МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В. Ф. УТКИНА»

Факультет вычислительной техники

Кафедра электронных вычислительных машин

К защите

Руководитель проекта

|  |
| --- |
|  |

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

по дисциплине

«Базы данных и клиент-серверные приложения»

по теме

«Разработка информационной системы с клиент-серверной архитектурой»

Выполнил:

студент группы 845

|  |  |
| --- | --- |
| Муханов В. С. |  |
|  | дата сдачи на проверку, подпись |

Руководитель проекта:

ассистент кафедры ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Хизриева Н.И. |  |  |  |  |
|  |  | оценка |  | дата защиты, подпись |

Рязань 2021 г

ЗАДАНИЕ

на курсовой проект по дисциплине   
«Базы данных и клиент-серверные приложения»”

Исполнитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема: «Разработка информационной системы с клиент-серверной архитектурой»

Вариант задания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Требования к используемому программному обеспечению:

* ОС: MS Windows;
* СУБД: MS SQL Server;
* система программирования: MS Visual Studio (язык программирования C#);
* программа инсталляции: Inno Setup.

Общие требования к проекту:

Разработать информационную систему (ИС) с клиент-серверной архитектурой, используя двухзвенную модель DBS (сервер баз данных).

Провести инфологическое и даталогическое проектирование базы данных (БД). Проверить все таблицы БД на соответствие нормальной форме Бойса-Кодда. Предусмотреть контроль целостности данных на уровне ограничений, триггеров, правил, умолчаний и хранимых процедур. Все действия по внесению изменений в БД оформить в виде хранимых процедур. В целом, в БД должно быть не менее 10 таблиц.

Клиентская часть ИС должна охватывать все информационные потребности пользователя. Предусмотреть возможность печати необходимых пользователю отчетов. Для доступа к БД использовать технологию ADO.NET.

С помощью инсталляционной программы разработать дистрибутив ИС, позволяющий установить клиентское приложение на локальной машине.

К защите представить:

- работоспособную программу и БД, установленные на компьютере в ауд.210 или ауд.122;

- пояснительную записку в распечатанном виде;

- пояснительную записку, дистрибутив ИС, исходные тексты программ, файлы БД, сценарий инсталляции ИС в электронном виде.

Требования к содержанию пояснительной записки:

ВВЕДЕНИЕ

1. Концептуальное описание предметной области
2. Характеристики ИС
3. Анализ существующих решений для предметной области
4. Разработка общей структуры ИС
5. Разработка серверной части информационной системы
   1. Инфологическое проектирование БД
   2. Даталогическое проектирование БД
   3. Программирование объектов БД
6. Разработка клиентского приложения
   1. Выбор программных компонентов клиентской части
   2. Разработка интерфейса пользователя
      1. Разработка форм
   3. Разработка сценария инсталляции клиентской программы
7. Разработка программной документации
8. Тестирование ИС

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЕ А: сценарий создания объектов БД.

ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД.

ПРИЛОЖЕНИЕ В: исходный текст клиентской программы.

ПРИЛОЖЕНИЕ Г: сценарий инсталляции программы.

Дата выдачи задания: 19 февраля 2021 г.

Срок сдачи проекта на проверку:

Подпись исполнителя:

Подпись руководителя:

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc75865444)

[1 АНАЛИЗ ЗАДАЧИ 6](#_Toc75865445)

[1.1 Концептуальное описание предметной области 6](#_Toc75865446)

[1.2 Характеристики ИС 6](#_Toc75865447)

[1.3 Анализ существующих решений для предметной области 7](#_Toc75865448)

[1.4 Разработка общей структуры ИС 7](#_Toc75865449)

[2 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ 9](#_Toc75865450)

[2.1 Инфологическое проектирование 9](#_Toc75865451)

[2.1.1 Выделение сущностей и связей в заданной предметной области. 12](#_Toc75865452)

[2.1.2 Построение ER-диаграмм 13](#_Toc75865453)

[2.2 Даталогическое проектирование БД 19](#_Toc75865454)

[2.2.1 Переход от ER-диаграммы к предварительным отношениям 19](#_Toc75865455)

[2.2.2 Подготовка списка всех представляющих интерес атрибутов, кроме тех, которые уже были выделены на ER-диаграмме классов 21](#_Toc75865456)

[2.2.3 Распределение атрибутов из списка, причем каждый атрибут помещается только в одно из отношений. 21](#_Toc75865457)

[2.2.4 Проверка отношений на БКНФ 22](#_Toc75865458)

[2.2.5 Пересмотр ER-диаграммы 29](#_Toc75865459)

[2.2.6 Построение схемы базы данных 30](#_Toc75865460)

[2.3 Программирование объектов БД 31](#_Toc75865461)

[2.3.1 Разработка хранимых процедур 31](#_Toc75865462)

[2.3.2 Разработка таблиц, триггеров и представлений. 35](#_Toc75865463)

[3 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 36](#_Toc75865464)

[3.1 Выбор программных компонентов клиентской части 36](#_Toc75865465)

[3.2 Разработка интерфейса пользователя 36](#_Toc75865466)

[3.2.1 Разработка форм 36](#_Toc75865467)

[3.3 Разработка сценария инсталляции клиентской программы 43](#_Toc75865468)

[4 РАЗРАБОТКА ПРОГРАММНОЙ ДОКУМЕНТАЦИИ 49](#_Toc75865469)

[5 ТЕСТИРОВАНИЕ ИС 50](#_Toc75865470)

[5.1 Тестирование объектов БД 50](#_Toc75865471)

[5.1.1 Тестирование частных ограничений целостности данных 50](#_Toc75865472)

[5.1.2 Тестирование представлений 53](#_Toc75865473)

[5.1.3 Тестирование ХП и триггеров 54](#_Toc75865474)

[5.2 Тестирование клиентской части 56](#_Toc75865475)

[ЗАКЛЮЧЕНИЕ 63](#_Toc75865476)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 64](#_Toc75865477)

[ПРИЛОЖЕНИЕ А: сценарий создания объектов БД. 65](#_Toc75865478)

[ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД. 72](#_Toc75865479)

[ПРИЛОЖЕНИЕ В: исходный текст клиентской программы. 75](#_Toc75865480)

ВВЕДЕНИЕ

В данном курсовом проекте рассматривается создание реляционной базы данных и приложения для управляющей компании, которое используется для создания и управления заявками пользователей, а также для хранения и управления информацией о сотрудниках и жильцах.

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных.

С ростом популярности СУБД в 70-80-х годах появилось множество различных моделей данных. У каждой из них имелись свои достоинства и недостатки, которые сыграли ключевую роль в развитии реляционной модели данных, появившейся во многом благодаря стремлению упростить и упорядочить первые модели данных.

1 АНАЛИЗ ЗАДАЧИ

## 1.1 Концептуальное описание предметной области

Управляющая компания — коммерческая организация, юридическое лицо, осуществляющее доверительное управление имуществом других физических и юридических лиц, переданным в соответствии с договором доверительного управления.

Целью деятельности управляющих организаций в сфере ЖКХ является содержание ОИ (общего имущества) МКД (в том числе придомовой территории) в надлежащем состоянии, а также обеспечение возможности использования ОИ по его назначению.

С точки зрения организации деятельности, управляющая организация является посредником между собственниками многоквартирного жилого дома и лицами, оказывающими услуги по его содержанию и ремонту общего имущества многоквартирного дома, РСО (ресурсоснабжающими организациями) и другими организациями, деятельность которых связана с ОИ МКД дома (арендаторы общедомового имущества, организации по охране ОИ и пр.).

Для учета обращений собственников и жильцов необходимо разработать программное обеспечение, предусматривающее контроль исполнения заявок и содержащие следующие сущности.

## 1.2 Характеристики ИС

Необходимо разработать базу данных управляющей компании и хранить в ней следующую информацию:

Обслуживаемые дома (Адрес, Кол-во Квартир, Кол-во Подъездов)

Находящиеся в домах квартиры (Код Квартиры, Адрес, Подъезд, Этаж, Номер квартиры)

Список физических лиц, проживающих в квартирах или являющихся сотрудниками (ФИО, Паспорт, Телефон)

Список сотрудников (ФИО, Должность, Стаж, email)

Должности сотрудников (Должность, Ставка)

Типы заявок, обрабатываемых управляющей компанией (Тип заявки, Приоритет)

Информацию о заявках (Номер Заявки, Тип заявки, Дата создания, Текущий статус, Регистратор, Исполнитель, Дата закрытия)

Статусы заявок (Статус заявки, Порядок)

Историю заявки, т. е. ее перемещение по статусам и обновление описания заявки (ID, Заявка, Статус заявки, Дата, Исполнитель)

## 1.3 Анализ существующих решений для предметной области

Найти существующие решений для данной предметной области не удалось, т. к. подобные решения являются закрытыми и используются внутри самих управляющих компаний, соответственно, доступ к подобным системам есть только у сотрудников управляющей компании на определенной должности.

## 1.4 Разработка общей структуры ИС

**Архитектура клиент-сервер**

*Архитектура клиент-сервер* предназначена для разрешения проблем файл-серверных приложений путем разделения компонентов приложения и размещения их там, где они будут функционировать наиболее эффективно. Особенностью архитектуры клиент-сервер является использование выделенных серверов баз данных, понимающих запросы на языке структурированных запросов SQL (Structured Query Language) и выполняющих поиск, сортировку и агрегирование информации.

Отличительная черта серверов БД — наличие справочника данных, в котором записана структура БД, ограничения целостности данных, форматы и даже серверные процедуры обработки данных по вызову или по событиям в программе. Объектами разработки в таких приложениях помимо диалога и логики обработки являются, прежде всего, реляционная модель данных и связанный с ней набор SQL-операторов для типовых запросов к базе данных.

Большинство конфигураций клиент-сервер использует двухуровневую модель, в которой клиент обращается к услугам сервера. Предполагается, что диалоговые компоненты PS и PL размещаются на клиенте, что позволяет обеспечить графический интерфейс. Компоненты управления данными DS и FS размещаются на сервере, а диалог (PS, PL), логика BL и DL — на клиенте. Двухуровневое определение архитектуры клиент-сервер использует именно этот вариант: приложение работает у клиента, СУБД — на сервере.

*Хранимая процедура* — процедура с операторами SQL для доступа к БД, вызываемая по имени с передачей требуемых параметров и выполняемая на сервере БД. Хранимые процедуры могут компилироваться, что повышает скорость их выполнения и сокращает нагрузку на сервер. Хранимые процедуры улучшают целостность приложений и БД, гарантируют актуальность коллективно используемых операций и вычислений. Улучшается сопровождение таких процедур, а также безопасность (нет прямого доступа к данным).

Следует помнить, что перегрузка хранимых процедур прикладной логикой может перегрузить сервер, что приведет к потере производительности. Эта проблема особенно актуальна при разработке крупных информационных систем, в которых к серверу может одновременно обращаться большое количество клиентов. Поэтому в большинстве случаев следует принимать компромиссные решения: часть логики приложения размещать на стороне сервера, часть — на стороне клиента. Такие клиент-серверные системы называются системами с разделенной логикой. Данная схема при удачном разделении логики позволяет получить более сбалансированную загрузку клиентов и сервера, но при этом затрудняется сопровождение приложений.

Создание архитектуры клиент-сервер возможно и на основе многотерминальной системы. В этом случае в многозадачной среде сервера приложений выполняются программы пользователей, а клиентские узлы вырождены и представлены терминалами. Подобная схема информационной системы характерна для UNIX. В настоящее время архитектура клиент-сервер получила признание и широкое распространение как способ организации приложений для рабочих групп и информационных систем корпоративного уровня. Подобная организация работы повышает эффективность выполнения приложений за счет использования возможностей сервера БД, разгрузки сети и обеспечения контроля целостности данных. Двухуровневые схемы архитектуры клиент-сервер могут привести к некоторым проблемам в сложных информационных приложениях с множеством пользователей и запутанной логикой. Решением этих проблем может стать использование многоуровневой архитектуры.

2 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 2.1 Инфологическое проектирование

Инфологическое проектирование основано на ER-методе.

Суть ER-метода: построение ER-диаграммы, отображающей в графической форме основные объекты предметной области, связи между ними и определение характеристик этих связей.

Затем, по определенным правилам выполняется переход от ER-диаграммы к таблицам БД. Результатом является схема БД.

В рамках ER-метода существует два типа диаграмм:

1.ER-диаграммы для экземпляров сущностей и связей.

2.ER-диаграммы для классов сущностей и связей.

Связь между сущностями имеет две характеристики:

* Степень или кратность связи:

1:1 – один к одному

1:N – один ко многим

N:M – многие ко многим

* Класс принадлежности сущности к связи:

Обязательный – каждый экземпляр одной сущности связан с конкретным экземпляром другой сущности, в противном случае класс сущности является необязательным.

***Правила формирования отношений по ER-диаграммам.***

Формирование отношений для бинарных связей 1:1.

**Правило 1:**

Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей к связи обязательный, то формируют одно отношение. Ключом может быть ключ любой сущности.

**Правило 2:**

Если связь 1:1 и класс принадлежности одной сущности обязательный, а другой необязательный, то формируется два отношения, по одному для каждой сущности, где ключом является ключ соответствующей сущности. Кроме этого ключ сущности с необязательным классом принадлежности добавляется в качестве атрибута в отношение для сущности с обязательным классом принадлежности.

**Правило 3:**

Если степень связи 1:1 и класс принадлежности обеих сущностей к связи необязательный, то формируют три отношения: два для сущностей и одно для связи. В первых двух отношениях ключом является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. В этом отношении ключом может быть ключ любой сущности.

**Формирование отношений для бинарных связей 1:N**

В такой ситуации используются два правила. Каждое из них определяется классом принадлежности N-связной сущности. Класс принадлежности односвязной сущности на результат не влияет.

**Правило 4:**

Если степень бинарной связи 1:N и класс принадлежности N-связной сущности обязательный, то формируется два отношения, по одному для каждой сущности. Ключ отношения - это ключ соответствующей сущности. Кроме этого в отношение для N-связной сущности добавляется в качестве атрибута ключ односвязной сущности.

**Правило 5:**

Если степень бинарной связи 1:N, а класс принадлежности N-связной сущности необязательный, то формируют три отношения: по одному для каждой сущности и одно для связи. Ключом в первых двух отношениях является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. Ключ этого отношения –это ключ N-связной сущности.

**Формирование отношений для бинарных связей N:M**

Используется одно правило. Класс принадлежности сущности к связи на результат не влияет.

**Правило 6:**

Если степень бинарной связи N:M, то формируется три отношения: два отношения для сущностей, где ключом является ключ соответствующей сущности и одно отношение для связи, где первичный ключ состоит из ключей обеих сущностей.

**Формирование отношений для тернарных связей.**

**Правило 7:**

При наличии тернарной связи необходимо использовать 4 отношения: три для каждой сущности, где ключом является ключ соответствующей сущности и одно для связи, где ключ содержит ключи всех сущностей. При наличии n-ой связи строится n+1 отношение.

Основные понятия ER-метода:

* Сущность – это объект предметной области.
* Экземпляр сущности – конкретный представитель объекта.
* Связь – соединение между двумя и более сущностями.
* Экземпляр связи – конкретная связь между экземплярами объектов.
* Ключ сущности – множество атрибутов для однозначной идентификации сущности.

**Проектирование базы данных с помощью ER-метода включает следующие этапы:**

## 2.1.1 Выделение сущностей и связей в заданной предметной области.

* Дом (Адрес)
* Квартира (Код Квартиры)
* Физ. лицо (ФИО)
* Сотрудники (ФИО)
* Должность (Наименование)
* Заявки (Номер)
* Типы заявок (Тип)
* Статусы заявок (Статус)
* История заявки (ID)

В предметной области можно выделить следующие связи между сущностями:

1. Дом содержит Квартиры
2. Квартирой владеет/проживает Физ. лицо
3. Физ. лицо является Сотрудником
4. Сотрудник имеет Должность
5. Должность обрабатывает Типы заявок
6. Заявка имеет Тип заявки
7. Заявка имеет Статус заявки
8. Заявку обрабатывает Сотрудник (Исполнитель)
9. Заявка имеет Историю заявки
10. История заявки содержит Статус заявки
11. Физ. лицо подает Заявку

## 2.1.2 Построение ER-диаграмм

Диаграммы должны включать все выделенные сущности и связи между ними. На данном этапе должны быть определены характеристики связи, а именно степень связи и класс принадлежности сущности к связи.

1. Связь Дом содержит Квартиры

Для степени связи:

* В Доме может быть много Квартир
* Квартира может быть только в одном Доме

Для класса принадлежности сущности к связи:

* Дом обязательно содержит Квартиры
* Квартира обязательно есть в Доме



Рисунок 2.1.2.1 - Связь Дом содержит Квартиры

2. Связь Квартирой владеет/проживает Физ. лицо

Для степени связи:

* Квартирой может владеть/проживать несколько Персон
* Физ. лицо может владеть несколькими Квартирами

Для класса принадлежности сущности к связи:

* Квартирой не обязательно владеет/проживает Физ. лицо
* Физ. лицо не обязательно владеет/проживает в Квартире

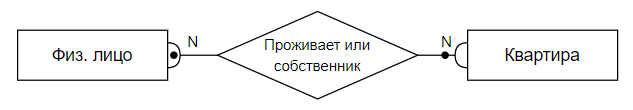


Рисунок 2.1.2.2 - Связь Квартирой владеет/проживает Физ. лицо

3. Связь Физ. лицо является Сотрудником

Для степени связи:

* Одна Физ. лицо может являться одним Сотрудником
* Один Сотрудник является Физ. Лицом

Для класса принадлежности сущности к связи:

* Физ. лицо не обязательно является Сотрудником
* Сотрудник обязательно является Физ. лицом



Рисунок 2.1.2.3 - Связь Физ. лицо является Сотрудником

4. Связь Сотрудник имеет Должность

Для степени связи:

* Один Сотрудник имеет одну Должность
* Одну Должность могут иметь много Сотрудников

Для класса принадлежности сущности к связи:

* Сотрудник обязательно имеет Должность
* Должность не обязательно занимает какой-либо Сотрудник

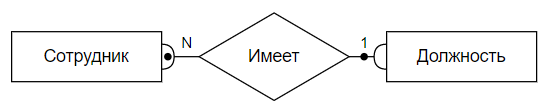


Рисунок 2.1.2.4 - Связь Сотрудник имеет Должность

5. Связь Должность обрабатывает Типы заявок

Для степени связи:

* Должность может обрабатывать разные Типы заявок
* Тип заявки обрабатывается одной Должностью

Для класса принадлежности сущности к связи:

* Должность не обязательно обрабатывает Тип заявки
* Тип заявки обязательно обрабатывается Должностью

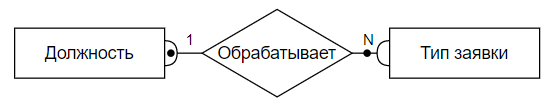


Рисунок 2.1.2.5 - Связь Должность обрабатывает Типы заявок

6. Связь Заявка имеет Тип заявки

Для степени связи:

* Заявка имеет один Тип заявки
* Тип заявки могут иметь много Заявок

Для класса принадлежности сущности к связи:

* Заявка обязательно имеет Тип заявки
* Тип заявки не обязательно имеет какая-либо Заявка

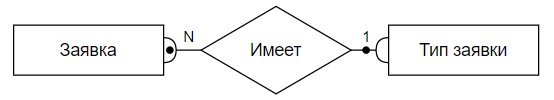


Рисунок 2.1.2.6 - Связь Заявка имеет Тип заявки

7. Связь Заявка имеет Статус заявки

Для степени связи:

* Заявка имеет один Статус заявки
* Статус заявки могут иметь много Заявок

Для класса принадлежности сущности к связи:

* Заявка обязательно имеет Статус заявки
* Статус заявки не обязательно имеет какая-либо Заявка

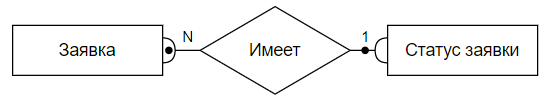


Рисунок 2.1.2.7 - Связь Заявка имеет Статус заявки

8. Связь Заявку обрабатывает Сотрудник (Исполнитель)

Для степени связи:

* Заявка может иметь много Исполнителей
* Исполнитель может обслуживать много Заявок

Для класса принадлежности сущности к связи:

* Заявка не обязательно имеет Исполнителя
* Исполнитель не обязательно обслуживает Заявку

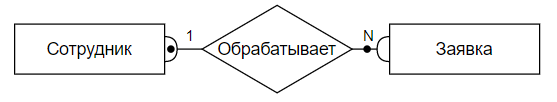


Рисунок 2.1.2.8 - Заявку обрабатывает Сотрудник (Исполнитель)

9. Связь Заявка имеет Историю заявки

Для степени связи:

* Заявка может иметь много Историй заявки
* История заявки может быть связана только с одной Заявкой

Для класса принадлежности сущности к связи:

* Заявка обязательно имеет Историю заявки
* История заявки содержит Заявку

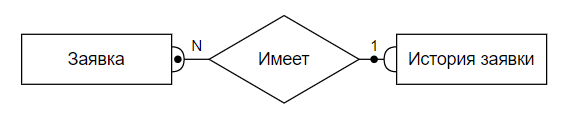


Рисунок 2.1.2.9 - Связь Заявка имеет Историю заявки

10. Связь История заявки содержит Статус заявки

Для степени связи:

* История заявки связана со многими Статусами заявок
* Статус заявки связан со многими Историями заявок

Для класса принадлежности сущности к связи:

* История заявки обязательно содержит Статус заявки
* Статус заявки не обязательно связан с Историей заявки



Рисунок 2.1.2.10 - Связь История заявки содержит Статус заявки

11. Связь Физ. лицо подает Заявку

Для степени связи:

* Физ. лицо может подать много Заявок
* Заявка подается одним Физ. лицом

Для класса принадлежности сущности к связи:

* Физ. лицо не обязательно подает Заявку
* Заявка обязательно подается Физ. лицом

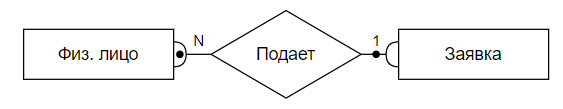


Рисунок 2.1.2.11 - Связь Физ. лицо подает Заявку

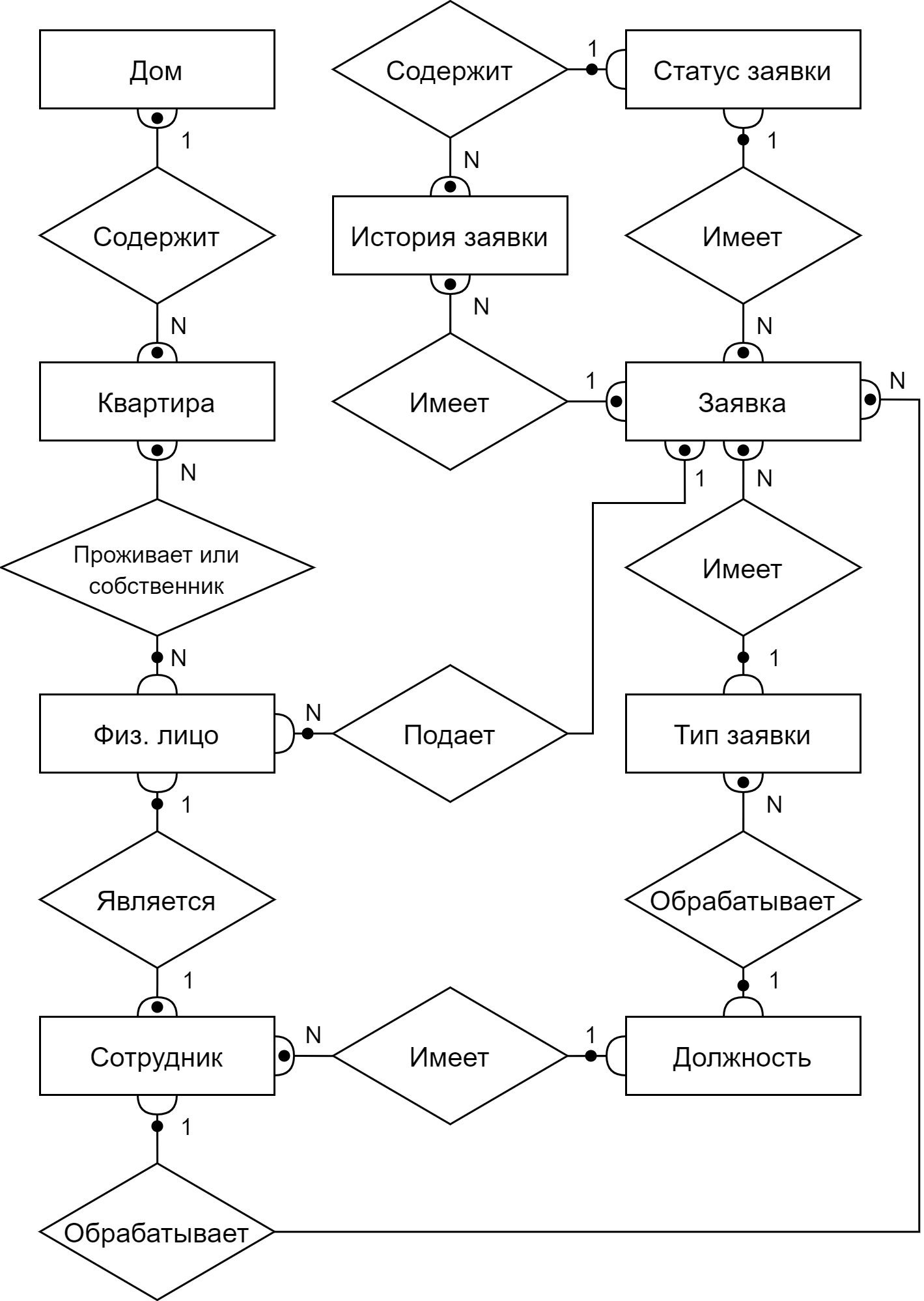


Рисунок 2.1.2.12 – Общая ER-диаграмма

# 2.2 Даталогическое проектирование БД

## 2.2.1 Переход от ER-диаграммы к предварительным отношениям

Даталогическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов.

