

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

Кафедра ЭВМ
К защите
Руководитель работы:

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ**

по дисциплине

«Клиент серверные приложения баз данных»

Тема:

«Разработка информационной системы общежития»

Выполнил студент группы 145

Жупин С.Ю.

дата сдачи на проверку, подпись

Руководитель работы
ассистент каф. ЭВМ
Баранова С.Н.

оценка

дата защиты, подпись

Рязань 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Выявление задач автоматизации	5
1.1 Описание автоматизированной области	5
1.2 Обоснование актуальности разработки	6
1.3 Постановка задачи	6
1.4 Разработка архитектуры ИС	6
2 Разработка серверной части ИС	9
2.1 Инфологическое проектирование БД	9
2.1.1 Требуемая информация	9
2.1.2 Выделение сущностей	10
2.1.3 Выделение связей	11
2.1.4 Построение ER диаграммы (Рисунки 2-11)	11
2.2 Дatalogическое проектирование БД	15
2.2.1 Формирование предварительных отношений	15
2.2.2 Распределение атрибутов по отношениям	18
2.2.3 Проверка отношений на БКНФ	20
2.3 Разработка объектов поддержания целостности данных	23
2.3.1 Разработка правил, умолчаний и типов	23
2.3.2 Разработка триггеров	26
2.3.3 Разработка процедур	27
3 Разработка клиентской части ИС	28
3.1 Разработка прототипа интерфейса пользователя	28
3.2 Реализация интерфейса пользователя	31
4 Тестирование основных функций приложения	36
ЗАКЛЮЧЕНИЕ	40
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	41
ПРИЛОЖЕНИЕ А: сценарий создания объектов БД	42
ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД	51

ПРИЛОЖЕНИЕ В:исходный код клиентского приложения	54
--	----

ВВЕДЕНИЕ

На сегодняшний день информационные системы (далее ИС) применяются во множестве различных сферах деятельности человека. ИС позволяют автоматизировать и ускорить процессы обработки, передачи и создания различных данных тем самым позволяя оптимизировать различные прикладные процессы. Благодаря цифровому представлению данных можно избавиться от большого количества бумажных архивов, значительно ускорить передачу информации и упростить обработку информации, а также обеспечить предоставление информации в удобной форме.

В данном курсовом проекте будет создана информационная система общежития, которая будет состоять из сервера состоящем из базы данных MS SQL Server и клиентского приложения на платформе Windows. Данная ИС позволит хранить, добавлять и обрабатывать информацию, требуемую для администрирования общежития в удобном виде, а также позволит делать это удаленно.

1 Выявление задач автоматизации

1.1 Описание автоматизированной области

Необходимо спроектировать ИС для студенческого общежития РГРТУ, предоставляющей проживание для студентов как самого РГРТУ, так и других вузов и профессиональных училищ. ИС. ИС разрабатывается для одного общежития, но при необходимости ее можно будет улучшить для администрирования сразу множества общежитий, например, в пределах одного студенческого городка.

Система хранит информацию о проживающих в общежитии студентах с указанием их персональных данных, местом их обучения, а также о тарифе и долгах за проживание.

Система хранит информацию о структуре общежития, такую как список блоков с указанием их типов и назначениях.

Система хранит информацию о всех помещениях общежития с указанием типа комнаты, ее принадлежности к блоку и количеству доступных для заселения мест.

Система хранит информацию о работниках общежития (вахтеры, уборщики, слесари и др.) с указанием их персональных данных и занимаемой должности.

Система хранит информацию о существующих в общежитии должностях. Должность определяет базовый оклад работника.

Ведется отслеживание различного рода инвентаря (мебель, фурнитура и т.д.) и хранение информации о нем, такой как название и описание, стоимость, дата поставки и комната его размещения.

При заселении нового или переселении уже существующего в системе студента администратор общежития вносит соответствующие изменения.

Проживающие в общежитии могут создать заявку с жалобой или предложением, на которую администратором будет назначен ответственный за ее решение из работников общежития.

При устройстве на работу в общежитии администратор вносит нового работника в систему с указанием его персональных данных и должности.

Администратор назначает дежурных по блокам общежития из работников, а также вахтеров на вахты из ранее размеченного расписания вахт.

Администратор составляет расписание вахт в общежитии с указанием даты захода и длительностью смены.

1.2 Обоснование актуальности разработки

При изучении предметной области было замечено что многая информация о проживающих в общежитии хранится в бумажном виде, а также отсутствует доступ о данных студентах из деканатов образовательных организаций, из-за этого возникают ситуации беготни с бумагами. Также хранение информации в бумажном виде сильно ограничивает скорость ее обработки и увеличивает шансы допущения ошибок, которые могут нарушить целостность данных.

Также электронное представление данных значительно оптимизирует работу администратора общежития что позволит повысить производительность труда.

1.3 Постановка задачи

Для данной предметной области основными задачами автоматизации являются:

- автоматизация операций заселения/переселения проживающих;
- автоматизация операций с заявками и предложениями;
- автоматизация инвентаризации;
- автоматизация назначения работ для работников общежития;
- удобный и быстрый поиск и отображение информации;
- удаленный просмотр информации.

1.4 Разработка архитектуры ИС

Для данной ИС была выбрана двухуровневая клиент-серверная

архитектура. Она позволяет значительно сэкономить на серверной части, так как в отличие от распределенных систем здесь требуется только один физический сервер.

Из других преимуществ такая архитектура способна поддерживать целостность данных на обоих уровнях и равномерно распределять нагрузку между клиентом и сервером. Также такая архитектура позволяет организовать централизованную защиту данных, т.к. все данные будут храниться на сервере. Клиентское приложение в свою очередь будет отправлять запросы на сервер и выводить пользователю полученную информацию в форматированном виде. Также с клиента можно отправлять параметризованные запросы с различными ограничениями на добавление/изменение/удаление данных, которые позволят исключить нарушение целостности данных и SQL-инъекции.

Централизованное хранилище данных является не только преимуществом, но и недостатком, так как при нарушении работы сервера все экземпляры приложений-клиентов потеряют доступ к данным, а также при несанкционированном доступе будет утечка сразу всех данных.

Схема архитектуры ИС представлена на рисунке 1. Виртуальный сервер БД физически располагается на устройстве-сервере, также сервер БД представляет часть слоя логики и выполняет большинство операций по обработке данных. Клиентское приложение содержит в себе остальную часть слоя логики который формирует запросы к серверу и слой представления данных полученных от сервера.

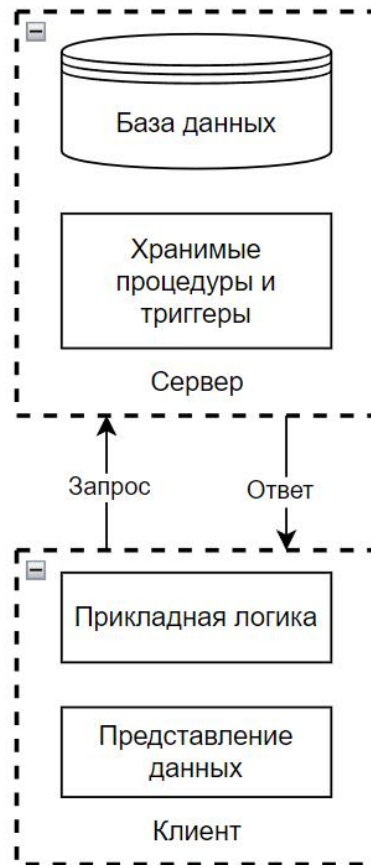


Рисунок 1 – Архитектура ИС

2 Разработка серверной части ИС

2.1 Инфологическое проектирование БД

2.1.1 Требуемая информация

Исходя из предметной области в БД требуется хранить следующую информацию:

1. Информацию о проживающих:

- ФИО;
- серия и номер паспорта;
- пол (муж/жен);
- дата рождения;
- номер телефона для связи;
- учебное заведение, в котором он обучается;
- комната проживания;
- тариф, который зависит от льгот и места обучения.

2. Информацию о комнате:

- тип комнаты;
- номер блока, в котором она находится;
- количество койко-мест, которое зависит от типа комнаты.

3. Информацию о блоках:

- тип блока, который определяет какие типы проживающих рекомендуется заселять в блок;
- принадлежность блока к крылу и этажу;
- информацию о работнике являющийся дежурным по блоку.

4. Информацию об инвентаре:

- наименование и описание;
- дата поставки;
- стоимость;
- номер комнаты размещения.

5. Информацию о вахтах:

- тип вахты;
- ставка почасового оклада за вахту в зависимости от типа вахты;
- дата и время заступления на вахту;
- длительность вахты в часах;
- назначенный на вахту работник общежития.

6. Информацию о работниках общежития:

- ФИО;
- серия и номер паспорта;
- телефон для связи;
- должность;
- базовый оклад, зависящий от должности.

7. Информацию о заявках:

- тема для удобного поиска и сортировки;
- основной текст;
- статус заявки (создана, закрыта, в процессе);
- дата создания;
- составитель из проживающих.

2.1.2 Выделение сущностей

Для хранения вышеуказанной информации необходимо выделить следующие сущности:

- Проживающий (Код проживающего);
- Тип проживающего (Название типа);
- Комната (Номер комнаты);
- Тип комнаты (Название типа);
- Блок (Номер блока);
- Тип блока (Название типа);
- Инвентарь (Наименование);
- Работник (Код работника);
- Должность (Название должности);

- Вахта (Номер вахты);
- Тип вахты (Название типа);
- Заявка (Номер заявки).

2.1.3 Выделение связей

- проживающий имеет тип проживающего;
- проживающий проживает в комнате;
- заявка составляется проживающим;
- комната имеет тип комнаты;
- комната содержит инвентарь;
- блок содержит комнаты;
- работник дежурит в блоке;
- работник дежурит на вахтах;
- работник занимает должность;
- вахта имеет тип вахты.

2.1.4 Построение ER диаграммы (Рисунки 2-11)

Проживающий имеет тип:

- проживающий имеет только один тип;
- проживающий обязательно имеет тип;
- один тип может иметь множество проживающих;
- тип не обязательно должен иметь проживающих.



Рисунок 2 – Проживающий имеет тип

Проживающий проживает в комнате:

- проживающий живет только в одной комнате;
- проживающий обязательно должен проживать в комнате;
- в одной комнате могут жить несколько проживающих;
- в комнате не обязательно должны проживать.

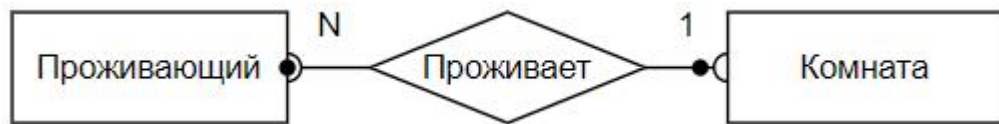


Рисунок 3 – Проживающий проживает в комнате

Заявка составляется проживающим:

- проживающий может составить множество заявок;
- проживающему не обязательно составлять заявки;
- одну заявку составляет только один проживающий;
- заявка обязательно содержит информацию о составителе.

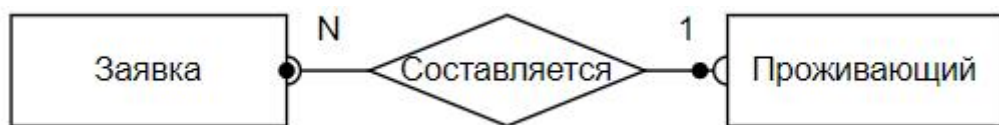


Рисунок 4 – Заявка составляется проживающим

Комната имеет тип:

- комната обязательно имеет тип;
- комната может иметь только один тип;
- одному типу может соответствовать множество комнат;
- типу не обязательно иметь соответствующую ему комнату.



Рисунок 5 – Комната имеет тип

Комната содержит инвентарь:

- в одной комнате может содержаться множество инвентаря;
- в комнате может не быть инвентаря;
- инвентарь обязательно имеет комнату, в которой он размещается;
- единица инвентаря размещается только в одной комнате.



Рисунок 6 – Комната содержит инвентарь

Блок содержит комнаты:

- в одном блоке может быть множество комнат;
- комната обязательно должна иметь блок, в котором она находится;
- комната может находиться только в одном блоке;
- блок необязательно должен иметь комнаты;



Рисунок 7 – Блок содержит комнаты

Работник дежурит в блоке:

- в одном блоке может дежурить только один работник;
- в блоке необязательно должны дежурить;
- работник может дежурить в множестве блоках;
- работнику необязательно дежурить.



Рисунок 8 – Работник дежурит в блоке

Работник дежурит на вахтах:

- работнику необязательно дежурить на вахте;
- работник может дежурить на нескольких вахтах;
- на одну вахту назначается только один работник;
- вахте обязательно должен быть назначен работник.



Рисунок 9 – Работник дежурит на вахтах

Работник занимает должность:

- работник занимает только одну должность;
- работнику обязательно занимать должность;
- на одной должности может состоять множество работников;

– должность может никто не занимать.

