

**Министерство науки и высшего образования РФ**  
**Федеральное государственное автономное образовательное учреждение**  
**высшего профессионального образования**  
**«Санкт-Петербургский государственный электротехнический**  
**Университет «ЛЭТИ» им. В.И.Ульянова(Ленина)»**  
**(СПБГЭТУ «ЛЭТИ»)**

**ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАТИКИ**  
**Кафедра вычислительной техники**  
**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ**  
**по дисциплине «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»**  
**«Создание программного комплекса средствами**  
**объектно-ориентированного программирования»**

Студент :

гр.0306 Семёнов Михаил

Преподаватель :

Павловский М.Г.

Санкт-Петербург

2022

# 1. Техническое задание

## 1.1 Введение

Программный комплекс (ПК) администрирования гостиницы предназначен для использования в составе системы программно-информационного обеспечения учета и администрирования.

## 1.2 Основание для разработки

Основанием для разработки ПК «Учет, администрирование и редактирование записей работников гостиницы» является курсовой проект по дисциплине «Объектно-ориентированное программирование».

## 1.3 Назначение разработки

ПК «Учет, администрирование и редактирование записей работников гостиницы» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности лица (ОЛ), ответственного за обработку информации станции технического обслуживания.

ПК «Учет, администрирование и редактирование записей работников гостиницы станции» предназначен для автоматизации следующих процессов:

- учет и администрирование информации о работниках гостиницы;
- учет и администрирование информации о номерах гостиницы;
- получение справочной информации о количестве занятых номеров;
- отчет о работе гостиницы в виде гостей в номерах и ФИО работников отеля, ответственных за эти номера.

## 1.4 Требования к программе

### 1.4.1 Требования к функциональным характеристикам

#### 1.4.1.1 Перечень функций

ПК «Учет, администрирование и редактирование записей работников гостиницы» должен обеспечивать выполнение следующих функций:

- просмотр, добавление, удаление и изменение в базы данных (БД);
- выдача справочной информации, хранимой в БД, по запросам ОЛ.

#### 1.4.1.2 Требования к составу выполняемых функций

##### 1.4.1.2.1 Функция «просмотр, добавление, удаление и изменение в базы данных (БД)»

Ввод, просмотр, добавление, удаление и изменение в БД должны обеспечивать ведение и хранение следующих данных:

- данных о работниках гостиницы;
- данных о занятых номерах;
- количестве свободных номеров;

#### *1.4.1.2 Требования к организации и форме представления выходных данных*

Выходные данные должны быть представлены в виде таблицы содержащий описание необходимых информационных объектов, выполненного по средствам представления его характеристик.

#### *1.4.1.3 Требования к организации и форме представления входных данных*

Входная информация для задачи «Учет, администрирование и редактирование записей работников гостиницы» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов.

#### **1.4.2 Требования к надежности**

ПК «Учет, администрирование и редактирование записей работников гостиницы» должен устойчиво функционировать при соблюдении гарантии устойчивого функционирования операционной системы и системы управления базой данных. Под устойчивой работой ПК понимается непрерывное функционирование программы в отсутствии критических сбоев, приводящих к аварийному завершению. Кроме того, должен быть обеспечен контроль входных данных на предмет соответствия предполагаемому типу.

#### **1.4.3 Условия эксплуатации**

Выполнение ПК «Учет, администрирование и редактирование записей работников гостиницы» своих функций должно быть обеспечено для однопользовательского режима работы с монопольным доступом к базе данных.

#### **1.4.4 Требование к составу и параметрам технических средств**

Задача должна решаться на ПЭВМ типа IBM PC или совместимой с ней с процессором Pentium III 500 и выше, ОЗУ не менее 128Мб, HDD не менее 4 Гб, монитор SVGA (цветной)15", видеокарта 64 Мб, клавиатура 102 кл., манипулятор типа "мышь".

#### **1.4.5 Требование к информационной и программной совместимости**

Выходная и входная информация ПК «Учет, администрирование и редактирование записей работников станции» должна быть удобна для визуального восприятия. ПК должен быть выполнен на языке программирования высокого уровня Java и должен быть совместим с операционной системой Windows.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

- закрытые и открытые члены классов;
- наследование;
- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

## 1.5 Требования к программной документации

Программная документация (ПД) должна удовлетворять требованиям стандартов ЕСПД.

Документация должна быть представлена в следующем составе:

1. описание процесса проектирования ПК;
2. руководство оператора;
3. исходные тексты ПК.

## 1.6 Стадии и этапы разработки

1. Разработка технического задания;
2. Описание вариантов использования ПК;
3. Создание прототипа интерфейса пользователя;
4. Разработка объектной модели ПК;
5. Построение диаграмм программных классов;
6. Описание поведения ПК;
7. Построение диаграмм действий;

## 1.7 Порядок контроля и приемки

В процессе приема работы устанавливается соответствие ПК и прилагаемой документации требованиям, обозначенным в техническом задании.

## 2 Проектирование ПК

### 2.1 Описание вариантов использования ПК

Развернутое описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Прецеденты являются сценарием или вариантом использования ПК при взаимодействии с внешней средой. Они являются продолжением описаний требований и функциональных спецификаций, указанных в техническом задании. Прецедент изображается в виде эллипса, в котором содержится имя прецедента. Название прецедента обязательно включает в себя глагол, выражающий суть выполняемой функции. С помощью прецедентов описывается функционирование ПК с точки зрения внешнего пользователя, который называется в UML актором (actor). Актор представляет собой любую внешнюю по отношению к моделируемой системе сущность (человек, программная система, устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Актор на диаграмме изображается пиктограммой в виде человечка, под которым указано его имя. Совокупность функций, реализуемых ПК, изображается в виде диаграммы (use case diagram). Для построения диаграммы необходимо определить акторы, прецеденты (функции) и взаимоотношение между акторами и прецедентами, и между прецедентами, если один прецедент расширяет или использует другой. В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи коммуникации (communication), использования (uses) и расширения (extends).

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. \*. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

Связь использования предполагает, что один прецедент всегда применяет функциональные возможности другого. С помощью таких связей структурируют прецеденты, показывая тем самым, какой прецедент является составной частью другого прецедента. Такой включаемый прецедент является абстрактным прецедентом в том смысле, что он не может исполняться независимо от других прецедентов, а лишь в их составе. Связь использования изображается с помощью стрелок и слова «uses» (использование). Направление стрелки указывает, какой прецедент используется для реализации функциональности другого прецедента.

Связь расширения задается в том случае, если необходимо показать родственные отношения между двумя прецедентами. Один из них является базовым, а другой его расширением. Базовый прецедент не зависит от расширяющих прецедентов и способен функционировать без них. С другой стороны, расширяющие прецеденты без базового прецедента функционировать не могут. Связи расширения изображают в виде стрелки со словом «extends» (расширение), которая имеет направление от базового прецедента к расширяемому.

Прецеденты необходимо ранжировать, чтобы в начальных циклах разработки реализовать наиболее приоритетные из них. Разбиение функциональности системы на отдельные прецеденты служит примерно той же цели, что и разбиение сложного алгоритма на подпрограммы. Основная стратегия должна заключаться в том, чтобы сначала сконцентрировать внимание на тех прецедентах, которые в значительной мере определяют базовую архитектуру ПК.

Диаграмма прецедентов представлена на рис. 2.1.

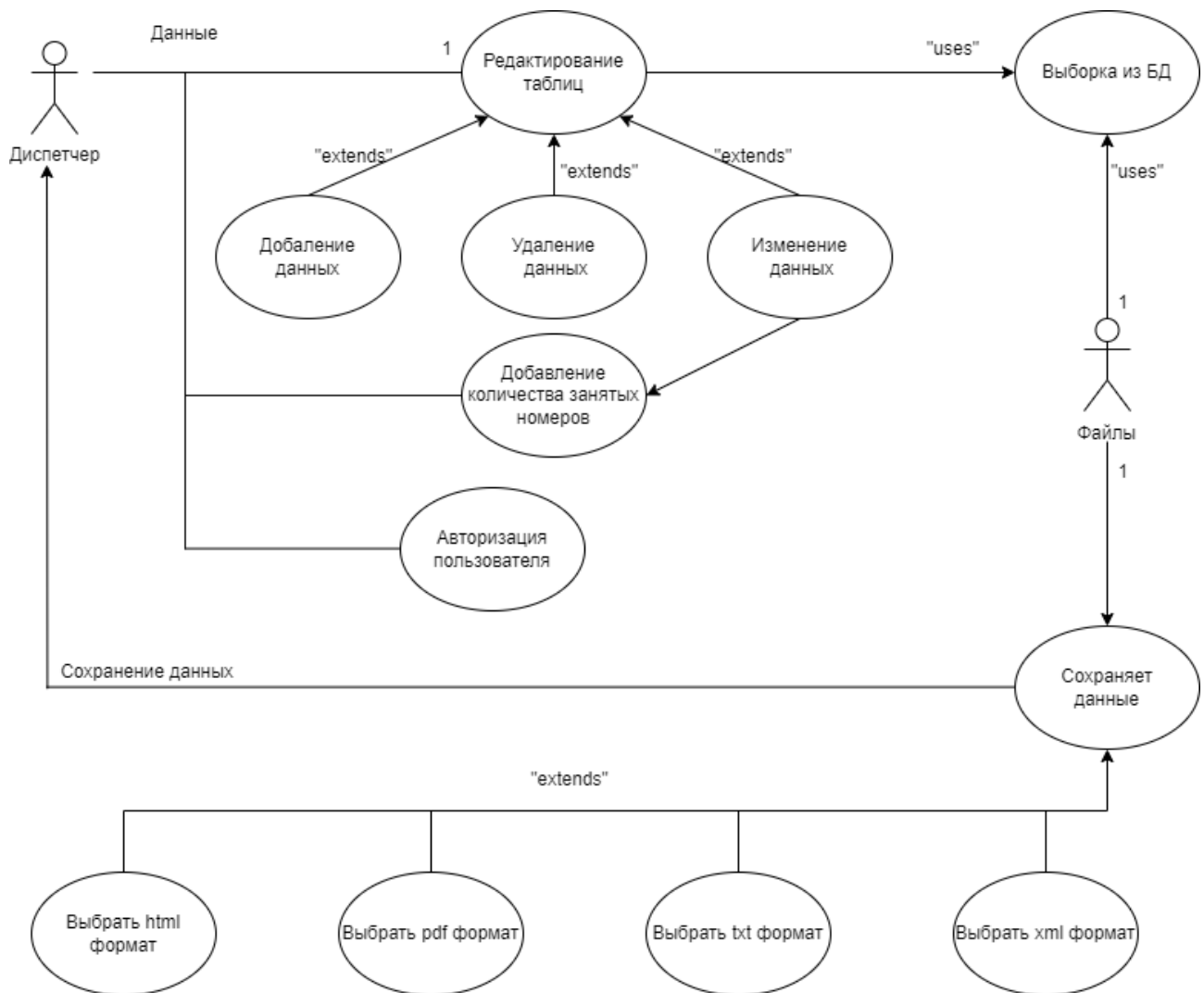
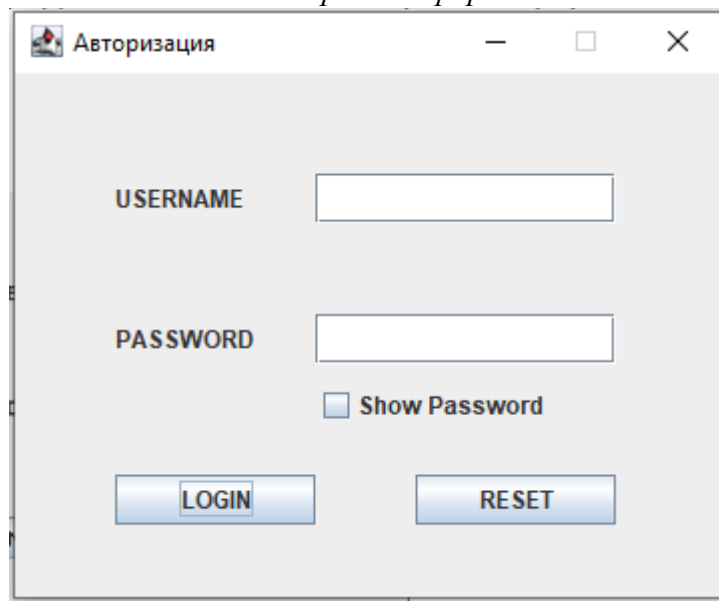


Рис.2.1. Диаграмма прецедентов

## 2.3 Создание прототипа интерфейса пользователя

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идет об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм.

Рис. 2.2 –экранная форма входа и регистрации; рис.2.3 –главное экранная форма; рис. 2.4 – экранная форма таблиц с данными; рис. 2.5 – экранная форма выбора формата отчёта; рис. 2.6 – экранная форма изменения количества занятых номеров; рис. 2.7 – экранная форма добавления строки в БД; рис. 2.8 – экранная форма сохранения таблицы в файл. В табл. 2.1 представлено описание экранных форм.



The image shows a window titled "Авторизация" (Authorization). It has a standard Windows-style title bar with a minimize button, a maximize button (disabled), and a close button. The main area is light gray and contains the following elements:

- A label "USERNAME" followed by a text input field.
- A label "PASSWORD" followed by a text input field.
- A checkbox labeled "Show Password" which is currently unchecked.
- Two buttons at the bottom: "LOGIN" and "RESET".

Рис. 2.2. Экранная форма входа и регистрации

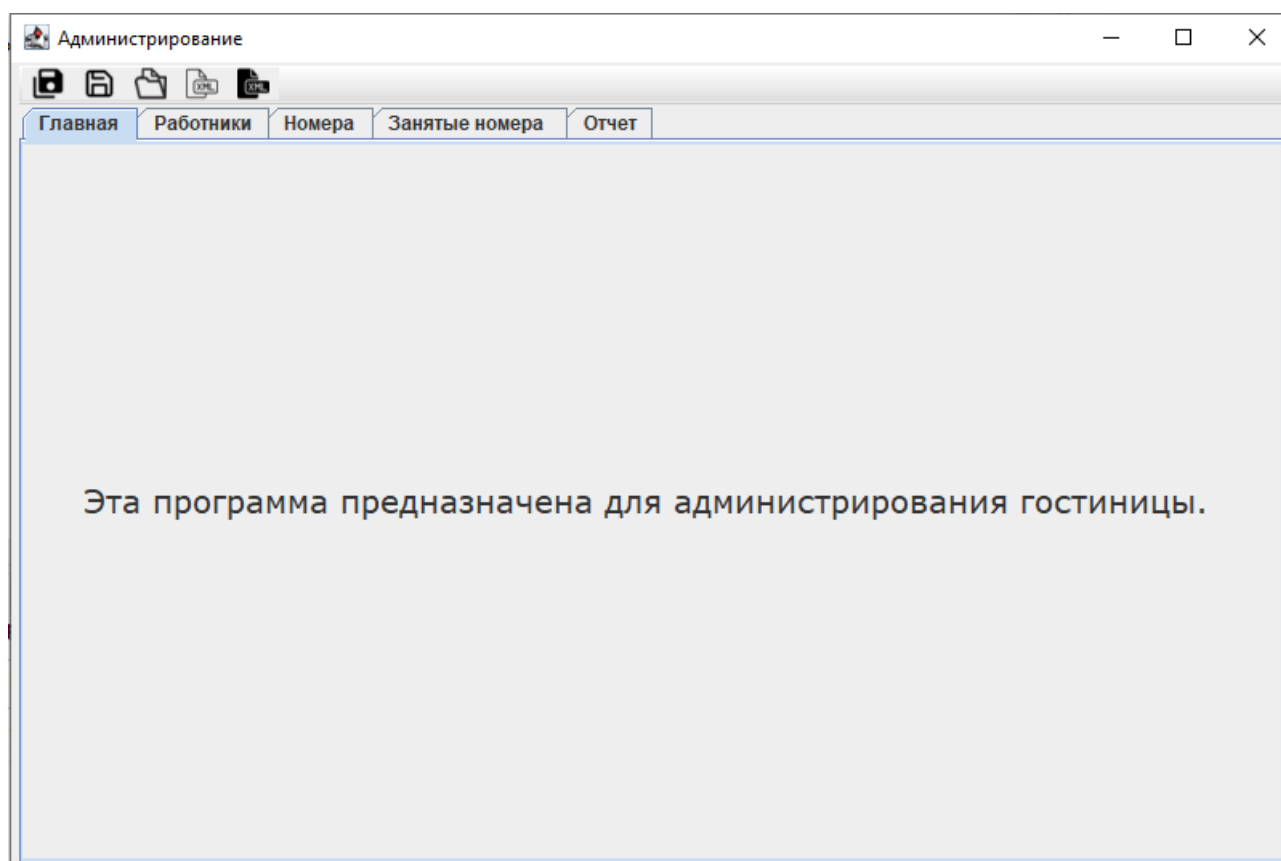
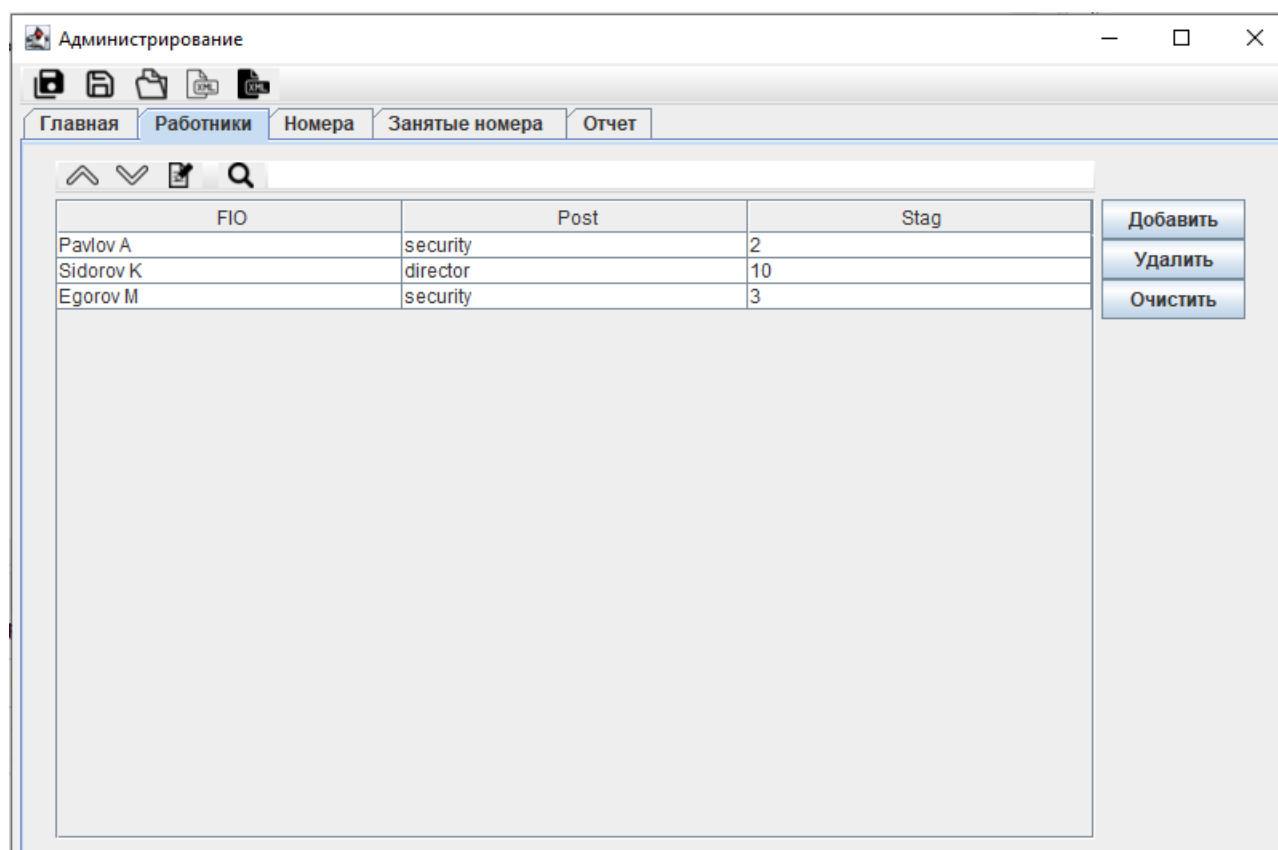


