Министерство образования и науки Российской Федерации

Муромский институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет

Имени Александра Григорьевича и Николая Григорьевича Столетовых»

(МИ ВлГУ)

Факультет ИТР

Кафедра: ФПМ

КУРСОВАЯ РАБОТА

По дисциплине Базы данных

Тема АИС система медицинской лаборатории

Руководитель

Колпаков А.А.

(оценка)  (фамилия, инициалы)

(подпись) (дата)

Члены комиссии: Студент

(группа)

(подпись) (Ф.И.О.) (фамилия, инициалы)

(подпись) (Ф.И.О.) (подпись) (дата)

Муром 2024г.

Содержание.

Введение………………………………………………..…………………….....5

1. Анализ технического задания……………………………………………....7

1.1. Описание предметной области…………………………………………...7

1.2. Анализ технического задания…………………………………………….7

2. Разработка моделей данных ………………………………………………..9

2.1. Разработка концептуальной модели………………………………….…..9

2.2. Разработка логической модели…………………………...……………....9

2.3. Разработка физической модели…………………………………………..10

3. Разработка и реализация АИС………………………………………..…….13

3.1. Разработка серверной части……………………………………………....13

3.2. Разработка клиентской части…………………………………..………....20

3.3. Руководство пользователю и администратору………………………......26

Заключение……………………………………….…………………………......38

Список используемой литературы…………………………………………….39

Приложение...........................................................................................................40

Введение

Система управления базами данных (СУБД) — это программное обеспечение, предназначенное для создания и управления базами данных. Она обеспечивает доступ к данным, их хранение, обновление, поиск и анализ, а также защиту информации и контроль доступа к ней.

База данных представляет собой организованную коллекцию данных, которая обеспечивает удобный доступ, управление и обновление информации. Это структурированный набор данных, хранящийся в компьютерной системе и пригодный для обработки с целью получения необходимой информации.

Создание приложений для работы с базами данных имеет огромное значение в современном мире информационных технологий. Такие приложения позволяют пользователям эффективно управлять и обрабатывать большие объемы данных, что особенно актуально в условиях современного информационного общества.

Во-первых, приложения для работы с базами данных обеспечивают быстрый и удобный доступ к информации. Это позволяет пользователям быстро находить нужные данные, проводить анализ и принимать обоснованные решения на основе полученной информации.

Во-вторых, такие приложения гарантируют целостность и безопасность данных. Они позволяют контролировать доступ к информации, выполнять резервное копирование, защищать данные от несанкционированного доступа и повреждений.

Кроме того, приложения для работы с базами данных помогают упростить процессы обработки информации, автоматизировать рутинные операции, повысить эффективность работы с данными и улучшить общую производительность.

Таким образом, создание приложения для работы с базами данных имеет важное значение для организации и управления информацией в различных сферах деятельности, способствуя эффективному использованию данных и повышению конкурентоспособности в современном мире.

Целью данной курсовой работы является разработка информационной системы для добавления, удаления и изменения записей в базе данных.

Для достижения этой цели были определены следующие задачи:

1. Проектирование концептуальной, логической и физической моделей базы данных.

2. Реализация физической модели в одной из современных СУБД.

3. Интеграция базы данных с приложением с использованием средств разработки.

4. Разработка функционала для работы с данными в базе данных, включая добавление, удаление, изменение и поиск записей.

1. Анализ технического задания.

1.1. Описание предметной области.

Зачастую частные организации или юридические лица с целью безопасности обращаются в охранные организации для того, что бы обезопасить себя от угроз и рисков, связанных с преступной деятельностью, вандализмом, кражами и другими правонарушениями. Охранные организации предоставляют широкий спектр услуг, включая физическую охрану объектов, установку систем видеонаблюдения, сигнализации и контроля доступа.

В данной курсовой работе поставлена задача создания приложения для работы с базой данных Охранного агентства. Для разработки такой системы необходимы СУБД для создания базы данных и среда разработки для создания интерфейса и функций по работе с БД.

Согласно заданию, в программе нужно учесть следующие особенности:

1. Возможность вести полный учёт о клиентах, добавлять, удалять, и редактировать по необходимости.
2. Добавлять новые сервисы по мере их появления в организации.
3. Вести учёт сотрудников, добавлять, удалять, и редактировать данные о них.
4. Вести учёт данных о заказах, а так же подкреплять необходимые документы при внесении любых изменений в заказ.

1.2. Анализ технического задания

Исходя из данных требований к приложению, база данных должна содержать несколько таблиц и данные таблицы должны быть связаны для целостности системы. Для разработки базы данных в рамках поставленной задачи можно использовать SQLite, которая как раз относится к СУБД реляционного типа.

Достоинством SQLite является то, что для работы не требуется устанавливать сложные серверные компоненты. Просто подключаете библиотеку к своему приложению, и можете сразу начинать работать с базой данных, а также благодаря своей легковесной архитектуре, SQLite обеспечивает высокую скорость выполнения запросов при работе с небольшими объёмами данных.

Для разработки программы, работающей с базой данных, нужно использовать объектно – ориентированный подход. Разработка приложения будет вестись на языке C#, помощью (Windows Forms(.NET Framework)). Этот фреймворк позволяет быстро разрабатывать интерфейсы с помощью визуального дизайнера и поддерживает широкий спектр стандартных элементов управления, что делает его идеальным для создания простых и функциональных приложений.

Для того чтобы создать программу, необходимо учесть то, что она создаётся, прежде всего, для пользователя, и поэтому немаловажным требованием к программе должен стать удобный и интуитивно понятный интерфейс. Необходимо предусмотреть все возможности управления приложением, чтобы упростить работу пользователя и максимально обеспечить эффективность работы.

Программа должна правильно работать с данными, т.е всегда должен выводится нужный результат, требуемый пользователю. Приложение должно мгновенно реагировать на действия пользователя и в зависимости от запроса с его стороны формировать выходной результат.

2 Разработка моделей данных

2.1 Концептуальная модель**.**

Создание концептуальной модели начинается с анализа предметной области и выделения сущностей.

Концептуальная схема или концептуальная модель данных представляет собой карту понятий и их взаимосвязей, используемых в базах данных. Она описывает семантику организации и включает в себя набор утверждений о ее сущности. В частности, модель определяет ключевые элементы, представляющие интерес для организации (классы сущностей), информацию о которых она собирает, а также их характеристики (атрибуты) и связи между парами этих значимых элементов (отношения). (Рис. 1).

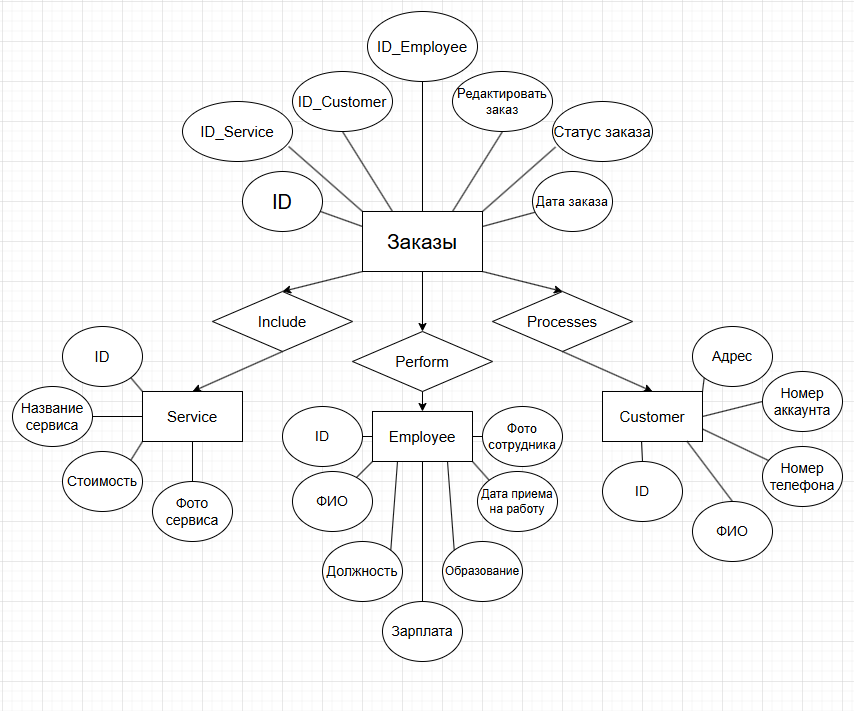


Рис. 1 - Концептуальная модель данных.

2.2. Разработка логической модели

Разработка логической модели базы данных - это этап проектирования, на котором определяется структура и организация данных без учёта конкретной реализации в виде таблиц и столбцов.

Логическая модель базы данных описывает структуру данных, их атрибуты и взаимосвязи, но не затрагивает вопросы хранения информации. На этом этапе проектирования определяются сущности (таблицы), их атрибуты (столбцы) и связи между ними. Логическая модель позволяет понять, какие данные будут храниться, как они связаны и как будут использоваться. Этот этап предшествует разработке физической модели базы данных, которая представляет собой реальную реализацию в виде таблиц, ключей и связей.