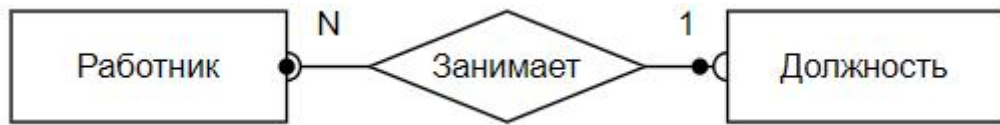


Рисунок 10 – Работник занимает должность

Вахта имеет тип вахты:

- вахта имеет только один тип вахты;
- вахта обязательно имеет тип;
- одному типу может соответствовать множество вахт;
- тип может не иметь соответствующих ему вахт.



Рисунок 11 – Вахта имеет тип вахты

Полная ER диаграмма представлена на рисунке 12.

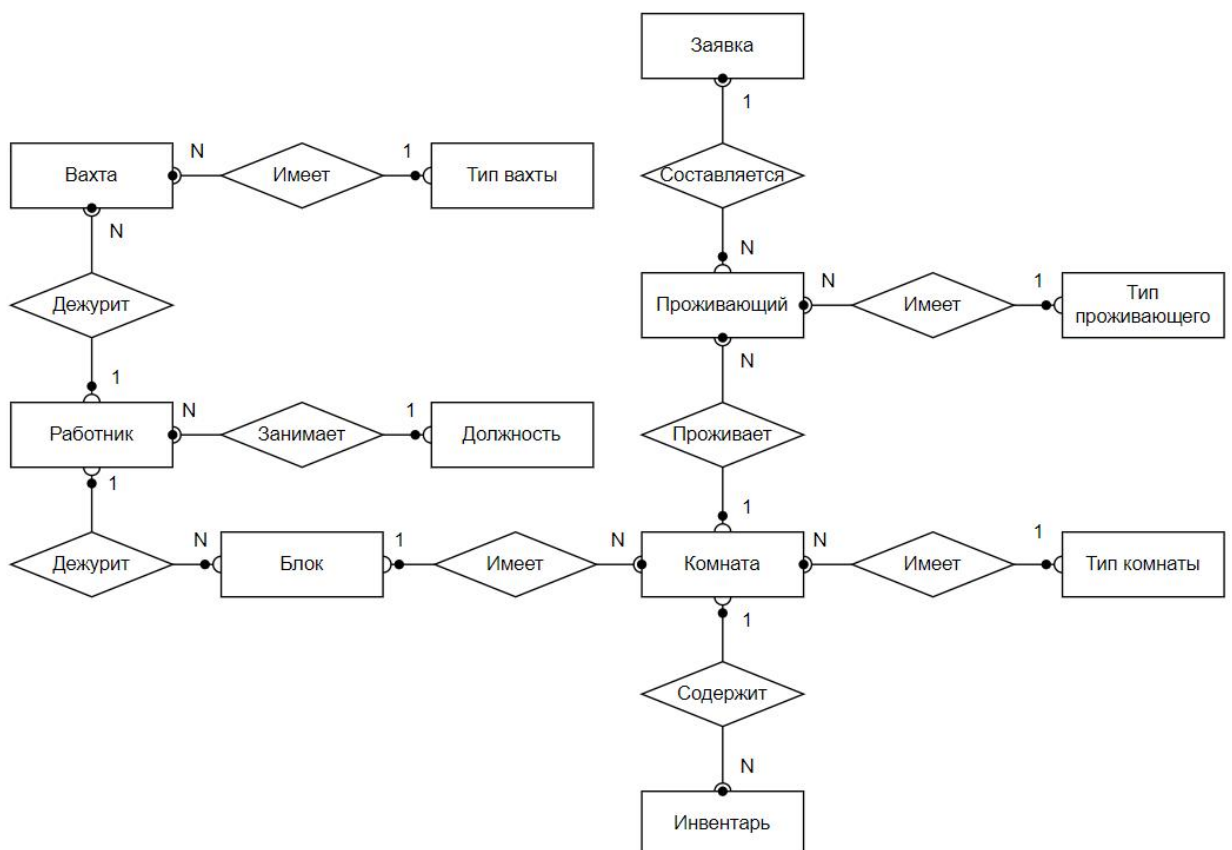


Рисунок 12 – Полная ER диаграмма

2.2 Даталогическое проектирование БД

2.2.1 Формирование предварительных отношений

Правила для формирования предварительных отношений:

- правило 1: если степени бинарной связи один-к-одному и класс принадлежности обеих сущностей к связи обязательный, то формируют одно отношение, ключом этого отношения может быть ключ любой из двух сущностей;

- правило 2: если степень бинарной связи один-к-одному, а класс принадлежности сущности к связи для одной — обязательный, а для другой — необязательный, то формируется два отношения: по одному для каждой сущности. При этом ключом отношения 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 бинарной связи — 1:1, а класс принадлежности обеих сущностей к связи необязательный, то формируется три отношения, по одному для каждой сущности и одно для связи. При этом ключом в первых двух отношениях является ключ соответствующей сущности, отношение для связи должно содержать ключи обеих сущностей, в отношении для связи в качестве ключа можно принять ключ любой из сущностей.

- правило 4: если степень бинарной связи 1:N, и класс принадлежности N-связной сущности обязательный, то формируют два отношения, по одному для каждой сущности. При этом ключом в каждом отношении является ключ соответствующей сущности. Кроме этого в отношение для N-связной сущности добавляется в качестве неключевого атрибута ключ односвязной сущности;

- правило 5: если степень бинарной связи 1:N, и класс принадлежности N-связной сущности необязательный, то формируют три

отношения: по одному для каждой сущности и одно для связи. При этом ключом в двух первых отношениях является ключ соответствующей сущности, отношение для связи должно содержать ключи обеих сущностей, при этом ключом отношения для связи является ключ N-связной сущности;

– правило 6: если степень бинарной связи N:N, то формируется три отношения: по одному для каждой сущности и одно для связи. При этом ключом в первых двух отношениях является ключ соответствующей сущности, отношение для связи должно содержать ключи обеих сущностей, которые совместно образуют ключ данного отношения.

Формирование предварительных отношений для каждой связи производится с указанными выше правилами.

Связь «Проживающий имеет тип проживающего» по правилу 4 формирует 2 отношения:

- Проживающий(Проживающий, ТипПроживающего);
- ТипПроживающего(ТипПроживающего).

Связь «Проживающий проживает в комнате» по правилу 4 формирует 2 отношения:

- Проживающий(Проживающий, ТипПроживающего, Комната);
- Комната(Комната).

Связь «Заявка составляется проживающим» по правилу 4 формирует 2 отношения:

- Заявка(Заявка, Проживающий);
- Проживающий(Проживающий, ТипПроживающего).

Связь «Комната имеет тип комнаты» по правилу 4 формирует 2 отношения:

- Комната(Комната, ТипКомнаты);
- ТипКомнаты(ТипКомнаты).

Связь «Комната содержит инвентарь» по правилу 4 формирует 2 отношения:

- Инвентарь(Инвентарь, Комната);
- Комната(Комната, ТипКомнаты).

Связь «Блок содержит комнаты» по правилу 4 формирует 2 отношения:

- Комната(Комната, ТипКомнаты,Блок);
- Блок(Блок).

Связь «Работник дежурит в блоке» по правилу 5 формирует 3 отношения:

- Дежурство(Блок, Работник);
- Блок(Блок);
- Работник(Работник).

Связь «Работник дежурит на вахте» по правилу 4 формирует 2 отношения:

- Вахта(Вахта, Работник);
- Работник(Работник).

Связь «Работник занимает должность» по правилу 4 формирует 2 отношения:

- Работник(Работник, Должность);
- Должность(Должность).

Связь «Вахта имеет тип вахты» по правилу 2 формирует 4 отношения:

- Вахта(Вахта, Работник ,ТипВахты);
- ТипВахты(ТипВахты);

Полная предварительная диаграмма отношений представлена на рисунке 14

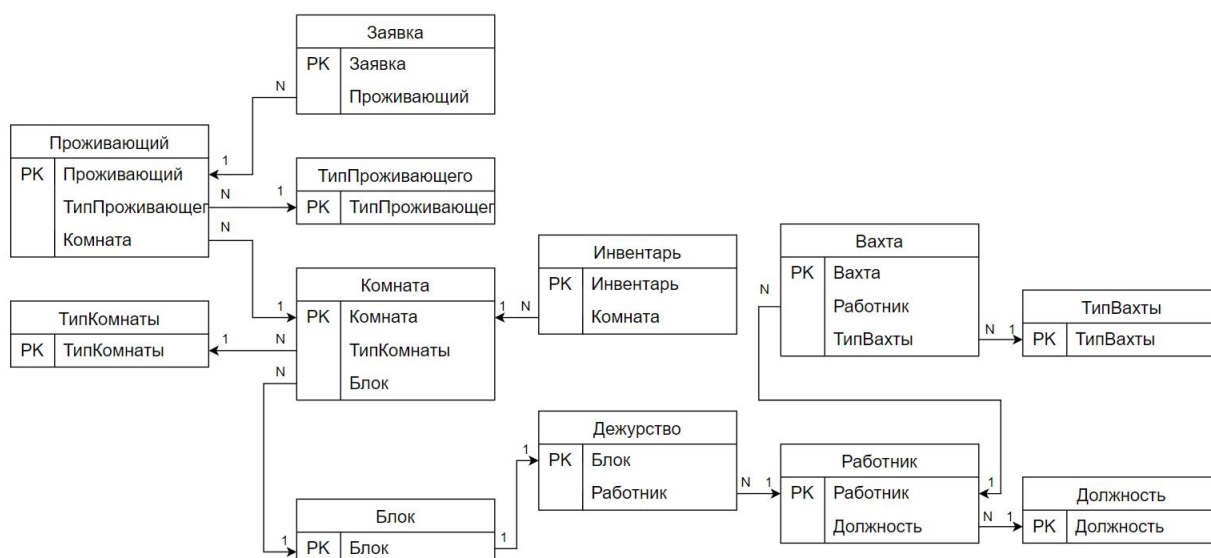


Рисунок 13 – Предварительная диаграмма отношений

2.2.2 Распределение атрибутов по отношениям

Отношение «Комната» содержит атрибуты:

- Номер комнаты;
- Тип;
- Количество мест;
- Блок.

Отношение «Проживающий» содержит атрибуты:

- Код проживающего (серия и номер паспорта);
- ФИО;
- Телефон;
- Пол;
- Дата рождения;
- Тип;
- Номер комнаты.

Отношение «Работник» содержит атрибуты:

- Код работника (серия и номер паспорта);
- ФИО;
- Телефон;
- Должность.

Отношение «Блок» содержит атрибуты:

- Код блока;
- Этаж;
- Крыло;
- Тип.

Отношение «Инвентарь» содержит атрибуты:

- Код инвентаря;
- Название;
- Дата поставки;
- Стоимость;
- Комната.

Отношение «Заявки» содержит атрибуты:

- Номер;
- Тема;
- Текст;
- Статус;
- Дата составления;
- Составитель.

Отношение «Вахты» содержит атрибуты:

- Номер;
- Тип;
- Дата захода;
- Длительность;
- Вахтер.

Отношение «Дежурство» содержит атрибуты:

- Блок;
- Дежурный.

Отношение «ТипПроживающего» содержит атрибуты:

- Название (место обучения + льготы);

– Тариф.

Отношение «ТипКомнаты» содержит атрибуты:

– Название;

– Количество мест.

Отношение «Должность» содержит атрибуты:

– Название;

– Оклад.

Отношение «ТипВахты» содержит атрибуты:

– Название;

– Ставка.

2.2.3 Проверка отношений на БКНФ

Все отношения находятся:

– В 1 НФ, так как все атрибуты являются атомарными;

– Во 2НФ, поскольку все неключевые элементы функционально полно зависят от первичного ключа;

– В 3НФ, так как в нем нет транзитивных зависимостей;

– В БКНФ, потому что детерминант функциональных зависимостей является единственным потенциальным ключом (первичным), исключением является отношения «Проживающий» и «Работник», которые имеют несколько потенциальных ключей (атрибуты «Код проживающего» и «Телефон», и «Код Работника» и «Телефон»), но данный случай можно опустить по причине малой целесообразности дополнительного разбиения.

Схемы функциональных зависимостей показаны на рисунках 14-21.