Рис. 2.3. Главная экранная форма





*Рис. 2.4. Экранная форма таблиц с данными*



*Рис. 2.5. Экранная форма выбора формата отчёта*

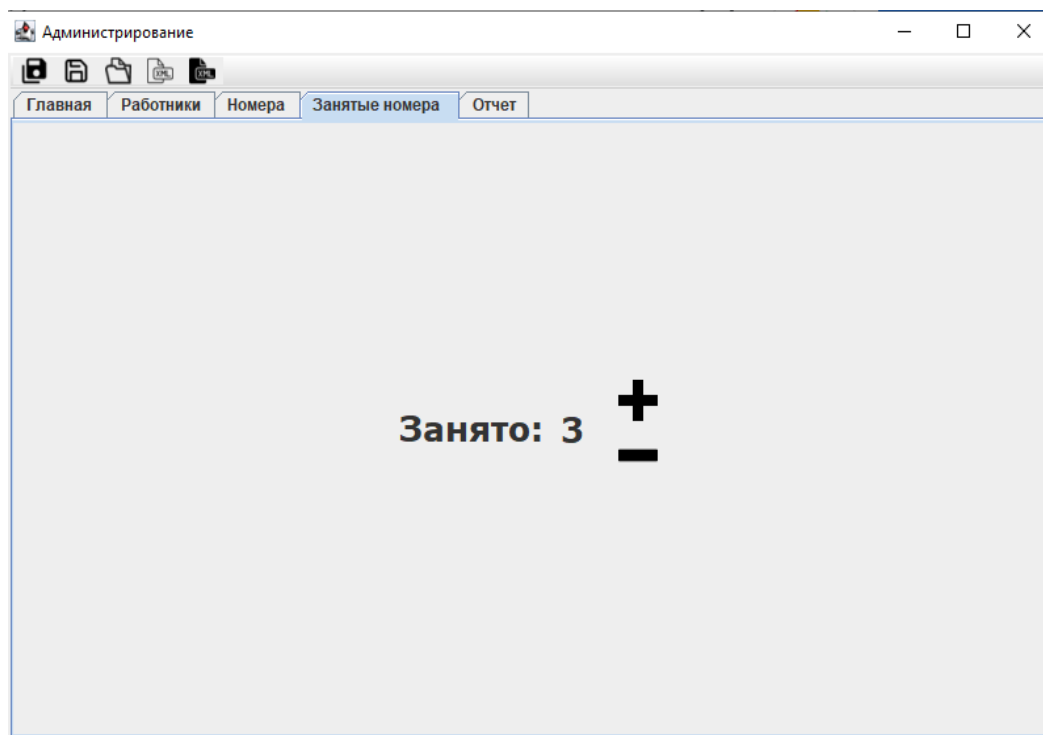


Рис. 2.6. Экранная форма изменения количества занятых номеров

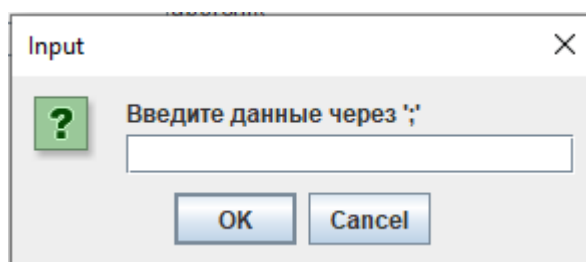


Рис. 2.7. Экранная форма добавления строки в БД

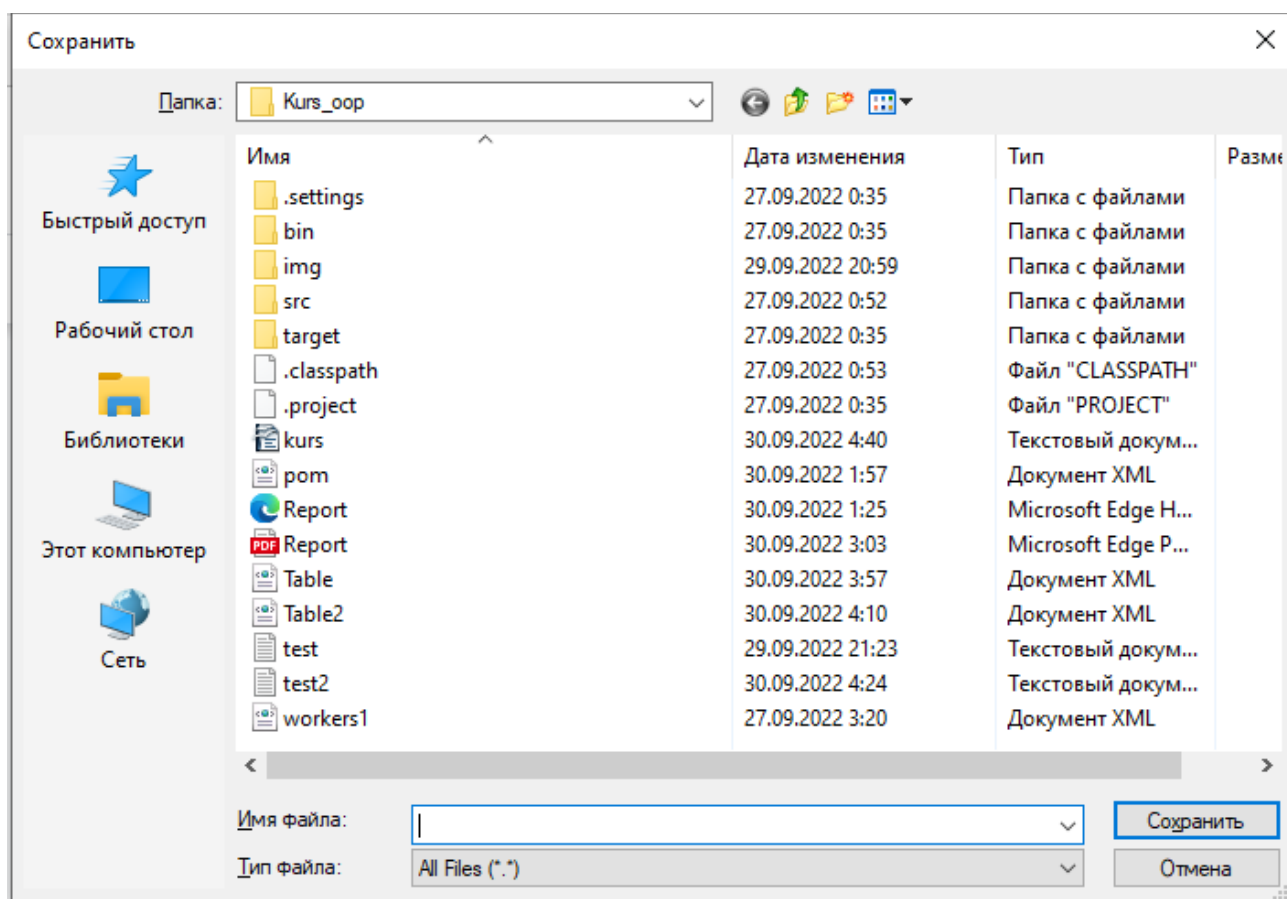


Рис. 2.8. Экранная форма сохранения в файл

Таблица 2.1

Экранная форма	Элементы управления	Действия пользователя	Отклик системы
Авторизация	Кнопки: «Login» «Reset» «Show Password»	Нажать кнопку «Login»	Открыть окно, уведомляющее об успешной авторизации, и войти в главное меню программы
		Нажать кнопку «Reset»	Сбросить введенный данные
		Нажатие кнопки «Show Password»	Показать введенный пароль
Главная экранная форма	Кнопки: «Сохранить все» «Сохранить» «Загрузить» «Сохранить в формате xml» «Загрузить из xml»  Список вкладок: «Главная» «Работники» «Номера» «Занятые номера» «Отчет»	Нажать кнопку «Сохранить все»	Сохранение данных таблиц из всех вкладок в формате txt
		Нажать кнопку «Сохранить»	Сохранение таблицы выбранной вкладки в формате txt
		Нажать кнопку «Загрузить»	Заполнение всех таблиц из файла формата txt
		Нажать кнопку «Сохранить в формате xml»	Сохранение данных из всех таблиц в формате xml
		Нажать кнопку «Загрузить из xml»	Заполнение всех таблиц из файла формата xml
		Выбрать вкладку с нужной таблицей	Открытие выбранной формы
Вкладка «Главная»	-		Выводит описание программы
Вкладка «Работники»	Кнопки: «Вверх» «Вниз» «Выбрать формат отчета» «Поиск» «Добавить» «Удалить» «Очистить»  Список элемента для ввода данных: «Строка для поиска в таблице»	Нажать кнопку «Вверх»	Сдвинуть вверх выбранную строку таблицы
		Нажать кнопку «Вниз»	Сдвинуть вниз выбранную строку таблицы
		Нажать кнопку «Выбрать формат отчета»	Открыть окно с выбором формата отчета
		Нажать кнопку «Поиск»	Выполнить поиск по таблице в соответствии с введенной строкой
		Нажать кнопку «Добавить»	Добавить строку в таблицу и занести изменения в БД
		Нажать кнопку «Удалить»	Удалить выбранную строку из таблицы и занести данные в БД
		Нажать кнопку «Очистить»	Удалить все данные из таблицы и занести все данные в БД
Вкладка «Номера»	Кнопки:	Нажать кнопку «Вверх»	Сдвинуть вверх

	«Вверх» «Вниз» «Выбрать формат отчета» «Поиск» «Добавить» «Удалить» «Очистить»  Список элемента для ввода данных: «Строка для поиска в таблице»		выбранную строку таблицы
		Нажать кнопку «Вниз»	Сдвинуть вниз выбранную строку таблицы
		Нажать кнопку «Выбрать формат отчета»	Открыть окно с выбором формата отчета
		Нажать кнопку «Поиск»	Выполнить поиск по таблице в соответствии с введенной строкой
		Нажать кнопку «Добавить»	Добавить строку в таблицу и занести данные в БД
		Нажать кнопку «Удалить»	Удалить выбранную строку из таблицы и занести данные в БД
		Нажать кнопку «Очистить»	Удалить все данные из таблицы и занести данные в БД
Вкладка «Занятые номера»	Кнопки: «+» «-»	Нажать кнопку «+»	Увеличить количество на 1
		Нажать кнопку «-»	Уменьшить количество на 1
Вкладка «Отчет»	Кнопки: «Вверх» «Вниз» «Выбрать формат отчета» «Поиск» «Добавить» «Удалить» «Очистить»  Список элемента для ввода данных: «Строка для поиска в таблице»	Нажать кнопку «Вверх»	Сдвинуть вверх выбранную строку таблицы
		Нажать кнопку «Вниз»	Сдвинуть вниз выбранную строку таблицы
		Нажать кнопку «Выбрать формат отчета»	Открыть окно с выбором формата отчета
		Нажать кнопку «Поиск»	Выполнить поиск по таблице в соответствии с введенной строкой
		Нажать кнопку «Добавить»	Добавить строку в таблицу
		Нажать кнопку «Удалить»	Удалить выбранную строку из таблицы
		Нажать кнопку «Очистить»	Удалить все данные из таблицы
Окно выбора формата отчета	Кнопки: «pdf» «Html»	Нажать кнопку «pdf»	Создать отчет выбранной вкладки в формате pdf
		Нажать кнопку «Html»	Создать отчет выбранной вкладки в формате html

## 2.3. Разработка объектной модели ПК

Объектная модель не описывает структуру ПК, она отображает основные понятия предметной области в виде совокупности типов объектов (сущностей). Сущности строятся путем выделения их из предметной области и анализа прецедентов. На диаграмме сущность обозначается прямоугольником, внутри которого записывается имя сущности, ее атрибуты и операции.

Атрибуты описывают свойства сущности. В объектную модель включаются те атрибуты, для которых определены соответствующие требования или для которых предполагается хранить определенную информацию. Атрибут характеризуется именем и типом. Для атрибута рекомендуется использовать простые типы данных (число, строка, дата, время и другие).

Описание операций помогает определить поведение объектов сущности. На этом этапе, прежде всего, определяется внутреннее поведение каждого объекта сущности, без учета взаимодействия с другими объектами предметной области. На диаграмме обычно

указывается только имя операции, а ее подробное описание приводится в отдельной таблице. В таблице должно содержаться краткое описание назначения операции, ее имя и список входных и выходных параметров.

Ассоциация между сущностями отражает некоторое бинарное отношение между ними. Ассоциация обозначается проведенной между сущностями линией, с которой связывается определенное имя. Имя записывается в глагольной форме, и оно должно отражать семантический смысл отношения. Стрелка на линии указывает, в каком направлении нужно читать имя. На концах линии могут содержаться выражения, определяющие количественную связь между экземплярами сущности (кратность). Кратность определяет, сколько экземпляров одной сущности может быть ассоциировано с одним экземпляром другой сущности. Примеры кратностей:

- 0 .. \* - нуль или больше,
- 1 .. \* - один или больше,
- 1 – ровно один.

Необходимо устанавливать отношения ассоциации между двумя сущностями в том случае, если объект одной сущности должен знать об объекте другой. Прежде всего, следует включать в модель те ассоциации, которые отражают структурные отношения («содержит», «включает», «хранит» и т.д.), или те, которые должны сохраняться в течение некоторого времени.

Диаграмма сущностей представлена на рис. 2.7. Детальное описание операций представлено в табл. 2.2.

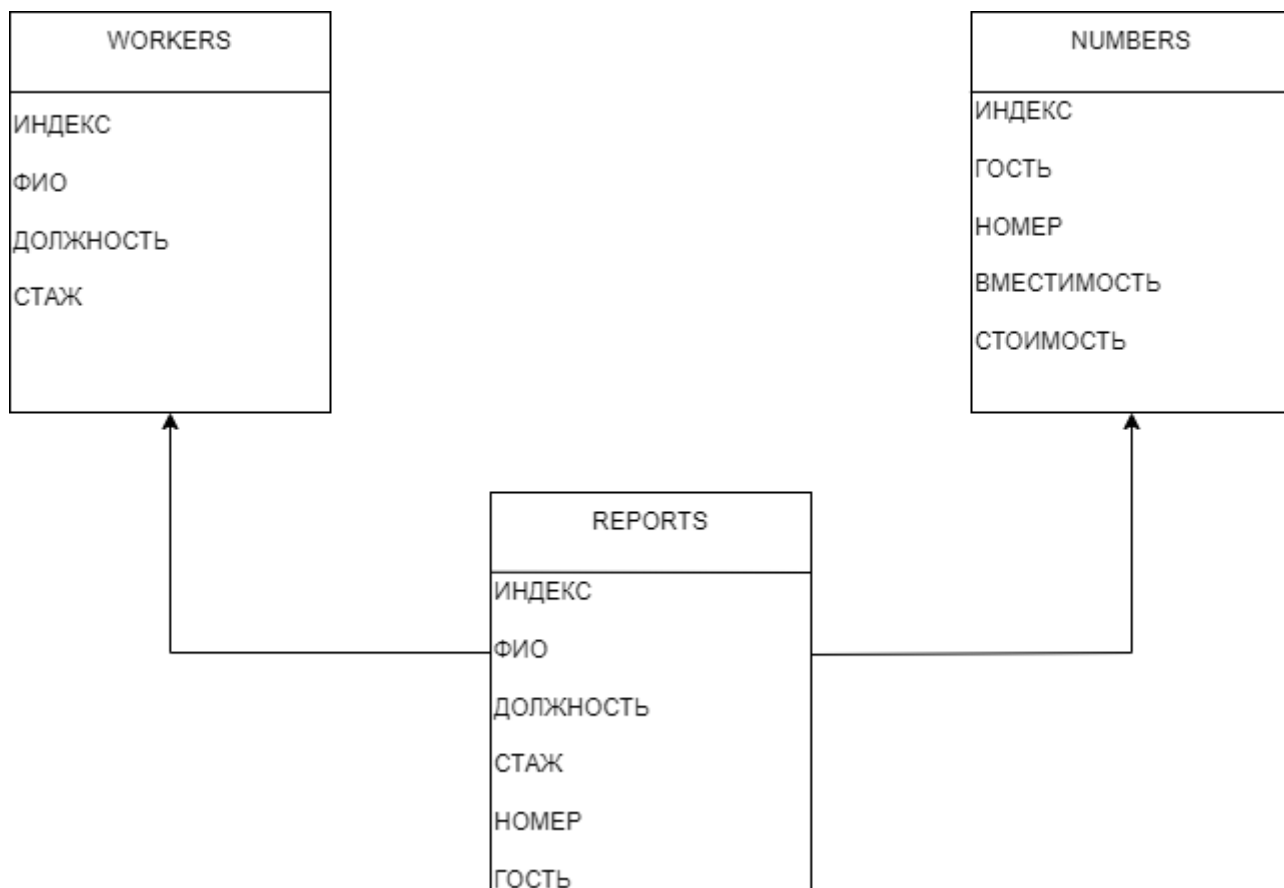


Рис. 2.7. Диаграмма сущностей

Таблица 2.2

Сущность	Имя операции	Параметры операции			Тип возвращаемого значения
		Вид	Название	Тип	
workers	Сохранить	Вых.	Сохранить выбранную таблицу	Файл	БД
	Добавить	Вх.	Данные	Строка	БД
numbers	Сохранить	Вых.	Сохранить выбранную таблицу	Файл	БД
	Добавить	Вх.	Данные	Строка	БД
reports	Сохранить	Вых.	Сохранить выбранную таблицу	Файл	БД
	Добавить	Вх.	Данные	Строка	БД

## 2.4 Построение диаграммы программных классов

Диаграмма классов (class diagram) иллюстрирует спецификации будущих программных классов и интерфейсов. Она строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника. Имя программного класса может совпадать с именем сущности или быть другим. Но поскольку для записи идентификаторов переменных в языках программирования используют латинские буквы, то и имена программных классов, и имена их атрибутов, как правило, записываются латинскими буквами. Атрибуты и операции класса перечисляются в горизонтальных отделениях этого прямоугольника. Атрибутам и методам классов должны быть присвоены права доступа. Права доступа помечаются специальными знаками:

- + - означает открытый (public) доступ;
- - означает скрытый (private) доступ;
- # - означает наследуемый (protected) доступ.

При описании атрибутов после двоеточия указывается их тип, а при описании методов класса возвращаемое значение (для конструкторов возвращаемое значение не указывается).

В диаграмме классов могут вводиться дополнительно новые атрибуты, операции и связи или осуществляться конкретизация ассоциаций, указанных в объектной модели. На диаграмме классов могут быть три вида отношений: ассоциация, агрегация и наследование.

На диаграмме классов ассоциация имеет такое же обозначение, как и в объектной модели. На линиях ассоциации может присутствовать стрелка. Это стрелка видимости, которая показывает направление послышки запросов в ассоциации. Стрелка видимости также показывает, какой из классов содержит компоненты для реализации отношения ассоциации, иными словами, кто является инициатором послышки запроса к другому объекту. Ассоциация без стрелки является двунаправленной.

