

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СТАРТАП-ПРОЕКТ. СИСТЕМА
АВТОМАТИЗАЦИИ РАБОТЫ ЗАВЕДЕНИЙ
ОБЩЕСТВЕННОГО ПИТАНИЯ

Студент

 04.06.24
(подпись, дата)

Мартынов Никита Владимирович
(ФИО полностью)

Руководитель

 04.06.24
(подпись, дата)

Карамышева Н. С.
(фамилия, инициалы)

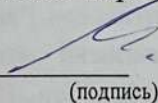
Нормоконтролёр

 04.06.24
(подпись, дата)

Бычков А. С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой

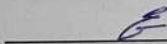

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

17.06.2024 № 22)

Секретарь ГЭК



Е.И. Гурин
(подпись)

(фамилия, инициалы)

Пенза 2024

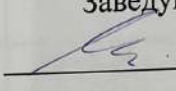
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«19» апреля 2024г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Мартынов Никита Владимирович гр. 20ВВП1 факультета вычислительная техника

2. Тема работы «Стартап-проект. Система автоматизации работы заведений общественного питания»

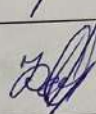
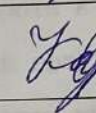
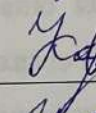
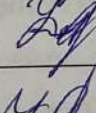
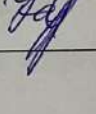
Тема утверждена приказом ПГУ №405/о от "19" апреля 2024г.

3. Руководитель бакалаврской работы к.т.н. доцент кафедры ВТ ПГУ Карамышева Н.С.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки - система автоматизации работы заведений общественного питания.
Разработать автоматизированную информационную систему "Система автоматизации работы заведений общественного питания", использующую клиент-серверную архитектуру, и базу данных PostgreSQL. Составить бизнес-модель и финансовый план. Язык программирования Java, СУБД PostgreSQL, среда разработки Apache NetBeans, графический клиент для управления БД pgAdmin4. Программа должна обладать интуитивно понятным интерфейсом.

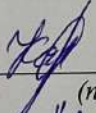
5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметной области, определить архитектуру приложения, выбрать инструменты для разработки, реализовать приложение.

6. Календарный график выполнения работы

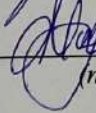
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10 %	19.04.24 - 25.04.24	
2.	Выбор архитектуры системы и технологий, применяемых при проектировании и разработке	15 %	26.04.24 - 02.05.24	
3.	Разработка интерфейсов	20 %	03.05.24 - 07.05.24	
4.	Разработка логики системы	25 %	08.05.24 - 19.05.24	
5.	Тестирование и отладка	15 %	20.05.24 - 22.05.24	
6.	Выполнение задания по БЖД	5 %	23.05.24 - 25.05.24	
7.	Оформление выпускной работы	10 %	25.05.23 - 31.05.23	

Дата выдачи задания "19" апреля 2024г.

Руководитель бакалаврской работы

 19.04.2024
(подпись, дата)

Задание к исполнению принял студент

 19.04.2024
(подпись, дата)

Работу к защите допустить
Декан факультета

 Л.Р. Рогов
07.06.2024 (подпись, дата)

Реферат

Бакалаврская работа содержит 179 л., 59 рис., 18 табл., 11 источн. и 3 прил.
АВТОМАТИЗИРОВАННАЯ СИСТЕМА, ОБРАБОТКА СОБЫТИЙ,
ЛОГИРОВАНИЕ, БАЗА ДАННЫХ, ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ,
JAVA, SWING, POSTGRESQL

Цель работы – разработка с обственной системы автоматизации работы заведений общественного питания малого масштаба с целью упрощения процесса выполнения повседневных и управленческих задач.

Для достижения поставленной цели было разработано настольное приложение, отвечающее следующим поставленным задачам: работа в системе под учетной записью сотрудника, управление учетными записями сотрудников, регистрация и ведение истории заказов посетителей, работа с меню заведения, отслеживание истории действий в системе, прием и обработка заказов клиентов, внесение сведений о блюдах на остатке, управление заявками на продукты.

В результате проведенной работы разработана автоматизированная система с графическим интерфейсом Swing GUI, клиент-серверной архитектурой на языке Java. Для хранения данных применена СУБД PostgreSQL в связке с графическим клиентом pgAdmin4.

					ПГУ 1.090301.17.001 ПЗ		
Изм	Лист	№ докум.	Подпись	Дата			
Разраб.		Мартынов Н.В.		04.06.24			
Пров.		Карамышева Н.С.		04.06.24			
Н. Контр.		Бычков А.С.		04.06.24			
Утв.		Митрохин М.А.		04.06.24			
					Стартап-проект. Система автоматизации работы заведений общественного питания		
					Пояснительная записка		
					Литера	Лист	Листов
						2	179
					ФВТ, каф. ВТ гр. 20ВВП1		

МИНОБРНАУКИ РОССИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра "Вычислительная техника"

УТВЕРЖДАЮ
Заведующий кафедрой

Митрохин М.А.

(Подпись)

(Фамилия, инициалы)

«___» _____ 20__ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ В ФОРМЕ СТАРТАПА

Студент Мартынов Никита Владимирович

Направление подготовки 09.03.01 Информатика и вычислительная техника.

Профиль Программное обеспечение средств вычислительной техники и автоматизированных систем

Форма обучения Очная

1. Тема: Бизнес-проект «Система автоматизации работы заведений общепита «Умный сервис». Информационная система «Умный сервис» является десктопным приложением, предназначенным для решения повседневных и управленческих задач заведений общественного питания.

Тема одобрена экспертной комиссией и утверждена приказом ректора от «___» _____ 20__ г. № _____.

2. Срок представления работы к защите «___» _____ 20__ г.

3. Общая часть ВКР:

Подготовка компьютерной презентации бизнес-проекта (стартапа).

4. Индивидуальная часть ВКР:

4.1. Исходные данные:

Разработать систему автоматизации работы заведений общественного питания "Умный сервис", представляющую из десктопное приложение. Составить бизнес-модель и финансовый план. Язык программирования Java, среда разработки Apache NetBeans. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Java

2. Swing GUI

3. PostgreSQL

4. Сетевое программирование

5. pgAdmin4

4.2. Содержание пояснительной записки ВКР (по разделам):

Анализ предметной области, обоснование актуальности разработки, обзор аналогов, анализ требований к разработке, постановка задачи, выбор инструментов разработки, архитектура приложения, проектирование и разработка, тестирование и отладка, БЖД, бизнес-план: финансовые и временные ограничения на проект (бюджет, сроки реализации, ожидаемые затраты и доходы и т.д.).

5. Перечень графического материала: UML-диаграммы (диаграмма классов, диаграмма развертывания, диаграмма деятельности, диаграмма вариантов использования, диаграмма последовательности)

Руководитель ВКР

Задание принял к исполнению

(подпись, дата)

(Ф.И.О.)

Экспертное заключение
о согласовании темы ВКР в форме стартапа

Экспертная комиссия в составе декан ФВТ д.т.н., профессор Л.Р. Фионова, доцент кафедры ВТ, к.э.н., И.В. Акифьев, ведущий системный архитектор ООО «Ай-теко ВЦ», к.т.н. М.П. Синев, рассмотрев бизнес-проект, стартап-проект «Система автоматизации работы заведений общественного питания» Мартынова Никиты Владимировича подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему «Стартап-проект. Система автоматизации работы заведений общественного питания» Мартынова Никиты Владимировича может выполняться в форме стартапа (бизнес-проекта).

Председатель ЭК

д.т.н., профессор
декан ФВТ



Л.Р. Фионова

Члены ЭК

к.э.н., доцент кафедры ВТ



И.В. Акифьев

к.т.н., ведущий системный архитектор
ООО «Ай-теко ВЦ»



М.П. Синев

Реферат

Бакалаврская работа содержит 179 л., 59 рис., 18 табл., 11 источн. и 3 прил.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА, ОБРАБОТКА СОБЫТИЙ, ЛОГИРОВАНИЕ, БАЗА ДАННЫХ, ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ, JAVA, SWING, POSTGRESQL

Цель работы – разработка с обственнй системы автоматизации работы заведений общественного питания малого масштаба с целью упрощения процесса выполнения повседневных и управленческих задач.

Для достижения поставленной цели было разработано настольное приложение, отвечающее следующим поставленным задачам: работа в системе под учетной записью сотрудника, управление учетными записями сотрудников, регистрация и ведение истории заказов посетителей, работа с меню заведения, отслеживание истории действий в системе, прием и обработка заказов клиентов, внесение сведений о блюдах на остатке, управление заявками на продукты.

В результате проведенной работы разработана автоматизированная система с графическим интерфейсом Swing GUI, клиент-серверной архитектурой на языке Java. Для хранения данных применена СУБД PostgreSQL в связке с графическим клиентом pgAdmin4.

					ПГУ 1.090301.17.001 ПЗ			
Изм	Лист	№ докум.	Подпись	Дата				
Разраб.		Мартынов Н.В.			Стартап-проект. Система автоматизации работы заведений общественного питания Пояснительная записка	Литера	Лист	Листов
Пров.		Карамышева Н.С.					2	179
Н. Контр.		Бычков А.С.				ФВТ, каф. ВТ гр. 20ВВП1		
Утв.		Митрохин М.А.						

Содержание

Введение.....	5
1 Система автоматизации работы заведений общественного питания	7
1.1 Анализ предметной области	7
1.1.1 Актуальность разработки	7
1.1.2 Анализ существующих аналогов	9
1.1.3 Обоснование актуальности разработки программы.....	23
1.1.4 Постановка задачи	23
1.2 Выбор технологий и средств разработки приложения	28
1.2.1 Выбор языка программирования.....	28
1.2.2 Выбор СУБД.....	29
1.2.3 Графический клиент pgAdmin 4	31
1.2.4 Выбор среды разработки.....	32
1.2.5 Выбор графической библиотеки	33
1.2.6. Обмен данными JSON	35
1.3 Проектирование информационной системы	35
1.3.1 Паттерн MVC	35
1.3.2 Архитектура приложения.....	37
1.3.3 Проектирование базы данных	43
1.4 Разработка приложения.....	56
1.4.1 Реализация БД	56
1.4.2 Описание программы	60
1.4.3 Руководство пользователя	79
1.4.4 Тестирование и отладка.....	89
2 Охрана труда.....	94
Заключение	97
Список использованных источников	99
Приложение А – UML-диаграммы	100
Приложение Б – Листинг программы	106
Приложение В – Презентация.....	169

Перечень сокращений и обозначений

В настоящей работе применяют следующие обозначения и сокращения

АС – автоматизированная система

БД – база данных

Десктопное приложение – программа, предназначенная для установки и запуска на персональном компьютере или ноутбуке

Многопоточность – способность программы выполнять несколько потоков (threads) одновременно

Рефакторинг кода – процесс улучшения внутренней структуры программного кода без изменения его внешнего поведения

СУБД – система управления базами данных

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

FK (Foreign key) – внешний ключ, понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных

GUI (Graphical User Interface) – графический интерфейс пользователя

ID (Identifier) – уникальный номер объекта

IDE (Integrated development environment) – интегрированная среда разработки

JSON (JavaScript Object Notation) – текстовый формат обмена данными

PK (Primary key) – первичный ключ, понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

Введение

В современном мире информационные технологии стали неотъемлемой частью жизни. Каждый день они помогают людям из самых разных сфер деятельности сделать процесс выполнения повседневных задач быстрее и эффективнее, значительно уменьшая риск возникновения каких-либо ошибок. Развитие информационных технологий происходит стремительными шагами, подстраиваясь под рынок и удовлетворяя большую часть его потребностей.

В предпринимательской деятельности современные технологии облегчают выполнение многих бизнес-процессов: одним из таких является автоматизация повседневных задач, таких как учет финансов, хранение и обработка различных данных, а также анализ работы бизнеса в целом. В этом случае на помощь приходит комплекс программно-аппаратных средств, именуемый как автоматизированная система, которая может значительно повысить уровень ряда показателей.

В сфере работы заведений общественного питания автоматизированная система является эффективным инструментом для решения различных повседневных и управленческих задач. Например, она позволяет грамотно вести клиентский учет, собирая и храня историю активности посетителей заведения, что дает возможность повысить лояльность клиентов путем внедрения разных бонусных или скидочных программ.

Если брать относительно крупные кафе или рестораны, то в них, как правило, в большинстве случаев уже внедрены и активно применяются различные популярные системы автоматизации работы. Установка, настройка, техническая поддержка и обслуживание обычно окупаются за короткий промежуток времени.