Для реляционной БД проектирование логической структуры заключается в том, чтобы разбить всю информацию по таблицам, определив состав полей для каждой из этих таблиц.

***Формирование предварительных отношений по ER-диаграмме:***

Связь Дом содержит Квартиры.

По правилу 4 получаем следующие отношения:

Дом (Адрес)

Квартира (Код Квартиры, Адрес)

Связь Квартирой владеет/проживает Физ. Лицо.

По правилу 6 получаем следующие отношения:

Квартира (Код Квартиры, Адрес)

Физ. лицо (ФИО)

Квартира – Физ. Лицо (Код Квартиры, ФИО)

Связь Физ. лицо является Сотрудником

По правилу 2 получаем следующие отношения:

Физ. лицо (ФИО)

Сотрудник (ФИО, Должность)

Связь Сотрудник имеет Должность

По правилу 4 получаем следующие отношения:

Сотрудник (ФИО, Должность)

Должность (Должность)

Связь Должность обрабатывает Типы заявок

По правилу 4 получаем следующие отношения:

Должность (Должность)

Тип заявки (Тип заявки, Должность)

Связь Заявка имеет Тип заявки

По правилу 4 получаем следующие отношения:

Тип заявки (Тип заявки, Должность)

Заявка (Номер Заявки, Тип заявки)

Связь Заявка имеет Статус заявки

По правилу 4 получаем следующие отношения:

Заявка (Номер Заявки, Тип заявки, Статус заявки)

Статус заявки (Статус заявки)

Связь Заявку обрабатывает Сотрудник (Исполнитель)

По правилу получаем следующие отношения:

Заявка (Номер Заявки, Тип заявки, Статус заявки, Исполнитель)

Связь Заявка имеет Историю заявки

По правилу 4 получаем следующие отношения:

Заявка (Номер Заявки, Тип заявки, Статус заявки, Исполнитель)

История заявки (ID, Заявка)

Связь История заявки содержит Статус заявки

По правилу получаем следующие отношения:

История заявки (ID, Заявка, Статус заявки)

Связь Физ. лицо подает Заявку

По правилу получаем следующие отношения:

Заявка (Номер Заявки, Тип заявки, Регистратор, Статус заявки, Исполнитель)

## 2.2.2 Подготовка списка всех представляющих интерес атрибутов, кроме тех, которые уже были выделены на ER-диаграмме классов

Для формирования окончательных отношений необходимы следующие атрибуты:

1. Кол-во Квартир
2. Кол-во Подъездов
3. Подъезд
4. Этаж
5. Номер квартиры
6. Паспорт
7. Телефон
8. Стаж
9. Email
10. Ставка
11. Приоритет
12. Дата создания
13. Текущий статус
14. Регистратор
15. Исполнитель
16. Дата закрытия
17. Порядок
18. Примечание

## 

## 2.2.3 Распределение атрибутов из списка, причем каждый атрибут помещается только в одно из отношений.

Таким образом, получаются следующие отношения:

Дом (Адрес, Кол-во Квартир, Кол-во Подъездов)

Квартира (Код Квартиры, Адрес, Подъезд, Этаж, Номер квартиры)

Физ. Лицо (ФИО, Паспорт, Телефон)

Квартира – Физ. Лицо (Код Квартиры, ФИО)

Сотрудник (ФИО, Должность, Стаж, email)

Должность (Должность, Ставка)

Тип заявки (Тип заявки, Приоритет)

Тип Заявки – Должность (Тип заявки, Должность)

Заявка (Номер Заявки, Тип заявки, Дата создания, Текущий статус, Регистратор, Исполнитель, Дата закрытия, Примечание)

Статус заявки (Статус заявки, Порядок)

История заявки (ID, Заявка, Статус заявки, Дата, Исполнитель, Примечание)

## 2.2.4 Проверка отношений на БКНФ

Существует два метода проектирования базы данных: это ER-метод и метод нормальных форм. В настоящее время используются оба метода. ER-метод непосредственно для проектирования базы данных, а метод нормальных форм для проверки правильности результата проектирования.

Метод нормальных форм предполагает, что вся информация первоначально хранится в одном отношении. Процесс проектирования заключается в переводе отношения из первой нормальной формы в более высокие нормальные формы по определенным правилам. Каждой нормальной форме соответствует определенный набор ограничений. Выделяют следующие нормальные формы:

* 1НФ (первая нормальная форма)
* 2НФ (вторая нормальная форма)
* 3НФ (третья нормальная форма)
* БКНФ (нормальная форма Бойса-Кодда)
* 4НФ (четвертая нормальная форма)
* 5НФ (пятая нормальная форма или проекционно-соединительная нормальная форма)
* ДКНФ (доменно-ключевая нормальная форма)

Каждая нормальная форма сохраняет свойства предыдущих нормальных форм. Переход к более высокой нормальной форме выполняется путем декомпозиции отношения на два или более отношений, которые удовлетворяют требованиям этой нормальной формы.

**1НФ** - отношение находится в 1НФ, если на пересечении каждой строки и столбца находится ровно одно значение.

**2НФ** – отношение находится в 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от любого потенциального ключа (в частности первичного)

**3НФ** – отношение находится в 3НФ, если оно находится в 2НФ и в нем нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа (в частности первичного)

**БКНФ** – отношение находится в БКНФ,если оно находится в 3НФ и детерминанты всех функциональных зависимостей являются потенциальными ключами.

**4НФ** – отношение находится в 4НФ, если оно находится в БКНФ и не содержит нетривиальных многозначных зависимостей.

**5НФ** – отношение находится в 5НФ, если оно находится в 4НФ и любая многозначная зависимость соединения в ней является тривиальной. Пятая нормальная форма в большей степени является теоретическим исследованием и практически не применяется при реальном проектировании баз данных.

**ДКНФ** – отношение находится в ДКНФ, если не имеет аномалий модификации. Другими словами, что бы ни менялось — ничего не потеряется, если соблюдены все ограничения относительно ключей и доменов. Формулировка слишком общая, но суть ее заключается в том, что если выполнять некоторые правила, то при любых действиях с таблицей ее целостность не пострадает и вся необходимая информация сохранится.

Более высокая нормальная форма не всегда предпочтительней в процессе проектирования базы данных, так как чем выше нормальная форма, тем больше таблиц в базе данных и тем больше потребуется выполнить операций соединения для получения окончательного результата. Следовательно, тем медленней информационная система будет реагировать на запросы пользователя. В данной курсовой работе база данных находится в нормальной форме Бойса-Кодда.

На данном этапе проектирования базы данных графически изображены функциональные зависимости атрибутов для каждого отношения в отдельности, причем ключевые атрибуты изображены прямоугольниками, а не ключевые овалами.

Рассмотрим отношение Дом*:*

Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

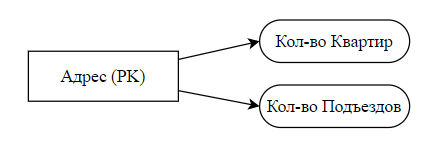


Рисунок 2.2.4.1 – отношение Дом

Рассмотрим отношение Квартира:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

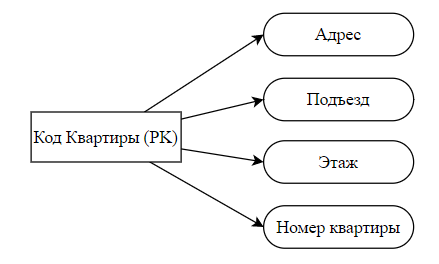


Рисунок 2.2.4.2 – отношение Квартира

Рассмотрим отношение Физ. Лицо:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

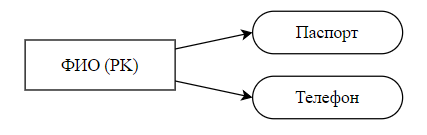


Рисунок 2.2.4.3 – отношение Физ. Лицо

Рассмотрим отношение Квартира – Физ. Лицо:

Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение. Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от потенциального ключа. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является потенциальный ключ.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.2.4.4 – отношение Квартира – Физ. Лицо

Рассмотрим отношение Сотрудник:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

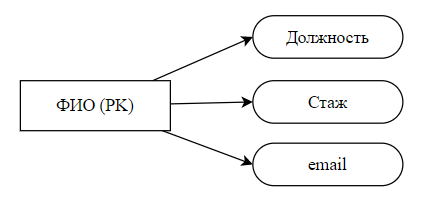


Рисунок 2.2.4.5 – отношение Сотрудник

Рассмотрим отношение Должность:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

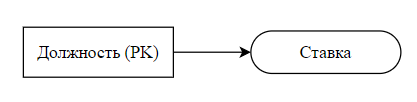


Рисунок 2.2.4.6 – отношение Должность

Рассмотрим отношение Тип заявки:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

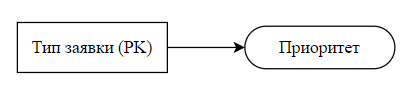


Рисунок 2.2.4.7 – отношение Тип заявки

Рассмотрим отношение Тип Заявки – Должность:

Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение. Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от потенциального ключа. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является потенциальный ключ.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.2.4.8 – отношение Тип Заявки – Должность

Рассмотрим отношение Статус заявки:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

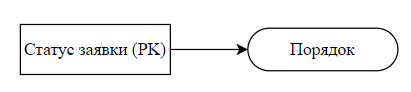


Рисунок 2.2.4.9 – отношение Статус заявки

Рассмотрим отношение Заявка:

Отношение находится в 1НФ, так как на пересечении строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

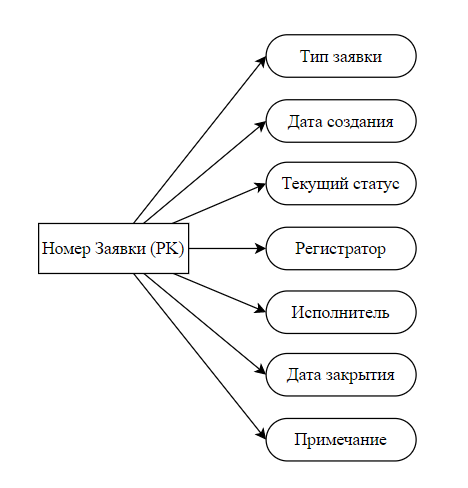


Рисунок 2.2.4.10 – отношение Заявка

Рассмотрим отношение История заявки*:*

Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение. Отношение находится в 2НФ, так как первичный ключ является простым. Отношение находится в 3НФ, так как в нем нет транзитивных зависимостей. Отношение находится в БКНФ, потому что детерминантом функциональной зависимости является первичный ключ.

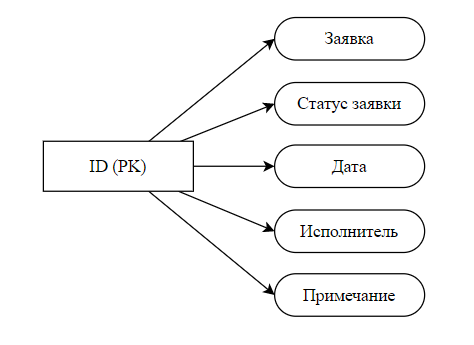


Рисунок 2.2.4.11 – отношение История заявки

## 2.2.5 Пересмотр ER-диаграммы

Данный этап проектирования базы данных выполняется в двух случаях:

а) некоторые отношения не находятся в БКНФ

б) некоторым атрибутам не находится логически обоснованных мест в предварительных отношениях

Все отношения находятся в БКНФ и все атрибуты логически распределены в отношениях.

## 2.2.6 Построение схемы базы данных

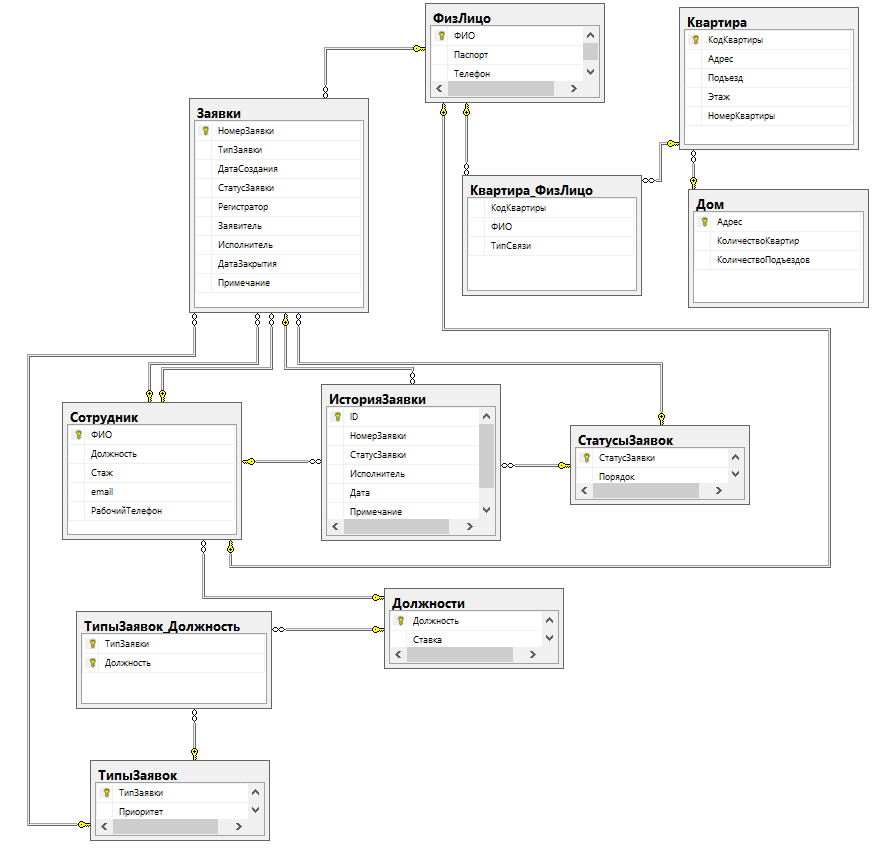


Рисунок 2.2.6.1 – Общая ER-диаграмма

# 2.3 Программирование объектов БД

## 2.3.1 Разработка хранимых процедур

Создание процедуры вставки дома:

create proc InsertHome @Адрес nvarchar(250), @КоличествоКвартир int, @КоличествоПодъездов int

as

begin

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

values (@Адрес, @КоличествоКвартир, @КоличествоПодъездов)

end;

go

Создание процедуры удаления дома:

create proc deleteHome @Адрес nvarchar(250)

as

begin

delete from Дом where Адрес = @Адрес

end;

go

Создание процедуры обновления дома:

create proc updateHome @Адрес nvarchar(250), @КоличествоКвартир int, @КоличествоПодъездов int

as

begin

UPDATE Дом

SET КоличествоКвартир = @КоличествоКвартир,

КоличествоПодъездов = @КоличествоПодъездов

WHERE Адрес = @Адрес

end;

go

Создание процедуры вставки заявки:

create proc insertIssue @ТипЗаявки nvarchar(150),

@Регистратор nvarchar(250),

@Заявитель nvarchar(250) = null,

@Примечание nvarchar(250) = null

as

begin

insert into Заявки (ТипЗаявки, Регистратор, Заявитель, Примечание)

values (@ТипЗаявки, @Регистратор, @Заявитель, @Примечание)

end;

go

Создание процедуры обновления заявки:

create proc updateIssue @НомерЗаявки int,

@СтатусЗаявки nvarchar(50),

@Исполнитель nvarchar(250) = null,

@Примечание nvarchar(250) = null,

@ДатаЗакрытия datetime = null

as

begin

UPDATE Заявки

SET СтатусЗаявки = @СтатусЗаявки,

Исполнитель = @Исполнитель,

Примечание = @Примечание,

ДатаЗакрытия = @ДатаЗакрытия

WHERE НомерЗаявки = @НомерЗаявки

end;

go

Создание процедуры удаления заявки:

create proc deleteIssue @НомерЗаявки int

as

begin

delete from ИсторияЗаявки where НомерЗаявки = @НомерЗаявки

delete from Заявки where НомерЗаявки = @НомерЗаявки

end;

go

Создание процедуры вставки квартиры:

create proc insertApart @Адрес nvarchar(250),

@Подъезд int,

@Этаж int,

@НомерКвартиры int

as

begin

insert into Квартира(Адрес, Подъезд, Этаж, НомерКвартиры)

values (@Адрес, @Подъезд, @Этаж, @НомерКвартиры)

end;

go

Создание процедуры обновления квартиры:

create proc updateApart @КодКвартиры int,

@Адрес nvarchar(250),

@Подъезд int,

@Этаж int,

@НомерКвартиры int

as

begin

update Квартира

set НомерКвартиры = @НомерКвартиры,

Адрес = @Адрес,

Подъезд = @Подъезд,

Этаж = @Этаж

where КодКвартиры = @КодКвартиры

end;

go

Создание процедуры удаления квартиры:

create proc deleteApart @КодКвартиры int

as

begin

delete from Квартира where КодКвартиры = @КодКвартиры

end;

go

Создание процедуры вставки физ. лица:

create proc InsertPerson @ФИО nvarchar(250), @Паспорт nvarchar(12), @Телефон nvarchar(12)

as

begin

INSERT INTO ФизЛицо(ФИО, Паспорт, Телефон)

values (@ФИО, @Паспорт, @Телефон)

end;

go

Создание процедуры обновления физ. лица:

create proc updatePerson @ФИО nvarchar(250), @Паспорт nvarchar(12), @Телефон nvarchar(12)

as

begin

update ФизЛицо

set Паспорт = @Паспорт, Телефон = @Телефон

where ФИО = @ФИО

end;

go

Создание процедуры удаления физ. лица:

create proc deletePerson @ФИО nvarchar(250)

as

begin

delete from ФизЛицо where ФИО = @ФИО

end;

go

Создание процедуры вставки должности:

create proc InsertRole @Должность nvarchar(250), @Ставка money

as

begin

INSERT INTO Должности(Должность, Ставка)

values (@Должность, @Ставка)

end;

go

Создание процедуры вставки сотрудника:

create proc InsertEmployee @ФИО nvarchar(250), @Должность nvarchar(250), @Стаж date, @email nvarchar(100), @РабочийТелефон nvarchar(12)

as

begin

INSERT INTO Сотрудник (ФИО, Должность, Стаж, email, РабочийТелефон)

values (@ФИО, @Должность, @Стаж, @email, @РабочийТелефон)

end;

go

Создание процедуры обновления сотрудника:

create proc updateEmployee @ФИО nvarchar(250), @Должность nvarchar(250), @Стаж date, @email nvarchar(100), @РабочийТелефон nvarchar(12)

as

begin

update Сотрудник

set Должность = @Должность, Стаж = @Стаж, email = @email, РабочийТелефон = @РабочийТелефон

where ФИО = @ФИО

end;

go

Создание процедуры удаления сотрудника:

create proc deleteEmployee @ФИО nvarchar(250)

as

begin

delete from Сотрудник where ФИО = @ФИО

end;

go

Создание процедуры вставки жильца:

create proc InsertTenant @КодКвартиры int, @ФИО nvarchar(250), @ТипСвязи nvarchar(250)

as

begin

INSERT INTO Квартира\_ФизЛицо (КодКвартиры, ФИО, ТипСвязи)

values (@КодКвартиры, @ФИО, @ТипСвязи)

end;

go

Создание процедуры обновления жильца:

create proc updateTenant @КодКвартиры int, @ФИО nvarchar(250), @ТипСвязи nvarchar(250)

as

begin

update Квартира\_ФизЛицо

set КодКвартиры = @КодКвартиры, ТипСвязи = @ТипСвязи

where ФИО = @ФИО

end;

go

Создание процедуры удаления жильца:

create proc deleteTenant @ФИО nvarchar(250)

as

begin

delete from Квартира\_ФизЛицо where ФИО = @ФИО

end;

go

## 2.3.2 Разработка таблиц, триггеров и представлений.

Текст создания таблиц, триггеров и представлений находится в приложении 1.

3 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

# 3.1 Выбор программных компонентов клиентской части

Разработка клиентской части информационной системы производилась в среде Microsoft Visual Studio 2017. В качестве языка программирования был выбран язык высокого уровня С#.