Агрегирование — это отношение между классами типа целое/часть. Агрегируемый класс в той или иной форме является частью агрегата. На практике это может быть реализовано по-разному. Например, объект класса-агрегата может хранить объект агрегируемого класса, или хранить ссылку на него. Агрегирование изображается на диаграмме полым ромбом на конце линии со стороны агрегирующего класса (агрегата). Если агрегируемый объект может быть создан только тогда, когда создан агрегат, а с уничтожением агрегата уничтожаются и все агрегируемые объекты, то такое агрегирование называется сильным и отображается в виде закрашенного ромба.

Наследование — это отношение типа общее-частное между классами. Его следует вводить в том случае, когда поведение и состояние различных классов имеют общие черты. Наследование связывает конкретные классы с общими или в терминологии языков

программирования производные классы (подклассы) с базовыми классами (суперклассами). На диаграммах наследование изображается в виде стрелки с полым треугольником, идущей от производного класса к базовому. Если один производный класс наследует несколько базовых, то такое наследование называется множественным.

Диаграмма классов представлена на рис. 2.8.

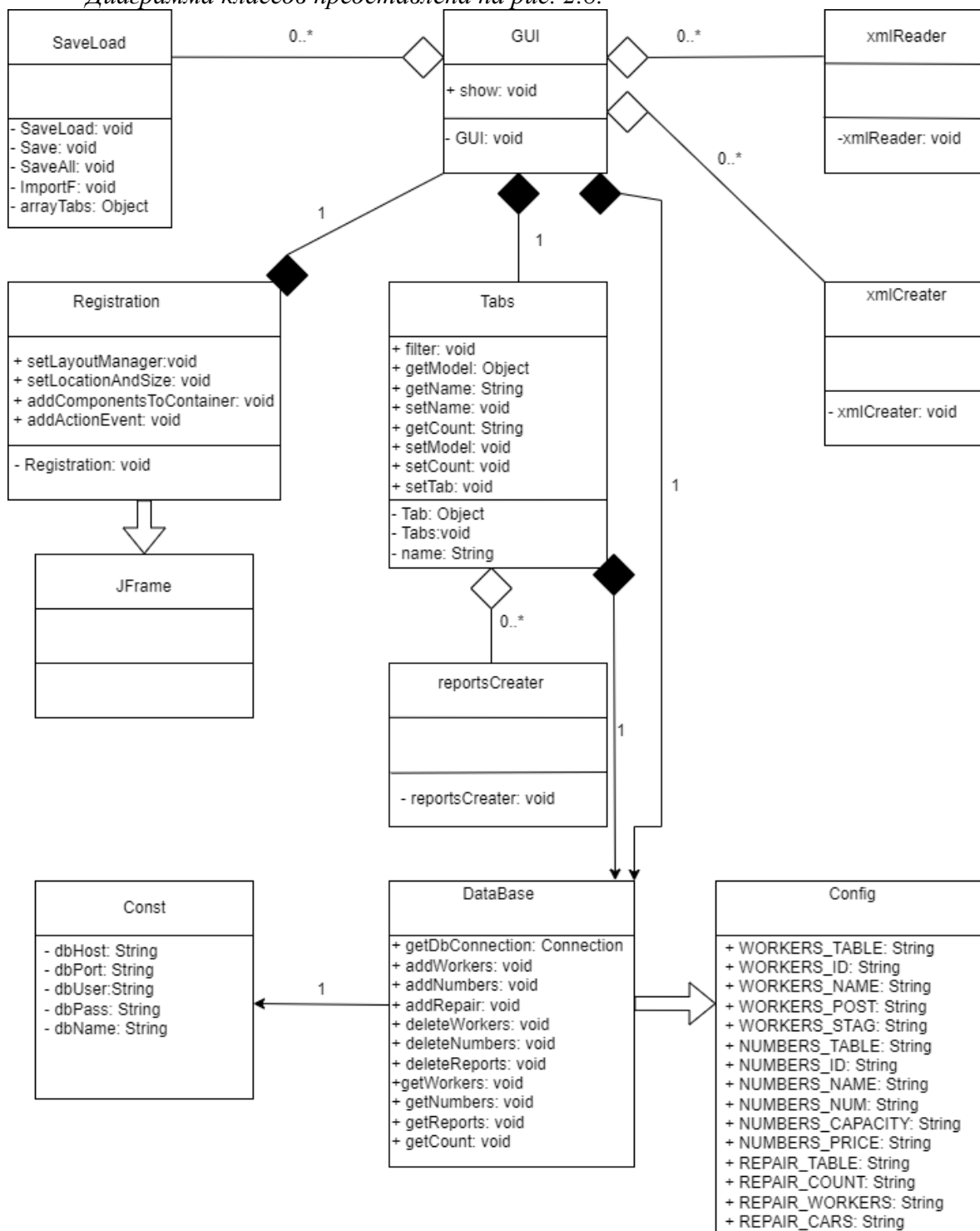


Рис. 2.8.Диаграмма классов



## 2.5 Описание поведения ПК

Поведение ПК представляет собой описание того, какие действия выполняет ПК, без определения механизма их реализации. Одной из составляющей такого описания является диаграмма последовательностей (sequence diagram). Диаграмма последовательностей является схемой, которая для определенного сценария прецедента показывает генерируемые пользователями и объектами события (запросы) на выполнение некоторой операции и их порядок. Диаграммы последовательности имеют две размерности: вертикальная представляет время, горизонтальная - различные объекты. Чтобы построить диаграмму последовательностей необходимо выполнить следующие действия:

1. Идентифицировать пользователей и объекты программных классов, участвующие в начальной стадии реализации сценария прецедента, и их изображения в виде прямоугольников расположить наверху в одну линию. Для каждого пользователя и объекта нарисовать вертикальную пунктирную линию, которая является линией их жизни. Внутри прямоугольника указываются подчеркнутое имя объекта и имя класса, к которому принадлежит объект.

2. Из объектной модели выбрать те операции, которые участвуют в реализации сценария. Если такие операции не были определены при построении диаграммы программных классов, то необходимо их описать и внести в модель.

3. На диаграмме последовательностей каждому запросу на выполнение операции должна соответствовать горизонтальная линия со стрелкой, начинающаяся от вертикальной линии того пользователя или объекта, который вызывает операцию, и заканчивающаяся на линии жизни того пользователя или объекта, который будет ее выполнять. Над стрелкой указывается номер операции, число итераций, имя операции и в скобках ее параметры. После описания операции может следовать комментарий, поясняющий смысл операции и начинающийся со знака "//".

Операция, которая реализует запрос, на линии жизни объекта обозначается прямоугольником. Порядок выполнения операций определяется ее номером, который указывается перед именем, и положением горизонтальной линии на диаграмме. Чем ниже горизонтальная линия, тем позже выполняется операция. В диаграммах последовательности принято применять вложенную систему нумерации, так как это позволяет отобразить их вложенность. Нумерация операций каждого уровня вложенности должна начинаться с 1.

На диаграмме последовательностей можно описать вызов операции по условию (конструкция if-else) и показать моменты создания и уничтожения объектов. Если объект создается или уничтожается на отрезке времени, представленном на диаграмме, то его линия жизни начинается и заканчивается в соответствующих точках, в противном случае линия жизни объекта проводится от начала до конца диаграммы. Символ объекта рисуется в начале его линии жизни; если объект создается не в начале диаграммы, то сообщение о создании объекта рисуется со стрелкой, проведенной к символу объекта. Если объект уничтожается не в конце диаграммы, то момент его уничтожения помечается большим крестиком "X".

Диаграмма последовательностей для операции добавления данных в БД, сохранения и создания отчета на рис. 2.9.

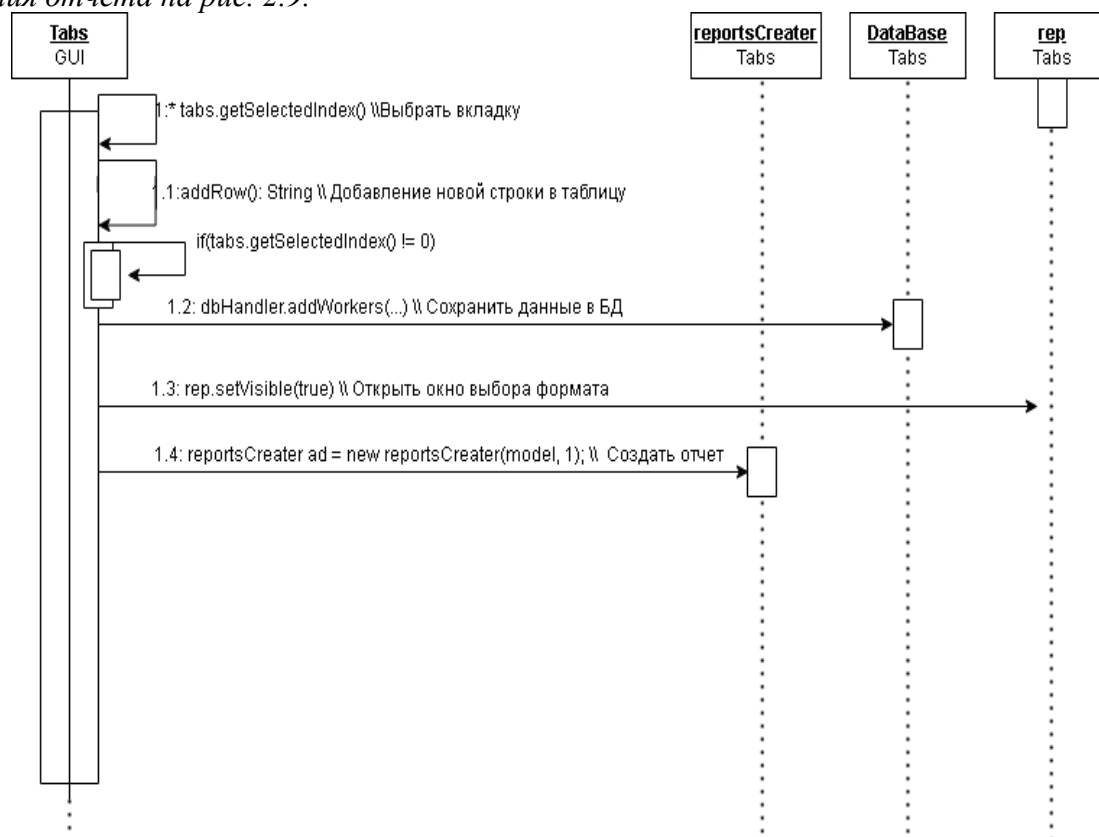


Рис.2.9 Диаграмма последовательностей для операции

## 2.6 Построение диаграммы действий

Диаграмма действий (activity diagram) строится для сложных операций. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются действия, а дугами — переходы от одного действия к другому. Она очень похожа на блок-схемы алгоритмов. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояние. Диаграмму деятельности принято строить таким образом, чтобы действия следовали сверху вниз. Отличительной чертой диаграммы действий является то, что в ней можно отобразить параллельные процессы. Для этой цели используется специальный символ (линия синхронизации), который позволяет задать разделение и слияние потоков управления. При этом разделение имеет один входящий переход и несколько выходящих, а слияние, наоборот, имеет несколько входящих переходов и один выходящий.

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Чтобы связать объекты с действиями, необходимо явно указать их на диаграмме деятельности. Для графического представления объектов, используются прямоугольник, в котором указывается подчеркнутое имя класса. Подчеркнутое имя означает, что на диаграмме задается объект, а не его класс. Далее после имени можно указать в прямых скобках значения атрибутов объекта после выполнения предшествующего действия. Такие прямоугольники объектов присоединяются к переходам отношением зависимости с помощью пунктирной линией со стрелкой.

Диаграмма действий представлена на рис. 2.10.

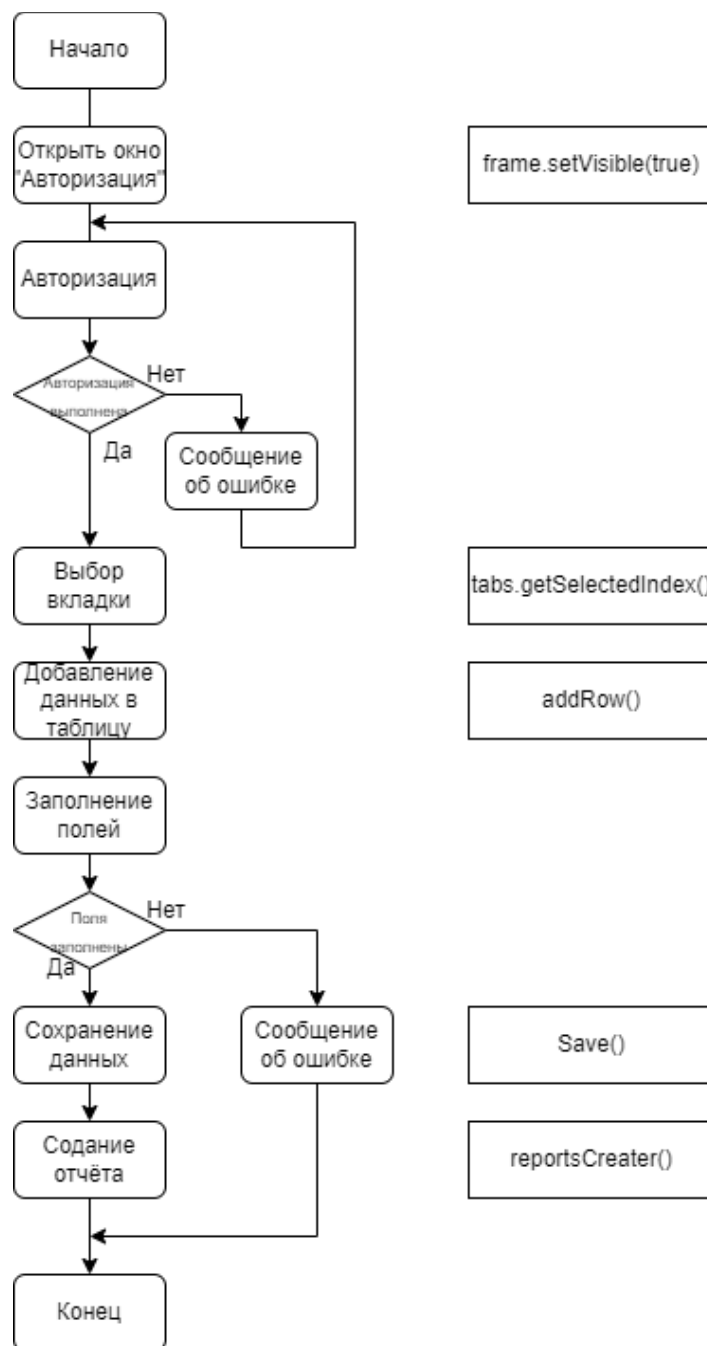


Рис. 2.10 Диаграмма действий

### 3. Руководство оператора

#### 3.1 Назначение программы

*ПК «Учет, администрирование и редактирование записей работников станции» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности ОЛ, ответственного за учет номеров в гостинице.*

В рамках ПК «Учет, администрирование и редактирование записей работников гостиницы» ОЛ может:

- добавлять, править и удалять информацию о работниках;
- добавлять, править и удалять информацию о номерах;
- добавлять, править и удалять информацию о клиентах;
- создавать отчёты;

#### 3.2 Условия выполнения программы

- ПК предназначен для функционирования под операционной средой Windows (10).
- Персональная электронно-вычислительная машина (ПЭВМ) должна обладать следующими характеристиками:
- тип процессора Pentium III 500 и выше;
- объем ОЗУ – не менее 128Мб;
- объем жесткого диска – не менее 4Гб;
- видеокарта – 64Мб;
- стандартная клавиатура;
- манипулятор типа "мышь".

#### 3.3 Описание задачи

В ПК должны храниться сведения о работниках, клиентах и номерах. Управляющий гостиницы может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- информация о сотрудниках;
- информация о номерах;
- информация о клиентах;

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

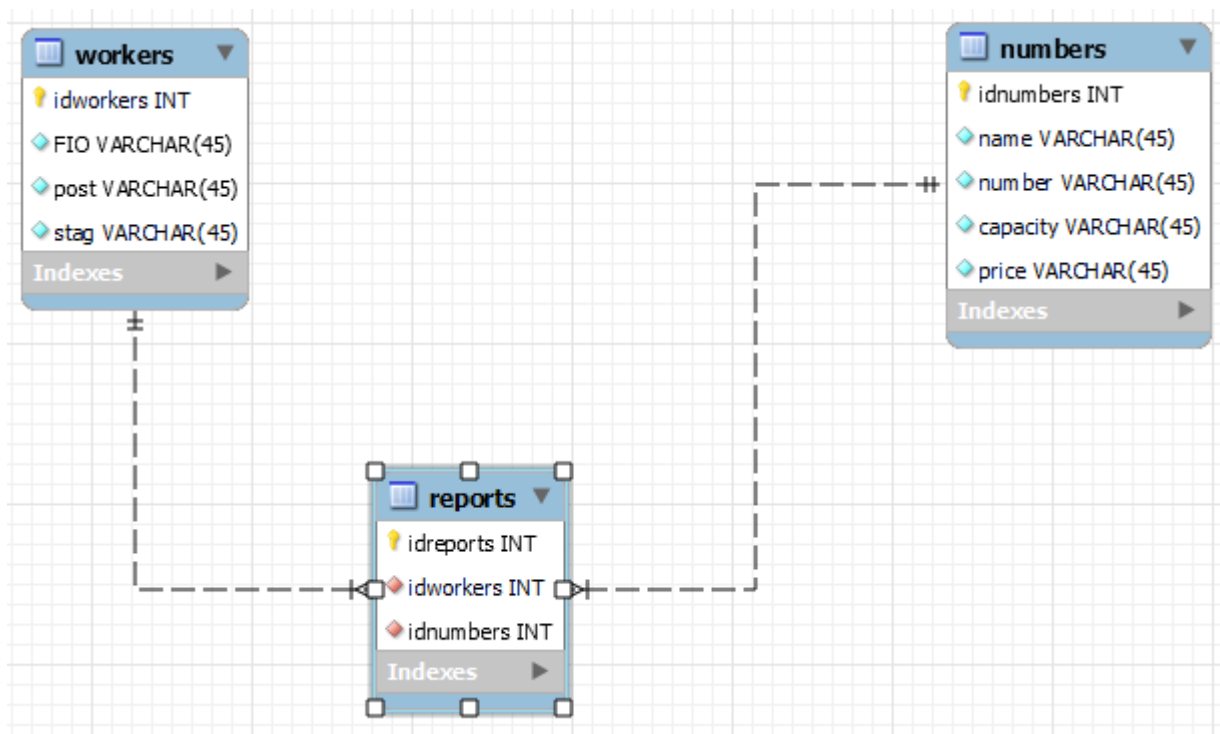
- закрытые и открытые члены классов;
- наследование;
- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

С целью выполнения поставленной задачи в процессе проектирования разработана общая модель ПК с выявлением основных объектов и связей между ними. На основании полученной модели разработаны программные классы.

Требования к коду ПК учтены при создании программных классов и непосредственном написании программы.

### 3.4 Входные и выходные данные

Исходные данные хранятся на сервере MySQL, при запуске программы таблицы workers и numbers заполняются из базы данных. Пользователь может добавить данные в эти таблицы с последующим автоматическим занесением их в БД. Таблица repairs содержит отчет о проделанной работе. Так, из первой таблицы выбирается сотрудник, ответственный за номер , а из второй — номер с данными гостя.