С заведениями малого масштаба ситуация обстоит иначе – окупаемость внедрения автоматизированной системы будет идти дольше, чем в крупных заведениях, так как этот процесс обусловлен рядом факторов: меньшее количество посетителей, более низкая покупательская способность, удаленность от крупных населенных пунктов, нехватка персонала или отсутствие

необходимых знаний по работе с техническими средствами. Совокупность вышеприведенных факторов приводит к тому, что клиентский и административный учет либо ведется традиционным способом (на бумажных носителях, а работа с данными осуществляется вручную), либо не ведется совсем.

Возникает потребность в разработке автоматизированной системы, способной учесть все особенности работы кафе и ресторанов малого масштаба. Требуется создать такую систему, которая обладает всем необходимым базовым функционалом и имеет невысокую стоимость внедрения, настройки, технической консультации и обучения персонала работы с программой. Именно поэтому выбор темы «Система автоматизации работы заведений общественного питания» в качестве выпускной квалификационной работы является чрезвычайно актуальным.

Целью выпускной работы является:

- исследование предметной области и анализ существующих решений;
- разработка десктопного приложения для упрощения хранения и обработки клиентских и внутренних данных;
- проектирование и разработка пользовательского интерфейса;
- тестирование разработанной системы;
- внедрение и развертывание автоматизированной системы.

Проект поможет предпринимателям малого бизнеса в сфере общественного питания повысить эффективность их работы, увеличит скорость обслуживания клиентов, а также даст возможность предприятию оставаться конкурентно-способными на рынке.

1 Система автоматизации работы заведений общественного питания

1.1 Анализ предметной области

1.1.1 Актуальность разработки

Известно, что всякий успешный бизнес, в первую очередь, начинается с грамотного и ответственного выполнения повседневных задач. Закупить необходимые продукты, найти хороших поставщиков с качественной и выгодной по цене продукцией, составить меню заведения, проверить, должным ли образом сотрудники исполняют свои обязанности (кого из них наградить премией за честную работу, а кому - сделать выговор за невыполнение поставленных целей), предоставить клиенту высококачественный сервис на основе его предпочтений – эти и другие, казалось бы, переходящие из дня в день хлопоты требуют от руководящего персонала достаточно большого количества времени.

Для решения этой проблемы были создано множество систем автоматизации работы предприятия, ставших сейчас неотъемлемой частью любого относительно крупного заведения. Суть применения таких систем заключается в том, что за выполнение типовых ежедневных задач, связанных с хранением, обработкой и анализом различных данных, отвечает специальный программно-аппаратный комплекс, получивший название «Автоматизированная система». Однажды внедрив в работу предприятия такую систему, сотрудникам, например, больше не придется вручную определять востребованность той или иной позиции в меню, составлять разные отчеты о доходах или расходах заведения за нужный период времени – это и многое другое способна с легкостью решить автоматизированная система. Сотруднику остается лишь проанализировать полученные данные и сделать соответствующие выводы. Значительную часть времени, тратить которую раньше приходилось на повседневные задачи, теперь можно направить на реализацию давних идей и инноваций, способных оказать положительное влияние, в первую очередь, на

доход предприятия, а также на общее впечатление клиентов о работе заведения общественного питания.

Ни трудно согласиться, что серьезный бизнес требует серьезных решений. Именно поэтому крупные заведения общественного питания зачастую используют автоматизированные системы, состоящие из нескольких модулей, каждый из которых, во-первых, нуждается в отдельной установке и настройке и, во-вторых, выполняет свою функцию: с помощью одного модуля осуществляется взаимодействие обслуживающего персонала с посетителями заведения, а когда заказ принят и отправлен в работу, то дальнейшие действия с ним производит повар уже при помощи своего другого модуля. Если возникла необходимость поработать со статистическими, на этот счёт также существует отдельный модуль. Стоит отметить тот факт, что чаще всего такой набор программных средств для автоматизации используется в связке с различными внешними устройствами, которые в совокупности с программной частью дают больший эффект.

Пока крупные предприятия тестируют и внедряют новые технологичные решения, заведения общественного питания малого масштаба либо совсем не применяют автоматизированные системы в своей работе, либо ведут всю отчетность традиционным способом – на бумажном носителе. Проблема автоматизации работы малых заведений общественного питания обусловлена множеством факторов: отсутствие технической возможности (например, в населенном пункте нет подключения к сети Интернет), затраты на приобретение и обслуживание автоматизированной системы могут составлять значимую часть от доходов всего предприятия, слабо развита клиентская база и, наконец, заведение не нуждается в подобных технологических решениях, потому что и без этого справляется с ведением бизнеса.

Каждый день сотрудникам заведений общественного питания малого масштаба приходится вручную, если отсутствуют какие-либо средства автоматизации, решать немаловажные задачи: учет числа совершенных заказов, подсчет выручки заведения в конце рабочего дня, фиксирование сведений об объемах приготовленных блюд, прием продукции от поставщиков и

документирование данного процесса в виде накладной, осуществление контроля за исполнением обязанностей сотрудников предприятия и др. Зачастую, вся информация, связанная с выполнением вышеперечисленных задач, хранится и обрабатывается сотрудниками заведения в бумажной форме.

Решение, разработанное в данной бакалаврской работе, призвано решить вышеперечисленные проблемы. Автоматизированная система будет иметь невысокую стоимость по сравнению с конкурентами, возможность настраивания под любую платформу, недорогое сопровождение и обслуживание, настройку под сторонние внешние устройства (принтеры, сканеры штрих-кодов, фискальный накопитель). Также система имеет АС имеет возможность работы без сети Интернет при помощи хранения всех данных локально. Немаловажным является тот факт, что программа будет обладать набором базовых функций, которые удовлетворят большую часть потребностей по автоматизации работы малого заведения общественного питания.

1.1.2 Анализ существующих аналогов

Среди аналогов систем автоматизации работы заведений общественного питания можно выделить следующие решения:

- конфигурация «Простая столовая»;
- автоматизированная система «BitRest»;
- система контроля и учета работы заведения «R-Keeper Lite».

Рассмотрим возможности АС «Простая столовая».

Общий вид данной программы представлен на рисунке 1.

Чтобы начать работу с программой, необходимо совершить вход от имени какого-либо пользователя (рисунке 2). Количество доступных функций определяется уровнем привилегий учетной записи. В данном вход осуществлен от имени администратора.

[illegible]

Рисунок 3 – Информация о блюдах

Для того чтобы посмотреть состав блюда, изменить или удалить его, необходимо один раз кликнуть на нужную позицию списка, после чего блюдо станет для просмотра, редактирования или удаления (рисунок 4). С помощью кнопки «Добавить» в систему вносятся сведения о новом блюде.

Range: B3:B10		Range: C3:E10					Formulas	
		Revenue (M)	Costs	Gross Profit	Operating Costs	Net Profit		
1	40	5,000	3,000	2,000	2,000	0		
2	45	5,500	3,300	2,200	2,200	0		
3	50	6,000	3,600	2,400	2,400	0		
4	55	6,500	3,900	2,600	2,600	0		
5	60	7,000	4,200	2,800	2,800	0		
6	65	7,500	4,500	3,000	3,000	0		
7	70	8,000	4,800	3,200	3,200	0		

Рисунок 4 – Работа с блюдами заведения

На вкладке «Меню» имеется возможность планирования меню на каждую неделю (рисунок 5).

Если сотрудник желает узнать блюда, которые будут продаваться в определенный день, достаточно кликнуть по нужному дню недели (рисунок 6). Например, проверим список блюд на 01.08.2012. Управление меню происходит с помощью кнопок «Добавить», «Изменить», «Удалить», расположенных справа от детализации блюд в меню (рисунок 7).

					ПГУ 1.090301.17.001 ПЗ	Лист
						11
Изм	Лист	№ докум.	Подпись	Дата		

Блюда № Меню Заказы клиентов Прием заказов Производство блюд Приготовление блюд Подарки на заказ				
Дерево: Меню недели				
Меню	№ Меню недели	Дата меню	Посетит день	Сотрудник
18	29	10.07.2012	Вт	админ
28		11.07.2012	Ср	админ
29		12.07.2012	Чт	админ
30		13.07.2012	Пт	админ
31		14.07.2012	Сб	админ
32		15.07.2012	Вс	админ
33	30	16.07.2012	Пн	админ
34		17.07.2012	Вт	админ
35		18.07.2012	Ср	админ
36		19.07.2012	Чт	админ
37		20.07.2012	Пт	админ
38		21.07.2012	Сб	админ
39		22.07.2012	Вс	админ
	31	23.07.2012	Пн	админ
		24.07.2012	Вт	админ
		25.07.2012	Ср	админ
		26.07.2012	Чт	админ
		27.07.2012	Пт	админ
		28.07.2012	Сб	админ
	32	29.07.2012	Вс	админ
		30.07.2012	Пн	админ
		31.07.2012	Вт	админ
		01.08.2012	Ср	админ

Рисунок 5 – Планирование меню заведения

18	30.07.2012	Пн	админ
28	31.07.2012	Вт	админ
30	01.08.2012	Ср	админ
32	02.08.2012	Чт	админ

№ п/п	Наименование блюда	Наличие	Посетит	Категория	Посетит
101	Салат из овощей с сыром	☒	☒	Парик	31
45	Салат из овощей	☒	☒	Парик	31
67	Суп из овощей (пюре)	☒	☒	Парик	31
68	Салат из овощей	☒	☒	Парик	31
69	Картофель жареный (пюре)	☒	☒	Парик	31
14	Рис	☒	☒	Парик	31
24	Салат	☒	☒	Парик	31
57	Пюре из картофеля	☒	☒	Парик	31
88	Картофель жареный	☒	☒	Парик	31

Рисунок 6 – Детализация блюд в меню заведения



Рисунок 7 – Кнопки управления меню заведения

Перейдя на вкладку «Заказы клиентов», можно увидеть информацию обо всех принятых заказах (рисунок8): номер заказа, дату, номер недели, кому принадлежит заказ, какой сотрудник принял, сумму заказа, примечания к заказу, а также статус его выполнения.

ID	Дата заказа	Номер заказа	Клиент	Подзаказ	Сотрудник	Сумма заказа	Статус заказа
02-07-2012_1	02-07-2012_1	25	Администратор	Заказ 1	админ	75,00	Заказан
02-07-2012_2	02-07-2012_2	26	Администратор	Заказ 2	админ	41,47	Исполнен
02-07-2012_3	02-07-2012_3	27	Администратор	Заказ 3	админ	0,00	Заказан

Рисунок 8 – Принятые заказы клиентов

Дважды кликнув по строке с заказом, откроется окно с подробной информацией о нем (рисунок 9).

№ заказа	Клиент	Примечание
02-07-2012_1_1	админ	

Дата заказа	Подзаказ	Статус заказа
02-07-2012_1	Заказ 1	Заказан

Номер заказа	Сотрудник	Сумма заказа
25	админ	0,00

ID	№	Код товара	Наименование	Количество	Цена	Сумма	Примечание	Дата заказа	Номер заказа
1	1	65	Салат из овощей и помидоров	1	0,00	0,00		02-07-2012	25

Рисунок 9 – Информация о заказе

Чтобы зафиксировать или просмотреть список продуктов, принятые от какого-либо поставщика, на вкладке «Приход продуктов» необходимо создать накладную (рисунок 10). Там же доступен просмотр, редактирование, удаление и добавление новых накладных.

Блюда Меню Заказы клиентов Прием продуктов Производство блюд Продажа блюд Продукты на складе									
Прием продуктов									
Выбор блюда по баркоду на складе от поставщика									
№ документа	Тип документа	№ документа	Дата операции	Код блюда	Код поставщика	Поставщик	Сотрудник	Сумма	Примечание
1	Поступление	1	18.09.2011	1	1	Мясная колбаса	админ	3 000.00	Ожидание
2	Поступление	2	09.09.2011	1	1	Мясная колбаса	админ	25 000.00	Ожидание
								28 000.00	
Поступившие продукты									
Поступившие продукты (3/4)									
№	Код продукта	Продукт	Ед. изм.	Количество	Цена	Сумма	Примечание	Код партии	Код поставщика
1	1	Картофель	кг	100.00	175.00	2 000.00			1
2	2	Морковь	кг	100.00	35.00	3 500.00			1
3	3	Картофельная панировка	кг	100.00	20.00	2 000.00			1
4	4	Лук репчатый	кг	100.00	16.50	1 650.00			1

Рисунок 10 – Просмотр поступивших продуктов

После приготовления блюд нужно внести соответствующую информацию о них в систему, перейдя на вкладку «Производство блюд», где по клику нужной строки раскроются сведения о названии блюда, его количестве и дате приготовления (рисунок 11).