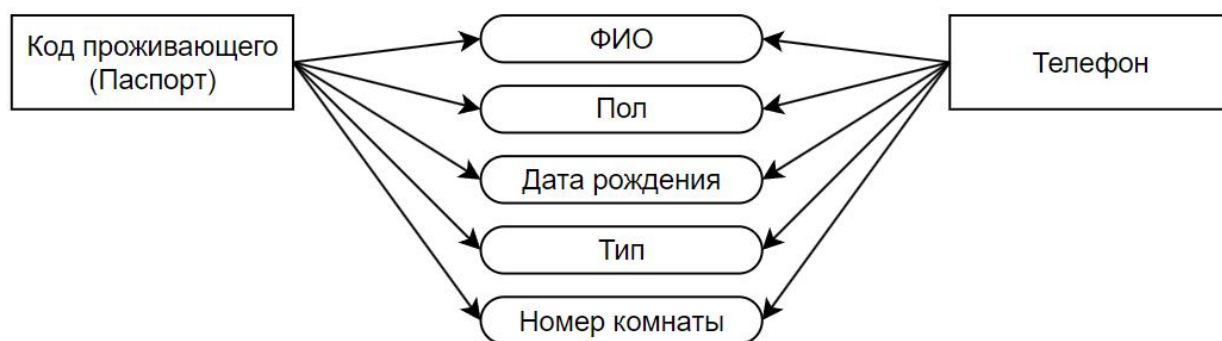


Рисунок 14 – Зависимости отношения «Проживающий»

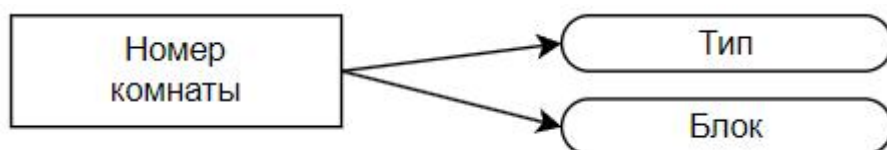


Рисунок 15 – Зависимости отношения «Комната»

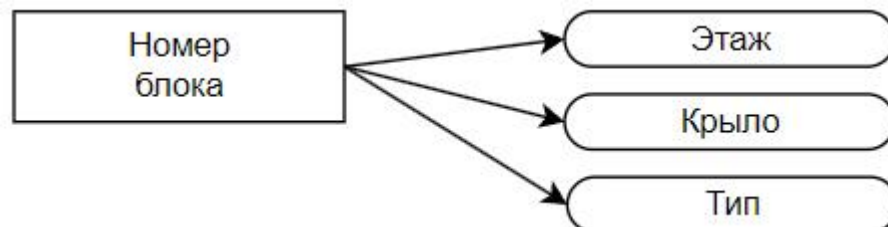


Рисунок 16 – Зависимости отношения «Блок»

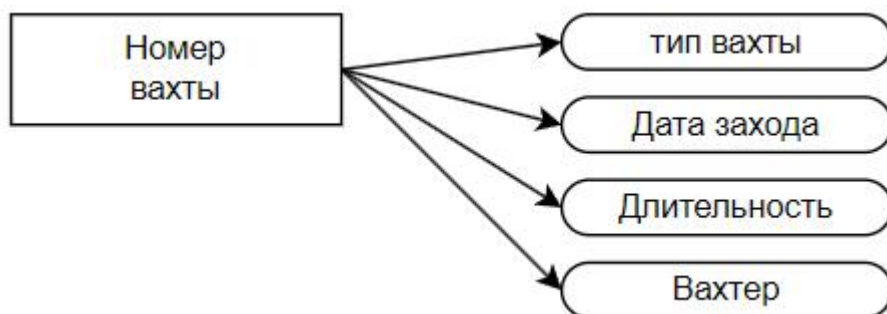


Рисунок 17 – Зависимости отношения «Вахта»



Рисунок 18 – Зависимости отношения «Работник»

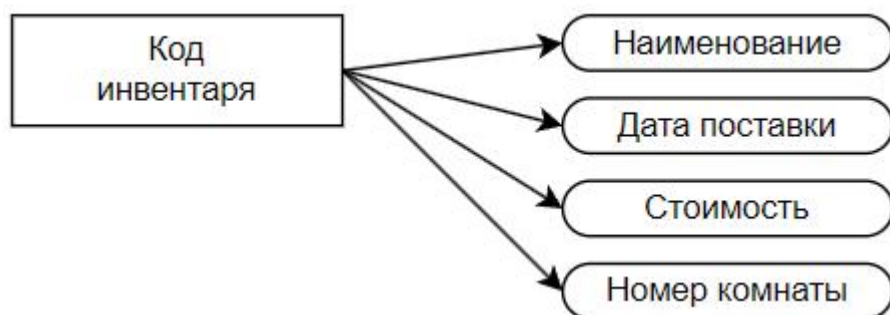


Рисунок 19 – Зависимости отношения «Инвентарь»

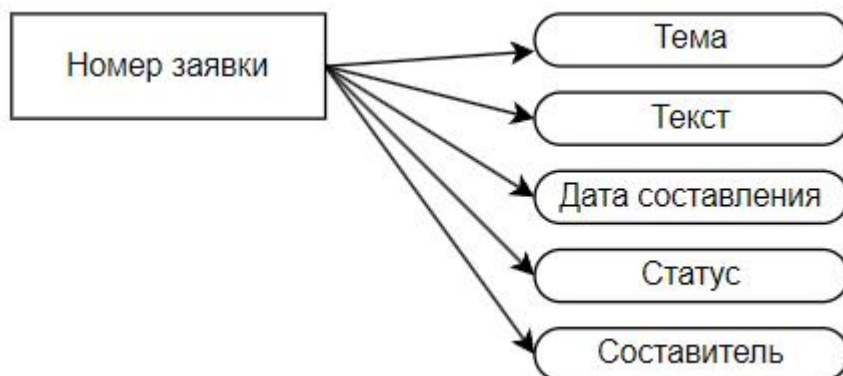


Рисунок 20 – Зависимости отношения «Заявка»

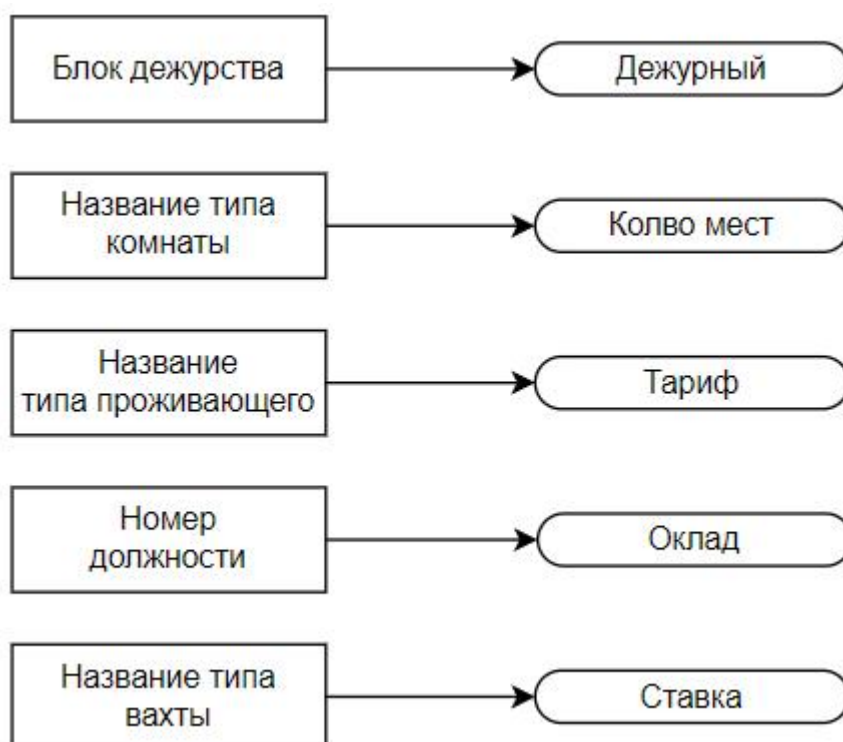


Рисунок 21 – Зависимости отношений «Дежурство», «ТипКомнаты», «ТипПроживающего», «Должность» и «ТипВахты»

Полная схема БД представлена на рисунке 22.

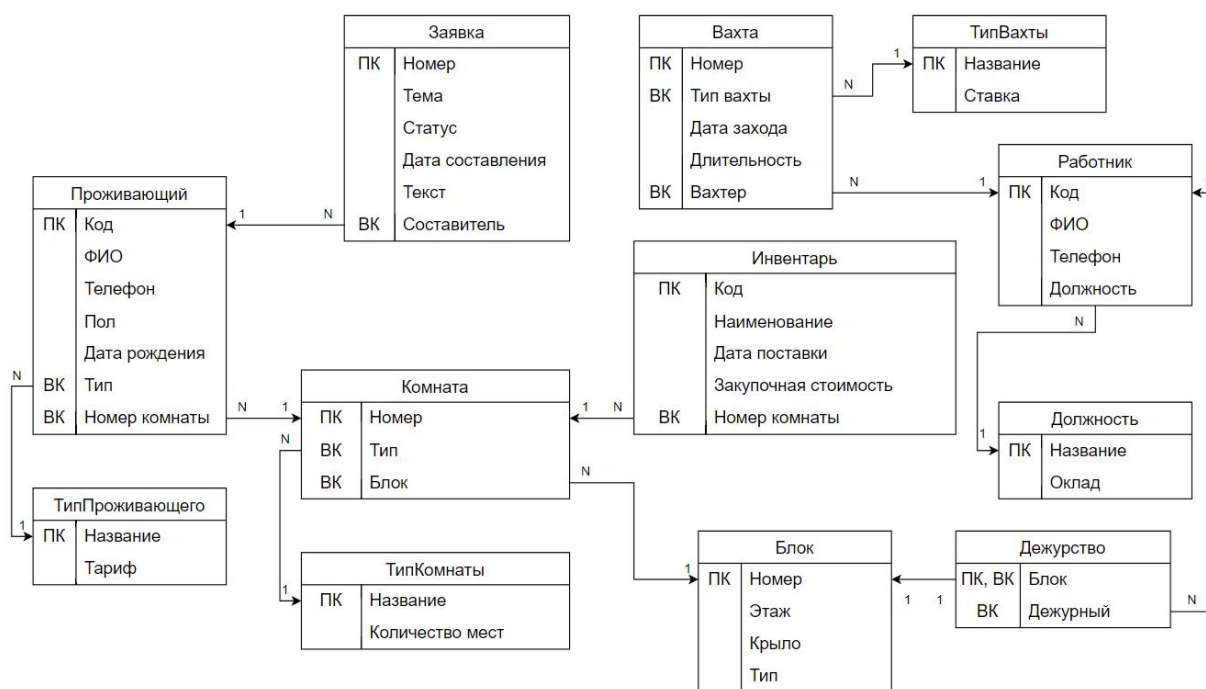


Рисунок 22 – Полная схема БД

2.3 Разработка объектов поддержания целостности данных

Сценарии создания объектов БД представлены в ПРИЛОЖЕНИИ А.

2.3.1 Разработка правил, умолчаний и типов

Таблица «Проживающие»:

- атрибут «Код» (номер паспорта) является первичным ключем, не может принимать NULL значения, значения находятся в диапазоне 1000000000 – 9999999999;
- атрибут «ФИО» не может принимать NULL значения, значения должны соответствовать шаблону [А-Я]% [А-Я]% [А-Я]%;
- атрибут «Телефон» не может принимать NULL значения, значения должны соответствовать шаблону +7([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9] или 0([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9];
- атрибут «Пол» не может принимать NULL значения, значения могут быть только «муж» или «жен»;
- атрибут «Дата рождения» не может принимать NULL значения;
- атрибут «Тип» не может принимать NULL значения;

- атрибут «Комната» не может принимать NULL значения.

Таблица «Комнаты»:

- атрибут «Номер» является первичным ключом, не может принимать NULL значения, не может быть отрицательным;
- атрибут «Тип» не может принимать NULL значения;
- атрибут «Блок» не может принимать NULL значения.

Таблица «Заявки»:

- атрибут «Номер» является первичным ключом, не может принимать NULL значения;
- атрибут «Тема» не может принимать NULL значения, значение по умолчанию «Без темы»;
- атрибут Статус не может принимать NULL значения, список допустимых значений: «Открыта», «Выполняется», «Отклонена» или «Выполнена», значение по умолчанию «Открыта»;
- атрибут «ДатаСоставления» не может принимать NULL значения, по умолчанию устанавливается текущая дата и время;
- атрибут Текст является необязательным;
- атрибут Составитель не может принимать NULL значения.

Таблица «Вахты»:

- атрибут «Номер» является первичным ключом, не может принимать NULL значения;
- атрибут «Тип» не может принимать NULL значения;
- атрибут «ДатаНачала» не может принимать NULL значения, по умолчанию устанавливается текущая дата и время;
- атрибут «Длительность» не может принимать NULL значения и значения меньше нуля;
- атрибут «Вахтер» не может принимать NULL значения.

Таблица «Инвентарь»:

- атрибут «Код» является первичным ключом, не может принимать

NULL значения;

- атрибут «Наименование» не может принимать NULL значения;
- атрибут «ДатаПоставки» не может принимать NULL значения, по умолчанию устанавливается текущая дата и время;
- атрибут «Стоимость» не может принимать NULL значения и отрицательные значения;
- атрибут «Комната» не может принимать NULL значения.