Выходные данные можно получить в виде формата txt, pdf, html, xml в виде таблицы, содержащей описание необходимых информационных объектов, выполненного посредством представления его характеристик, также они заносятся в БД.

### 3.5 Выполнение программы

#### 3.5.1 Подготовка к запуску (осуществляется один раз после установки ПК на ЭВМ)

Подготовка не требуется

#### 3.5.2. Запуск программы

При запуске программы на экране появится диалоговое окно представленное на рис.

3.1.

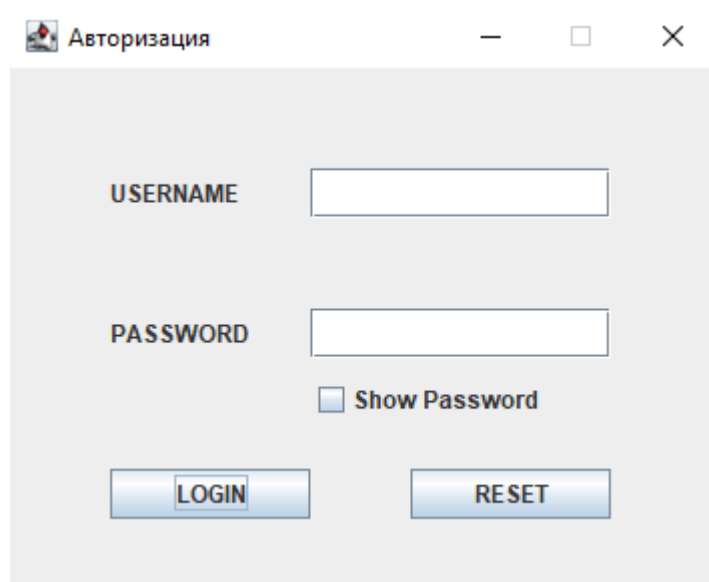
The image shows a Windows-style dialog box titled 'Авторизация' (Authorization). It has a standard title bar with minimize, maximize, and close buttons. The main area is light gray and contains two text input fields. The first field is labeled 'USERNAME' and the second is labeled 'PASSWORD'. Below the password field is a checkbox labeled 'Show Password'. At the bottom of the dialog, there are two buttons: 'LOGIN' on the left and 'RESET' on the right.

Рис. 3.1 Главное диалоговое окно

#### 3.5.3. Выполнение основных функций

##### 3.5.3.1 Регистрация и вход в главное меню

На главном диалоговом окне, рис. 3.1, ввести логин и пароль *admin/admin*. Затем нажать кнопку Login.

Заполнив все поля и нажав на кнопку «Login», пользователю будет выведено сообщение, представленное на рис. 3.2, о удачной регистрации.

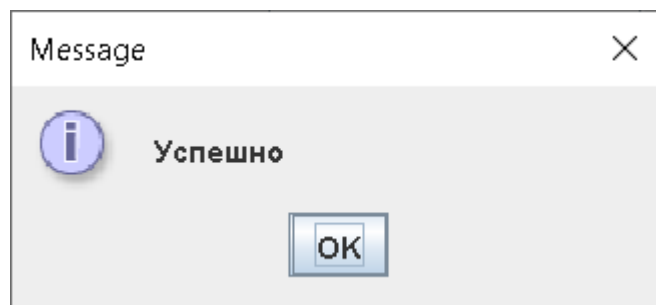


Рис. 3.2 Сообщение об удачной регистрации

После диалоговое окно регистрации закроется, далее выведется главное меню, представленное на рис. 3.3.

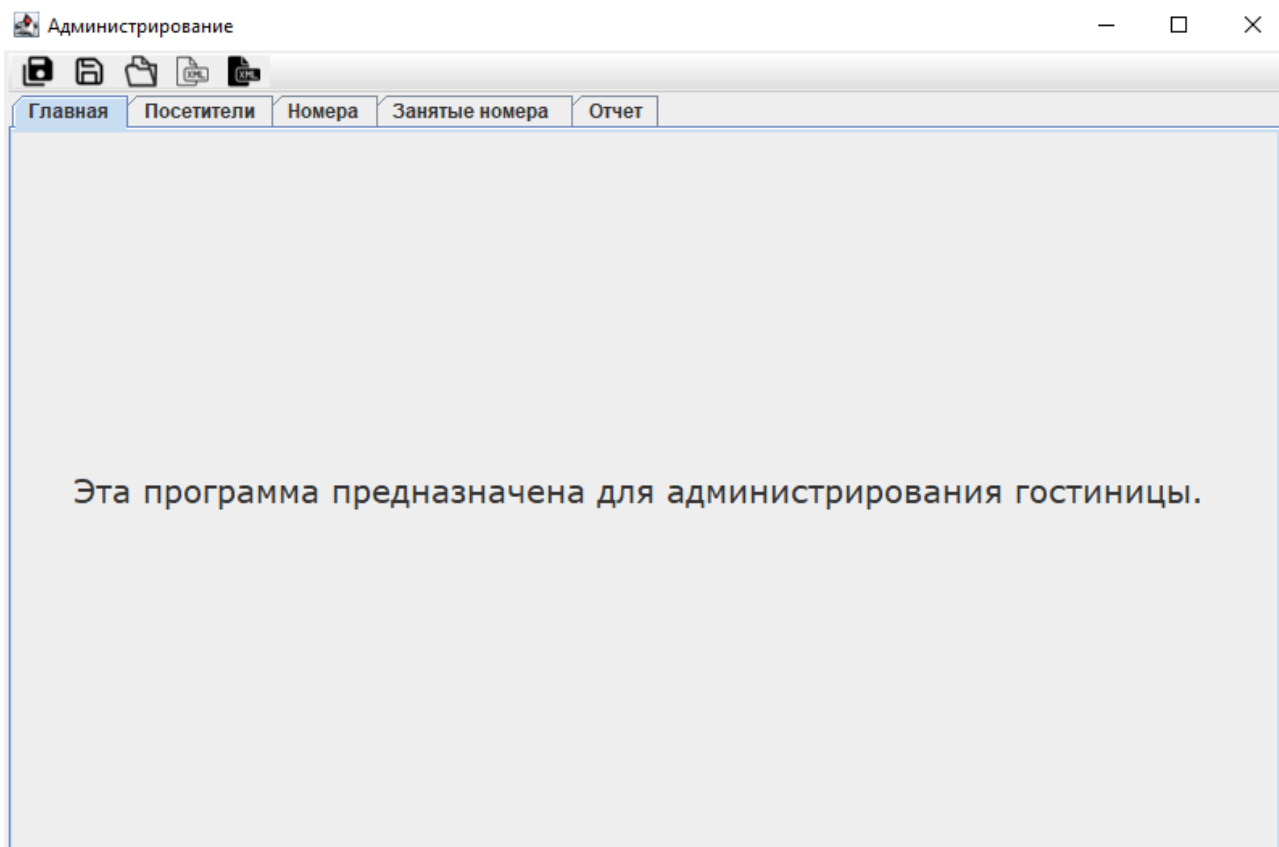


Рис. 3.3 Диалоговое окно главного меню

#### 3.5.3.2 Ввод информации

При открытии диалогового окна программа, сразу автоматически загружаются все данные в таблицу из БД.

#### 3.5.3.2 Ввод информации

При открытии диалогового окна программа, сразу автоматически загружаются все данные в таблицу из БД.

#### 3.5.3.3 Протоколирование

Протоколирование ведётся в консоль, рис. 3.4.

```
сент. 30, 2022 3:19:42 AM Main main
INFO: Start app
сент. 30, 2022 3:19:43 AM Main main
INFO: Finish app
сент. 30, 2022 3:19:52 AM GUI$tabListener stateChanged
INFO: User selected 'Работники'
сент. 30, 2022 3:21:16 AM GUI$tabListener stateChanged
INFO: User selected 'Номера'
сент. 30, 2022 3:21:18 AM GUI$tabListener stateChanged
INFO: User selected 'Занятые номера'
```

Рис.3.4. Протоколирование

#### 3.5.3.4 Редактирование данных в таблицах

Редактирование данных представлено на рисунках 3.5 и 3.6.

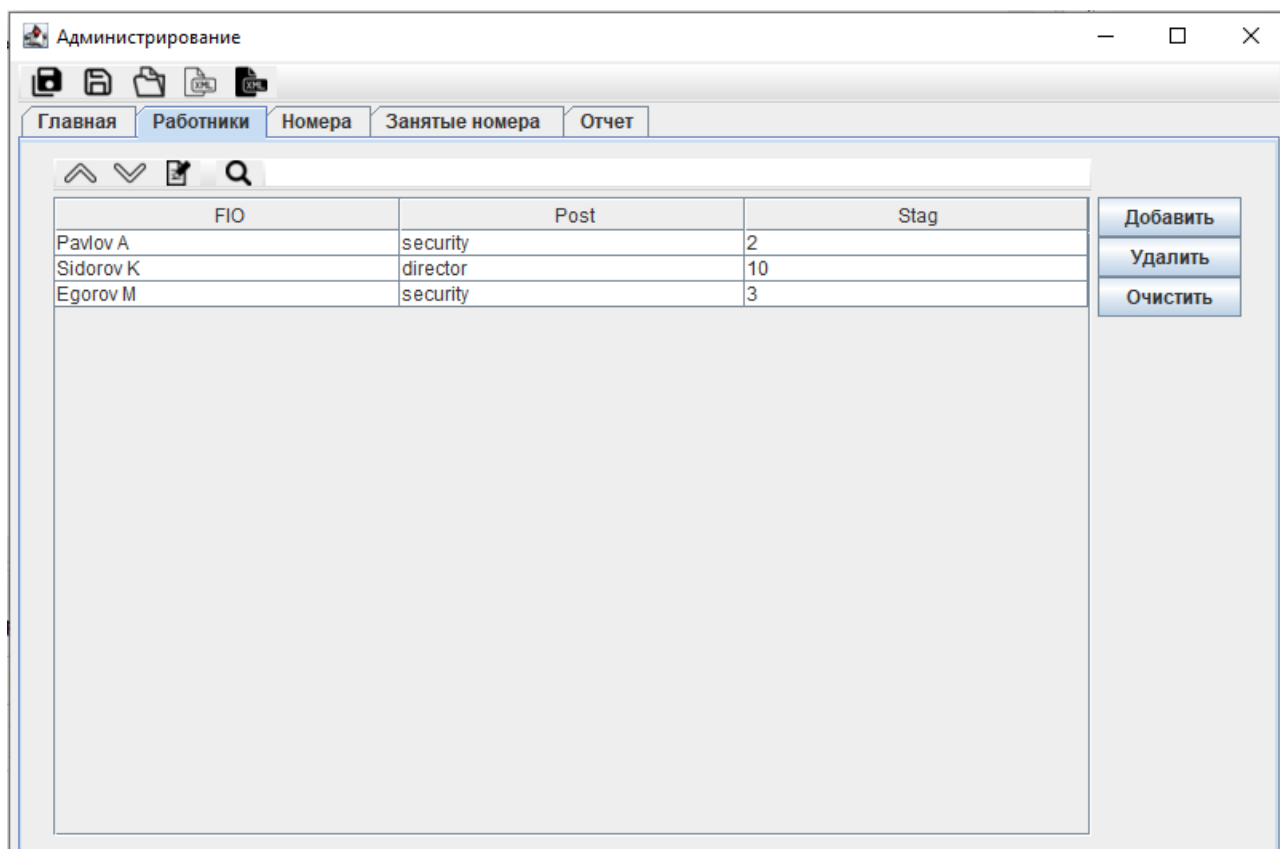


Рис. 3.5 Открытая вкладка с таблицей

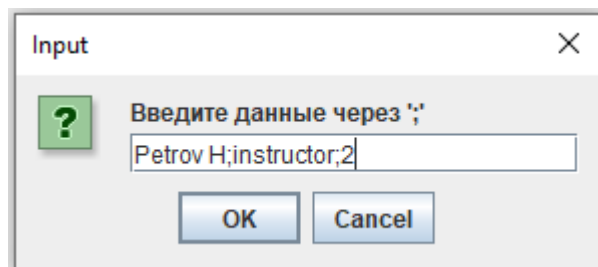


Рис. 3.6 Добавление строки

Если пользователя не заполнил все столбцы, то будет выведено сообщение, представленное на рис. 3.7.

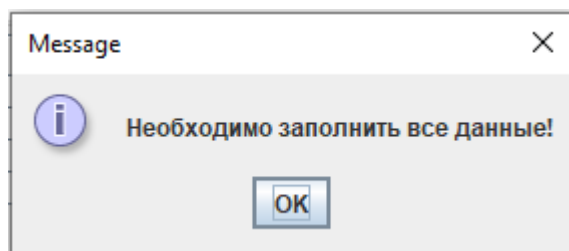


Рис. 3.7 Сообщение об ошибке



В противном случае, строка добавится в таблицу рис. 3.8.

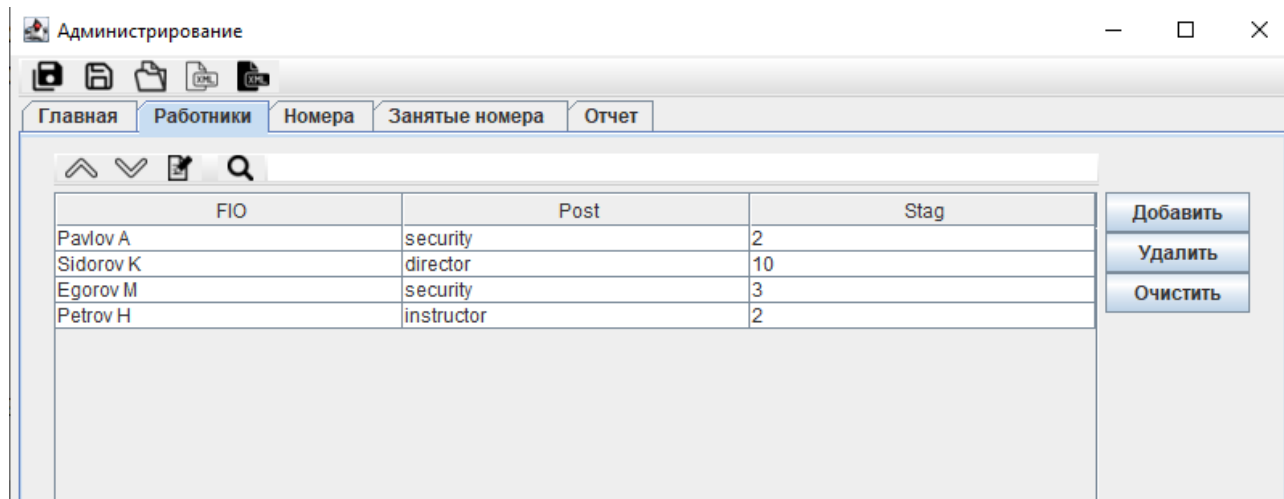


Рис. 3.8 Таблица с добавленной строкой

Чтобы удалить конкретную строку, необходимо нажать на нее, а затем на кнопку «Удалить», Рис. 3.9. или «Очистить», чтобы очистить всю таблицу, Рис.3.10.

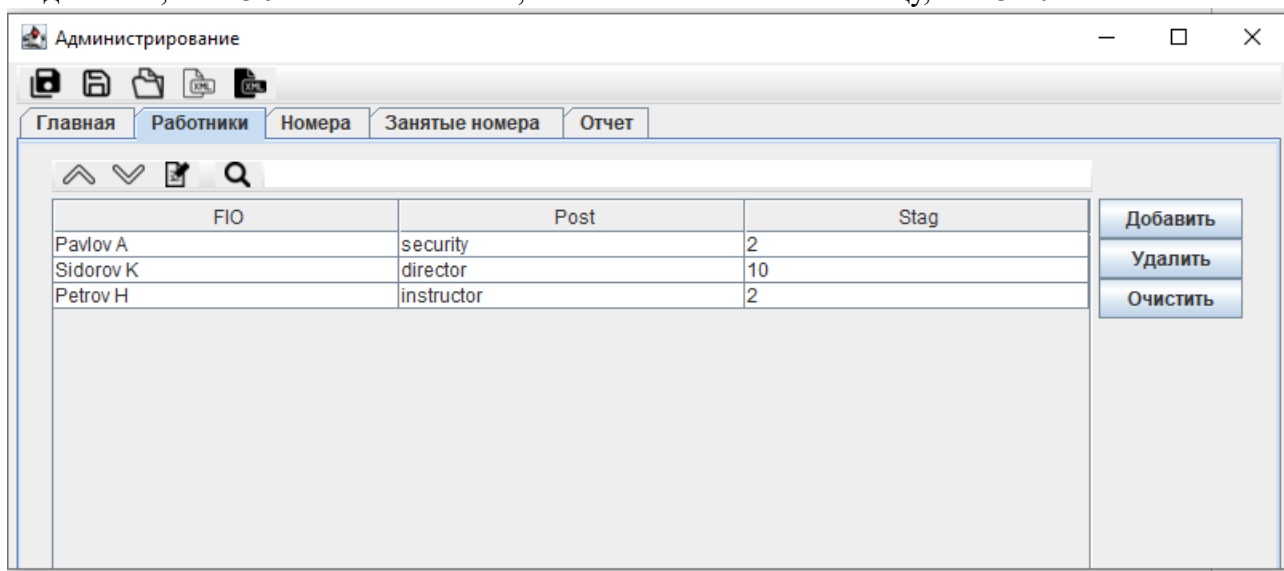


Рис. 3.9 Таблица после удаления строки

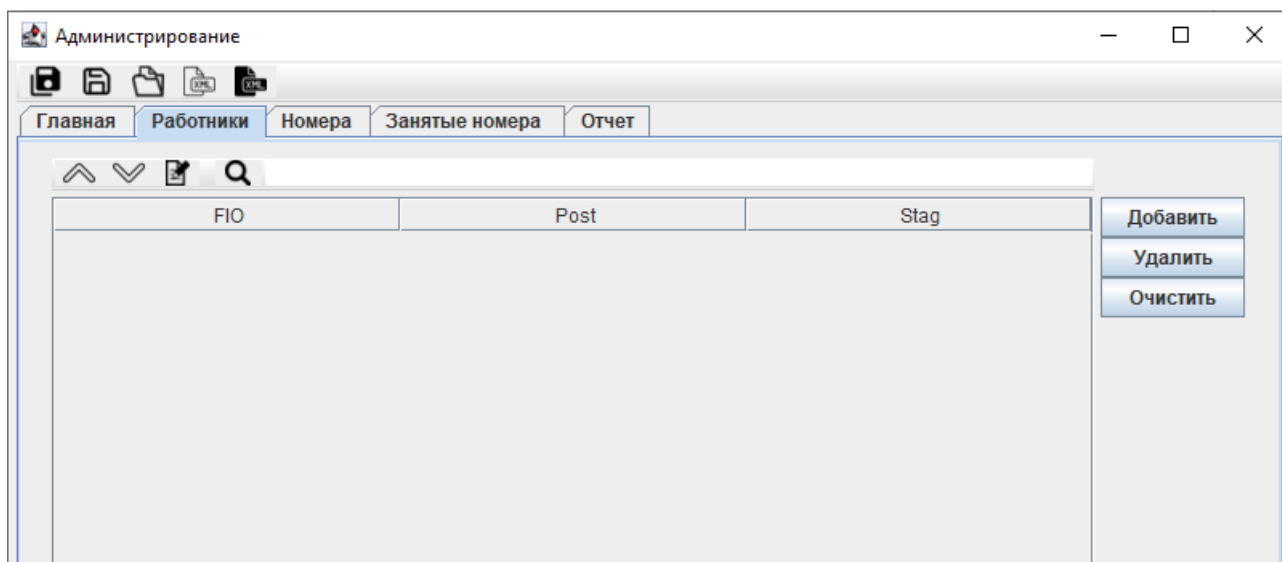


Рис. 3.10 Таблица после очистки

Если ввести в строку поиск ключевые слова и нажать кнопку «Поиск», то в таблице останется только результат поиска рис. 3.11.