Блюда | Меню | Заказы клиентов | Прием продуктов | **Производство блюд** | Продажа блюд | Продукты на складе

Производство блюд Создание блюда из продуктов

№ документа	№ документа	Дата операции	Ссылка на документ	Сотрудник	Статус	Примечание
1-Акт	1	12.09.2011		админ		
2-Акт	2	24.09.2011		админ		
3-Акт	45	21.04.2024		админ	Ожидание	

Произведенные блюда Расход продуктов

Произведенные блюда (3/3)

№	№	Код блюда	Блюдо	Количество	Примечание	Код операции
1	1	1	Суп овощной	1		3
2	2	2	Суп рыбный	3		3
3	3	10	Воры	2		3

Рисунок 11 – Работа с произведенными блюдами

На вкладке «Продукты на складе» отображается остаток продуктов, из которых повара готовят блюда (рисунок 12).

Блюда Меню Заказы клиентов Прием продуктов Производство блюд Продажа блюд Продукты на складе									
Продукты на складе									
Просмотреть склад									
Код продукта	Код продукта	Продукт	Остаток	Цена	Сумма	Ед. изм.	Прим.	Код партии	Код поставщика
1	1	Картофель	100.00	175.00	2 000.00	кг			1
2	2	Морковь	100.00	35.00	3 500.00	кг			1
3	3	Картофельная панировка	100.00	20.00	2 000.00	кг			1
4	4	Лук репчатый	100.00	16.50	1 650.00	кг			1

Рисунок 12 – Просмотр продуктов на складе

Программа обладает следующими преимуществами:

- наличие учетных записей пользователей, от имени которых ведется работа с программой. Это позволяет разграничить доступный функционал;
- программа обладает простой установкой и не требует много ресурсов системы;
- система имеет весь необходимый базовый функционал, отлично подходящий для заведения общественного питания малого характера;
- присутствует возможность формирования отчетов, содержащих сведения о работе предприятия за указанный промежуток времени;
- возможность сортировки желаемой информации.

Из минусов данной автоматизированной системы можно выделить следующее:

- устаревший, громоздкий интерфейс приложения;
- функционал каждой из вкладок либо не до конца проработан, либо должен быть изменен и перемещен в другой раздел, либо вовсе вырезан из программы;
- программа распространяется на коммерческой основе;
- отсутствует ведение истории действий сотрудников.

Автоматизированная система «BitRest».

Чтобы начать работу с программой, необходимо пройти авторизацию (рисунок 13).



Рисунок 13 – Авторизация в приложении «BitRest»

После авторизации откроется окно со списком совершенных заказов. В каждой строке содержится информация о зале, в котором сделан заказ, столик, сумма заказа, время, количество гостей за столом и место какого-либо гостя за столом (рисунок 14).

Окно заказов	Текущий сотрудник - Admin						18 Июнь 12, 17:30
Зал	Стол	Сумма	Сумма2	Время	Платеж	Гарантия	Примечания
Основной зал	1	10.00	10.00	23.00 15:53	5	1	Admin
Основной зал	1	50.00	50.00	23.00 15:58	8	1	Счет
Основной зал	1	30.00		23.00 16:06	11	1	Счет
Основной зал	1	150.00	150.00	23.00 16:41	10	2	Admin
Основной зал	3	12.00	12.00	23.00 15:55	1	1	Admin
Основной зал	4	100.00	100.00	23.00 14:40	1	1	Admin
Основной зал	5	90.00	—	23.00 14:40	22	1	Admin
Основной зал	6	400.00	—	23.00 14:44	1	1	Admin
Основной зал	7	50.00	50.00	23.00 13:29	1	1	Admin
Основной зал	7	90.00	—	20.00 13:33	2	1	Admin

Рисунок 14 – Окно заказов

Прежде чем создать новый заказ, нужно открыть карту столиков заведения (рисунок 15). Если столик свободен, тогда система позволит прикрепить к нему новый заказ, иначе будет выведено соответствующее сообщение.



Рисунок 15 – Карта состояния столиков заведения

Выбрав необходимый свободный столик, пользователь увидит окно, в котором требуется указать число гостей (рисунок 16).

Рисунок 16 – Окно создания нового заказа

Далее в заказ необходимо будет добавить позиции, которые гость захотел приобрести (рисунок 17). В этом окне можно найти нужное блюдо, указать желаемое количество, а также указать особые пожелания клиента по приготовлению какого-либо блюда.

Рисунок 17 – Оформление заказа

В данной программе предусмотрена возможность выбора способа оплаты заказа: наличными, дисконтной картой, дебетовой или кредитной картой (рисунок 18).



Рисунок 18 – Окно выбора способа оплаты заказа

Чтобы применить дисконтную карту, необходимо ввести её номер в специальное диалоговое окно (рисунок 19) либо провести карту магнитной полосой по считывателю (требуется наличие специального оборудования).

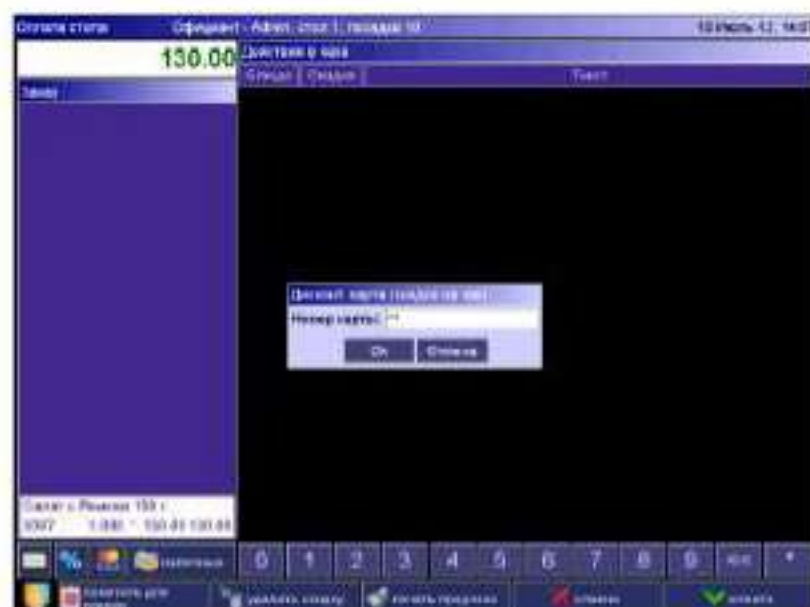


Рисунок 19 – Применение дисконтной карты клиента

Для пользователя с учетной записью администратора доступны следующие функции (рисунок 20):



Рисунок 20 – Функции администратора в «BitRest»

Программа обладает следующими преимуществами:

- простой и интуитивно понятный интерфейс;
- каждый заказ привязан к конкретному номеру стола, что делает работу официанта проще и удобнее;
- в системе присутствует возможность выбора способа оплаты. Доступна оплата сразу несколькими способами;
- расширенный набор отдельных функций для администратора;
- поддержка работы с различными внешними устройства (кассовый ящик, сканер штрих-кодов, принтер чеков и т.д.).

Из минусов данной автоматизированной системы можно выделить следующее:

- устаревший дизайн приложения;
- в программе не предусмотрена работа с поставщиками продуктов;
- программа распространяется на коммерческой основе;
- в приложении нельзя создать профиль клиента для дальнейшего учета информации о нем.

Обзор возможностей АС «R-Keeper Lite».

Чтобы начать работу с программой, в окне авторизации необходимо ввести персональный PIN-код (рисунок 21).

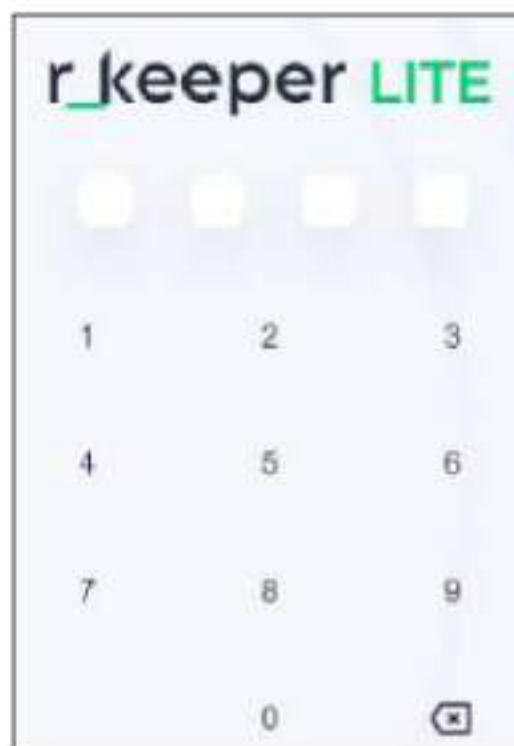


Рисунок 21 – Страница авторизации в «R-Keeper Lite»

После прохождения авторизации пользователь попадает на главный экран приложения (рисунок 22), логически разделенный на залы (холлы). В каждом зале отображается информация обо всех столах.



Рисунок 22 – Главное меню приложения

Для регистрации нового заказа потребуется кликнуть по нужному столику. Далее мы сможем добавить в заказ нужные позиции (рисунок 23).

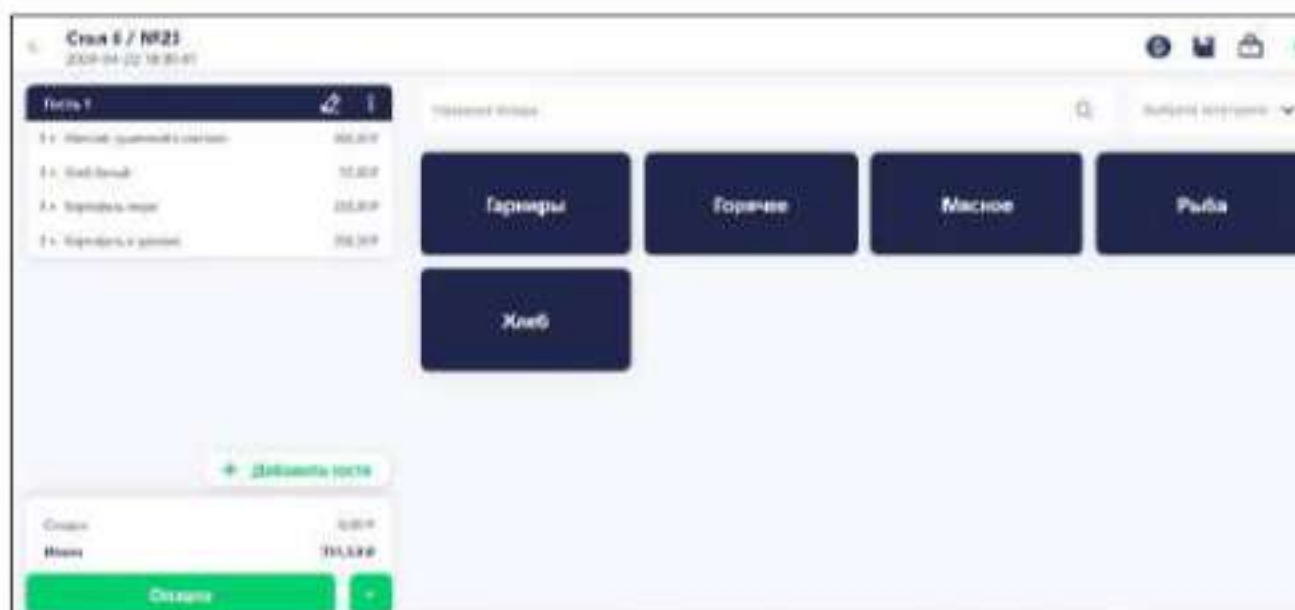


Рисунок 23 – Формирование нового заказа

Когда заказ сформирован, его необходимо оплатить, нажав кнопку «Оплата». Среди доступных способов оплаты – наличными, картой, дисконтной картой (рисунок 24).



Рисунок 24 – Оплата сформированного заказа

В автоматизированной системе «R-Keeper Lite» существует меню «Инструменты», в котором располагаются различные административные функции (рисунок 25).



Рисунок 25 – Меню приложения

Программа обладает следующими преимуществами:

- наличие учетных записей пользователей, от имени которых ведется работа с программой. Это позволяет разграничить доступный функционал;
- АС «R-Keeper Lite» является облачным решением, поэтому вести работу заведения можно с любого устройства, подключенного к сети Интернет;
- возможность просмотра истории заказов и поиска чеков;
- проверка состояния кассы заведения на текущий момент времени;
- возможность сортировки желаемой информации.