# 3.2 Разработка интерфейса пользователя

## 3.2.1 Разработка форм

Пользователю предоставляется главная форма для работы:

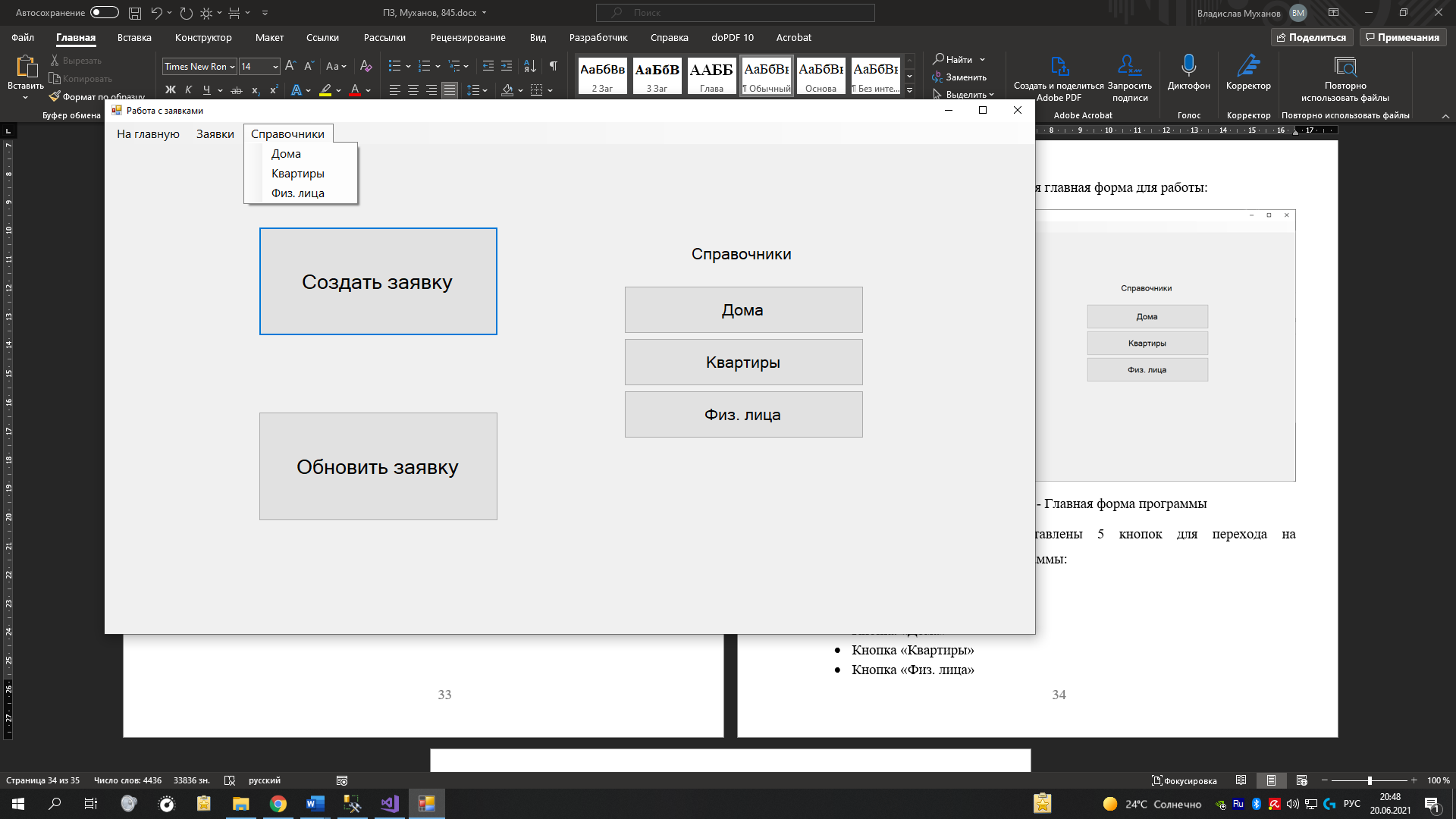


Рисунок 3.2.1.1 - Главная форма программы

На главной странице (рис. 3.2.1.1) представлены 5 кнопок для перехода на соответствующие страницы программы:

* Кнопка «Создать заявку»
* Кнопка «Обновить заявку»
* Кнопка «Дома»
* Кнопка «Квартиры»
* Кнопка «Физ. лица»

На панели навигации, доступной на любой из страниц программы, есть кнопка «На главную», для возврата на главную страницу, представленную на рисунке 3.2.1.1, и два выпадающих пункта меню, дублирующих функционал кнопок главной страницы.

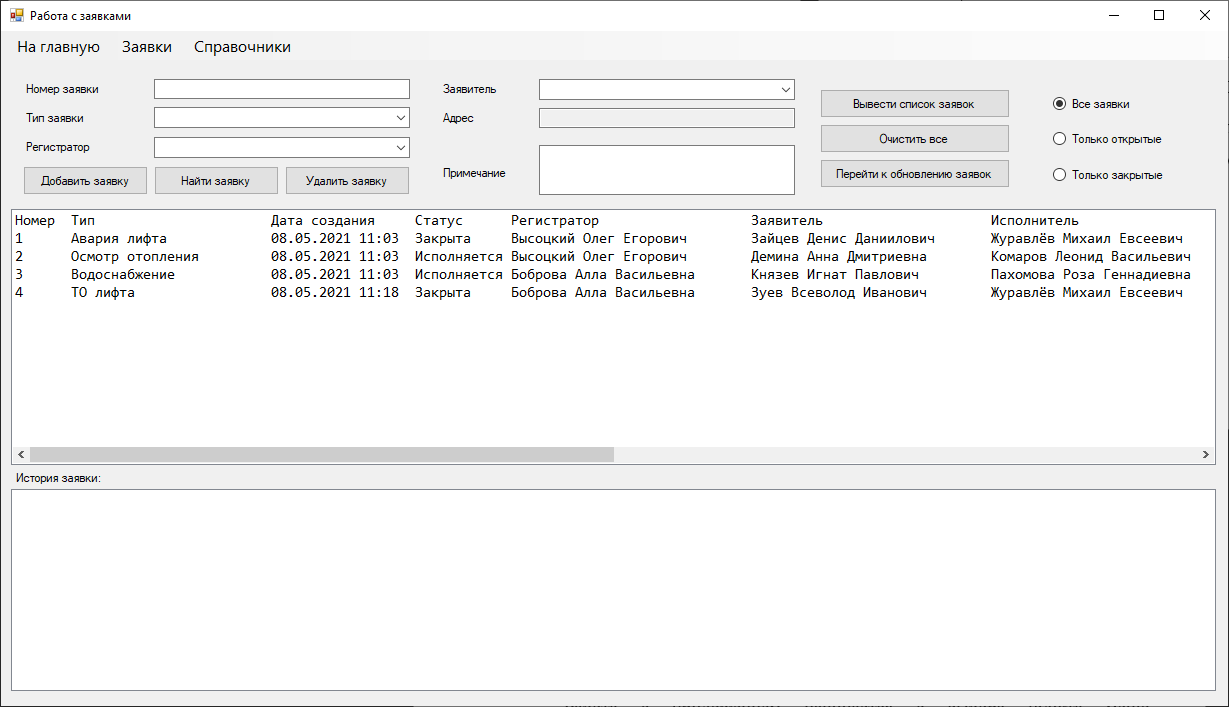


Рисунок 3.2.1.2 – Страница создания заявок

На странице создания заявок (рис. 3.2.1.2) пользователю выводится список всех существующих заявок. Пользователь может:

* Добавить новую заявку, указав вручную или выбрав из выпадающего меню тип заявки, регистратора, заявителя и примечание к заявке. Примечание указывать не обязательно.
* Найти заявку по номеру.
* Удалить заявку по номеру.
* Вывести только открытые, только закрытые или все заявки.

При клике по конкретной заявке в нижней части формы выводится история заявки (рис. 3.2.1.3), т. е. все ее перемещения по статусам, тексты примечаний и даты обновления статусов. Также данные заявки автоматически подставляются в соответствующие текстовые поля для удобства работы.

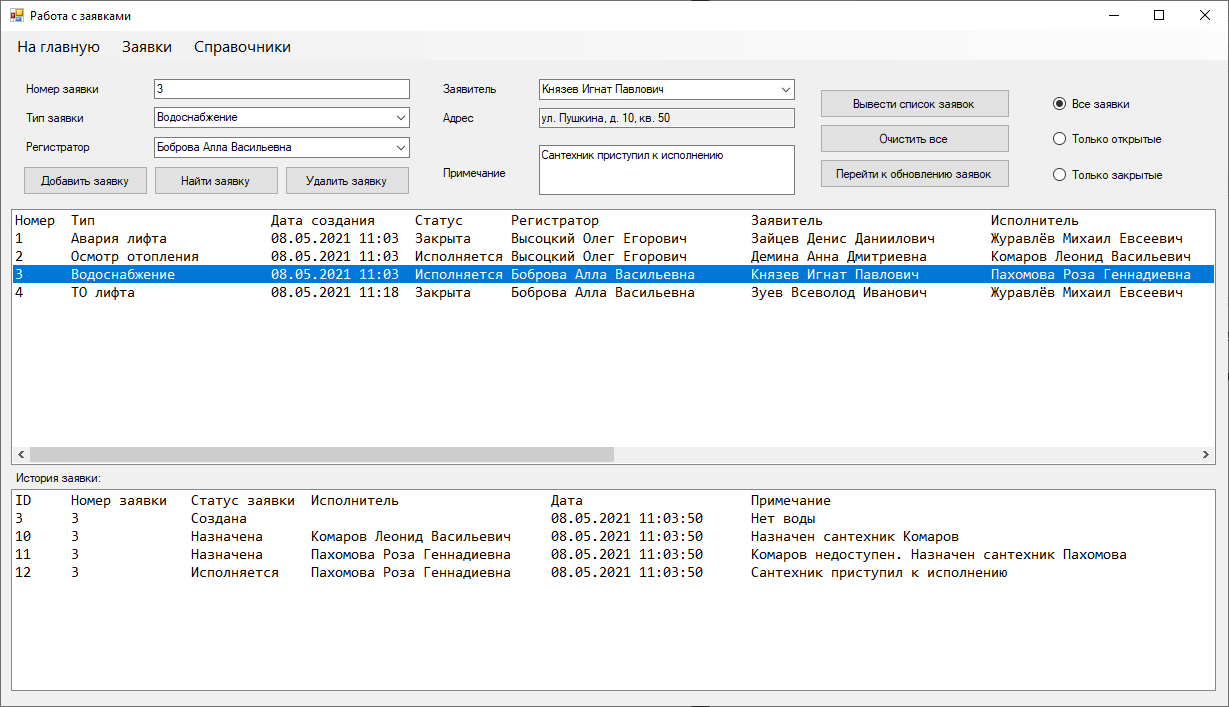


Рисунок 3.2.1.3 – История заявки

Кнопка «Перейти к обновлению заявок» переведет на аналогичную страницу со списком заявок, где пользователь может обновить заявку, поменяв ей статус или примечание и назначив исполнителя.

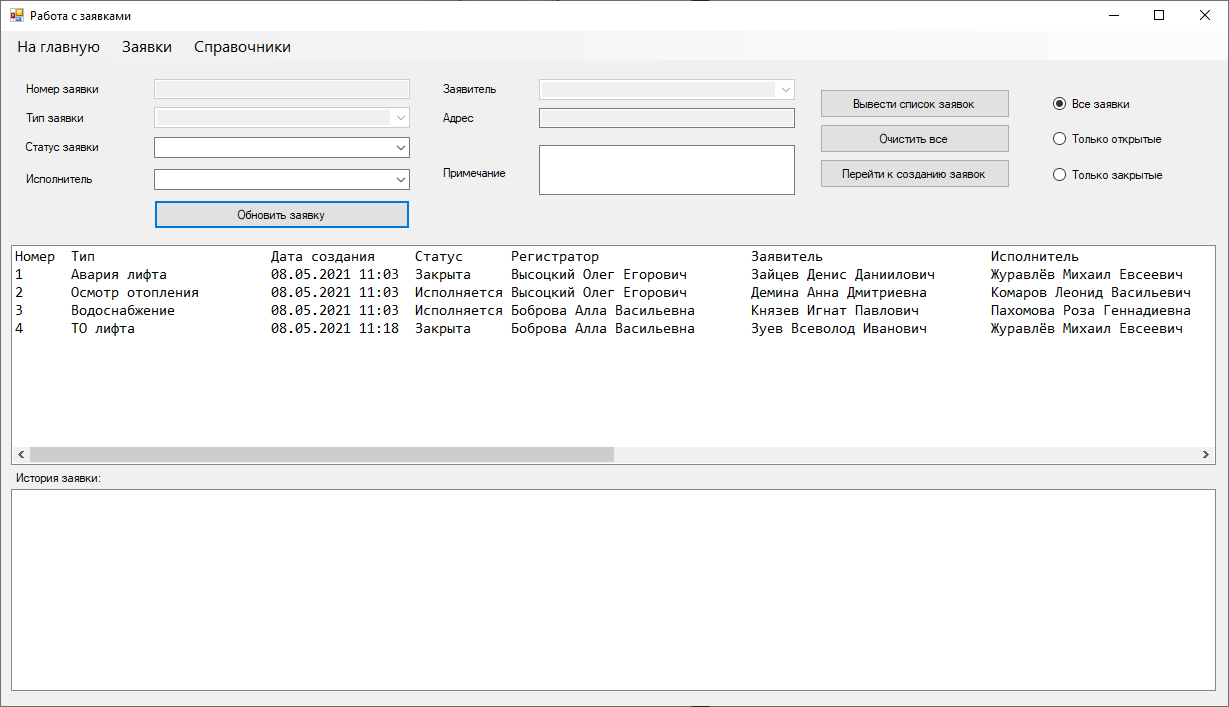


Рисунок 3.2.1.4 – Страница обновления заявок

На странице обновления заявок (рис. 3.2.1.4) пользователю выводится список всех существующих заявок. Пользователь может обновить заявку, кликнув на нее и поменяв информацию в доступных текстовых полях.

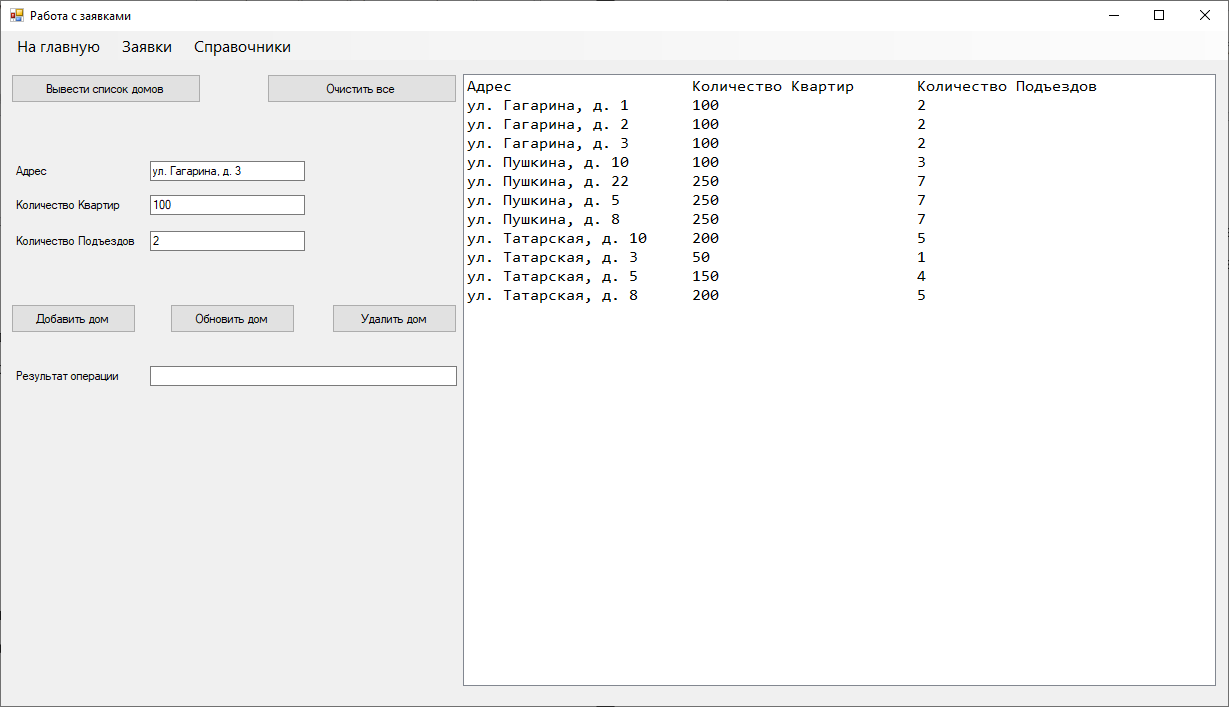


Рисунок 3.2.1.5 – Страница со справочником домов

На странице справочника домов (рис. 3.2.1.5) пользователь может добавить новый дом, обновить информацию о существующем доме или удалить из БД определенный дом.

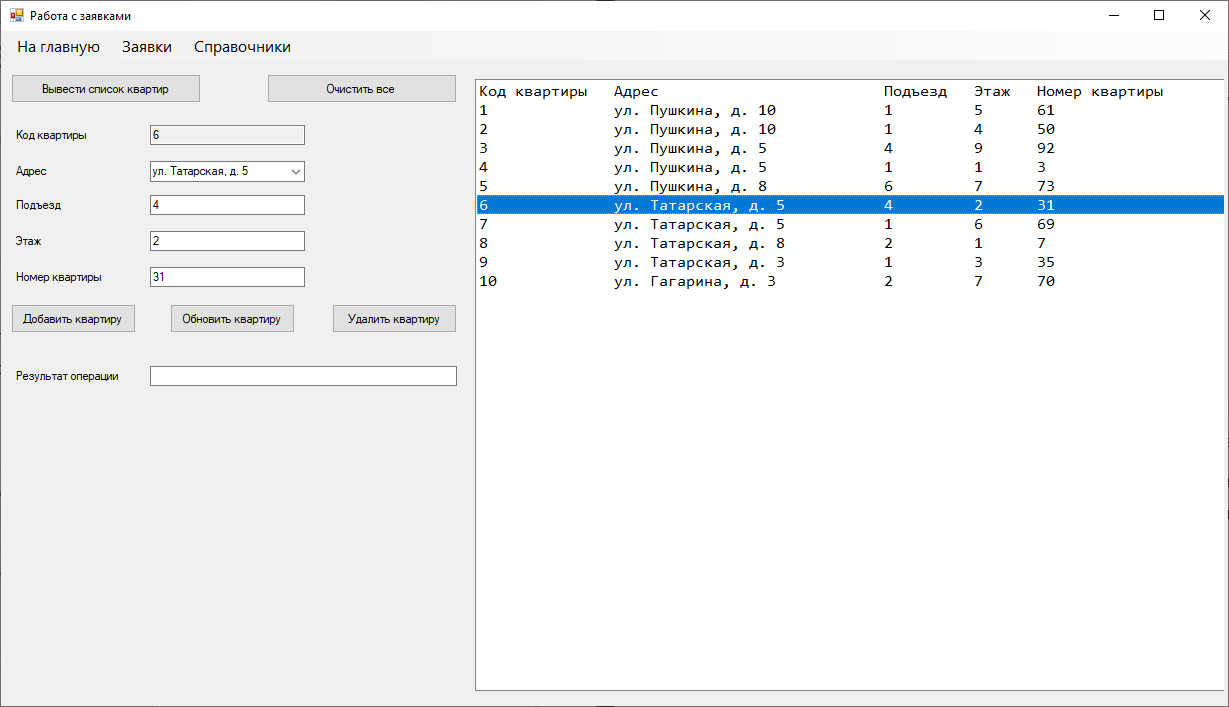


Рисунок 3.2.1.6 – Страница со справочником квартир

На странице справочника квартир (рис. 3.2.1.6) пользователь может добавить новую квартиру в один из существующих домов, обновить информацию о квартире или удалить из БД определенную квартиру.

На странице справочника физ. лиц (рис. 3.2.1.7) можно увидеть полную информацию о каждом физ. лице. Пользователь может:

* Добавить новое физ. лицо, указав ФИО, паспорт и номер телефона
* Удалить физ. лицо по ФИО
* Обновить любую информацию о физ. лице
* Добавить новую должность, указав ее название и оклад
* Добавить нового сотрудника, указав его ФИО (из существующих фил. Лиц), должность, стаж, email и рабочий телефон
* Удалить сотрудника по ФИО
* Обновить любую информацию о сотруднике
* Добавить нового жильца по ФИО и номеру квартиры
* Удалить жильца по ФИО
* Обновить любую информацию о жильце

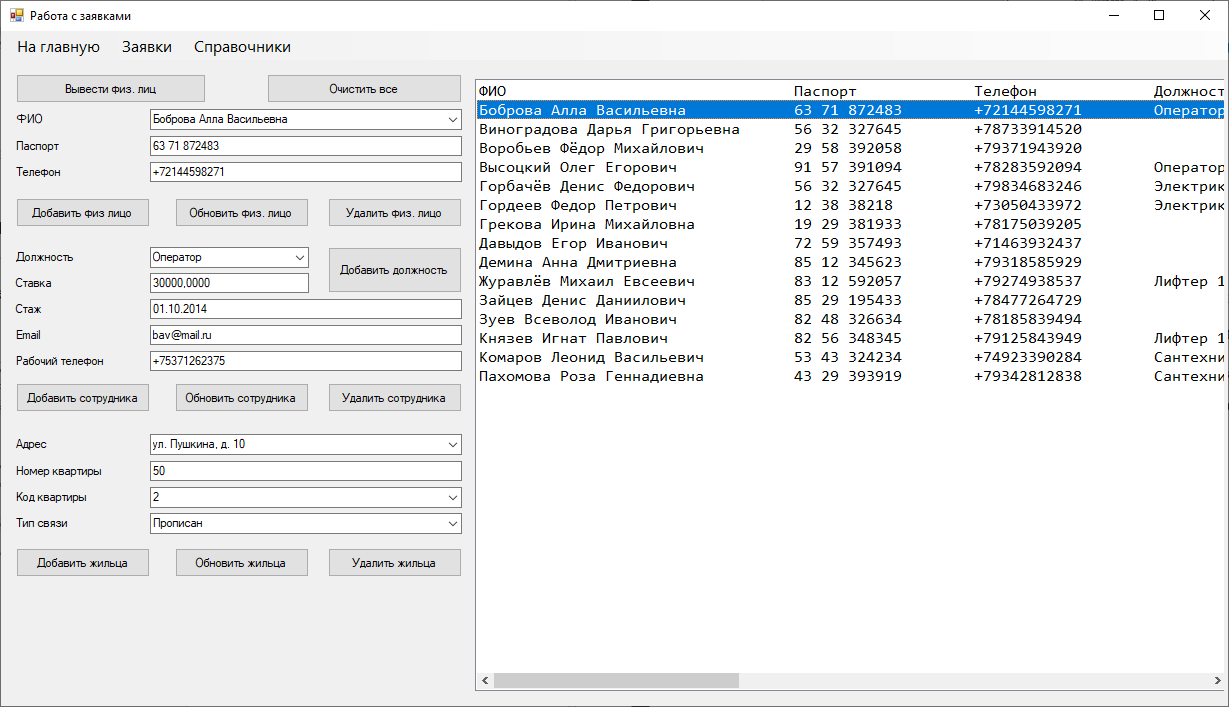


Рисунок 3.2.1.7 – Страница со справочником физ. лиц

При наведении курсора на одну из кнопок добавления (рис. 3.2.1.8), обновления или удаления для наглядности подсвечиваются ключевые поля, данные из которых будут использованы при выполнении данной операции.