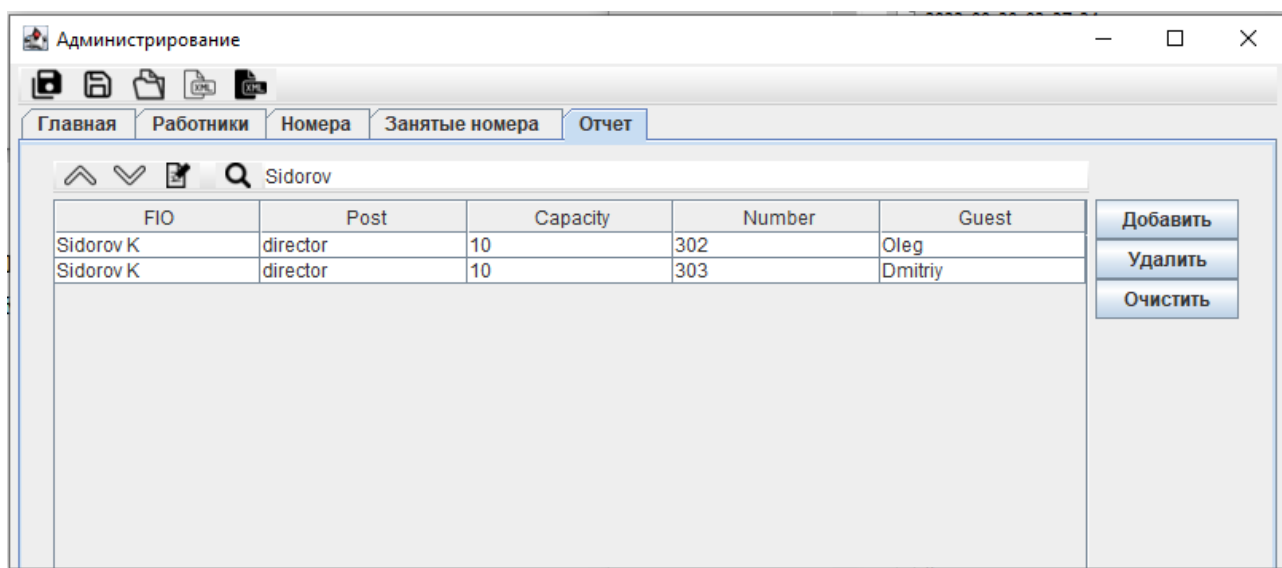


Рис.3.11 Результат поиска

Выбранную строку можно передвинуть вверх или вниз соответствующими кнопками со стрелкой, рис. 3.12.

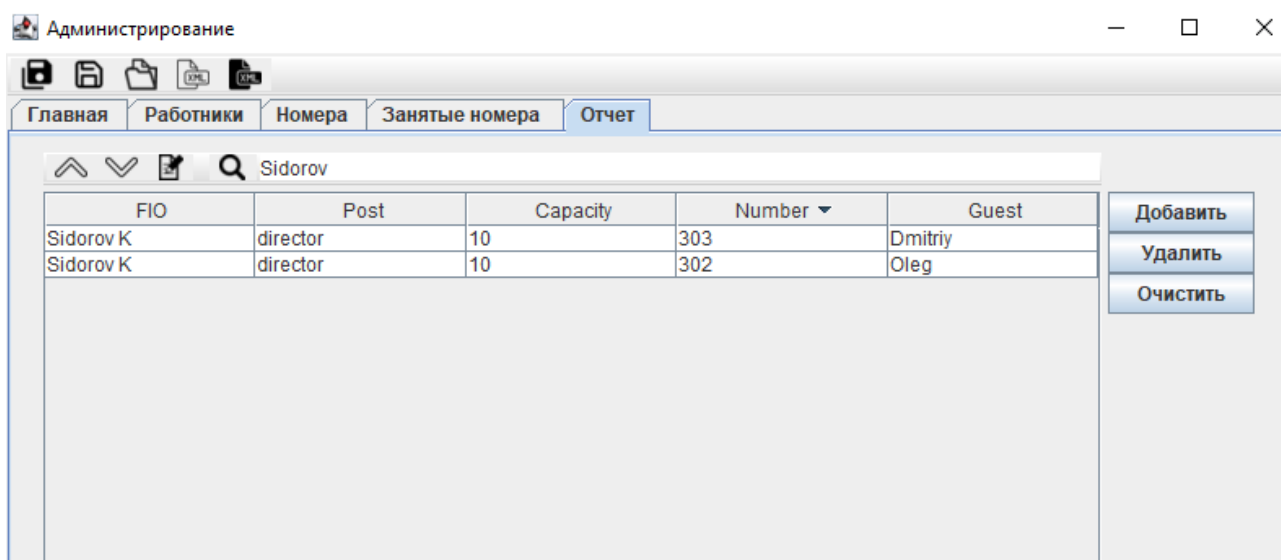


Рис.3.12 Таблица после сдвига первой строки вниз.

#### 3.5.3.4 Сохранение и создание отчетов

При нажатии на кнопку сохранить, можно сохранить текущую таблицу в формате txt, рис. 3.13. или сохранить все, нажав на соответствующую кнопку.

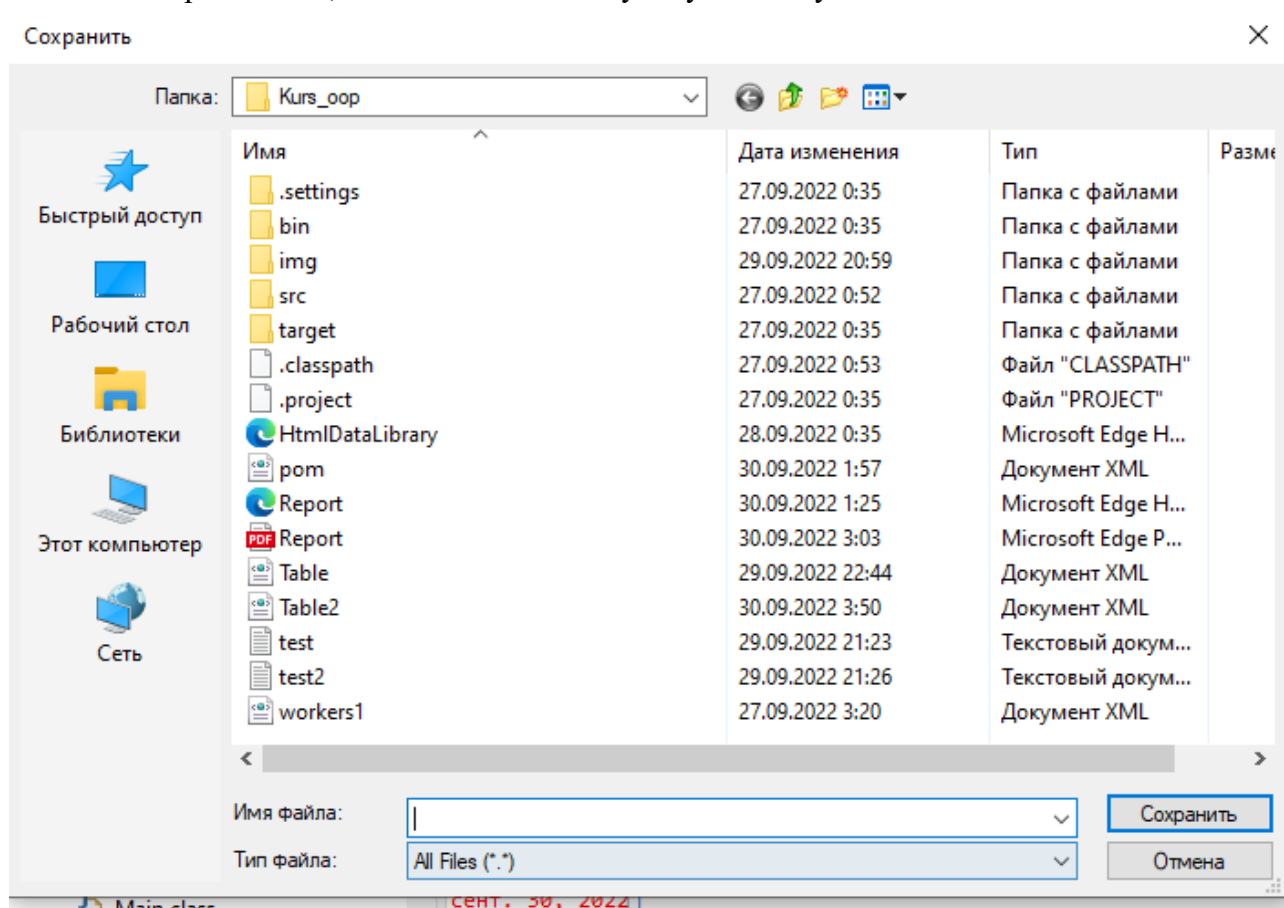


Рис. 3.13 Диалоговое окно «Сохранить»

Также пользователь может создать xml-файл со всеми данными, рис.3.14. или же загрузить данные из него, рис.3.15.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Table>
  - <workers>
    <WORKERS Stag="2" Post="security" FIO="Pavlov A"/>
    <WORKERS Stag="10" Post="director" FIO="Sidorov K"/>
    <WORKERS Stag="3" Post="security" FIO="Egorov M"/>
  </workers>
  - <numbers>
    <NUMBERS Price="1000" Guest="Alexey" Class="301" Capacity="2"/>
    <NUMBERS Price="1500" Guest="Oleg" Class="302" Capacity="3"/>
    <NUMBERS Price="2000" Guest="Dmitriy" Class="303" Capacity="4"/>
  </numbers>
  - <repair>
    <REPAIRS Занято="3"/>
  </repair>
  - <reports>
    <REPORTS Post="security" FIO="Pavlov A" Capacity="2" Number="301"/>
    <REPORTS Post="director" FIO="Sidorov K" Capacity="10" Number="302"/>
    <REPORTS Post="director" FIO="Sidorov K" Capacity="10" Number="303"/>
    <REPORTS Post="security" FIO="Pavlov A" Capacity="2" Number="302"/>
  </reports>
</Table>
```

Рис. 3.14 Сохраненный xml-файл

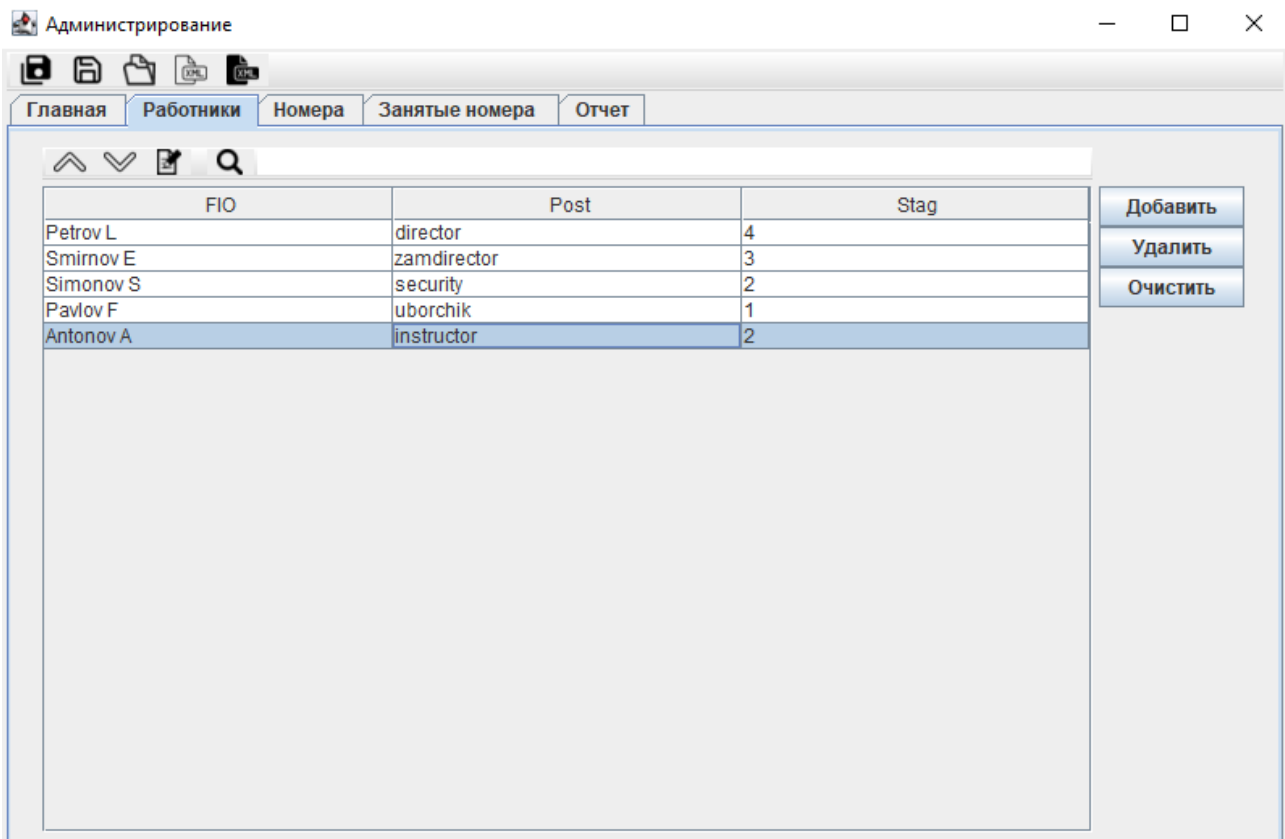


Рис. 3.15 Данные, загруженные из xml-файла

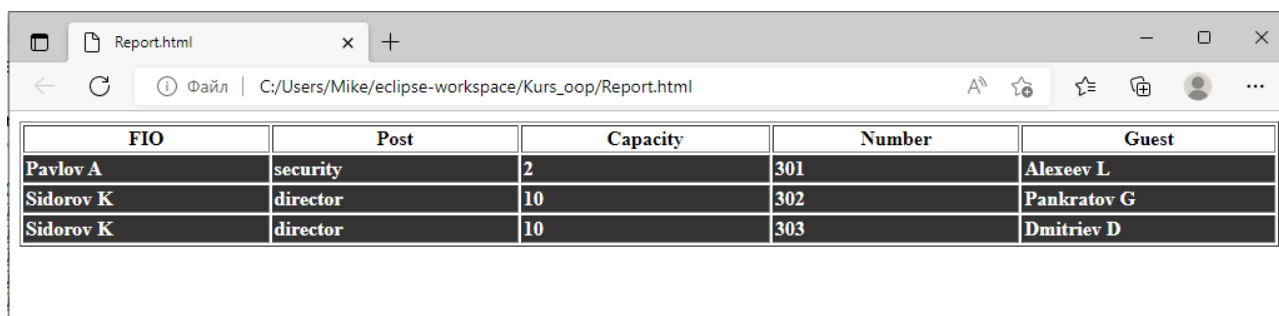
В выбранной вкладке с таблицей, пользователь может выбрать формат отчета, рис.3.16, и создать его, рис.3.17.



Рис. 3.16 Выбор формата отчета

FIO	Post	Capacity	Number	Guest
Pavlov A	security	2	301	Alexey
Sidorov K	director	10	302	Oleg
Sidorov K	director	10	303	Dmitriy

Рис. 3.17 Созданный отчет в формате pdf



FIO	Post	Capacity	Number	Guest
Pavlov A	security	2	301	Alexeev L
Sidorov K	director	10	302	Pankratov G
Sidorov K	director	10	303	Dmitriev D

Рис. 3.18 Созданный отчет в формате html

### 3.5.3.5 Загрузка данных

Пользователь может загрузить данные в таблицу из txt-файла для конкретной таблицы или же для всех сразу, рис.3.19.

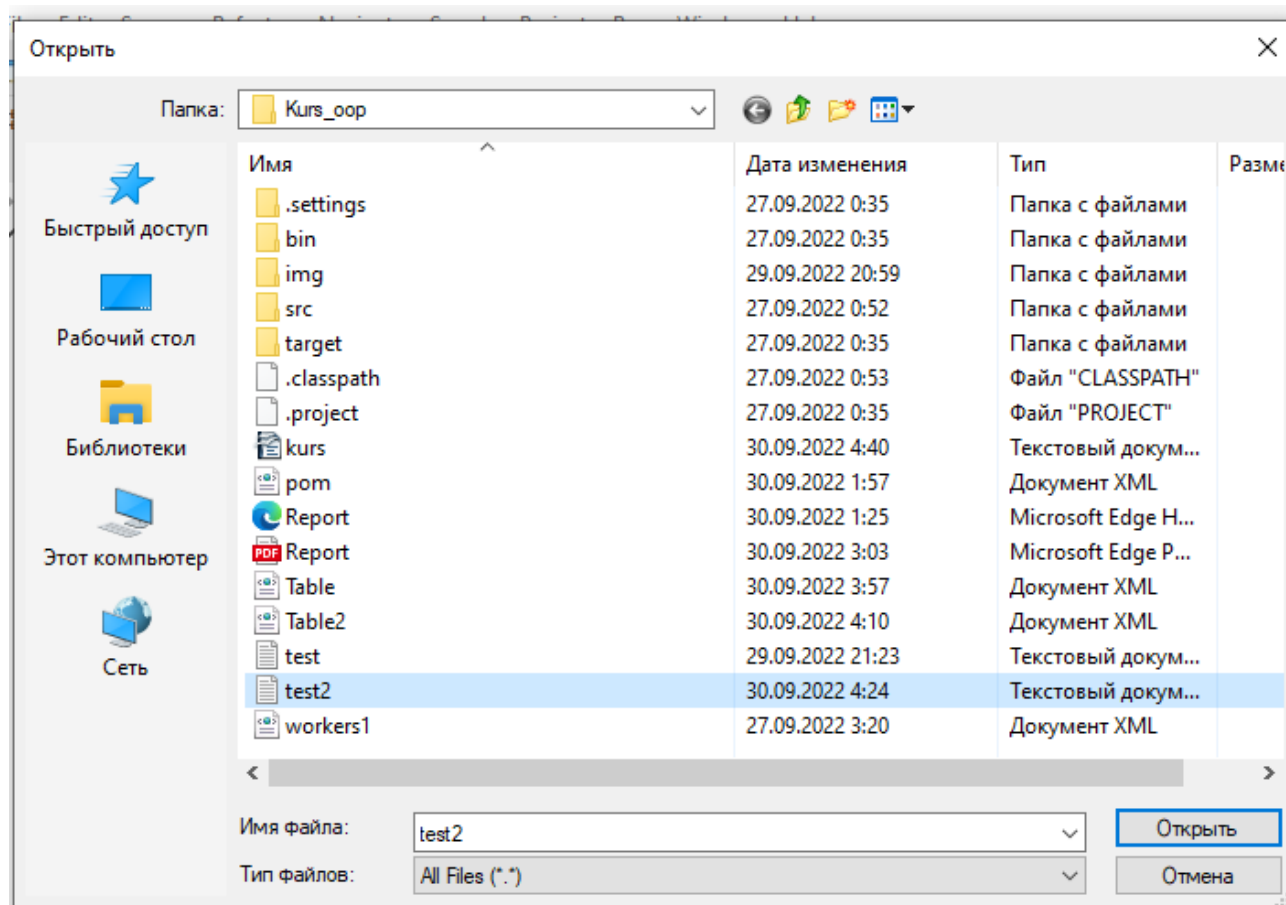
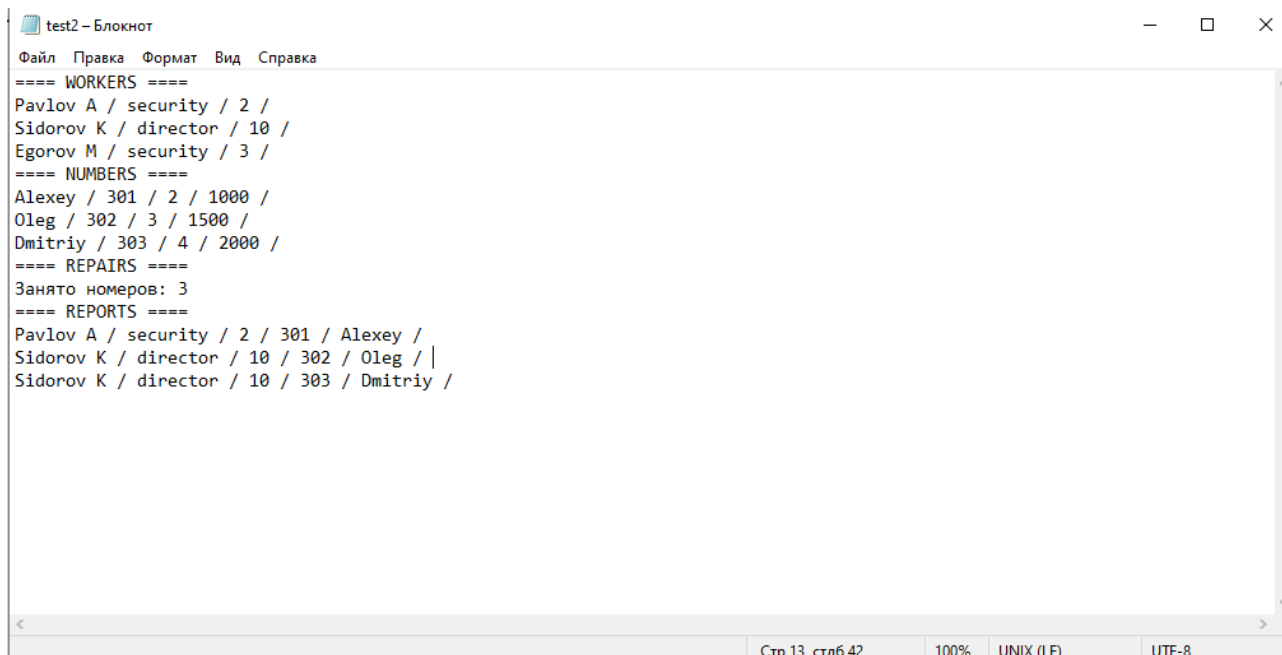


Рис. 3.19 Выбор файла для загрузки

Сохраненные данные в формате txt представлены на рис. 3.20.