Таблица «Блоки»:

- атрибут «Номер» является первичным ключом, не может принимать NULL значения;
- атрибут «Этаж» не может принимать NULL значения;
- атрибут «Крыло» не может принимать NULL значения;
- атрибут «Тип» не может принимать NULL значения, список допустимых значений: «Мужской», «Женский», «Общий» «Преподавательский», «Семейный» или «Служебный» .

Таблица «Работники»:

- атрибут «Код» (номер паспорта) является первичным ключом, не может принимать NULL значения, значения находятся в диапазоне 1000000000 – 9999999999;
- атрибут «ФИО» не может принимать NULL значения, значения должны соответствовать шаблону [А-Я]% [А-Я]% [А-Я]%;
- атрибут «Телефон» не может принимать NULL значения, значения должны соответствовать шаблону +7([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9] или 0([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9];
- атрибут «Должность» не может принимать NULL значения.

Таблица «Дежурство»:

- атрибут «Блок» является первичным ключом, не может принимать NULL значения;
- атрибут «Дежурный» не должен принимать NULL значения.

Таблица «ТипыПроживающих»:

- атрибут «Название» является первичным ключем, не может принимать NULL значения;
- атрибут «Тариф» не должен принимать NULL значения, не может быть отрицательным.

Таблица «ТипыКомнат»:

- атрибут «Название» является первичным ключем, не может принимать NULL значения;
- атрибут «КойкоМеста» не может принимать NULL значения, не может принимать отрицательные значения.

Таблица «ТипыВахт»:

- атрибут «Название» является первичным ключем, не может принимать NULL значения;
- атрибут «Ставка» не может принимать NULL значения, не может принимать отрицательные значения.

Таблица «Должности»:

- атрибут «Название» является первичным ключем, не может принимать NULL значения;
- атрибут «Оклад» не может принимать NULL значения, не может принимать отрицательные значения.

2.3.2 Разработка триггеров

Для таблицы «Блоки»:

- проверяет на уникальность множество (Этаж, Крыло).

Для таблицы «заявки»:

- если статус заявки «Выполнена» или «Отклонена», не дает изменять значения, так как эти заявки уже логически завершены.

Для таблицы «Вахты»

- дает назначить из работников только того, кто имеет должность «Вахтер».

Для таблицы «Дежурство:

- дает назначить только работника с должностью «Уборщик».

Для таблицы «Комнаты»:

- триггер для каскадного удаления инвентаря и проживающих, а также их заявок.

2.3.3 Разработка процедур

Для работы с БД из приложения необходимо разработать следующие процедуры:

- процедуры добавления/обновления/удаления вахт;
- процедуры добавления/обновления/удаления инвентаря;
- процедуры добавления/изменения статуса заявок;
- процедура вычисления стоимости всего инвентаря.

2.3.4 Разработка представлений

Для отображение информации в таблицах клиентского приложения будут разработаны следующие представления:

- представление для вывода информации о хранящимся в общежитии инвентаре содержащее всю информацию о инвентаре с дополнительной колонкой, которая отображает общее количество каждого наименования инвентаря;
- представление для вывода информации о заявках, которое помимо информации о самих заявках выводит дополнительную информацию о составителе;
- представление для вывода информации о вахтах, которое дополнительно выводит информацию о работниках, закрепленных за вахтами.

3 Разработка клиентской части ИС

3.1 Разработка прототипа интерфейса пользователя

Для разработки пользовательского интерфейса клиентского приложения для начала необходимо разработать прототип интерфейса, по которому в дальнейшем будет реализован реальный интерфейс приложения. Прототип интерфейса будет сделан в графическом редакторе «Figma».

Клиентская часть разрабатывается для администратора общежития и содержит пользовательские интерфейсы для операций над вахтами, инвентарем и заявками.

После запуска приложения в него необходимо войти с помощью пароля администратора, прототип окна входа изображен на рисунке 23. В случае ввода несоответствующего пароля пользователю будет выведено окно с сообщением об ошибке, прототип которого представлен на рисунке 24.

После успешного входа пользователю будет показано главное окно, которое содержит вкладки для управления данными. На каждой из вкладок, которые соответствуют обрабатываемым ими сущностям, в виде меню располагаются доступные действия, которые можно провести над текущей сущностью (зависит от текущей выбранной вкладки). Прототип главного окна представлен на рисунке 25.

Нажатие на пункты меню вызывает диалоги, соответствующие действию пункта меню. Прототипы для типовых диалогов добавления, обновления и удаления записей, а также вывода результатов процедур представлены на рисунках 26-29. В случае каких-либо ошибок произошедших во время действий с данными пользователю будет также показано окно с сообщением об ошибке, прототип которого аналогичен с прототипом окна ошибки входа.

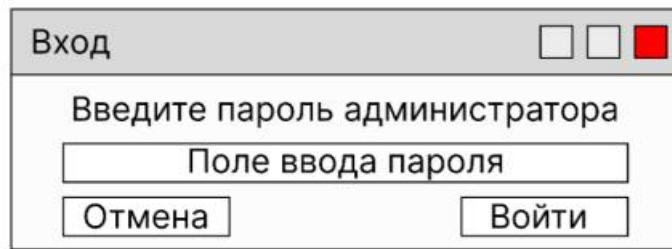


Рисунок 23 – Прототип окна входа

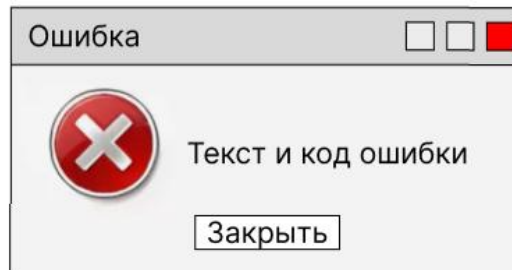


Рисунок 24 – Прототип сообщения об ошибке

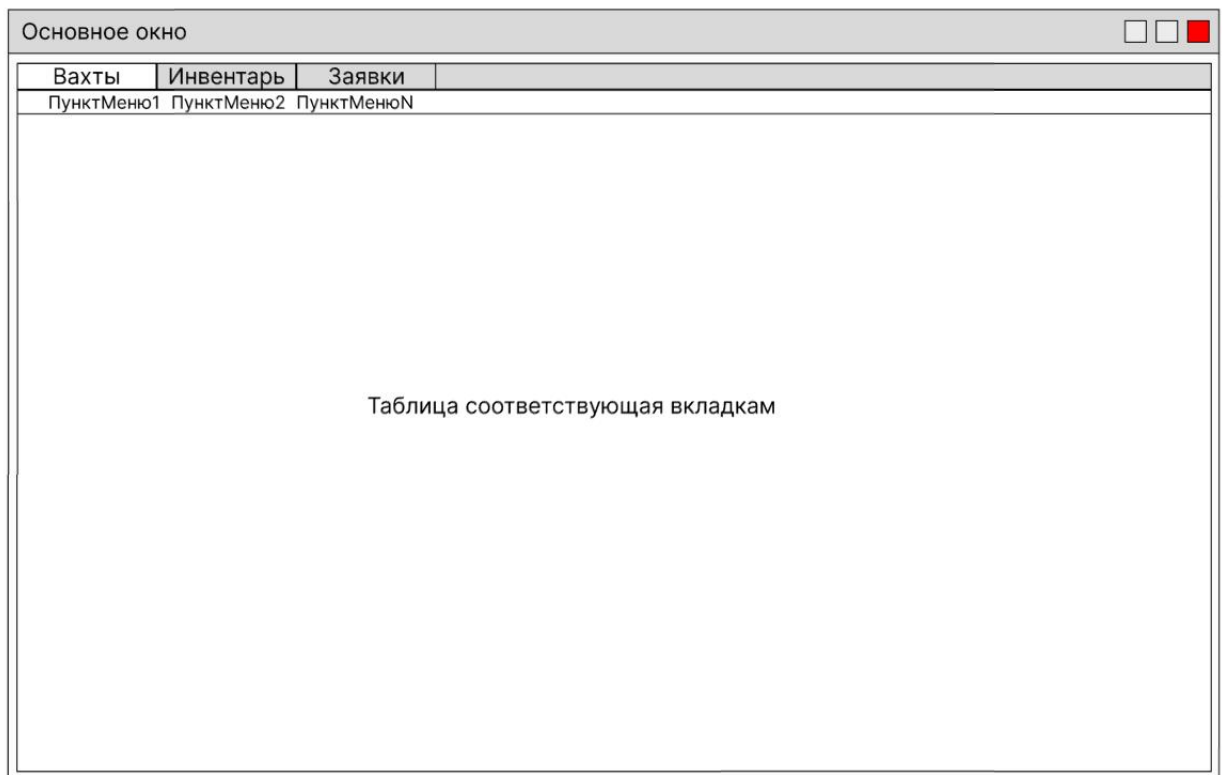


Рисунок 25 – Прототип основного окна

Добавление

Атрибут 1

Атрибут 2

...

Атрибут N

Отмена Добавить

Поля ввода

Рисунок 26 – Прототип окна добавления записи

Обновление

Атрибут 1

Атрибут 2

...

Атрибут N

Отмена Обновить

Поля ввода

Рисунок 27 – Прототип окна обновления записи

Удаление

Атрибут 1

Атрибут 2

...

Атрибут N

Отмена Удалить

Рисунок 28 – Прототип окна удаления записи

Результат

Результат работы процедуры в текстовом виде

Рисунок 29 – Прототип окна результата выполнения процедур

3.2 Реализация интерфейса пользователя

Реализация интерфейса пользователя будет сделана с помощью конструктора графических форм из интегрированной среды разработки Visual Studio для windows form .NET приложений.

Реализация окна входа (Рисунок 30).

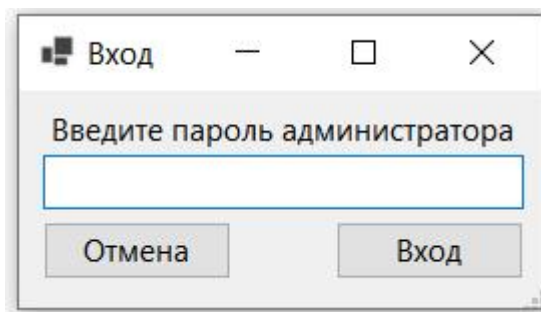


Рисунок 30 – Окно входа

Реализация окна сообщения об ошибке (Рисунок 31).

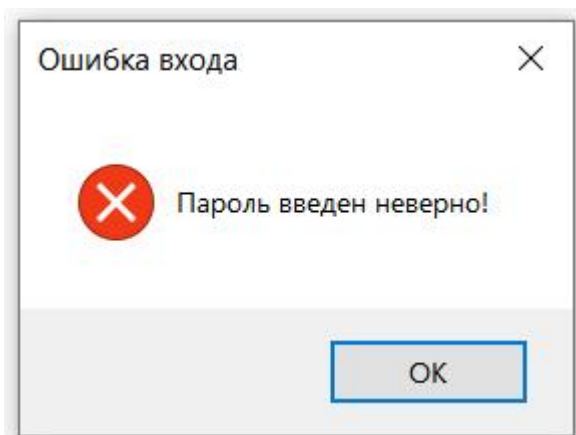


Рисунок 31 – Ошибка входа

Реализация основного окна на примере вкладки «Вахты» (Рисунок 32).

	Номер	Тип	ДатаНачала	Длительность	Вахтер	Фино
▶	1	Дневная	11.05.2024 12:00	12	1	Никулина Алл...
	3	Ночная	11.05.2024	12	1	Никулина Алл...
	4	Дневная в пра...	27.05.2024 10:46	4	1	Никулина Алл...

Рисунок 32 – Основное окно

Раздел «Вахты» позволяет вызвать 3 диалоговых окна для операций добавления, изменения и удаления вахт (Рисунки 33-35).

Добавить вахту

Тип: Дневная

Дата захода: 27 мая 2024 г.

Длительность в часах: 0

Вахтер: 1 Никулина Алла Ивановна

Отмена Добавить

Рисунок 33 – Диалог добавления вахты

Номер	1
Тип	Дневная
Дата захода	11 мая 2024 г.
Длительность в часах	12
Вахтер	1 Никулина Алла Ивановна

Отмена Обновить

Рисунок 34 – Диалог обновления вахты

Номер	1
Тип	Дневная
Дата захода	11.05.2024 12:00:00
Длительность	12
Вахтер	1 Никулина Алла Ивановна

Отмена Удалить

Рисунок 35 – Диалог удаления вахты

Раздел «Инвентарь» также позволяет вызвать 3 диалоговых окна для операций добавления, изменения и удаления записей об инвентаре, а также имеется возможность вывести общую стоимость всего инвентаря(Рисунки 36-39).