Из минусов данной автоматизированной системы можно выделить следующее:

- так как АС «R-Keeper Lite» - это веб-приложение, для его работы требуется постоянное наличие интернет-соединения. В заведениях общественного питания, расположенных в отдаленной местности, с этим могут возникать проблемы;

– упрощенная версия программы «R-Keeper Lite» имеет сильно ограниченный функционал. Чтобы воспользоваться остальными важными функциями, потребуется приобрести полную настольную версию «R-Keeper»;

– программа распространяется на коммерческой основе;

– в Lite-версии приложения отсутствует возможность работы с внешними устройствами.

1.1.3 Обоснование актуальности разработки программы

Разработанная в данной работе АС будет обладать невысокой по сравнению с конкурентами ценой, возможность настраивания под любую платформу, недорогое сопровождение и обслуживание, а также настройку под сторонние устройства (сканеры, принтеры чеков, банковские терминалы и др.). Кроме того, при разработке будет учтена возможность использования продукта в рамках тестового периода, что позволит заведениям общественного питания малого масштаба оценить все преимущества автоматизации. Также программа будет обладать базовым функционалом, необходимым для работы заведения. Планируется реализация автономной работы для уменьшения зависимости от постоянного интернет-соединения.

1.1.4 Постановка задачи

Требуется реализовать приложение, которое будет соответствовать следующим функциональным требованиям:

– возможность работы с профилями посетителей заведения (регистрация новых клиентов, изменение учетных данных, удаление профиля клиента, просмотр информации, хранящейся в профиле клиента);

– прием и работа с заказами (создание нового заказа путем добавления в него нужных клиенту позиций меню, изменение добавленных в заказ позиций, отмена совершенного заказа, выбор профиля посетителя для возможности списания, начисления бонусных баллов, а также для сохранения информации в истории заказов);

– конфигурирование меню заведения (добавление новых позиций в меню, изменений действующих позиций меню, аннулирование действующих позиций);

					ПГУ 1.090301.17.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

– возможность работы с заявками на приобретение нужной продукции (просмотр списка заявок, создание заявок со списком необходимых продуктов, закрытие заявки как результат закупки продукции, отклонение заявки, принятие заявки в работу);

– администрирование учетных записей сотрудников (просмотр информации в профиле сотрудника, внесение персональных данных о сотруднике, изменение учетных данных профиля, автоматическое создание учетной записи сотрудника для входа в систему в момент первичной регистрации, аннулирование учетной записи сотрудника при его увольнении);

– работа с поставщиками продуктов (добавление новых поставщиков, изменение сведений о действующих поставщиках, удаление информации о них, прием товаров путем формирования соответствующей накладной);

– анализ данных, связанных с работой заведения (просмотр числа заказов за выбранный период, формирование краткой выписки с итогами работы заведения или сотрудника за выбранный период, получение информации об активности сотрудников, выявление популярности (актуальности) той или иной позиции меню заведения).

На рисунке 26 изображена диаграмма вариантов использования приложения при входе в систему с учетной записью администратора и с учетной записью кассира-повара (рисунок 27).

Разрабатываемая автоматизированная система должна соответствовать следующим техническим требованиям:

– удобный и интуитивно понятный интерфейс для работы с программой;

– расположение графических элементов должно предоставлять пользователю быстрое ориентирование в интерфейсе;

– возможность масштабирования интерфейса в процессе работы;

– разработанная программа является десктопным решением;

– наличие возможности сохранения данных локально, пока отсутствует подключение к серверу;

– операционная система: Windows 7 и более поздние версии;

- оперативная память: не менее 512 МБ ОЗУ;
- незанятое дисковое пространство: не менее 1,5 Гб ПЗУ;
- процессор: Intel Pentium III 800 MHz;
- разрешение экрана: 1024 x 728.

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

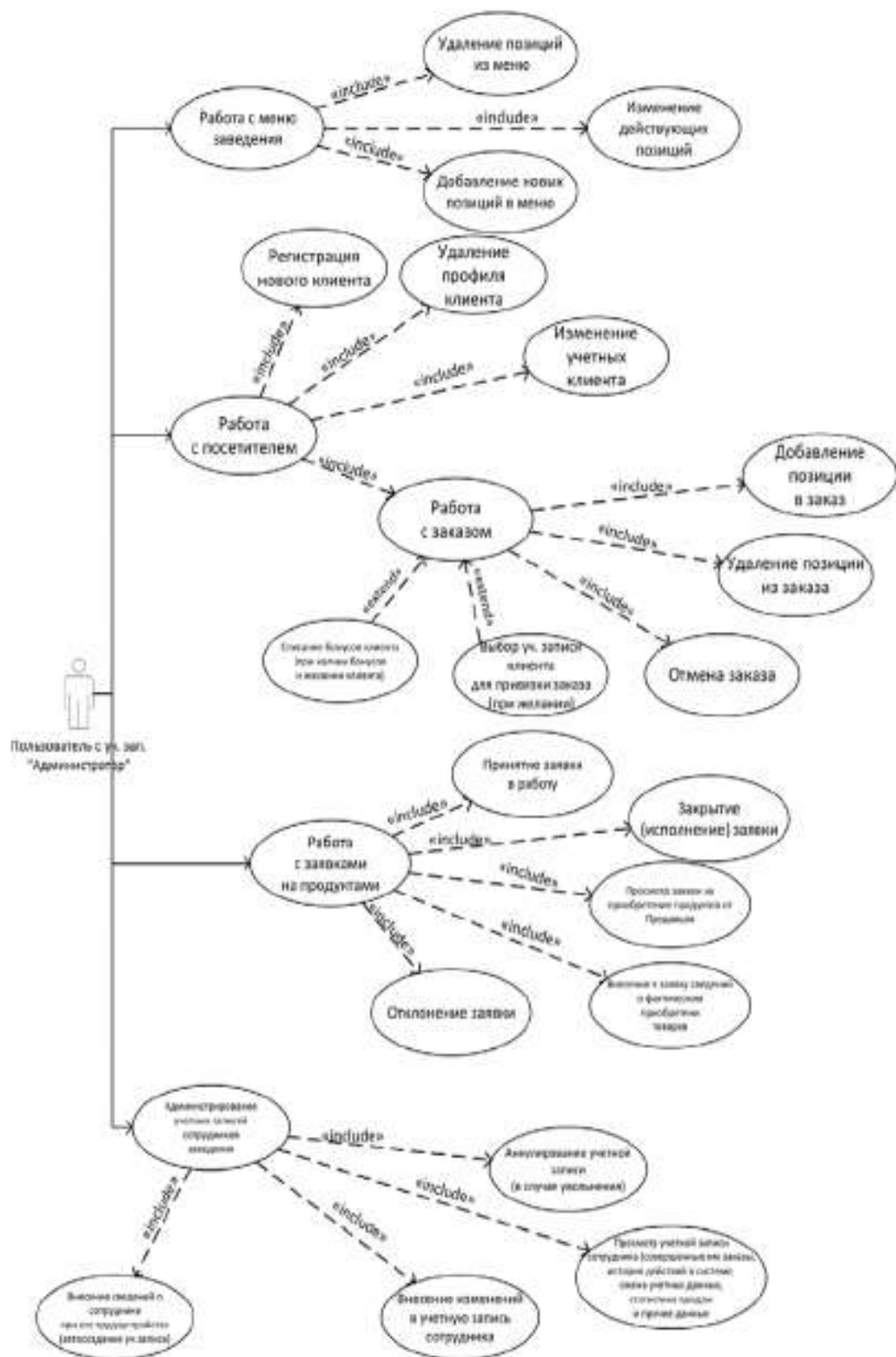


Рисунок 26 – Диаграмма вариантов использования приложения пользователя с учетной записью администратора

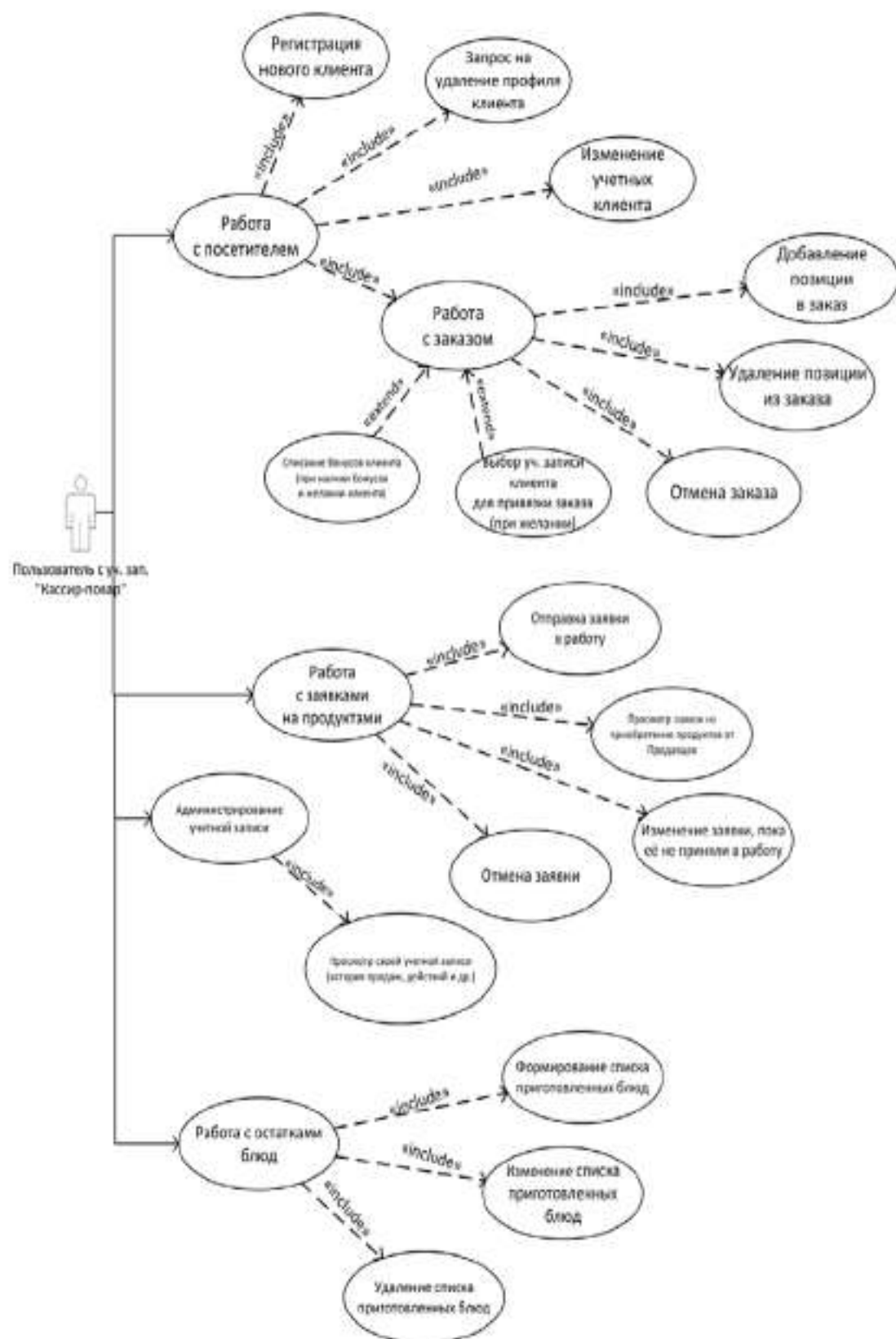


Рисунок 27 – Диаграмма вариантов использования приложения с правами
обычного пользователя

1.2 Выбор технологий и средств разработки приложения

В данном разделе будут описаны технологии и средства разработки, использованные при разработке программы по автоматизации работы общественного питания.

Ниже приведен список выбранных технологий и средств разработки:

- Java;
- Swing GUI;
- Apache NetBeans;
- PostgreSQL;
- pgAdmin4;
- JSON.

1.2.1 Выбор языка программирования

Язык программирования C++.

C++ - это высокоуровневый язык программирования, обладающий возможностями как низкоуровневых языков (например, C), так и объектно-ориентированных языков [1]. Данный язык позволяет писать эффективный и быстродействующий код благодаря возможности работы с указателями, множеством библиотек и шаблонами. Он также поддерживает многопоточность и множество парадигм программирования. Однако C++ не является достаточно гибким для создания графического интерфейса приложения.

Язык программирования Python.

Python - высокоуровневый интерпретируемый язык программирования с динамической типизацией [2]. Python известен своей простотой и читаемостью кода, богатой стандартной библиотекой, поддержкой различных парадигм программирования (функциональное, объектно-ориентированное). Он также широко используется в области науки о данных и искусственного интеллекта.

Самой простой библиотекой для создания графического интерфейса в Python является Tkinter. Эта библиотека предоставляет базовые средства для

создания простых GUI-приложений, однако для решения более серьезных задач не пригодна.

Язык программирования Java.

Java – это объектно-ориентированный язык программирования, который изначально разрабатывался для создания кроссплатформенных приложений [3].

Java обладает простым синтаксисом, автоматическим управлением памятью, механизмом сборки мусора. Этот язык позволяет создавать надежные и безопасные приложения благодаря своей виртуальной машине JVM (JavaVirtualMachine).

Для создания графического интерфейса в Java часто применяется библиотека Swing, предоставляющая широкие возможности для создания различных элементов интерфейса.

После сравнения возможностей одних из популярных языков для реализации серверной и клиентской частей приложения была выбрана Java, так как она работает на виртуальной машине JVM [4], что позволяет запускать Java-приложения на различных операционных системах без изменений в исходном коде. Это обеспечивает максимальную переносимость и универсальность приложений. Использование графической библиотеки Swing предоставляет широкий набор графических компонентов (кнопки, текстовые поля, таблицы и др.), что упрощает разработку и позволяет создавать достойные пользовательские интерфейсы.

1.2.2 Выбор СУБД

СУБД PostgreSQL.

В качестве СУБД была PostgreSQL - свободно распространяемая объектно-реляционная система управления базами данных, предоставляющая широкий спектр функций для хранения, организации и обработки данных [5]. Данная СУБД распространяется на бесплатной основе (OpenSource) и имеет открытый исходный код.

Среди преимуществ PostgreSQL можно выделить:

– позволяет работать с большими размерами данных;

- кроссплатформенность;
- расширяемость и богатый набор типов данных;
- наследование;
- активное сообщество пользователей;
- наличие официальной исчерпывающей документации.

Решением использовать PostgreSQL стало то, что данную СУБД можно быстро и бесплатно установить на любую необходимую платформу. Кроме того, для PostgreSQL были созданы различные графические библиотеки и приложения для удобной работы с ним.

СУБД MySQL.

MySQL — это одна из самых популярных открытых реляционных систем управления базами данных (СУБД). Она широко используется для хранения данных веб-приложений, бизнес-приложений и других типов приложений.

Преимущества MySQL:

- доступна бесплатно для использования, что делает ее доступной для широкого круга разработчиков.
- предлагает простой и интуитивно понятный язык запросов SQL для работы с данными.
- обладает хорошей производительностью и способна эффективно обрабатывать большие объемы данных.
- легко масштабируется как вертикально (добавление ресурсов на одном сервере), так и горизонтально (добавление дополнительных серверов).

Однако стоит помнить, что MySQL не всегда поддерживает хранимые процедуры и триггеры так эффективно, как другие СУБД. Кроме того, могут возникнуть сложности с масштабированием БД на большие объемы данных.

СУБД SQLite.

SQLite - это компактная встраиваемая реляционная система управления базами данных (СУБД). Она отличается от клиент-серверных СУБД, таких как MySQL, тем, что база данных SQLite хранится в одном файле, который не требует

отдельного сервера для работы. SQLite широко используется в мобильных приложениях, встроенных системах и других проектах, где требуется легкая и простая в использовании база данных.

Преимущества SQLite:

- легко встраивается в приложения и не требует отдельного сервера или сложной настройки;
- обеспечивает надежное хранение данных и обеспечивает транзакционную безопасность;
- поскольку SQLite работает внутри приложения, она не требует постоянного соединения с удаленным сервером;
- отлично подходит для небольших приложений или проектов с невысокой нагрузкой.

Стоит помнить, что SQLite может иметь проблемы с производительностью при работе с большими объемами данных или при большой нагрузке. Применение SQLite не рекомендуется для проектов, где требуется одновременный доступ к базе данных от нескольких пользователей.

1.2.3 Графический клиент pgAdmin 4

pgAdmin 4 – приложение с открытым исходным кодом, разработанное для облегчения администрирования системы управления базами данных PostgreSQL. Представляет из себя графический клиент с веб-интерфейсом для возможности быстрого выполнения различных запросов к базе данных, удобного анализа полученных результатов и мониторинга других важных параметров, связанных с работой БД.

К основным достоинствам pgAdmin 4 можно отнести:

- интуитивно понятный и удобный пользовательский интерфейс, который облегчает работу с базами данных PostgreSQL;
- широкий набор инструментов для управления базами данных PostgreSQL, включая создание, изменение и удаление объектов базы данных, выполнение SQL-запросов, анализ выполнения запросов и многое другое;

- предоставление возможности для безопасного подключения к базам данных PostgreSQL через шифрование и аутентификацию;
- наличие графического редактора запросов, который облегчает написание и отладку SQL-запросов;
- инструменты для мониторинга производительности баз данных PostgreSQL, анализа выполнения запросов и оптимизации работы с базами данных.

1.2.4 Выбор среды разработки

Среда разработки Apache NetBeans.

Apache NetBeans – это кросс-платформенная интегрированная среда разработки программного обеспечения на различных языках программирования, включая Java, Python, PHP, JavaScript, C, C++ [6]. Первая публичная версия продукта была разработана компанией студентов Пражского университета в 1996 году.

NetBeans IDE обладает рядом преимуществ, упрощающих разработку программного обеспечения:

- среда разработки имеет редактор кода, который имеет функции подсветки синтаксиса языка, автозаполнения частей кода, отладки программ и другие возможности, делающие написание кода более быстрым и комфортным;
- встроенный графический редактор NetBeans IDE GUI Builder упрощает процесс проектирования пользовательского интерфейса приложения, работа в нём осуществляется путем перетаскивая функциональных компонентов (кнопки, текстовые поля, надписи и др.) на рабочую область макета;
- среда разработки NetBeans предоставляет возможность управлять зависимостями между различными компонентами приложения. Это позволяет автоматически подключать и обновлять необходимые библиотеки и фреймворки;
- NetBeans является свободно распространяемым продуктом с открытым исходным кодом, что позволяет разработчикам гибко настраивать среду разработки под свои потребности;

– NetBeans IDE поддерживает работу в различных операционных системах, включая наиболее распространенные Windows, macOS и Linux. Разработчик может пользоваться ей без ограничений в своей ОС и работать над групповыми проектами с пользователями других платформ.

Данная среда разработки была выбрана потому, что обладает интуитивно понятным интерфейсом, легка в установке и настройке, имеет встроенный графический редактор, упрощающий создание интерфейса приложения.

Среда разработки IntelliJ IDEA.

IntelliJ IDEA — это интегрированная среда разработки (IDE) для программирования на языках Java, Kotlin, Groovy, Scala и других, разработанная компанией JetBrains и предлагающая широкий набор инструментов для повышения производительности разработчика [7].

Преимущества IntelliJ IDEA:

- удобный и интуитивно понятный интерфейс;
- мощные инструменты рефакторинга кода;
- поддержка широкого спектра технологий и фреймворков;
- автоматическое завершение кода и подсказки;
- встроенная система контроля версий;
- широкие возможности интеграции с другими инструментами разработки.

Данная среда разработки требует достаточно высоких ресурсов компьютера, что может сказаться на производительности. Чтобы использовать все возможности программы, необходима платная лицензия. Интерфейс IntelliJ IDEA является достаточно громоздким из-за большого количества функций.

Исходя из выше перечисленных недостатков, был сделан выбор в пользу среды разработки Apache NetBeans.

1.2.5 Выбор графической библиотеки

Графическая библиотека Swing GUI.

Swing GUI (Graphical User Interface) - это набор инструментов и компонентов, предназначенных для создания интерактивных пользовательских

интерфейсов [8]. Эти инструменты позволяют легко добавлять кнопки, текстовые поля, таблицы и другие элементы в разрабатываемые приложения, делая их более удобными для пользователей. Swing обладает переносимостью, гибкостью, простотой использования и поддержкой многоязычности, что делает его популярным выбором для создания кроссплатформенных приложений.

Swing предоставляет простой и интуитивно понятный API для создания пользовательского интерфейса, обладает широким набором базовых компонентов для создания различных элементов интерфейса, встроен в среду разработки Apache NetBeans IDE, что значительно упрощает процесс проектирования интерфейса будущего приложения.

Графическая библиотека JavaFX.

JavaFX — это современная платформа для создания богатых интерфейсов пользовательского взаимодействия для приложений на языке Java. Она предоставляет различные инструменты для создания графических интерфейсов, анимации, мультимедийных элементов и других элементов, необходимых для разработки современных приложений[9].

Преимущества JavaFX:

- обладает широким набором графических возможностей, включая векторную графику, анимацию, 3D-графику и многое другое;
- предлагает простой и интуитивно понятный API для создания интерфейсов, что упрощает разработку;
- приложения, разработанные с использованием JavaFX, могут запускаться на различных платформах без изменений кода;
- хорошо интегрирована с языком Java, что обеспечивает легкость перехода от разработки бэкенда к фронтенду.

Несмотря на перечисленные достоинства, JavaFX свойственны недостаточная документация (некоторые разработчики указывают на недостаток документации и примеров кода), меньшее сообщество разработчиков по

сравнению с другими платформами, что может затруднить получение помощи и поддержки.

Исходя из вышеперечисленных недостатков, был сделан выбор в пользу Apache NetBeans, так как эта среда разработки имеет большое количество информации для реализации поставленных задач.

1.2.6. Обмен данными JSON

JSON (Java Script Object Notation) – стандартный текстовый формат для хранения и передачи структурированных данных [10]. Он основан на синтаксисе объекта в JavaScript, но не привязан к нему. Работу с ним поддерживают многие современные языки программирования: Python, Java и другие.

JSON-формат хранения данных имеет ряд преимуществ:

- простой и легко читаемый синтаксис, который понятен и компьютеру, и человеку;
- JSON-файлы весят меньше, чем файлы других форматов, и загружаются быстрее;
- для работы с JSON можно использовать почти любой популярный язык программирования;
- JSON можно использовать для хранения массивов, упорядоченных списков и коллекций пар «ключ — значение»;
- в JSON можно включить метаданные, которые описывают представленные данные, упрощая работу с ними.

В данной бакалаврской работе JSON-формат будет использоваться для обмена данными между клиентской и серверной частями разрабатываемого приложения.

1.3 Проектирование информационной системы

1.3.1 Паттерн MVC

MVC (Model, View, Controller) – это паттерн (шаблон) проектирования, делящий архитектуру приложения на три основных компонента [11]: модель

(Model), представление (View) и контроллер (Controller). С помощью данного шаблона представляется возможным изменение любого из компонентов приложения независимо друг от друга, что упрощает процесс разработки различных продуктов.

Модель реализует основную логику приложения, отвечающую за данные, методы работы с ними и общую структуру программы. Модель реагирует на команды, посланные ей контроллером, выдавая определенную информацию и изменяя свое состояние. После чего модель передает данные в представление.

Представление – это отображение информации, получаемой от модели, в виде пользовательского графического интерфейса. Представление определяет внешний вид приложения и способы взаимодействия с ним.

Контроллер – это компонент, обеспечивающий взаимодействие с системой: производит обработку действий пользователя, осуществляет валидацию введенных пользователем данных, а затем передает подготовленную информацию модели. Контроллер определяет, как приложение будет реагировать на действия пользователя.

Логическая схема MVC-приложения представлена на рисунке 28.

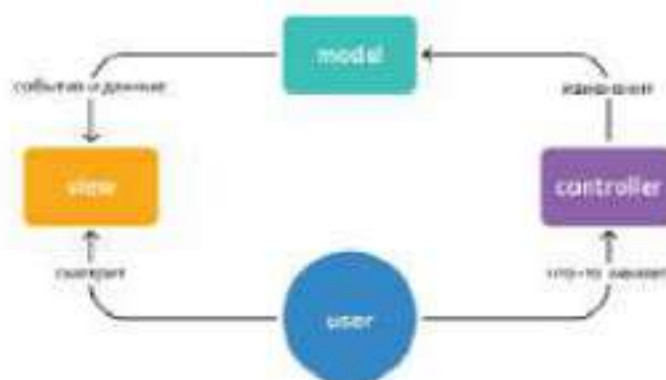


Рисунок 28 – Логическая схема MVC-приложения

Наряду с паттерном MVC, также активно используется шаблон проектирования MVVM (Model, View, ViewModel):

– модель (Model): представляет данные и бизнес-логику приложения;

– представление (View): отображает данные пользователю и реагирует на действия пользователя;

– ViewModel: служит прослойкой между моделью и представлением, обрабатывает логику представления и управляет данными, которые отображаются в представлении.

В качестве шаблона проектирования был выбран паттерн MVC для клиент-серверного приложения с базой данных, потому что MVC обеспечивает четкое разделение ответственностей между компонентами, что делает код более организованным и легким для поддержки, а контроллер в MVC эффективно управляет запросами к базе данных и обновлением данных, что важно для клиент-серверных приложений.

1.3.2 Архитектура приложения

На рисунке 29 представлена архитектура приложения для автоматизации работы заведения общественного питания.

Во время работы с программой пользователь выполняет какие-либо действия, обращаясь к каким-либо элементам графического интерфейса, в котором каждое окно представляет из себя компонент View. Например, заполняя форму регистрации в системе, пользователь сначала вводит информацию в заданные поля, после чего осуществляет регистрацию нажатием кнопки или клавиши.

Когда пользователь нажимает кнопку регистрации, в этот момент компонент группы Controller проводит проверку введенных данных, прежде чем они будут отправлены в дальнейшую обработку. Если данные пользователя соответствуют заявленным требованиям, свою работу начинает Model-компонент, задача которого обработать данные в соответствии с заложенной логикой. Затем этот компонент сообщит View об успешной регистрации. В случае неуспешной валидации данных, компонент Controller сообщит об этом нужному View-компоненту, который уведомит пользователя о неверно введенных данных.

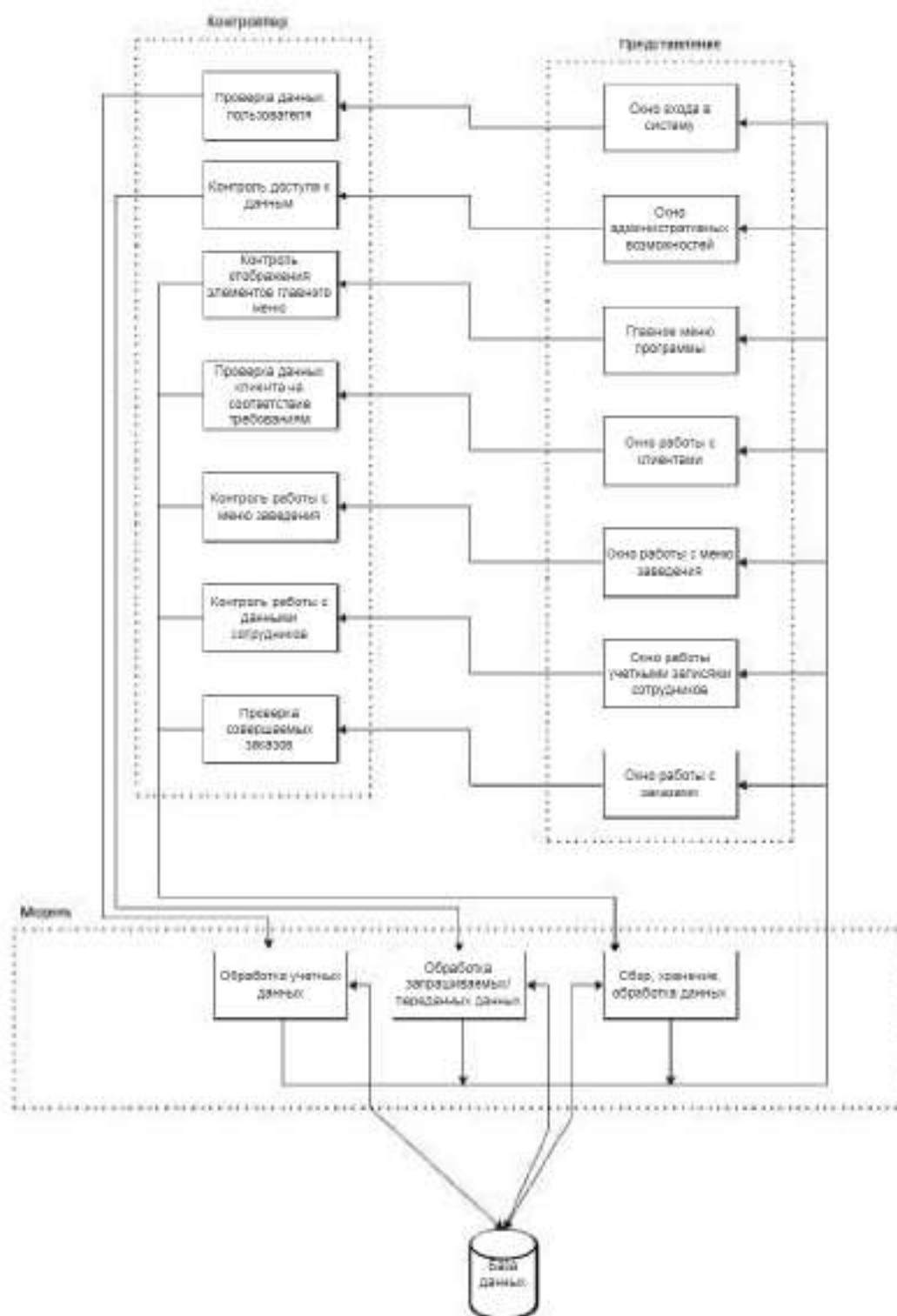


Рисунок 29 – Архитектура приложения

Вся работа с приложением будет осуществляться в виде диалога с пользователем. Диалоговая система состоит из трех ключевых компонентов (рисунок 30):

- клиентская часть, которая обеспечивает пользовательский интерфейс для взаимодействия с системой, включая кнопки, поля ввода, выпадающие списки, таблицы, чекбоксы и др.;

- серверная часть, которая обрабатывает данные, отправленные пользователем, выполняет необходимые алгоритмы и задачи;

- база данных, где хранятся данные с определенными правилами и принципами управления данными.

Система управления базами данных (СУБД) представляет собой набор программных инструментов, которые позволяют создавать, управлять и администрировать базы данных. Другими словами, СУБД предоставляет возможности для организации, контроля и администрирования данных в базах данных.

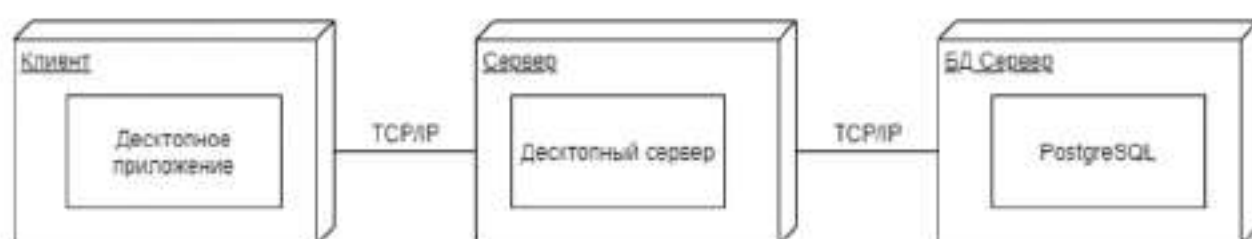


Рисунок 30 – Диаграмма развертывания приложения

На рисунке 31 представлено взаимодействие компонентов приложения в виде диаграммы последовательности.

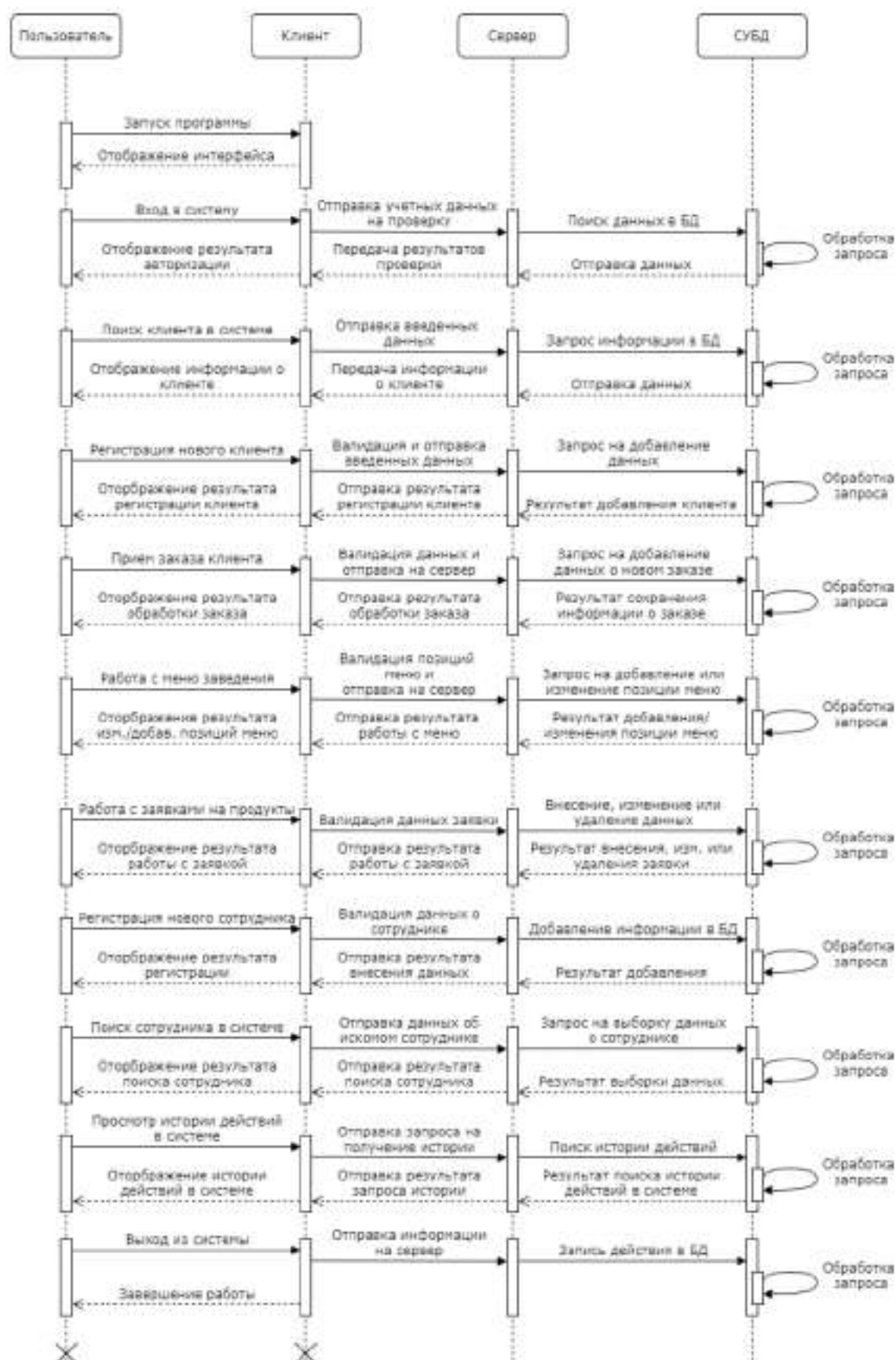


Рисунок 31 – Диаграмма последовательности

1.3.3 Проектирование базы данных

Для хранения информации необходимо реализовать базу данных со следующими таблицами:

- users – таблица, содержащая данные учетных записей сотрудников заведения (таблица 1);
- history – таблица, хранящая историю действий сотрудников в системе (таблица 2);
- employee_passport – таблица, хранящая паспортные данные сотрудников (таблица 3);
- employees – таблица, содержащая основные сведения о сотруднике (таблица 4);
- clients – таблица, содержащая информацию о зарегистрированных посетителях заведения (таблица 5);
- bonus_balance – таблица, содержащая историю изменения бонусного счета посетителя заведения (таблица 6);
- orders – таблица, содержащая историю заказов посетителей заведения (таблица 7);
- ordered – связующая с таблицей orders таблица, отражающая список позиций заказа (таблица 8);
- menu – таблица, хранящая информацию о блюдах заведения, доступных к покупке посетителем (таблица 9);
- product_request – таблица, содержащая заявки на приобретение необходимых продуктов для приготовления блюд (таблица 10);
- requested – связующая с таблицей product_request таблица, содержащая список запрошенных продуктов (таблица 11);
- product_remains – таблица, содержащая информацию об остатках приготовленных блюд и их дате производства (таблица 12);
- invoices – таблица, содержащая информацию о накладных на приобретение продуктов для приготовления блюд (таблица 13);

- product – связующая с таблицей invoices таблица, содержащая список продуктов накладной (таблица 14);
- providers – таблица, содержащая информацию о поставщиках продуктов питания (таблица 15);
- provider_details – таблица, содержащая реквизиты поставщиков продуктов питания (таблица 16).