```
test2 - Блокнот
Файл  Правка  Формат  Вид  Справка

==== WORKERS ====
Pavlov A / security / 2 /
Sidorov K / director / 10 /
Egorov M / security / 3 /
==== NUMBERS ====
Alexey / 301 / 2 / 1000 /
Oleg / 302 / 3 / 1500 /
Dmitriy / 303 / 4 / 2000 /
==== REPAIRS ====
Занято номеров: 3
==== REPORTS ====
Pavlov A / security / 2 / 301 / Alexey /
Sidorov K / director / 10 / 302 / Oleg / |
Sidorov K / director / 10 / 303 / Dmitriy /

Стр 13, столб 42  100%  UNIX (LF)  UTF-8
```

Рис. 3.20 Файл .txt

Автоматическое сохранение данных из таблицы в БД представлено на рисунке 3.21

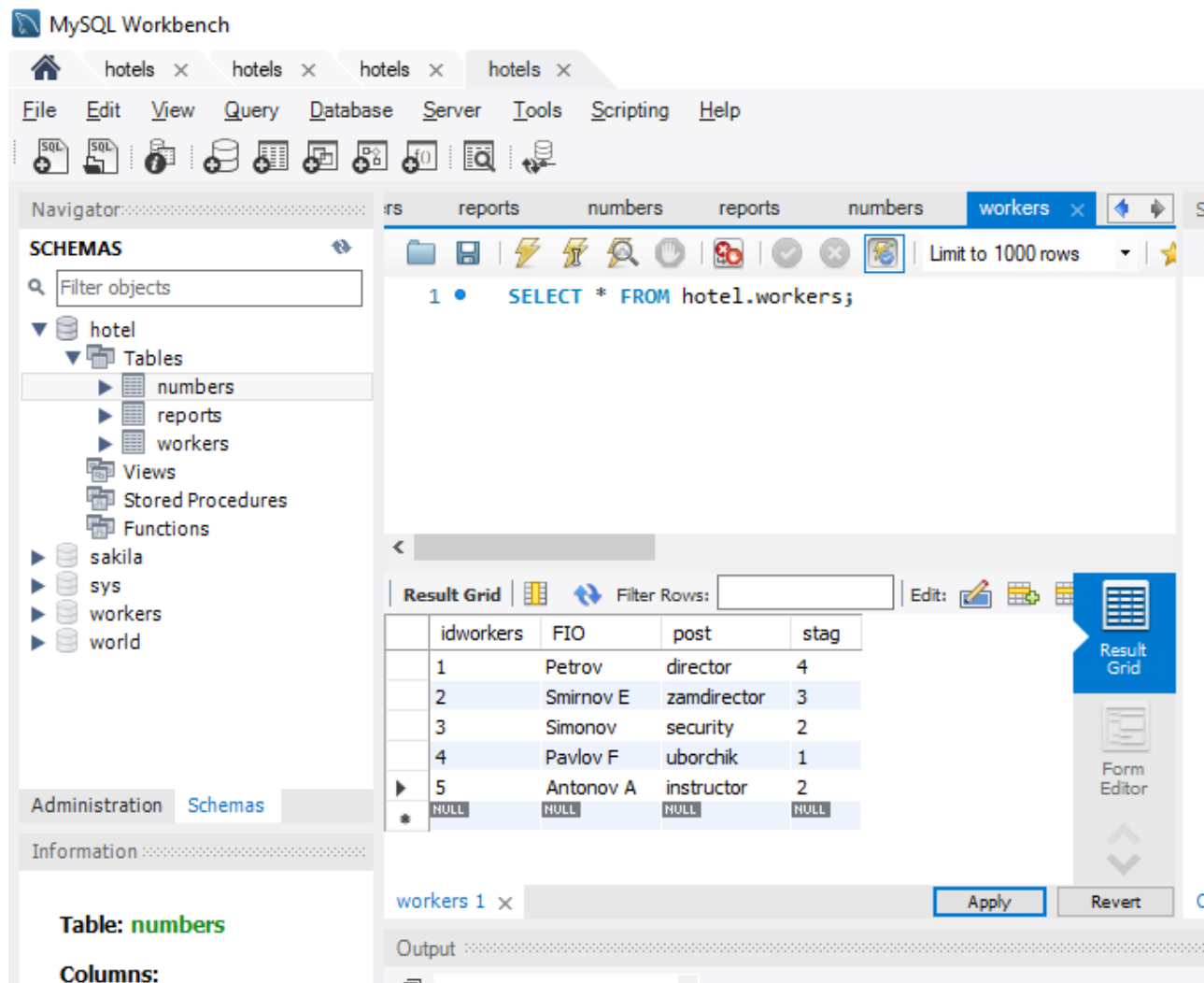


Рис. 3.21 База Данных



Связь «все-со-всеми» - составление отчета на основе двух таблиц представлена на рисунке 3.22

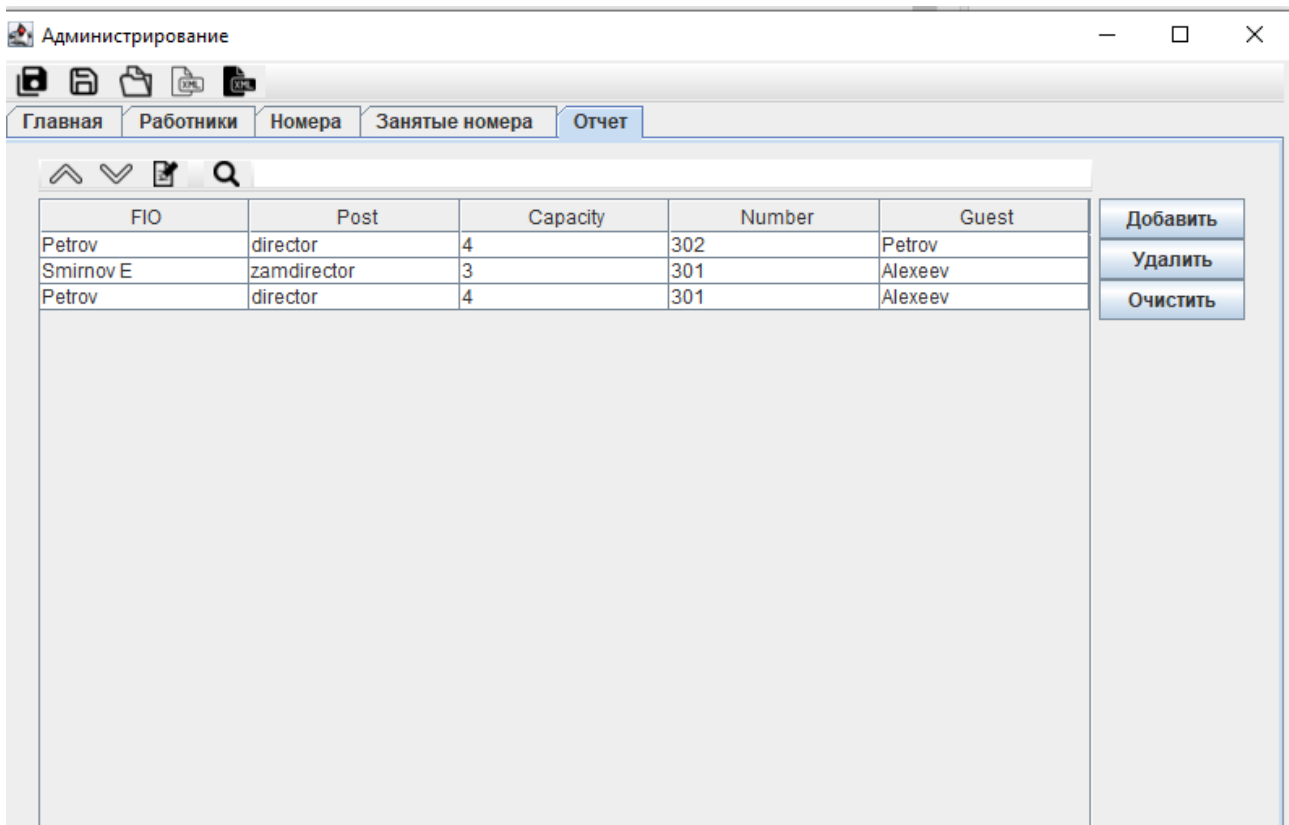


Рис. 3.22 связь «все-со-всеми»

## 4. Исходные тексты ПК

### 4.1. Main

```
import java.util.logging.Logger;
import javax.swing.JFrame;

/**
 * main
 * @author Semenov Mikhail 0306
 */

public class Main {

    public static final Logger log = Logger.getLogger(Main.class.getName());

    /**
     * @param args - вводимая строка
     */
    public static void main(String[] args)
    {
        log.info("Start app");
        Registration frame = new Registration();
        frame.setTitle("Авторизация");
        frame.setVisible(true);
        frame.setBounds(600, 300, 370, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```

        frame.setResizable(false);
        log.info("Finish app");
    }
}

```

## 4.2 Registration

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Registration extends JFrame implements ActionListener {
    Container container = getContentPane();
    JLabel userLabel = new JLabel("USERNAME");
    JLabel passwordLabel = new JLabel("PASSWORD");
    JTextField userTextField = new JTextField();
    JPasswordField passwordField = new JPasswordField();
    JButton loginButton = new JButton("LOGIN");
    JButton resetButton = new JButton("RESET");
    JCheckBox showPassword = new JCheckBox("Show Password");

    Registration() {
        setLayoutManager();
        setLocationAndSize();
        addComponentsToContainer();
        addActionEvent();
    }

    public void setLayoutManager() {
        container.setLayout(null);
    }

    public void setLocationAndSize() {
        userLabel.setBounds(50, 50, 100, 25);
        passwordLabel.setBounds(50, 120, 100, 25);
        userTextField.setBounds(150, 50, 150, 25);
        passwordField.setBounds(150, 120, 150, 25);
        showPassword.setBounds(150, 150, 150, 30);
        loginButton.setBounds(50, 200, 100, 25);
        resetButton.setBounds(200, 200, 100, 25);
    }

    public void addComponentsToContainer() {
        container.add(userLabel);
        container.add(passwordLabel);
        container.add(userTextField);
        container.add(passwordField);
        container.add(showPassword);
        container.add(loginButton);
        container.add(resetButton);
    }

    public void addActionEvent() {
        loginButton.addActionListener(this);
        resetButton.addActionListener(this);
        showPassword.addActionListener(this);
    }
}

```

```

    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loginButton) {
            String userText;
            String pwdText;
            userText = userTextField.getText();
            pwdText = passwordField.getText();
            if (userText.equalsIgnoreCase("Admin") &&
pwdText.equalsIgnoreCase("Admin")) {
                JOptionPane.showMessageDialog(this, "Успешно");
                this.setVisible(false);
                this.dispose();
                GUI app = new GUI();
                app.show();
            } else {
                JOptionPane.showMessageDialog(this, "Неверное Имя пользователя или
Пароль!");
            }

        }
        if (e.getSource() == resetButton) {
            userTextField.setText("");
            passwordField.setText("");
        }
        if (e.getSource() == showPassword) {
            if (showPassword.isSelected()) {
                passwordField.setEchoChar((char) 0);
            } else {
                passwordField.setEchoChar('*');
            }
        }
    }
}
}
}

```

## 4.3 GUI

```

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import java.util.logging.Logger;

/**
 * Класс описания интерфейса
 * @author Semenov Mike 0306
 * @version 1.00
 */

public class GUI{
    private JFrame frame;
    private JTabbedPane tabs;

```

```

private JToolBar bar;
private JPanel topBox;
private JPanel main;
private JPanel workers;
private JPanel numbers;
private JPanel repair;
private JPanel reports;
private JButton saveAll;
private JButton save;
private JButton open;
private JButton createXML;
private JButton readXML;
private Tabs workersTab;
private Tabs numbersTab;
private Tabs repairTab;
private Tabs reportsTab;

public static final Logger LogGUI = Logger.getLogger(GUI.class.getName());

public void show() {
    // Инициализация фрейма
    frame = new JFrame("Администрирование");
    frame.setBounds(400, 200, 800, 500);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Инициализация панелей
    main = new JPanel ();
    workers = new JPanel ();
    numbers = new JPanel ();
    repair = new JPanel ();
    reports = new JPanel ();
    saveAll = new JButton(new ImageIcon("./img/save_all.png"));
    saveAll.setPreferredSize(new Dimension(20, 20));
    save = new JButton(new ImageIcon("./img/save.png"));
    save.setPreferredSize(new Dimension(20,20));

    open = new JButton(new ImageIcon("./img/open.png"));
    open.setPreferredSize(new Dimension(20, 20));
    createXML = new JButton(new ImageIcon("./img/xml.png"));
    createXML.setPreferredSize(new Dimension(20, 20));
    readXML = new JButton(new ImageIcon("./img/xmlc.png"));
    readXML.setPreferredSize(new Dimension(20, 20));

    // Добавление панелей во вкладки
    tabs = new JTabbedPane();
    tabs.addTab("Главная", main);
    tabs.addTab("Работники", workers);
    tabs.addTab("Номера", numbers);
    tabs.addTab("Занятые номера ", repair);
    tabs.addTab("Отчет", reports);
    tabs.addChangeListener(new TabListener());
    tabs.setAlignmentX(Component.LEFT_ALIGNMENT);

    // Настройка кнопок тулбара
    createXML.setIconTextGap (0);
    createXML.setBorderPainted(false);
    createXML.setFocusPainted(false);
    createXML.setBorderPainted (false);

```

```

        createXML.addActionListener (new XMLListenerW());
        readXML.setIconTextGap (0);
        readXML.setBorderPainted(false);
        readXML.setFocusPainted(false);
        readXML.setBorderPainted (false);
        readXML.addActionListener (new XMLListenerR());
        open.setIconTextGap (0);
        open.setBorderPainted(false);
        open.setFocusPainted(false);
        open.setBorderPainted (false);
        open.addActionListener (new ImportListener());
        save.setIconTextGap (0);
        save.setBorderPainted(false);
        save.setFocusPainted(false);
        save.setBorderPainted (false);
        save.addActionListener (new SaveListener());
        saveAll.setIconTextGap (0);
        saveAll.setBorderPainted(false);
        saveAll.setFocusPainted(false);
        saveAll.setBorderPainted (false);
        saveAll.addActionListener (new SaveAllListener());

// Добавление элементов в тулбар и топбокс
bar = new JToolBar();
bar.setFloatable(false); //Отключение перемещения тулбара
    bar.add(saveAll);
    bar.add(save);
    bar.add(open);
    bar.add(createXML);
    bar.add(readXML);
    bar.setMaximumSize(new Dimension(800, 40));
bar.setAlignmentX(Component.LEFT_ALIGNMENT);

//Настройка main
main.setLayout(new GridBagLayout());
JLabel Hello = new JLabel("Эта программа предназначена для
администрирования"
        + " гостиницы. ");
Font fonthello = new Font("Verdana", Font.PLAIN, 20);
Hello.setFont(fonthello);
main.add(Hello, new GridBagConstraints(0, 0, 1, 1, 0, 0,
        GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL,
        new Insets(2, 2, 2, 2), 0, 0));

// Настройка workers
String[] columnsW = {"FIO", "Post", "Stag"};
String[][] dataW = {{}};

workers.setLayout(new GridBagLayout());
workersTab = new Tabs(dataW, columnsW, "==== WORKERS ===");
workersTab.setTab(workers, 0, 0, 0, 1, 1, 1, 17, 17, 11, 3, 2, 3, true);

//Настройка numbers
String[] columnsC = {"Guest", "Class", "Capacity", "Price"};
String[][] dataC = {{}};

numbers.setLayout(new GridBagLayout());
numbersTab = new Tabs(dataC, columnsC, "==== NUMBERS ===");
numbersTab.setTab(numbers, 0, 0, 0, 1, 1, 1, 17, 17, 11, 3, 2, 3, true);