Наименование	
Дата поставки	27 мая 2024 г.
Закупочная стоимость	0
Номер комнаты	421

Отмена Добавить

Рисунок 36 – Диалог добавления инвентаря

Номер	2
Наименование	Стул "Комфорт"
Дата поставки	23 апреля 2022 г.
Закупочная стоимость	900
Номер комнаты	476

Отмена Обновить

Рисунок 37 – Диалог обновления инвентаря

Номер	2
Наименование	Стул "Комфорт"
Дата поставки	23.04.2022 0:00:00
Закупочная стоимость	900
Комната	476

Отмена Удалить

Рисунок 38 – Диалог удаления инвентаря

Общая стоимость всего инвентаря : 97170 руб.

Рисунок 39 – Диалог отчета

Раздел «Заявки» позволяет вызвать 2 диалоговых окна для операций добавления новой заявки и изменения статуса заявки (Рисунки 40-41)

Добавить заявку

Тема

Основной текст

Составитель 2

Отмена Добавить

Рисунок 40 – Диалог добавления заявки

Изменить статус

3

Здоровье

Статус:

Отклонена

Дата составления: 01.01.1900 0:00:00

Текст

Требую 8ми часовой сом

Составитель: Сивер Илья Иванович

Подтвердить

Рисунок 41 – Диалог обновления статуса заявки

4 Тестирование основных функций приложения

Будет протестирована функция входа, добавления заявок, а также функция изменения их статуса по принципу «черного ящика».

Правила для входа:

- при успешном вводе пароля открывается основное окно;
- при неверном вводе пользователю будет выведена ошибка входа а приложение будет закрыто.

Ввод неправильного пароля (Рисунок 42) и вывод ошибки (Рисунок 43).

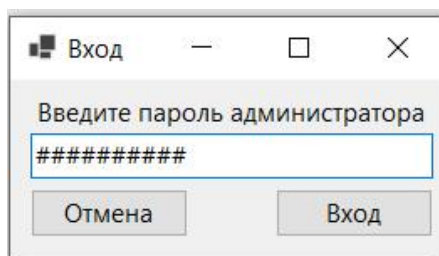


Рисунок 42 – Ввод пароля

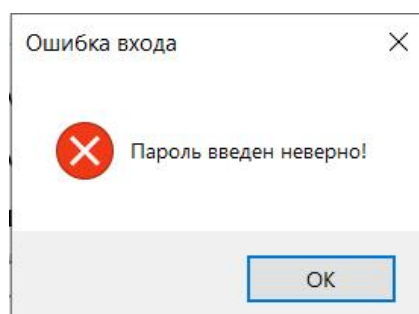
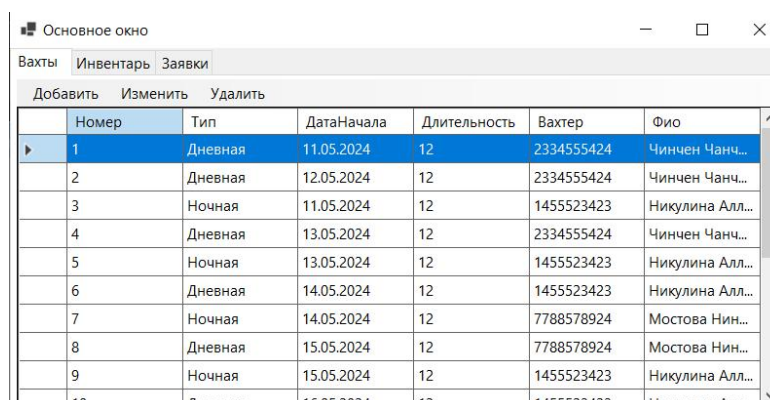


Рисунок 43 – Ошибка входа

Ввод правильного пароля (тестовый пароль – «linkedList») и открытие основного окна (рисунок 44).



Номер	Тип	ДатаНачала	Длительность	Вахтер	Фιο
1	Дневная	11.05.2024	12	2334555424	Чинчен Чанч...
2	Дневная	12.05.2024	12	2334555424	Чинчен Чанч...
3	Ночная	11.05.2024	12	1455523423	Никулина Алл...
4	Дневная	13.05.2024	12	2334555424	Чинчен Чанч...
5	Ночная	13.05.2024	12	1455523423	Никулина Алл...
6	Дневная	14.05.2024	12	1455523423	Никулина Алл...
7	Ночная	14.05.2024	12	7788578924	Мостова Нин...
8	Дневная	15.05.2024	12	7788578924	Мостова Нин...
9	Ночная	15.05.2024	12	1455523423	Никулина Алл...
10	Дневная	16.05.2024	12	1455523423	Никулина Алл...

Рисунок 44 – Вывод основного окна

Функция ввода пароля работает правильно.

Правила для добавления заявок

- один проживающий может оставлять любое количество заявок на любые темы и с любым текстом.

- статус заявки после добавления всегда «Открыта»;

- номер заявки уникален и генерируется автоматически;

Добавление нескольких заявок с одинаковой темой, текстом и составителем (Рисунок 42).

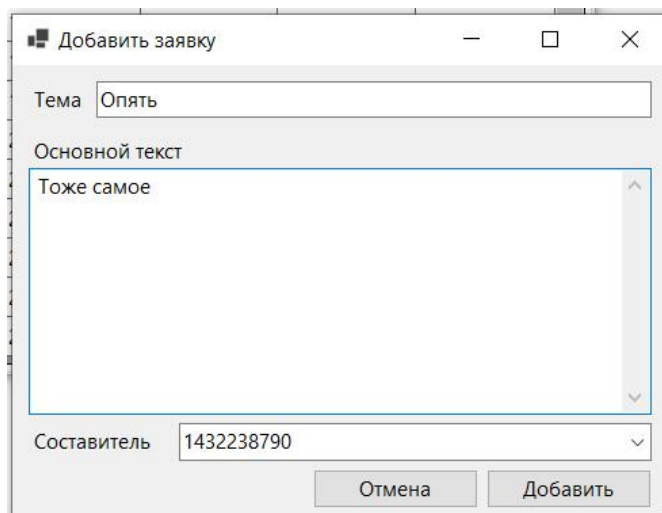


Рисунок 45 – Параметры добавляемых заявок

В итоге создаются 2 заявки с разным номером, но одинаковой информацией и составителем, а также статусом «Открыта».

16	Опять	Открыта	17.06.2024	Тоже самое	Конев Кири...	1432238790
17	Опять	Открыта	17.06.2024	Тоже самое	Конев Кири...	1432238790

Рисунок 46 – Итог создания заявок

Вывод: функция добавления заявок работает в соответствии с правилами.

Правила для обновления заявок:

- заявки нельзя удалить, так как они должны храниться в качестве архива в статусах «Закрыта» или «Отклонена»;

- в заявках имеющих статус «Открыта» или «Выполняется» можно

изменять только статус а текст, тему и составителя менять нельзя;

– Если заявка уже имеет статус «Закрыта» или «Отклонена», изменить статус уже нельзя, так как архивные данные изменению не подлежат.

Тестирование изменения статуса с «Открыта» на «Выполняется» (Рисунки 44-56).

Вахты	Инвентарь	Заявки					
Добавить Обновить статус							
	Номер	Тема	Статус	ДатаСоздани	Текст	Составитель	Код
▶	1	Предложен...	Открыта	01.01.2024	Как насчет ...	Конев Кири...	1432238790

Рисунок 47 – Исходная заявка

Изменить статус

1
Предложение
Статус:

Открыта ▼

Открыта 01.2024 0:00:00
Выполняется Кирилл Остапович
Отклонена
Выполнена

как насчет того чтобы ставить в комнаты сабы на 150 ватт?

Отмена Подтвердить

Рисунок 48 – Изменение статуса

Вахты	Инвентарь	Заявки					
Добавить Обновить статус							
	Номер	Тема	Статус	ДатаСоздани	Текст	Составитель	Код
▶	1	Предложен...	Выполняет...	01.01.2024	Как насчет ...	Конев Кири...	1432238790

Рисунок 49 – Итог

Тестирование изменения статуса с «Закрыта» на «Выполняется» (Рисунки 47-49).

Вахты	Инвентарь	Заявки					
Добавить Обновить статус							
	Номер	Тема	Статус	ДатаСоздани	Текст	Составитель	Код
▶	1	Предложен...	Отклонена	01.01.2024	Как насчет ...	Конев Кири...	1432238790

Рисунок 50 – Исходная заявка

Рисунок 51 – Обновление статуса

Вахты	Инвентарь	Заявки					
Добавить Обновить статус							
	Номер	Тема	Статус	ДатаСоздани	Текст	Составитель	Код
▶	1	Предложен...	Отклонена	01.01.2024	Как насчет ...	Конев Кири...	1432238790

Рисунок 52 – Итог

В первом случае заявка изменила статус, а во втором случае статус не поменялся. Вывод: функция обновления статуса заявок работает в соответствии с правилами.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была рассмотрена предметная область «Информационная система общежития» и исследована актуальность автоматизации.

По итогу была разработана модель данных с ограничениями, соответствующими предметной области, разработана БД на основе этой модели а также создано клиентское приложение для удаленной работы с данными.

Клиентское приложение успешно прошло тестирование на корректность работы своей работы.

При выполнении данной курсовой работы была изучена разработка правил, умолчаний, триггеров, курсоров и пользовательских типов на SQL. Также были изучены принципы разработки оконных приложений, а также манипулирование данными из БД с помощью фреймворка .NET 8 на языке программирования C#.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Базы данных. Программирование на SQL : учебник / Н.Н. Гринченко, Н.И. Хизриева. — Москва: КУРС. — 1 файл.pdf: 240 с. — Электронная копия печатной версии.
2. Базы данных. Разработка клиентских приложений на платформе .Net: учебник / Н.Н. Гринченко, А.Ю. Громов, А.В. Благодаров. — Москва: КУРС, 2023. — 1 файл.pdf: 288 с. — Электронная копия печатной версии.
3. Программирование клиентских приложений на языке C#: методические указания к лабораторным работам/ Рязан. гос. радиотехн. ун-т; сост.: Н.Н. Гринченко, Н.И. Хизриева, С.Н. Баранова. – Рязань, 2020. – 36 с.
4. "C# Coding. (n.d.). Выгрузка данных из SQL в Excel. Retrieved 20.05.2024, <https://csharpcoding.org/vygruzka-dannyx-iz-sql-v-excel/>"
- 5 Разработка клиентского приложения: методические указания к курсовому проектированию/ Рязан. гос. радиотехн. ун-т; сост.: Н.Н. Гринченко, Н.И. Хизриева, С.Н. Баранова. – Рязань, 2020. – 24 с.

ПРИЛОЖЕНИЕ А: сценарий создания объектов БД

```
use master
go
use CW
go
BEGIN TRANSACTION addTables
--Пользовательский тип для типа блока--
CREATE TYPE blockType FROM VARCHAR(127)
GO
CREATE RULE blockTypeRule AS
@type = 'Мужской' OR
@type = 'Женский' OR
@type = 'Общий' OR
@type = 'Преподавательский' OR
@type = 'Семейный' OR
@type = 'Служебный'
GO
EXEC sp_bindRule blockTypeRule, blockType
GO

--Пользовательский тип для статуса заявок--
CREATE TYPE statusType FROM VARCHAR(127)
GO
CREATE RULE statusRule AS
@status = 'Открыта' OR
@status = 'Выполняется' OR
@status = 'Отклонена' OR
@status = 'Выполнена'
GO
EXEC sp_bindRule statusRule, statusType
GO
CREATE DEFAULT issueStatusDefault AS
'Открыта'
GO
EXEC sp_bindefault issueStatusDefault, statusType
GO

CREATE TABLE ТипыПроживающих (
    Название varchar(127) PRIMARY KEY NOT NULL,
    Тариф int NOT NULL
)
GO
CREATE TABLE ТипыКомнат (
    Название varchar(127) PRIMARY KEY NOT NULL,
    КойкоМеста int NOT NULL
)
GO
CREATE TABLE Должности (
    Название varchar(127) PRIMARY KEY NOT NULL,
    Оклад int NOT NULL
)
GO
CREATE TABLE ТипыВахт (
    Название varchar(127) PRIMARY KEY NOT NULL,
    Ставка int NOT NULL
)
GO
CREATE TABLE Работники (
    Код bigint PRIMARY KEY NOT NULL,
    ФИО varchar(127) NOT NULL,
    Телефон varchar(127) NOT NULL,
```

```

        Должность varchar(127) NOT NULL,
        FOREIGN KEY (Должность) REFERENCES Должности
    )
GO
CREATE TABLE Блоки (
    Номер int PRIMARY KEY NOT NULL,
    Этаж int NOT NULL,
    Крыло int NOT NULL,
    Тип blockType NOT NULL
)
GO
CREATE TABLE Комнаты (
    Номер int PRIMARY KEY NOT NULL,
    Тип varchar(127) NOT NULL,
    Блок int NOT NULL,
    FOREIGN KEY (Тип) REFERENCES ТипыКомнат,
    FOREIGN KEY (Блок) REFERENCES Блоки
)
GO
CREATE TABLE Проживающие (
    Код bigint PRIMARY KEY NOT NULL,
    ФИО varchar(127) NOT NULL,
    Телефон varchar(127) NOT NULL,
    Пол varchar(3) NOT NULL,
    ДатаРождения date NOT NULL,
    Тип varchar(127) NOT NULL,
    Комната int NOT NULL,
    FOREIGN KEY (Тип) REFERENCES ТипыПроживающих,
    FOREIGN KEY (Комната) REFERENCES Комнаты
)
GO
CREATE TABLE Заявки (
    Номер int PRIMARY KEY NOT NULL IDENTITY(1,1),
    Тема varchar(127) NOT NULL,
    Статус statusType NOT NULL,
    ДатаСоздания date NOT NULL default GETDATE(),
    Текст varchar(MAX),
    Составитель bigint NOT NULL,
    FOREIGN KEY (Составитель) REFERENCES Проживающие
)
GO
CREATE TABLE Инвентарь (
    Код int PRIMARY KEY NOT NULL IDENTITY(1,1),
    Название varchar(127) NOT NULL,
    ДатаПоставки date NOT NULL default GETDATE(),
    Стоимость int NOT NULL,
    Комната int NOT NULL,
    FOREIGN KEY (Комната) REFERENCES Комнаты,
)
GO
CREATE TABLE Вахты (
    Номер int PRIMARY KEY NOT NULL IDENTITY(1,1),
    Тип varchar(127) NOT NULL,
    FOREIGN KEY (Тип) REFERENCES ТипыВахт,
    ДатаНачала date NOT NULL default GETDATE(),
    Длительность int NOT NULL,
    Вахтер bigint,
    FOREIGN KEY (Вахтер) REFERENCES Работники
)
GO
CREATE TABLE Дежурство(
    Блок int PRIMARY KEY NOT NULL,