Таблица 1 – Атрибуты таблицы users

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
user_id	SERIAL	Идентификатор учетной записи	PK	1
login	TEXT	Логин для входа		petrov
pass	TEXT	Пароль для входа		qwerty123
date_upd_pass	TIMESTAMP	Дата изменения пароля		2024-10-10 00:00
privilege_level	TEXT	Уровень доступа учетной записи		Пользователь
status	TEXT	Статус учетной записи		Активен
secret_word	TEXT	Кодовое слово для восстановления доступа в систему		Солнце
date_upd_word	TIMESTAMP	Дата изменения кодового слова		2024-10-10 00:00

Таблица 2 – Атрибуты таблицы history

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
action_id	SERIAL	Идентификатор действия	PK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
action_type	TEXT	Тип действия		Вход в систему
action_description	TEXT	Описание действия		Пользователь @retrov вошел в систему
action_date	TIMESTAMP	Дата действия		2024-10-10 00:00
user_id	INTEGER	Идентификатор учетной записи сотрудника	FK	1

Таблица 3 – Атрибуты таблицы employee_passport

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
passport_id	SERIAL	Идентификатор паспорта сотрудника	PK	1
passport_series	INTEGER	Серия паспорта		5622
passport_number	INTEGER	Номер паспорта		675443
date_birthday	DATE	Дата рождения		2024-10-10
place_birth	TEXT	Место рождения		г. Пенза Пензенская обл.
place_issue	TEXT	Место выдачи паспорта		УМВД России по Пензенской области
date_issue	DATE	Дата выдачи паспорта		2024-10-10
department_code	TEXT	Код подразделения		580-011
registration_address	TEXT	Адрес прописки сотрудника		г. Пенза, ул. Красная, д. 40

Таблица 4 – Атрибуты таблицы employees

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
employee_id	SERIAL	Идентификатор сотрудника	PK	1
surname	TEXT	Фамилия сотрудника		Петров
name	TEXT	Имя сотрудника		Петр
patronymic	TEXT	Отчество сотрудника		Петрович
passport_id	INTEGER	Идентификатор паспорта	FK	1
phone	TEXT	Номер телефона сотрудника		+79271234567
locality	TEXT	Населенный пункт сотрудника		Пенза
street	TEXT	Улица, где проживает сотрудник		Красная
house_number	INTEGER	Номер дома		10
apartment_number	INTEGER	Номер квартиры		10
current_position	TEXT	Занимаемая должность		Повар
date_work_experience	DATE	Дата трудоустройства		2024-10-10
user_id	INTEGER	Идентификатор учетной записи сотрудника	FK	1

Таблица 5 – Атрибуты таблицы clients

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
client_id	SERIAL	Идентификатор клиента	PK	1
surname	TEXT	Фамилия клиента		Петров
name	TEXT	Имя клиента		Петр

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
patronymic	TEXT	Отчество клиента		Петрович
phone	TEXT	Номер телефона		+79271234567
date_registration	TIMESTAMP	Дата регистрации в системе		2024-10-10 00:00
employee_id	INTEGER	Идентификатор сотрудника, выполнившего регистрацию клиента	FK	1

Таблица 6 – Атрибуты таблицы bonus_balance

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
operation_id	SERIAL	Идентификатор операции с балансом	PK	1
operation_type	TEXT	Тип операции		Пополнение
old_balance	DECIMAL(10,2)	Предыдущий баланс		85,3
new_balance	DECIMAL(10,2)	Новый баланс		85,3
date_operation	TIMESTAMP	Дата изменения баланса		2024-10-10 00:00
order_id	INTEGER	Заказ, которым связано изменение баланса	FK	1

Таблица 7 – Атрибуты таблицы orders

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
order_id	SERIAL	Идентификатор заказа	PK	1
employee_id	INTEGER	Идентификатор сотрудника, принявшего заказ	FK	1
client_id	INTEGER	Идентификатор клиента, совершившего заказ	FK	1
date_order	TIMESTAMP	Дата заказа		2024-10-10 00:00
order_status	TEXT	Статус заказа		Исполнен
total	DECIMAL(10,2)	Общая стоимость заказа		500,00

Таблица 8 – Атрибуты таблицы ordered

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
order_id	SERIAL	Идентификатор заказа	FK	1
position_id	INTEGER	Идентификатор заказанной позиции меню	FK	1
position_count	INTEGER	Количество приобретаемого блюда		1

Таблица 9 – Атрибуты таблицы menu

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
position_id	SERIAL	Идентификатор позиции меню	PK	1
position_name	TEXT	Название позиции меню		Борщ

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
unit_measurement	TEXT	Единица измерения, исходя из которой рассчитывается цена блюда		гр.
minimum_volume	DECIMAL(10,2)	Объем, на основе которого указана цена блюда		100
price	DECIMAL(10,2)	Стоимость блюда за указанный объем		100,00
date_create_position	TIMESTAMP	Дата утверждения позиции меню		2024-10-10 00:00
Имя столбца	Тип данных	Описание	Роль атрибута	Пример
position-status	TEXT	Статус позиции		Действующая
employee_id	INTEGER	Идентификатор сотрудника, выполнившего действие с позицией меню	FK	1

Таблица 10 – Атрибуты таблицы product_requests

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
request_id	SERIAL	Идентификатор учетной записи	PK	1
employee_id	INTEGER	Идентификатор сотрудника, начавшего работу с заявкой	FK	1
date_request	TIMESTAMP	Дата создания заявки		2024-10-10 00:00
date_complete	TIMESTAMP	Дата обработки заявки		2024-10-10 00:00
request_status	TEXT	Статус заявки		Отклонен

Таблица 11 – Атрибуты таблицы product_remains

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
position_id	INTEGER	Идентификатор остатков блюда	FK	1
employee_id	INTEGER	Идентификатор сотрудника, внесшего сведения об остатках	FK	1
date_manufacture	DATE	Дата производства блюда		2024-10-10
capacity	DECIMAL(10,2)	Объем произведенного блюда		100,00
current_remains	DECIMAL(10,2)	Текущий остаток блюда		100,00

Таблица 12 – Атрибуты таблицы requested

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
request_id	INTEGER	Идентификатор заявки	FK	1
product	TEXT	Название запрашиваемого продукта		Морковь
unit_measurement	TEXT	Единица измерения		кг
request_count	DECIMAL(10,2)	Запрашиваемое количество продукта		100,00
purchase_count	DECIMAL(10,2)	Приобретенное количество продукта		100,00

Таблица 13 – Атрибуты таблицы invoices

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
invoice_id	SERIAL	№ накладной	PK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
provider_id	INTEGER	Идентификатор поставщика товара	FK	1
employee_id	INTEGER	Идентификатор сотрудника, обработавшего заявку	FK	1
date_document	TIMESTAMP	Дата создания накладной		2024-10-10 00:00
document_status	TEXT	Статус обработки накладной		Закрыт

Таблица 14 – Атрибуты таблицы product

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
invoice_id	SERIAL	Идентификатор накладной	FK	1
product_name	TEXT	Название продукта		Морковь
product_price	DECIMAL(10,2)	Цена продукта		100,00
product_count	DECIMAL(10,2)	Количество купленного продукта		100,00

Таблица 15 – Атрибуты таблицы providers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
provider_id	SERIAL	Идентификатор поставщика	PK	1
name	TEXT	Наименование поставщика		Магазин «Ермак»
phone	TEXT	Телефон поставщика		+79271234567
contact_person	TEXT	Контактное лицо		Мария
locality_index	INTEGER	Индекс поставщика		442373

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
region	TEXT	Регион поставщика		Пензенская обл.
area	TEXT	Название района поставщика		Каменский р-н
locality	TEXT	Название населенного пункта поставщика		Каменка
street	TEXT	Название улицы поставщика		ул. Центральная
house_number	INTEGER	Номер дома		8
apartament_number	INTEGER	Номер квартиры		10

Таблица 16 – Атрибуты таблицы provider_details

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
detail_id	SERIAL	Идентификатор реквизитов поставщика	PK	1
provider_id	INTEGER	Наименование поставщика, с которым связаны реквизиты	FK	1
organization_type	TEXT	Тип организации поставщика		ИП
inn	INTEGER	ИНН поставщика		580045424
kpp	INTEGER	КПП поставщика		8793834
bank_account	TEXT	Расчетный счет поставщика		4800012674863949
bik	TEXT	БИК банка получателя		54424321
okpo	TEXT	ОКПО поставщика		8788757

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
okato	TEXT	ОКАТО поставщика		9784734

Физическая модель базы данных приведена на рисунке 34.

На рисунке 35 изображена логическая схема базы данных.

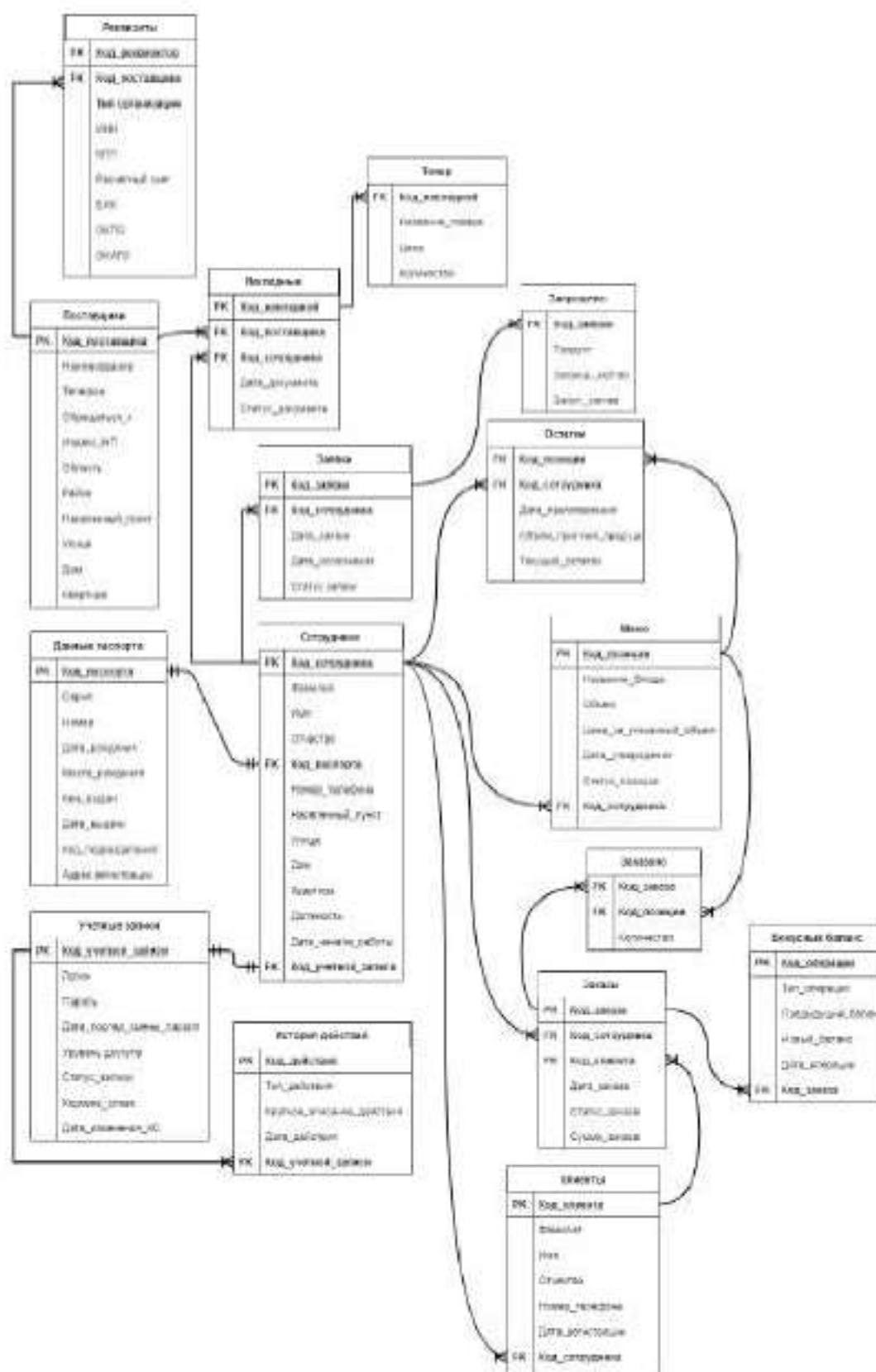


Рисунок 35 – Логическая схема БД

1.4 Разработка приложения

1.4.1 Реализация БД

Перед началом реализации БД её необходимо создать с помощью SQL-запроса:

```
CREATE DATABASE IF NOT EXISTS cafe;
```

Затем создадим внутри БД необходимые таблицы. В данном разделе будет описан процесс создания и работа с основными таблицами БД.