```

```

//Настройка repair
repair.setLayout(new GridBagLayout());

String text = "Занято: ";
Font font = new Font("Verdana", Font.BOLD, 25);
repairTab = new Tabs(text, font, "==== REPAIRS ====");
repairTab.setTab(repair, 0, 0, 1, 0, 2, 0, 17, 17, 11, 3, 2, 3, false);

//Настройка reports
String[] columnsR = {"FIO", "Post", "Capacity", "Number", "Guest"};
String[][] dataR = {{}};

reports.setLayout(new GridBagLayout());
reportsTab = new Tabs(dataR, columnsR, "==== REPORTS ====");
reportsTab.setTab(reports, 0, 0, 0, 1, 1, 1, 17, 17, 11, 3, 2, 3, true);

Tabs[] arrayTabs = {workersTab, numbersTab, repairTab, reportsTab};

DataBase dbHandler = new DataBase();
ResultSet result,result2;
try {
    result = dbHandler.getWorkers();
    while(result.next()) {
        String FIO = result.getString(2);
        String post = result.getString(3);
        String document = result.getString(4);
        String[] data = {FIO, post, document};
        workersTab.getModel().addRow(data);
    }
    result = dbHandler.getNumbers();
    int tmp = 0;
    while(result.next()) {
        String guest = result.getString(2);
        String number = result.getString(3);
        String capacity = result.getString(4);
        String price = result.getString(5);
        String[] data = {guest, number, capacity, price};
        numbersTab.getModel().addRow(data);
        tmp++;
    }

    result2 = dbHandler.getReports();
    int r ;

    while(result2.next()) {
        int i = 1;
        int j = 1;
        int idworker = result2.getInt(2);
        int idnumber = result2.getInt(3);

        String FIO = "";
        String post = "";
        String stag = "" ;

        result = dbHandler.getWorkers();
        while(i<idworker+1) {
            result.next();

```

```

        FIO = result.getString(2);
        post = result.getString(3);
        stag = result.getString(4);
        i++;
    }

    String guest = "";
    String number = "";
    result = null;
    result = dbHandler.getNumbers();
    while(j<idnumber+1) {
        result.next();
        j++;
        guest = result.getString(2);
        number = result.getString(3);
    }

    String[] data = {FIO,post,stag,number,guest};
    reportsTab.getModel().addRow(data);
}

} catch (ClassNotFoundException e) {
    // TODO Автоматически созданный блок catch
    e.printStackTrace();
} catch (SQLException e) {
    // TODO Автоматически созданный блок catch
    e.printStackTrace();
}

// Настройка топбокса
topBox = new JPanel();
topBox.setPreferredSize(new Dimension(800, 500));
topBox.setLayout(new BoxLayout(topBox, BoxLayout.Y_AXIS));
topBox.setBorder(BorderFactory.createEmptyBorder(2, 5, 0, 5));
topBox.add(bar);
topBox.add(tabs);

// Добавление на фрейм
frame.add(topBox, BorderLayout.NORTH);
frame.pack();
frame.setVisible(true);
}

class tabListener implements ChangeListener {
    public void stateChanged(ChangeEvent e) {
        LogGUI.info("User selected '" +
tabs.getTitleAt(tabs.getSelectedIndex()) + "'");
    }
}

class XMLListenerR implements ActionListener {
    public void actionPerformed (ActionEvent event) {
        if (tabs.getSelectedIndex() != 0) {
            Tabs[] arrayTabs = {workersTab, numbersTab, repairTab,
reportsTab};

            xmlReader xmlDoc = new xmlReader(arrayTabs);
            xmlDoc = null;
        }
    }
}

```

```

    }

    class XMLListenerW implements ActionListener {
        public void actionPerformed (ActionEvent event) {
            if (tabs.getSelectedIndex() != 0) {
                Tabs[] arrayTabs = {workersTab, numbersTab, repairTab,
reportsTab};

                xmlCreator xmlDoc = new xmlCreator(arrayTabs);
                xmlDoc = null;
            }
        }
    }

    class ImportListener implements ActionListener {
        public void actionPerformed (ActionEvent event) {
            if (tabs.getSelectedIndex() != 0) {
                Tabs[] arrayTabs = {workersTab, numbersTab, repairTab,
reportsTab};

                SaveLoad importF = new SaveLoad(arrayTabs, tabs, "Открыть",
0, frame);

                importF = null;
            }
        }
    }

    class SaveListener implements ActionListener {
        public void actionPerformed (ActionEvent event) {
            /*System.out.println("Поток начало");
            Tabs[] arrayTabs = {workersTab, carsTab, repairTab, reportsTab};
            Object shared = new Object();
            new ThreadThree(1, workersTab.getModel(), arrayTabs,
shared).start();
            new ThreadThree(2, workersTab.getModel(), arrayTabs,
shared).start();
            new ThreadThree(3, workersTab.getModel(), arrayTabs,
shared).start();
            System.out.println("Поток конец");*/

            //TableRat r = new TableRat(carsTab.getModel(),
reportsTab.getModel());
            if (tabs.getSelectedIndex() != 0) {
                Tabs[] arrayTabs = {workersTab, numbersTab, repairTab,
reportsTab};

                SaveLoad SaveF = new SaveLoad(arrayTabs, tabs, "Сохранить",
1, frame);

                SaveF = null;
            }
        }
    }

    class SaveAllListener implements ActionListener {
        public void actionPerformed (ActionEvent event) {
            Tabs[] arrayTabs = {workersTab, numbersTab, repairTab, reportsTab};
            SaveLoad SaveAllF = new SaveLoad(arrayTabs, tabs, "Сохранить все",
1, frame);

            SaveAllF = null;
        }
    }
}

```



## 4.4 Tabs

```
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Logger;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JToolBar;
import javax.swing.RowFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.net.URL;
import org.apache.logging.log4j.*;
import org.xml.sax.SAXException;
import java.util.logging.Logger;

public class Tabs{
    private JTable table;
    private JScrollPane scroll;
    private DefaultTableModel model;
    private JToolBar bar;
    private JButton up;
    private JButton down;
    private JButton add;
    private JButton delete;
    private JButton plus;
    private JButton minus;
    private JButton search;
    private JButton deleteAll;
    private JButton pdf;
    private JButton html;
    private JButton doc;
    private JDialog rep;
    private JPanel AD;
    private JPanel repPanel;
    private JTextArea searchFilter;
    private JTextArea out;
    private JLabel label;
    private String name;

    public static final Logger LogTabs = Logger.getLogger(Tabs.class.getName());
```

```

Tabs(String[][] data, String[] columns, String nameTab){
    //Описание таблицы
    model = new DefaultTableModel(data, columns);
    if(data[0].length == 0) model.setRowCount(0);
    table = new JTable(model);
    table.setAutoCreateRowSorter(true);
    scroll = new JScrollPane(table);
    scroll.setPreferredSize(new Dimension(650, 400));
    name = nameTab;

    //Инициализация кнопок и бара
    bar = new JToolBar();
    add = new JButton("Добавить");
    delete = new JButton("Удалить");
    deleteAll = new JButton("Очистить");
    up = new JButton(new ImageIcon("./img/up.png"));
    down = new JButton(new ImageIcon("./img/down.png"));
    search = new JButton(new ImageIcon("./img/search.png"));
    searchFilter = new JTextArea();
    AD = new JPanel();
    pdf = new JButton(new ImageIcon("./img/pdf.png"));
    pdf.setPreferredSize(new Dimension(50, 50));
    html = new JButton(new ImageIcon("./img/html.png"));
    html.setPreferredSize(new Dimension(50, 50));
    doc = new JButton(new ImageIcon("./img/doc.png"));
    doc.setPreferredSize(new Dimension(30, 16));
    rep = new JDialog();
    rep.setLayout(new GridBagLayout());
    rep.setBounds(600, 400, 200, 100);
    repPanel = new JPanel();
    repPanel.setLayout(new BoxLayout(repPanel, BoxLayout.X_AXIS));

    repPanel.add(Box.createHorizontalStrut(5));
    repPanel.add(pdf);
    repPanel.add(Box.createHorizontalStrut(5));
    repPanel.add(html);

    //Размеры
    doc.setIconTextGap(0);
    doc.setBorderPainted(false);
    doc.setFocusPainted(false);
    doc.setBorderPainted(false);
    doc.addActionListener(new DocListener());

    pdf.setFocusPainted(false);
    pdf.setContentAreaFilled(false);
    html.setFocusPainted(false);
    html.setContentAreaFilled(false);
    add.setMaximumSize(new Dimension(100, 25));
    add.setFocusPainted(false);
    delete.setMaximumSize(new Dimension(100, 25));
    delete.setFocusPainted(false);
    deleteAll.setMaximumSize(new Dimension(100, 25));
    deleteAll.setFocusPainted(false);
    up.setMaximumSize(new Dimension(30, 16));
    up.setBorderPainted(false);
    up.setFocusPainted(false);
    up.setBorderPainted(false);
    down.setMaximumSize(new Dimension(30, 16));
    down.setBorderPainted(false);
    down.setFocusPainted(false);
}

```

```

down.setBorderPainted(false);
search.setMaximumSize(new Dimension(30, 16));
search.setBorderPainted(false);
search.setFocusPainted(false);
search.setBorderPainted(false);
searchFilter.setMaximumSize(new Dimension(560, 25));
bar.setPreferredSize(new Dimension(650, 20));
bar.setAlignmentX(Component.LEFT_ALIGNMENT);
bar.setFloatable(false);
AD.setLayout(new BorderLayout(AD, BorderLayout.Y_AXIS));

//Добавление элементов в бар и панель
bar.add(up);
bar.add(down);
bar.add(doc);
bar.add(Box.createHorizontalStrut(10));
bar.add(search);
bar.add(Box.createHorizontalStrut(2));
bar.add(searchFilter);
AD.add(add);
AD.add(delete);
AD.add(deleteAll);
rep.add(repPanel, new GridBagConstraints(0, 0, 1, 1, 0, 0,
GridBagConstraints.CENTER, GridBagConstraints.NONE,
new Insets(2, 2, 2, 2), 0, 0));

//Добавление слушателей
add.addActionListener (new AddListener());
delete.addActionListener (new DeleteListener());
deleteAll.addActionListener (new DeleteAllListener());
up.addActionListener (new UpListener());
down.addActionListener (new DownListener());
search.addActionListener (new SearchListener());
pdf.addActionListener (new PdfListener());
html.addActionListener(new HtmlListener());
}

private static Logger Logger(Class<Tabs> class1) {
    // TODO Auto-generated method stub
    return null;
}

Tabs(String text, Font font, String nameTab){
    //Инициализация кнопок
    label = new JLabel(text);
    AD = new JPanel();
    plus = new JButton(new ImageIcon("./img/plus.png"));
    minus = new JButton(new ImageIcon("./img/minus.png"));
    out = new JTextArea("5");
    name = nameTab;

    //Размеры
    AD.setLayout(new BorderLayout(AD, BorderLayout.Y_AXIS));
    plus.setPreferredSize(new Dimension(30, 30));
    plus.setIconTextGap(0);
    plus.setBorderPainted(false);
    plus.setContentAreaFilled(false);
    minus.setPreferredSize(new Dimension(30, 30));
    minus.setIconTextGap (0);
    minus.setBorderPainted(false);
    minus.setContentAreaFilled(false);
    out.setPreferredSize(new Dimension(40, 30));

```

```

        label.setFont(font);
        out.setFont(font);
        out.setEditable(false);
        out.setOpaque(false);

        //Добавление элементов в панель
        AD.add(plus);
        AD.add(Box.createVerticalStrut(10));
        AD.add(minus);

        plus.addActionListener (new PlusListener());
        minus.addActionListener (new MinusListener());
    }

    Tabs() {
        model = new DefaultTableModel();
        name = new String("Test");
    }
    public void filter(String txtArea) {
        TableRowSorter<DefaultTableModel> f = new
TableRowSorter<DefaultTableModel>(model);
        table.setRowSorter(f);
        f.setRowFilter(RowFilter.regexFilter(txtArea));
    }

    public DefaultTableModel getModel() {
        return model;
    }

    public String getName() {
        return name;
    }

    public void setName(String nameTab) {
        name = nameTab;
    }

    public String getCount() {
        return out.getText();
    }

    public void setModel(DefaultTableModel modelTab) {
        model = modelTab;
    }

    public void setCount(String countStr) {
        out.setText(countStr);
    }

    class PdfListener implements ActionListener {
        public void actionPerformed (ActionEvent event) {
            reportsCreator ad = new reportsCreator(model);
        }
    }

    class HtmlListener implements ActionListener {
        public void actionPerformed (ActionEvent event) {
            reportsCreator ad = new reportsCreator(model, 1);
        }
    }
}

```

```

class DocListener implements ActionListener {
    public void actionPerformed (ActionEvent event) {
        rep.setVisible(true);
    }
}

class AddListener implements ActionListener {
    public void actionPerformed (ActionEvent event) {
        String rowDataString;
        String[] rowDataArray;
        rowDataString = JOptionPane.showInputDialog ("Введите данные через
';");

        if (rowDataString != null) {
            if (rowDataString.length() != 0 ) {
                rowDataArray = rowDataString.split("; |");
                if ((rowDataArray.length ==
model.getColumnCount()) || /*((model.getColumnCount() == 5))*/(rowDataArray.length ==2))
{
                    DataBase dbHandler = new DataBase();
                    try {
                        if (model.getColumnCount() == 3)

dbHandler.addWorkers(rowDataArray[0], rowDataArray[1], rowDataArray[2]);
                        else if (model.getColumnCount() == 4)

dbHandler.addNumbers(rowDataArray[0], rowDataArray[1], rowDataArray[2],
rowDataArray[3]);

                        else if (model.getColumnCount() == 5){
                            ResultSet result1;
                            ResultSet result2;
                            result1 = dbHandler.getWorkers();
                            result2 = dbHandler.getNumbers();
                            for (int i = 0; i <
Integer.parseInt(rowDataArray[0]); i++) {
                                result1.next();
                            }
                            for (int i = 0; i <
Integer.parseInt(rowDataArray[3]); i++) {
                                result2.next();
                            }
                            String newRow[] =
{result1.getString(2), result1.getString(3), result1.getString(4),
                                result2.getString(3),
                                result2.getString(2)};

                                model.addRow(newRow);

                            dbHandler.addReports(result1.getInt(1),result2.getInt(1));
                        }
                    } catch (ClassNotFoundException e) {
                        // TODO Автоматически созданный блок catch
                        e.printStackTrace();
                    } catch (SQLException e) {
                        // TODO Автоматически созданный блок catch
                        e.printStackTrace();
                    }
                    if (model.getColumnCount() != 5)
                        model.addRow(rowDataArray);
                }
            } else JOptionPane.showMessageDialog(table, "Необходимо
заполнить все данные!");
        }
        else JOptionPane.showMessageDialog(table, "Вы не ввели

```

```
данные!");
```

```
    }  
    }  
}  
  
class DeleteListener implements ActionListener {  
    /**  
     * Класс DeleteListener кнопки "Удалить"  
     * @throws ChooseErrorException() Если строка строка таблицы не выбрана  
     * @exception ChooseErrorException()  
     */  
    public void actionPerformed (ActionEvent event) {  
        DataBase dbHandler = new DataBase();  
  
        try {  
            if (table.isRowSelected(table.getSelectedRow())) {  
                int t = table.getSelectedRow();  
                model.removeRow(t);  
  
                if (model.getColumnCount() == 3)  
                    dbHandler.deleteWorkers(t+1);  
  
                if (model.getColumnCount() == 4)  
                    dbHandler.deleteNumbers(t+1);  
                if (model.getColumnCount() == 5)  
                    dbHandler.deleteReports(t+1);  
            }  
            else throw new ChooseErrorException();  
        }  
        catch (ClassNotFoundException e) {  
            // TODO АВТОМАТИЧЕСКИ СОЗДАНЫЙ БЛОК catch  
            e.printStackTrace();  
        }  
        catch (SQLException e) {  
            e.printStackTrace();  
        }  
        catch (ChooseErrorException ex) {  
            JOptionPane.showMessageDialog(table, ex.getMessage());  
            // logTabs.warn("Line not selected! ", ex);  
        }  
    }  
}
```

```
class DeleteAllListener implements ActionListener {  
    /**  
     * Класс DeleteAllListener кнопки "ОЧИСТИТЬ"  
     */  
    public void actionPerformed (ActionEvent event) {  
        DataBase dbHandler = new DataBase();  
        try {  
  
            if (model.getColumnCount() == 3)  
                dbHandler.deleteAllWorkers();  
  
            if (model.getColumnCount() == 4)  
                dbHandler.deleteAllNumbers(model.getRowCount());  
  
            if (model.getColumnCount() == 5)  
                dbHandler.deleteAllReports();  
        }  
    }  
}
```

```

    }
    catch (ClassNotFoundException e) {
        // TODO Автоматически созданный блок catch
        e.printStackTrace();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }

    while (model.getRowCount() > 0) {
        model.removeRow(0);
    }
}

class UpListener implements ActionListener {
    /**
     * Класс UpListener кнопки "Up" - поднимает выбранную строку
     * @throws ChooseErrorException() Если строка строка таблицы не выбрана
     * @exception ChooseErrorException() Записывает сообщение
     * @exception IndexOutOfBoundsException()
     */
    public void actionPerformed (ActionEvent event) {
        try {
            if (table.isRowSelected(table.getSelectedRow())) {

model.moveRow(table.convertRowIndexToView(table.getSelectedRow()),

table.convertRowIndexToView(table.getSelectedRow()),

table.convertRowIndexToView(table.getSelectedRow() - 1)); // Поднимаем выделенную
строку

            }
            else throw new ChooseErrorException();
        }
        catch (ChooseErrorException ex) {
            JOptionPane.showMessageDialog(table, ex.getMessage());
            logTabs.warn("Line not selected! ", ex);
        }
        catch (IndexOutOfBoundsException ex) { logTabs.debug("Index out of
Bounds! ", ex); }
    }
}

class DownListener implements ActionListener {
    /**
     * Класс DownListener кнопки "Down" - опускает выбранную строку
     * @throws ChooseErrorException() Если строка строка таблицы не выбрана
     * @exception ChooseErrorException() Записывает сообщение
     * @exception IndexOutOfBoundsException()
     */
    public void actionPerformed (ActionEvent event) {
        try {
            if (table.isRowSelected(table.getSelectedRow())) {

model.moveRow(table.convertRowIndexToView(table.getSelectedRow()),

table.convertRowIndexToView(table.getSelectedRow()),

table.convertRowIndexToView(table.getSelectedRow() + 1)); // Поднимаем выделенную
строку

```

```

        }
        else throw new ChooseErrorException();
    }
    catch (ChooseErrorException ex) {
        JOptionPane.showMessageDialog(table, ex.getMessage());
    }
    catch (IndexOutOfBoundsException ex) {}
}

class SearchListener implements ActionListener {
    public void actionPerformed (ActionEvent event) {
        String ftxt = searchFilter.getText();
        filter(ftxt);
    }
}

class PlusListener implements ActionListener {
    public void actionPerformed (ActionEvent event) {
        String countStr = out.getText();
        int countInt = Integer.parseInt(countStr);
        countInt++;
        out.setText(Integer.toString(countInt));
    }
}

class MinusListener implements ActionListener {
    public void actionPerformed (ActionEvent event) {
        String countStr = out.getText();
        int countInt = Integer.parseInt(countStr);
        if(countInt > 0) countInt--;
        out.setText(Integer.toString(countInt));
    }
}

private class ChooseErrorException extends Exception {
    /**
     * Класс ChooseErrorException (Если пользователь не выбрал строку)
     *
     */
    public ChooseErrorException() {
        super ("Не выбрана строка!");
    }
}

public void setTab(JPanel tab, int gx1, int gy1, int gx2, int gy2,
    int gx3, int gy3, int an1, int an2, int an3, int fill1,
    int fill2, int fill3, boolean f) {
    if(f) {
        tab.add(this.bar, new GridBagConstraints(gx1, gy1, 1, 1, 0, 0,
            an1, fill1,
            new Insets(2, 2, 2, 2), 0, 0));
        tab.add(this.scroll, new GridBagConstraints(gx2, gy2, 1, 1, 0, 0,
            an2, fill2,
            new Insets(2, 2, 2, 2), 0, 0));
    }
    else {
        tab.add(this.label, new GridBagConstraints(gx1, gy1, 1, 1, 0, 0,

```



```

        an1, fill1,
        new Insets(2, 2, 2, 2), 0, 0));
tab.add(this.out, new GridBagConstraints(gx2, gy2, 1, 1, 0, 0,
        an2, fill2,
        new Insets(2, 2, 2, 2), 0, 0));

    }
tab.add(this.AD, new GridBagConstraints(gx3, gy3, 1, 1, 0, 0,
        an3, fill3,
        new Insets(2, 2, 2, 2), 0, 0));

    }

}

```

## 4.5 xmlReader

```

import org.w3c.dom.*;
import org.xml.sax.SAXException;
import javax.swing.table.*;
import javax.xml.parsers.*;
import java.io.*;

public class xmlReader {

    private DefaultTableModel model;
    private DocumentBuilder dBuilder;
    private Document doc;
    private File f;

    xmlReader(Tabs[] arrayTabs){
        f = new File("Table2.xml");
        try {
            // Создание парсера документа
            dBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            // Создание пустого документа
            doc = dBuilder.newDocument();
            // Чтение документа из файла
            doc = dBuilder.parse(f);
            // Нормализация документа
            doc.getDocumentElement().normalize();
        }
        catch (ParserConfigurationException e) { e.printStackTrace(); }
        catch (SAXException e) { e.printStackTrace(); }
        catch (IOException e) { e.printStackTrace(); }
        // Получение списка элементов с именем book

        for (int i = 0; i < 4; i++) {
            if(i != 2) {
                NodeList nlBooks =
doc.getElementsByTagName(arrayTabs[i].getName().replaceAll("[= ]", ""));
                // Цикл просмотра списка элементов и запись данных в таблицу
                for (int temp = 0; temp < nlBooks.getLength(); temp++) {
                    model = arrayTabs[i].getModel();

                    // Выбор очередного элемента списка
                    Node elem = nlBooks.item(temp);

```

```

        // Получение списка атрибутов элемента
        NamedNodeMap attrs = elem.getAttributes();

        // Чтение атрибутов элемента
        String[] tableList = new
String[model.getColumnCount()];
        for (int j = 0; j < model.getColumnCount(); j++){
            String row =
attrs.getNamedItem(model.getColumnName(j).replace(" ", "")).getNodeValue();
            System.out.print(row + " ");
            tableList[j] = row;
        }
        System.out.print("\n");
        // Запись данных в таблицу
        model.addRow(tableList);
    }
}
else {
    NodeList nlBooks =
doc.getElementsByTagName(arrayTabs[i].getName().replaceAll("[= ]", ""));
    Node elem = nlBooks.item(0);
    NamedNodeMap attrs = elem.getAttributes();

    int t = model.getRowCount();
    String count = new String(""+t);
    arrayTabs[i].setCount(count);
}
}
}
}
}

```

#### 4.6xmlCreator

```

import org.w3c.dom.*;

import javax.swing.table.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;
import java.io.FileWriter;
public class xmlCreator {

    private DefaultTableModel model;
    private Node list;
    private Node tag;
    private DocumentBuilder builder;
    private Document doc;
    private Transformer trans;
    private FileWriter fw;

    xmlCreator(Tabs[] arrayTabs) {
        try {
            // Создание парсера документа
            builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
            // Создание пустого документа
            doc = builder.newDocument();
        }
        catch (ParserConfigurationException e) { e.printStackTrace(); }
    }
}

```

```

String[] tags = {"workers", "numbers", "repair", "reports"};
list = doc.createElement("Table");
doc.appendChild(list);
// Создание дочерних элементов book и присвоение значений атрибутам
for (int i = 0; i < 4; i++) {
    if (i != 2) {
        tag = doc.createElement(tags[i]);
        list.appendChild(tag);
        model = arrayTabs[i].getModel();
        for (int j = 0; j < model.getRowCount(); j++) {
            Element elem =
doc.createElement(arrayTabs[i].getName().replaceAll("[= ]", ""));
            tag.appendChild(elem);

            elem.setAttribute((String)model.getColumnName(0).replace(" ", ""),
(String)model.getValueAt(j, 0));

            elem.setAttribute((String)model.getColumnName(1).replace(" ", ""),
(String)model.getValueAt(j, 1));

            elem.setAttribute((String)model.getColumnName(2).replace(" ", ""),
(String)model.getValueAt(j, 2));
            if (model.getColumnCount() > 3)

            elem.setAttribute((String)model.getColumnName(3).replace(" ", ""),
(String)model.getValueAt(j, 3));
            if (model.getColumnCount() > 4)

            elem.setAttribute((String)model.getColumnName(4).replace(" ", ""),
(String)model.getValueAt(j, 4));
        }
    }
    else {
        tag = doc.createElement(tags[i]);
        list.appendChild(tag);
        Element repairs =
doc.createElement(arrayTabs[i].getName().replaceAll("[= ]", ""));
        tag.appendChild(repairs);
        repairs.setAttribute("Занято",
(String)arrayTabs[i].getCount());
    }
}
try {
    // Создание преобразователя документа
    trans = TransformerFactory.newInstance().newTransformer();
    // Создание файла с именем books.xml для записи документа
    fw = new FileWriter("Table2.xml");
    // Запись документа в файл
    trans.transform(new DOMSource(doc), new StreamResult(fw));
}
catch (TransformerConfigurationException e) { e.printStackTrace(); }
catch (TransformerException e) { e.printStackTrace(); }
catch (IOException e) { e.printStackTrace(); }
}
}

```

## 4.7 SaveLoad

```
import java.awt.*;
import java.io.*;
import javax.swing.*;
import javax.swing.table.*;

/**
 * Класс загрузки и сохранения файлов
 * @author Semenov Mikhail 0306
 * @version 1.00
 */

public class SaveLoad {

    private DefaultTableModel model;
    private JTabbedPane tabs;
    private Tabs[] arrayTabs;

    SaveLoad (Tabs[] arrayTabsGui, JTabbedPane tabsPane, String sao, int mod, Frame
frame) {
        tabs = tabsPane;
        arrayTabs = arrayTabsGui;
        FileDialog dlg = new FileDialog(frame, sao, mod);
        dlg.setFile(".txt");
        dlg.setVisible(true);
        if(dlg.getFile() != null) {
            String fileName = dlg.getDirectory() + dlg.getFile();
            if (sao == "Сохранить") Save(fileName);
            else if (sao == "Сохранить все") SaveAll(fileName);
            else if (sao == "Открыть") ImportF(fileName);
        }
    }

    void Save(String fileName) {
        try {
            BufferedWriter writer = new BufferedWriter (new
FileWriter(fileName));
            if(tabs.getSelectedIndex() != 3) {
                model = (arrayTabs[tabs.getSelectedIndex() - 1]).getModel();
                writer.write(arrayTabs[tabs.getSelectedIndex() -
1].getName());

                writer.write("\n");
                for (int i = 0; i < model.getRowCount(); i++) { // Для всех
строк
                    for (int j = 0; j < model.getColumnCount(); j++) { //
Для всех столбцов
                        writer.write ((String) model.getValueAt(i, j));
                        writer.write(" / "); // Записать символ перевода
каретки
                    }
                }
                writer.write("\n");
            }
        }
        else {
            writer.write(arrayTabs[2].getName());
            writer.write("\n");
            writer.write("Занято номеров: ");
            writer.write(arrayTabs[2].getCount());
            writer.write("\n");
        }
    }
}
```

```

        }
        writer.close();
    }
    catch(IOException e) // Ошибка записи в файл
    { e.printStackTrace(); }
}

void SaveAll(String fileName) {
    try {
        BufferedWriter writer = new BufferedWriter (new
FileWriter(fileName));
        for (int k = 0; k < 4; k++) {
            writer.write(arrayTabs[k].getName());
            writer.write("\n");
            if (k != 2) {
                model = (arrayTabs[k]).getModel();
                for (int i = 0; i < model.getRowCount(); i++) { // Для
всех строк
                    for (int j = 0; j < model.getColumnCount(); j++)
{ // Для всех столбцов
                        writer.write ((String) model.getValueAt(i,
j)); // Записать значение из ячейки
                        writer.write(" / "); // Записать символ
перевода каретки
                    }
                    writer.write("\n");
                }
            }
            else {
                writer.write("Занято номеров: ");
                writer.write(arrayTabs[k].getCount());
                writer.write("\n");
            }
        }
        writer.close();
    }
    catch(IOException e) // Ошибка записи в файл
    { e.printStackTrace(); }
}

void ImportF(String fileName) {
    try {
        BufferedReader reader = new BufferedReader(new
FileReader(fileName));
        String rowDataString;
        String[] rowDataArray;
        if(tabs.getSelectedIndex() != 3) {
            model = (arrayTabs[tabs.getSelectedIndex() - 1]).getModel();
            int rows = model.getRowCount();
            for (int i = 0; i < rows; i++) model.removeRow(0); // Очистка
таблицы

            rowDataString = reader.readLine();
            arrayTabs[tabs.getSelectedIndex() -
1].setName(rowDataString);
            do {
                rowDataString = reader.readLine();
                if(rowDataString != null) {
                    rowDataArray = rowDataString.split(" / ");
                    model.addRow(rowDataArray); // Запись строки в
таблицу
                }
            } while(rowDataString != null);
        }
    }
}

```

```

    }
    else {
        rowDataString = reader.readLine();
        arrayTabs[2].setName(rowDataString);
        rowDataString = reader.readLine();
        String countStr = "";
        for (int i = 0; i < rowDataString.length(); i++) {
            if(Character.isDigit(rowDataString.charAt(i))) countStr
= countStr + rowDataString.charAt(i);
        }
        arrayTabs[2].setCount(countStr);
    }
    reader.close();
}
catch (FileNotFoundException e) {e.printStackTrace();} // файл не найден
catch (IOException e) {e.printStackTrace();}
}
}

```

## 4.8 reportsCreator

```

import java.awt.FileDialog;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

import javax.swing.table.DefaultTableModel;

import com.itextpdf.kernel.geom.PageSize;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.Cell;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Table;

```

```

public class reportsCreator {

    reportsCreator(DefaultTableModel tableModel) {

        PrintWriter pw = null;
        try {
            pw = new PrintWriter(new FileWriter("./Report.pdf"));

            PdfWriter writer = new PdfWriter("./Report.pdf");
            PdfDocument pdfDoc = new PdfDocument(writer);
            pdfDoc.addNewPage();
            Document document = new Document(pdfDoc);
            Table table = new Table(tableModel.getColumnCount());

            for(int i = 0; i < tableModel.getColumnCount(); i++)
            {
                table.addCell(new Cell().add(tableModel.getColumnName(i).toString()));
            }
        }
        catch (IOException e) {e.printStackTrace();}
    }
}

```

```

        }
        for(int i = 0; i < tableModel.getRowCount(); i++){
            for(int j = 0; j < tableModel.getColumnCount(); j++){
                table.addCell(new
Cell().add(tableModel.getValueAt(i,j).toString()));
            }
        }
        document.add(table);
        document.close();
    }
    catch (IOException e) { e.printStackTrace();
    }

}

reportsCreator(DefaultTableModel tableModel, int r) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(new FileWriter("./Report.html"));
    } catch (IOException e) { e.printStackTrace();
    }

    pw.println("<TABLE BORDER><TR>");
    for(int i = 0; i < tableModel.getColumnCount(); i++)
    {
        pw.println("<TH>" + tableModel.getColumnName(i));
    }
    pw.println("</TR>");

    for(int i = 0; i < tableModel.getRowCount(); i++) {
        pw.println("<TR>");
        for(int j = 0; j < tableModel.getColumnCount(); j++)
            pw.println("<TD style = \"background: #333; width:256px;
color: white; font-weight: bold; \">" + (String) tableModel.getValueAt(i,j));
        }
        pw.println("</TABLE>");
        pw.close();
    }
}

```

## 4.9 ThreadThree

```

import javax.swing.table.DefaultTableModel;

/**
 * Класс трех потоков
 * @author Semenov Mikhail 0306
 * @version 1.00
 */
public class ThreadThree extends Thread {
    private int type;
    private DefaultTableModel tableModel;
    private Tabs[] arrayTabs;
    private Object shared;

    public ThreadThree(int i, DefaultTableModel Model, Tabs[] arrayTabs1, Object
shared) {
        this.shared = shared;
    }
}

```

```

        type=i;
        tableModel = Model;
        arrayTabs = arrayTabs1;
    }

    public void run() {

        if (type==1) {
            synchronized (shared) {
                try {
                    shared.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                try {
                    xmlReader q = new xmlReader(arrayTabs);
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
        }

        if (type==2) {
            synchronized (shared) {
                shared.notifyAll();
                try {
                    shared.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                tableModel.addRow(new String[] {} );
                shared.notifyAll();
            }
        }

        if (type==3) {
            synchronized (shared) {
                shared.notifyAll();
                try {
                    shared.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                reportsCreator q1 = new reportsCreator(tableModel);
                reportsCreator q2 = new reportsCreator(tableModel, 1);
            }
        }
    }
}

```



## 4.10 DataBase

```
public class DataBase extends Config {
    Connection dbConnection;

    public Connection getDbConnection() throws ClassNotFoundException, SQLException {
        String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort + "/" +
dbName + "?serverTimezone=UTC";

        Class.forName("com.mysql.cj.jdbc.Driver");

        dbConnection = DriverManager.getConnection(connectionString, dbUser,
dbPass);

        return dbConnection;
    }

    public void addWorkers(String FIO, String post, String stag) throws
ClassNotFoundException, SQLException {
        String insert = "INSERT INTO " + Const.WORKERS_TABLE + "(" +
Const.WORKERS_NAME + "," +
Const.WORKERS_POST +
"," + Const.WORKERS_STAG + ")" +
"VALUES(?, ? , ?)";
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, FIO);
        prSt.setString(2, post);
        prSt.setString(3, stag);

        prSt.executeUpdate();
    }

    public void addNumbers(String guest, String number, String capacity, String
price) throws ClassNotFoundException, SQLException {
        String insert = "INSERT INTO " + Const.NUMBERS_TABLE + "(" +
Const.NUMBERS_NAME + "," + Const.NUMBERS_NUM + "," +
Const.NUMBERS_CAPACITY + "," + Const.NUMBERS_PRICE + ")" +
"VALUES(?, ?, ?, ?)";
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);

        prSt.setString(1, guest);
        prSt.setString(2, number);
        prSt.setString(3, capacity);
        prSt.setString(4, price);

        prSt.executeUpdate();
    }

    public void deleteNumbers(int i) throws ClassNotFoundException, SQLException {
        String str = "" + i;
        PreparedStatement prSt = getDbConnection().prepareStatement("DELETE FROM
numbers WHERE idnumbers = " + str);
        prSt.executeUpdate();

        prSt.executeUpdate();
    }

    public void deleteWorkers(int i) throws ClassNotFoundException, SQLException {
        String str = "" + i;
```

```

        PreparedStatement prSt = getDbConnection().prepareStatement("DELETE FROM
workers WHERE idworkers = " + str);
        prSt.executeUpdate();

        prSt.executeUpdate();
    }
    public void deleteReports(int i) throws ClassNotFoundException, SQLException {
        String str = "" + i;
        PreparedStatement prSt = getDbConnection().prepareStatement("DELETE FROM
reports WHERE idreports = " + str);
        prSt.executeUpdate();

        prSt.executeUpdate();
    }

    public void deleteAllWorkers() throws ClassNotFoundException, SQLException {
        PreparedStatement prSt = getDbConnection().prepareStatement("DELETE FROM
workers");
        prSt.executeUpdate();

        prSt.executeUpdate();
    }
    public void deleteAllNumbers(int i) throws ClassNotFoundException, SQLException {
        int j = 1;
        while(j<i+1) {
            String str = null;
            str = "" + j;
            PreparedStatement prSt = getDbConnection().prepareStatement("DELETE FROM
numbers WHERE idnumbers = "+str);
            prSt.executeUpdate();
            j++;
            System.out.println("888");
        }
    }
    public void deleteAllReports() throws ClassNotFoundException, SQLException {
        PreparedStatement prSt = getDbConnection().prepareStatement("DELETE FROM
reports");
        prSt.executeUpdate();

        prSt.executeUpdate();
    }

    public void addReports(int workers,int numbers) throws ClassNotFoundException,
SQLException {
        String insert = "INSERT INTO " + Const.REPORT_TABLE + "(" +
Const.REPORT_WORKERS + "," + Const.REPORT_NUMBERS+ ")" + "VALUES(?, ?)";
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);

        prSt.setInt(1, workers);
        prSt.setInt(2, numbers);
        prSt.executeUpdate();
    }

    public ResultSet getWorkers() throws ClassNotFoundException, SQLException {
        ResultSet resSet = null;
        String select = "SELECT * FROM " + Const.WORKERS_TABLE;

        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        resSet = prSt.executeQuery();
    }

```

```

        return resSet;
    }

    public ResultSet getNumbers() throws ClassNotFoundException, SQLException {
        ResultSet resSet = null;
        String select = "SELECT * FROM " + Const.NUMBERS_TABLE;

        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        resSet = prSt.executeQuery();

        return resSet;
    }

    public ResultSet getReports() throws ClassNotFoundException, SQLException {
        ResultSet resSet = null;
        String select = "SELECT * FROM " + Const.REPORT_TABLE;

        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        resSet = prSt.executeQuery();

        return resSet;
    }

    public ResultSet getCount() throws ClassNotFoundException, SQLException {
        ResultSet resSet = null;
        String select = "SELECT * FROM " + Const.REPORT_TABLE;

        PreparedStatement prSt = getDbConnection().prepareStatement(select);
        resSet = prSt.executeQuery();

        return resSet;
    }
}

```

#### 4.11 Config

```

public class Config {
    protected String dbHost = "127.0.0.1";
    protected String dbPort = "3306";
    protected String dbUser = "root";
    protected String dbPass = "user";
    protected String dbName = "hotel";
}

```

#### 4.12 Const

```

public class Const {
    public static final String WORKERS_TABLE = "workers";

    public static final String WORKERS_ID = "idworkers";
    public static final String WORKERS_NAME = "FIO";
    public static final String WORKERS_POST = "post";
    public static final String WORKERS_STAG = "stag";

    public static final String NUMBERS_TABLE = "numbers";

    public static final String NUMBERS_ID = "idnumbers";
    public static final String NUMBERS_NAME = "guest";
}

```

```

    public static final String NUMBERS_NUM = "number";
    public static final String NUMBERS_CAPACITY = "capacity";
    public static final String NUMBERS_PRICE = "price";

    public static final String REPORT_TABLE = "reports";
    public static final String REPORT_COUNT = "idreports";
    public static final String REPORT_WORKERS = "idworkers";
    public static final String REPORT_NUMBERS = "idnumbers";

}

```

#### 4.13 JUnitTabsTest

```

import org.junit.Test;
import org.junit.Assert;
import javax.swing.table.DefaultTableModel;
public class JUnitTabsTest {

    @Test
    public void testGetModel() {
        Tabs tab1 = new Tabs();
        Assert.assertTrue(tab1.getModel() instanceof DefaultTableModel);
    }

    @Test
    public void testGetName() {
        Tabs tab2 = new Tabs();
        Assert.assertTrue(tab2.getName() != null);
    }

}

```

## Заключение

В результате проделанной работы разработан ПК «Учет, администрирование и редактирование записей работников гостиницы», предназначенный для администрирования и учета информации по работникам гостиницы, гостям и номерам.

В процессе проектирования созданы описание вариантов использования ПК, прототип интерфейса пользователя, объектная модель ПК, диаграмма классов, описание поведения ПК, диаграмма действия ПК.

Курсовой проект удовлетворяет поставленным требованиям.