```

```

        Дежурный bigint NOT NULL,
        FOREIGN KEY(Блок) REFERENCES Блоки,
        FOREIGN KEY(Дежурный) REFERENCES Работники
    )
GO
COMMIT TRANSACTION addTables

USE CW
GO
--Представление для вывода заявок в приложении--
CREATE VIEW issuesView AS
SELECT З.Номер,З.Тема,З.Статус,З.ДатаСоздания,З.Текст,п.ФИО as Составитель,
З.Составитель AS Код
FROM Заявки З JOIN Проживающие П ON З.Составитель = П.Код
GO
--Представление для вывода инвентаря в приложении--
CREATE VIEW inventView AS
SELECT И.*, К.Блок as Блок, Всего =
    (SELECT count(*)
     FROM Инвентарь В
     WHERE В.Название = И.Название)
FROM Инвентарь И JOIN Комнаты К ON Комната = К.Номер
GO
--Представление для вывода вахт в приложении--
CREATE VIEW vahtsView AS
SELECT В.*, Р.ФИО AS Фио
FROM Вахты В JOIN Работники Р ON В.Вахтер = Р.Код
GO

--Правило для номера телефона--
CREATE RULE telRule AS
(@tel LIKE '+7([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]') OR
(@tel LIKE '8([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]')
GO
EXEC sp_bindrule telRule, 'Проживающие.Телефон'
GO
EXEC sp_bindrule telRule, 'Работники.Телефон'
GO

--Правило для ФИО--
CREATE RULE fioRule AS
@fio LIKE '[А-Я]% [А-Я]% [А-Я]%'
GO
EXEC sp_bindrule fioRule, 'Проживающие.ФИО'
GO
EXEC sp_bindrule fioRule, 'Работники.ФИО'
GO

--Правило для значений которые должны быть больше нуля--
CREATE RULE notNegativeRule AS
@value >= 0
GO
EXEC sp_bindRule notNegativeRule, 'Должности.Оклад'
GO
EXEC sp_bindRule notNegativeRule, 'ТипыВахт.Ставка'
GO
EXEC sp_bindRule notNegativeRule, 'Вахты.Длительность'
GO
EXEC sp_bindRule notNegativeRule, 'ТипыПроживающих.Тариф'
GO
EXEC sp_bindRule notNegativeRule, 'ТипыКомнат.КойкоМеста'

```

```

GO
EXEC sp_bindRule notNegativeRule, 'Инвентарь.Стоимость'
GO

--Правило для номера паспорта--
CREATE RULE passportRule AS
    @value >= 1000000000 AND @value <= 9999999999
GO
EXEC sp_bindRule passportRule, 'Работники.Код'
GO
EXEC sp_bindRule passportRule, 'Проживающие.Код'
GO

--Триггер каскадного удаления проживающих в удаляемой комнате, их заявок и инвентаря--
CREATE TRIGGER cascadeDeleteRoom ON Комнаты
INSTEAD OF DELETE AS
BEGIN
    DECLARE @room int

    DECLARE roomCur CURSOR FOR
        SELECT deleted.Номер
        FROM deleted

    OPEN roomCur
    FETCH NEXT FROM roomCur INTO @room
    WHILE @@FETCH_STATUS = 0
    BEGIN

        DECLARE @student int

        DECLARE studentCur CURSOR FOR
            SELECT Проживающие.Код
            FROM Проживающие
            WHERE Проживающие.Комната = @room

        OPEN studentCur
        FETCH NEXT FROM studentCur INTO @student
        WHILE @@FETCH_STATUS = 0
        BEGIN
            DELETE FROM Заявки
            WHERE Заявки.Составитель = @student
            FETCH NEXT FROM studentCur INTO @student
        END
        CLOSE studentCur
        DEALLOCATE studentCur

        DELETE FROM Проживающие
        WHERE Проживающие.Комната = @room

        DELETE FROM Инвентарь
        WHERE Инвентарь.Комната = @room

        DELETE FROM Комнаты
        WHERE Комнаты.Номер = @room

        FETCH NEXT FROM roomCur INTO @room
    END
    CLOSE roomCur
    DEALLOCATE roomCur
END
GO

```

```

--Триггер для проверки на уникальность значений этажа и крыла добавляемых блоков--
CREATE TRIGGER insertBlock ON Блоки
INSTEAD OF INSERT AS
BEGIN
    DECLARE @block int
    DECLARE @level int
    DECLARE @wing int
    DECLARE @type varchar(127)
    DECLARE blockCur CURSOR FOR
        SELECT inserted.Номер, inserted.Этаж, inserted.Крыло, inserted.Тип
        FROM inserted
    OPEN blockCur
    FETCH NEXT FROM blockCur INTO @block, @level, @wing, @type
    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF NOT EXISTS(SELECT * FROM Блоки WHERE (Блоки.Номер = @block) OR
(Блоки.Этаж = @level and Блоки.Крыло = @wing))
        BEGIN
            INSERT INTO Блоки VALUES (@block,@level,@wing,@type)
        END
        FETCH NEXT FROM blockCur INTO @block, @level, @wing, @type
    END
    CLOSE blockCur
    DEALLOCATE blockCur
END
GO

--Триггер на изменение статуса заявки--
CREATE TRIGGER updateIssueT ON Заявки
INSTEAD OF UPDATE AS
BEGIN
    DECLARE @issue int
    DECLARE @status varchar(127)

    DECLARE issueCur CURSOR FOR
        SELECT inserted.Номер, inserted.Статус
        FROM inserted

    OPEN issueCur
    FETCH NEXT FROM issueCur INTO @issue, @status
    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF EXISTS(SELECT * FROM Заявки WHERE Заявки.Номер = @issue and
(Заявки.Статус <> 'Отклонена' OR Заявки.Статус <> 'Выполнена'))
        BEGIN
            UPDATE Заявки
            SET Заявки.Статус = @status
            WHERE Заявки.Номер = @issue
        END
        FETCH NEXT FROM issueCur INTO @issue, @status
    END
    CLOSE issueCur
    DEALLOCATE issueCur
END
GO

--Триггер на добавление вахты, в которой может дежурить только работник с должностью
вахтера--
CREATE TRIGGER addVaht On Вахты
INSTEAD OF INSERT AS
BEGIN

```

```

DECLARE @vaht int
DECLARE @type varchar(127)
DECLARE @date datetime
DECLARE @duration int
DECLARE @worker bigint

DECLARE vahtCur CURSOR FOR
    SELECT *
    FROM inserted

OPEN vahtCur
    FETCH NEXT FROM vahtCur INTO @vaht,@type, @date,@duration, @worker
    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF EXISTS(SELECT * FROM Работники WHERE Работники.Код = @worker AND
Работники.Должность = 'Вахтер')
        BEGIN
            INSERT INTO Вахты VALUES
                (@type, @date,@duration, @worker)
        END
        FETCH NEXT FROM vahtCur INTO @vaht,@type, @date,@duration, @worker
    END
CLOSE vahtCur
DEALLOCATE vahtCur

END
Go

--Триггер на добавление дежурного в блок--
CREATE TRIGGER addDuty ON Дежурство
INSTEAD OF INSERT AS
BEGIN
    DECLARE @worker bigint
    DECLARE @block int
    DECLARE dutyCur CURSOR FOR
        SELECT inserted.Блок, inserted.Дежурный
        FROM inserted

    OPEN dutyCur
        FETCH NEXT FROM dutyCur INTO @block, @worker
        WHILE @@FETCH_STATUS = 0
        BEGIN
            IF EXISTS(SELECT * FROM Работники WHERE Работники.Код = @worker AND
Работники.Должность = 'Уборщик')
            BEGIN
                INSERT INTO Дежурство VALUES
                    (@block, @worker)
            END
            FETCH NEXT FROM dutyCur INTO @block, @worker
        END
        CLOSE dutyCur
        DEALLOCATE dutyCur

    END
GO

--Процедура добавления новой вахты--
CREATE PROC insertVaht (@type varchar(127), @date datetime, @duration int, @worker
bigint)
AS
IF EXISTS(SELECT * FROM Работники WHERE Работники.Код = @worker AND Работники.Должность
= 'Вахтер')
BEGIN
    INSERT INTO Вахты (Тип, ДатаНачала, Длительность, Вахтер) VALUES

```

```

        (@type, @date, @duration, @worker)
        RETURN 0
END
ELSE
BEGIN
RETURN 1
END
GO

--Процедура обновления вахты--
CREATE PROC updateVaht (@num int, @type varchar(127), @date datetime, @duration int,
@worker bigint)
AS
IF EXISTS(SELECT * FROM Работники WHERE Работники.Код = @worker and Должность = 'Вахтер')
BEGIN
    UPDATE Вахты
    SET Тип = @type, ДатаНачала = @date, Длительность = @duration, Вахтер = @worker
    WHERE Вахты.Номер = @num
    RETURN 0
END
ELSE
BEGIN
    RETURN 1
END
GO

--Процедура Добавления заявки--
CREATE PROC insertIssue (@title varchar(127), @text varchar(max), @author bigint)
AS
IF EXISTS(SELECT * FROM Проживающие WHERE Проживающие.Код = @author)
BEGIN
    INSERT INTO Заявки (Заявки.Тема, Заявки.Текст, Заявки.Составитель) VALUES
    (@title, @text, @author )
    RETURN 0
END
ELSE
BEGIN
    RETURN 1
END
GO

--Процедура обновления статуса заявки--
CREATE PROC updateIssue (@num int, @status statusType)
AS
IF EXISTS(SELECT * FROM Заявки WHERE Заявки.Номер = @num and (Заявки.Статус = 'Открыта'
OR Заявки.Статус = 'Выполняется'))
    BEGIN
        UPDATE Заявки
        SET Заявки.Статус = @status
        WHERE Заявки.Номер = @num
        RETURN 0
    END
ELSE
BEGIN
    RETURN 1
END
Go

--Процедура добавления нового инвентаря--
CREATE PROC insertInvent (@name varchar(127), @date datetime, @cost int, @room int)
AS
IF EXISTS(SELECT * FROM Комнаты WHERE Комнаты.Номер = @room)

```



```

        BEGIN
        INSERT INTO Инвентарь VALUES
        (@name , @date , @cost , @room )
        RETURN 0
        END
ELSE
BEGIN
        RETURN 1
END
GO

--Процедура удаления инвентаря--
CREATE PROC deleteInvent (@code int)
AS
IF EXISTS(SELECT * FROM Инвентарь WHERE Инвентарь.Код = @code)
BEGIN
        DELETE Инвентарь
        WHERE Инвентарь.Код = @code
        RETURN 0
END
ELSE
BEGIN
        RETURN 1
END
GO

--Процедура обновления инвентаря--
CREATE PROC updateInvent (@code int, @name varchar(127), @date datetime, @cost int,
@room int)
AS
IF EXISTS(SELECT * FROM Инвентарь WHERE Инвентарь.Код = @code)
BEGIN
        UPDATE Инвентарь
        SET Название = @name, ДатаПоставки = @date, Стоимость = @cost, Комната = @room
        WHERE Инвентарь.Код = @code
        RETURN 0
END
ELSE
BEGIN
        RETURN 1
END
GO

--Процедура удаления вахты--
CREATE PROC deleteVaht (@num int)
AS
IF EXISTS(SELECT * FROM Вахты WHERE Вахты.Номер = @num)
BEGIN
        DELETE Вахты
        WHERE Вахты.Номер = @num
        RETURN 0
END
ELSE
BEGIN
        RETURN 1
END
GO

CREATE PROC computeAllCost (@cost int output)
AS
SELECT @cost = sum(Стоимость)
FROM Инвентарь