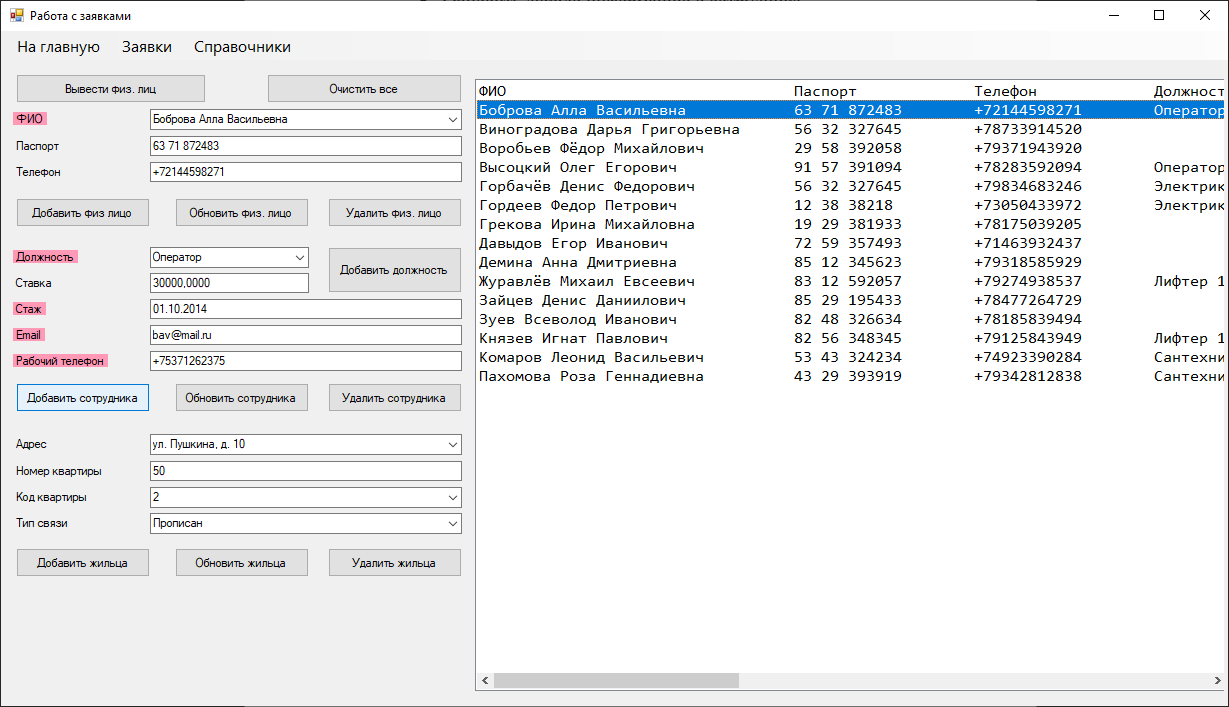


Рисунок 3.2.1.8 – Наведение на кнопку «Добавить сотрудника»

# 3.3 Разработка сценария инсталляции клиентской программы

Для создания установочного файла использовалась бесплатно распространяемая программа «Smart Install Maker 5.04». Рассмотрим основные этапы создания установщика.

Задание основных характеристик инсталлятора:

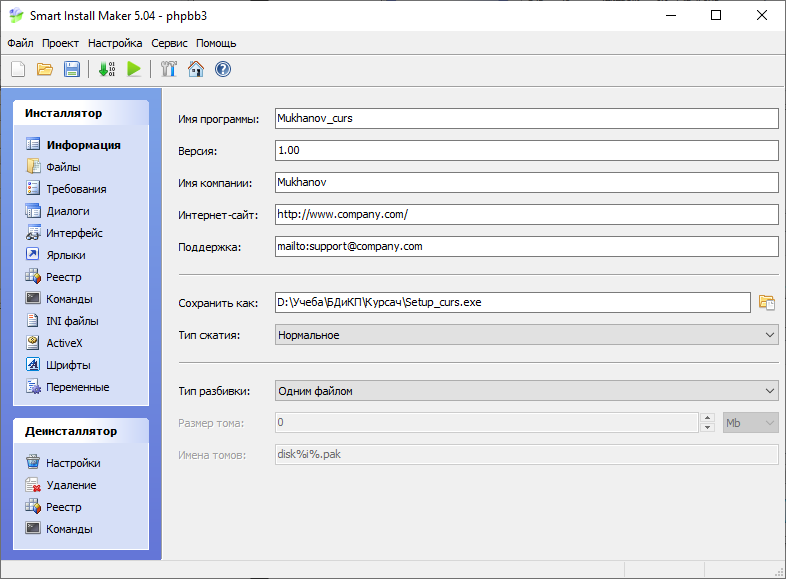


Рисунок 3.3.1 – Параметры создания

Добавление всех необходимых файлов:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.3.2 - Файлы

Выполнение компиляции проекта:

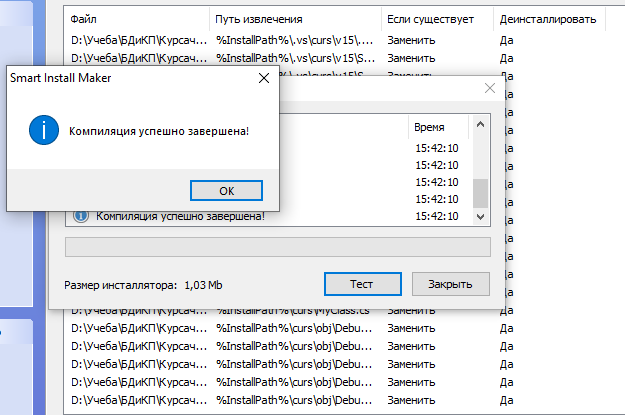


Рисунок 3.3.3 – Успешное завершение компиляции

В результате создан файл Setup\_curs.exe, содержащий установщик разработанного клиентского приложения.

Произведём проверку работоспособности созданного инсталлятора:

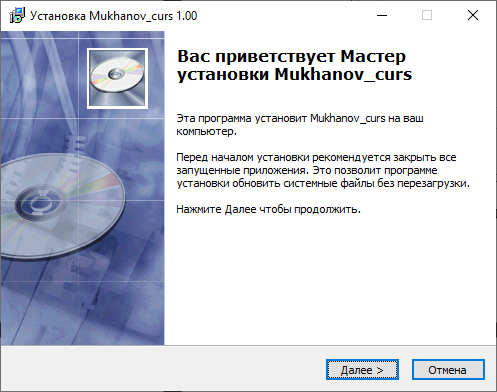


Рисунок 3.3.4 – Запуск инсталлятора

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.3.5 – Выбор пути для установки

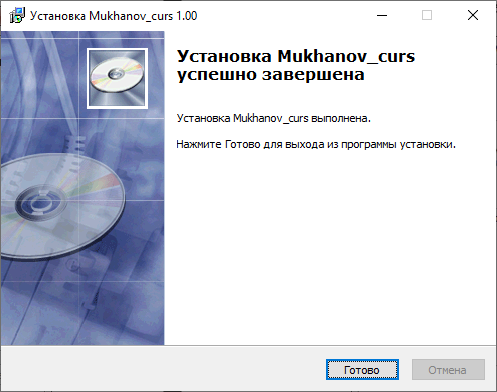


Рисунок 3.3.6 – Успешное завершение установки

Проверим работоспособность установленного приложения:

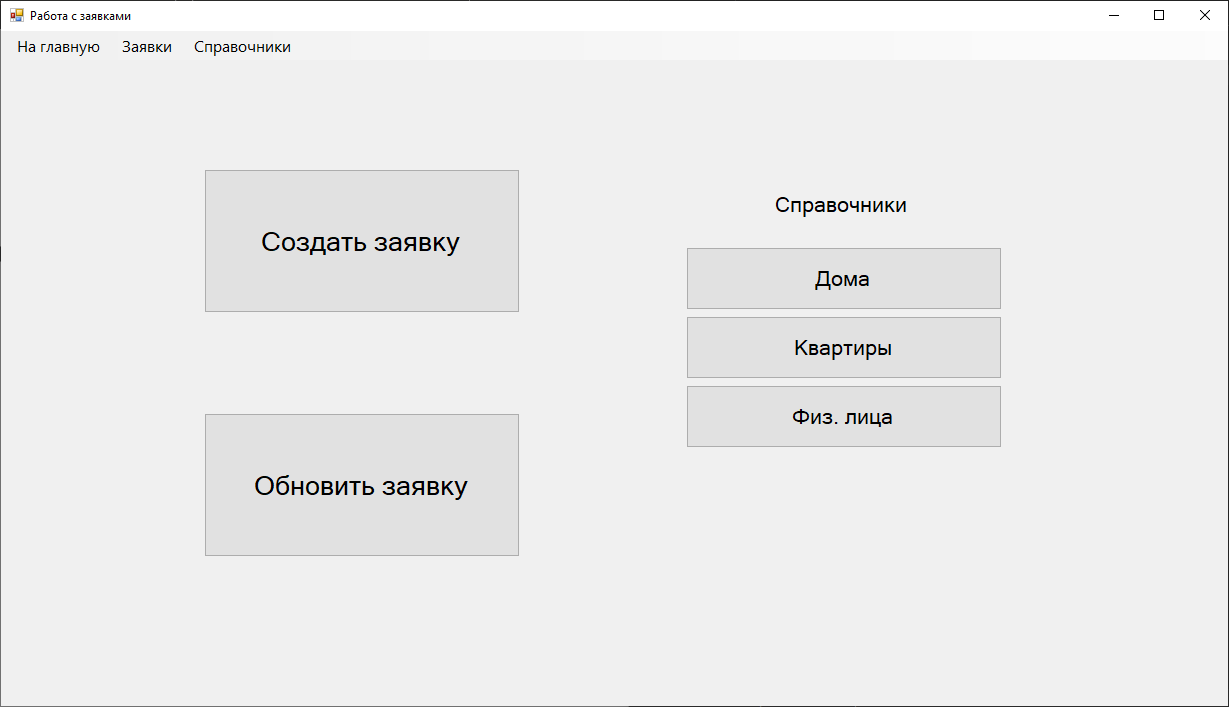


Рисунок 3.3.7 – Работоспособное установленное приложение

Помимо всех необходимых файлов для работы приложения пакет установщика создаёт и деинсталлятор:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.3.8 – Деинсталлировать приложение

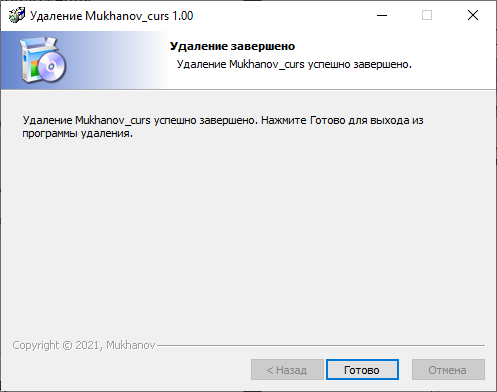


Рисунок 3.3.9 – Завершение удаления программы

Таким образом, был создан полностью рабочий пакет установки разработанного приложения.

Проделаем аналогичные шаги в программе Inno Setup 6.

Добавление всех необходимых файлов в инсталлятор:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.3.10 – Добавление файлов в инсталлятор

Завершение установки:

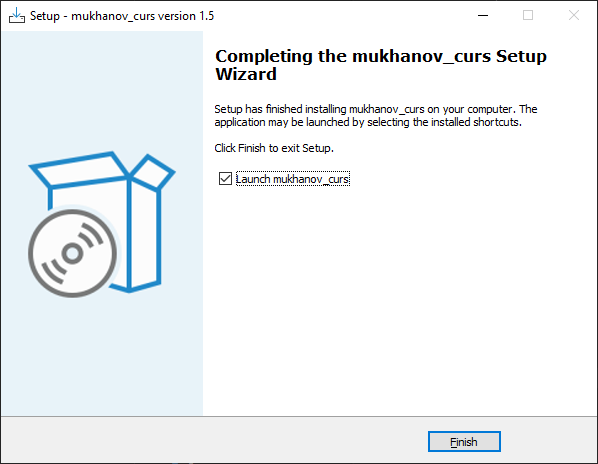


Рисунок 3.3.11 – Успешное завершение установки

4 РАЗРАБОТКА ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

**4.1 Описание программной единицы**

**Общие сведения:**

Наименование программы: curs.exe.

Для функционирования программы требуется SQL Management Studio c подключенной базой данных.

Языки программирования, на которых написана программа: С#.

**Функциональное назначение:**

Программа используется для создания и управления заявками пользователей, а также для хранения и управления информацией о сотрудниках и жильцах, домах и квартирах.

**Используемые технические средства:**

Персональный компьютер под управлением операционной системы Windows 10

**Вызов и загрузка:**

Способ вызова программы с соответствующего носителя данных: запуск файла curs.exe

**Входные данные:**

Числовые и строковые данные, относящиеся к заявкам, физ. лицам, сотрудникам, домам и квартирам.

**Выходные данные:**

Результаты выполнения операций вставки, удаления, обновления и вывод таблиц и представлений.

5 ТЕСТИРОВАНИЕ ИС

# 5.1 Тестирование объектов БД

## 5.1.1 Тестирование частных ограничений целостности данных

Тестирование частных ограничений целостности данных для отношения Дом:

1) Тест «Добавление дома»

Текст запроса:

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

values ('ул. Колотушкина, д. 17', 200, 6)

select \* from Дом where Адрес = 'ул. Колотушкина, д. 17'

Исходные данные:

* 'ул. Колотушкина, д. 17', 200, 6.

Ожидаемый результат:

* В отношении Дом появится запись.

Фактический результат:

* Фактический результат теста «Добавление дома» представлен на рисунке 5.1.1.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.1.1.1 – Фактический результат теста «Добавление дома»

2) Тест «Добавление дома с повторяющимся адресом»

Текст запроса:

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

values ('ул. Колотушкина, д. 17', 200, 6)

Исходные данные:

* 'ул. Колотушкина, д. 17', 200, 6.

Ожидаемый результат:

* Нарушение ограничения первичного ключа.

Фактический результат:

* Фактический результат теста «Добавление дома с повторяющимся адресом» представлен на рисунке 3.1.2



Рисунок 5.1.1.2 – Фактический результат теста «Добавление дома с повторяющимся адресом»

3) Тест «Добавление дома с NULL наименованием»

Текст запроса:

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

values (null, 200, 6)

Исходные данные:

* null, 200, 6.

Ожидаемый результат:

* Ошибка добавления.

Фактический результат:

* Фактический результат теста «Добавление дома с NULL наименованием» представлен на рисунке 5.1.1.3

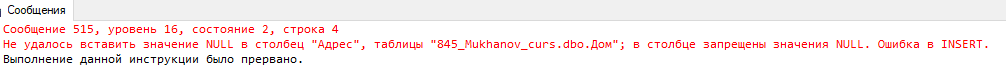


Рисунок 5.1.1.3 – Фактический результат теста «Добавление военкомата с NULL наименованием»

4) Тест «Проверка умолчания»

Текст запроса:

insert into ТипыЗаявок (ТипЗаявки)

values ('Тестовый тип')

select \* from ТипыЗаявок where ТипЗаявки = 'Тестовый тип'

Ожидаемый результат:

* Запись добавится с умолчанием «1» в столбце «Приоритет»

Фактический результат:

* Фактический результат теста «Проверка умолчания» представлен на рисунке 5.1.1.4

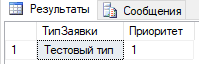


Рисунок 5.1.1.4 – Фактический результат теста «Проверка умолчания»

## 5.1.2 Тестирование представлений

Тестирование представления «ТипыЗаявок\_Должность\_Сотрудники»:

1) Тест «Создание представления»

Код запроса:

create view ТипыЗаявок\_Должность\_Сотрудники

as

select ТипыЗаявок\_Должность.ТипЗаявки, Сотрудник.ФИО

from ТипыЗаявок\_Должность

inner join Сотрудник on Сотрудник.Должность = ТипыЗаявок\_Должность.Должность

go

Ожидаемый результат: в базе данных появится представление «ТипыЗаявок\_Должность\_Сотрудники», в котором появятся следующая информация:

* Тип заявки
* ФИО

Фактический результат: создано представление «ТипыЗаявок\_Должность\_Сотрудники» (рис. 5.1.2.1).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.1.2.1 – Фактический результат теста «Создание представления»

## 5.1.3 Тестирование ХП и триггеров

Тестирование хранимой процедуры «InsertHome»:

1) Тест «Создание ХП»

Текст запроса:

create proc InsertHome @Адрес nvarchar(250), @КоличествоКвартир int, @КоличествоПодъездов int

as

begin

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

values (@Адрес, @КоличествоКвартир, @КоличествоПодъездов)

end;

Ожидаемый результат:

* Хранимая процедура будет создана в БД.

Фактический результат:

* Хранимая процедура создана в БД

2) Тест «Использование ХП»

Текст запроса:

exec InsertHome 'ул. Колотушкина, д. 18', 150, 7

select \* from Дом where Адрес = 'ул. Колотушкина, д. 18'

Ожидаемый результат:

* В отношении Дом появится запись.

Фактический результат:

* Фактический результат теста «Использование ХП» представлен на рисунке 5.1.3.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.1.3.1 – Фактический результат теста «Использование ХП»

Тестирование триггера «TR\_Update\_Заявки»:

1) Тест «Создание триггера»

Текст запроса:

CREATE TRIGGER TR\_Update\_Заявки on Заявки

AFTER UPDATE

AS

insert into ИсторияЗаявки (НомерЗаявки, СтатусЗаявки, Исполнитель, Дата, Примечание)

select i.НомерЗаявки, i.СтатусЗаявки, i.Исполнитель, SYSDATETIME(), i.Примечание from inserted i;

GO

Ожидаемый результат:

* Триггер будет создан в БД.

Фактический результат:

* Триггер создан в БД

2) Тест «Проверка работы триггера»

Текст запроса:

exec insertIssue 'Уборка территории', 'Боброва Алла Васильевна', 'Зуев Всеволод Иванович', 'Тестовая заявка'

exec updateIssue 6, 'Закрыта', null, 'Тестовая заявка'

select \* from ИсторияЗаявки where НомерЗаявки = 6

Ожидаемый результат:

* В отношении ИсторияЗаявки будет 2 записи по заявке с номером 6.

Фактический результат:

* Фактический результат теста «Проверка работы триггера» представлен на рисунке 5.1.3.1

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 5.1.3.2 – Фактический результат теста «Проверка работы триггера»

# 5.2 Тестирование клиентской части

Для проверки работоспособности программы было проведено тестирование всех её функциональных возможностей.

Запуск программы (рис. 5.2.1):

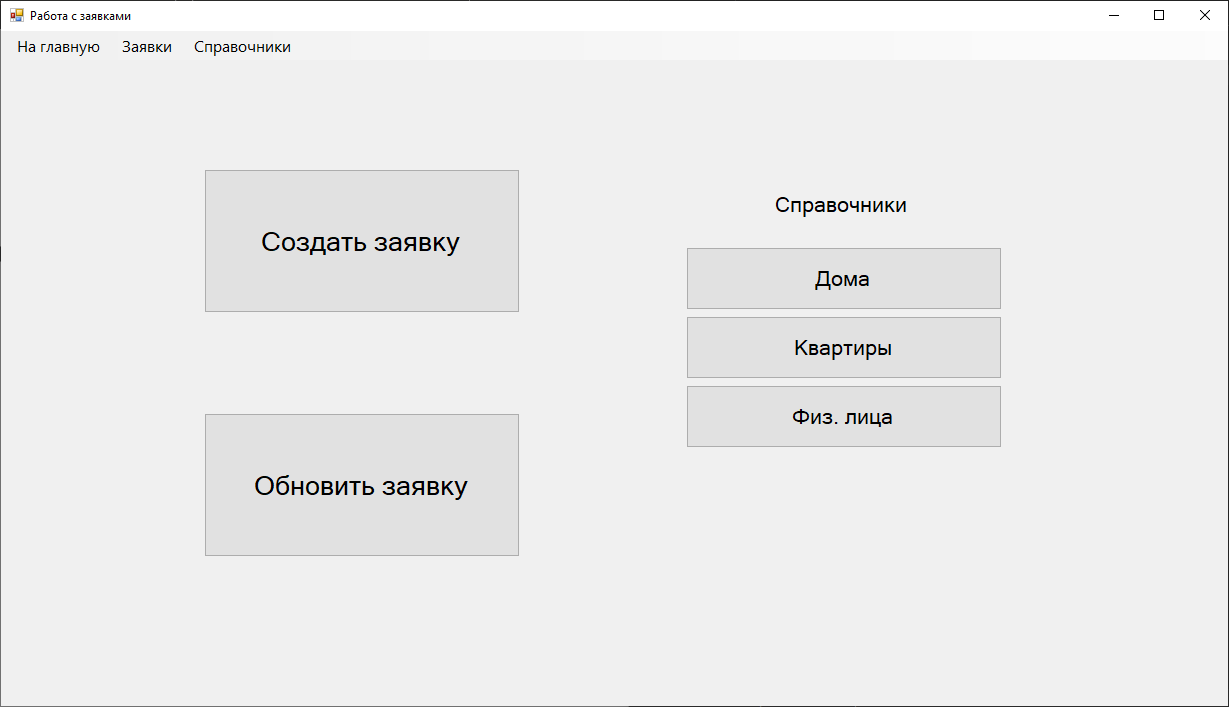


Рисунок 5.2.1 – Запущенное приложение

Проверим основной функционал программы. Создадим заявку. Введем необходимые данные (рис. 5.2.2):

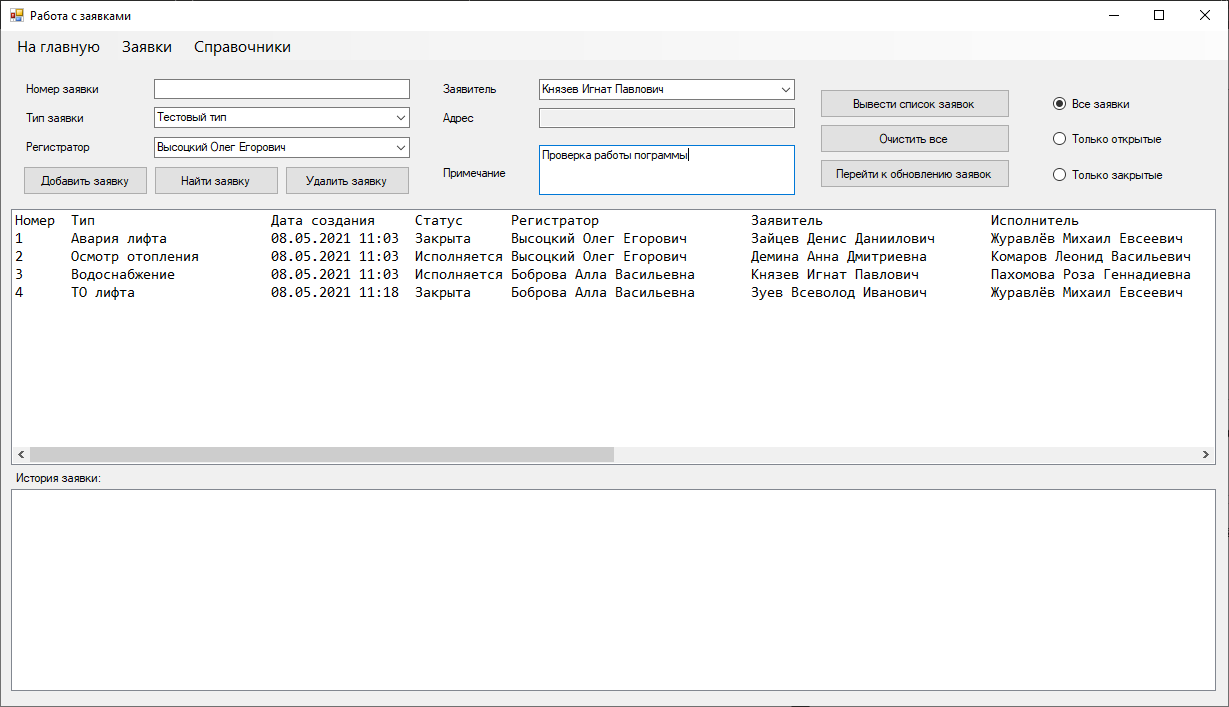


Рисунок 5.2.2 – Ввод данных о заявке

Создадим заявку кнопкой «Добавить заявку» (рис. 5.2.3):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.2.3 – Заявка создана

Перейдем к обновлению заявки и введем новые данные (рис. 5.2.4).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.2.4 – Ввод новых данных о заявке

Обновим заявку (рис. 5.2.5).

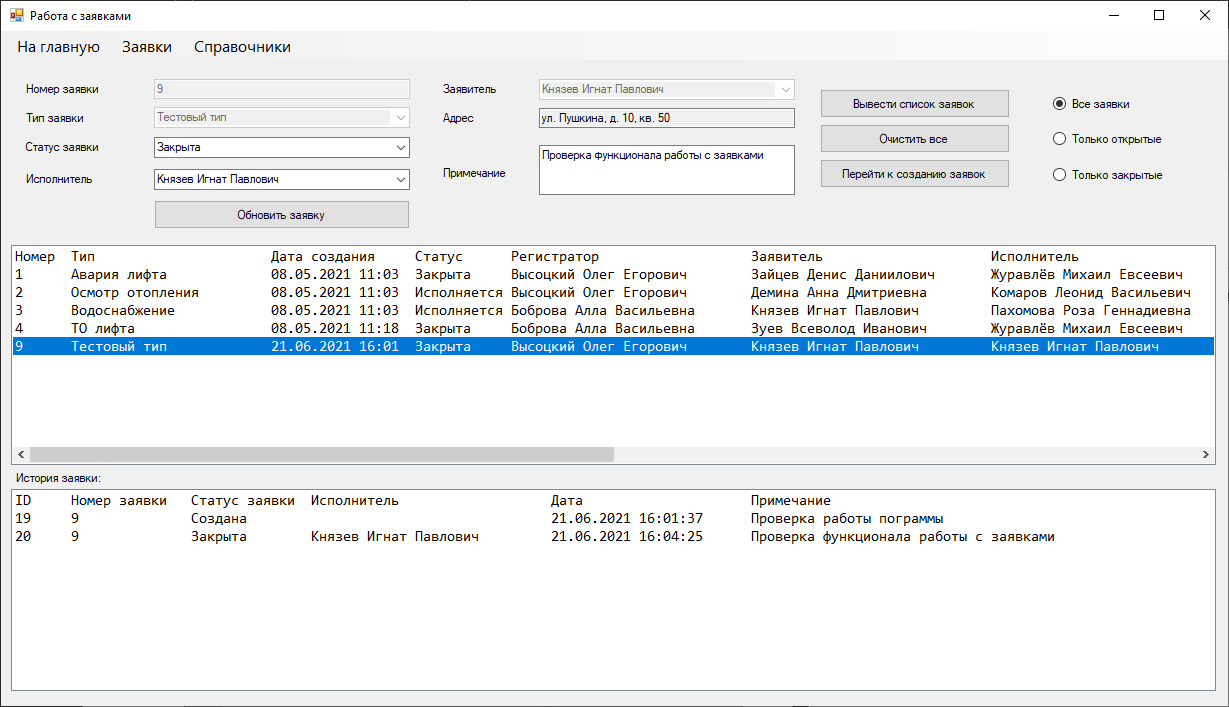


Рисунок 5.2.5 – Обновленная заявка

Удалим заявку (рис. 5.2.6).

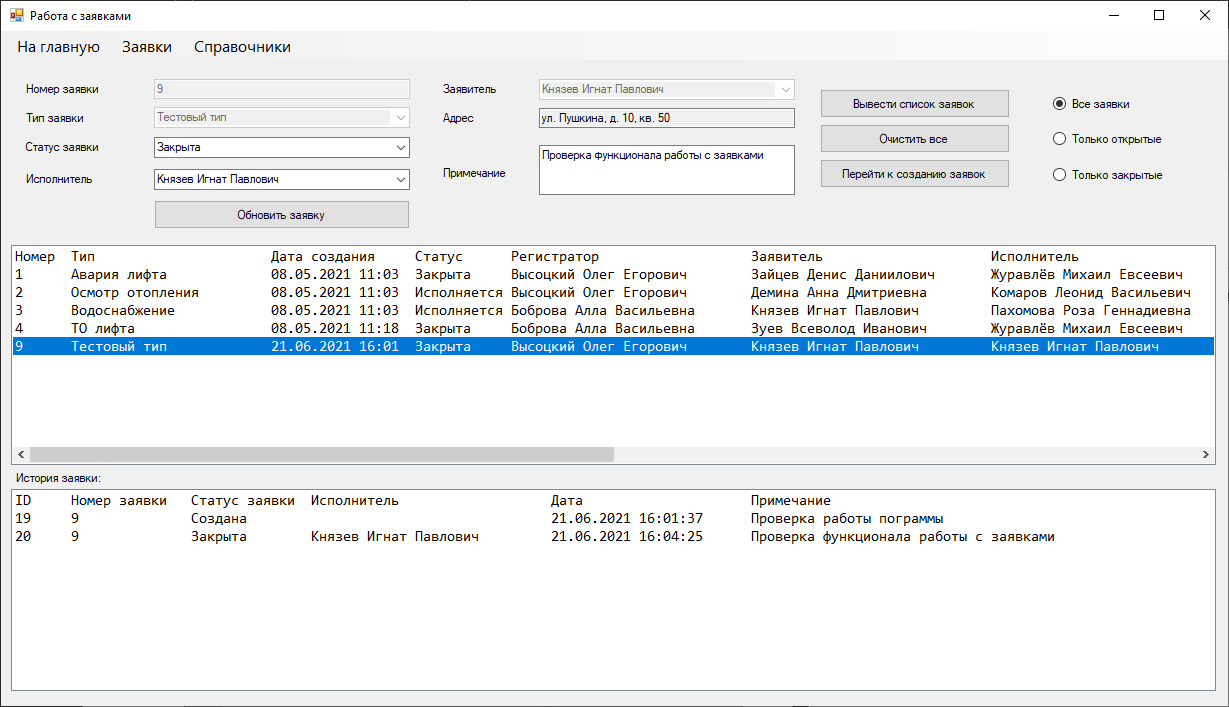


Рисунок 5.2.6 – Заявка удалена

Проверим работу кнопки «Очистить все» (рис. 5.2.7).

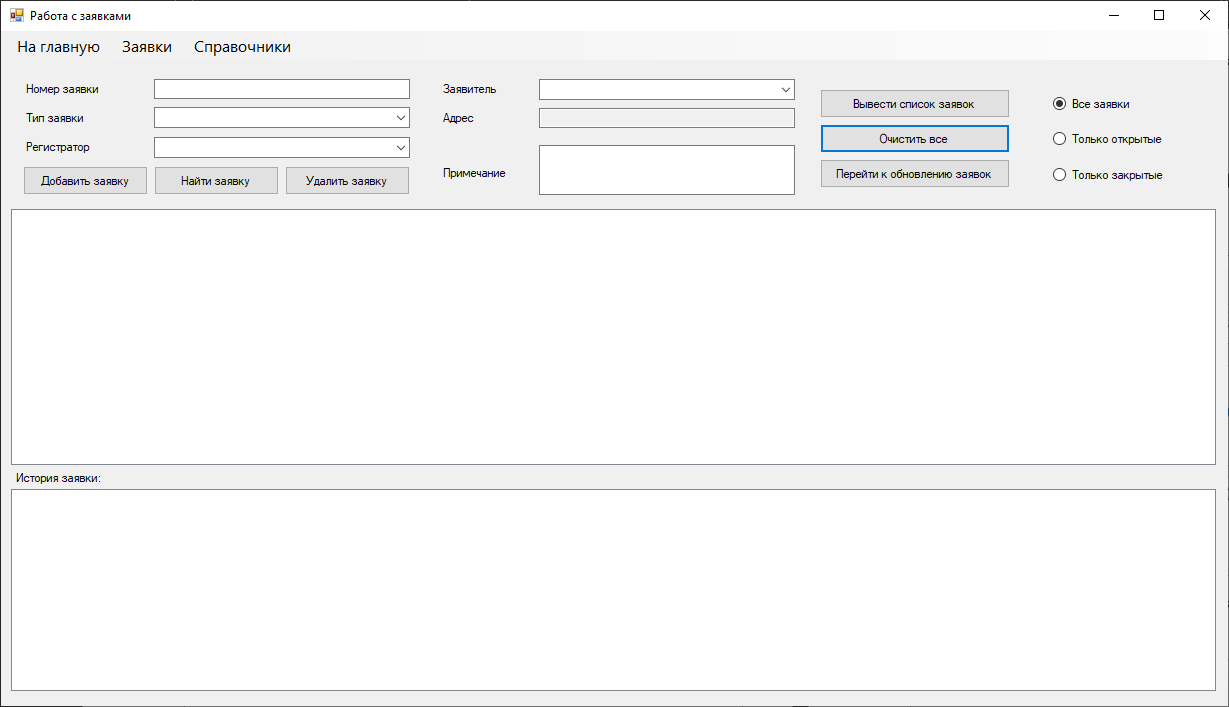


Рисунок 5.2.7 – Нажата кнопка «Очистить все»

Проверим работу кнопки «Вывести список заявок» (рис. 5.2.8).

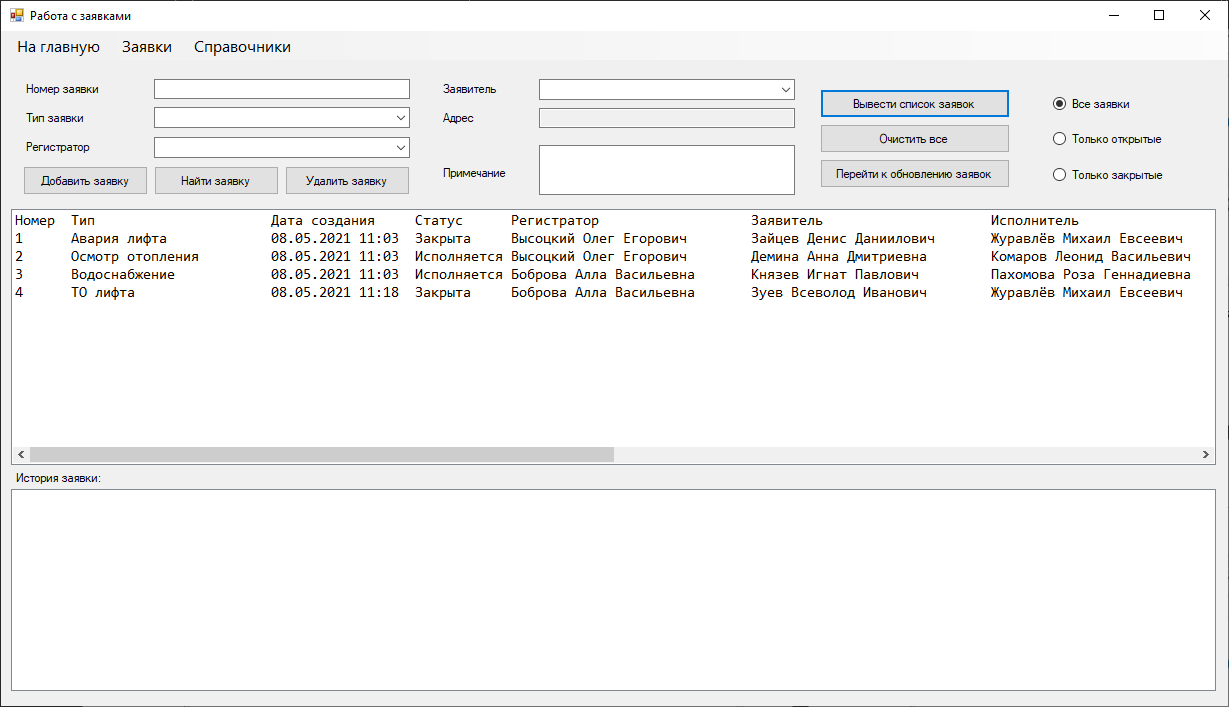


Рисунок 5.2.8 – Нажата кнопка «Вывести список заявок»

Проверим отбор заявок по открытым и закрытым (рис. 5.2.9).

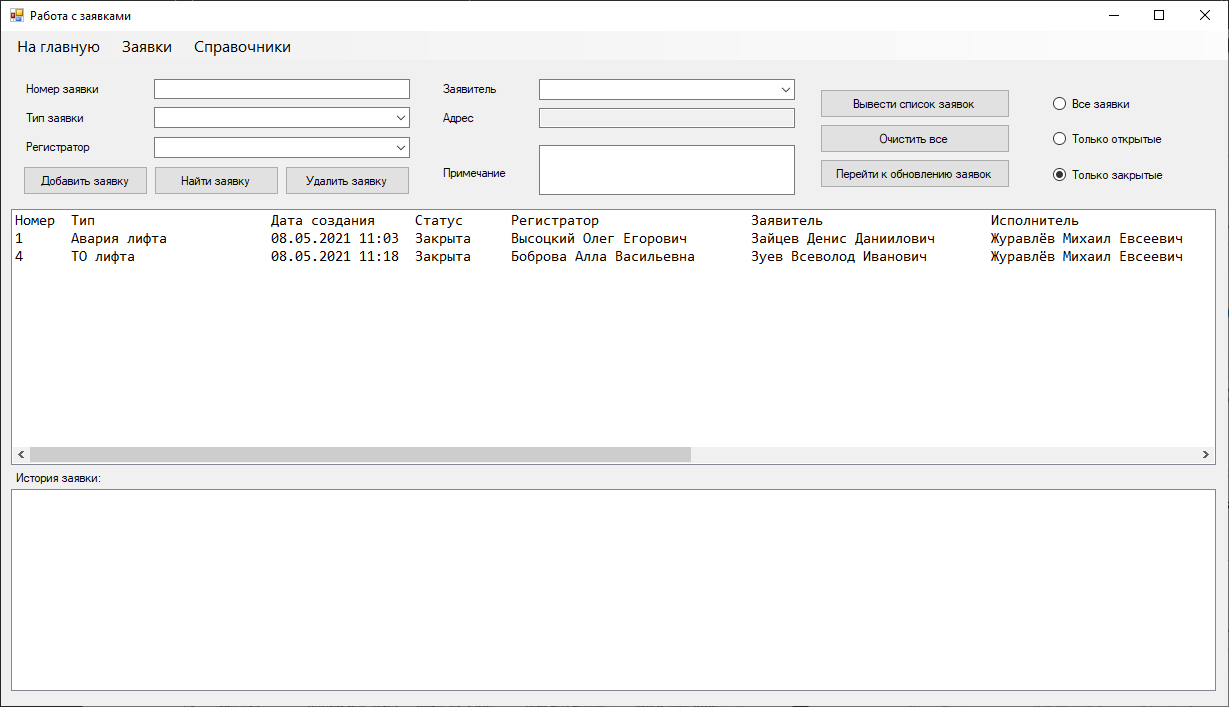


Рисунок 5.2.9 – Отобраны только закрытые заявки

Проверим поиск заявок по номеру (рис. 5.2.10).

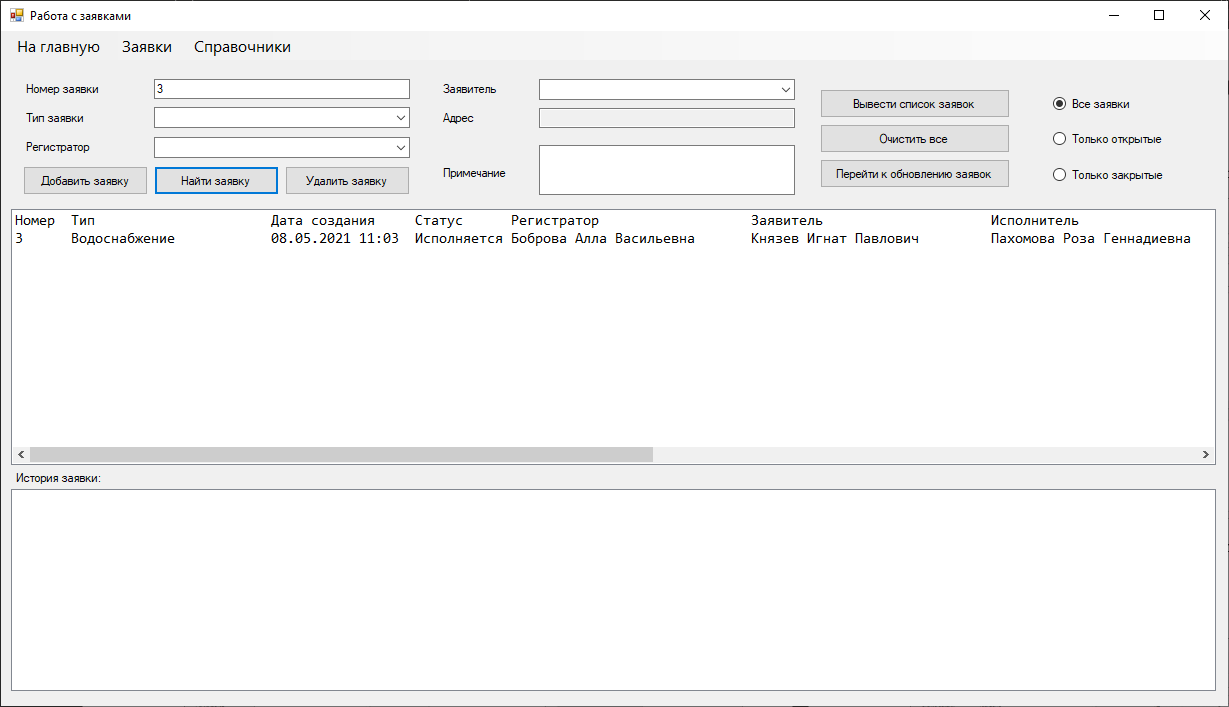


Рисунок 5.2.10 – Нажата кнопка «Найти заявку»

Работу справочников проверим на справочнике физ. лиц.

Имеем исходные данные (рис. 5.2.11):

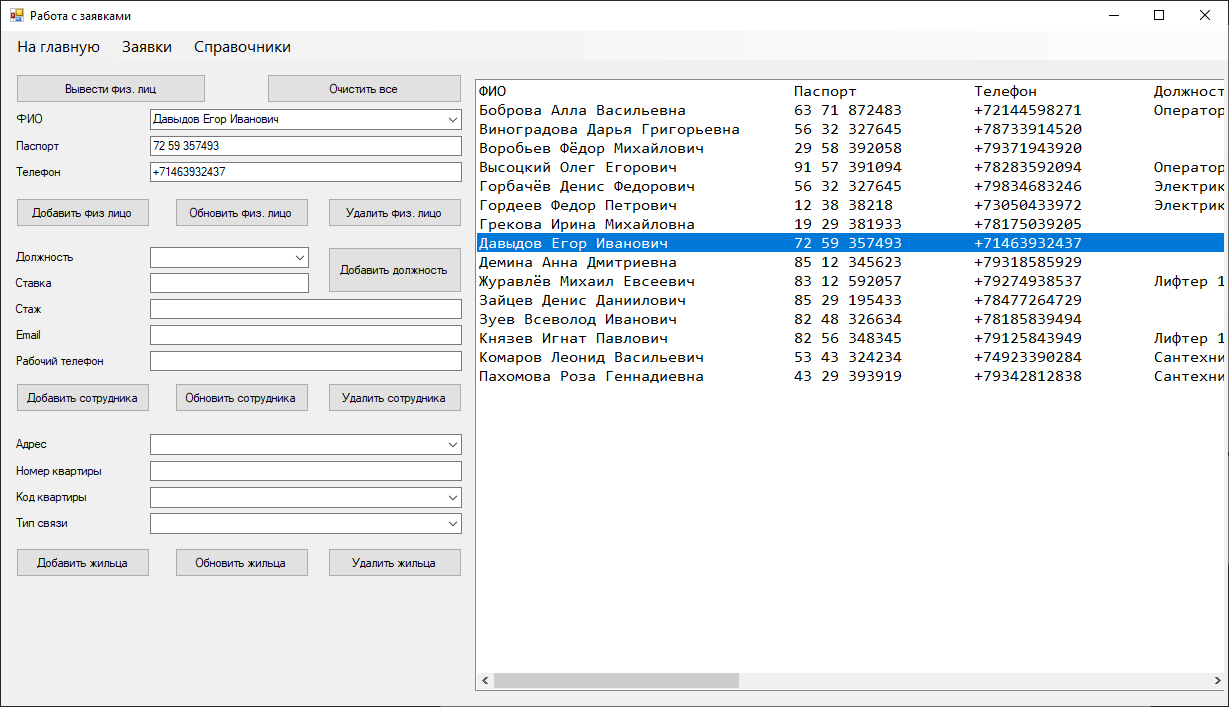


Рисунок 5.2.11 – Исходные данные физ. лица

Обновим ФИО и паспорт физ. лица, создадим новую должность, назначим ее физ. лицу (рис. 5.2.12).

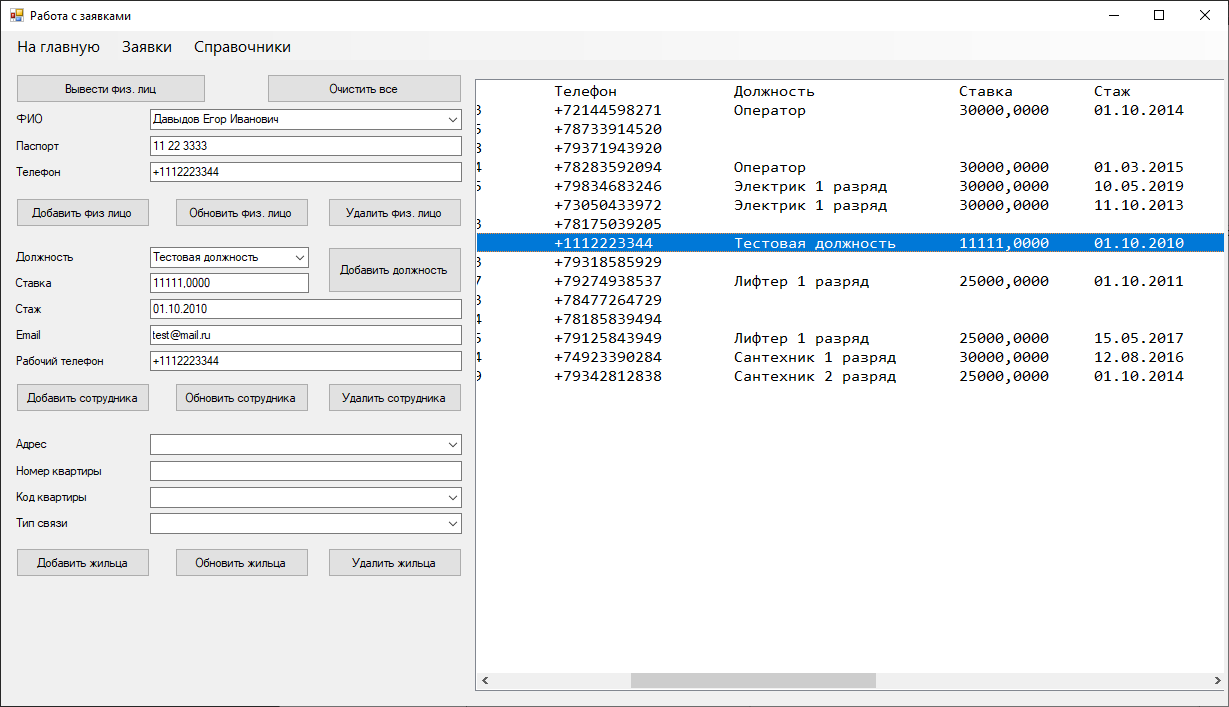


Рисунок 5.2.12 – Данные обновлены, должность назначена

Пропишем физ. лицо в квартиру (рис. 5.2.13).

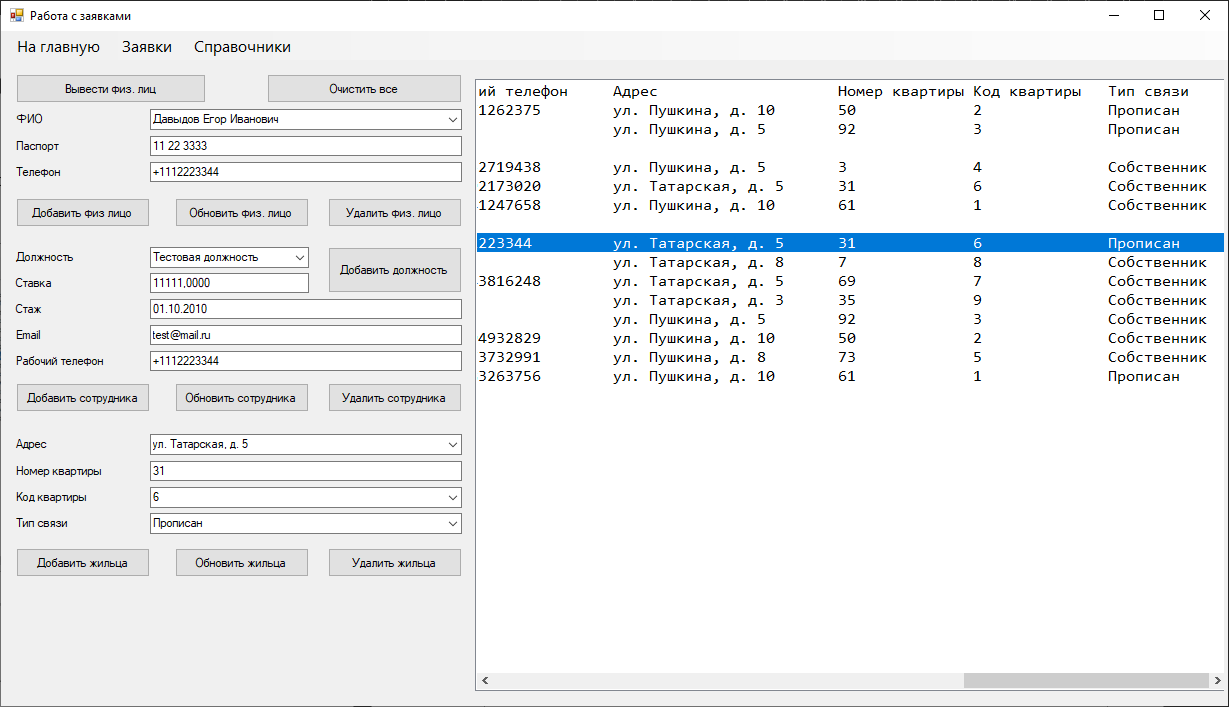


Рисунок 5.2.13 – Физ. лицо прописано в квартиру

Удаляем жильца, сотрудника и физ. лицо (рис. 5.2.14).

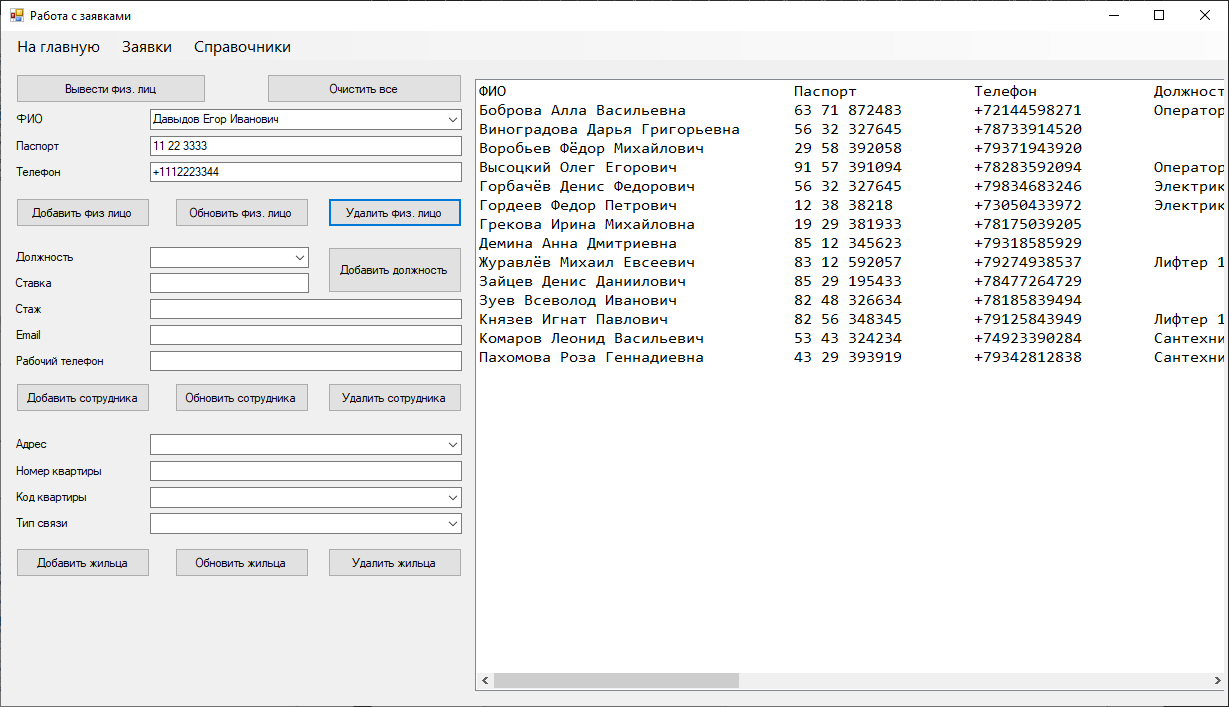


Рисунок 5.2.14 – Физ. лицо удалено

Поскольку полученные в результате функционального тестирования результаты совпадают с ожидаемыми, то можно сделать вывод о том, что программа работает корректно.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта было разработанное клиент-серверное приложение для управляющей компании, выполняющее функцию управления заявками. Была разработана база данных для взаимодействия с клиентским приложением, в разработке использовались как стандартные запросы, так и хранимые процедуры, триггеры, и представления. Эти задачи были реализованы на языке СУБД – SQL. В процессе выполнения курсовой работы были рассмотрены и изучены различные операторы языка SQL. Получен опыт разработки приложений на языке C# в среде Microsoft Visual Studio 2017.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Дейт К. Дж.: Введение в системы баз данных /Пер. с англ. [Текст] / Дейт К. Дж.; -М.: Издательский дом «Вильямс», 2001. – 1072с.: Библиогр.: с.4
2. Райордан Р. :Основы реляционных баз данных/Пер. с англ. [Текст] / Райордан Р. ; — М.: Издательско-торговый дом «Русская Редакция», 2001. — 384 с. ; Библиогр.: с.6
3. Хомоненко А.Д., Циганков В.М., Мальцев М.Г.:Базы данных:Учебник для высших учебных заведений. [Текст] /Под ред. проф. Хомоненко А.Д.; - СПб:КОРОНА принт, 2004. – 736 с.; Библиогр.: с. 2
4. Microsoft Corporation: Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCAD/MCSE, MCDBA/Пер. с англ. — 2-е изд., испр. . [Текст] / Microsoft Corporation; — М.: Издательско-торговый дом «Русская Редакция»,2003. - 512с.; Библиогр.: с. 2

# ПРИЛОЖЕНИЕ А: сценарий создания объектов БД.

USE master

GO

CREATE DATABASE [845\_Mukhanov\_curs]

GO

USE [845\_Mukhanov\_curs]

GO

CREATE TABLE Дом (

Адрес nvarchar(250) NOT NULL,

КоличествоКвартир int NULL,

КоличествоПодъездов int NULL,

CONSTRAINT PK\_Дом PRIMARY KEY (Адрес))

GO

CREATE TABLE Квартира (

КодКвартиры int NOT NULL,

Адрес nvarchar(250) NOT NULL,

Подъезд int NULL,

Этаж int NOT NULL,

НомерКвартиры int NOT NULL,

CONSTRAINT PK\_Квартира PRIMARY KEY (КодКвартиры),

CONSTRAINT FK\_Дом\_Квартира FOREIGN KEY (Адрес)

REFERENCES Дом(Адрес))

GO

CREATE SEQUENCE Sequence\_Квартира

START WITH 1

INCREMENT BY 1;

GO

CREATE TRIGGER TR\_Insert\_Квартира on Квартира

INSTEAD OF INSERT

AS

insert into Квартира (КодКвартиры, Адрес, Подъезд, Этаж, НомерКвартиры)

select NEXT VALUE FOR Sequence\_Квартира, i.Адрес, i.Подъезд, i.Этаж, i.НомерКвартиры from inserted i;

GO

CREATE TABLE ФизЛицо (

ФИО nvarchar(250) NOT NULL,

Паспорт nvarchar(12) NULL,

Телефон nvarchar(12) NULL,

CONSTRAINT PK\_ФизЛицо PRIMARY KEY (ФИО)

)

GO

CREATE TABLE Квартира\_ФизЛицо (

КодКвартиры int NOT NULL,

ФИО nvarchar(250) NOT NULL,

ТипСвязи nvarchar(250) NULL,

CONSTRAINT FK\_Квартира\_ФизЛицо\_Квартира FOREIGN KEY (КодКвартиры)

REFERENCES Квартира(КодКвартиры),

CONSTRAINT FK\_Квартира\_ФизЛицо\_ФизЛицо FOREIGN KEY (ФИО)

REFERENCES ФизЛицо(ФИО)

)

GO

CREATE TABLE Должности (

Должность nvarchar(250) NOT NULL,

Ставка money NULL,

CONSTRAINT PK\_Должности PRIMARY KEY (Должность))

GO

CREATE TABLE Сотрудник (

ФИО nvarchar(250) NOT NULL,

Должность nvarchar(250) NOT NULL,

Стаж date NOT NULL,

email nvarchar(100) NULL,

РабочийТелефон nvarchar(12) NULL,

CONSTRAINT PK\_Сотрудник PRIMARY KEY (ФИО),

CONSTRAINT FK\_Сотрудник\_ФизЛицо FOREIGN KEY (ФИО)

REFERENCES ФизЛицо(ФИО),

CONSTRAINT FK\_Сотрудник\_Должности FOREIGN KEY (Должность)

REFERENCES Должности(Должность)

)

GO

create view ФизЛицоСотрудник\_КвартираФизЛицо\_Квартира

as

select ФизЛицо.ФИО, Паспорт, Телефон, Сотрудник.Должность, Ставка, Стаж, email, РабочийТелефон, Адрес, НомерКвартиры, Квартира.КодКвартиры, ТипСвязи

from ФизЛицо

left join Сотрудник on ФизЛицо.ФИО = Сотрудник.ФИО

left join Должности on Сотрудник.Должность = Должности.Должность

left join Квартира\_ФизЛицо on ФизЛицо.ФИО = Квартира\_ФизЛицо.ФИО

left join Квартира on Квартира\_ФизЛицо.КодКвартиры = Квартира.КодКвартиры

go

CREATE TABLE ТипыЗаявок (

ТипЗаявки nvarchar(150) NOT NULL,

Приоритет int DEFAULT 1,

CONSTRAINT PK\_ТипЗаявки PRIMARY KEY (ТипЗаявки)

)

GO

CREATE TABLE ТипыЗаявок\_Должность (

ТипЗаявки nvarchar(150) NOT NULL,

Должность nvarchar(250) NOT NULL,

CONSTRAINT PK\_ТипыЗаявок\_Должность PRIMARY KEY (ТипЗаявки, Должность),

CONSTRAINT FK\_ТипыЗаявок\_Должность\_ТипЗаявки FOREIGN KEY (ТипЗаявки)

REFERENCES ТипыЗаявок(ТипЗаявки),

CONSTRAINT FK\_ТипыЗаявок\_Должность\_Должность FOREIGN KEY (Должность)

REFERENCES Должности(Должность)

)

GO

create view ТипыЗаявок\_Должность\_Сотрудники

as

select ТипыЗаявок\_Должность.ТипЗаявки, Сотрудник.ФИО

from ТипыЗаявок\_Должность

inner join Сотрудник on Сотрудник.Должность = ТипыЗаявок\_Должность.Должность

go

CREATE TABLE СтатусыЗаявок (

СтатусЗаявки nvarchar(50) NOT NULL,

Порядок int NOT NULL,

CONSTRAINT PK\_СтатусыЗаявок PRIMARY KEY (СтатусЗаявки)

)

GO

CREATE TABLE Заявки (

НомерЗаявки int NOT NULL,

ТипЗаявки nvarchar(150) NOT NULL,

ДатаСоздания datetime NOT NULL,

СтатусЗаявки nvarchar(50) NOT NULL,

Регистратор nvarchar(250) NOT NULL,

Заявитель nvarchar(250) NULL,

Исполнитель nvarchar(250) NULL,

ДатаЗакрытия datetime NULL,

Примечание nvarchar(250) NULL,

CONSTRAINT PK\_Заявки PRIMARY KEY (НомерЗаявки),

CONSTRAINT FK\_Заявки\_ТипЗаявки FOREIGN KEY (ТипЗаявки)

REFERENCES ТипыЗаявок(ТипЗаявки),

CONSTRAINT FK\_Заявки\_СтатусыЗаявок FOREIGN KEY (СтатусЗаявки)

REFERENCES СтатусыЗаявок(СтатусЗаявки),

CONSTRAINT FK\_Заявки\_Сотрудник\_Регистратор FOREIGN KEY (Регистратор)

REFERENCES Сотрудник(ФИО),

CONSTRAINT FK\_Заявки\_Заявитель FOREIGN KEY (Заявитель)

REFERENCES ФизЛицо(ФИО),

CONSTRAINT FK\_Заявки\_Сотрудник\_Исполнитель FOREIGN KEY (Исполнитель)

REFERENCES Сотрудник(ФИО)

)

GO

create view Заявки\_Квартиры

as

select НомерЗаявки,

ТипЗаявки,

ДатаСоздания,

СтатусЗаявки,

Регистратор,

Заявитель,

Исполнитель,

ДатаЗакрытия,

Примечание,

Квартира.Адрес,

Квартира.НомерКвартиры

from Заявки

left join Квартира\_ФизЛицо on Заявки.Заявитель = Квартира\_ФизЛицо.ФИО

left join Квартира on Квартира\_ФизЛицо.КодКвартиры = Квартира.КодКвартиры

go

CREATE SEQUENCE Sequence\_Заявки

START WITH 1

INCREMENT BY 1;

GO

-------------------------------------------

CREATE TRIGGER TR\_Insert\_Заявки on Заявки

INSTEAD OF INSERT

AS

DECLARE @Статус nvarchar(50)

DECLARE @CurrentDate datetime

DECLARE @IssueNumber int

DECLARE CUR\_Статус CURSOR SCROLL

FOR SELECT СтатусЗаявки

FROM СтатусыЗаявок

where Порядок = 1

OPEN CUR\_Статус

FETCH FIRST FROM CUR\_Статус

INTO @Статус

set @CurrentDate = SYSDATETIME()

set @IssueNumber = NEXT VALUE FOR Sequence\_Заявки

insert into Заявки (НомерЗаявки, ТипЗаявки, ДатаСоздания, СтатусЗаявки, Регистратор, Заявитель, Исполнитель, Примечание)

select @IssueNumber, i.ТипЗаявки, @CurrentDate, @Статус, i.Регистратор, i.Заявитель, i.Исполнитель, i.Примечание from inserted i;

insert into ИсторияЗаявки (НомерЗаявки, СтатусЗаявки, Исполнитель, Дата, Примечание)

select @IssueNumber, @Статус, i.Исполнитель, @CurrentDate, i.Примечание from inserted i;

CLOSE CUR\_Статус

DEALLOCATE CUR\_Статус

GO

-----------------------------------------------------------------

CREATE TRIGGER TR\_IOF\_Update\_Заявки on Заявки

INSTEAD OF UPDATE

AS

update Заявки

set

СтатусЗаявки= i.СтатусЗаявки,

Исполнитель = i.Исполнитель,

Примечание = i.Примечание,

ДатаЗакрытия = case when i.СтатусЗаявки = 'Закрыта' then SYSDATETIME() else null end

FROM inserted i

where Заявки.НомерЗаявки = i.НомерЗаявки

GO

CREATE TRIGGER TR\_Update\_Заявки on Заявки

AFTER UPDATE

AS

insert into ИсторияЗаявки (НомерЗаявки, СтатусЗаявки, Исполнитель, Дата, Примечание)

select i.НомерЗаявки, i.СтатусЗаявки, i.Исполнитель, SYSDATETIME(), i.Примечание from inserted i;

GO

-----------------------------------------------------------------

CREATE TABLE ИсторияЗаявки (

ID int NOT NULL,

НомерЗаявки int NOT NULL,

СтатусЗаявки nvarchar(50) NOT NULL,

Исполнитель nvarchar(250) NULL,

Дата datetime NULL,

Примечание nvarchar(250) NULL,

CONSTRAINT PK\_ИсторияЗаявки PRIMARY KEY (ID),

CONSTRAINT FK\_ИсторияЗаявки\_Заявки FOREIGN KEY (НомерЗаявки)

REFERENCES Заявки(НомерЗаявки),

CONSTRAINT FK\_ИсторияЗаявки\_СтатусыЗаявок FOREIGN KEY (СтатусЗаявки)

REFERENCES СтатусыЗаявок(СтатусЗаявки),

CONSTRAINT FK\_ИсторияЗаявки\_Сотрудник FOREIGN KEY (Исполнитель)

REFERENCES Сотрудник(ФИО)

)

GO

CREATE SEQUENCE Sequence\_ИсторияЗаявки

START WITH 1

INCREMENT BY 1;

GO

CREATE TRIGGER TR\_Insert\_ИсторияЗаявки on ИсторияЗаявки

INSTEAD OF INSERT

AS

insert into ИсторияЗаявки (ID, НомерЗаявки, СтатусЗаявки, Исполнитель, Дата, Примечание)

select NEXT VALUE FOR Sequence\_ИсторияЗаявки, i.НомерЗаявки, i.СтатусЗаявки, i.Исполнитель, i.Дата, i.Примечание from inserted i;

GO

-----------------Создание процедур------------------------------------------------

create proc InsertHome @Адрес nvarchar(250), @КоличествоКвартир int, @КоличествоПодъездов int

as

begin

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

values (@Адрес, @КоличествоКвартир, @КоличествоПодъездов)

end;

go

create proc deleteHome @Адрес nvarchar(250)

as

begin

delete from Дом where Адрес = @Адрес

end;

go

create proc updateHome @Адрес nvarchar(250), @КоличествоКвартир int, @КоличествоПодъездов int

as

begin

UPDATE Дом

SET КоличествоКвартир = @КоличествоКвартир,

КоличествоПодъездов = @КоличествоПодъездов

WHERE Адрес = @Адрес

end;

go

create proc insertIssue @ТипЗаявки nvarchar(150),

@Регистратор nvarchar(250),

@Заявитель nvarchar(250) = null,

@Примечание nvarchar(250) = null

as

begin

insert into Заявки (ТипЗаявки, Регистратор, Заявитель, Примечание)

values (@ТипЗаявки, @Регистратор, @Заявитель, @Примечание)

end;

go

create proc updateIssue @НомерЗаявки int,

@СтатусЗаявки nvarchar(50),

@Исполнитель nvarchar(250) = null,

@Примечание nvarchar(250) = null,

@ДатаЗакрытия datetime = null

as

begin

UPDATE Заявки

SET СтатусЗаявки = @СтатусЗаявки,

Исполнитель = @Исполнитель,

Примечание = @Примечание,

ДатаЗакрытия = @ДатаЗакрытия

WHERE НомерЗаявки = @НомерЗаявки

end;

go

create proc deleteIssue @НомерЗаявки int

as

begin

delete from ИсторияЗаявки where НомерЗаявки = @НомерЗаявки

delete from Заявки where НомерЗаявки = @НомерЗаявки

end;

go

create proc insertApart @Адрес nvarchar(250),

@Подъезд int,

@Этаж int,

@НомерКвартиры int

as

begin

insert into Квартира(Адрес, Подъезд, Этаж, НомерКвартиры)

values (@Адрес, @Подъезд, @Этаж, @НомерКвартиры)

end;

go

create proc updateApart @КодКвартиры int,

@Адрес nvarchar(250),

@Подъезд int,

@Этаж int,

@НомерКвартиры int

as

begin

update Квартира

set НомерКвартиры = @НомерКвартиры,

Адрес = @Адрес,

Подъезд = @Подъезд,

Этаж = @Этаж

where КодКвартиры = @КодКвартиры

end;

go

create proc deleteApart @КодКвартиры int

as

begin

delete from Квартира where КодКвартиры = @КодКвартиры

end;

go

create proc InsertPerson @ФИО nvarchar(250), @Паспорт nvarchar(12), @Телефон nvarchar(12)

as

begin

INSERT INTO ФизЛицо(ФИО, Паспорт, Телефон)

values (@ФИО, @Паспорт, @Телефон)

end;

go

create proc updatePerson @ФИО nvarchar(250), @Паспорт nvarchar(12), @Телефон nvarchar(12)

as

begin

update ФизЛицо

set Паспорт = @Паспорт, Телефон = @Телефон

where ФИО = @ФИО

end;

go

create proc deletePerson @ФИО nvarchar(250)

as

begin

delete from ФизЛицо where ФИО = @ФИО

end;

go

create proc InsertRole @Должность nvarchar(250), @Ставка money

as

begin

INSERT INTO Должности(Должность, Ставка)

values (@Должность, @Ставка)

end;

go

create proc InsertEmployee @ФИО nvarchar(250), @Должность nvarchar(250), @Стаж date, @email nvarchar(100), @РабочийТелефон nvarchar(12)

as

begin

INSERT INTO Сотрудник (ФИО, Должность, Стаж, email, РабочийТелефон)

values (@ФИО, @Должность, @Стаж, @email, @РабочийТелефон)

end;

go

create proc updateEmployee @ФИО nvarchar(250), @Должность nvarchar(250), @Стаж date, @email nvarchar(100), @РабочийТелефон nvarchar(12)

as

begin

update Сотрудник

set Должность = @Должность, Стаж = @Стаж, email = @email, РабочийТелефон = @РабочийТелефон

where ФИО = @ФИО

end;

go

create proc deleteEmployee @ФИО nvarchar(250)

as

begin

delete from Сотрудник where ФИО = @ФИО

end;

go

create proc InsertTenant @КодКвартиры int, @ФИО nvarchar(250), @ТипСвязи nvarchar(250)

as

begin

INSERT INTO Квартира\_ФизЛицо (КодКвартиры, ФИО, ТипСвязи)

values (@КодКвартиры, @ФИО, @ТипСвязи)

end;

go

create proc updateTenant @КодКвартиры int, @ФИО nvarchar(250), @ТипСвязи nvarchar(250)

as

begin

update Квартира\_ФизЛицо

set КодКвартиры = @КодКвартиры, ТипСвязи = @ТипСвязи

where ФИО = @ФИО

end;

go

create proc deleteTenant @ФИО nvarchar(250)

as

begin

delete from Квартира\_ФизЛицо where ФИО = @ФИО

end;

go

# ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД.

INSERT INTO ТипыЗаявок (ТипЗаявки, Приоритет)