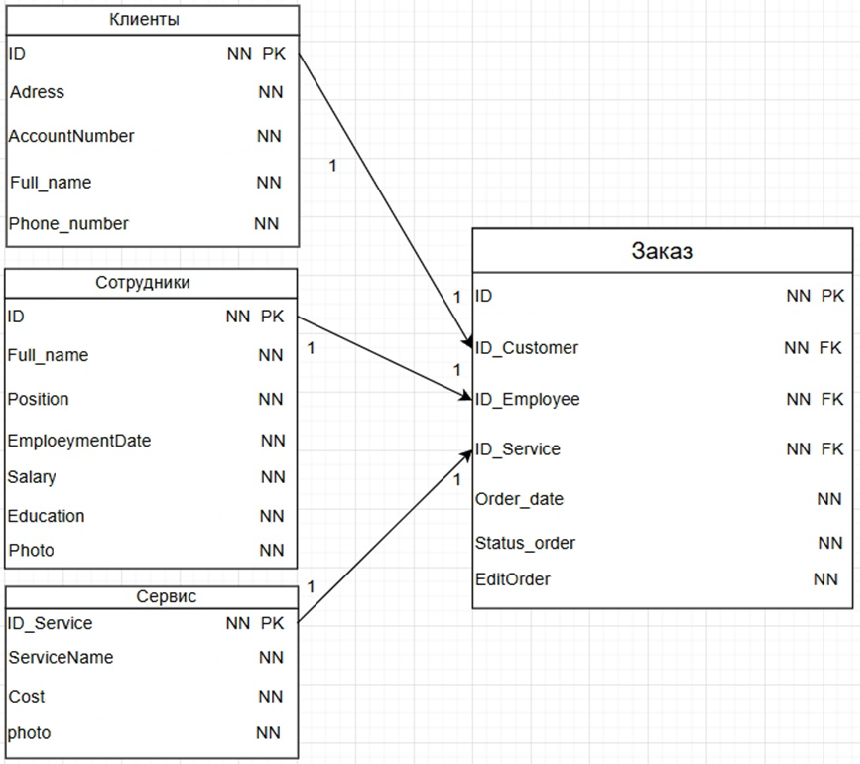


Рис. 2 - Логическая модель данных.

2.3. Разработка физической модели

Процесс разработки физической модели базы данных включает в себя несколько последовательных шагов, направленных на создание структуры для хранения и организации данных. Основные этапы этого процесса следующие:

1. Анализ требований: На этом этапе осуществляется сбор и анализ требований к базе данных, что включает определение сущностей и их атрибутов, а также выявление взаимосвязей между данными.

2. Физическая реализация: Здесь создается физическая структура базы данных, включая выбор типов данных, определение размеров полей и индексов, а также оптимизацию производительности.

3. Тестирование и оптимизация: После реализации физической модели базы данных проводится тестирование её функционирования и оптимизация для обеспечения эффективной работы.

Эти основные этапы помогают разработать физическую модель базы данных, которая будет эффективно хранить и обрабатывать данные.

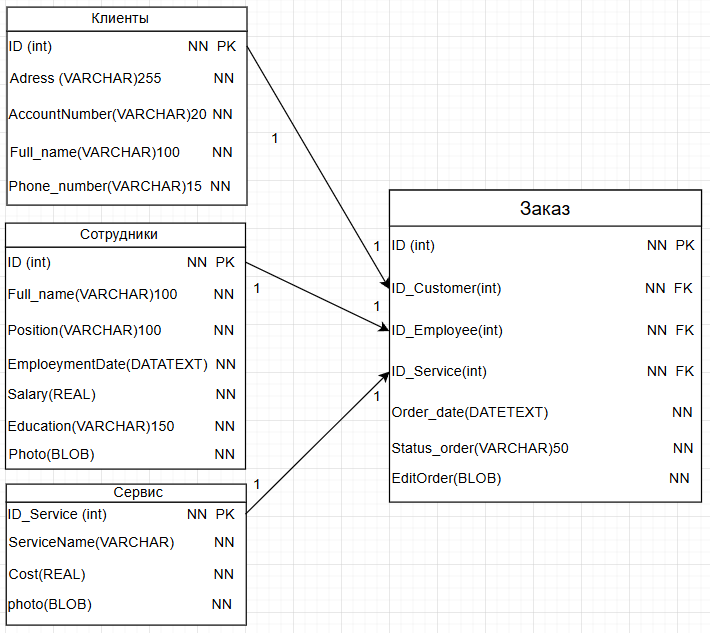


Рис.3 - Физическая модель.

В результате разработки этих моделей достигается понимание оптимизации структуры данных и обеспечение эффективного хранения и обработки информации. Каждая модель играет свою роль в процессе создания базы данных, начиная с общего представления данных и заканчивая их конкретной реализацией.

3. Разработка и реализация АИС

3.1. Разработка серверной части

На данном этапом нам нужно провести анализ требований к серверной части базы данных. Для её создания будет использоваться программное обеспечение SQLite Expert Personal, в процессе проектирования которой, мы будем обращаться к разработанной нами раннее физической модели. В этой модели будут представлены описания всех сущностей, их атрибуты и связи между ними.

Физическая модель базы данных описывает структуру и организацию данных на уровне их физического хранения. Она позволяет определить, как будет выглядеть наша база данных и как данные будут размещаться и храниться. Понимание этой концепции помогает разобраться в расположении данных в таблицах и установить связи между ними, что дает представление о готовой базе данных.

Кроме того, поддержка и обновление играют ключевую роль в обеспечении стабильной и безопасной работы информационной системы. Регулярные обновления серверной части базы данных помогают исправлять ошибки, устранять уязвимости и добавлять новые функции. Это способствует безопасности данных, а также повышает производительность и эффективность работы базы данных. Учитывая все вышеперечисленные аспекты, были разработаны следующие таблицы:

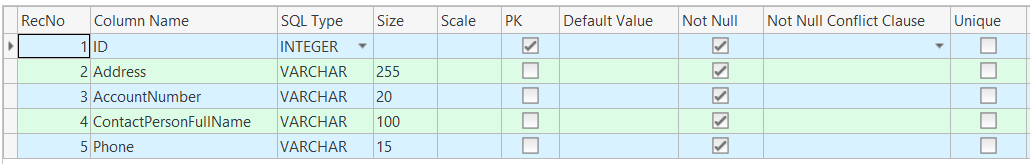


Рис. 4.1 - Таблица заказчиков в SQLite Expert Personal

Таблица 1 содержит информацию такую как ID заказчика, адрес, номер аккаунта, ФИО заказчика, номер телефона.

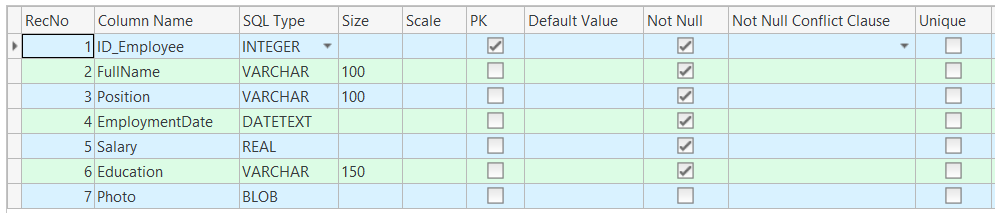


Рис. 4.2 - Таблица сотрудников в SQLite Expert Personal

Таблица 2 содержит информацию такую как ID сотрудника, ФИО сотрудника, должность, дата приема на работу, заработная плата, образование, фото сотрудника.

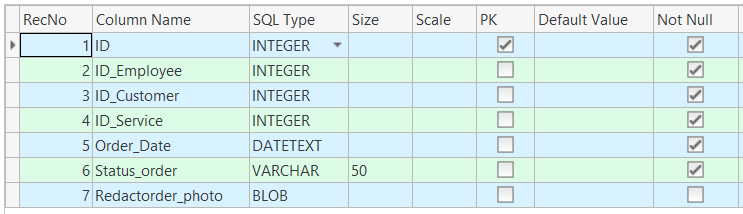


Рис. 4.3 - Таблица заказов в SQLite Expert Personal

Таблица 3 содержит информацию такую как ID заказа, ID сотрудника, ID заказчика, ID услуги, дата заказа, статус заказа, фото изменения заказа.

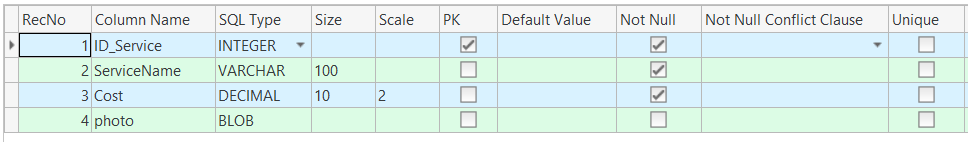


Рис. 4.4 - Таблица услуг в SQLite Expert Personal

Таблица 4 содержит информацию такую как ID услуги, название сервиса, стоимость услуги, фото услуги.

Разработка серверной части приложения для охранной организации завершилась успешно, из этого следует, что данная структура данных гарантирует надёжную основу для работы всего приложения.

Разработка SQL запросов для АИС Охранной организации.