```

GO

ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД

```
use master
GO
use CW
GO
BEGIN TRAN addRows
```

```
INSERT INTO ТипыКомнат VALUES
```

```
( 'Царская' , 4 ),
( 'Сычевальня' , 1 ),
( 'Двухместная' , 2 ),
( 'Трехместная' , 3 ),
( 'Служебная' , 0 )
```

```
Go
```

```
INSERT INTO Блоки VALUES
```

```
( 12 , 1 , 2 , 'Мужской' ),
( 15 , 1 , 5 , 'Женский' ),
( 38 , 3 , 8 , 'Мужской' ),
( 46 , 4 , 6 , 'Женский' ),
( 39 , 3 , 9 , 'Мужской' ),
( 37 , 3 , 7 , 'Служебный' ),
( 41 , 4 , 1 , 'Служебный' ),
( 47 , 4 , 7 , 'Мужской' ),
( 42 , 4 , 2 , 'Женский' )
```

```
Go
```

```
INSERT INTO Комнаты VALUES
```

```
( 121 , 'Трехместная' , 12 ),
( 122 , 'Двухместная' , 12 ),
( 127 , 'Двухместная' , 12 ),
( 381 , 'Трехместная' , 38 ),
( 385 , 'Трехместная' , 38 ),
( 421 , 'Двухместная' , 42 ),
( 422 , 'Двухместная' , 42 ),
( 423 , 'Трехместная' , 42 ),
( 475 , 'Трехместная' , 47 ),
( 476 , 'Двухместная' , 47 ),
( 477 , 'Двухместная' , 47 ),
( 478 , 'Трехместная' , 47 )
```

```
Go
```

```
INSERT INTO Инвентарь VALUES
```

```
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 476 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 476 ),
( 'Стол "Ученический"' , '23.04.2022' , 2200 , 476 ),
( 'Шкаф 2200x2000x400' , '23.04.2022' , 12000 , 476 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 477 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 477 ),
( 'Стол "Ученический"' , '23.04.2022' , 2200 , 477 ),
( 'Шкаф 2200x2000x400' , '23.04.2022' , 12000 , 477 ),
( 'Холодильник "Полюс"' , '01.09.1970' , 120 , 476 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 122 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 122 ),
( 'Стол "Ученический"' , '23.04.2022' , 2200 , 122 ),
( 'Шкаф 2200x2000x400' , '23.04.2022' , 12000 , 122 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 127 ),
( 'Стул "Комфорт"' , '23.04.2022' , 900 , 127 ),
( 'Стол "Ученический"' , '23.04.2022' , 2200 , 127 ),
( 'Шкаф 2200x2000x400' , '23.04.2022' , 12000 , 127 ),
( 'Холодильник "Полюс"' , '01.09.1970' , 120 , 127 )
```

```

Go
INSERT INTO ТипыПроживающих VALUES
('ССУЗ', 800),
('ВУЗ', 3500),
('ВУЗ Льготный', 3500),
('ССУЗ Льготный', 400),
('РГУ Очная', 2000),
('РГУ Очная Льготный', 1000),
('РГРТУ Очная', 1000),
('РГРТУ Заочная', 1000),
('РГРТУ Очная Льготный', 0)
Go
INSERT INTO Проживающие VALUES
--(2342345676, 'Сивер Илья Иванович', '8(923)345-14-23', 'муж', '25.11.2004', 'РГРТУ
Очная', 475),
(1432987567, 'Вареньев Максим Алексеевич', '8(961)555-53-53', 'муж', '01.01.2003', 'РГУ
Очная', 477),
(1432376564, 'Сидорова Ирина Васильевна', '8(923)362-75-57', 'жен', '05.02.2003', 'РГУ
Очная', 421),
(2342234643, 'Иванов Дмитрий Олегович', '8(961)124-89-98', 'муж', '10.02.2004', 'РГРТУ
Очная', 477),
(1432238790, 'Конев Кирилл Остапович', '8(980)876-25-25', 'муж', '15.02.2003', 'РГРТУ
Очная', 475),
(2342097654, 'Петрова Ольга Валерьевна', '8(923)323-50-10', 'жен', '20.02.2004', 'РГРТУ
Очная', 421),
(6744146734, 'Брыськин Ярослав Сергеевич', '8(980)998-05-00', 'муж', '25.03.2002', 'РГРТУ
Очная', 127),
(6744937856, 'Горянов Артем Сергеевич', '8(923)000-50-60', 'муж', '30.03.2002', 'РГРТУ
Очная', 127),
(2342070775, 'Масяев Андрей Александрович', '8(980)340-88-44', 'муж', '5.05.2004', 'РГРТУ
Очная', 122),
(6744887065, 'Козловская Дарья Владиславовна', '8(923)064-76-
34', 'жен', '10.05.2002', 'ССУЗ', 422),
(2342057345, 'Вы То Чен', '8(961)356-96-69', 'муж', '16.08.2004', 'РГРТУ Очная', 122)
Go
INSERT INTO Заявки VALUES
('Предложение', 'Открыта', '01.01.2024', 'Как насчет того чтобы ставить в комнаты сабы на
150 ватт?', 1432238790),
('Предложение', 'Открыта', '22.05.2024', 'Поставить кулеры в блоки', 1432238790),
('Предложение', 'Открыта', '02.01.2024', 'Закупить духовые шкафы', 1432238790),
('Ремонт', 'Открыта', '11.05.2023', 'Ремонт балконов', 6744146734),
('Ремонт', 'Открыта', '22.05.2024', 'Разбита плитка в 12 блоке', 6744146734),
('Ремонт', 'Открыта', '21.05.2024', 'В 15 блоке скрипит дверь', 6744146734),
('Ремонт', 'Открыта', '12.05.2023', 'Перегорела лампочка в 127 комнате', 6744146734),
('Жалоба', 'Открыта', '16.04.2024', 'Соседи сверху сверлят', 6744887065),
('Жалоба', 'Открыта', '18.03.2022', 'Алкаш в 477 комнате', 1432238790),
('Жалоба', 'Открыта', '22.05.2024', 'Крики ночью из 476 комнаты', 1432987567),
('Жалоба', 'Открыта', '24.12.2022', 'Когда конец сессии?', 2342057345),
('Жалоба', 'Открыта', '22.05.2024', 'Вахтерша не пустила в общагу', 1432238790),
('Жалоба', 'Открыта', '22.08.2024', 'Драка в 12 блоке', 1432238790),
('Жалоба', 'Открыта', '22.07.2023', 'Курыт на балконе', 1432238790),
('Жалоба', 'Открыта', '22.11.2024', 'А почему я не поеду в Египет?', 2342057345)
Go
INSERT INTO Должности VALUES
('Вахтер', 19000),
('Уборщик', 17900),
('Слесарь', 32000),
('Кладовщик', 26000)
Go
INSERT INTO Работники VALUES
(2334555424, 'Чинчен Чанчон Чочи', '8(961)222-11-00', 'Вахтер'),

```

```
(1455523423, 'Никулина Алла Ивановна', '8(955)452-53-22', 'Вахтер'),
(2134134123, 'Борисов Иван Ильич', '8(921)432-77-53', 'Слесарь'),
(3989456732, 'Сидоров Олег Абдулович', '8(920)666-99-69', 'Уборщик'),
(7788578924, 'Мостова Нина Юрьевна', '8(955)777-33-77', 'Вахтер'),
(2334976522, 'Пронов Никита Игоревич', '8(900)354-23-89', 'Слесарь'),
(2134245367, 'Лосева Татьяна Сергеевна', '8(655)877-87-88', 'Уборщик')
```

Go

```
INSERT INTO ТипыВахт VALUES
```

```
('Ночная', 2),
('Дневная', 1),
('Ночная в праздник', 3),
('Дневная в праздник', 2)
```

Go

```
INSERT INTO Вахты VALUES
```

```
('Дневная', '11.05.2024 12:00', 12, 2334555424),
('Дневная', '12.05.2024 12:00', 12, 2334555424),
('Ночная', '11.05.2024 00:00', 12, 1455523423),
('Дневная', '13.05.2024 12:00', 12, 2334555424),
('Ночная', '13.05.2024 00:00', 12, 1455523423),
('Дневная', '14.05.2024 12:00', 12, 1455523423),
('Ночная', '14.05.2024 00:00', 12, 7788578924),
('Дневная', '15.05.2024 12:00', 12, 7788578924),
('Ночная', '15.05.2024 00:00', 12, 1455523423),
('Дневная', '16.05.2024 12:00', 12, 1455523423),
('Ночная', '16.05.2024 00:00', 12, 7788578924),
('Дневная', '17.05.2024 12:00', 12, 7788578924),
('Ночная', '17.05.2024 00:00', 12, 1455523423),
('Дневная', '18.05.2024 12:00', 12, 1455523423),
('Ночная', '18.05.2024 00:00', 12, 7788578924),
('Дневная', '19.05.2024 12:00', 12, 7788578924),
('Ночная', '19.05.2024 00:00', 12, 1455523423),
('Дневная', '20.05.2024 12:00', 12, 1455523423),
('Ночная', '20.05.2024 00:00', 12, 2334555424)
```

Go

```
INSERT INTO Дежурство VALUES
```

```
(12, 3989456732),
(15, 3989456732),
(46, 3989456732),
(39, 3989456732),
(47, 2134245367),
(42, 2134245367)
```

```
COMMIT TRAN addRows
```

ПРИЛОЖЕНИЕ В:исходный код клиентского приложения

Файл Program.cs

```
using CourseWorkApp.View;
namespace CourseWorkApp
{
    internal static class Program
    {
        const long passwordHash = 73973732; // "linkedList"
        [STAThread]
        static void Main()
        {
            ApplicationConfiguration.Initialize();
            var a = "sdsd".ToCharArray().GetHashCode();
            loginForm loginForm = new loginForm();
            switch (loginForm.ShowDialog())
            {
                case DialogResult.OK:
                    if (PassHashFunc(loginForm.passTextBox.Text.ToString())==
passwordHash)
                        Application.Run(new MainForm());
                    else
                    {
                        MessageBox.Show("Пароль введен неверно!", "Ошибка входа",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    break;
                default:
                    break;
            }
        }
        static private long PassHashFunc(string arg)
        {
            long sum = 0;

            foreach (char c in arg)
            {
                sum += c * 256;
            }
            return ((sum * sum) >> (int)(sum % 21)) * 17;
        }
    }
}
```

Файл App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <connectionStrings>
        <add name="Config"
            providerName="System.Data.SqlClient"
            connectionString="Server = POZERPC;
Database = CW;
Integrated Security=true;
Encrypt = True;
TrustServerCertificate = true;" />
    </connectionStrings>
</configuration>
```

Файл MainForm.cs

```
using CourseWorkApp.View;
using Microsoft.Data.SqlClient;
using Microsoft.Identity.Client.NativeInterop;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CourseWorkApp
{
    public partial class MainForm : Form
    {
        SqlConnection conn = new SqlConnection();
        List<string> statusList = new List<string>();

        void ConnectDb()
        {
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings["Config"].ConnectionString;
            try
            {
                conn.Open();
            }
            catch (SqlException ex)
            {
                conn.Close();
                MessageBox.Show(ex.Message, "Ошибка подключения",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        public MainForm()
        {
            statusList.Add("Открыта");
            statusList.Add("Выполняется");
            statusList.Add("Отклонена");
            statusList.Add("Выполнена");
            InitializeComponent();
            ConnectDb();
            FillIssueDgv();
            FillInventDgv();
            FillVahtDgv();
        }

        void FillIssueDgv()
        {
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandText = "SELECT *" +
                "FROM issuesView";

            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataTable dataTable = new DataTable();
            issuesDgv.DataSource = dataTable;
            dataTable.Clear();
            adapter.Fill(dataTable);
        }

        void FillInventDgv()
        {
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandText = "SELECT *" +
```

```

        "FROM inventView";

        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        DataTable dataTable = new DataTable();
        inventDgv.DataSource = dataTable;
        dataTable.Clear();
        adapter.Fill(dataTable);
    }

    void FillVahtDgv()
    {
        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "SELECT *" +
            "FROM vahtsView";

        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        DataTable dataTable = new DataTable();
        vahtsDgv.DataSource = dataTable;
        dataTable.Clear();
        adapter.Fill(dataTable);
    }
    private void добавитьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        inventAdd inventAddDlg = new inventAdd();
        inventAddDlg.roomComboBox.DataSource = FillRoomsNumbers();

        switch (inventAddDlg.ShowDialog())
        {
            case DialogResult.OK:
            {
                SqlCommand cmd = new SqlCommand("insertInvent", conn);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@name",
inventAddDlg.nameTextBox.Text);
                cmd.Parameters.AddWithValue("@date",
inventAddDlg.dateTimePicker.Value);
                cmd.Parameters.AddWithValue("@cost",
inventAddDlg.costNumeric.Value);
                cmd.Parameters.AddWithValue("@room",
int.Parse(inventAddDlg.roomComboBox.SelectedItem.ToString()));
                cmd.ExecuteNonQuery();
                FillInventDgv();
                break;
            }
            case DialogResult.Cancel:
            {
                break;
            }
            default:
            {
                break;
            }
        }
    }