VALUES

('Осмотр сантехники', 2),

('Замена сантехники', 1),

('Осмотр электрики', 2),

('Замена электрики', 1),

('Авария лифта', 1),

('ТО лифта', 3),

('Ремонт лифта', 2),

('Уборка помещений', 2),

('Уборка территории', 2),

('Водоснабжение', 1),

('Ремонт отопления', 1),

('Осмотр отопления', 2)

GO

INSERT INTO СтатусыЗаявок (СтатусЗаявки, Порядок)

VALUES

('Создана', 1),

('Назначена', 2),

('Исполняется', 3),

('Исполнена', 4),

('Закрыта', 5)

GO

INSERT INTO Должности (Должность, Ставка)

VALUES

('Директор', 100000),

('Главный инженер', 80000),

('Главный бухгалтер', 80000),

('Бухгалтер', 40000),

('Дворник 1 разряд', 15000),

('Дворник 2 разряд', 20000),

('Сантехник 1 разряд', 30000),

('Сантехник 2 разряд', 25000),

('Сантехник 3 разряд', 20000),

('Лифтер 1 разряд', 25000),

('Лифтер 2 разряд', 20000),

('Электрик 1 разряд', 30000),

('Электрик 2 разряд', 25000),

('Электрик 3 разряд', 20000),

('Оператор', 30000)

GO

INSERT INTO ТипыЗаявок\_Должность (ТипЗаявки, Должность)

VALUES

('Осмотр сантехники','Сантехник 1 разряд'),

('Осмотр сантехники','Сантехник 2 разряд'),

('Осмотр сантехники','Сантехник 3 разряд'),

('Замена сантехники','Сантехник 1 разряд'),

('Водоснабжение','Сантехник 1 разряд'),

('Водоснабжение','Сантехник 2 разряд'),

('Ремонт отопления','Сантехник 1 разряд'),

('Ремонт отопления','Сантехник 2 разряд'),

('Осмотр отопления','Сантехник 1 разряд'),

('Осмотр отопления','Сантехник 2 разряд'),

('Осмотр отопления','Сантехник 3 разряд'),

('Авария лифта','Лифтер 1 разряд'),

('ТО лифта','Лифтер 1 разряд'),

('ТО лифта','Лифтер 2 разряд'),

('Ремонт лифта','Лифтер 1 разряд'),

('Замена электрики','Электрик 1 разряд'),

('Осмотр электрики','Электрик 1 разряд'),

('Осмотр электрики','Электрик 2 разряд')

GO

INSERT INTO Дом (Адрес, КоличествоКвартир, КоличествоПодъездов)

VALUES

('ул. Пушкина, д. 10', 100, 3),

('ул. Пушкина, д. 22', 250, 7),

('ул. Пушкина, д. 5', 250, 7),

('ул. Пушкина, д. 8', 250, 7),

('ул. Татарская, д. 3', 50, 1),

('ул. Татарская, д. 5', 150, 4),

('ул. Татарская, д. 8', 200, 5),

('ул. Гагарина, д. 1', 100, 2),

('ул. Гагарина, д. 2', 100, 2),

('ул. Гагарина, д. 3', 100, 2)

GO

INSERT INTO Квартира (Адрес, Подъезд, Этаж, НомерКвартиры)

VALUES

('ул. Пушкина, д. 10', 1, 5, 61),

('ул. Пушкина, д. 10', 1, 4, 50),

('ул. Пушкина, д. 5', 4, 9, 92),

('ул. Пушкина, д. 5', 1, 1, 3),

('ул. Пушкина, д. 8', 6, 7, 73),

('ул. Татарская, д. 5', 4, 2, 31),

('ул. Татарская, д. 5', 1, 6, 69),

('ул. Татарская, д. 8', 2, 1, 7),

('ул. Татарская, д. 3', 1, 3, 35),

('ул. Гагарина, д. 3', 2, 7, 70)

GO

INSERT INTO ФизЛицо (ФИО, Паспорт, Телефон)

VALUES

('Комаров Леонид Васильевич', '53 43 324234','+74923390284'),

('Горбачёв Денис Федорович', '56 32 327645','+79834683246'),

('Журавлёв Михаил Евсеевич', '83 12 592057','+79274938537'),

('Высоцкий Олег Егорович', '91 57 391094','+78283592094'),

('Гордеев Федор Петрович', '12 38 38218','+73050433972'),

('Князев Игнат Павлович', '82 56 348345','+79125843949'),

('Зуев Всеволод Иванович', '82 48 326634','+78185839494'),

('Пахомова Роза Геннадиевна', '43 29 393919','+79342812838'),

('Боброва Алла Васильевна', '63 71 872483','+72144598271'),

('Виноградова Дарья Григорьевна', '56 32 327645','+78733914520'),

('Воробьев Фёдор Михайлович', '29 58 392058','+79371943920'),

('Грекова Ирина Михайловна', '19 29 381933','+78175039205'),

('Давыдов Егор Иванович', '72 59 357493','+71463932437'),

('Демина Анна Дмитриевна', '85 12 345623','+79318585929'),

('Зайцев Денис Даниилович', '85 29 195433','+78477264729')

GO

INSERT INTO Сотрудник (ФИО, Должность, Стаж, email, РабочийТелефон)

VALUES

('Комаров Леонид Васильевич', 'Сантехник 1 разряд', '2016-08-12', 'klv@mail.ru','+79173732991'),

('Горбачёв Денис Федорович', 'Электрик 1 разряд', '2019-05-10', 'gdf@mail.ru','+79992173020'),

('Гордеев Федор Петрович', 'Электрик 1 разряд', '2013-10-11', 'gfp@mail.ru','+77641247658'),

('Журавлёв Михаил Евсеевич', 'Лифтер 1 разряд', '2011-10-01', 'gme@mail.ru','+79543816248'),

('Князев Игнат Павлович', 'Лифтер 1 разряд', '2017-05-15', 'kip@mail.ru','+72394932829'),

('Пахомова Роза Геннадиевна', 'Сантехник 2 разряд', '2014-10-01', 'prg@mail.ru','+73683263756'),

('Боброва Алла Васильевна', 'Оператор', '2014-10-01', 'bav@mail.ru','+75371262375'),

('Высоцкий Олег Егорович', 'Оператор', '2015-03-01', 'voe@mail.ru','+74932719438')

GO

INSERT INTO Квартира\_ФизЛицо (КодКвартиры, ФИО, ТипСвязи)

VALUES

(1, 'Гордеев Федор Петрович','Собственник'),

(1, 'Пахомова Роза Геннадиевна','Прописан'),

(2, 'Князев Игнат Павлович','Собственник'),

(2, 'Боброва Алла Васильевна','Прописан'),

(3, 'Зуев Всеволод Иванович','Собственник'),

(3, 'Виноградова Дарья Григорьевна','Прописан'),

(4, 'Высоцкий Олег Егорович','Собственник'),

(5, 'Комаров Леонид Васильевич','Собственник'),

(6, 'Горбачёв Денис Федорович','Собственник'),

(7, 'Журавлёв Михаил Евсеевич','Собственник'),

(8, 'Демина Анна Дмитриевна','Собственник'),

(9, 'Зайцев Денис Даниилович','Собственник')

GO

exec insertIssue 'Авария лифта', 'Высоцкий Олег Егорович','Зайцев Денис Даниилович', 'Застряли в лифте на Татарской д.3'

exec insertIssue 'Осмотр отопления', 'Высоцкий Олег Егорович','Демина Анна Дмитриевна', 'Капает вода из батареи'

exec insertIssue 'Водоснабжение', 'Боброва Алла Васильевна','Князев Игнат Павлович', 'Нет воды'

go

exec updateIssue 1, 'Назначена', 'Журавлёв Михаил Евсеевич', 'Назначен мастер Журавлёв'

exec updateIssue 1, 'Исполняется', 'Журавлёв Михаил Евсеевич', 'Мастер выехал к месту аварии'

exec updateIssue 1, 'Исполнена', 'Журавлёв Михаил Евсеевич', 'Авария устранена'

exec updateIssue 1, 'Закрыта', 'Журавлёв Михаил Евсеевич', 'Заявка закрыта'

exec updateIssue 2, 'Назначена', 'Комаров Леонид Васильевич', 'Назначен сантехник Комаров'

exec updateIssue 2, 'Исполняется', 'Комаров Леонид Васильевич', 'Сантехник определил необходимость замены радиатора'

exec updateIssue 3, 'Назначена', 'Комаров Леонид Васильевич', 'Назначен сантехник Комаров'

exec updateIssue 3, 'Назначена', 'Пахомова Роза Геннадиевна', 'Комаров недоступен. Назначен сантехник Пахомова'

exec updateIssue 3, 'Исполняется', 'Пахомова Роза Геннадиевна', 'Сантехник приступил к исполнению'

go

# ПРИЛОЖЕНИЕ В: исходный текст клиентской программы.

Текст mainForm.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.Data.SqlClient;

using System.Windows.Forms;

using System.Configuration;

namespace curs

{

public partial class mainForm : Form

{

public mainForm()

{

InitializeComponent();

ПанельВкладок.Appearance = TabAppearance.FlatButtons;

ПанельВкладок.ItemSize = new Size(0, 1);

ПанельВкладок.SizeMode = TabSizeMode.Fixed;

}

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

{

ПанельВкладок.SelectTab(1);

ВывестиДома\_Click(sender, e);

}

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

{

ПанельВкладок.SelectTab(0);

}

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

{

ПанельВкладок.SelectTab(5);

ЗаполнитьComboBoxВФизЛицах();

ОчиститьФизЛиц\_Click(sender, e);

ВывестиФизЛиц\_Click(sender, e);

}

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

{

ПанельВкладок.SelectTab(4);

ЗаполнитьComboBoxВКвартирах();

АдресКвартиры.Text = "";

ВывестиКвартиры\_Click(sender, e);

}

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

{

ПанельВкладок.SelectTab(2);

ЗаполнитьВыпадающиеМеню();

ОчиститьЗаявки\_Click(sender, e);

ВывестиЗаявки\_Click(sender, e);

}

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

{

ПанельВкладок.SelectTab(3);

ЗаполнитьВыпадающиеМеню();

ОсчиститьЗаявкиСобновления\_Click(sender, e);

ВывестиЗаявки\_Click(sender, e);

}

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

{

HousesOperations.выводДомов();

string listFormat = "{0,-25}{1,-25}{2,-25}";

СписокДомов.Items.Clear();

СписокДомов.Items.Add(String.Format(listFormat, "Адрес", "Количество Квартир", "Количество Подъездов"));

for (int i = 0; i < Houses.Адрес.Count; i++)

{

СписокДомов.Items.Add(String.Format(listFormat,

Houses.Адрес[i],

Houses.КоличествоКвартир[i],

Houses.КоличествоПодъездов[i]));

}

}

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

{

СписокДомов.Items.Clear();

АдресДома.Text = "";

КоличествоКвартир.Text = "";

КоличествоПодъездов.Text = "";

ИзмененоДомов.Text = "";

}

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

{

string ПараметрАдресДома = АдресДома.Text.ToString();

string ПараметрКоличествоКвартир = КоличествоКвартир.Text;

string ПараметрКоличествоПодъездов = КоличествоПодъездов.Text;

if (HousesOperations.добавлениеДома(ПараметрАдресДома, ПараметрКоличествоКвартир, ПараметрКоличествоПодъездов) > 0)

ИзмененоДомов.Text = "Добавлено домов: 1";

else

ИзмененоДомов.Text = "Дом не добавлен";

ВывестиДома\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "deleteHome";

cmd.Parameters.Add("@Адрес", SqlDbType.NVarChar, 250).Value = АдресДома.Text.ToString();

int result = cmd.ExecuteNonQuery();

if (result > 0)

ИзмененоДомов.Text = "Удалено домов: " + result.ToString();

else

ИзмененоДомов.Text = "Дом не удален";

con.Close();

ВывестиДома\_Click(sender, e);

}

private void СписокДомов\_MouseClick(object sender, MouseEventArgs e)

{ if (СписокДомов.SelectedItem != null)

{

АдресДома.Text = СписокДомов.SelectedItem.ToString().Substring(0, 25).Trim();

КоличествоКвартир.Text = СписокДомов.SelectedItem.ToString().Substring(25, 25).Trim();

КоличествоПодъездов.Text = СписокДомов.SelectedItem.ToString().Substring(50, 25).Trim();

}

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "updateHome";

cmd.Parameters.Add("@Адрес", SqlDbType.NVarChar, 250).Value = АдресДома.Text.ToString();

cmd.Parameters.Add("@КоличествоКвартир", SqlDbType.Int).Value = КоличествоКвартир.Text;

cmd.Parameters.Add("@КоличествоПодъездов", SqlDbType.Int).Value = КоличествоПодъездов.Text;

int result = cmd.ExecuteNonQuery();

if (result > 0)

ИзмененоДомов.Text = "Обновлено домов: " + result.ToString();

else

ИзмененоДомов.Text = "Дом не обновлен";

con.Close();

ВывестиДома\_Click(sender, e);

}

//----------------------------Работа с заявками --------------------------------------------

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

{

bool всеЗаявки = ВсеЗаявки.Checked ? true : false;

bool толькоОткрытые = ТолькоОткрытые.Checked ? true : false;

bool толькоЗакрытые = ТолькоЗакрытые.Checked ? true : false;

IssuesOperations.вывестиЗаявки(всеЗаявки, толькоОткрытые, толькоЗакрытые);

вывестиЗаявки();

}

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

{

bool всеЗаявки = ВсеЗаявки.Checked ? true : false;

bool толькоОткрытые = ТолькоОткрытые.Checked ? true : false;

bool толькоЗакрытые = ТолькоЗакрытые.Checked ? true : false;

if (НомерЗаявки.Text != "") {

IssuesOperations.вывестиЗаявки(всеЗаявки, толькоОткрытые, толькоЗакрытые, true, Int16.Parse(НомерЗаявки.Text));

}

вывестиЗаявки();

}

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

{

IssuesOperations.добавитьЗаявку(ТипЗаявки.Text.ToString(), Регистратор.Text.ToString(),

Заявитель.Text.ToString(), Примечание.Text.ToString());

ВывестиЗаявки\_Click(sender, e);

}

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

{

IssuesOperations.удалитьЗаявку(НомерЗаявки.Text.ToString());

ВывестиЗаявки\_Click(sender, e);

}

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

{

IssuesOperations.обновитьЗаявку(НомерЗаявкиОбновление.Text, СтатусЗаявки.Text.ToString(),

Исполнитель.Text.ToString(), ПримечаниеОбновление.Text.ToString());

ВывестиЗаявки\_Click(sender, e);

ВывестиИсториюЗаявкиОбновление();

}

private void вывестиЗаявки()

{

string listFormat = "{0,-7}{1,-25}{2,-18}{3,-12}{4,-30}{5,-30}{6,-30}{7,-25}{8,-35}{9,-20}{10,-180}";

СписокЗаявок.Items.Clear();

СписокЗаявокОбновление.Items.Clear();

СписокЗаявок.Items.Add(String.Format(listFormat, "Номер", "Тип", "Дата создания", "Статус", "Регистратор",

"Заявитель", "Исполнитель", "Дата закрытия", "Адрес", "Номер квартиры", "Примечание"));

СписокЗаявокОбновление.Items.Add(String.Format(listFormat, "Номер", "Тип", "Дата создания", "Статус", "Регистратор",

"Заявитель", "Исполнитель", "Дата закрытия", "Адрес", "Номер квартиры", "Примечание"));

for (int i = 0; i < Issues.НомерЗаявки.Count; i++)

{

СписокЗаявок.Items.Add(

String.Format(listFormat,Issues.НомерЗаявки[i],Issues.ТипЗаявки[i],Issues.ДатаСоздания[i],

Issues.СтатусЗаявки[i],Issues.Регистратор[i],Issues.Заявитель[i],Issues.Исполнитель[i],

Issues.ДатаЗакрытия[i], Issues.Адрес[i],Issues.НомерКвартиры[i],Issues.Примечание[i]));

СписокЗаявокОбновление.Items.Add(

String.Format(listFormat, Issues.НомерЗаявки[i], Issues.ТипЗаявки[i], Issues.ДатаСоздания[i],

Issues.СтатусЗаявки[i], Issues.Регистратор[i], Issues.Заявитель[i], Issues.Исполнитель[i],

Issues.ДатаЗакрытия[i], Issues.Адрес[i], Issues.НомерКвартиры[i], Issues.Примечание[i]));

}

}

private void ВывестиИсториюЗаявки()

{

bool ВСпискеЗаявокВыбранЭлемент = (СписокЗаявок.SelectedItem != null) ? true : false;

bool ВСпискеЗаявокВыбраноНеОглавление = (СписокЗаявок.SelectedItem != СписокЗаявок.Items[0]) ? true : false;

IssuesHistoryOperations.вывестиИсториюЗаявки(ВСпискеЗаявокВыбранЭлемент, ВСпискеЗаявокВыбраноНеОглавление, НомерЗаявки.Text);

string listFormat = "{0,-7}{1,-15}{2,-15}{3,-30}{4,-25}{5,-90}";

ИсторияЗаявки.Items.Clear();

ИсторияЗаявки.Items.Add(String.Format(listFormat, "ID", "Номер заявки", "Статус заявки", "Исполнитель", "Дата", "Примечание"));

for (int i = 0; i < IssuesHistory.ID.Count; i++)

{

ИсторияЗаявки.Items.Add(

String.Format(listFormat, IssuesHistory.ID[i], IssuesHistory.НомерЗаявки[i], IssuesHistory.СтатусЗаявки[i],

IssuesHistory.Исполнитель[i], IssuesHistory.Дата[i], IssuesHistory.Примечание[i]));

}

}

private void ЗаполнитьВыпадающиеМеню()

{

IssuesOperations.заполнитьВыпадающееМеню(ТипЗаявкиОбновление.Text.ToString());

Регистратор.DisplayMember = "ФИО";

Регистратор.DataSource = Issues.операторы.Tables[0].DefaultView;

ТипЗаявки.DisplayMember = "ТипЗаявки";

ТипЗаявки.DataSource = Issues.типыЗаявок.Tables[0].DefaultView;

Заявитель.DisplayMember = "ФИО";

Заявитель.DataSource = Issues.жильцы.Tables[0].DefaultView;

СтатусЗаявки.DisplayMember = "СтатусЗаявки";

СтатусЗаявки.DataSource = Issues.статусыЗаявок.Tables[0].DefaultView;

if (ТипЗаявкиОбновление.Text.ToString() != "") {

Исполнитель.DisplayMember = "ФИО";

Исполнитель.DataSource = Issues.исполнители.Tables[0].DefaultView;

}

}

private void СписокЗаявок\_MouseClick(object sender, MouseEventArgs e)

{

if (СписокЗаявок.SelectedItem != null)

{

НомерЗаявки.Text = СписокЗаявок.SelectedItem.ToString().Substring(0, 7).Trim();

ТипЗаявки.Text = СписокЗаявок.SelectedItem.ToString().Substring(7, 25).Trim();

Регистратор.Text = СписокЗаявок.SelectedItem.ToString().Substring(62, 30).Trim();

Заявитель.Text = СписокЗаявок.SelectedItem.ToString().Substring(92, 30).Trim();

ПолныйАдрес.Text = СписокЗаявок.SelectedItem.ToString().Substring(177, 35).Trim() +

", кв. " + СписокЗаявок.SelectedItem.ToString().Substring(212, 20).Trim();

if (ПолныйАдрес.Text.ToString().Trim() == ", кв.")

{

ПолныйАдрес.Text = "";

}

Примечание.Text = СписокЗаявок.SelectedItem.ToString().Substring(232).Trim();

ВывестиИсториюЗаявки();

}

}

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

{

СписокЗаявок.Items.Clear();

ИсторияЗаявки.Items.Clear();

НомерЗаявки.Text = "";

ТипЗаявки.Text = "";

Регистратор.Text = "";

Заявитель.Text = "";

ПолныйАдрес.Text = "";

Примечание.Text = "";

}

private void СписокЗаявокОбновление\_MouseClick(object sender, MouseEventArgs e)

{

if ((СписокЗаявокОбновление.SelectedItem != null) & (СписокЗаявокОбновление.SelectedItem != СписокЗаявокОбновление.Items[0])) {

НомерЗаявкиОбновление.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(0, 7).Trim();

ТипЗаявкиОбновление.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(7, 25).Trim();

Исполнитель.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(92, 30).Trim();

ЗаявительОбновление.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(92, 30).Trim();

АдресОбновление.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(177, 35).Trim() +

", кв. " + СписокЗаявокОбновление.SelectedItem.ToString().Substring(212, 20).Trim();

if (АдресОбновление.Text.ToString().Trim() == ", кв.")

{

АдресОбновление.Text = "";

}

ПримечаниеОбновление.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(232).Trim();

ЗаполнитьВыпадающиеМеню();

СтатусЗаявки.Text = СписокЗаявокОбновление.SelectedItem.ToString().Substring(50, 12).Trim();

ВывестиИсториюЗаявкиОбновление();

}

}

private void ВывестиИсториюЗаявкиОбновление()

{

bool ВСпискеЗаявокВыбранЭлемент = (СписокЗаявокОбновление.SelectedItem != null) ? true : false;

bool ВСпискеЗаявокВыбраноНеОглавление = (СписокЗаявокОбновление.SelectedItem != СписокЗаявокОбновление.Items[0]) ? true : false;

IssuesHistoryOperations.вывестиИсториюЗаявки(true, ВСпискеЗаявокВыбраноНеОглавление, НомерЗаявкиОбновление.Text);

string listFormat = "{0,-7}{1,-15}{2,-15}{3,-30}{4,-25}{5,-90}";

ИсторияЗаявкиОбновление.Items.Clear();

ИсторияЗаявкиОбновление.Items.Add(String.Format(listFormat, "ID", "Номер заявки", "Статус заявки", "Исполнитель", "Дата", "Примечание"));

for (int i = 0; i < IssuesHistory.ID.Count; i++)

{

ИсторияЗаявкиОбновление.Items.Add(

String.Format(listFormat, IssuesHistory.ID[i], IssuesHistory.НомерЗаявки[i], IssuesHistory.СтатусЗаявки[i],

IssuesHistory.Исполнитель[i], IssuesHistory.Дата[i], IssuesHistory.Примечание[i]));

}

}

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

{

СписокЗаявокОбновление.Items.Clear();

ИсторияЗаявкиОбновление.Items.Clear();

НомерЗаявкиОбновление.Text = "";

ТипЗаявкиОбновление.Text = "";

СтатусЗаявки.Text = "";

Исполнитель.Text = "";

ЗаявительОбновление.Text = "";

АдресОбновление.Text = "";

ПримечаниеОбновление.Text = "";

}

//---------------------Работа с квартирами-------------------------------------

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select \* from Квартира";

SqlDataReader reader = cmd.ExecuteReader();

string listFormat = "{0,-15}{1,-30}{2,-10}{3,-7}{4,-20}";

СписокКвартир.Items.Clear();

СписокКвартир.Items.Add(String.Format(listFormat, "Код квартиры", "Адрес", "Подъезд", "Этаж", "Номер квартиры"));

while (reader.Read())

{

СписокКвартир.Items.Add(String.Format(listFormat,

reader["КодКвартиры"].ToString().Trim(),

reader["Адрес"].ToString().Trim(),

reader["Подъезд"].ToString().Trim(),

reader["Этаж"].ToString().Trim(),

reader["НомерКвартиры"].ToString().Trim()

));

}

con.Close();

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "insertApart";

cmd.Parameters.Add("@Адрес", SqlDbType.NVarChar, 250).Value = АдресКвартиры.Text.ToString();

cmd.Parameters.Add("@Подъезд", SqlDbType.Int).Value = Подъезд.Text;

cmd.Parameters.Add("@Этаж", SqlDbType.Int).Value = Этаж.Text;

cmd.Parameters.Add("@НомерКвартиры", SqlDbType.Int).Value = НомерКвартиры.Text;

int result = cmd.ExecuteNonQuery();

if (result > 0)

ИзмененоКвартир.Text = "Добавлено квартир: " + result.ToString();

else

ИзмененоКвартир.Text = "Квартира не добавлена";

con.Close();

ВывестиКвартиры\_Click(sender, e);

}

private void ЗаполнитьComboBoxВКвартирах()

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

DataSet ds = new DataSet();

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select Адрес from Дом";

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = cmd;

da.Fill(ds);

АдресКвартиры.DisplayMember = "Адрес";

АдресКвартиры.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

private void СписокКвартир\_MouseClick(object sender, MouseEventArgs e)

{

if (СписокКвартир.SelectedItem != null)

{

КодКвартиры.Text = СписокКвартир.SelectedItem.ToString().Substring(0, 15).Trim();

АдресКвартиры.Text = СписокКвартир.SelectedItem.ToString().Substring(15, 30).Trim();

Подъезд.Text = СписокКвартир.SelectedItem.ToString().Substring(45, 10).Trim();

Этаж.Text = СписокКвартир.SelectedItem.ToString().Substring(55, 7).Trim();

НомерКвартиры.Text = СписокКвартир.SelectedItem.ToString().Substring(62).Trim();

}

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "updateApart";

cmd.Parameters.Add("@КодКвартиры", SqlDbType.Int).Value = КодКвартиры.Text.ToString();

cmd.Parameters.Add("@Адрес", SqlDbType.NVarChar, 250).Value = АдресКвартиры.Text.ToString();

cmd.Parameters.Add("@Подъезд", SqlDbType.Int).Value = Подъезд.Text;

cmd.Parameters.Add("@Этаж", SqlDbType.Int).Value = Этаж.Text;

cmd.Parameters.Add("@НомерКвартиры", SqlDbType.Int).Value = НомерКвартиры.Text;

int result = cmd.ExecuteNonQuery();

if (result > 0)

ИзмененоКвартир.Text = "Обновлено квартир: " + result.ToString();

else

ИзмененоКвартир.Text = "Квартира не обновлена";

con.Close();

ВывестиКвартиры\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "deleteApart";

cmd.Parameters.Add("@КодКвартиры", SqlDbType.Int).Value = КодКвартиры.Text.ToString();

int result = cmd.ExecuteNonQuery();

if (result > 0)

ИзмененоКвартир.Text = "Удалено квартир: " + result.ToString();

else

ИзмененоКвартир.Text = "Квартира не удалена";

con.Close();

ВывестиКвартиры\_Click(sender, e);

}

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

{

СписокКвартир.Items.Clear();

КодКвартиры.Text = "";

АдресКвартиры.Text = "";

Подъезд.Text = "";

Этаж.Text = "";

НомерКвартиры.Text = "";

ИзмененоКвартир.Text = "";

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select \* from ФизЛицоСотрудник\_КвартираФизЛицо\_Квартира";

SqlDataReader reader = cmd.ExecuteReader();

string listFormat = "{0,-35}{1,-20}{2,-20}{3,-25}{4,-15}{5,-20}{6,-20}{7,-20}{8,-25}{9,-15}{10,-15}{11,-20}";

СписокФизЛиц.Items.Clear();

СписокФизЛиц.Items.Add(String.Format(listFormat, "ФИО", "Паспорт", "Телефон", "Должность",

"Ставка", "Стаж", "email", "Рабочий телефон", "Адрес", "Номер квартиры", "Код квартиры", "Тип связи"));

while (reader.Read())

{

string strСтаж;

if (reader["Стаж"].ToString().Trim() == "") {

strСтаж = reader["Стаж"].ToString().Trim();

}

else strСтаж = reader["Стаж"].ToString().Substring(0, 10).Trim();

СписокФизЛиц.Items.Add(String.Format(listFormat,

reader["ФИО"].ToString().Trim(),

reader["Паспорт"].ToString().Trim(),

reader["Телефон"].ToString().Trim(),

reader["Должность"].ToString().Trim(),

reader["Ставка"].ToString().Trim(),

strСтаж,

reader["email"].ToString().Trim(),

reader["РабочийТелефон"].ToString().Trim(),

reader["Адрес"].ToString().Trim(),

reader["НомерКвартиры"].ToString().Trim(),

reader["КодКвартиры"].ToString().Trim(),

reader["ТипСвязи"].ToString().Trim()

));

}

con.Close();

}

private void СписокФизЛиц\_MouseClick(object sender, MouseEventArgs e)

{

if (СписокФизЛиц.SelectedItem != null)

{

ФИО.Text = СписокФизЛиц.SelectedItem.ToString().Substring(0, 35).Trim();

Паспорт.Text = СписокФизЛиц.SelectedItem.ToString().Substring(35, 20).Trim();

Телефон.Text = СписокФизЛиц.SelectedItem.ToString().Substring(55, 20).Trim();

Должность.Text = СписокФизЛиц.SelectedItem.ToString().Substring(75, 25).Trim();

Ставка.Text = СписокФизЛиц.SelectedItem.ToString().Substring(100, 15).Trim();

Стаж.Text = СписокФизЛиц.SelectedItem.ToString().Substring(115, 20).Trim();

email.Text = СписокФизЛиц.SelectedItem.ToString().Substring(135, 20).Trim();

РабочийТелефон.Text = СписокФизЛиц.SelectedItem.ToString().Substring(155, 20).Trim();

АдресКвартирыФизЛица.Text = СписокФизЛиц.SelectedItem.ToString().Substring(175, 25).Trim();

НомерКвартирыФизЛица.Text = СписокФизЛиц.SelectedItem.ToString().Substring(200, 15).Trim();

КодКвартирыФизЛица.Text = СписокФизЛиц.SelectedItem.ToString().Substring(215, 15).Trim();

ВидСвязи.Text = СписокФизЛиц.SelectedItem.ToString().Substring(230).Trim();

}

}

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

{

СписокФизЛиц.Items.Clear();

ФИО.Text = "";

Паспорт.Text = "";

Телефон.Text = "";

Должность.Text = "";

Ставка.Text = "";

Стаж.Text = "";

email.Text = "";

РабочийТелефон.Text = "";

АдресКвартирыФизЛица.Text = "";

НомерКвартирыФизЛица.Text = "";

КодКвартирыФизЛица.Text = "";

ВидСвязи.Text = "";

}

private void ЗаполнитьComboBoxВФизЛицах()

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

DataSet ds = new DataSet();

DataSet ds1 = new DataSet();

DataSet ds3 = new DataSet();

DataSet ds4 = new DataSet();

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select ФИО from ФизЛицо";

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = cmd;

da.Fill(ds);

ФИО.DisplayMember = "ФИО";

ФИО.DataSource = ds.Tables[0].DefaultView;

cmd.CommandText = "select Должность from Должности";

da.SelectCommand = cmd;

da.Fill(ds1);

Должность.DisplayMember = "Должность";

Должность.DataSource = ds1.Tables[0].DefaultView;

cmd.CommandText = "select Адрес from Дом";

da.SelectCommand = cmd;

da.Fill(ds3);

АдресКвартирыФизЛица.DisplayMember = "Адрес";

АдресКвартирыФизЛица.DataSource = ds3.Tables[0].DefaultView;

cmd.CommandText = "select КодКвартиры from Квартира";

da.SelectCommand = cmd;

da.Fill(ds4);

КодКвартирыФизЛица.DisplayMember = "КодКвартиры";

КодКвартирыФизЛица.DataSource = ds4.Tables[0].DefaultView;

con.Close();

}

private void ДобавитьФизЛицо\_MouseHover(object sender, EventArgs e)

{

label27.BackColor = Color.FromArgb(255, 153, 182);

label32.BackColor = Color.FromArgb(255, 153, 182);

label31.BackColor = Color.FromArgb(255, 153, 182);

}

private void ДобавитьФизЛицо\_MouseLeave(object sender, EventArgs e)

{

label27.BackColor = Color.Transparent;

label32.BackColor = Color.Transparent;

label31.BackColor = Color.Transparent;

}

private void ДобавитьСотрудника\_MouseHover(object sender, EventArgs e)

{

label27.BackColor = Color.FromArgb(255, 153, 182);

label30.BackColor = Color.FromArgb(255, 153, 182);

label28.BackColor = Color.FromArgb(255, 153, 182);

label29.BackColor = Color.FromArgb(255, 153, 182);

label34.BackColor = Color.FromArgb(255, 153, 182);

}

private void ДобавитьСотрудника\_MouseLeave(object sender, EventArgs e)

{

label27.BackColor = Color.Transparent;

label30.BackColor = Color.Transparent;

label28.BackColor = Color.Transparent;

label29.BackColor = Color.Transparent;

label34.BackColor = Color.Transparent;

}

private void УдалитьФизЛицо\_MouseHover(object sender, EventArgs e)

{

label27.BackColor = Color.FromArgb(255, 153, 182);

}

private void УдалитьФизЛицо\_MouseLeave(object sender, EventArgs e)

{

label27.BackColor = Color.Transparent;

}

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

{

label33.BackColor = Color.FromArgb(255, 153, 182);

label30.BackColor = Color.FromArgb(255, 153, 182);

}

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

{

label33.BackColor = Color.Transparent;

label30.BackColor = Color.Transparent;

}

private void ДобавитьЖильца\_MouseHover(object sender, EventArgs e)

{

label27.BackColor = Color.FromArgb(255, 153, 182);

label36.BackColor = Color.FromArgb(255, 153, 182);

label38.BackColor = Color.FromArgb(255, 153, 182);

}

private void ДобавитьЖильца\_MouseLeave(object sender, EventArgs e)

{

label27.BackColor = Color.Transparent;

label36.BackColor = Color.Transparent;

label38.BackColor = Color.Transparent;

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "InsertPerson";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

cmd.Parameters.Add("@Паспорт", SqlDbType.NVarChar, 12).Value = Паспорт.Text.ToString();

cmd.Parameters.Add("@Телефон", SqlDbType.NVarChar, 12).Value = Телефон.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "updatePerson";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

cmd.Parameters.Add("@Паспорт", SqlDbType.NVarChar, 12).Value = Паспорт.Text.ToString();

cmd.Parameters.Add("@Телефон", SqlDbType.NVarChar, 12).Value = Телефон.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "deletePerson";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "InsertRole";

cmd.Parameters.Add("@Должность", SqlDbType.NVarChar, 250).Value = Должность.Text.ToString();

cmd.Parameters.Add("@Ставка", SqlDbType.NVarChar, 250).Value = Ставка.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ЗаполнитьComboBoxВФизЛицах();

ФИО.Text = "";

АдресКвартирыФизЛица.Text = "";

КодКвартирыФизЛица.Text = "";

Должность.Text = "Должность добавлена";

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "InsertEmployee";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

cmd.Parameters.Add("@Должность", SqlDbType.NVarChar, 250).Value = Должность.Text.ToString();

cmd.Parameters.Add("@Стаж", SqlDbType.NVarChar, 250).Value = Стаж.Text.ToString();

cmd.Parameters.Add("@email", SqlDbType.NVarChar, 100).Value = email.Text.ToString();

cmd.Parameters.Add("@РабочийТелефон", SqlDbType.NVarChar, 12).Value = РабочийТелефон.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "updateEmployee";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

cmd.Parameters.Add("@Должность", SqlDbType.NVarChar, 250).Value = Должность.Text.ToString();

cmd.Parameters.Add("@Стаж", SqlDbType.NVarChar, 250).Value = Стаж.Text.ToString();

cmd.Parameters.Add("@email", SqlDbType.NVarChar, 100).Value = email.Text.ToString();

cmd.Parameters.Add("@РабочийТелефон", SqlDbType.NVarChar, 12).Value = РабочийТелефон.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "deleteEmployee";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "InsertTenant";

cmd.Parameters.Add("@КодКвартиры", SqlDbType.Int).Value = КодКвартирыФизЛица.Text.ToString();

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

cmd.Parameters.Add("@ТипСвязи", SqlDbType.NVarChar, 250).Value = ВидСвязи.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "updateTenant";

cmd.Parameters.Add("@КодКвартиры", SqlDbType.Int).Value = КодКвартирыФизЛица.Text.ToString();

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

cmd.Parameters.Add("@ТипСвязи", SqlDbType.NVarChar, 250).Value = ВидСвязи.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "deleteTenant";

cmd.Parameters.Add("@ФИО", SqlDbType.NVarChar, 250).Value = ФИО.Text.ToString();

int result = cmd.ExecuteNonQuery();

con.Close();

ВывестиФизЛиц\_Click(sender, e);

}

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

{

ВсеЗаявки.Checked = true;

}

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

{

ТолькоОткрытые.Checked = true;

}

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

{

ТолькоЗакрытые.Checked = true;

}

}

}

Текст Houses.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace curs

{

public static class Houses

{

public static List<string> Адрес = new List<string>();

public static List<string> КоличествоКвартир = new List<string>();

public static List<string> КоличествоПодъездов = new List<string>();

}

}

Текст HousesOperations.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.Data.SqlClient;

using System.Windows.Forms;

using System.Configuration;

namespace curs

{

public static class HousesOperations

{

public static void выводДомов()

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select \* from Дом";

SqlDataReader reader = cmd.ExecuteReader();

Houses.Адрес.Clear();

Houses.КоличествоКвартир.Clear();

Houses.КоличествоПодъездов.Clear();

while (reader.Read())

{

Houses.Адрес.Add(reader["Адрес"].ToString().Trim());

Houses.КоличествоКвартир.Add(reader["КоличествоКвартир"].ToString().Trim());

Houses.КоличествоПодъездов.Add(reader["КоличествоПодъездов"].ToString().Trim());

}

con.Close();

}

public static int добавлениеДома(string ПараметрАдресДома, string ПараметрКоличествоКвартир, string ПараметрКоличествоПодъездов)

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "InsertHome";

cmd.Parameters.Add("@Адрес", SqlDbType.NVarChar, 250).Value = ПараметрАдресДома;

cmd.Parameters.Add("@КоличествоКвартир", SqlDbType.Int).Value = ПараметрКоличествоКвартир;

cmd.Parameters.Add("@КоличествоПодъездов", SqlDbType.Int).Value = ПараметрКоличествоПодъездов;

int result = cmd.ExecuteNonQuery();

con.Close();

return result;

}

}

}

Текст Issues.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.Data.SqlClient;

using System.Windows.Forms;

using System.Configuration;

namespace curs

{

public static class Issues

{

public static List<string> НомерЗаявки = new List<string>();

public static List<string> ТипЗаявки = new List<string>();

public static List<string> ДатаСоздания = new List<string>();

public static List<string> СтатусЗаявки = new List<string>();

public static List<string> Регистратор = new List<string>();

public static List<string> Заявитель = new List<string>();

public static List<string> Исполнитель = new List<string>();

public static List<string> ДатаЗакрытия = new List<string>();

public static List<string> Адрес = new List<string>();

public static List<string> НомерКвартиры = new List<string>();

public static List<string> Примечание = new List<string>();

public static DataSet операторы = new DataSet();

public static DataSet типыЗаявок = new DataSet();

public static DataSet жильцы = new DataSet();

public static DataSet статусыЗаявок = new DataSet();

public static DataSet исполнители = new DataSet();

}

}

Текст IssuesOperations.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.Data.SqlClient;

using System.Windows.Forms;

using System.Configuration;

namespace curs

{

public static class IssuesOperations

{

public static void вывестиЗаявки(bool всеЗаявки, bool толькоОткрытые, bool толькоЗакрытые, bool найтиЗаявку = false, int номерЗаявки = 0)

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

if (всеЗаявки)

{

cmd.CommandText = "select \* from Заявки\_Квартиры";

}

else if (толькоОткрытые)

{

cmd.CommandText = "select \* from Заявки\_Квартиры where СтатусЗаявки != @ЗакрытыеЗаявки";

cmd.Parameters.Add("@ЗакрытыеЗаявки", SqlDbType.NVarChar, 250).Value = "Закрыта";

}

else if (толькоЗакрытые)

{

cmd.CommandText = "select \* from Заявки\_Квартиры where СтатусЗаявки = @ЗакрытыеЗаявки";

cmd.Parameters.Add("@ЗакрытыеЗаявки", SqlDbType.NVarChar, 250).Value = "Закрыта";

}

if (найтиЗаявку)

{

cmd.CommandText = "select \* from Заявки\_Квартиры where НомерЗаявки = @НомерЗаявки";

cmd.Parameters.Add("@НомерЗаявки", SqlDbType.Int).Value = номерЗаявки;

}

SqlDataReader reader = cmd.ExecuteReader();

Issues.НомерЗаявки.Clear();

Issues.ТипЗаявки.Clear();

Issues.ДатаСоздания.Clear();

Issues.СтатусЗаявки.Clear();

Issues.Регистратор.Clear();

Issues.Заявитель.Clear();

Issues.Исполнитель.Clear();

Issues.ДатаЗакрытия.Clear();

Issues.Адрес.Clear();

Issues.НомерКвартиры.Clear();

Issues.Примечание.Clear();

while (reader.Read())

{

Issues.НомерЗаявки.Add(reader["НомерЗаявки"].ToString().Trim());

Issues.ТипЗаявки.Add(reader["ТипЗаявки"].ToString().Trim());

Issues.ДатаСоздания.Add(reader["ДатаСоздания"].ToString().Substring(0, 16).Trim());

Issues.СтатусЗаявки.Add(reader["СтатусЗаявки"].ToString().Trim());

Issues.Регистратор.Add(reader["Регистратор"].ToString().Trim());

Issues.Заявитель.Add(reader["Заявитель"].ToString().Trim());

Issues.Исполнитель.Add(reader["Исполнитель"].ToString().Trim());

Issues.ДатаЗакрытия.Add(reader["ДатаЗакрытия"].ToString().Trim());

Issues.Адрес.Add(reader["Адрес"].ToString().Trim());

Issues.НомерКвартиры.Add(reader["НомерКвартиры"].ToString().Trim());

Issues.Примечание.Add(reader["Примечание"].ToString().Trim());

}

con.Close();

}

public static void добавитьЗаявку(string ТипЗаявки, string Регистратор, string Заявитель, string Примечание)

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "insertIssue";

cmd.Parameters.Add("@ТипЗаявки", SqlDbType.NVarChar, 150).Value = ТипЗаявки;

cmd.Parameters.Add("@Регистратор", SqlDbType.NVarChar, 250).Value = Регистратор;

if (Заявитель != "")

{

cmd.Parameters.Add("@Заявитель", SqlDbType.NVarChar, 250).Value = Заявитель;

}

cmd.Parameters.Add("@Примечание", SqlDbType.NVarChar, 250).Value = Примечание;

int result = cmd.ExecuteNonQuery();

con.Close();

}

public static void удалитьЗаявку(string НомерЗаявки)

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "deleteIssue";

cmd.Parameters.Add("@НомерЗаявки", SqlDbType.Int).Value = НомерЗаявки;

int result = cmd.ExecuteNonQuery();

con.Close();

}

public static void обновитьЗаявку(string НомерЗаявки, string СтатусЗаявки, string Исполнитель, string Примечание)

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "updateIssue";

cmd.Parameters.Add("@НомерЗаявки", SqlDbType.Int).Value = НомерЗаявки;

cmd.Parameters.Add("@СтатусЗаявки", SqlDbType.NVarChar, 50).Value = СтатусЗаявки;

if (Исполнитель != "")

{

cmd.Parameters.Add("@Исполнитель", SqlDbType.NVarChar, 250).Value = Исполнитель;

}

cmd.Parameters.Add("@Примечание", SqlDbType.NVarChar, 250).Value = Примечание;

int result = cmd.ExecuteNonQuery();

con.Close();

}

public static void заполнитьВыпадающееМеню(string ТипЗаявки = "")

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = new SqlCommand("select ФИО from Сотрудник where Должность = 'Оператор'", con);

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = cmd;

Issues.операторы.Clear();

da.Fill(Issues.операторы);

cmd = new SqlCommand("select ТипЗаявки from ТипыЗаявок", con);

da.SelectCommand = cmd;

Issues.типыЗаявок.Clear();

da.Fill(Issues.типыЗаявок);

cmd = new SqlCommand("select ФИО from Квартира\_ФизЛицо", con);

da.SelectCommand = cmd;

Issues.жильцы.Clear();

da.Fill(Issues.жильцы);

cmd = new SqlCommand("select СтатусЗаявки from СтатусыЗаявок", con);

da.SelectCommand = cmd;

Issues.статусыЗаявок.Clear();

da.Fill(Issues.статусыЗаявок);

if (ТипЗаявки != "") {

cmd = new SqlCommand("select ФИО from ТипыЗаявок\_Должность\_Сотрудники where ТипЗаявки = @ТипЗаявки", con);

cmd.Parameters.Add("@ТипЗаявки", SqlDbType.NVarChar, 150).Value = ТипЗаявки;

da.SelectCommand = cmd;

Issues.исполнители.Clear();

da.Fill(Issues.исполнители);

}

con.Close();

}

}

}

Текст IssuesHistory.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.Data.SqlClient;

using System.Windows.Forms;

using System.Configuration;

namespace curs

{

public static class IssuesHistory

{

public static List<string> ID = new List<string>();

public static List<string> НомерЗаявки = new List<string>();

public static List<string> СтатусЗаявки = new List<string>();

public static List<string> Исполнитель = new List<string>();

public static List<string> Дата = new List<string>();

public static List<string> Примечание = new List<string>();

}

}

Текст IssuesHistoryOperations.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.Data.SqlClient;

using System.Windows.Forms;

using System.Configuration;

namespace curs

{

class IssuesHistoryOperations

{

public static void вывестиИсториюЗаявки(bool ВСпискеЗаявокВыбранЭлемент, bool ВСпискеЗаявокВыбраноНеОглавление, string НомерЗаявки)

{

if ((ВСпискеЗаявокВыбранЭлемент) & (ВСпискеЗаявокВыбраноНеОглавление))

{

SqlConnection con = new SqlConnection();

con.ConnectionString = ConfigurationManager.ConnectionStrings["Config"].ConnectionString;

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select \* from ИсторияЗаявки where НомерЗаявки = @НомерЗаявки";

cmd.Parameters.Add("@НомерЗаявки", SqlDbType.Int).Value = НомерЗаявки;

SqlDataReader reader = cmd.ExecuteReader();

IssuesHistory.ID.Clear();

IssuesHistory.НомерЗаявки.Clear();

IssuesHistory.СтатусЗаявки.Clear();

IssuesHistory.Исполнитель.Clear();

IssuesHistory.Дата.Clear();

IssuesHistory.Примечание.Clear();

while (reader.Read())

{

IssuesHistory.ID.Add(reader["ID"].ToString().Trim());

IssuesHistory.НомерЗаявки.Add(reader["НомерЗаявки"].ToString().Trim());

IssuesHistory.СтатусЗаявки.Add(reader["СтатусЗаявки"].ToString().Trim());

IssuesHistory.Исполнитель.Add(reader["Исполнитель"].ToString().Trim());

IssuesHistory.Дата.Add(reader["Дата"].ToString().Trim());

IssuesHistory.Примечание.Add(reader["Примечание"].ToString().Trim());

}

con.Close();

}

}

}

}