1. Form1

Для отображения данных в таблице используется запрос:

Запрос 1

SELECT Orders.ID, Employees.FullName AS Employee\_name,Customers.ContactPersonFullName AS Customer\_name, Services.ServiceName AS Service\_name, Orders.Order\_Date, Orders.Status\_order\r\nFROM Orders \r\nINNER JOIN Employees ON Orders.ID\_Employee = Employees.ID\_Employee\r\nINNER JOIN Customers ON Orders.ID\_Customer = Customers.ID\r\nINNER JOIN Services ON Orders.ID\_Service = Services.ID\_Service;

Для отображения заказчиков используется запрос:

Запрос 2

SELECT DISTINCT Customers.ContactPersonFullName FROM Customers

Для отображения сотрудников используется запрос:

Запрос 3

SELECT DISTINCT Employees.FullName FROM Employees

Для отображения вида сервиса используется запрос:

Запрос 4

SELECT DISTINCT Services.ServiceName FROM Services

Для внесения изменения в заказ, используется подгрузка копии документа, используя запрос:

Запрос 5

UPDATE Orders SET Redactorder\_photo = @Photo WHERE ID = @id

Для добавления заказа используется запрос:

Запрос 6

INSERT INTO Orders (ID\_Employee, ID\_Customer, ID\_Service, Order\_Date, Status\_order) VALUES ((SELECT ID\_Employee FROM Employees WHERE FullName = @fio\_e),(SELECT ID FROM Customers WHERE ContactPersonFullName = @fio\_c),(SELECT ID\_Service FROM Services WHERE ServiceName = @ser), @dat, @stt

Для обновления статуса заказа используется запрос:

Запрос 7

UPDATE Orders SET Status\_order = @stt WHERE ID = @id

Для выбора записи в таблице при двойном нажатии используется запрос:

SELECT Orders.ID, Employees.FullName AS Employee\_name,Customers.ContactPersonFullName AS Customer\_name, Services.ServiceName AS Service\_name, Orders.Order\_Date, Orders.Status\_order\r\nFROM Orders \r\nINNER JOIN Employees ON Orders.ID\_Employee = Employees.ID\_Employee\r\nINNER JOIN Customers ON Orders.ID\_Customer = Customers.ID\r\nINNER JOIN Services ON Orders.ID\_Service = Services.ID\_Service WHERE Orders.ID = @id;

1. Employees

Для отображения данных в таблице используется запрос:

Запрос 8

SELECT ID\_Employee, FullName FROM Employees

Для отображения данных выбранной записи используется запрос:

Запрос 9

SELECT FullName, Position, EmploymentDate, Salary, Education, Photo FROM Employees WHERE ID\_Employee = @id

Для добавления нового сотрудника используется запрос:

Запрос 10

INSERT INTO Employees (FullName, Position, EmploymentDate, Salary, Education, Photo) VALUES (@fio, @pos, @dat, @sal, @edu, @ph)

Для удаления сотрудника используется запрос:

Запрос 11

DELETE FROM Employees WHERE ID\_Employee = @id

Для обновления данных о сотруднике используется запрос:

Запрос 12

UPDATE Employees SET FullName = @fio, Position = @pos, EmploymentDate = @dat, Salary = @sal, Education = @edu WHERE ID\_Employee = @id

1. Customer

Для отображения данных в таблице используется запрос:

Запрос 13

SELECT \* FROM Customers

Для отображения выбранной записи в текстовых полях при двойном нажатии используется запрос:

Запрос 14

SELECT Address, AccountNumber, ContactPersonFullName, Phone FROM Customers WHERE ID = @id

Для добавления нового заказчика используется запрос:

Запрос 15

INSERT INTO Customers (Address, AccountNumber, ContactPersonFullName, Phone) VALUES (@one, @two, @three, @four)

Для удаления заказчика используется запрос:

Запрос 16

DELETE FROM Customers WHERE ID = @id

Для обновления данных о заказчике используется запрос:

Запрос 17

UPDATE Customers SET Address = @one, AccountNumber = @two, ContactPersonFullName = @three, Phone = @four WHERE ID = @id

1. Servis

Для отображения данных в таблице используется запрос:

Запрос 18

SELECT ID\_Service,ServiceName FROM Services

Для обновления данных о сервисе используется запрос:

Запрос 19

UPDATE Services SET ServiceName = @sv\_nm, Cost = @cst WHERE ID\_Service = @id

Для отображения данных выбранной записи в текстовых полях используется запрос:

Запрос 20

SELECT ID\_Service, ServiceName, Cost, photo FROM Services WHERE ID\_Service = @id

Для добавления новой записи сервиса используется запрос:

Запрос 21

INSERT INTO Services (ServiceName, Cost, photo) VALUES (@nm\_sv, @pr, @ph)

Для удаления сервиса используется запрос:

Запрос 22

DELETE FROM Services WHERE ID\_Service = @id

1. Rep\_rec

Для подсчета общей суммы заказов используется запрос:

Запрос 23

SELECT SUM(Services.Cost) AS TotalCost\r\nFROM Orders \r\nJOIN Services ON Orders.ID\_Service = Services.ID\_Service \r\nWHERE Orders.Order\_Date BETWEEN @startdate AND @enddate;

Для подсчета количества заказов используется запрос:

Запрос 24

SELECT COUNT(\*) AS OrderCount FROM Orders WHERE Order\_Date BETWEEN @startdate AND @enddate;

1. Statistic

Для выбора сотрудника используется запрос:

Запрос 25

SELECT Employees.FullName FROM Employees

Для подсчета количества заказов на определенного сотрудника используется запрос:

Запрос 26

SELECT COUNT(\*) AS stt FROM Orders WHERE ID\_Employee = (SELECT ID\_Employee FROM Employees WHERE FullName = @ide)

3.2. Разработка клиентской части

Интерфейс представляет собой способ взаимодействия между человеком и компьютерной системой, программой или устройством. Он включает в себя набор элементов, позволяющих пользователю взаимодействовать с приложением или устройством, выполнять определённые действия и получать нужную информацию.

Удобный интерфейс играет ключевую роль в обеспечении положительного пользовательского опыта, поскольку он делает работу с приложением более понятной и комфортной. Важно, чтобы интерфейс был простым и легким в освоении, чтобы пользователи могли быстро и эффективно достигать своих целей. Хорошо продуманный интерфейс также помогает уменьшить количество ошибок и улучшает общее впечатление от приложения. Поэтому необходимо уделять особое внимание созданию интуитивно понятного и удобного интерфейса для обеспечения положительного опыта пользователей. При этом важно, чтобы интерфейс соответствовал необходимому функционалу, что также играет значительную роль в разработке приложения.

Исходя из всего вышесказанного был разработан графический интерфейс пользователя:

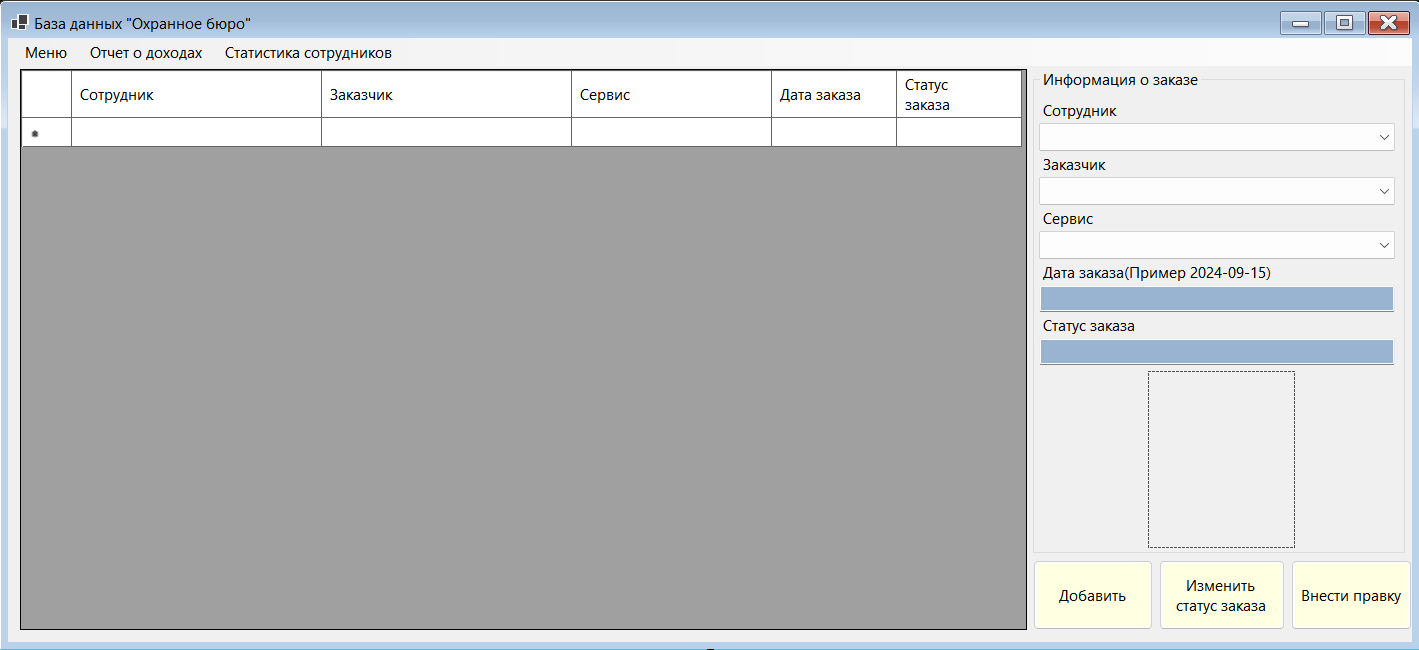


Рис. 5.1 - Форма заказов

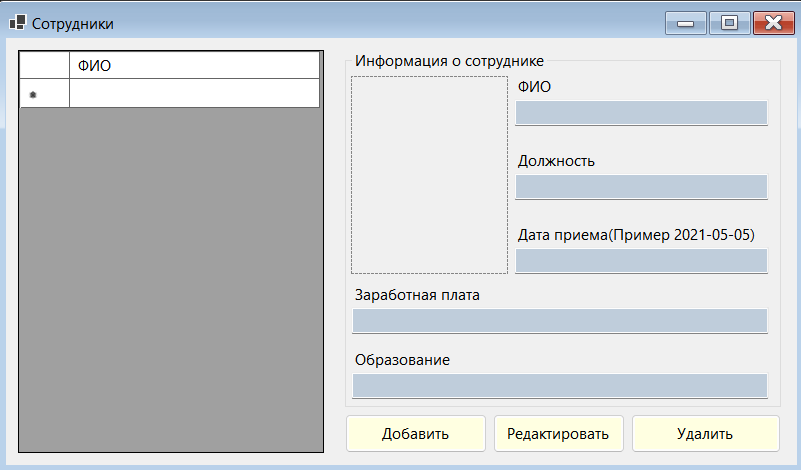


Рис. 5.2 - Форма сотрудников

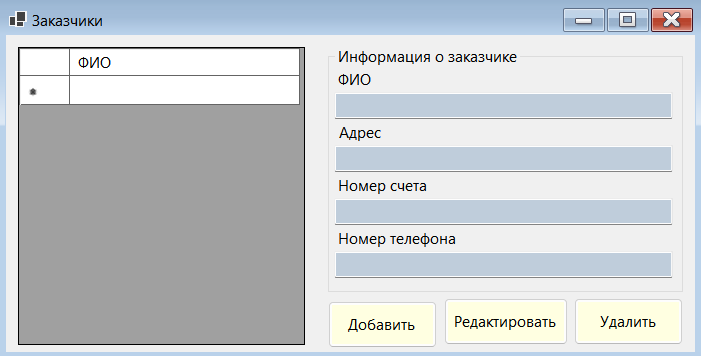


Рис. 5.3 - Форма заказчиков

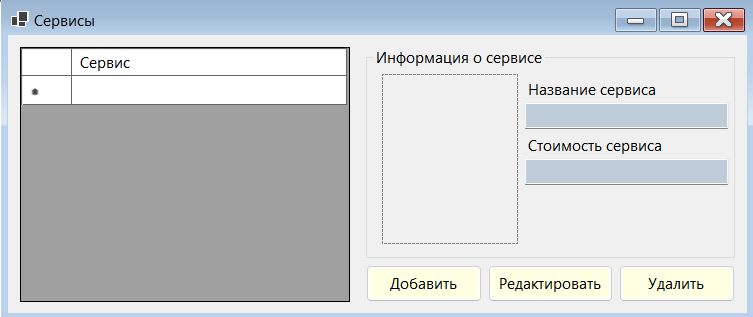


Рис. 5.4 - Форма сервисов

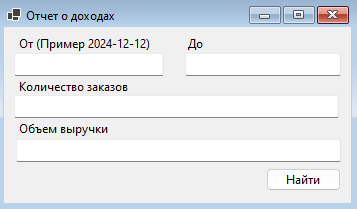


Рис. 5.5 - Форма отчета о доходах

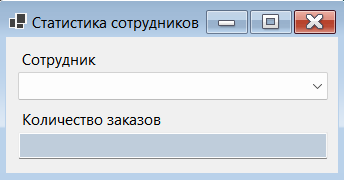


Рис. 5.6 - Форма статистики сотрудников

Разработка подпрограмм

1. Form1

Событие 1

private void Form1\_Load(object sender, EventArgs e) - Данное событие помогает отобразить все данные в таблице, а так же в текстовых полях.

Событие 2

private void button2\_Click(object sender, EventArgs e) - Данное событие помогает внести изменение в заказ за счет прикрепления копии документа.

Событие 3

private void button1\_Click(object sender, EventArgs e) - Данное событие помогает добавить запись о заказе в базу данных.

Событие 4

private void button3\_Click(object sender, EventArgs e) - Данное событие помогает изменить статус заказа.

Событие 5

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e) - Данное событие помогает отобразить данные выбранные в таблице при двойном нажатии.

1. Employees

Событие 6

private void Employees\_Load(object sender, EventArgs e) - Данное событие помогает отобразить в таблице данные о сотрудниках.

Событие 7

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e) - Данное событие помогает отобразить данные о сотруднике в текстовых полях при двойном нажатии записи в таблице.

Событие 8

private void button1\_Click(object sender, EventArgs e) - Данное событие помогает добавить нового сотрудника в базу.

Событие 9

private void button3\_Click(object sender, EventArgs e) - Данное событие помогает удалить запись о сотруднике.

Событие 10

private void button2\_Click(object sender, EventArgs e) - Данное событие помогает обновить данные о сотруднике.

1. Customer

Событие 11

private void Customer\_Load(object sender, EventArgs e) - Данное событие помогает подгрузить данные о заказчике в таблицу.

Событие 12

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e) - Данное событие помогает загрузить данные в текстовые поля при двойном нажатии на запись в таблице.

Событие 13

private void button1\_Click(object sender, EventArgs e) - Данное событие помогает добавить новую запись о заказчике.

Событие 14

private void button3\_Click(object sender, EventArgs e) - Данное событие помогает удалить запись о заказчике.

Событие 15

private void button2\_Click(object sender, EventArgs e) - Данное событие помогает обновить данные о заказчике.

1. Servis

Событие 16

private void Servis\_Load(object sender, EventArgs e) - Данное событие помогает загрузить данные о сервисе в таблицу.

Событие 17

private void button2\_Click(object sender, EventArgs e) - Данное событие помогает обновить данные о сервисе.

Событие 18

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e) - Данное событие помогает отобразить данные выбранной записи в таблице при двойном нажатии.

Событие 19

private void button1\_Click(object sender, EventArgs e) - Данное событие помогает добавить в базу новую запись о сервисе.

Событие 20

private void button3\_Click(object sender, EventArgs e) - Данное событие помогает удалить вид сервиса из базы.

1. Rep\_rec

Событие 21

private void Rep\_rec\_Load(object sender, EventArgs e) - Данное событие помогает подсчитать общую вырученную сумму с заказов, и так же подсчитывает общее количество заказов.

1. Statistic

Событие 22

private void Statistic\_Load(object sender, EventArgs e) - Данное событие помогает подгрузить список сотрудников.

Событие 23

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e) - Данное событие помогает подсчитать количество заказов выполненное конкретным сотрудником.

3.3. Руководство пользователю и администратору

Руководство администратора.

Чтобы мы смогли воспользоваться приложением, его необходимо распаковать и сохранить в папке на своём персональном компьютере:

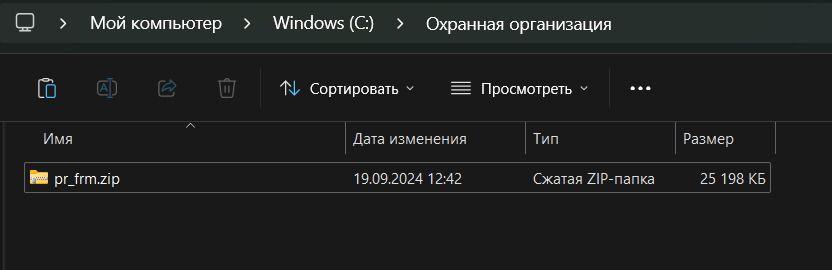


Рис. 5.7 - Извлечение файлов приложения из архива.

И финальным шагом, чтобы его запустить нужно открыть файл с расширением .sln:

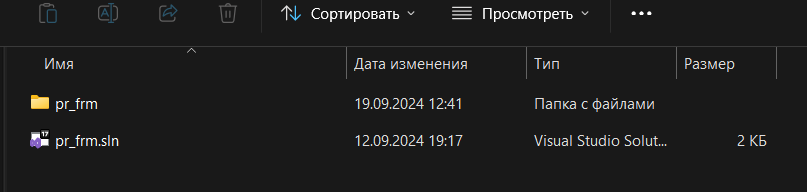


Рис. 5.8 - Запуск приложения.

Для использования нашей базы, так же необходимо установить архив с нашей базой, распаковать его и поместить в папку на своём ПК:

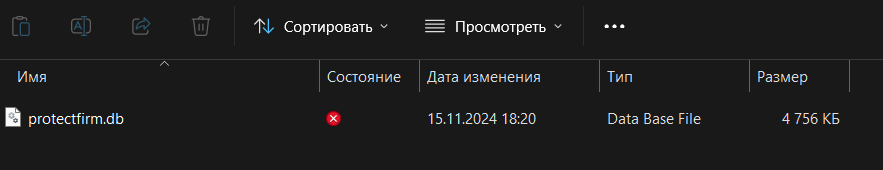


Рис. 5.9 - Размещение файла БД.

Следующим шагом разберём примеры использования приложения, приводя примеры функционала:

При запуске приложения мы видим список заказов:

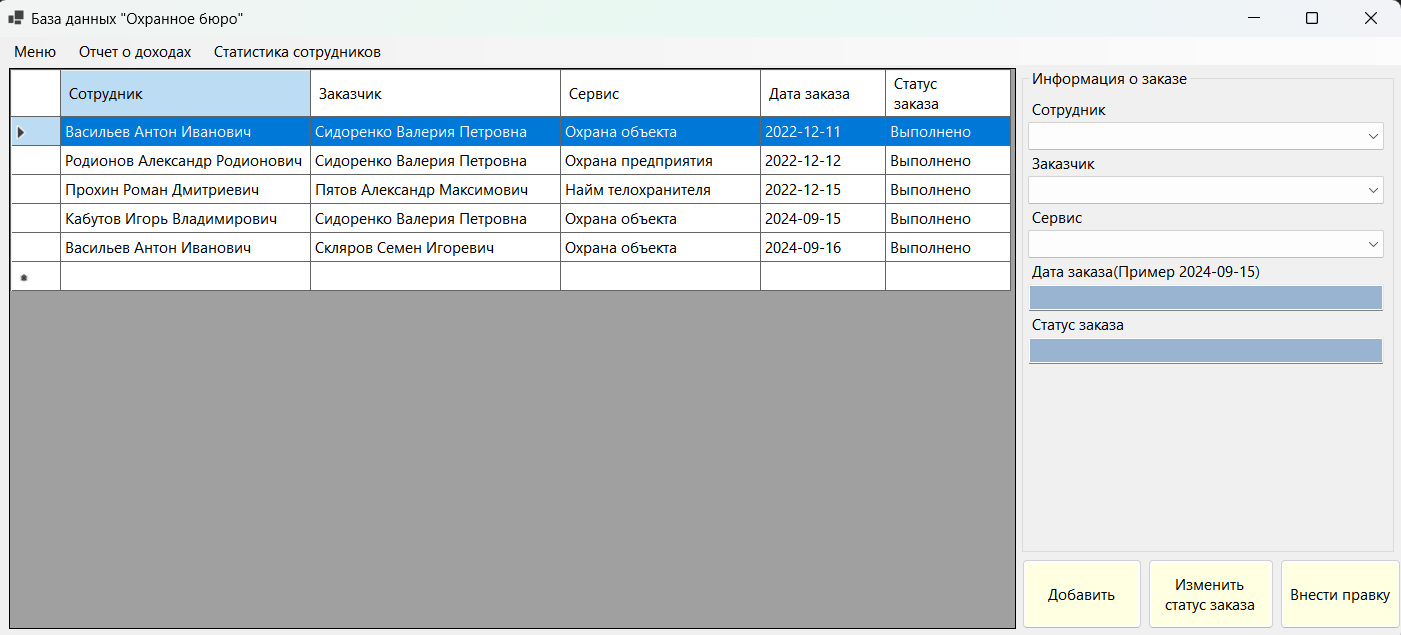


Рис. 5.10 - Окно заказов.

В процессе добавления заказа нам необходимо добавить нового заказчика в базу. Чтобы это реализовать нам необходимо перейти в меню в раздел «Клиенты».

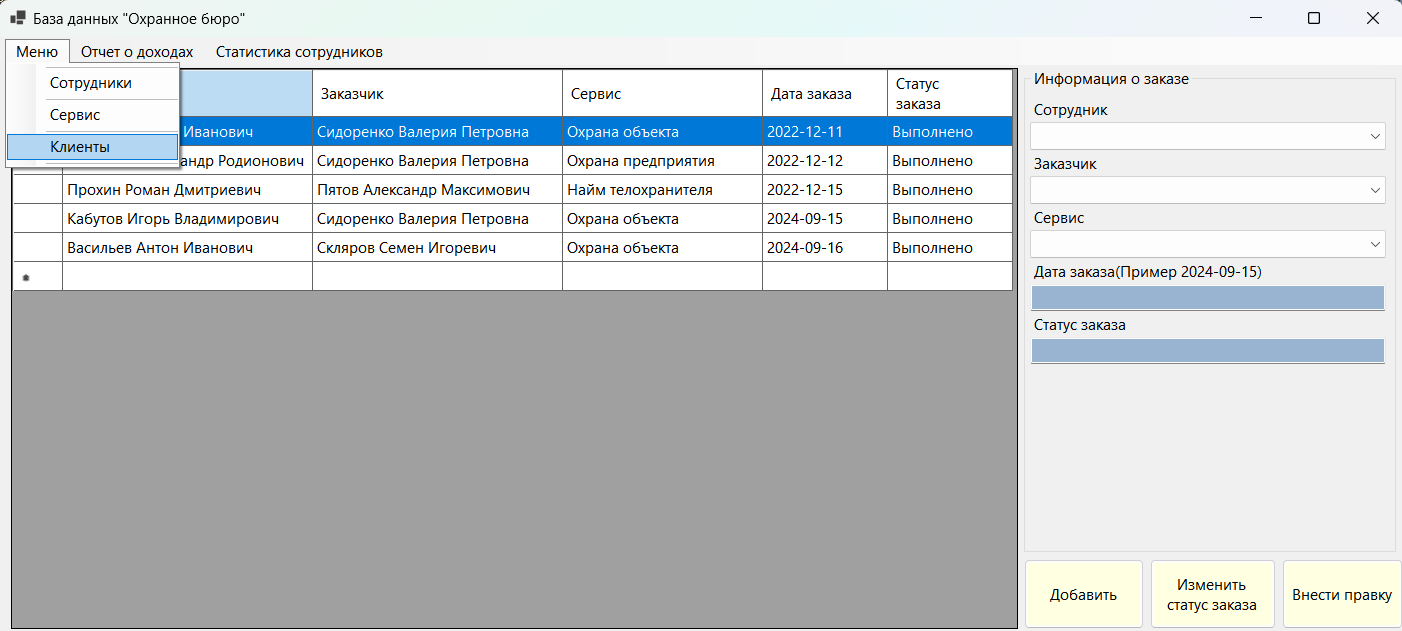


Рис. 5.11 - Раздел Клиенты.

В последствии чего мы видим как нам открывается окно со списком всех наших клиентов:

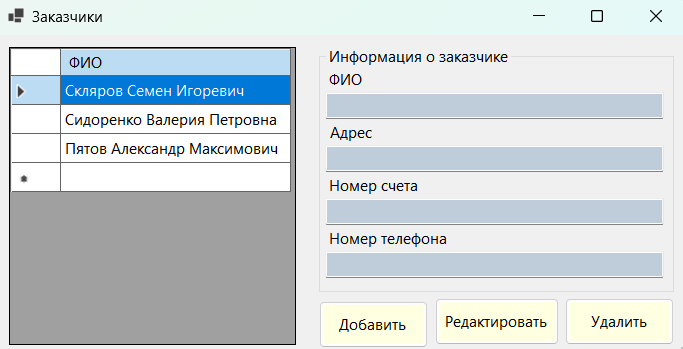


Рис. 5.12 - Окно списка Клиентов.

Для добавления нового Клиента необходимо заполнить все текстовые поля и нажать кнопку «Добавить».

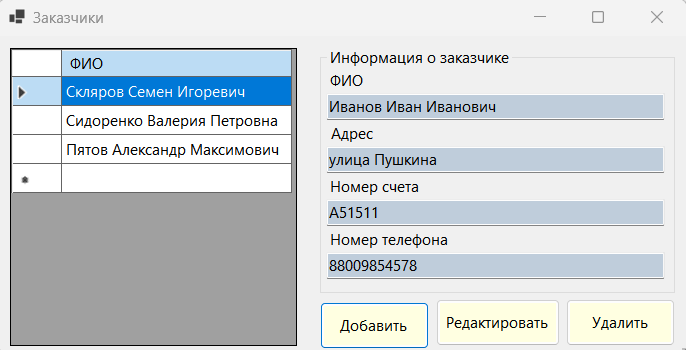


Рис. 5.13 - Внесение данных о Клиенте.

Корректно записав все данные мы получаем сообщение о том, что запись успешно добавлена в базу:

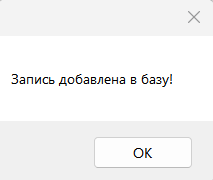


Рис. 5.14 - Сообщение об успешном добавлении записи.

Как мы можем наблюдать далее, запись была успешно добавлена и отображается в нашем списке:

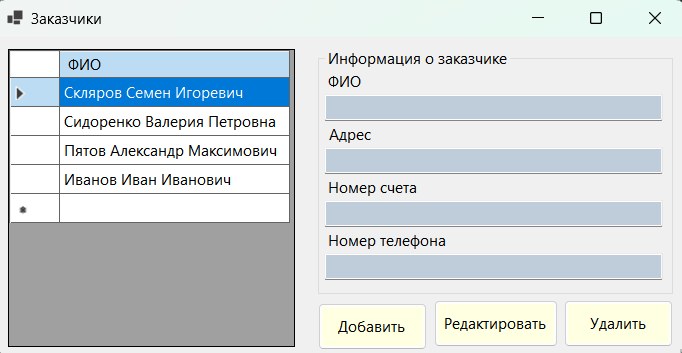


Рис. 5.15 - Отображение добавленной записи.

Если вдруг случается такая ситуация, что мы допустили ошибку при заполнении данных клиента, то нам необходимо нажать два раза на его запись в таблице на форме Заказчики (Рис. 5.16).

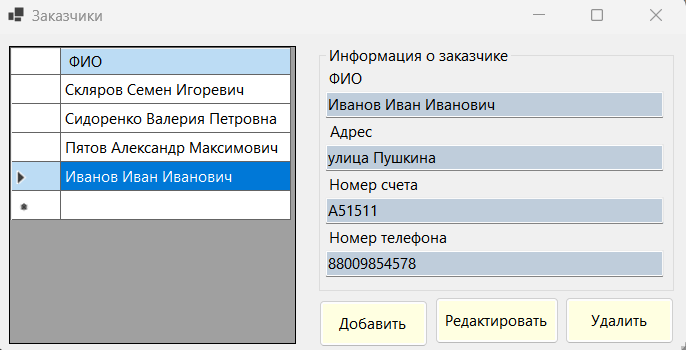


Рис. 5.16 - Выбор записи для редактирования.

При двойном нажатии на запись заказчика в текстовых полях появляются его данные, которые можно изменить, в последствии внесения корректных данных нажимаем кнопку «Редактировать» после чего у нас появляется окно, которое сообщает нам о том, что данные успешно изменены:

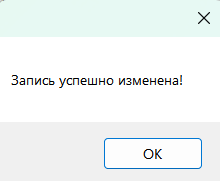


Рис. 5.17 - Окно успешного изменения данных.

После добавления нового заказчика и возможной корректировки его записи, мы можем добавить новый заказ. Нам нужно выбрать сотрудника, заказчика и вид сервиса, а так же указать дату заказа, статус заказа может быть заполнен как сразу, в случае если заказ был отменен, или же после его выполнения.

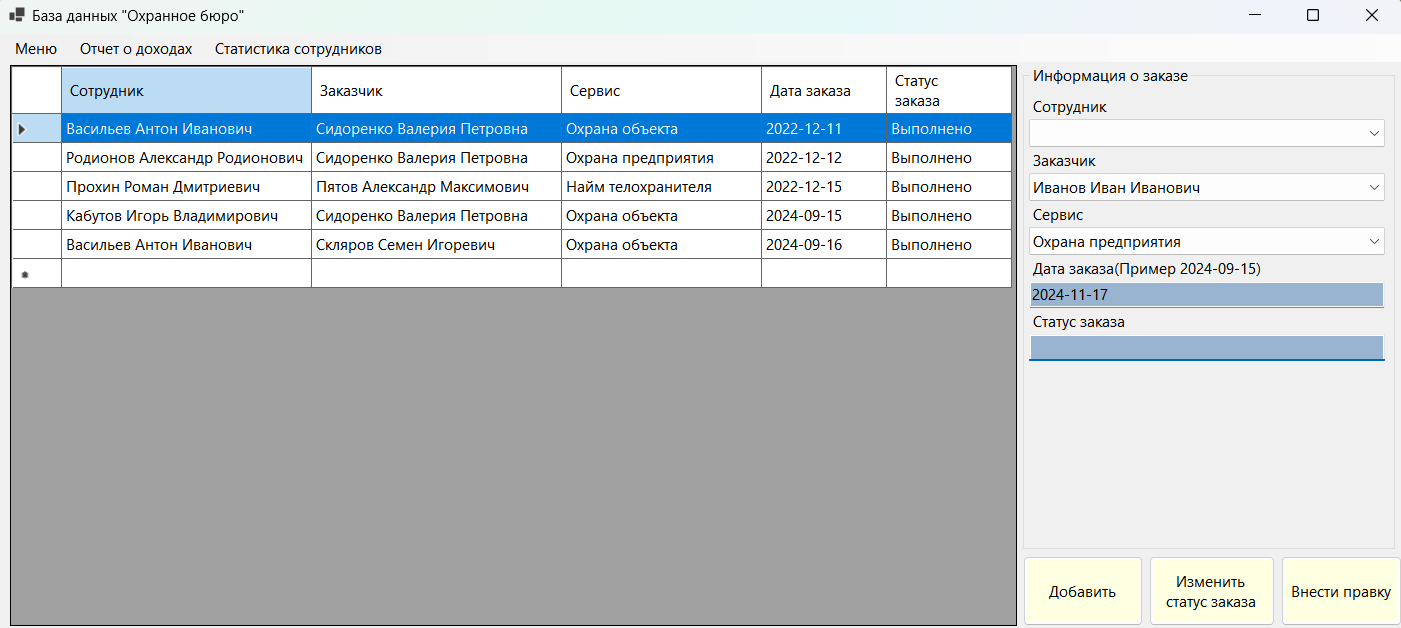


Рис. 5.18 - Окно заполнения данных о заказе.

Внеся все данные в запись о заказе, мы нажимаем на кнопку «Добавить», и у нас появляется окно подтверждающее добавление новой записи:

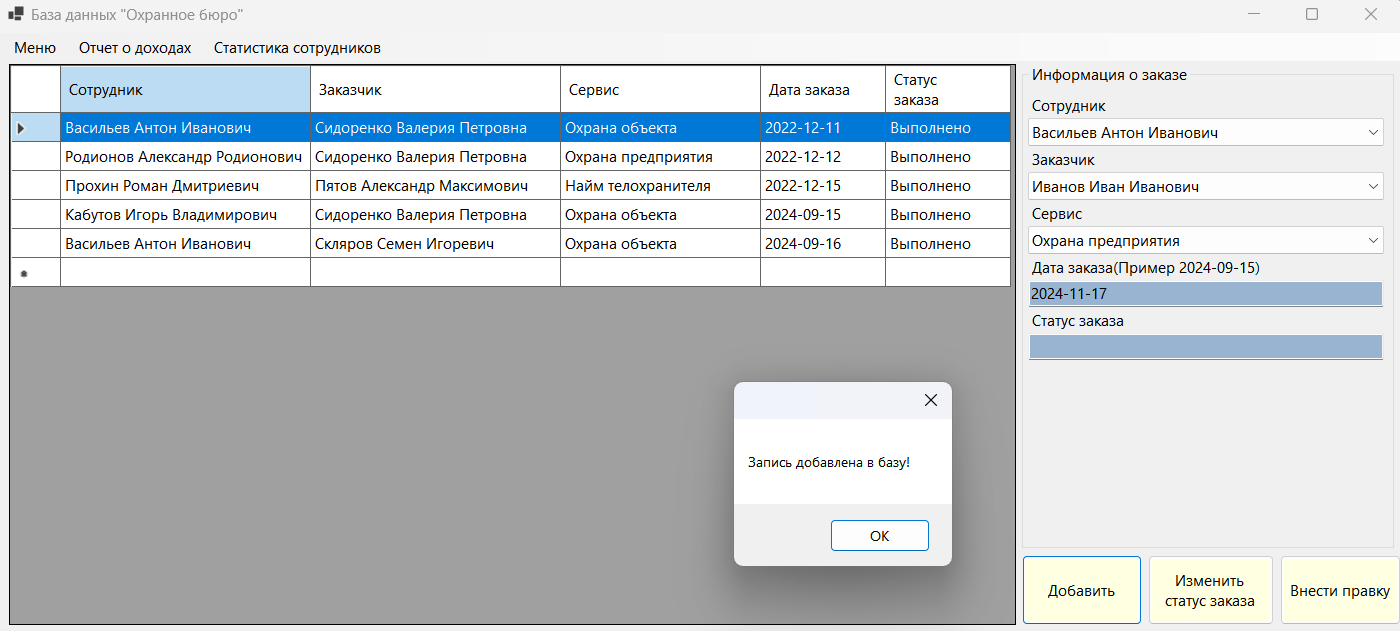


Рис. 5.19 - Успешное добавление заказа.

Так же при тестировании необходимо проверить работу остальных форм. Приступим к тестированию формы сервисов. Открыв меню, перейдём в пункт «Сервис»:

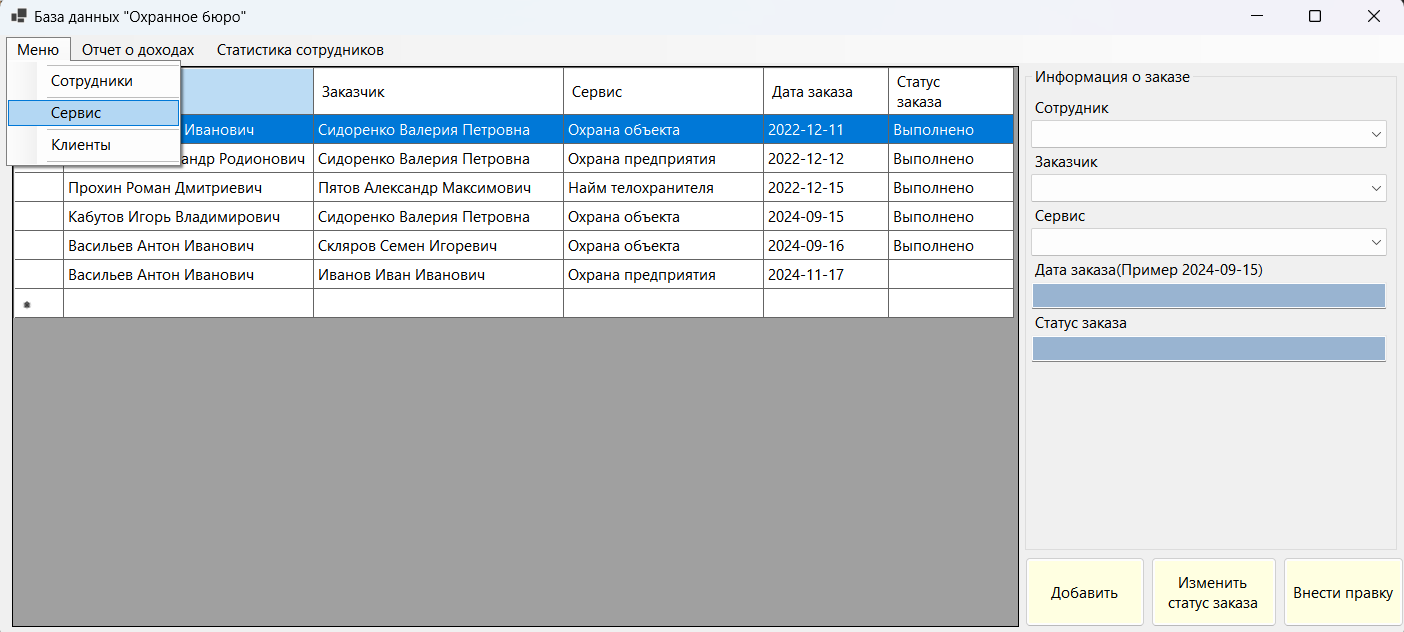


Рис. 5.20 - Раздел Сервис.

Перейдя в данный пункт перед нами появляются все наши лаборатории:

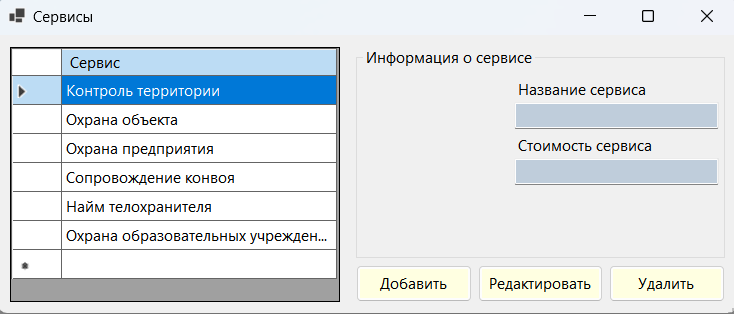


Рис. 5.21 - Окно сервисов.

Чтобы добавить новый сервис, заполняем все необходимые текстовые поля, а так же в дополнительно окне выбираем фотографию соответствующую нашему сервису, после чего получаем сообщение о том что данные успешно добавлены:

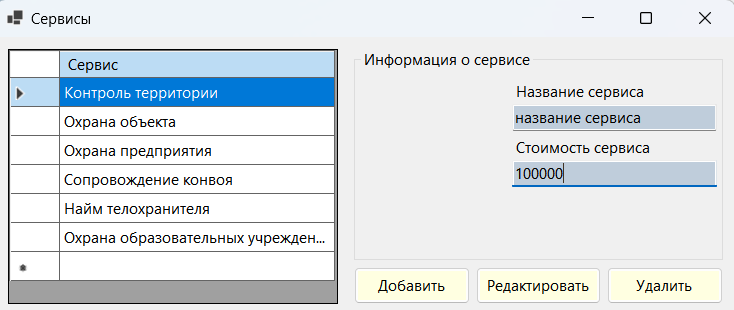


Рис. 5.22 - Заполнение данных.

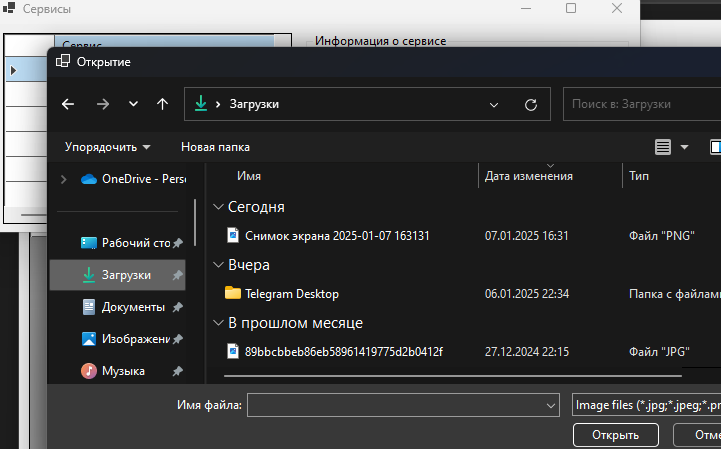


Рис. 5.23 - Дополнительное окно для выбора фотографии.

После чего мы видим, что запись успешнор добавлена:

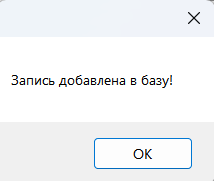


Рис. 5.24 - Сообщение об успешном добавлении записи.

Для редактирования записи так же используется двойное нажатие в таблице, после чего данные переносятся в текстовые поля для редактирования:

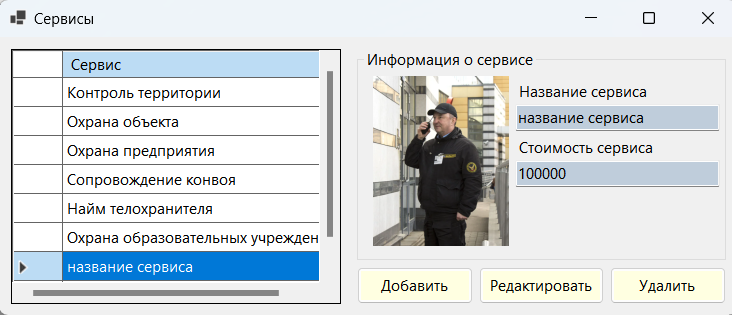


Рис. 5.25 - Отображение редактирования записи.

После внесения изменений нажимаем кнопку «Редактировать», и получаем сообщение об успешном внесении изменений:

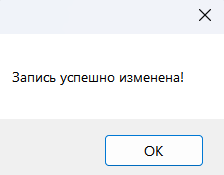


Рис. 5.26 - Сообщение об успешном изменении.

Если нам необходимо удалить запись, то нужно выбрать запись в таблице одним нажатием и нажать кнопку удалить, после чего получим сообщение об успешном удалении записи:

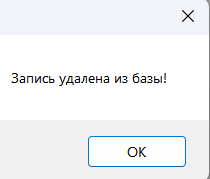


Рис. 5.26 - Сообщение об удалении записи.

Далее будет рассмотрена форма сотрудников, для этого нам нужно перейти во вкладку меню и выбрать пункт «Сотрудники»:

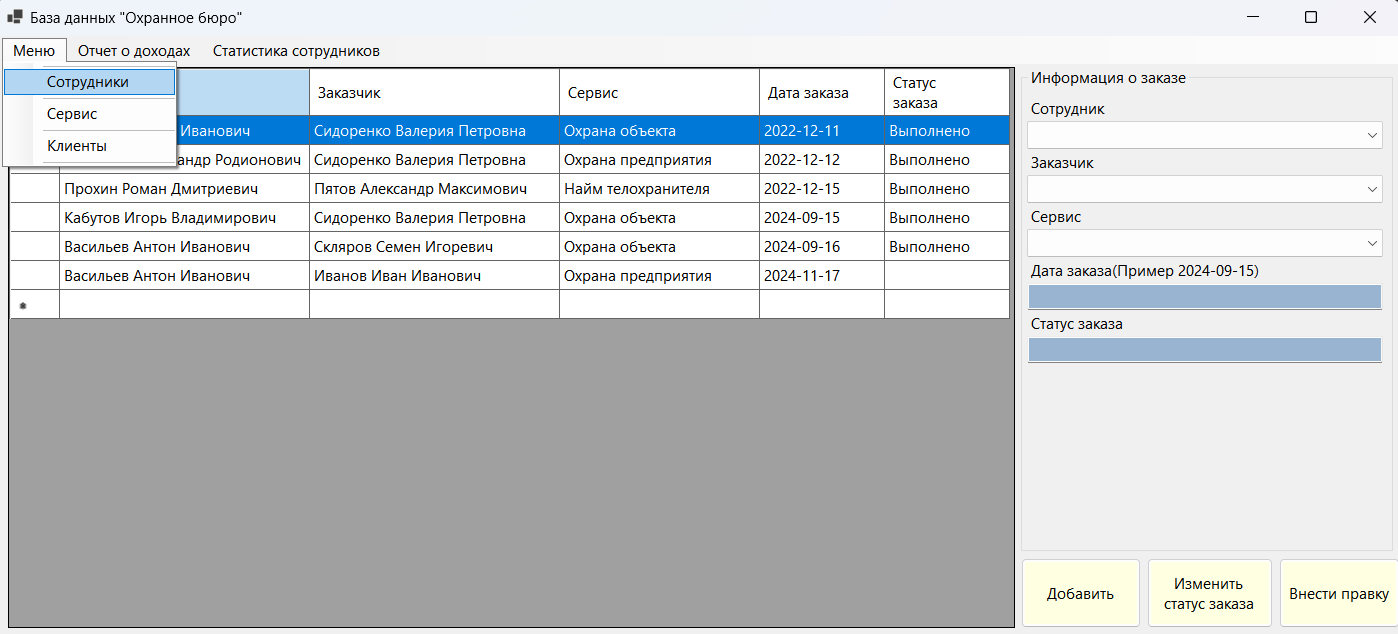


Рис. 5.27 - Раздел Сотрудники.

После чего перед нами отобразится окно сотрудников:

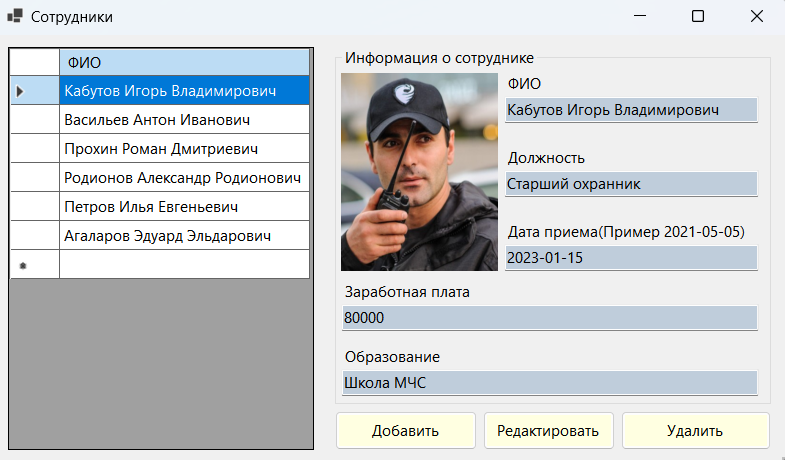


Рис. 5.28 - Окно сотрудников.

Для добавления нового сотрудника нужно заполнить все текстовые поля, после чего выбрать фотографию сотрудника и нажать кнопку добавить, после чего появится кнопка об успешном добавлении записи:

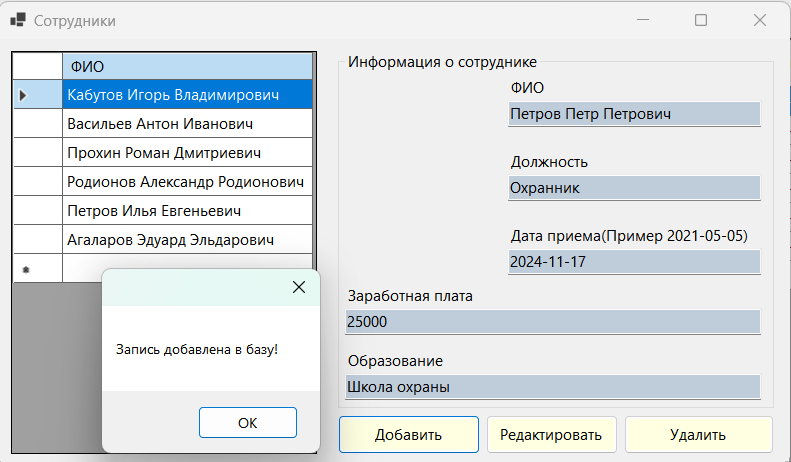


Рис. 5.29 - Сообщение об успешном добавлении записи.

Если нам нужно отредактировать запись о сотруднике, то нам нужно два раза нажать на запись в таблице и данные перенесутся в текстовые поля, редактируем их и нажимаем кнопку «Редактировать»:

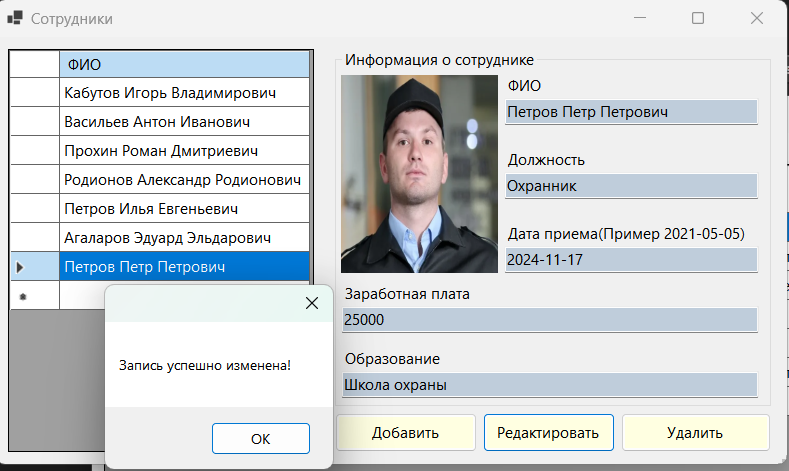


Рис. 5.30 - Сообщение об успешном изменении записи.

Если нам нужно удалить запись, то просто выбираем запись в таблице одним нажатием и нажимаем кнопку удалить, после чего появится сообщение об успешном удалении:

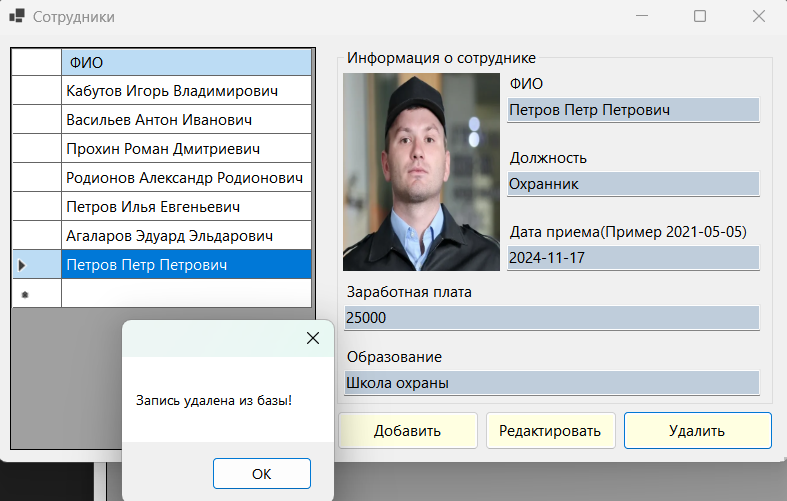


Рис. 5.31 - Сообщение об успешном удалении записи.

А так же перейдя на вкладку отчет о доходах можно посмотреть какую общую сумму заработало предприятие со всех заказов или доход за определенный период:

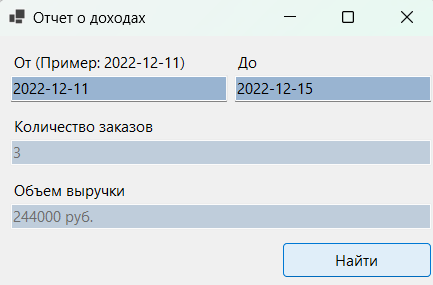


Рис. 5.32 - Окно отчета о доходах.

Последним этапом будет тестирование окна статистики сотрудников, в нем нам необходимо выбрать сотрудника статистика которого нас интересует, и в другом текстовом поле появится нужная информация:

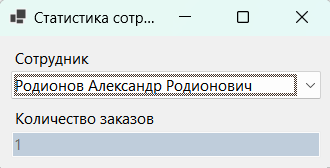


Рис. 5.33 - Окно статистики сотрудников.

Тестирование автоматизированной информационной системы (АИС) близится к завершению. Проведённые испытания подтвердили, что приложение соответствует всем заданным требованиям и работает корректно. В ходе тестирования была проверена работа всех компонентов системы и их взаимодействие. Это дало возможность убедиться в готовности информационной системы к практическому использованию, а также в её высоком качестве и надёжности.

Заключение

В ходе выполнения курсовой работы была разработана и реализована информационная система для охранной организации, которая позволяет автоматизировать основные процессы учёта, и отслеживания заказов. Создание данной автоматизированной информационной системы позволит значительно повысить эффективность работы предприятия, а так же улучшить качество предоставляемых услуги и упростить управление данными.

Автоматизированная информационная система охранной организации предоставляет широкий спектр функциональных возможностей. В результате проведённого тестирования было установлено, что разработанная информационная система полностью соответствует поставленным требованиям, а также обеспечивает эффективное взаимодействие с другими информационными системами. Таким образом, создание автоматизированной информационной системы для охранной организации является актуальной и перспективной задачей, способной значительно улучшить качество услуг сфер безопасности.

Список используемой литературы.

1. Джей А. Крейбич «Использование SQLite» (2010 г.) О’Рейли Медиа, Инк. - 530 с.

2. Мюллер Д.П. «С# для чайников», [ООО "Диалектика"](https://coollib.com/p/25371-ooo-dialektika) 2019 г. - 609 с.

3. С.Э. Мурадханов «Разработка на языке C# приложений с графическим интерфейсом : использование Windows Forms» 2019 г. [Издательский Дом НИТУ «МИСиС»](https://znanium.ru/catalog/publishers/books?ref=def4cf26-34c4-11e4-b05e-00237dd2fde2). - 396 с.

4. Алан Блюли «Изучаем SQL» 2007 г. О’Рейли Медиа, Инк. - 309 с.

# 5. «Проектирование баз данных: основные принципы»

# https://habr.com/ru/companies/otus/articles/731010/

6. «Работа с базой данных SQLite» https://habr.com/ru/articles/265303/

Приложение

Ссылка на проект: https://github.com/CHAIsBERGAMOTOM/KURSASH