    List<string> FillRoomsNumbers()
    {
        List<string> dataList = new List<string>();

        SqlCommand addRoomsCmd = conn.CreateCommand();
        addRoomsCmd.CommandText = "SELECT Комнаты.Номер FROM Комнаты";
        var reader = addRoomsCmd.ExecuteReader();
        while (reader.Read())
        {
            dataList.Add(reader["Номер"].ToString());
        }
        reader.Close();
    }

```



```

        return dataList;
    }

    List<string> FillStudNumbers()
    {
        List<string> dataList = new List<string>();

        SqlCommand addStudCmd = conn.CreateCommand();
        addStudCmd.CommandText = "SELECT Проживающие.Код FROM Проживающие";
        var reader = addStudCmd.ExecuteReader();
        while (reader.Read())
        {
            dataList.Add(reader["Код"].ToString());
        }
        reader.Close();
        return dataList;
    }

    List<string> FillvahtWorkerNumbers()
    {
        List<string> dataList = new List<string>();

        SqlCommand addWorkersCmd = conn.CreateCommand();
        addWorkersCmd.CommandText = "SELECT Работники.Код, Работники.ФИО FROM
Работники WHERE Должность = 'Вахтер'";
        var reader = addWorkersCmd.ExecuteReader();
        while (reader.Read())
        {
            dataList.Add(reader["Код"].ToString() + " " + reader["ФИО"]);
        }
        reader.Close();
        return dataList;
    }

    List<string> FillVahtTypes()
    {
        List<string> dataList = new List<string>();

        SqlCommand addVahtTypesCmd = conn.CreateCommand();
        addVahtTypesCmd.CommandText = "SELECT ТипыВахт.Название FROM ТипыВахт";
        var reader = addVahtTypesCmd.ExecuteReader();
        while (reader.Read())
        {
            dataList.Add(reader["Название"].ToString());
        }
        reader.Close();
        return dataList;
    }

    private void обновитьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        DataGridViewRow selectedRow = inventDgv.SelectedRows[0];

        inventUpdate inventUpdateDlg = new inventUpdate();
        int code = (int)selectedRow.Cells[0].Value;
        string name = (string)selectedRow.Cells[1].Value;
        DateTime date = (DateTime)selectedRow.Cells[2].Value;
        int cost = (int)selectedRow.Cells[3].Value;
        int room = (int)selectedRow.Cells[4].Value;

        inventUpdateDlg.numLabel.Text = code.ToString();
        inventUpdateDlg.nameTextBox.Text = name;
        inventUpdateDlg.dateTimePicker.Value = date;
        inventUpdateDlg.costNumeric.Value = cost;
        inventUpdateDlg.roomComboBox.DataSource = FillRoomsNumbers();
        inventUpdateDlg.roomComboBox.SelectedItem = room.ToString();
        switch (inventUpdateDlg.ShowDialog())
        {
            case DialogResult.OK:

```

```

        {
            SqlCommand cmd = new SqlCommand("updateInvent", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@code", code);
            cmd.Parameters.AddWithValue("@name",
inventUpdatedDlg.nameTextBox.Text);
            cmd.Parameters.AddWithValue("@date",
inventUpdatedDlg.dateTimePicker.Value);
            cmd.Parameters.AddWithValue("@cost",
inventUpdatedDlg.costNumeric.Value);
            cmd.Parameters.AddWithValue("@room",
int.Parse(inventUpdatedDlg.roomComboBox.SelectedItem.ToString()));
            cmd.ExecuteNonQuery();
            FillInventDgv();
            break;
        }
        case DialogResult.Cancel:
        {
            break;
        }
        default:
        {
            break;
        }
    }
}

private void удалитьToolStripMenuItem1_Click(object sender, EventArgs e)
{
    inventRemove inventRemoveDlg = new inventRemove();

    DataGridViewRow selectedRow = inventDgv.SelectedRows[0];

    int code = (int)selectedRow.Cells[0].Value;
    string name = (string)selectedRow.Cells[1].Value;
    DateTime date = (DateTime)selectedRow.Cells[2].Value;
    int cost = (int)selectedRow.Cells[3].Value;
    int room = (int)selectedRow.Cells[4].Value;

    inventRemoveDlg.numLabel.Text = code.ToString();
    inventRemoveDlg.nameLabel.Text = name;
    inventRemoveDlg.dateLabel.Text = date.ToString();
    inventRemoveDlg.costLabel.Text = cost.ToString();
    inventRemoveDlg.roomLabel.Text = room.ToString();

    switch (inventRemoveDlg.ShowDialog())
    {
        case DialogResult.OK:
        {
            SqlCommand cmd = new SqlCommand("deleteInvent", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@code", code);
            cmd.ExecuteNonQuery();
            FillInventDgv();
            break;
        }
        case DialogResult.Cancel:
        {
            break;
        }
        default:
        {
            break;
        }
    }
}

private void добавитьToolStripMenuItem1_Click(object sender, EventArgs e)

```

```

{
    VahtAddDlg vahtAddDlg = new VahtAddDlg();
    vahtAddDlg.typeComboBox.DataSource = FillVahtTypes();
    vahtAddDlg.workerComboBox.DataSource = FillvahtWorkerNumbers();
    switch (vahtAddDlg.ShowDialog())
    {
        case DialogResult.OK:
        {
            SqlCommand cmd = new SqlCommand("insertVaht", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@type",
vahtAddDlg.typeComboBox.SelectedItem.ToString());
            cmd.Parameters.AddWithValue("@date",
vahtAddDlg.dateTimePicker.Value);
            cmd.Parameters.AddWithValue("@duration",
vahtAddDlg.durationNumeric.Value);

            cmd.Parameters.AddWithValue("@worker", long.Parse(vahtAddDlg.workerComboBox.SelectedItem.ToString().Split(" ")[0]));
            cmd.ExecuteNonQuery();
            FillVahtDgv();
            break;
        }
        case DialogResult.Cancel:
        {
            break;
        }
        default:
        {
            break;
        }
    }
}

private void изменитьToolStripMenuItem_Click(object sender, EventArgs e)
{
    VahtUpdDlg vahtUpdDlg = new VahtUpdDlg();
    DataGridViewRow selectedRow = vahtsDgv.SelectedRows[0];
    int code = (int)selectedRow.Cells[0].Value;
    string type = (string)selectedRow.Cells[1].Value;
    DateTime date = (DateTime)selectedRow.Cells[2].Value;
    int duration = (int)selectedRow.Cells[3].Value;
    string worker = selectedRow.Cells[4].Value.ToString() + " " +
(string)selectedRow.Cells[5].Value;

    vahtUpdDlg.codeLabel.Text = code.ToString();
    vahtUpdDlg.typeComboBox.DataSource = FillVahtTypes();
    vahtUpdDlg.typeComboBox.SelectedItem = type;
    vahtUpdDlg.dateTimePicker.Value = date;
    vahtUpdDlg.timeNumeric.Value = duration;
    vahtUpdDlg.workerComboBox.DataSource = FillvahtWorkerNumbers();
    vahtUpdDlg.workerComboBox.SelectedItem = worker;
    switch (vahtUpdDlg.ShowDialog())
    {
        case DialogResult.OK:
        {
            SqlCommand cmd = new SqlCommand("updateVaht", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@num", code);
            cmd.Parameters.AddWithValue("@type",
vahtUpdDlg.typeComboBox.SelectedItem.ToString());
            cmd.Parameters.AddWithValue("@date",
vahtUpdDlg.dateTimePicker.Value);
            cmd.Parameters.AddWithValue("@duration",
vahtUpdDlg.timeNumeric.Value);
            cmd.Parameters.AddWithValue("@worker",
long.Parse(vahtUpdDlg.workerComboBox.SelectedItem.ToString().Split(" ")[0]));

```

```

        cmd.ExecuteNonQuery();
        FillVahtDgv();
        break;
    }
    case DialogResult.Cancel:
    {
        break;
    }
    default:
    {
        break;
    }
}

private void удалитьToolStripMenuItem_Click(object sender, EventArgs e)
{
    VahtRemovedDlg vahtRemovedDlg = new VahtRemovedDlg();

    DataGridViewRow selectedRow = vahtsDgv.SelectedRows[0];
    int code = (int)selectedRow.Cells[0].Value;
    string type = (string)selectedRow.Cells[1].Value;
    DateTime date = (DateTime)selectedRow.Cells[2].Value;
    int duration = (int)selectedRow.Cells[3].Value;
    string worker = selectedRow.Cells[4].Value.ToString() + " " +
(string)selectedRow.Cells[5].Value;

    vahtRemovedDlg.numLabel.Text = code.ToString();
    vahtRemovedDlg.typeLabel.Text = type;
    vahtRemovedDlg.dateLabel.Text = date.ToString();
    vahtRemovedDlg.timeLabel.Text = duration.ToString();
    vahtRemovedDlg.workerLabel.Text = worker;
    switch (vahtRemovedDlg.ShowDialog())
    {
        case DialogResult.OK:
        {
            SqlCommand cmd = new SqlCommand("deleteVaht", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@num", code);
            cmd.ExecuteNonQuery();
            FillVahtDgv();
            break;
        }
        case DialogResult.Cancel:
        {
            break;
        }
        default:
        {
            break;
        }
    }
}

private void добавитьToolStripMenuItem2_Click(object sender, EventArgs e)
{
    IssueAddDlg issueAddDlg = new IssueAddDlg();

    issueAddDlg.AutorComboBox.DataSource = FillStudNumbers();

    switch (issueAddDlg.ShowDialog())
    {
        case DialogResult.OK:
        {
            SqlCommand cmd = new SqlCommand("insertIssue", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@title",
issueAddDlg.titleTextBox.Text);

```

```

        cmd.Parameters.AddWithValue("@text",
issueAddDlg.textTextBox1.Text);
        cmd.Parameters.AddWithValue("@author",
long.Parse(issueAddDlg.AutorComboBox.SelectedItem.ToString()));
        cmd.ExecuteNonQuery();
        FillIssueDgv();
        break;
    }
    case DialogResult.Cancel:
    {
        break;
    }
    default:
    {
        break;
    }
}
}

private void обновитьСтатусToolStripMenuItem_Click(object sender, EventArgs e)
{
    IssueUpdateDlg issueUpdateDlg = new IssueUpdateDlg();
    DataGridViewRow selectedRow = issuesDgv.SelectedRows[0];

    int code = (int)selectedRow.Cells[0].Value;
    string title = (string)selectedRow.Cells[1].Value;
    string status = (string)selectedRow.Cells[2].Value;
    DateTime date = (DateTime)selectedRow.Cells[3].Value;
    string text = (string)selectedRow.Cells[4].Value;
    string authorName = (string)selectedRow.Cells[5].Value;

    issueUpdateDlg.numLabel.Text = code.ToString();
    issueUpdateDlg.titleLabel.Text = title;
    issueUpdateDlg.statusComboBox.DataSource = statusList;
    issueUpdateDlg.statusComboBox.SelectedItem = status;
    issueUpdateDlg.dataLabel.Text = "Дата составления: " + date.ToString();
    issueUpdateDlg.textTextBox.Text = text;
    issueUpdateDlg.authorLabel.Text = "Составитель: " + authorName;

    switch (issueUpdateDlg.ShowDialog())
    {
        case DialogResult.OK:
        {
            SqlCommand cmd = new SqlCommand("updateIssue", conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@num", code);
            cmd.Parameters.AddWithValue("@status",
issueUpdateDlg.statusComboBox.SelectedItem.ToString());
            cmd.ExecuteNonQuery();
            FillIssueDgv();
            break;
        }
        case DialogResult.Cancel:
        {
            break;
        }
        default:
        {
            break;
        }
    }
}

private void вычислитьОбщуюСтоимостьToolStripMenuItem_Click(object sender,
EventArgs e)
{
    inventAllCostDlg allCostDlg = new inventAllCostDlg();
    SqlCommand cmd = new SqlCommand("computeAllCost", conn);

```

```

        cmd.CommandType = CommandType.StoredProcedure;
        int cost = 0;
        cmd.Parameters.AddWithValue("@cost",cost);
        cmd.Parameters[0].Direction = ParameterDirection.Output;
        cmd.ExecuteNonQuery();
        cost = (int)cmd.Parameters[0].Value;
        allCostDlg.allCostLabel.Text = "Общая стоимость всего инвентаря : " +
cost.ToString() + " руб.";
        allCostDlg.Show();
    }
}
}

```

Код автогенерируемый конструктором форм Visual studio представлен на гитхабе:

<https://github.com/KyKyPy3HuK/ClientServerEF>

QR код:

