboard	Screwdriver, thermal paste	Motherboard	200,00	Consulted with network specialist

Рисунок 3.23 – Таблица с итоговым отчетом

## ЗАКЛЮЧЕНИЕ

В ходе работы было создано приложение для автоматизации деятельности обслуживания компьютерных средств, заключающейся в регистрации заявок неисправностей, просмотре статуса выполнения записей и редактирования при необходимости со стороны исполнителей заявки, все действия позволяют полноценно работать с базой данных по заданной предметной области и это облегчает процесс создания и обработки заявок. Были выполнены следующие задачи:

- описана предметная область;
- спроектирована система с помощью различных диаграмм;
- описаны функциональные требования к системе (спецификация)
- были созданы 3 пользователя: клиент, исполнитель, и администратор;
- была создана схема данных;
- была создана структура программы;
- было проведено тестирование и отладка программы;
- было создано руководство пользователя;
- была обеспечена безопасность приложения путем хеширования паролей с помощью хеш-функции SHA-256.

Таким образом поставленная цель достигнута, задачи выполнены.

## СПИСОК ЛИТЕРАТУРЫ

- 1) Гражданский кодекс Российской Федерации, Закон РФ от 07.02.1992 г. №2300-І «О защите прав потребителей».
- 2) Кудрина, Е. В. Основы алгоритмизации и программирования на языке C# : учебное пособие для среднего профессионального образования / Е. В. Кудрина, М. В. Огнева. — Москва: Издательство Юрайт, 2023. (дата обращения: 03.02.2024).
- 3) Казанский, А. А. Программирование на Visual C# : учебное пособие для вузов / А. А. Казанский. —Москва : Издательство Юрайт, 2023г. (дата обращения: 03.02.2024).
- 4) Мигель Гринберг. Flask Web Development. – М.: Юрайт, 2018.
- 5) Информационная система «Read the Docs». URL: <https://readthedocs.org>
- 6) Робсон Э. "Изучаем HTML, XHTML и CSS".
- 7) Паленов, В. Н. Базы данных / В.Н. Паленов, Д.Э. Фуфаев. -М.:Академия, 2019 год - 189 с
- 8) Лутц, Марк. Л86 Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019
- 9) "Mastering CSS: Advanced Techniques for Modern Web Design" 2019 года
- 10) Чистый Python. Тонкости программирования для профи, Бейдер Дэн
- 11) Официальная документация по языку C#. URL - <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
- 12) Официальная документация по Windows Forms. URL - <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-8.0>
- 13) Информационная система «Github» URL - <https://github.com>

## **ПРИЛОЖЕНИЯ**

**ПРИЛОЖЕНИЕ А**  
**(обязательное)**  
**Листинг программы**

## А.1 Листинг программы

```
// УП по ПМ 0.2 - Осуществление интеграции программных модулей
// по теме: Учет неисправностей компьютерных систем
// Разработал: Владимиров Иван Сергеевич
// Группа: ТИП-62
// Дата и номер версии: 30.02.2024 v1.0
// Язык: C#
// Краткое описание: программа предназначена для записей неисправностей КС
//
// Задание:
// 1) Создать главную форму приложения
// 2) Создать формы для авторизации и регистрации пользователей системы
// 3) Создать пользовательские формы, соответствующие таблицам
// базы данных, с возможностью просмотра, добавления, удаления,
// обновления, поиска, фильтрации записей в таблицах базы данных
// 4) Создать формы для выполнения вычисляемой функции
//
// Используемые формы:
// AdminMenuForm, UsersMenuForm, EmployeeMenuForm - Навигационные
форма
// AdminsProfileForm, EmployeeProfileForm, UsersProfileForm - Формы профиля
// UsersProfileAdminForm, EmployeeProfileAdminForm,
EditApplicationAdminForm,
// EditApplicationForm, - Формы редактирования
// ApplicationForm- Форма создания заявки
// ApplicationListForm - Форма списка всех заявок
// LoginForm - Форма авторизации
// RegisterForm - Форма регистрации
// StaticticsForm - Форма статистики
```

```

// StatusApplicationForm, StatusApplicationEmployeeForm - Форма статуса заявки
//
// Используемые граф. элементы:
// Form – Форма
// Label - Текстовый элемент
// Button - кнопка
// TextBox - Поле ввода
// ComboBox - Выпадающий список
// DateTimePicker - Поле выбора даты и времени
// Panel - Группирующий элемент
// DataGridView - таблица
//
// Данный файл является точкой старта приложения
// Функции:
// Initialize - Инициализация конфигурации приложения
// Run(form: Form) - Запуск приложения
namespace WinFormsApp1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high DPI settings or
default font,
            // see https://aka.ms/applicationconfiguration.
            Application.EnableVisualStyles();

```



```
Application.SetCompatibleTextRenderingDefault(false);
    ApplicationConfiguration.Initialize();
    Application.Run(new LoginForm());
}
}
}
```

## A.2 Листинг файла Authorization.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
namespace WinFormsApp1
{
    internal class Authorization
    {
        static public string Role, FIO, User;
        static public void Authorization1(string login, string password)
        {
            try
            {
                using (MySqlConnection connection = new MySqlConnection("user=root;
host=localhost; password=IvanVladimirov888; database=Computer_systems;"))
                {
                    connection.Open();
                    // Массив запросов
                    string[] queries1 = new string[]
                    {
                        @"SELECT Name FROM Roles INNER JOIN Clients on Roles.ID =
Clients.ID_Roles WHERE Email = @Email AND Password = @Password",
                        @"SELECT Name FROM Roles INNER JOIN Executors on Roles.ID =
Executors.ID_Roles WHERE Email = @Email AND Password = @Password",
```

```

        @"SELECT Name FROM Roles INNER JOIN Admins on Roles.ID =
Admins.ID_Roles WHERE Email = @Email AND Password = @Password"
    };
    string[] queries2 = new string[]
    {
        @"SELECT Name FROM Roles INNER JOIN Clients on Roles.ID =
Clients.ID_Roles WHERE Email = @Email AND Password = @hashedPassword",
        @"SELECT Name FROM Roles INNER JOIN Executors on Roles.ID =
Executors.ID_Roles WHERE Email = @Email AND Password = @hashedPassword",
        @"SELECT Name FROM Roles INNER JOIN Admins on Roles.ID =
Admins.ID_Roles WHERE Email = @Email AND Password = @hashedPassword"
    };
    string hashedPassword = HashPassword(password);
    bool foundResult = false;
    if (!foundResult)
    {
        for (int i = 0; i < queries1.Length; i++)
        {
            string query = queries1[i];
            MySqlCommand command = new MySqlCommand(query,
connection);

            command.Parameters.AddWithValue("@Email", login);
            command.Parameters.AddWithValue("@Password", password);
            Object result = command.ExecuteScalar();
            if (result != null)
            {
                Role = result.ToString();
                User = login;
                foundResult = true; // Устанавливаем флаг, что результат
получен

```

```

        break; // Выходим из цикла, если результат получен
    }
    else
    {
        Role = null;
        FIO = null;
    }
}
}
if (!foundResult)
{
    for (int i = 0; i < queries2.Length; i++)
    {
        string query = queries2[i];
        MySqlCommand command = new MySqlCommand(query,
connection);

        command.Parameters.AddWithValue("@Email", login);
        command.Parameters.AddWithValue("@HashedPassword",
hashedPassword);

        Object result = command.ExecuteScalar();
        if (result != null)
        {
            Role = result.ToString();
            User = login;
            foundResult = true; // Устанавливаем флаг, что результат
получен

            break; // Выходим из цикла, если результат получен
        }
        else
        {

```

```

        Role = null;
        FIO = null;
    }
}
}
connection.Close();
}
}
catch
{
    Role = User = null;
    MessageBox.Show("Ошибка при авторизации!");
}
}
private static string HashPassword(string password)
{
    using (SHA256 sha256 = SHA256.Create())
    {
        byte[] bytes = Encoding.UTF8.GetBytes(password);
        byte[] hash = sha256.ComputeHash(bytes);
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < hash.Length; i++)
        {
            result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX
        }
        return result.ToString();
    }
}

```

строки

```

static public string AuthorizationName(string login)
{
    try
    {
        string FIO = null; // Переменная для хранения найденной фамилии
        using (MySQLConnection connection = new MySQLConnection("user=root;
host=localhost; password=IvanVladimirov888; database=Computer_systems;"))
        {
            connection.Open();
            string[] queries = new string[]
            {
                @"SELECT FIO FROM Clients WHERE Email = @Email",
                @"SELECT FIO FROM Executors WHERE Email = @Email",
                @"SELECT FIO FROM Admins WHERE Email = @Email"
            };
            for (int i = 0; i < queries.Length; i++)
            {
                using (MySQLCommand command = new MySQLCommand(queries[i],
connection))
                {
                    command.Parameters.AddWithValue("@Email", login);
                    Object result = command.ExecuteScalar();
                    if (result != null)
                    {
                        FIO = result.ToString();
                        break; // Завершаем цикл, если совпадение найдено
                    }
                }
            }
        }
    }
}

```

```
        return FIO; // Возвращаем фамилию или null, если ничего не найдено
    }
    catch
    {
        return null;
    }
}
}
```

### A.3 Листинг файла DB.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace WinFormsApp1
{
    internal class DB
    {
        MySqlConnection connection = new MySqlConnection("user=root;
host=localhost; password=IvanVladimirov888; database=Computer_systems;");
        public void openConnection()
        {
            if (connection.State == System.Data.ConnectionState.Closed)
                connection.Open();
        }
        public void closeConnection()
        {
            if (connection.State == System.Data.ConnectionState.Open)
                connection.Close();
        }
        public MySqlConnection getConnection()
        {
            return connection;
        }
        /*static string DBConnect = "user=root; host=localhost;
password=IvanVladimirov888; database=Computer_systems";
        static public MySqlDataAdapter msDataAdapter;
```



```

static MySqlConnection myConnect;
static public MySqlCommand msCommand;
public static bool ConnectionDB()
{
    try
    {
        myConnect = new MySqlConnection(DBConnect);
        myConnect.Open();
        msCommand = new MySqlCommand();
        msCommand.Connection = myConnect;
        msDataAdapter = new MySqlDataAdapter(msCommand);
        return true;
    }
    catch
    {
        MessageBox.Show("Ошибка соединения с базой данных!", "Ошибка!",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return false;
}
public static void CloseDB()
{
    myConnect.Close();
}
public MySqlConnection getConnection()
{
    return myConnect;
}*/
}
}

```

#### А.4 ЛИСТИНГ файла AdminMenuForm.cs

```
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;

namespace WinFormsApp1
{
    public partial class AdminMenuForm : Form
    {
        public AdminMenuForm()
        {
            InitializeComponent();
        }

        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Hide();
            LoginForm loginForm = new LoginForm();
            loginForm.Show();
            // Очистим информацию о пользователе
            Authorization.Role = null;
            Authorization.User = null;
        }
    }
}
```

```

        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }

    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Red;
    }

    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }

    private void ToLoginLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        LoginForm loginForm = new LoginForm();
        loginForm.Show();
        // Очистим информацию о пользователе
        Authorization.Role = null;
        Authorization.User = null;
        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }

    private void ToLoginLabel_MouseEnter(object sender, EventArgs e)
    {

```

```

        ToLoginLabel.ForeColor = Color.Gray;
    }

    private void ToLoginLabel_MouseLeave(object sender, EventArgs e)
    {
        ToLoginLabel.ForeColor = Color.White;
    }

    /// <summary>
    /// Передвижение окна.
    /// </summary>
    Point lastPoint;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }
    /// <summary>
    /// Запоминание позиции курсора.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }

```

```

private void CreateApplication_Click(object sender, EventArgs e)
{
    this.Hide();
    EditApplicationAdminForm editApplicationAdmin = new
EditApplicationAdminForm();
    editApplicationAdmin.Show();
}
private void UsersProfile_Click(object sender, EventArgs e)
{
    this.Hide();
    UsersProfileAdminForm usersProfileAdmin = new
UsersProfileAdminForm();
    usersProfileAdmin.Show();
}
private void EmployeeProfile_Click(object sender, EventArgs e)
{
    this.Hide();
    EmployeeProfileAdminForm employeeProfileAdmin = new
EmployeeProfileAdminForm();
    employeeProfileAdmin.Show();
}
private void AdminsProfile_Click(object sender, EventArgs e)
{
    this.Hide();
    AdminsProfileForm adminsProfile = new AdminsProfileForm();
    adminsProfile.Show();
}
}
}

```

## A.5 Листинг файла AdminsProfileForm.cs

```
using Microsoft.VisualBasic.Logging;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
namespace WinFormsApp1
{
    public partial class AdminsProfileForm : Form
    {
        public AdminsProfileForm()
        {
            InitializeComponent();
        }

        private void EditProfile_Click(object sender, EventArgs e)
        {
            // Сделать все текстовые поля доступными для редактирования
            textBox1.Enabled = true;
            textBox2.Enabled = true;
            textBox3.Enabled = true;
        }
    }
}
```

```

        textBox4.Enabled = true;
        ConfirmationButton.Visible = true;
    }

    private string HashPassword(string password)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(password);
            byte[] hash = sha256.ComputeHash(bytes);
            StringBuilder result = new StringBuilder();

            for (int i = 0; i < hash.Length; i++)
            {
                result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX
            }

            return result.ToString();
        }
    }
}

```

строки

```

// Обработчик кнопки ConfirmationButton
private void ConfirmationButton_Click(object sender, EventArgs e)
{
    // Проверка введенных данных и их сохранение в базе
    string fio = textBox1.Text;
    string phone = textBox2.Text;
}

```

```

// Дополнительно получаем введенные логин и пароль
string login = textBox3.Text;
string password = HashPassword(textBox4.Text);
string password1 = textBox4.Text;
// Проверка формата введенных данных
if (!IsValidFIO(fio))
{
    MessageBox.Show("Фамилия должна содержать только буквы");
    return;
}
if (!IsValidPhone(phone))
{
    MessageBox.Show("Номер телефона должен содержать ровно 11
цифр");
    return;
}
if (!IsValidPassword(password1))
{
    MessageBox.Show("Пароль должен содержать от 4 до 16 символов");
    return;
}
// Проверка на использование определенных символов
string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№",
"\", "'", "#", ";", ":",
"%", "&", ",", "?", "~", "!", "=", "\\", "/" };
if (restrictedSymbols.Any(symbol => textBox3.Text.Contains(symbol)))
{
    MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
    return; // Прерываем выполнение метода

```



```

    }
    if (restrictedSymbols.Any(symbol => textBox4.Text.Contains(symbol)))
    {
        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода
    }
    // Получение введенных логина и пароля
    string newLogin = textBox3.Text;
    string newPassword = HashPassword(textBox4.Text);
    string oldPasswordFromDB;
    // Получение старого логина из глобальной переменной или другого
    места, где он хранится
    string oldLogin = LoginForm.loginActive; // Примерное название
    переменной, где хранится текущий логин
    // Получение старого пароля из базы данных по
    старому логину
    string oldPasswordFromDBQuery = $"SELECT Password FROM Admins
Where Email = '{oldLogin}'";
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        // Открываем соединение
        connection.Open();
        using (MySqlCommand oldPasswordFromDBCommand = new
MySqlCommand(oldPasswordFromDBQuery, connection))
        {
            oldPasswordFromDB =
(string)oldPasswordFromDBCommand.ExecuteScalar();
        }
    }

```

```

        // Закрываем соединение
        connection.Close();
    }
    if (newLogin != LoginForm.loginActive || newPassword !=
oldPasswordFromDB)
    {
        // Выводим окно с подтверждением изменения логина и пароля
        DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите
изменить логин и пароль?", "Подтверждение изменений", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            // Обновляем логин и пароль в базе данных
            string updateLoginQuery = $"UPDATE Admins SET Email =
'{newLogin}' WHERE Email = '{oldLogin}'";
            string updatePasswordQuery = $"UPDATE Admins SET Password =
'{newPassword}' WHERE Email = '{newLogin}'";
            using (MySQLConnection connection = db.getConnection())
            {
                // Открываем соединение
                connection.Open();
                using (MySQLCommand updateLoginCommand = new
MySQLCommand(updateLoginQuery, connection))
                using (MySQLCommand updatePasswordCommand = new
MySQLCommand(updatePasswordQuery, connection))
                {
                    updateLoginCommand.ExecuteNonQuery();
                    updatePasswordCommand.ExecuteNonQuery();
                }
            }
            // Закрываем соединение
            connection.Close();
        }
    }
}

```

```

    }
    // Обновление данных в базе данных
    string updateQuery = $"UPDATE Admins SET FIO = '{fio}', Phone =
    '{phone}' WHERE Email = '{newLogin}';";
    using (MySQLConnection connection = db.getConnection())
    {
        // Открываем соединение
        connection.Open();
        using (MySQLCommand updateQueryCommand = new
        MySQLCommand(updateQuery, connection))
        {
            updateQueryCommand.ExecuteNonQuery();
        }
        // Закрываем соединение
        connection.Close();
    }
    // Сделать текстовые поля недоступными для редактирования
    textBox1.Enabled = false;
    textBox2.Enabled = false;
    textBox3.Enabled = false;
    textBox4.Enabled = false;
    ConfirmationButton.Visible = false;
}
else
{
    // Блокируем поля логина и пароля и оставляем старые данные
    textBox3.Text = oldLogin;
    textBox4.Text = oldPasswordFromDB; // Допустим, что мы не
    обновляем пароль, если изменение не подтверждено
    // Обновление данных в базе данных

```

```

        string updateQuery = $"UPDATE Admins SET FIO = '{fio}', Phone =
        '{phone}' WHERE Email = '{oldLogin}';";
        using (SqlConnection connection = db.getConnection())
        {
            connection.Open();
            using (SqlCommand updateQueryCommand = new
            MySqlCommand(updateQuery, connection))
            {
                updateQueryCommand.ExecuteNonQuery();
            }
            connection.Close();
        }
        // Сделать текстовые поля недоступными для редактирования
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        textBox3.Enabled = false;
        textBox4.Enabled = false;
        ConfirmationButton.Visible = false;
    }

}
textBox4.Text = "*****";
textBox4.PasswordChar = '*';
textBox4.UseSystemPasswordChar = true;
}

// Метод для проверки формата ФИО
private bool IsValidFIO(string fio)
{
    // Проверяем, что строка состоит только из букв и пробелов

```

```

        return !string.IsNullOrEmpty(fio) && fio.All(c => char.IsLetter(c) ||
char.IsWhiteSpace(c));
    }

    // Метод для проверки формата номера телефона
    private bool IsValidPhone(string phone)
    {
        // Проверяем, что строка состоит из 11 цифр
        return !string.IsNullOrEmpty(phone) && phone.Length == 11 &&
phone.All(c => char.IsDigit(c));
    }

    private bool IsValidPassword(string password)
    {
        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(password) && password.Length == 16 &&
password.Length > 4;
    }

    private void AdminsProfileForm_Load(object sender, EventArgs e)
    {
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();

            string oldLogin = LoginForm.loginActive;// Примерное название
переменной, где хранится текущий логин

            // Подготавливаем SQL-запрос
            string selectQuery = "SELECT FIO, Phone, Password, Email FROM
Admins WHERE Email = @Email";

```

```

MySQLCommand command = new MySQLCommand(selectQuery,
connection);

command.Parameters.AddWithValue("@Email", oldLogin);

// Используем читатели для получения данных из базы
using (MySQLDataReader reader = command.ExecuteReader())
{
    if (reader.Read())
    {
        // Заполняем поля на форме полученными данными
        textBox1.Text = reader["FIO"].ToString();
        textBox2.Text = reader["Phone"].ToString();
        textBox3.Text = reader["Email"].ToString();
        // В данном случае пароль получаем в открытом виде
        textBox4.Text = "*****";
    }
}

// Закрываем соединение
db.closeConnection();
}
}

Point lastPoint;
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
}

```

```

/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}

private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
}

private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}

private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}

```

```

    }

    private void ToAdminsMenuLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        AdminMenuForm adminMenu = new AdminMenuForm();
        adminMenu.Show();
    }

    private void ToAdminsMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToAdminsMenuLabel.ForeColor = Color.Gray;
    }

    private void ToAdminsMenuLabel_MouseLeave(object sender, EventArgs e)
    {
        ToAdminsMenuLabel.ForeColor = Color.White;
    }
}

```



## A.6 Листинг файла ApplicationForm.cs

```
using MySql.Data.MySqlClient;
using MySqlX.XDevAPI;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.ComponentModel.Design.ObjectSelectorEditor;

namespace WinFormsApp1
{
    public partial class ApplicationForm : Form
    {
        public ApplicationForm()
        {
            InitializeComponent();
        }

        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Hide();
            LoginForm loginForm = new LoginForm();
            loginForm.Show();
        }
    }
}
```

```

// Очистим информацию о пользователе
Authorization.Role = null;
Authorization.User = null;
Authorization.FIO = null;
LoginForm.loginActive = "";
RegisterForm.loginActive = "";
}

private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}

private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}

private void ToUsersMenuLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    UsersMenuForm usersMenu = new UsersMenuForm();
    usersMenu.Show();
}

private void ToUsersMenuLabel_MouseEnter(object sender, EventArgs e)
{
    ToUsersMenuLabel.ForeColor = Color.Gray;
}

private void ToUsersMenuLabel_MouseLeave(object sender, EventArgs e)
{

```

```

        ToUsersMenuLabel.ForeColor = Color.White;
    }

    private void CreateApplication_Click(object sender, EventArgs e)
    {
        if (textBox1.Text != "" && textBox2.Text != "" && dateTimePicker1.Value
            != null && textBox3.Text != "" && textBox4.Text != "")
        {
            // Проверка на ввод числовых данных в поле Serial_Number
            if (!Regex.IsMatch(textBox2.Text, @"^\d+$"))
            {
                MessageBox.Show("Поле Serial_Number должно содержать только
числовые символы!");
                return; // Прерываем выполнение метода
            }

            // Проверка на использование определенных символов
            string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "@", "_", "(",
            ")", "№", "\", "", "#", ";", ":", "%", "&", ".", ",", "?", "~", "!", "=", "\\", "/" };
            if (restrictedSymbols.Any(symbol => textBox1.Text.Contains(symbol) ||
            textBox2.Text.Contains(symbol) || textBox3.Text.Contains(symbol)))
            {
                MessageBox.Show("Поля не должны содержать запрещенные
символы: " + string.Join(" ", restrictedSymbols));
                return; // Прерываем выполнение метода
            }
            // Все поля заполнены

            // Получаем подключение к базе данных
            DB db = new DB();

```

```

using (SqlConnection connection = db.getConnection())
{
    //Сегодняшняя дата
    DateTime today = DateTime.Today;
    string formDate = today.ToString("yyyy-MM-dd");
    //Время, которое сейчас
    DateTime currentTime = DateTime.Now;
    string formattedTime = currentTime.ToString("HH:mm:ss");

    db.openConnection();
    // Создаем команду для выполнения запроса INSERT
    MySqlCommand command1 = new MySqlCommand("INSERT INTO
Errors (Name) VALUES (@Name)", connection);

    MySqlCommand command2 = new MySqlCommand("INSERT INTO
Requests (Equipment_Type, Serial_Number, Creation_Date, Problem_Description,
Errors_ID) VALUES (@Equipment_Type, @Serial_Number, @Creation_Date,
@Problem_Description, @Errors_ID)", connection);

    MySqlCommand command3 = new MySqlCommand("INSERT INTO
Repair_Reports (Clients_ID, Request_ID, Executor_ID, Begin_Date, End_Date,
Description_all_work, Used_resources, Spare_parts_needed, Total_cost,
Coordination_with_other_specialists) VALUES ( @ClientID, @Request_ID, 1," + formDate
+ "," + formDate + ", ", ", ", 0, ")", connection);

    MySqlCommand command4 = new MySqlCommand("INSERT INTO
Request_Monitoring (Request_ID, Admin_ID, Registered,Time_receipt, Time_acceptance,
Processing_time) VALUES (@Request_ID, 1, False, " + formattedTime + "," +
formattedTime + ", 0)", connection);

    MySqlCommand command5 = new MySqlCommand("INSERT INTO
Work_done (Work_time, Description_work, Expenses, Request_ID, Executor_ID) VALUES
(0, ", 0.00,@Request_ID,1)", connection);

```

```

        // Получение ID клиента
        MySqlCommand getClientIDCommand = new
        MySqlCommand($"SELECT ID FROM Clients WHERE Email = '{LoginForm.loginActive
        ?? RegisterForm.loginActive}'", connection);

        int clientID =
        (int)Convert.ToInt64(getClientIDCommand.ExecuteScalar());

        command3.Parameters.Add("@ClientID", MySqlDbType.Int64).Value =
        clientID;

        command1.Parameters.Add("@Name", MySqlDbType.VarChar).Value
        = textBox3.Text;

        // Открываем соединение
        command1.ExecuteNonQuery();
        int lastInsertedID1 = (int)command1.LastInsertedId;

        // Задаем параметры для запроса
        command2.Parameters.Add("@Equipment_Type",
        MySqlDbType.VarChar).Value = textBox1.Text;
        command2.Parameters.Add("@Serial_Number",
        MySqlDbType.VarChar).Value = textBox2.Text;
        string formattedDate = dateTimePicker1.Value.ToString("yyyy-MM-
        dd");

        command2.Parameters.Add("@Creation_Date",
        MySqlDbType.Date).Value = formattedDate;
        command2.Parameters.Add("@Problem_Description",
        MySqlDbType.Text).Value = textBox4.Text;
        command2.Parameters.Add("@Errors_ID", MySqlDbType.Int64).Value
        = lastInsertedID1;

        // Выполняем запрос

```

```

        command2.ExecuteNonQuery();

        int lastInsertedID2 = (int)command2.LastInsertedId;
        command3.Parameters.Add("@Request_ID",
MySqlDbType.Int64).Value = lastInsertedID2;
        command3.ExecuteNonQuery();
        command4.Parameters.Add("@Request_ID",
MySqlDbType.Int64).Value = lastInsertedID2;
        command4.ExecuteNonQuery();
        command5.Parameters.Add("@Request_ID",
MySqlDbType.Int64).Value = lastInsertedID2;
        command5.ExecuteNonQuery();
        // Закрываем соединение
        db.closeConnection();
        MessageBox.Show("Заявка создана.");
    }
}
else
{
    // Не все поля заполнены
    MessageBox.Show("Необходимо заполнить все поля!");
}

// По выполнению добавления возвращаемся на прошлую форму
this.Hide();
UsersMenuForm usersMenu = new UsersMenuForm();
usersMenu.Show();
}

/// <summary>

```

```

    /// Передвижение окна.
    /// </summary>
    Point lastPoint;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }
    /// <summary>
    /// Запоминание позиции курсора.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }
}

```

## A.7 Листинг файла ApplicationListForm.cs

```
using MySql.Data.MySqlClient;
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;

namespace WinFormsApp1
{
    public partial class ApplicationListForm : Form
    {
        public ApplicationListForm()
        {
            InitializeComponent();
        }

        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Hide();
            LoginForm loginForm = new LoginForm();
            loginForm.Show();

            // Очистим информацию о пользователе
            Authorization.Role = null;
        }
    }
}
```



```
Authorization.User = null;
Authorization.FIO = null;
LoginForm.loginActive = "";
RegisterForm.loginActive = "";
}
```

```
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}
```

```
private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}
```

```
private void ToEmployeeLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    EmployeeMenuForm employeeMenu = new EmployeeMenuForm();
    employeeMenu.Show();
}
```

```
private void ToEmployeeLabel_MouseEnter(object sender, EventArgs e)
{
    ToEmployeeLabel.ForeColor = Color.Gray;
}
```

```
private void ToEmployeeLabel_MouseLeave(object sender, EventArgs e)
{
```

```

        ToEmployeeLabel.ForeColor = Color.White;
    }

    /// <summary>
    /// Передвижение окна.
    /// </summary>
    Point lastPoint;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }

    /// <summary>
    /// Запоминание позиции курсора.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }

    private void ApplicationListForm_Load(object sender, EventArgs e)
    {
        FillDataGridView();
    }

    private void FillDataGridView()

```

```

    {
        try
        {
            string connectionString = "user=root; host=localhost;
password=IvanVladimirov888; database=Computer_systems"; // Замените на вашу строку
подключения

            using (MySQLConnection connection = new
MySQLConnection(connectionString))
            {
                connection.Open();

                string query = "SELECT Repair_Reports.Begin_Date,
Request_Monitoring.Registered, Priority.Name, Equipment_Type, Serial_Number,
Creation_Date, Problem_Description, Errors.Name " +
                    "FROM Requests " +
                    "INNER JOIN Errors ON Requests.Errors_ID = Errors.ID " +
                    "INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID " +
                    "INNER JOIN Executors ON Executors.ID =
Repair_Reports.Executor_ID " +
                    "INNER JOIN Request_Monitoring ON Requests.ID =
Request_Monitoring.Request_ID " +
                    "INNER JOIN Work_done ON Requests.ID =
Work_done.Request_ID " +
                    "INNER JOIN Priority ON Priority.ID =
Request_Monitoring.Priority_ID " +
                    "WHERE Request_Monitoring.Registered = true " +
                    "ORDER BY Repair_Reports.Begin_Date DESC";

```

```

        MySqlCommand command = new MySqlCommand(query, connection);

        using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))
        {
            DataTable dataTable = new DataTable();
            adapter.Fill(dataTable);

            dataGridView1.DataSource = dataTable;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка данных: " + ex.Message);
    }
}
}

```

## A.8 Листинг файла EditApplicationAdminForm.cs

```
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 MySql.Data.MySqlClient;
using Microsoft.Office.Interop.Word;
using System.Reflection;
using Point = System.Drawing.Point;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;
namespace WinFormsApp1
{
    public partial class EditApplicationAdminForm : Form
    {
        private int selectedClientId; // переменная для хранения выбранного ID
клиента
        public EditApplicationAdminForm()
        {
            InitializeComponent();
        }
        private void CloseButton_MouseEnter(object sender, EventArgs e)
        {
            CloseButton.ForeColor = Color.Red;
        }
    }
}
```

```

    }
    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }
    private void ToUsersMenuLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        AdminMenuForm adminMenu = new AdminMenuForm();
        adminMenu.Show();
    }
    private void ToUsersMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.Gray;
    }
    private void ToUsersMenuLabel_MouseLeave(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.White;
    }
    /// <summary>
    /// Передвижение окна.
    /// </summary>
    Point lastPoint;
    private List<string> serialNumbers;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }

```

```

    }
}
/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panell1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
private void CreateApplication_Click(object sender, EventArgs e)
{
    this.Hide();
    EditApplicationAdminForm editApplicationAdmin = new
EditApplicationAdminForm();
    editApplicationAdmin.Show();
}
private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
    RegisterForm.loginActive = "";
}

```

```

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        // При выборе ФИО загружаем соответствующие серийные номера и
получаем ID клиента
        string selectedFIO = comboBox1.SelectedItem.ToString();
        comboBox3.Items.Clear(); // Очищаем ComboBox перед добавлением
новых значений
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();
            // Выполняем запрос на получение ID клиента по выбранному ФИО
            MySqlCommand command = new MySqlCommand("SELECT ID
FROM Clients WHERE FIO = @selectedFIO", connection);
            command.Parameters.Add("@selectedFIO",
MySqlDbType.VarChar).Value = selectedFIO;
            // Используем ExecuteScalar для получения единственного значения
(ID клиента)
            object result = command.ExecuteScalar();
            if (result != null)
            {
                selectedClientId = Convert.ToInt32(result); // преобразуем результат
в int и сохраняем в переменную*/
                LoadSerialNumbers(); // загружаем соответствующие серийные
номера
            }
        }
    }
}

```



```

        db.closeConnection();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
private string FindSerialNumberByFIO()
{
    if (comboBox1.SelectedItem != null)
    {
        string selectedFIO = comboBox1.SelectedItem.ToString();
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            db.openConnection();
            try
            {
                MySqlCommand clientIdCommand = new
                MySqlCommand("SELECT ID FROM Clients WHERE FIO = @selectedFIO", connection);
                clientIdCommand.Parameters.Add("@selectedFIO",
                MySqlDbType.VarChar).Value = selectedFIO;
                int clientId = Convert.ToInt32(clientIdCommand.ExecuteScalar());
                MySqlCommand serialNumberCommand = new
                MySqlCommand("SELECT Serial_Number FROM Requests INNER JOIN Errors ON
                Errors.ID = Requests.Errors_ID INNER JOIN Repair_Reports ON Requests.ID =
                Repair_Reports.Request_ID INNER JOIN Clients ON Repair_Reports.Clients_ID =

```

```

Clients.ID WHERE Clients.ID = @clientId AND Begin_Date = @selectedBegin_Date",
connection);

        serialNumberCommand.Parameters.Add("@clientId",
MySqlDbType.Int32).Value = clientId;

        DateTime day = dateTimePicker2.Value;
        string formDate = day.ToString("yyyy-MM-dd");
        serialNumberCommand.Parameters.Add("@selectedBegin_Date",
MySqlDbType.VarChar).Value = formDate;

        string serialNumber =
serialNumberCommand.ExecuteScalar()?.ToString();

        db.closeConnection();
        return serialNumber;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
        return "";
    }
}

return "";
}

private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
{
    // Получаем выбранный серийный номер
    var selectedSerialNumber = FindSerialNumberByFIO();
    if (!string.IsNullOrEmpty(selectedSerialNumber))
    {
        // Получаем доступ к базе данных
        DB db = new DB();

```

```

using (SqlConnection connection = db.getConnection())
{
    // Открываем соединение
    db.openConnection();

    // Выполняем запрос на получение данных из базы данных
    MySqlCommand command = new MySqlCommand("SELECT
Serial_Number, Creation_Date, Equipment_Type, Problem_Description, Errors.Name,
Executors.FIO, Request_Monitoring.Registered FROM Requests INNER JOIN Errors ON
Errors.ID = Requests.Errors_ID INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID INNER JOIN Executors ON Executors.ID =
Repair_Reports.Executor_ID INNER JOIN Request_Monitoring ON Requests.ID =
Request_Monitoring.Request_ID WHERE Requests.Serial_Number =
@selectedSerialNumber AND Repair_Reports.Begin_Date = @selectedBegin_Date",
connection);

    command.Parameters.Add("@selectedSerialNumber",
MySqlDbType.VarChar).Value = selectedSerialNumber;

    DateTime day = dateTimePicker2.Value;
    string formDate = day.ToString("yyyy-MM-dd");
    command.Parameters.Add("@selectedBegin_Date",
MySqlDbType.VarChar).Value = formDate;

    // Выполняем запрос и используем читатель для получения данных
    using (MySqlDataReader reader = command.ExecuteReader())
    {
        if (reader.Read())
        {
            // Заполняем поля на форме полученными данными
            textBox1.Text = reader["Equipment_Type"].ToString();
            textBox2.Text = reader["Serial_Number"].ToString();
            dateTimePicker1.Value = (DateTime)reader["Creation_Date"];
            textBox3.Text = reader["Name"].ToString();
        }
    }
}

```

```

        textBox5.Text = reader["FIO"].ToString();
        textBox4.Text = reader["Problem_Description"].ToString();
        // Заполняем значение чекбокса
        checkBox1.Checked = Convert.ToBoolean(reader["Registered"]);
    }
    else
    {
        MessageBox.Show("Запрос не вернул результатов");
        // Если запрос не вернул результатов, очищаем текстовые поля
        textBox1.Text = "";
        textBox2.Text = "";
        dateTimePicker1.Value = DateTime.Now;
        textBox3.Text = "";
        textBox4.Text = "";
        textBox5.Text = "";
        checkBox1.Checked = false;
        // Делаем поля недоступными для редактирования
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        dateTimePicker1.Enabled = false;
        textBox3.Enabled = false;
        textBox4.Enabled = false;
        textBox5.Enabled = false;
        checkBox1.Enabled = false;
    }
}
// Закрываем соединение
db.closeConnection();
}
}

```

```

    }
    private void EditApplicationAdminForm_Load(object sender, EventArgs e)
    {
        comboBox1.Text = "";
        comboBox2.Text = "";
        comboBox3.Text = "";
        LoadSerialNumbers(); // Загрузить данные при загрузке формы
        LoadFIO();
        LoadPriority();
        FillExecutorsComboBox();
        PriorityComboBox();
    }
    private void LoadSerialNumbers()
    {
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();
            // Изменяем команду запроса, добавляя параметр сортировки по ID
клиента
            MySqlCommand command = new MySqlCommand("SELECT
Serial_Number FROM Requests " +
            "INNER JOIN Repair_Reports ON Repair_Reports.Request_ID =
Requests.ID " +
            "INNER JOIN Clients ON Repair_Reports.Clients_ID = Clients.ID
WHERE Clients.ID = @ID ORDER BY Clients.ID", connection);
            command.Parameters.Add("@ID", MySqlDbType.Int32).Value =
selectedClientId;

```

```

        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                comboBox3.Items.Add(reader["Serial_Number"].ToString());
            }
        }
        db.closeConnection();
    }
}

private void LoadFIO()
{
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        db.openConnection();

        MySqlCommand command = new MySqlCommand("SELECT FIO FROM
Clients", connection);

        comboBox1.Items.Clear(); // Очищаем комбобокс перед добавлением
        НОВЫХ значений
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                comboBox1.Items.Add(reader["FIO"].ToString());
            }
        }
        db.closeConnection();
    }
}

```

```

private void FillExecutorsComboBox()
{
    try
    {
        comboBox2.Items.Clear(); // Очистим comboBox2 перед добавлением
        новых значений

        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();

            // Выполняем запрос на получение всех ФИО исполнителей
            MySqlCommand command = new MySqlCommand("SELECT FIO
            FROM Executors", connection);

            // Используем читатели для получения данных из базы
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    comboBox2.Items.Add(reader["FIO"].ToString());
                }
            }
            db.closeConnection();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}

```

```

    }
    private void LoadPriority()
    {
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            db.openConnection();
            MySqlCommand command = new MySqlCommand("SELECT Name
FROM Priority", connection);
            comboBox4.Items.Clear(); // Очищаем комбобокс перед добавлением
НОВЫХ значений
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    comboBox4.Items.Add(reader["Name"].ToString());
                }
            }
            db.closeConnection();
        }
    }
    private string PriorityComboBox()
    {
        string selectedPriorityName = string.Empty;
        try
        {
            comboBox4.Items.Clear();
            DB db = new DB();
            using (MySqlConnection connection = db.getConnection())
            {

```



```

        db.openConnection();
        MySqlCommand command = new MySqlCommand("SELECT Name
FROM Priority", connection);
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                comboBox4.Items.Add(reader["Name"].ToString());
            }
        }
        if (comboBox4.SelectedItem != null)
        {
            selectedPriorityName = comboBox4.SelectedItem.ToString();
        }
        db.closeConnection();
    }
    return selectedPriorityName;
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message);
    return "";
}
}

private string FindIDByPriority()
{
    if (comboBox4.SelectedItem != null)
    {
        string selectedName = comboBox4.SelectedItem.ToString();
        // Получаем доступ к базе данных

```

```

        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            db.openConnection();
            try
            {
                MySqlCommand priorityIdCommand = new
                MySqlCommand("SELECT ID FROM Priority WHERE Name = @selectedName",
                connection);

                priorityIdCommand.Parameters.Add("@selectedName",
                MySqlDbType.VarChar).Value = selectedName;

                int priorityId = Convert.ToInt32(priorityIdCommand.ExecuteScalar());
                db.closeConnection();
                return priorityId.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
                return "";
            }
        }
        return "";
    }

    // Обработчик события comboBox2_SelectedIndexChanged
    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox2.SelectedItem != null)
        {
            // Присваиваем выбранное ФИО из comboBox2 в textBox5

```

```

        textBox5.Text = comboBox2.SelectedItem.ToString();
    }
}

private void EditApplication_Click(object sender, EventArgs e)
{
    ConfirmationButton.Visible = true;
    LoadSerialNumbers(); // Загрузить данные при загрузке формы
    FillExecutorsComboBox();
    PriorityComboBox();
    // Делаем поля доступными для редактирования
    textBox1.Enabled = true;
    textBox2.Enabled = true;
    dateTimePicker1.Enabled = true;
    textBox3.Enabled = true;
    textBox4.Enabled = true;
    textBox5.Enabled = true;
    comboBox2.Enabled = true;
    comboBox4.Enabled = true;
    checkBox1.Enabled = true;
}

private void ConfirmationButton_Click(object sender, EventArgs e)
{
    var selectedSerialNumber = FindSerialNumberByFIO();
    var selectedPriorityID = FindIDByPriority();
    // Получаем доступ к базе данных
    DB db = new DB();
    try
    {
        using (SqlConnection connection = db.getConnection())
        {

```

```

// Открываем соединение
db.openConnection();
// Создаем команду SQL для обновления данных
MySQLCommand command = new MySQLCommand(@"UPDATE
Requests
INNER JOIN Errors ON Requests.Errors_ID =
Errors.ID
INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID
INNER JOIN Executors ON Executors.ID =
Repair_Reports.Executor_ID
INNER JOIN Request_Monitoring ON Requests.ID =
Request_Monitoring.Request_ID
INNER JOIN Work_done ON Requests.ID =
Work_done.Request_ID
SET Requests.Equipment_Type = @Equipment_Type,
Requests.Serial_Number = @Serial_Number,
Requests.Creation_Date = @Creation_Date,
Errors.Name = @Name,
Repair_Reports.Executor_ID = @ExecutorID,
Requests.Problem_Description =
@Problem_Description,
Work_done.Executor_ID = @ExecutorID,
Request_Monitoring.Priority_ID = @PriorityID
WHERE Requests.Serial_Number = @SerialNumber
AND Repair_Reports.Begin_Date = @selectedBeginDate;", connection);
// Установка значений параметров
command.Parameters.AddWithValue("@Equipment_Type",
textBox1.Text);

```

```

        command.Parameters.AddWithValue("@Serial_Number",
textBox2.Text);

        command.Parameters.AddWithValue("@Creation_Date",
dateTimePicker2.Value);

        command.Parameters.AddWithValue("@Name", textBox3.Text);
        command.Parameters.AddWithValue("@FIO",    textBox5.Text);    //
Заменили textBox5.Text на textBox с выбором ID исполнителя
        command.Parameters.AddWithValue("@Problem_Description",
textBox4.Text);

        // Получаем ExecutorID
        MySqlCommand command1 = new MySqlCommand("SELECT ID
FROM Executors WHERE FIO = @FIO", connection);
        command1.Parameters.AddWithValue("@FIO", textBox5.Text);
        int executorID = Convert.ToInt32(command1.ExecuteScalar());
        command1.ExecuteNonQuery();
        command.Parameters.AddWithValue("@ExecutorID", executorID);
        command.Parameters.AddWithValue("@PriorityID",
selectedPriorityID);

        // Форматируем выбранную дату
        string                selectedBeginDate                =
dateTimePicker2.Value.Date.ToString("yyyy-MM-dd");
        command.Parameters.AddWithValue("@selectedBeginDate",
selectedBeginDate);

        // Добавляем ID заявки
        command.Parameters.AddWithValue("@SerialNumber",
selectedSerialNumber);

        // Выполняем запрос на обновление данных в базе
        command.ExecuteNonQuery();
        ConfirmationButton.Visible    =    false;    //    Скрываем    кнопку
подтверждения

```

```

        db.closeConnection();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    var selectedSerialNumber = FindSerialNumberByFIO();
    // Проверяем, является ли чекбокс отмеченным
    if (checkBox1.Checked = true)
    {
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();
            // Получаем текущее время
            DateTime timeAcceptance = DateTime.Now;
            try
            {
                // Создаем команду SQL для обновления данных
                MySqlCommand commandUpdate = new MySqlCommand("UPDATE
Request_Monitoring INNER JOIN Requests ON Requests.ID =
Request_Monitoring.Request_ID INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID SET Registered = true, Time_acceptance = @Time_acceptance,
Processing_time = TIMESTAMPDIFF(MINUTE, Time_receipt, Time_acceptance) WHERE

```

```

Serial_Number = @selectedSerialNumber AND Repair_Reports.Begin_Date =
@selectedBegin_Date;", connection);

        // Присваиваем значения параметрам для запроса
        commandUpdate.Parameters.AddWithValue("@Time_acceptance",
timeAcceptance);

        commandUpdate.Parameters.Add("@selectedSerialNumber",
MySqlDbType.VarChar).Value = selectedSerialNumber;

        DateTime day = dateTimePicker2.Value;
        string formDate = day.ToString("yyyy-MM-dd");
        commandUpdate.Parameters.Add("@selectedBegin_Date",
MySqlDbType.VarChar).Value = formDate;

        // Выполняем запрос на обновление данных
        commandUpdate.ExecuteNonQuery();

        // Делаем чекбокс недоступным для редактирования
        checkBox1.Enabled = false;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при обновлении данных: " +
ex.Message);
    }

    // Закрываем соединение
    db.closeConnection();
}

}

private void Deletebutton_Click(object sender, EventArgs e)
{
    // Подключение к базе данных
    DB db = new DB();

```

```

using (SqlConnection connection = db.getConnection())
{
    // Получение серийного номера из формы
    string serialNumber = textBox2.Text;
    connection.Open();
    MySqlCommand command = new MySqlCommand("DELETE FROM
Requests WHERE Serial_Number = @SerialNumber;", connection);
    command.Parameters.AddWithValue("@SerialNumber", serialNumber);
    command.ExecuteNonQuery();
    connection.Close();
}
textBox1.Text = "";
dateTimePicker1.Value = DateTime.Now;
textBox2.Text = "";
textBox3.Text = "";
textBox4.Text = "";
textBox5.Text = "";
comboBox1.Text = "";
comboBox2.Text = "";
comboBox3.Text = "";
LoadSerialNumbers();
LoadFIO();
LoadPriority();
FillExecutorsComboBox();
PriorityComboBox();
}
}
}

```



## A.9 Листинг файла EditApplicationForm.cs

```
using MySql.Data.MySqlClient;
using Org.BouncyCastle.Tls;
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;
namespace WinFormsApp1
{
    public partial class EditApplicationForm : Form
    {
        public EditApplicationForm()
        {
            InitializeComponent();
        }
        /// <summary>
        /// Передвижение окна.
        /// </summary>
        Point lastPoint;
        private void panel1_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                this.Left += e.X - lastPoint.X;
            }
        }
    }
}
```

```

        this.Top += e.Y - lastPoint.Y;
    }
}
/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panell1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
    RegisterForm.loginActive = "";
}
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}
private void CloseButton_MouseLeave(object sender, EventArgs e)
{

```

```

        CloseButton.ForeColor = Color.White;
    }
    private void ToUsersMenuLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        UsersMenuForm usersMenu = new UsersMenuForm();
        usersMenu.Show();
    }
    private void ToUsersMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.Gray;
    }
    private void ToUsersMenuLabel_MouseLeave(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.White;
    }
    private void LoadSerialNumbers()
    {
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();

            // Выполняем запрос на получение серийных номеров из базы данных
            MySqlCommand command = new MySqlCommand($"SELECT
Serial_Number FROM Requests INNER JOIN Repair_Reports on
Repair_Reports.Request_ID = Requests.ID INNER JOIN Clients on
Repair_Reports.Clients_ID = Clients.ID WHERE Email = '{LoginForm.loginActive ??
RegisterForm.loginActive}'", connection);

```

```

using (MySqlDataReader reader = command.ExecuteReader())
{
    // Очищаем ComboBox
    comboBox1.Items.Clear();
    // Добавляем значения в ComboBox из базы данных
    while (reader.Read())
    {
        comboBox1.Items.Add(reader["Serial_Number"].ToString());
    }
}
// Закрываем соединение
db.closeConnection();
}
}
// Используйте этот метод вместо EditApplication_Click
private void EditApplication_Click(object sender, EventArgs e)
{
    LoadSerialNumbers(); // Загрузить серийные номера в ComboBox
    ConfirmationButton.Visible = false; // Скрыть кнопку подтверждения, пока
не выбран серийный номер
}
// Используйте этот код для выбора данных при выборе элемента в
ComboBox
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // Получаем выбранный серийный номер
    var selectedSerialNumber = comboBox1.SelectedItem?.ToString();
    if (!string.IsNullOrEmpty(selectedSerialNumber))
    {
        // Получаем доступ к базе данных

```

```

DB db = new DB();
using (SqlConnection connection = db.getConnection())
{
    // Открываем соединение
    db.openConnection();
    // Выполняем запрос на получение данных из базы данных
    MySqlCommand command = new MySqlCommand("SELECT
Serial_Number, Creation_Date, Equipment_Type, Problem_Description, Errors.Name
FROM Requests INNER JOIN Errors ON Requests.Errors_ID = Errors.ID INNER JOIN
Repair_Reports ON Requests.ID = Repair_Reports.Request_ID WHERE Serial_Number =
@selectedSerialNumber AND Begin_Date = @selectedBegin_Date", connection);
    command.Parameters.Add("@selectedSerialNumber",
MySqlDbType.VarChar).Value = selectedSerialNumber;
    command.Parameters.Add("@selectedBegin_Date",
MySqlDbType.Date).Value = dateTimePicker2.Value;
    // Используем читателя для получения данных из базы
    using (MySqlDataReader reader = command.ExecuteReader())
    {
        if (reader.Read())
        {
            // Заполняем поля на форме полученными данными
            textBox1.Text = reader["Equipment_Type"].ToString();
            textBox2.Text = reader["Serial_Number"].ToString();
            dateTimePicker1.Value = (DateTime)reader["Creation_Date"];
            textBox3.Text = reader["Name"].ToString();
            textBox4.Text = reader["Problem_Description"].ToString();
            // Делаем поля доступными для редактирования
            textBox1.Enabled = true;
            textBox2.Enabled = true;
            dateTimePicker1.Enabled = true;
        }
    }
}

```

```

        textBox3.Enabled = true;
        textBox4.Enabled = true;
        ConfirmationButton.Visible = true;
    }
}
// Закрываем соединение
db.closeConnection();
}
}
private void ConfirmationButton_Click(object sender, EventArgs e)
{
    try
    {
        var selectedSerialNumber = comboBox1.SelectedItem?.ToString();
        DateTime date1 = dateTimePicker2.Value;
        string formDate1 = date1.ToString("yyyy-MM-dd");
        DateTime date2 = dateTimePicker1.Value;
        string formDate2 = date2.ToString("yyyy-MM-dd");
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();

            // Выполняем запрос на обновление данных в базе
            MySqlCommand command = new MySqlCommand("UPDATE Requests

```

" +

```

= Repair_Reports.Request_ID " +
                                "INNER JOIN Repair_Reports ON Requests.ID
                                "INNER JOIN Errors ON Requests.Errors_ID =
Errors.ID " +
                                "SET      Requests.Equipment_Type      =
@Equipment_Type, " +
                                "Requests.Creation_Date = @Creation_Date, "
+
                                "Requests.Problem_Description          =
@Problem_Description, " +
                                "Errors.Name = @Name, " +
                                "Requests.Serial_Number                =
@NewSerial_Number " +
                                "WHERE      Requests.Serial_Number      =
@selectedSerialNumber  AND  Repair_Reports.Begin_Date  =  @selectedBegin_Date;",
connection);

```

```

                                command.Parameters.AddWithValue("@Equipment_Type",
textBox1.Text);
                                command.Parameters.AddWithValue("@Creation_Date", formDate2);
                                command.Parameters.AddWithValue("@Problem_Description",
textBox4.Text);
                                command.Parameters.AddWithValue("@Name", textBox3.Text);
                                command.Parameters.AddWithValue("@NewSerial_Number",
textBox2.Text);
                                command.Parameters.AddWithValue("@selectedSerialNumber",
selectedSerialNumber);
                                command.Parameters.AddWithValue("@selectedBegin_Date",
formDate1);

```

```

        // Выполняем запрос на обновление данных
        command.ExecuteNonQuery();

        // Закрываем соединение
        db.closeConnection();
    }

    // Делаем поля доступными для редактирования
    textBox1.Enabled = false;
    textBox2.Enabled = false;
    dateTimePicker1.Enabled = false;
    textBox3.Enabled = false;
    textBox4.Enabled = false;
    ConfirmationButton.Visible = false;
    MessageBox.Show("Заявка изменена");
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка: " + ex.Message);
}
}

private void EditApplicationForm_Load(object sender, EventArgs e)
{
    LoadSerialNumbers(); // Загрузить данные при загрузке формы
}
}
}

```



## A.10 Листинг файла EmployeeMenuForm.cs

```
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;

namespace WinFormsApp1
{
    public partial class EmployeeMenuForm : Form
    {
        public EmployeeMenuForm()
        {
            InitializeComponent();
        }

        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Hide();
            LoginForm loginForm = new LoginForm();
            loginForm.Show();
            // Очистим информацию о пользователе
            Authorization.Role = null;
            Authorization.User = null;
        }
    }
}
```

```

        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }

    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Red;
    }

    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }

    private void ToLoginLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        LoginForm loginForm = new LoginForm();
        loginForm.Show();
        // Очистим информацию о пользователе
        Authorization.Role = null;
        Authorization.User = null;
        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }

    private void ToLoginLabel_MouseEnter(object sender, EventArgs e)
    {

```

```

        ToLoginLabel.ForeColor = Color.Gray;
    }

    private void ToLoginLabel_MouseLeave(object sender, EventArgs e)
    {
        ToLoginLabel.ForeColor = Color.White;
    }

    /// <summary>
    /// Передвижение окна.
    /// </summary>
    Point lastPoint;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }
    /// <summary>
    /// Запоминание позиции курсора.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }

```

```

private void CreateApplication_Click(object sender, EventArgs e)
{
    this.Hide();
    ApplicationListForm applicationList = new ApplicationListForm();
    applicationList.Show();
}

private void StatusApplication_Click(object sender, EventArgs e)
{
    this.Hide();
    StatusAppplicationEmployeeForm statusAppplicationEmployee = new
StatusAppplicationEmployeeForm();
    statusAppplicationEmployee.Show();
}

private void Statistics_Click(object sender, EventArgs e)
{
    this.Hide();
    StatisticsForm statistics = new StatisticsForm();
    statistics.Show();
}

private void EmployeeProfile_Click(object sender, EventArgs e)
{
    this.Hide();
    EmployeeProfileForm employeeProfile = new EmployeeProfileForm();
    employeeProfile.Show();
}
}
}

```

## A.11 Листинг файла EmployeeProfileAdminForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp1
{
    public partial class EmployeeProfileAdminForm : Form
    {
        public EmployeeProfileAdminForm()
        {
            InitializeComponent();

            Point lastPoint;
            private int selectedClientId;

            private void panel1_MouseMove(object sender, MouseEventArgs e)
            {
                if (e.Button == MouseButtons.Left)
                {
```

```

        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}

private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
}

private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}

```

```
private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}
```

```
private void ToAdminsMenuLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    AdminMenuForm adminMenu = new AdminMenuForm();
    adminMenu.Show();
}
```

```
private void ToAdminsMenuLabel_MouseEnter(object sender, EventArgs e)
{
    ToAdminsMenuLabel.ForeColor = Color.Gray;
}
```

```
private void ToAdminsMenuLabel_MouseLeave(object sender, EventArgs e)
{
    ToAdminsMenuLabel.ForeColor = Color.White;
}
```

```
private void LoadFIO()
{
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        db.openConnection();
    }
}
```

```

        MySqlCommand command = new MySqlCommand("SELECT FIO FROM
Executors", connection);
        comboBox1.Items.Clear(); // Очищаем комбобокс перед добавлением
НОВЫХ значений
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                comboBox1.Items.Add(reader["FIO"].ToString());
            }
        }
        db.closeConnection();
    }
}
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        string selectedFIO = comboBox1.SelectedItem.ToString();
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();
            string selectQuery = "SELECT FIO, Position, Department, Phone, Email
FROM Executors WHERE FIO = @selectedFIO";
            MySqlCommand command = new MySqlCommand(selectQuery,
connection);
            command.Parameters.AddWithValue("@selectedFIO", selectedFIO);
            using (MySqlDataReader reader = command.ExecuteReader())

```



```

    {
        if (reader.Read())
        {
            textBox1.Text = reader["FIO"].ToString();
            textBox2.Text = reader["Position"].ToString();
            textBox3.Text = reader["Department"].ToString();
            textBox4.Text = reader["Phone"].ToString();
            textBox5.Text = reader["Email"].ToString();
            textBox6.Text = "*****";
        }
    }
    connection.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка данных: " + ex.Message);
}
}
private void EditProfile_Click(object sender, EventArgs e)
{
    // Сделать все текстовые поля доступными для редактирования
    textBox1.Enabled = true;
    textBox2.Enabled = true;
    textBox3.Enabled = true;
    textBox4.Enabled = true;
    textBox5.Enabled = true;
    textBox6.Enabled = true;
    ConfirmationButton.Visible = true;
}
}

```

```

private string HashPassword(string password)
{
    using (SHA256 sha256 = SHA256.Create())
    {
        byte[] bytes = Encoding.UTF8.GetBytes(password);
        byte[] hash = sha256.ComputeHash(bytes);
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < hash.Length; i++)
        {
            result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX
        }
        return result.ToString();
    }
}

// Обработчик кнопки ConfirmationButton
private void ConfirmationButton_Click(object sender, EventArgs e)
{
    // Проверка введенных данных и их сохранение в базе
    string fio = textBox1.Text;
    string position = textBox2.Text;
    string department = textBox3.Text;
    string phone = textBox4.Text;
    // Дополнительно получаем введенные логин и пароль
    string login = textBox5.Text;
    string password = HashPassword(textBox6.Text);
    string password1 = textBox6.Text;
    // Проверка формата введенных данных
    if (!IsValidFIO(fio))
    {

```

строки

```

        MessageBox.Show("Фамилия должна содержать только буквы");
        return;
    }
    if (!IsValidPhone(phone))
    {
        MessageBox.Show("Номер телефона должен содержать ровно 11
цифр");

        return;
    }
    if (!IsValidPosition(position))
    {
        MessageBox.Show("Должность должна содержать только буквы");
        return;
    }
    if (!IsValidPassword(password1))
    {
        MessageBox.Show("Пароль должен содержать от 4 до 16 символов");
        return;
    }
    // Проверка на использование определенных символов
    string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№",
"\", \"", "#", ";", ":",
        "%", "&", ",", "?", "~", "!", "=", "\\", "/" };
    if (restrictedSymbols.Any(symbol => textBox5.Text.Contains(symbol)))
    {
        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода
    }
    if (restrictedSymbols.Any(symbol => textBox6.Text.Contains(symbol)))

```

```

    {
        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода
    }
    // Получение введенных логина и пароля
    string newLogin = textBox5.Text;
    string newPassword = HashPassword(textBox6.Text);
    string oldPasswordFromDB;
    // Получение старого логина из глобальной переменной или другого
места, где он хранится
    string selectedFIO = comboBox1.SelectedItem.ToString();
    string oldLogin = GetLoginFromDB(selectedFIO); // Примерное название
переменной, где хранится текущий логин
// Получение старого пароля из базы данных
по старому логину
    string oldPasswordFromDBQuery = $"SELECT Password FROM Executors
Where Email = '{oldLogin}'";
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        // Открываем соединение
        connection.Open();
        using (MySqlCommand oldPasswordFromDBCommand = new
MySqlCommand(oldPasswordFromDBQuery, connection))
        {
            oldPasswordFromDB =
(string)oldPasswordFromDBCommand.ExecuteScalar();
        }
        // Закрываем соединение

```

```

        connection.Close();
    }
    if (newLogin != oldLogin || newPassword != oldPasswordFromDB)
    {
        // Выводим окно с подтверждением изменения логина и пароля
        DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите
изменить логин и пароль?", "Подтверждение изменений", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            // Обновляем логин и пароль в базе данных
            string updateLoginQuery = $"UPDATE Executors SET Email =
'{newLogin}' WHERE Email = '{oldLogin}'";
            string updatePasswordQuery = $"UPDATE Executors SET Password =
'{newPassword}' WHERE Email = '{newLogin}'";
            using (SqlConnection connection = db.getConnection())
            {
                // Открываем соединение
                connection.Open();
                using (SqlCommand updateLoginCommand = new
SqlCommand(updateLoginQuery, connection))
                using (SqlCommand updatePasswordCommand = new
SqlCommand(updatePasswordQuery, connection))
                {
                    updateLoginCommand.ExecuteNonQuery();
                    updatePasswordCommand.ExecuteNonQuery();
                }
                // Закрываем соединение
                connection.Close();
            }
            // Обновление данных в базе данных

```

```

        string updateQuery = $"UPDATE Executors SET FIO = '{fio}', Position
= '{position}', Department = '{department}', Phone = '{phone}' WHERE Email =
'{'newLogin}';";

        using (SqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();

            using (SqlCommand updateQueryCommand = new
SqlCommand(updateQuery, connection))
            {
                updateQueryCommand.ExecuteNonQuery();
            }

            // Закрываем соединение
            connection.Close();
        }

        // Сделать текстовые поля недоступными для редактирования
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        textBox3.Enabled = false;
        textBox4.Enabled = false;
        textBox5.Enabled = false;
        textBox6.Enabled = false;
        ConfirmationButton.Visible = false;
    }
    else
    {
        // Блокируем поля логина и пароля и оставляем старые данные
        textBox5.Text = oldLogin;
        textBox6.Text = oldPasswordFromDB; // Допустим, что мы не
обновляем пароль, если изменение не подтверждено

```

// Обновление данных в базе данных

```
string updateQuery = $"UPDATE Executors SET FIO = '{fio}', Position  
= '{position}', Department = '{department}', Phone = '{phone}' WHERE Email =  
'{oldLogin}';";
```

```
using (SqlConnection connection = db.getConnection())  
{  
    connection.Open();  
    using (SqlCommand updateQueryCommand = new  
SqlCommand(updateQuery, connection))  
    {  
        updateQueryCommand.ExecuteNonQuery();  
    }  
    connection.Close();  
}
```

// Сделать текстовые поля недоступными для редактирования

```
textBox1.Enabled = false;  
textBox2.Enabled = false;  
textBox3.Enabled = false;  
textBox4.Enabled = false;  
textBox5.Enabled = false;  
textBox6.Enabled = false;  
ConfirmationButton.Visible = false;  
  
}  
  
}  
textBox6.Text = "*****";  
textBox6.PasswordChar = '*';  
textBox6.UseSystemPasswordChar = true;  
  
}  
  
// Метод для проверки формата ФИО  
private bool IsValidFIO(string fio)
```

```

    {
        // Проверяем, что строка состоит только из букв и пробелов
        return !string.IsNullOrEmpty(fio) && fio.All(c => char.IsLetter(c) ||
char.IsWhiteSpace(c));
    }
    // Метод для проверки формата паспортных данных
    private bool IsValidPosition(string position)
    {
        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(position) && position.All(c =>
char.IsLetter(c) || char.IsWhiteSpace(c));
    }
    // Метод для проверки формата номера телефона
    private bool IsValidPhone(string phone)
    {
        // Проверяем, что строка состоит из 11 цифр
        return !string.IsNullOrEmpty(phone) && phone.Length == 11 &&
phone.All(c => char.IsDigit(c));
    }
    private bool IsValidPassword(string password)
    {
        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(password) && password.Length == 16 &&
password.Length > 4;
    }
    private string GetLoginFromDB(string selectedFIO)
    {
        string login = "";
        DB db = new DB();
        using (SqlConnection connection = db.getConnection())

```



```

        {
            db.openConnection();
            MySqlCommand command = new MySqlCommand("SELECT Email
FROM Executors WHERE FIO = @selectedFIO", connection);
            command.Parameters.Add("@selectedFIO",
MySqlDbType.VarChar).Value = selectedFIO;
            object result = command.ExecuteScalar();
            if (result != null)
            {
                login = result.ToString(); // Присваиваем значение логина
            }
            db.closeConnection();
        }
        return login;
    }
    private void DeleteButton_Click(object sender, EventArgs e)
    {
        string selectedFIO = comboBox1.SelectedItem.ToString();
        string oldLogin = GetLoginFromDB(selectedFIO);
        // Подключение к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            connection.Open();
            MySqlCommand command = new MySqlCommand("DELETE FROM
Executors WHERE Email = @Login;", connection);
            command.Parameters.AddWithValue("@Login", oldLogin);
            command.ExecuteNonQuery();
            connection.Close();
        }
    }

```

```

        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        textBox5.Text = "";
        textBox6.Text = "";
        comboBox1.Text = "";
        LoadFIO();
    }
    private void EmployeeProfileAdminForm_Load(object sender, EventArgs e)
    {
        LoadFIO();
    }
}

```

## A.12 Листинг файла EmployeeProfileForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WinFormsApp1
{
    public partial class EmployeeProfileForm : Form
    {
        public EmployeeProfileForm()
        {
            InitializeComponent();
        }
        private void EditProfile_Click(object sender, EventArgs e)
        {
            // Сделать все текстовые поля доступными для редактирования
            textBox1.Enabled = true;
            textBox2.Enabled = true;
            textBox3.Enabled = true;
            textBox4.Enabled = true;
            textBox5.Enabled = true;
            textBox6.Enabled = true;
            ConfirmationButton.Visible = true;
        }
    }
}
```

```

    }
    private string HashPassword(string password)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(password);
            byte[] hash = sha256.ComputeHash(bytes);
            StringBuilder result = new StringBuilder();
            for (int i = 0; i < hash.Length; i++)
            {
                result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX строки
            }
            return result.ToString();
        }
    }
    // Обработчик кнопки ConfirmationButton
    private void ConfirmationButton_Click(object sender, EventArgs e)
    {
        // Проверка введенных данных и их сохранение в базе
        string fio = textBox1.Text;
        string position = textBox2.Text;
        string department = textBox3.Text;
        string phone = textBox4.Text;
        // Дополнительно получаем введенные логин и пароль
        string login = textBox5.Text;
        string password = HashPassword(textBox6.Text);
        string password1 = textBox6.Text;
        // Проверка формата введенных данных
        if (!IsValidFIO(fio))
        {
            MessageBox.Show("Фамилия должна содержать только буквы");
            return;
        }
    }

```

```

if (!IsValidPhone(phone))
{
    MessageBox.Show("Номер телефона должен содержать ровно 11 цифр");
    return;
}
if (!IsValidPosition(position))
{
    MessageBox.Show("Должность должна содержать только буквы");
    return;
}
if (!IsValidPassword(password1))
{
    MessageBox.Show("Пароль должен содержать от 4 до 16 символов");
    return;
}
// Проверка на использование определенных символов
string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№", "\"", "'",
"#", ";", ":",
    "%", "&", ", ", "?", "~", "!", "=", "\\", "/" };
if (restrictedSymbols.Any(symbol => textBox5.Text.Contains(symbol)))
{
    MessageBox.Show("Поля не должны содержать запрещенные символы: " +
string.Join(" ", restrictedSymbols));
    return; // Прерываем выполнение метода
}
if (restrictedSymbols.Any(symbol => textBox6.Text.Contains(symbol)))
{
    MessageBox.Show("Поля не должны содержать запрещенные символы: " +
string.Join(" ", restrictedSymbols));
    return; // Прерываем выполнение метода
}
// Получение введенных логина и пароля
string newLogin = textBox5.Text;

```

```

string newPassword = HashPassword(textBox6.Text);
string oldPasswordFromDB;
// Получение старого логина из глобальной переменной или другого места, где
он хранится
string oldLogin = LoginForm.loginActive;// Примерное название переменной, где
хранится текущий логин

// Получение старого пароля из базы данных по старому логину
string oldPasswordFromDBQuery = $"SELECT Password FROM Executors Where
Email = '{oldLogin}'";
DB db = new DB();
using (MySqlConnection connection = db.getConnection())
{
    // Открываем соединение
    connection.Open();
    using (MySqlCommand oldPasswordFromDBCommand = new
MySqlCommand(oldPasswordFromDBQuery, connection))
    {
        oldPasswordFromDB =
(string)oldPasswordFromDBCommand.ExecuteScalar();
    }
    // Закрываем соединение
    connection.Close();
}
if (newLogin != oldLogin || newPassword != oldPasswordFromDB)
{
    // Выводим окно с подтверждением изменения логина и пароля
    DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите
изменить логин и пароль?", "Подтверждение изменений", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        // Обновляем логин и пароль в базе данных

```

```

        string updateLoginQuery = $"UPDATE Executors SET Email = '{newLogin}'
WHERE Email = '{oldLogin}'";

        string updatePasswordQuery = $"UPDATE Executors SET Password =
'{newPassword}' WHERE Email = '{newLogin}'";

        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();

            using (MySqlCommand updateLoginCommand = new
MySqlCommand(updateLoginQuery, connection))
            using (MySqlCommand updatePasswordCommand = new
MySqlCommand(updatePasswordQuery, connection))
            {
                updateLoginCommand.ExecuteNonQuery();
                updatePasswordCommand.ExecuteNonQuery();
            }

            // Закрываем соединение
            connection.Close();
        }

        // Обновление данных в базе данных
        string updateQuery = $"UPDATE Executors SET FIO = '{fio}', Position =
'{position}', Department = '{department}', Phone = '{phone}' WHERE Email = '{newLogin}'";

        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();

            using (MySqlCommand updateQueryCommand = new
MySqlCommand(updateQuery, connection))
            {
                updateQueryCommand.ExecuteNonQuery();
            }

            // Закрываем соединение
            connection.Close();

```

```

    }
    // Сделать текстовые поля недоступными для редактирования
    textBox1.Enabled = false;
    textBox2.Enabled = false;
    textBox3.Enabled = false;
    textBox4.Enabled = false;
    textBox5.Enabled = false;
    textBox6.Enabled = false;
    ConfirmationButton.Visible = false;
}
else
{
    // Блокируем поля логина и пароля и оставляем старые данные
    textBox5.Text = oldLogin;
    textBox6.Text = oldPasswordFromDB; // Допустим, что мы не обновляем
пароль, если изменение не подтверждено

    // Обновление данных в базе данных
    string updateQuery = $"UPDATE Executors SET FIO = '{fio}', Position =
'{'position}', Department = '{department}', Phone = '{phone}' WHERE Email = '{oldLogin}';";
    using (MySQLConnection connection = db.getConnection())
    {
        connection.Open();
        using (MySQLCommand updateQueryCommand = new
MySQLCommand(updateQuery, connection))
        {
            updateQueryCommand.ExecuteNonQuery();
        }
        connection.Close();
    }
    // Сделать текстовые поля недоступными для редактирования
    textBox1.Enabled = false;
    textBox2.Enabled = false;
    textBox3.Enabled = false;

```



```

        textBox4.Enabled = false;
        textBox5.Enabled = false;
        textBox6.Enabled = false;
        ConfirmationButton.Visible = false;

    }
    textBox6.Text = "*****";
    textBox6.PasswordChar = '*';
    textBox6.UseSystemPasswordChar = true;
}
// Метод для проверки формата ФИО
private bool IsValidFIO(string fio)
{
    // Проверяем, что строка состоит только из букв и пробелов
    return !string.IsNullOrEmpty(fio) && fio.All(c => char.IsLetter(c) ||
char.IsWhiteSpace(c));
}
// Метод для проверки формата паспортных данных
private bool IsValidPosition(string position)
{
    // Проверяем, что строка состоит из 10 цифр
    return !string.IsNullOrEmpty(position) && position.All(c => char.IsLetter(c) ||
char.IsWhiteSpace(c));
}
// Метод для проверки формата номера телефона
private bool IsValidPhone(string phone)
{
    // Проверяем, что строка состоит из 11 цифр
    return !string.IsNullOrEmpty(phone) && phone.Length == 11 && phone.All(c
=> char.IsDigit(c));
}
private bool IsValidPassword(string password)
{

```

```

        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(password) && password.Length == 16 &&
password.Length > 4;
    }
    private void EmployeeProfileForm_Load(object sender, EventArgs e)
    {
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();
            string oldLogin = LoginForm.loginActive; // Примерное название переменной,
где хранится текущий логин
            // Подготавливаем SQL-запрос
            string selectQuery = "SELECT FIO, Position, Department, Phone, Email FROM
Executors WHERE Email = @Email";
            MySqlCommand command = new MySqlCommand(selectQuery, connection);
            command.Parameters.AddWithValue("@Email", oldLogin);
            // Используем читатели для получения данных из базы
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    // Заполняем поля на форме полученными данными
                    textBox1.Text = reader["FIO"].ToString();
                    textBox2.Text = reader["Position"].ToString();
                    textBox3.Text = reader["Department"].ToString();
                    textBox4.Text = reader["Phone"].ToString();
                    textBox5.Text = reader["Email"].ToString();
                    textBox6.Text = "*****";
                }
            }
        }
    }

```

```

        // Закрываем соединение
        db.closeConnection();
    }
}
Point lastPoint;
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
    RegisterForm.loginActive = "";
}

```

```

    }
    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Red;
    }
    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }
    private void ToEmployeeMenuLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        EmployeeMenuForm employeeMenu = new EmployeeMenuForm();
        employeeMenu.Show();
    }
    private void ToEmployeeMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToEmployeeMenuLabel.ForeColor = Color.Gray;
    }

    private void ToEmployeeMenuLabel_MouseLeave(object sender, EventArgs e)
    {
        ToEmployeeMenuLabel.ForeColor = Color.White;
    }
}
}

```

## A.13 Листинг файла LoginForm.cs

```
using Microsoft.VisualBasic.Devices;
using MySql.Data.MySqlClient;
using System.Data;
using System.Security.Cryptography;
using System.Text;
using System.Windows.Forms;
namespace WinFormsApp1
{
    public partial class LoginForm : Form
    {
        public static LoginForm? Instance { get; set; }
        public static RegisterForm? Instance1 = null;
        public static UsersMenuForm? Instance2 = null;
        public static EmployeeMenuForm? Instance3 = null;
        static public string loginActive;
        static public string WhoIs;
        /// <summary>
        /// Закрытие окна.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CloseButton_Click_1(object sender, EventArgs e)
        {
            Application.Exit();
            System.Diagnostics.Process.GetCurrentProcess().Kill();
        }
        /// <summary>
        /// Форматирование для пароля.
```

```

/// </summary>
public LoginForm()
{
    InitializeComponent();
    this.PassField.AutoSize = false;
    this.PassField.Size = new Size(this.PassField.Size.Width, 45);
}
/// <summary>
/// Событие по наведению мышкой на кнопку.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}
/// <summary>
/// Событие по отпусканию кнопки.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}
/// <summary>
/// Передвижение окна.
/// </summary>
Point lastPoint;
private void panell1_MouseMove(object sender, MouseEventArgs e)

```

```

{
    if (e.Button == MouseButton.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}

/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}

private string HashPassword(string password)
{
    using (SHA256 sha256 = SHA256.Create())
    {
        byte[] bytes = Encoding.UTF8.GetBytes(password);
        byte[] hash = sha256.ComputeHash(bytes);
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < hash.Length; i++)
        {
            result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX
        }
    }
}

```

строки

```

        return result.ToString();
    }
}

private void ButtonLogin_Click(object sender, EventArgs e)
{
    String passUser = PassField.Text;
    String loginUser = LoginField.Text;
    DB db = new DB();
    loginActive = loginUser;
    DataTable table1 = new DataTable();
    MySqlDataAdapter adapter1 = new MySqlDataAdapter();
    MySqlCommand command1 = new MySqlCommand("SELECT * FROM
Clients Where Email = @uL AND (Password = @uP1 OR Password = @uP2);",
db.getConnection());
    command1.Parameters.Add("@uL",    MySqlDbType.VarChar).Value    =
loginUser;
    command1.Parameters.Add("@uP1",    MySqlDbType.VarChar).Value    =
passUser;
    command1.Parameters.Add("@uP2",    MySqlDbType.VarChar).Value    =
HashPassword(passUser);
    adapter1.SelectCommand = command1;
    adapter1.Fill(table1);

    DataTable table2 = new DataTable();
    MySqlDataAdapter adapter2 = new MySqlDataAdapter();
    MySqlCommand command2 = new MySqlCommand("SELECT * FROM
Executors Where Email = @uL AND (Password = @uP1 OR Password = @uP2);",
db.getConnection());

```



```

        command2.Parameters.Add("@uL",    MySqlDbType.VarChar).Value    =
loginUser;

        loginActive = loginUser;

        command2.Parameters.Add("@uP1",    MySqlDbType.VarChar).Value    =
passUser;

        command2.Parameters.Add("@uP2",    MySqlDbType.VarChar).Value    =
HashPassword(passUser);

        adapter2.SelectCommand = command2;
        adapter2.Fill(table2);


        DataTable table3 = new DataTable();
        MySqlDataAdapter adapter3 = new MySqlDataAdapter();
        MySqlCommand command3 = new MySqlCommand("SELECT * FROM
Admins Where Email = @uL AND (Password = @uP1 OR Password = @uP2);",
db.getConnection());

        command3.Parameters.Add("@uL",    MySqlDbType.VarChar).Value    =
loginUser;

        command3.Parameters.Add("@uP1",    MySqlDbType.VarChar).Value    =
passUser;

        command3.Parameters.Add("@uP2",    MySqlDbType.VarChar).Value    =
HashPassword(passUser);

        adapter3.SelectCommand = command3;
        adapter3.Fill(table3);


        DataTable table4 = new DataTable();
        MySqlConnection connection = new MySqlConnection("user=root;
host=localhost; password=IvanVladimirov888; database=Computer_systems;");
        connection.Open();

```

```

MySQLDataAdapter adapter4 = new MySQLDataAdapter();
string[] commands = new string[]
{
    @"SELECT Email, Password FROM Clients WHERE Email = @uL",
    @"SELECT Email, Password FROM Executors WHERE Email = @uL",
    @"SELECT Email, Password FROM Admins WHERE Email = @uL"
};

foreach (string commandText in commands)
{
    MySqlCommand command = new MySqlCommand(commandText,
connection);

    command.Parameters.Add("@uL", MySQLDbType.VarChar).Value =
loginUser; // loginUser - ваш параметр поиска
    adapter4.SelectCommand = command;
    adapter4.Fill(table4);
}

// Проверка на использование определенных символов
string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№",
"\", "\"", "#", ";", ":",
"%", "&", ", ", "?", "~", "!", "=", "\\", "/" };
if (restrictedSymbols.Any(symbol => LoginField.Text.Contains(symbol)))
{
    MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
    return; // Прерываем выполнение метода
}
if (restrictedSymbols.Any(symbol => PassField.Text.Contains(symbol)))
{

```

```

        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода
    }

    if ((table4.Rows.Count > 0) && ((table4.Rows[0]["Password"].ToString() !=
HashPassword(passUser)) && (table4.Rows[0]["Email"].ToString() == loginUser)))
    {
        MessageBox.Show("Введен неправильный пароль", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    else if (string.IsNullOrEmpty(loginUser))
    {
        MessageBox.Show("Введите логин");
    }
    else if (string.IsNullOrEmpty(passUser))
    {
        MessageBox.Show("Введите пароль");
    }
    else
    {
        if (passUser != "" && loginUser != "")
        {

            Authorization.Authorization1(loginUser, passUser);
            switch (Authorization.Role)
            {
                case null:

```

```

        {
            MessageBox.Show("Такого аккаунта не существует!",
"Проверьте данные и попробуйте снова.", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            break;
        }
case "admin":
    {
        WhoIs = "Администратор";
        Authorization.User = LoginField.Text;

        string FIO = Authorization.AuthorizationName(loginUser);
        Authorization.FIO = FIO;
        MessageBox.Show(FIO + ", добро пожаловать в меню
администратора!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);
        this.Hide();
        AdminMenuForm admin = new AdminMenuForm();
        admin.Show();
        break;
    }
case "user":
    {
        WhoIs = "Пользователь";
        Authorization.User = LoginField.Text;

        string FIO = Authorization.AuthorizationName(loginUser);
        Authorization.FIO = FIO;
        MessageBox.Show(FIO + ", добро пожаловать в меню
пользователя!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);
        this.Hide();
    }

```

```

        UsersMenuForm user = new UsersMenuForm();
        user.Show();
        break;
    }
    case "employee":
    {
        WhoIs = "Сотрудник";
        Authorization.User = LoginField.Text;

        string FIO = Authorization.AuthorizationName(loginUser);
        Authorization.FIO = FIO;
        MessageBox.Show(FIO + ", добро пожаловать в меню
сотрудника!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);
        this.Hide();
        EmployeeMenuForm employee = new EmployeeMenuForm();
        employee.Show();
        break;
    }
}
}
else
{

    MessageBox.Show("Заполните все поля", "Заполнение полей",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}
}

```

```

private void registerLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    RegisterForm registerForm = new RegisterForm();
    registerForm.Show();
}

private void registerLabel_MouseEnter(object sender, EventArgs e)
{
    registerLabel.ForeColor = Color.AliceBlue;
}

private void registerLabel_MouseLeave(object sender, EventArgs e)
{
    registerLabel.ForeColor = Color.DarkSlateGray;
}

}
}

```

## A.14 Листинг файла RegisterForm.cs

```
using MySql.Data.MySqlClient;
using System.Data;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;

namespace WinFormsApp1
{
    public partial class RegisterForm : Form
    {
        public static RegisterForm? Instance1 { get; set; }
        public static LoginForm? Instance = null;
        public static UsersMenuForm? Instance2 = null;
        public static EditApplicationAdminForm? Instance3 = null;
        static public string loginActive;

        /// <summary>
        /// Форматирование для пароля.
        /// </summary>
        public RegisterForm()
        {
            InitializeComponent();

            this.UserInitialsField.AutoSize = false;
            this.UserInitialsField.Size = new Size(this.UserInitialsField.Size.Width, 45);
        }
    }
}
```

```

private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
    RegisterForm.loginActive = "";
}

/// <summary>
/// Событие по наведению мышкой на кнопку.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}

/// <summary>
/// Событие по отпусканию кнопки.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void CloseButton_MouseLeave(object sender, EventArgs e)
{

```



```

        CloseButton.ForeColor = Color.White;
    }
    /// <summary>
    /// Передвижение окна.
    /// </summary>
    Point lastPoint;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }
    /// <summary>
    /// Запоминание позиции курсора.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }

    private string HashPassword(string password)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(password);
            byte[] hash = sha256.ComputeHash(bytes);

```

```

        StringBuilder result = new StringBuilder();

        for (int i = 0; i < hash.Length; i++)
        {
            result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX
строки
        }

        return result.ToString();
    }
}

private void ButtonRegistration_Click(object sender, EventArgs e)
{
    String passUser = HashPassword(PassField.Text); // Хешируем введенный
пароль

    String loginUser = LoginField.Text;
    String initialsuser = UserInitialsField.Text;
    LoginForm.loginActive = "";

    if (isUserExists())
        return;

    // Проверка на использование определенных символов
    string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№",
"\", \"", "#", ";", ":",
        "%", "&", ",", "?", "~", "!", "=", "\\", "/" };
    if (restrictedSymbols.Any(symbol => LoginField.Text.Contains(symbol)) ||
(restrictedSymbols.Any(symbol => PassField.Text.Contains(symbol))))
    {

```

```

        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода

    }
    else if (string.IsNullOrEmpty(initialsuser))
    {
        MessageBox.Show("Введите ФИО");
    }
    else if (string.IsNullOrEmpty(loginUser))
    {
        MessageBox.Show("Введите логин");
    }
    else if (string.IsNullOrEmpty(passUser))
    {
        MessageBox.Show("Введите пароль");
    }
    else
    {
        DB db = new DB();
        MySqlCommand command1 = new MySqlCommand("INSERT INTO
Clients (FIO, Passport, Address, Phone, Password, Email, ID_Roles) VALUES (@uI,0, ", ",
@uP, @uL, 3)", db.getConnection());
        command1.Parameters.Add("@uI",    MySqlDbType.VarChar).Value    =
initialsuser;
        command1.Parameters.Add("@uP",    MySqlDbType.VarChar).Value    =
passUser;
        command1.Parameters.Add("@uL",    MySqlDbType.VarChar).Value    =
loginUser;
        loginActive = loginUser;
    }
}

```

```

        db.openConnection();
        command1.ExecuteNonQuery();
        db.closeConnection();
        MessageBox.Show("Регистрация аккаунта произошла успешно!");
        this.Hide();
        UsersMenuForm usersMenu = new UsersMenuForm();
        usersMenu.Show();
    }
}

public Boolean isUserExists()
{
    String passUser = UserInitialsField.Text;
    String loginUser = LoginField.Text;

    DB db = new DB();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand("SELECT * FROM
Clients Where Email = @uL", db.getConnection());
    command.Parameters.Add("@uL",    MySqlDbType.VarChar).Value    =
loginUser;

    adapter.SelectCommand = command;
    adapter.Fill(table);

    if (table.Rows.Count > 0)
    {
        MessageBox.Show("Под таким логином уже существует аккаунт.
Введите другой!");
    }
}

```

```

        return true;
    }
    else
        return false;
}

private void ToLoginLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
}

private void ToLoginLabel_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Gray;
}

private void ToLoginLabel_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}
}
}

```

## A.15 Листинг файла StatisticsForm.cs

```
using Microsoft.VisualBasic.Logging;
using MySql.Data.MySqlClient;
using System;
using System.CodeDom.Compiler;
using System.Collections;
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 static System.Runtime.InteropServices.JavaScript.JSType;

namespace WinFormsApp1
{
    public partial class StatisticsForm : Form
    {
        public StatisticsForm()
        {
            InitializeComponent();
        }

        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Hide();
            LoginForm loginForm = new LoginForm();
        }
    }
}
```

```

        loginForm.Show();
        // Очистим информацию о пользователе
        Authorization.Role = null;
        Authorization.User = null;
        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }

    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Red;
    }

    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }

    private void ToEmployeeMenuLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        EmployeeMenuForm employeeMenu = new EmployeeMenuForm();
        employeeMenu.Show();
    }

    private void ToEmployeeMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToEmployeeMenuLabel.ForeColor = Color.Gray;
    }

```

```
private void ToEmployeeMenuLabel_MouseLeave(object sender, EventArgs e)
{
    ToEmployeeMenuLabel.ForeColor = Color.White;
}
```

```
/// <summary>
```

```
/// Передвижение окна.
```

```
/// </summary>
```

```
Point lastPoint;
```

```
private void panel1_MouseMove(object sender, MouseEventArgs e)
```

```
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
```

```
/// <summary>
```

```
/// Запоминание позиции курсора.
```

```
/// </summary>
```

```
/// <param name="sender"></param>
```

```
/// <param name="e"></param>
```

```
private void panel1_MouseDown(object sender, MouseEventArgs e)
```

```
{
    lastPoint = new Point(e.X, e.Y);
}
```

```
private void StatisticsForm_Load(object sender, EventArgs e)
```

```
{
```



```

        CalculateData();
        FillErrorsComboBox();
    }

```

```

private const string ConnectionString = "user=root; host=localhost;
password=IvanVladimirov888; database=Computer_systems";

```

```

public void CalculateData()
{
    try
    {
        long completedRequestsCount = CalculateCompletedRequestsCount(); //

```

Изменено на тип long

```

        double averageExecutionTime = CalculateAverageExecutionTime();
        textBox1.Text = completedRequestsCount.ToString();
        textBox2.Text = averageExecutionTime.ToString();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка данных: " + ex.Message);
    }
}

```

private long CalculateCompletedRequestsCount() // Изменен тип  
возвращаемого значения на long

```

{
    long completedRequestsCount = 0;
    string Login = LoginForm.loginActive;
    DB db = new DB();
    using (SqlConnection connection = db.getConnection())

```

```

        {
            connection.Open();

            MySqlCommand command = new MySqlCommand("SELECT COUNT(*)
FROM Requests INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID INNER JOIN Executors ON Executors.ID =
Repair_Reports.Executor_ID WHERE Repair_Reports.End_Date <>
Repair_Reports.Begin_Date AND Executors.Email = @Login;", connection);

            command.Parameters.Add("@Login", MySqlDbType.VarChar).Value =
Login;

            {
                completedRequestsCount = (long)command.ExecuteScalar(); //
Изменено на тип long
            }

            connection.Close();

            return completedRequestsCount;
        }
    }

    private double CalculateAverageExecutionTime()
    {
        double averageExecutionTime = 0;
        string Login = LoginForm.loginActive;
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            connection.Open();

            MySqlCommand command = new MySqlCommand("SELECT
AVG(Work_time) FROM Work_done INNER JOIN Executors ON Executors.ID =
Work_done.Executor_ID WHERE Executors.Email = @Login;", connection);

            command.Parameters.Add("@Login", MySqlDbType.VarChar).Value =
Login;

```

```

        {
            object result = command.ExecuteScalar();
            if (result != DBNull.Value)
            {
                averageExecutionTime = Convert.ToDouble(result);
            }
        }
        connection.Close();
        return averageExecutionTime;
    }
}

private void FillErrorsComboBox()
{
    comboBox1.Items.Clear(); // Очистка comboBox1 перед заполнением
    string Login = LoginForm.loginActive;
    using (MySqlConnection connection = new
MySqlConnection(ConnectionString))
    {
        try
        {
            connection.Open();
            string query = "SELECT Name FROM Requests INNER JOIN Errors
ON Requests.Errors_ID = Errors.ID INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID INNER JOIN Executors ON Executors.ID =
Repair_Reports.Executor_ID WHERE Executors.Email = @Login;";
            MySqlCommand command = new MySqlCommand(query, connection);
            command.Parameters.Add("@Login", MySqlDbType.VarChar).Value =
Login;

            MySqlDataReader reader = command.ExecuteReader();
            while (reader.Read())

```

```

        {
            string errorName = reader["Name"].ToString();
            comboBox1.Items.Add(errorName);
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    string connectionString = "user=root; host=localhost;
password=IvanVladimirov888; database=Computer_systems"; // Замените на вашу строку
подключения

    string ErrorsName = comboBox1.SelectedItem as string;
    string Login = LoginForm.loginActive;
    using (MySqlConnection connection = new
MySqlConnection(connectionString))
    {
        connection.Open();

        string query = "SELECT Repair_Reports.Begin_Date,
Repair_Reports.End_Date, Priority.Name, Equipment_Type, Serial_Number,
Creation_Date, Problem_Description, Errors.Name " +
            " FROM Requests " +
            " INNER JOIN Errors ON Requests.Errors_ID = Errors.ID " +

```

```

        " INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID " +
        " INNER JOIN Executors ON Executors.ID =
Repair_Reports.Executor_ID " +
        " INNER JOIN Request_Monitoring ON Requests.ID =
Request_Monitoring.Request_ID " +
        " INNER JOIN Work_done ON Requests.ID =
Work_done.Request_ID " +
        " INNER JOIN Priority ON Priority.ID =
Request_Monitoring.Priority_ID " +
        " WHERE Begin_Date <> End_Date AND Errors.Name =
@Name AND Executors.Email = @Login;";

        MySqlCommand command = new MySqlCommand(query, connection);
        command.Parameters.Add("@Login", MySqlDbType.VarChar).Value =
Login;

        command.Parameters.Add("@Name", MySqlDbType.VarChar).Value =
ErrorsName;

        using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))
        {
            DataTable dataTable = new DataTable();
            adapter.Fill(dataTable);

            dataGridView1.DataSource = dataTable;
        }
    }
}
}
}
}

```

## A.16 Листинг файла StatusApplicationForm.cs

```
using Microsoft.VisualBasic.ApplicationServices;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Office.Interop.Word;
using System.IO;
using NPOI.XWPF.UserModel;
using System.Diagnostics;

namespace WinFormsApp1
{
    public partial class StatusApplicationForm : Form
    {
        public StatusApplicationForm()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Передвижение окна.
    }
}
```

```

/// </summary>
System.Drawing.Point lastPoint;
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new System.Drawing.Point(e.X, e.Y);
}

private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
}

```

```

        RegisterForm.loginActive = "";
    }

    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Red;
    }

    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }

    private void ToUsersMenuLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        UsersMenuForm usersMenu = new UsersMenuForm();
        usersMenu.Show();
    }

    private void ToUsersMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.Gray;
    }

    private void ToUsersMenuLabel_MouseLeave(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.White;
    }

```



```

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // Получаем выбранный серийный номер
    var selectedSerialNumber = comboBox1.SelectedItem?.ToString();
    // Получаем доступ к базе данных
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        // Открываем соединение
        db.openConnection();

        // Выполняем запрос на получение данных из базы данных
        MySqlCommand command = new MySqlCommand($"SELECT
Equipment_Type, Serial_Number, Begin_Date, End_Date, Processing_time,
Description_all_work, Total_cost, Registered FROM Requests INNER JOIN Repair_Reports
on Repair_Reports.Request_ID = Requests.ID INNER JOIN Clients on
Repair_Reports.Clients_ID = Clients.ID INNER JOIN Request_Monitoring ON
Request_Monitoring.Request_ID = Requests.ID WHERE Serial_Number =
{@selectedSerialNumber} AND Begin_Date = {@selectedBegin_Date}", connection);
        command.Parameters.Add("@selectedSerialNumber",
MySqlDbType.VarChar).Value = selectedSerialNumber;
        command.Parameters.Add("@selectedBegin_Date",
MySqlDbType.Date).Value = dateTimePicker2.Value;
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                // Заполняем поля на форме
                textBox1.Text = reader["Equipment_Type"].ToString();
                textBox2.Text = reader["Serial_Number"].ToString();
            }
        }
    }
}

```

```

dateTimePicker1.Value = (DateTime)reader["Begin_Date"];
textBox3.Text = reader["Processing_time"].ToString();
textBox4.Text = reader["Description_all_work"].ToString();
textBox5.Text = reader["Total_cost"].ToString();
checkBox1.Checked = bool.Parse(reader["Registered"].ToString());

// Проверяем условия для checkBox1
decimal totalCost = decimal.Parse(reader["Total_cost"].ToString());
DateTime beginDate =
DateTime.Parse(reader["Begin_Date"].ToString());
DateTime endDate = DateTime.Parse(reader["End_Date"].ToString());
string descriptionAllWork =
reader["Description_all_work"].ToString();
bool registered = bool.Parse(reader["Registered"].ToString());

if ((totalCost != 0) && (beginDate != DateTime.Today) && (endDate
!= beginDate) && (descriptionAllWork != "") && (registered == true))
{
    checkBox2.Checked = true;
}
else
{
    checkBox2.Checked = false;
}
}

// Закрываем соединение
db.closeConnection();
}

```

```

    }

    private void StatusApplicationForm_Layout(object sender, EventArgs e)
    {
        LoadSerialNumbers(); // Загрузить данные при загрузке формы
    }

    private void LoadSerialNumbers()
    {
        // Получаем доступ к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();

            // Выполняем запрос на получение серийных номеров из базы данных
            MySqlCommand command = new MySqlCommand($"SELECT
Serial_Number FROM Requests INNER JOIN Repair_Reports on
Repair_Reports.Request_ID = Requests.ID INNER JOIN Clients on
Repair_Reports.Clients_ID = Clients.ID WHERE Email = '{LoginForm.loginActive ??
RegisterForm.loginActive}'", connection);

            using (MySqlDataReader reader = command.ExecuteReader())
            {
                // Очищаем ComboBox
                comboBox1.Items.Clear();

                // Добавляем значения в ComboBox из базы данных
                while (reader.Read())
                {

```

```

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

// Закрываем соединение
db.closeConnection();
}
}

```

```

private void ConfirmationButton_Click(object sender, EventArgs e)
{
    if (checkBox2.Checked)
    {
        var selectedSerialNumber = comboBox1.SelectedItem?.ToString();
        string Login = "";
        if (!string.IsNullOrEmpty(LoginForm.loginActive))
        {
            Login = LoginForm.loginActive;
        }
        else if (!string.IsNullOrEmpty(RegisterForm.loginActive))
        {
            Login = RegisterForm.loginActive;
        }
        else
        {
            // Обработка ситуации, если ни одно из значений не возвращено
            MessageBox.Show("Ошибка: Невозможно получить текущий
логин.");

```

```

        return;
    }

    // Создаем новый документ Word
    XWPFDocument doc = new XWPFDocument();

    // Добавляем параграф с заголовком
    XWPFParagraph title = doc.CreateParagraph();
    title.Alignment = ParagraphAlignment.CENTER;
    XWPFRun titleRun = title.CreateRun();
    titleRun.IsBold = true;
    titleRun.FontSize = 14;
    titleRun.SetText("Итоговый отчет");

    // Добавляем таблицу
    XWPFTable table = doc.CreateTable(1, 7);
    string[] headers = { "Дата начала работы", "Дата конца работы",
        "Описание проделанной работы", "Затраченные ресурсы ", "Потребность в запчастях ",
        "Итоговая цена", "Координация с другими специалистами" };

    for (int i = 0; i < headers.Length; i++)
    {
        table.GetRow(0).GetCell(i).SetText(headers[i]);
    }

    // Добавление данных из базы данных
    DB db = new DB();
    using (SqlConnection connection = db.getConnection())
    {
        db.openConnection();
    }

```

```

        MySqlCommand command = new MySqlCommand($"SELECT * FROM
Repair_Reports INNER JOIN Requests on Repair_Reports.Request_ID = Requests.ID
INNER JOIN Clients on Repair_Reports.Clients_ID = Clients.ID WHERE Email = '{Login}'
AND Serial_Number = '{selectedSerialNumber}'", connection);

        MySqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            XWPFTableRow newRow = table.CreateRow();
            for (int i = 0; i < 7; i++)
            {
                newRow.GetCell(i).SetText(reader[i].ToString());
            }
        }

        reader.Close();
        db.closeConnection();
    }

    // Сохраняем документ Word
    string fileName = @"D:\Visual Studio Community 2022\MyProject\Project
PP\Repair_Reports.docx";

    using (FileStream fs = new FileStream(fileName, FileMode.Create,
FileAccess.Write))
    {
        doc.Write(fs);
    }

    MessageBox.Show("Документ Word успешно создан.");

```

```

        // Открываем файл
        try
        {
            ProcessStartInfo startInfo = new ProcessStartInfo
            {
                FileName = fileName,
                UseShellExecute = true
            };
            Process.Start(startInfo);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка при открытии файла: " + ex.Message);
        }
    }
    else
    {
        MessageBox.Show("Заявка еще не выполнена.");
    }
}
}
}

```

## A.17 Листинг файла StatusAppplicationEmployeeForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsApp1
{
    public partial class StatusAppplicationEmployeeForm : Form
    {
        public StatusAppplicationEmployeeForm()
        {
            InitializeComponent();
        }
        /// <summary>
        /// Передвижение окна.
        /// </summary>
        Point lastPoint;
        private void panell1_MouseMove(object sender, MouseEventArgs e)
        {
```



```

        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }

    /// <summary>
    /// Запоминание позиции курсора.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }

    private void CloseButton_Click(object sender, EventArgs e)
    {
        this.Hide();
        LoginForm loginForm = new LoginForm();
        loginForm.Show();
        // Очистим информацию о пользователе
        Authorization.Role = null;
        Authorization.User = null;
        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }

    private void CloseButton_MouseEnter(object sender, EventArgs e)

```

```

{
    CloseButton.ForeColor = Color.Red;
}

```

```

private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}

```

```

private void ToEmployeeMenuLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    EmployeeMenuForm employeeMenu = new EmployeeMenuForm();
    employeeMenu.Show();
}

```

```

private void ToEmployeeMenuLabel_MouseEnter(object sender, EventArgs e)
{
    ToEmployeeMenuLabel.ForeColor = Color.Gray;
}

```

```

private void ToEmployeeMenuLabel_MouseLeave(object sender, EventArgs e)
{
    ToEmployeeMenuLabel.ForeColor = Color.White;
}

```

```

private void StatusApplicationEmployeeForm_Load(object sender, EventArgs

```

e)

```

{
    LoadSerial_Number();
}

```

```

    }

    private void LoadSerial_Number()
    {
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            db.openConnection();
            MySqlCommand command = new MySqlCommand("SELECT
Serial_Number FROM Requests", connection);

            // Очищаем комбобокс перед добавлением новых значений
            comboBox2.Items.Clear();
            comboBox1.Items.Clear();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    comboBox2.Items.Add(reader["Serial_Number"].ToString());
                    comboBox1.Items.Add(reader["Serial_Number"].ToString());
                }
            }

            db.closeConnection();
        }
    }

    private void Add_Click(object sender, EventArgs e)
    {
        if (textBox6.Text != "" && textBox5.Text != "" && textBox3.Text != "")
        {

```

```

try
{
    if (!Regex.IsMatch(textBox6.Text, @"^\d+$"))
    {
        MessageBox.Show("Поле должно содержать только числовые
символы!");

        return;
    }
    if (!Regex.IsMatch(textBox5.Text, @"^\d+$"))
    {
        MessageBox.Show("Поле должно содержать только числовые
символы!");

        return;
    }

    string selectedIDRequests = comboBox2.SelectedItem.ToString();
    string selectedDate = comboBox3.SelectedItem.ToString();
    // Указываем формат даты 'dd.MM.yyyy'
    DateTime dateValue = DateTime.ParseExact(selectedDate,
"dd.MM.yyyy", CultureInfo.InvariantCulture);
    string formattedDate = dateValue.ToString("yyyy-MM-dd");
    // Получаем подключение к базе данных
    DB db = new DB();
    using (SqlConnection connection = db.getConnection())
    {
        db.openConnection();
        // Создаем команду для выполнения запроса INSERT
        MySqlCommand command = new MySqlCommand("INSERT INTO
Work_done (Work_time, Description_work, Expenses, Request_ID, Executor_ID) VALUES
(@Time, @Description_work, @Expenses, @Request_ID, @Executor_ID)", connection);

```

```

        // Получение ID
        MySqlCommand getExecutorIDCommand = new
        MySqlCommand($"SELECT ID FROM Executors WHERE Email =
        '{LoginForm.loginActive}'", connection);

        int executorID =
        (int)Convert.ToInt64(getExecutorIDCommand.ExecuteScalar());

        command.Parameters.Add("@Executor_ID",
        MySqlDbType.Int64).Value = executorID;

        command.Parameters.Add("@Time", MySqlDbType.VarChar).Value
        = textBox6.Text;

        command.Parameters.Add("@Description_work",
        MySqlDbType.VarChar).Value = textBox3.Text;

        command.Parameters.Add("@Expenses",
        MySqlDbType.VarChar).Value = textBox5.Text;

        // Получение ID
        MySqlCommand getRequestIDCommand = new
        MySqlCommand($"SELECT Requests.ID FROM Requests INNER JOIN Repair_Reports
        ON Requests.ID = Repair_Reports.Request_ID WHERE Serial_Number =
        '{selectedIDRequests}' AND Repair_Reports.Begin_Date = '{formattedDate}'", connection);

        int serialID =
        (int)Convert.ToInt64(getRequestIDCommand.ExecuteScalar());

        command.Parameters.Add("@Request_ID",
        MySqlDbType.Int64).Value = serialID;

        command.ExecuteNonQuery();

        // Получение текущей общей стоимости из таблицы Repair_Reports
        MySqlCommand getTotalCostCommand = new
        MySqlCommand($"SELECT Total_cost FROM Repair_Reports INNER JOIN Requests ON

```

```

Requests.ID      =      Repair_Reports.Request_ID      WHERE      Serial_Number      =
'{selectedIDRequests}' AND Begin_Date = '{formattedDate}''', connection);

double currentTotalCost =
Convert.ToDouble(getTotalCostCommand.ExecuteScalar());

// Суммирование текущей общей стоимости с добавленными
расходами

double newTotalCost = currentTotalCost +
Convert.ToDouble(textBox5.Text);

// Обновление общей стоимости в таблице Repair_Reports
MySQLCommand updateTotalCostCommand = new
MySQLCommand($"UPDATE Repair_Reports INNER JOIN Requests ON Requests.ID =
Repair_Reports.Request_ID SET Total_cost = @NewTotalCost WHERE Serial_Number =
'{selectedIDRequests}' AND Begin_Date = '{formattedDate}''', connection);

updateTotalCostCommand.Parameters.AddWithValue("@NewTotalCost", newTotalCost);
updateTotalCostCommand.ExecuteNonQuery();
// Закрываем соединение
db.closeConnection();
MessageBox.Show("Заявка создана.");
}
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка: " + ex.Message);
}
else
{

```

```

        // Не все поля заполнены
        MessageBox.Show("Необходимо заполнить все поля!");
    }
}

private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
    comboBox3.Enabled = true;
    string selectedSerial_Number = comboBox2.SelectedItem.ToString();
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        db.openConnection();
        MySqlCommand command = new MySqlCommand($"SELECT
Begin_Date FROM Requests INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID WHERE Serial_Number = '{selectedSerial_Number}'",
connection);

        comboBox3.Items.Clear();
        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                DateTime beginDate = Convert.ToDateTime(reader["Begin_Date"]);
                comboBox3.Items.Add(beginDate.Date.ToShortDateString());
            }
        }
        db.closeConnection();
    }
}

```

```

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        string selectedSerialNumber = comboBox1.SelectedItem.ToString();
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            db.openConnection();

            MySqlCommand command = new MySqlCommand($"SELECT
Begin_Date FROM Requests INNER JOIN Repair_Reports ON Requests.ID =
Repair_Reports.Request_ID WHERE Serial_Number = '{selectedSerialNumber}'",
connection);

            comboBox4.Items.Clear();
            using (MySqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    DateTime beginDate =
Convert.ToDateTime(reader["Begin_Date"]);
                    comboBox4.Items.Add(beginDate.Date.ToShortDateString());
                }
            }
            db.closeConnection();
        }

        textBox4.Text = "";
        textBox7.Text = "";
        textBox8.Text = "";
        textBox9.Text = "";
    }
}

```



```

        textBox10.Text = "";
        dateTimePicker1.Value = DateTime.Now;
        dateTimePicker2.Value = DateTime.Now;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка данных: " + ex.Message);
    }
}

private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        string selectedSerial_Number = comboBox2.SelectedItem.ToString();
        string selectedDate = comboBox3.SelectedItem.ToString();
        // Указываем формат даты 'dd.MM.yyyy'
        DateTime dateValue = DateTime.ParseExact(selectedDate, "dd.MM.yyyy",
CultureInfo.InvariantCulture);
        string formattedDate = dateValue.ToString("yyyy-MM-dd");

        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            connection.Open();

            string selectQuery = "SELECT * FROM Requests INNER JOIN
Repair_Reports ON Requests.ID = Repair_Reports.Request_ID WHERE Serial_Number =
@"selectedSerial_Number AND Repair_Reports.Begin_Date = @"Date";

```

```

        MySqlCommand command = new MySqlCommand(selectQuery,
connection);

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

        command.Parameters.AddWithValue("@Date", formattedDate);

        using (MySqlDataReader reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                textBox3.Enabled = true;
                textBox5.Enabled = true;
                textBox6.Enabled = true;
            }
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка данных: " + ex.Message);
}
}

private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        string selectedSerial_Number = comboBox1.SelectedItem.ToString();
        string selectedDate = comboBox4.SelectedItem.ToString();
        // Указываем формат даты 'dd.MM.yyyy'

```

```

        DateTime dateValue = DateTime.ParseExact(selectedDate, "dd.MM.yyyy",
CultureInfo.InvariantCulture);

        string formattedDate = dateValue.ToString("yyyy-MM-dd");

        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();

            string selectQuery = "SELECT Begin_Date, End_Date,
Description_all_work, Used_resources, Spare_parts_needed, TRUNCATE(Total_cost, 0) as
Total_cost, Coordination_with_other_specialists FROM Repair_Reports INNER JOIN
Requests ON Requests.ID = Repair_Reports.Request_ID Where Serial_Number =
@selectedSerial_Number AND Repair_Reports.Begin_Date = @Date";

            MySqlCommand command = new MySqlCommand(selectQuery,
connection);

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

            command.Parameters.Add("@Date", MySqlDbType.VarChar).Value =
formattedDate;

            using (MySqlDataReader reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    textBox7.Text = reader["Total_cost"].ToString();
                    textBox9.Text = reader["Used_resources"].ToString();
                    textBox8.Text = reader["Spare_parts_needed"].ToString();
                }
            }
        }
    }
}

```

```

        textBox10.Text
reader["Coordination_with_other_specialists"].ToString();
        textBox4.Text = reader["Description_all_work"].ToString();
        dateTimePicker1.Value = (DateTime)reader["Begin_Date"];
        dateTimePicker2.Value = (DateTime)reader["End_Date"];
    }
}
connection.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка данных: " + ex.Message);
}
}

```

```

private void Edit_Click(object sender, EventArgs e)
{
    // Сделать все текстовые поля доступными для редактирования
    textBox4.Enabled = true;
    textBox7.Enabled = true;
    textBox8.Enabled = true;
    textBox9.Enabled = true;
    textBox10.Enabled = true;
    dateTimePicker1.Enabled = true;
    dateTimePicker2.Enabled = true;
    ConfirmationButton.Visible = true;
}
private void ConfirmationButton_Click(object sender, EventArgs e)
{

```

```

try
{
    // Проверка введенных данных и их сохранение в базе
    string description = textBox4.Text;
    string totalCost = textBox7.Text;
    string sparePartsNeeded = textBox8.Text;
    string usedResources = textBox9.Text;
    string coordination = textBox10.Text;
    DateTime beginDate = dateTimePicker1.Value;
    DateTime endDate = dateTimePicker2.Value;
    string formBeginDate = beginDate.ToString("yyyy-MM-dd");
    string formEndDate = beginDate.ToString("yyyy-MM-dd");
    if (endDate < beginDate)
    {
        MessageBox.Show("Дата окончания не может быть раньше даты
начала!");
        return;
    }
    if (!Regex.IsMatch(textBox7.Text, @"^\d+(\.\d+)?$"))
    {
        MessageBox.Show("Поле должно содержать только числовые
символы!");
        return;
    }
    DB db = new DB();
    using (SqlConnection connection = db.getConnection())
    {
        connection.Open();
        // Обновление данных в базе данных
        string updateQuery = $"UPDATE Repair_Reports " +

```

```

        $"SET Begin_Date = '{formBeginDate}', " +
        $"End_Date = '{formEndDate}', " +
        $"Description_all_work = '{description}', " +
        $"Used_resources = '{usedResources}', " +
        $"Spare_parts_needed = '{sparePartsNeeded}', " +
        $"Total_cost = '{totalCost}', " +
        $"Coordination_with_other_specialists = '{coordination}' " +
        $"WHERE Executor_ID = @Executor_ID AND Request_ID =
@Request_ID ";

        MySqlCommand command = new MySqlCommand(updateQuery,
connection);

        // Получение ID
        MySqlCommand getExecutorIDCommand = new
        MySqlCommand($"SELECT ID FROM Executors WHERE Email =
'{LoginForm.loginActive}'", connection);

        int executorID =
(int)Convert.ToInt64(getExecutorIDCommand.ExecuteScalar());

        command.Parameters.Add("@Executor_ID",
        MySqlDbType.Int64).Value = executorID;

        string selectedIDRequests = comboBox1.SelectedItem.ToString();
        string selectedDate = comboBox4.SelectedItem.ToString();
        // Указываем формат даты 'dd.MM.yyyy'
        DateTime dateValue = DateTime.ParseExact(selectedDate,
        "dd.MM.yyyy", CultureInfo.InvariantCulture);

        string formattedDate = dateValue.ToString("yyyy-MM-dd");
        // Получение ID
        MySqlCommand getRequestIDCommand = new
        MySqlCommand($"SELECT Requests.ID FROM Requests INNER JOIN Repair_Reports
ON Requests.ID = Repair_Reports.Request_ID WHERE Serial_Number =
'{selectedIDRequests}' AND Repair_Reports.Begin_Date = '{formattedDate}'", connection);

```

```

        int serialID =
(int)Convert.ToInt64(getRequestIDCommand.ExecuteScalar());
        command.Parameters.Add("@Request_ID", MySqlDbType.Int64).Value
= serialID;

        command.ExecuteNonQuery();
        connection.Close();
        // Сделать текстовые поля недоступными для редактирования
        textBox4.Enabled = false;
        textBox7.Enabled = false;
        textBox8.Enabled = false;
        textBox9.Enabled = false;
        textBox10.Enabled = false;
        dateTimePicker1.Enabled = false;
        dateTimePicker2.Enabled = false;
        ConfirmationButton.Visible = false;
        MessageBox.Show("Итоговый отчет изменён!");
    }
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка данных: " + ex.Message);
}
}
}
}

```

## A.18 Листинг файла UsersMenuForm.cs

```
using MySql.Data.MySqlClient;

namespace WinFormsApp1
{
    public partial class UsersMenuForm : Form
    {
        public static MySqlConnection con = new MySqlConnection("user=root;
host=localhost; password=IvanVladimirov888; database=Computer_systems;");
        public static UsersMenuForm? Instance { get; set; }
        public static EditApplicationAdminForm? Instance2 = null;
        public UsersMenuForm()
        {
            InitializeComponent();
            //con.Open();
            Instance = this;
        }

        private void UsersMenuForm_FormClosed(object sender,
FormClosedEventArgs e)
        {
            //con.Close();
        }

        private void CreateApplication_Click(object sender, EventArgs e)
        {
            this.Hide();
            ApplicationForm application = new ApplicationForm();
        }
    }
}
```



```
        application.Show();  
    }
```

```
private void StatusApplication_Click(object sender, EventArgs e)  
{  
    this.Hide();  
    StatusApplicationForm statusApplication = new StatusApplicationForm();  
    statusApplication.Show();  
}
```

```
private void EditApplication_Click(object sender, EventArgs e)  
{  
    this.Hide();  
    EditApplicationForm editApplication = new EditApplicationForm();  
    editApplication.Show();  
}
```

```
private void UsersProfile_Click(object sender, EventArgs e)  
{  
    this.Hide();  
    UsersProfileForm usersProfileForm = new UsersProfileForm();  
    usersProfileForm.Show();  
}
```

```
private void CloseButton_Click(object sender, EventArgs e)  
{  
    this.Hide();  
    LoginForm loginForm = new LoginForm();  
    loginForm.Show();  
    // Очистим информацию о пользователе
```

```

        Authorization.Role = null;
        Authorization.User = null;
        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }
    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Red;
    }
    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.White;
    }
    private void ToLoginLabel_Click(object sender, EventArgs e)
    {
        this.Hide();
        LoginForm loginForm = new LoginForm();
        loginForm.Show();
        // Очистим информацию о пользователе
        Authorization.Role = null;
        Authorization.User = null;
        Authorization.FIO = null;
        LoginForm.loginActive = "";
        RegisterForm.loginActive = "";
    }
    private void ToLoginLabel_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Gray;
    }
}

```

```

private void ToLoginLabel_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}
/// <summary>
/// Передвижение окна.
/// </summary>
Point lastPoint;
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
}
}

```

## A.19 Листинг файла UsersProfileAdminForm.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsApp1
{
    public partial class UsersProfileAdminForm : Form
    {
        public UsersProfileAdminForm()
        {
            InitializeComponent();

            Point lastPoint;
            private int selectedClientId;

            private void panell1_MouseMove(object sender, MouseEventArgs e)
            {
                if (e.Button == MouseButtons.Left)
```

```

    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}

/// <summary>
/// Запоминание позиции курсора.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}

private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
}

private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}

```

```

}

private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}

private void ToAdminsMenuLabel_Click(object sender, EventArgs e)
{
    this.Hide();
    AdminMenuForm adminMenu = new AdminMenuForm();
    adminMenu.Show();
}

private void ToAdminsMenuLabel_MouseEnter(object sender, EventArgs e)
{
    ToAdminsMenuLabel.ForeColor = Color.Gray;
}

private void ToAdminsMenuLabel_MouseLeave(object sender, EventArgs e)
{
    ToAdminsMenuLabel.ForeColor = Color.White;
}

private void LoadFIO()
{
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        db.openConnection();
    }
}

```

```

        MySqlCommand command = new MySqlCommand("SELECT FIO FROM
Clients", connection);

        comboBox1.Items.Clear(); // Очищаем комбобокс перед добавлением новых
значений

        using (MySqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                comboBox1.Items.Add(reader["FIO"].ToString());
            }
        }

        db.closeConnection();
    }
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        string selectedFIO = comboBox1.SelectedItem.ToString();

        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();

```

```
string selectQuery = "SELECT FIO, Passport, Address, Phone, Email FROM  
Clients WHERE FIO = @selectedFIO";
```

```
MySQLCommand command = new MySQLCommand(selectQuery, connection);  
command.Parameters.AddWithValue("@selectedFIO", selectedFIO);
```

```
using (MySQLDataReader reader = command.ExecuteReader())  
{  
    if (reader.Read())  
    {  
        textBox1.Text = reader["FIO"].ToString();  
        textBox2.Text = reader["Passport"].ToString();  
        textBox3.Text = reader["Address"].ToString();  
        textBox4.Text = reader["Phone"].ToString();  
        textBox5.Text = reader["Email"].ToString();  
        textBox6.Text = "*****";  
    }  
}  
connection.Close();  
}  
}  
catch (Exception ex)  
{  
    MessageBox.Show("Ошибка данных: " + ex.Message);  
}  
}
```

```
private void EditProfile_Click(object sender, EventArgs e)  
{  
    // Сделать все текстовые поля доступными для редактирования
```



```

        textBox1.Enabled = true;
        textBox2.Enabled = true;
        textBox3.Enabled = true;
        textBox4.Enabled = true;
        textBox5.Enabled = true;
        textBox6.Enabled = true;
        ConfirmationButton.Visible = true;
    }

    private string HashPassword(string password)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(password);
            byte[] hash = sha256.ComputeHash(bytes);
            StringBuilder result = new StringBuilder();

            for (int i = 0; i < hash.Length; i++)
            {
                result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX строки
            }

            return result.ToString();
        }
    }

    // Обработчик кнопки ConfirmationButton
    private void ConfirmationButton_Click(object sender, EventArgs e)
    {
        // Проверка введенных данных и их сохранение в базе

```

```

string fio = textBox1.Text;
string passport = textBox2.Text;
string address = textBox3.Text;
string phone = textBox4.Text;
// Дополнительно получаем введенные логин и пароль
string login = textBox5.Text;
string password = HashPassword(textBox6.Text);
string password1 = textBox6.Text;
// Проверка формата введенных данных
if (!IsValidFIO(fio))
{
    MessageBox.Show("Фамилия должна содержать только буквы");
    return;
}
if (!IsValidPhone(phone))
{
    MessageBox.Show("Номер телефона должен содержать ровно 11 цифр");
    return;
}
if (!IsValidPassport(passport))
{
    MessageBox.Show("Паспорт должен содержать ровно 10 цифр");
    return;
}
if (!IsValidPassword(password1))
{
    MessageBox.Show("Пароль должен содержать от 4 до 16 символов");
    return;
}
// Проверка на использование определенных символов

```

```

string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№", "\"", "'",
"#", ";", ":",
"%", "&", ",", "?", "~", "!", "=", "\\", "/" };
if (restrictedSymbols.Any(symbol => textBox5.Text.Contains(symbol)))
{
    MessageBox.Show("Поля не должны содержать запрещенные символы: " +
string.Join(" ", restrictedSymbols));
    return; // Прерываем выполнение метода
}
if (restrictedSymbols.Any(symbol => textBox6.Text.Contains(symbol)))
{
    MessageBox.Show("Поля не должны содержать запрещенные символы: " +
string.Join(" ", restrictedSymbols));
    return; // Прерываем выполнение метода
}
// Получение введенных логина и пароля
string newLogin = textBox5.Text;
string newPassword = HashPassword(textBox6.Text);
string oldPasswordFromDB;
// Получение старого логина из глобальной переменной или другого места, где
он хранится
string selectedFIO = comboBox1.SelectedItem.ToString();
string oldLogin = GetLoginFromDB(selectedFIO); // Примерное название
переменной, где хранится текущий логин
// Получение старого пароля из базы данных по старому логину
string oldPasswordFromDBQuery = $"SELECT Password FROM Clients Where
Email = '{oldLogin}'";
DB db = new DB();
using (SqlConnection connection = db.getConnection())
{

```

```

        // Открываем соединение
        connection.Open();
        using (MySQLCommand oldPasswordFromDBCommand = new
MySQLCommand(oldPasswordFromDBQuery, connection))
        {
            oldPasswordFromDB =
(string)oldPasswordFromDBCommand.ExecuteScalar();
        }
        // Закрываем соединение
        connection.Close();
    }
    if (newLogin != oldLogin || newPassword != oldPasswordFromDB)
    {
        // Выводим окно с подтверждением изменения логина и пароля
        DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите
изменить логин и пароль?", "Подтверждение изменений", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            // Обновляем логин и пароль в базе данных
            string updateLoginQuery = $"UPDATE Clients SET Email = '{newLogin}'
WHERE Email = '{oldLogin}'";
            string updatePasswordQuery = $"UPDATE Clients SET Password =
'{newPassword}' WHERE Email = '{newLogin}'";
            using (MySQLConnection connection = db.getConnection())
            {
                // Открываем соединение
                connection.Open();
                using (MySQLCommand updateLoginCommand = new
MySQLCommand(updateLoginQuery, connection))

```

```

        using (MySqlCommand updatePasswordCommand = new
MySqlCommand(updatePasswordQuery, connection))
        {
            updateLoginCommand.ExecuteNonQuery();
            updatePasswordCommand.ExecuteNonQuery();
        }
        // Закрываем соединение
        connection.Close();
    }
    // Обновление данных в базе данных
    string updateQuery = $"UPDATE Clients SET FIO = '{fio}', Passport =
{passport}, Address = '{address}', Phone = '{phone}' WHERE Email = '{newLogin}';";
    using (MySqlConnection connection = db.getConnection())
    {
        // Открываем соединение
        connection.Open();
        using (MySqlCommand updateQueryCommand = new
MySqlCommand(updateQuery, connection))
        {
            updateQueryCommand.ExecuteNonQuery();
        }
        // Закрываем соединение
        connection.Close();
    }
    // Сделать текстовые поля недоступными для редактирования
    textBox1.Enabled = false;
    textBox2.Enabled = false;
    textBox3.Enabled = false;
    textBox4.Enabled = false;
    textBox5.Enabled = false;

```

```

        textBox6.Enabled = false;
        ConfirmationButton.Visible = false;
    }
    else
    {
        // Блокируем поля логина и пароля и оставляем старые данные
        textBox5.Text = oldLogin;
        textBox6.Text = oldPasswordFromDB; // Допустим, что мы не обновляем
        пароль, если изменение не подтверждено

        // Обновление данных в базе данных
        string updateQuery = $"UPDATE Clients SET FIO = '{fio}', Passport =
        {passport}, Address = '{address}', Phone = '{phone}' WHERE Email = '{oldLogin}';";
        using (MySQLConnection connection = db.getConnection())
        {
            connection.Open();
            using (MySQLCommand updateQueryCommand = new
            MySQLCommand(updateQuery, connection))
            {
                updateQueryCommand.ExecuteNonQuery();
            }
            connection.Close();
        }
        // Сделать текстовые поля недоступными для редактирования
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        textBox3.Enabled = false;
        textBox4.Enabled = false;
        textBox5.Enabled = false;
        textBox6.Enabled = false;
        ConfirmationButton.Visible = false;
    }

```

```

    }

}

textBox6.Text = "*****";
textBox6.PasswordChar = '*';
textBox6.UseSystemPasswordChar = true;
}

// Метод для проверки формата ФИО
private bool IsValidFIO(string fio)
{
    // Проверяем, что строка состоит только из букв и пробелов
    return !string.IsNullOrEmpty(fio) && fio.All(c => char.IsLetter(c) ||
char.IsWhiteSpace(c));
}

// Метод для проверки формата паспортных данных
private bool IsValidPassport(string passport)
{
    // Проверяем, что строка состоит из 10 цифр
    return !string.IsNullOrEmpty(passport) && passport.Length == 10 &&
passport.All(c => char.IsDigit(c));
}

// Метод для проверки формата номера телефона
private bool IsValidPhone(string phone)
{
    // Проверяем, что строка состоит из 11 цифр
    return !string.IsNullOrEmpty(phone) && phone.Length == 11 && phone.All(c
=> char.IsDigit(c));
}

private bool IsValidPassword(string password)

```

```

    {
        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(password) && password.Length == 16 &&
password.Length > 4;
    }
    private string GetLoginFromDB(string selectedFIO)
    {
        string login = "";
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            db.openConnection();
            MySqlCommand command = new MySqlCommand("SELECT Email FROM
Clients WHERE FIO = @selectedFIO", connection);
            command.Parameters.Add("@selectedFIO", MySqlDbType.VarChar).Value =
selectedFIO;
            object result = command.ExecuteScalar();
            if (result != null)
            {
                login = result.ToString(); // Присваиваем значение логина
            }

            db.closeConnection();
        }
        return login;
    }
    // Ваш обработчик события загрузки формы UsersProfileAdminForm_Load с
использованием новой функции
    private void UsersProfileAdminForm_Load(object sender, EventArgs e)
    {

```



```

        LoadFIO();
    }
    private void DeleteButton_Click(object sender, EventArgs e)
    {
        string selectedFIO = comboBox1.SelectedItem.ToString();
        string oldLogin = GetLoginFromDB(selectedFIO);
        // Подключение к базе данных
        DB db = new DB();
        using (MySqlConnection connection = db.getConnection())
        {
            connection.Open();
            MySqlCommand command = new MySqlCommand("DELETE FROM Clients
WHERE Email = @Login;", connection);
            command.Parameters.AddWithValue("@Login", oldLogin);
            command.ExecuteNonQuery();
            connection.Close();
        }
        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        textBox5.Text = "";
        textBox6.Text = "";
        comboBox1.Text = "";
        LoadFIO();
    }
}
}

```

## A.20 Листинг файла UsersProfileForm.cs

```
using Microsoft.VisualBasic.Logging;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WinFormsApp1
{
    public partial class UsersProfileForm : Form
    {
        MySqlCommand cmd = UsersMenuForm.con.CreateCommand();
        String fioGlobal = "";
        public UsersProfileForm()
        {
            InitializeComponent();
        }
        private void EditProfile_Click(object sender, EventArgs e)
        {
            // Сделать все текстовые поля доступными для редактирования
            textBox1.Enabled = true;
            textBox2.Enabled = true;
            textBox3.Enabled = true;
        }
    }
}
```

```

        textBox4.Enabled = true;
        textBox5.Enabled = true;
        textBox6.Enabled = true;
        ConfirmationButton.Visible = true;
    }
    private string HashPassword(string password)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(password);
            byte[] hash = sha256.ComputeHash(bytes);
            StringBuilder result = new StringBuilder();
            for (int i = 0; i < hash.Length; i++)
            {
                result.Append(hash[i].ToString("x2")); // Преобразуем байты в HEX
            }
            return result.ToString();
        }
    }
    // Обработчик кнопки ConfirmationButton
    private void ConfirmationButton_Click(object sender, EventArgs e)
    {
        // Проверка введенных данных и их сохранение в базе
        string fio = textBox1.Text;
        string passport = textBox2.Text;
        string address = textBox3.Text;
        string phone = textBox4.Text;
        // Дополнительно получаем введенные логин и пароль
        string login = textBox5.Text;

```

строки

```

string password = HashPassword(textBox6.Text);
string password1 = textBox6.Text;
// Проверка формата введенных данных
if (!IsValidFIO(fio))
{
    MessageBox.Show("Фамилия должна содержать только буквы");
    return;
}
if (!IsValidPhone(phone))
{
    MessageBox.Show("Номер телефона должен содержать ровно 11
цифр");
    return;
}
if (!IsValidPassport(passport))
{
    MessageBox.Show("Паспорт должен содержать ровно 10 цифр");
    return;
}
if (!IsValidPassword(password1))
{
    MessageBox.Show("Пароль должен содержать от 4 до 16 символов");
    return;
}
// Проверка на использование определенных символов
string[] restrictedSymbols = { "*", "&", "{", "}", "|", "+", "-", "(", ")", "№",
"\", \"", "#", ";", ":",
"%", "&", ",", "?", "~", "!", "=", "\\", "/" };
Func<string, bool> predicate = symbol => textBox5.Text.Contains(symbol);
if (restrictedSymbols.Any(predicate))

```

```

    {
        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода
    }
    if (restrictedSymbols.Any(symbol => textBox6.Text.Contains(symbol)))
    {
        MessageBox.Show("Поля не должны содержать запрещенные символы:
" + string.Join(" ", restrictedSymbols));
        return; // Прерываем выполнение метода
    }
    // Получение введенных логина и пароля
    string newLogin = textBox5.Text;
    string newPassword = HashPassword(textBox6.Text);
    string oldPasswordFromDB;
    // Получение старого логина из глобальной переменной или другого
    места, где он хранится
    string oldLogin = LoginForm.loginActive ?? RegisterForm.loginActive; //
    Примерное название переменной, где хранится текущий логин
    // Получение старого пароля из базы данных по старому логину
    string oldPasswordFromDBQuery = $"SELECT Password FROM Clients
Where Email = '{oldLogin}'";
    DB db = new DB();
    using (MySqlConnection connection = db.getConnection())
    {
        // Открываем соединение
        connection.Open();
        using (MySqlCommand oldPasswordFromDBCommand = new
MySqlCommand(oldPasswordFromDBQuery, connection))
        {

```

```

oldPasswordFromDB
=
(string)oldPasswordFromDBCommand.ExecuteScalar();
    }
    // Закрываем соединение
    connection.Close();
}
if (newLogin != LoginForm.loginActive || newPassword !=
oldPasswordFromDB)
{
    // Выводим окно с подтверждением изменения логина и пароля
    DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите
изменить логин и пароль?", "Подтверждение изменений", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        // Обновляем логин и пароль в базе данных
        string updateLoginQuery = $"UPDATE Clients SET Email =
'{newLogin}' WHERE Email = '{oldLogin}'";
        string updatePasswordQuery = $"UPDATE Clients SET Password =
'{newPassword}' WHERE Email = '{newLogin}'";
        using (MySQLConnection connection = db.getConnection())
        {
            // Открываем соединение
            connection.Open();
            using (MySQLCommand updateLoginCommand = new
MySQLCommand(updateLoginQuery, connection))
            using (MySQLCommand updatePasswordCommand = new
MySQLCommand(updatePasswordQuery, connection))
            {
                updateLoginCommand.ExecuteNonQuery();
                updatePasswordCommand.ExecuteNonQuery();
            }
        }
    }
}

```

```

    }
    // Закрываем соединение
    connection.Close();
}
// Обновление данных в базе данных
string updateQuery = $"UPDATE Clients SET FIO = '{fio}', Passport =
{passport}, Address = '{address}', Phone = '{phone}' WHERE Email = '{newLogin}';";
using (MySqlConnection connection = db.getConnection())
{
    // Открываем соединение
    connection.Open();
    using (MySqlCommand updateQueryCommand = new
MySqlCommand(updateQuery, connection))
    {
        updateQueryCommand.ExecuteNonQuery();
    }
    // Закрываем соединение
    connection.Close();
}
// Сделать текстовые поля недоступными для редактирования
textBox1.Enabled = false;
textBox2.Enabled = false;
textBox3.Enabled = false;
textBox4.Enabled = false;
textBox5.Enabled = false;
textBox6.Enabled = false;
ConfirmationButton.Visible = false;
}
else
{

```

```

// Блокируем поля логина и пароля и оставляем старые данные
textBox5.Text = oldLogin;
textBox6.Text = oldPasswordFromDB; // Допустим, что мы не
обновляем пароль, если изменение не подтверждено

// Обновление данных в базе данных
string updateQuery = $"UPDATE Clients SET FIO = '{fio}', Passport =
{passport}, Address = '{address}', Phone = '{phone}' WHERE Email = '{oldLogin}';";
using (SqlConnection connection = db.getConnection())
{
    connection.Open();
    using (SqlCommand updateQueryCommand = new
SqlCommand(updateQuery, connection))
    {
        updateQueryCommand.ExecuteNonQuery();
    }
    connection.Close();
}
// Сделать текстовые поля недоступными для редактирования
textBox1.Enabled = false;
textBox2.Enabled = false;
textBox3.Enabled = false;
textBox4.Enabled = false;
textBox5.Enabled = false;
textBox6.Enabled = false;
ConfirmationButton.Visible = false;
}
}
textBox6.Text = "*****";
textBox6.PasswordChar = '*';
textBox6.UseSystemPasswordChar = true;

```



```

    }
    // Метод для проверки формата ФИО
    private bool IsValidFIO(string fio)
    {
        // Проверяем, что строка состоит только из букв и пробелов
        return !string.IsNullOrEmpty(fio) && fio.All(c => char.IsLetter(c) ||
char.IsWhiteSpace(c));
    }
    // Метод для проверки формата паспортных данных
    private bool IsValidPassport(string passport)
    {
        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(passport) && passport.Length == 10 &&
passport.All(c => char.IsDigit(c));
    }
    // Метод для проверки формата номера телефона
    private bool IsValidPhone(string phone)
    {
        // Проверяем, что строка состоит из 11 цифр
        return !string.IsNullOrEmpty(phone) && phone.Length == 11 &&
phone.All(c => char.IsDigit(c));
    }
    private bool IsValidPassword(string password)
    {
        // Проверяем, что строка состоит из 10 цифр
        return !string.IsNullOrEmpty(password) && password.Length == 16 &&
password.Length > 4;
    }
    private void UsersProfileForm_Load(object sender, EventArgs e)
    {

```

```

        // Получаем доступ к базе данных
        DB db = new DB();
        using (SqlConnection connection = db.getConnection())
        {
            // Открываем соединение
            db.openConnection();
            string oldLogin = "";
            if (!string.IsNullOrEmpty(LoginForm.loginActive))
            {
                oldLogin = LoginForm.loginActive;
            }
            else if (!string.IsNullOrEmpty(RegisterForm.loginActive))
            {
                oldLogin = RegisterForm.loginActive;
            }
            else
            {
                // Обработка ситуации, если ни одно из значений не возвращено
                MessageBox.Show("Ошибка: Невозможно получить текущий логин.");
                return;
            }

            // Подготавливаем SQL-запрос
            string selectQuery = "SELECT FIO, Passport, Address, Phone, Password,
Email FROM Clients WHERE Email = @Email";

            MySqlCommand command = new MySqlCommand(selectQuery,
connection);

            command.Parameters.AddWithValue("@Email", oldLogin);
            // Используем читатели для получения данных из базы
            using (MySqlDataReader reader = command.ExecuteReader())
            {

```

```

        if (reader.Read())
        {
            // Заполняем поля на форме полученными данными
            textBox1.Text = reader["FIO"].ToString();
            textBox2.Text = reader["Passport"].ToString();
            textBox3.Text = reader["Address"].ToString();
            textBox4.Text = reader["Phone"].ToString();
            textBox5.Text = reader["Email"].ToString();
            // В данном случае пароль получаем в открытом виде
            textBox6.Text = "*****";
            // Преобразуем хешированный пароль в нормальные символы

        }
    }
    // Закрываем соединение
    db.closeConnection();
}
}

```

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

```

{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}

```

/// <summary>

/// Запоминание позиции курсора.

```

/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
private void CloseButton_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    // Очистим информацию о пользователе
    Authorization.Role = null;
    Authorization.User = null;
    Authorization.FIO = null;
    LoginForm.loginActive = "";
    RegisterForm.loginActive = "";
}
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Red;
}
private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.White;
}
private void ToUsersMenuLabel_Click(object sender, EventArgs e)
{
    this.Hide();
}

```

```

        UsersMenuForm usersMenu = new UsersMenuForm();
        usersMenu.Show();
    }
    private void ToUsersMenuLabel_MouseEnter(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.Gray;
    }
    private void ToUsersMenuLabel_MouseLeave(object sender, EventArgs e)
    {
        ToUsersMenuLabel.ForeColor = Color.White;
    }
}
}

```

**ПРИЛОЖЕНИЕ Б**  
**(обязательное)**  
**Результаты выполнения программы**

На рисунке Б.1 представлен итоговый отчет.

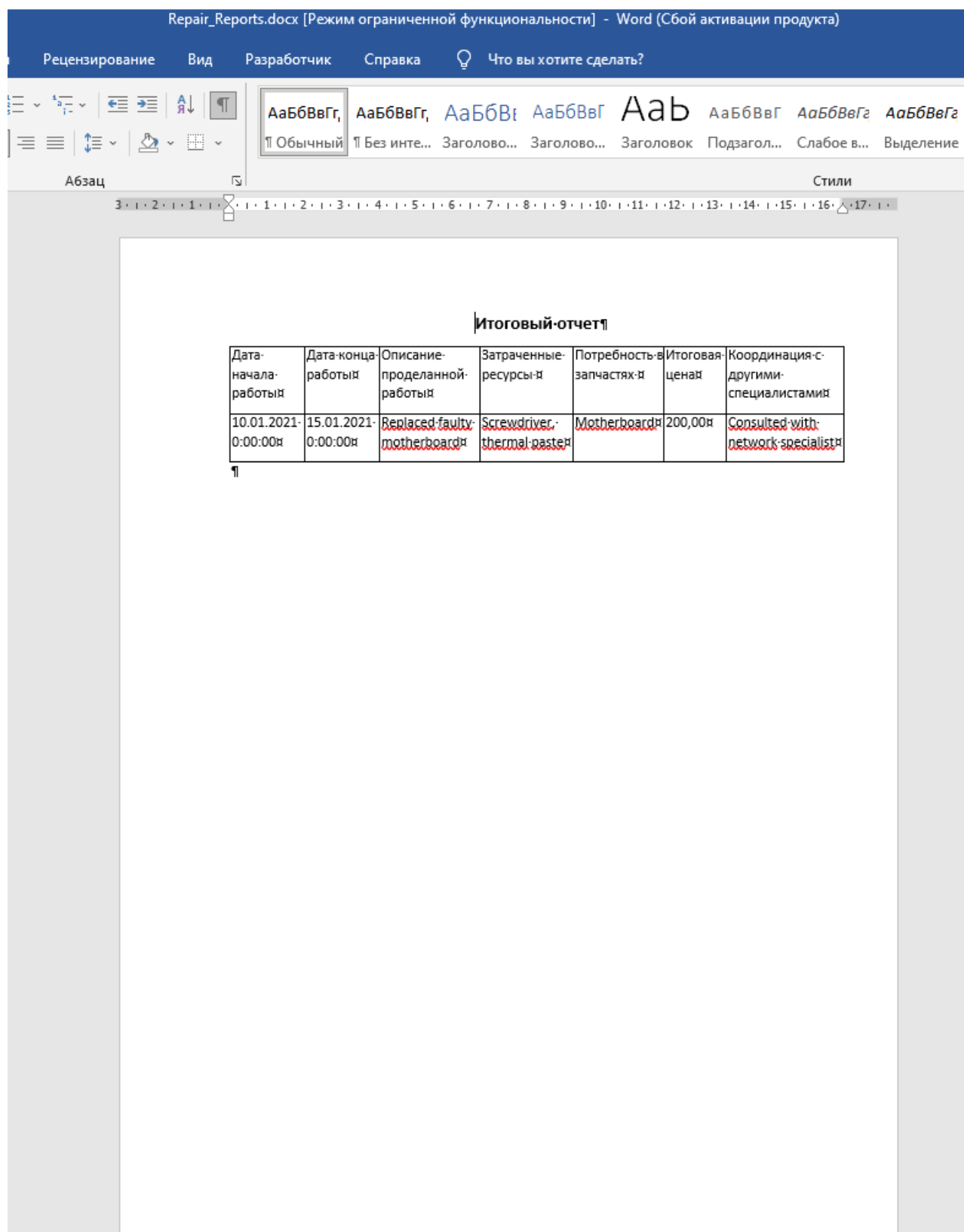


Рисунок Б.1 –Итоговый отчет

# Дневник

прохождения учебной практики УП02

по профессиональному модулю ПМ.02 Осуществление интеграции программных модулей

Код, специальность, группа : 09.02.07 Информационные системы и программирование

Квалификация - Программист, ТИП 62

Студента(ки)

Владимирова Илона Сергеевна

(Ф.И.О.)

Дата	Тема занятия	Объем выполненной работы	Отметка о выполнении руководителя практики
12.01.24	Анализ предметной области	6	Выполнено
13.01.24	Разработка и оформление технического задания	6	Выполнено
15.01.24	Проектирование информационной системы	6	Выполнено
16.01.24	Конструирование прототипа программы.	6	Выполнено
17.01.24	Разработка интерфейса программы.	6	Выполнено
18.01.24	Разработка функциональной части программы	6	Выполнено
19.01.24	Отладка программы с использованием специализированных средств отладки.	6	Выполнено
20.01.24	Интеграция модулей в программную систему	6	Выполнено
22.01.24-23.01.24	Тестирование программной системы	12	Выполнено
24.01.24	Разработка документации для программной системы	6	Выполнено
25.01.24	Оформление презентации	6	Выполнено
26.01.24-27.01.24	Оформление отчета	12	Выполнено
29.01.24	Зачет	6	

Итоговая оценка отлично

Руководитель практики от предприятия

(подпись)

(Ф.И.О.)



**Аттестационный лист  
по учебной практике**

студента (ки) Московского техникума космического приборостроения МГТУ им. Н.Э. Баумана  
Владимирова Ивана Сергеевича  
(Ф.И.О. студента)

Группа ТИП-62

Специальность 09.02.07 Информационные системы и программирование Квалификация - Программист

(код, наименование специальности)

прошел (ла) учебную практику ПП 02 ПМ.02 Осуществление интеграции программных модулей

(наименование практики)

по профессиональному модулю ПМ.02 Осуществление интеграции программных модулей в объеме 90 часов

(наименование профессионального модуля)

с « 30 » января 2024 года по « 15 » февраля 2024 года  
на предприятии (организации) МТК МГТУ им. Н.Э. Баумана, Москва, Волховский пер., д. 11.  
(юридический адрес предприятия (организации))

**Виды и качество работ в период учебной практики**

Виды работ, выполненные студентом во время практики, согласно программы учебной практики	Результат (по 5-ти бальной шкале)
Осуществление интеграции программных модулей и соответствующими ему компетенциями	<u>отлично</u>

**В ходе учебной практики студентом освоены следующие профессиональные компетенции**

Код и название профессиональной компетенции	Результат освоения (освоена/не освоена)
ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.	<u>освоена</u>
ПК 2.2. Выполнять интеграцию модулей в программное обеспечение	<u>освоена</u>
ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств	<u>освоена</u>
ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.	<u>освоена</u>
ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.	<u>освоена</u>

Характеристика профессиональной деятельности обучающегося во время учебной практики: задание выполнено в срок, в полной мере.

Рекомендуемая оценка по практике отлично

Руководитель практики от предприятия (организации)

Должность

Подпись

Итоговая оценка по практике отлично

Руководитель практики от образовательного учреждения

Должность

Подпись

М.П.

Учебная  
часть

Ф.И.О. руководителя практики

Ф.И.О. руководителя практики