Для создания таблицы, которая будет хранить историю действий в системе, напомним следующий запрос:

```
CREATE TABLE IF NOT EXISTS history (  
    action_id SERIAL,  
    action_type TEXT,  
    action_description TEXT,  
    action_date TIMESTAMP,  
    user_id INTEGER,  
    FOREIGN KEY (user_id) REFERENCES users(userid)  
);
```

Для создания таблицы, которая будет хранить информацию об учетных записях сотрудников, напомним следующий запрос:

```
CREATE TABLE IF NOT EXISTS users (  
    userId SERIAL PRIMARY KEY,  
    login CHARACTER VARYING(30),  
    pass CHARACTER VARYING(30),  
    date_upd_pass TIMESTAMP,  
    privilege_level CHARACTER VARYING(15),  
    status CHARACTER VARYING(30),  
    secret_word CHARACTER VARYING(30),  
    date_upd_word TIMESTAMP  
);
```

Для создания таблицы, которая будет хранить паспортные данные сотрудников, напомним следующий запрос:

```
CREATE TABLE IF NOT EXISTS employee_passport (  
    passport_id SERIAL PRIMARY KEY,  
    passport_series INTEGER,  
    passport_number INTEGER,
```

```

date_birthday DATE,
place_birth TEXT,
place_issue TEXT,
date_issue DATE,
departament_code TEXT,
registration_address TEXT
);

```

Для создания таблицы, которая будет хранить общие сведения о сотрудниках, напомним следующий запрос:

```

CREATE TABLE IF NOT EXISTS employees (
    employee_id SERIAL PRIMARY KEY,
    surname TEXT,
    name TEXT,
    patronymic TEXT,
    passport_id INTEGER,
    phone TEXT,
    locality TEXT,
    street TEXT,
    house_number INTEGER,
    apartament_number INTEGER,
    current_position TEXT,
    date_work_experience DATE,
    user_id INTEGER,
    FOREIGN KEY(user_id) REFERENCES users(userid),
    FOREIGN KEY(passport_id) REFERENCES
employee_passport(passport_id)
);

```

Для создания таблицы, которая будет хранить информацию о посетителях заведения, напомним следующий запрос:

```

CREATE TABLE IF NOT EXISTS clients (
    client_id SERIAL PRIMARY KEY,
    surname TEXT,
    name TEXT,
    patronymic TEXT,
    phone TEXT,
    date_reg TIMESTAMP,
    employee_id INTEGER,
    FOREIGN KEY(employee_id) REFERENCES employees(employee_id)
);

```

Для создания таблицы, которая будет хранить информацию о заказах клиентов, напомним следующий запрос:

```

CREATE TABLE IF NOT EXISTS orders (
    order_id SERIAL PRIMARY KEY,
    employee_id INTEGER,
    client_id INTEGER,
    date_order TIMESTAMP,
    order_status TEXT,
    total DECIMAL(10,2),
    FOREIGN KEY(employee_id) REFERENCES employees(employee_id),
    FOREIGN KEY(client_id) REFERENCES clients(client_id)
);

```

Для создания таблицы, которая будет хранить информацию о меню заведения, напомним следующий запрос:

```

CREATE TABLE IF NOT EXISTS menu (
    position_id SERIAL PRIMARY KEY,
    position_name TEXT,
    unit_measurement TEXT,
    minimum_volume DECIMAL(10,2),
    price DECIMAL(10,2),
    date_create_position TIMESTAMP,
    position_status TEXT,
    employee_id INTEGER,
    FOREIGN KEY(employee_id) REFERENCES employees(employee_id)
);

```

Для создания таблицы, которая будет хранить информацию о заявках на продукты, напомним следующий запрос:

```

CREATE TABLE IF NOT EXISTS product_requests (
    request_id SERIAL PRIMARY KEY,
    employee_id INTEGER,
    date_request TIMESTAMP,
    date_complete TIMESTAMP,
    request_status TEXT,
    FOREIGN KEY(employee_id) REFERENCES employees(employee_id)
);

```

Для создания таблицы, которая будет хранить общие сведения о поставщике продуктов, напомним следующий запрос:

```

CREATE TABLE IF NOT EXISTS providers (
    provider_id SERIAL PRIMARY KEY,
    name TEXT,
    phone TEXT,
    contact_person TEXT,
    locality_index INTEGER,

```

```

        region TEXT,
        area TEXT,
        locality TEXT,
        street TEXT,
        house_number INTEGER,
        apartment_number INTEGER
    );

```

Для создания таблицы, которая будет хранить информацию об операциях с бонусным балансом клиента заведения:

```

CREATE TABLE IF NOT EXISTS bonus_balance (
    operation_id SERIAL PRIMARY KEY,
    operation_type TEXT,
    old_balance DECIMAL(10,2),
    new_balance DECIMAL(10,2),
    date_operation TIMESTAMP,
    order_id INTEGER,
    FOREIGN KEY(order_id) REFERENCES orders(order_id)
);

```

Для добавления данных в таблицу использовался оператор INSERT. Например, следующий запрос добавляет в таблицу History информацию о совершенном пользователем действии:

```

INSERT INTO history (action_type, action_description,
                    action_date, user_id)
VALUES ('Вход в систему',
        'Пользователь @gordeeva вошла в систему',
        to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD
HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS'),
        3);

```

Для добавления данных в таблицу использовался оператор UPDATE. Например, следующий запрос изменяет в таблице Employees_pasport сведения о номере паспорта сотрудника с логином @verhovsky:

```

UPDATE employees_passport
SET passport_number=653131
WHERE login='verhovsky';

```

Для удаления данных из таблицы использовался оператор DELETE. Например, следующий запрос удаляет в таблице users сведения об учетной записи сотрудника с логином @gordeeva:

```
DELETE FROM users
WHERE login='gordeeva';
```

Для создания триггера, который будет автоматически вычислять общую стоимость заказа в таблице orders на основе данных из таблицы ordered, сначала создадим функцию для обновления общей стоимости заказа:

```
CREATE OR REPLACE FUNCTION update_total_price()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE orders
    SET total_price = (
        SELECT SUM(position_count * price)
        FROM ordered
        WHERE order_id = NEW.order_id
    )
    WHERE orders_id = NEW.order_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Далее был создан триггер, который вызывает функцию update_total_price после каждого добавления, обновления или удаления записи в таблице ordered:

```
CREATE TRIGGER update_total_price_trigger
AFTER INSERT OR UPDATE OR DELETE ON ordered
FOR EACH ROW
EXECUTE FUNCTION update_total_price();
```

1.4.2 Описание программы

Описание клиентской части

В папке app/logs содержатся текстовые файлы, содержащие сведения об ошибках, которые могут возникать в работе приложения. Эти файлы нужны для повышения стабильности работы программы и устранения критических ошибок.

В папке app/orders в качестве резервной копии хранится текстовая версия принятых заказов.

В папке app/src/menu_icons хранятся изображения размером 40x40, необходимые для отображения иконки кнопок выбора действия в системе.

Папка app/src/tab_icons содержит картинки, имеющие размер 30x30, которые отображаются в заголовке открытой вкладки.

В папке app/src/program_files располагаются файлы-классы, описывающие модель базы данных.

Файл CurrentUser.java представляет из себя модель таблицы БД users. Данный класс предназначен для работы с информацией о пользователе, вошедшем в систему, а также для просмотра сведений о других учетных записях.

```
public class CurrentUser {
    private int userId;
    private String login;
    private String pass;
    private String dateUpdPass;
    private String privilegeLevel;
    private String status;
    private String secretWord;
    private String dateUpdWord;

    public CurrentUser(int userId, String login, String pass,
String dateUpdPass,
String privilegeLevel, String
status, String secretWord, String dateUpdWord)
    {
        this.userId = userId;
        this.login = login;
        this.pass = pass;
        this.dateUpdPass = dateUpdPass;
        this.privilegeLevel = privilegeLevel;
        this.status = status;
        this.secretWord = secretWord;
        this.dateUpdWord = dateUpdWord;
    }

    public String getPrivilegeLevel() {
        return privilegeLevel;
    }

    public int getUserId() {
        return userId;
    }

    public String getLogin() {
        return login;
    }
}
```

```

    public String getPassword() {
        return pass;
    }

    public String getDateUpdPass() {
        return dateUpdPass;
    }

    public String getStatus() {
        return status;
    }

    public String getSecretWord() {
        return secretWord;
    }

    public String getDateUpdWord() {
        return dateUpdWord;
    }

    public void setSecretWord(String word) {
        secretWord = word;
    }

    public void setNewPassword(String pass) {
        this.pass = pass;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}

```

Файл EmployeeData.java является классом, описывающим сущность employees БД, и содержит в себе методы, необходимые для получения и изменения общих сведений о сотруднике заведения.

```

public class EmployeeData {
    private int employeeId;
    private String surname;
    private String name;
    private String patronymic;
    private int passportId;
    private String phone;
    private String locality;
    private String street;
    private int houseNumber;
    private int apartmentNumber;
    private String currentPosition;
}

```

```

        private String dateWorkExperience;
        private int userId;
        private String[] passportData;

        public EmployeeData(int employeeId, String surname, String
name, String patronymic, int passportId,
        String phone, String locality, String street,
        int houseNumber, int apartmentNumber,
        String currentPosition, String dateWorkExperience,
        int userId, String[] passportData)
        {
            this.employeeId = employeeId;
            this.surname = surname;
            this.name = name;
            this.patronymic = patronymic;
            this.passportId = passportId;
            this.phone = phone;
            this.locality = locality;
            this.street = street;
            this.houseNumber = houseNumber;
            this.apartmentNumber = apartmentNumber;
            this.currentPosition = currentPosition;
            this.dateWorkExperience = dateWorkExperience;
            this.userId = userId;
            this.passportData = passportData;
        }

        public void setSurname(String surname) {
            this.surname = surname;
        }

        public void setName(String name) {
            this.name = name;
        }

        public void setPatronymic(String patronymic) {
            this.patronymic = patronymic;
        }

        public void setPhone(String phone) {
            this.phone = phone;
        }

        public void setLocality(String locality) {
            this.locality = locality;
        }

        public void setStreet(String street) {
            this.street = street;
        }

        public void setHouseNumber(int house) {
            houseNumber = house;
        }
    
```

```

    }

    public void setApartamentNumber(int apart) {
        apartamentNumber = apart;
    }

    public void setSurname(intapartament) {
        apartamentNumber = apartament;
    }

    public void setCurrentPosition(String position) {
        currentPosition = position;
    }

    public void setDateWorkExperience(String date) {
        dateWorkExperience = date;
    }

    public intgetEmployeeId() {
        return employeeId;
    }

    public String getName() {
        return name;
    }

    public String getSurname() {
        return surname;
    }

    public String getPatronymic() {
        return patronymic;
    }

    public intgetPassportId() {
        return passportId;
    }

    public String getPhone() {
        return phone;
    }

    public String getLocality() {
        return locality;
    }

    public String getStreet() {
        return street;
    }

    public intgetHouseNumber() {
        return houseNumber;
    }

```

```

    public int getApartamentNumber() {
        return apartamentNumber;
    }

    public String getCurrentPosition() {
        return currentPosition;
    }

    public String getDateWorkExperience() {
        return dateWorkExperience;
    }

    public int getUserId() {
        return userId;
    }

    public String[] getPassportData() {
        return passportData;
    }
}

```

Файл History.java хранит информацию обо всех действиях пользователей в системе.

```

public class History {
    private int actionId;
    private String actionType;
    private String actionDescription;
    private String actionDate;
    private String login;
    private int countActionsDisplay;
    private ArrayList<Integer> sortedHistory;

    public History(int actionId, String actionType, String
actionDescription, String actionDate, String login) {
        this.actionId = actionId;
        this.actionType = actionType;
        this.actionDescription = actionDescription;
        this.actionDate = actionDate;
        this.login = login;
    }

    public History() {
        countActionsDisplay = 30;
        sortedHistory = new ArrayList<>();
    }

    public int getActionId() {
        return actionId;
    }
}

```

```

    public String getActionType() {
        return actionType;
    }

    public String getActionDescription() {
        return actionDescription;
    }

    public String getActionDate() {
        return actionDate;
    }

    public String getLogin() {
        return login;
    }

    public void setCountActionsDisplay(int count) {
        countActionsDisplay = count;
    }

    public int getCountActionsDisplay() {
        return countActionsDisplay;
    }

    public ArrayList<Integer> getSortedHistory() {
        return sortedHistory;
    }
}

```

Содержит не только стандартные методы получения и изменения содержимого полей класса, но и специальные методы.

Метод `addNewItemIntoSortedHistory(int index)` хранит действия в системе, отсортированные по типу действия.

```

public void addNewItemIntoSortedHistory(int index) {
    sortedHistory.add(index);
}

```

Метод `removeItemSortedHistory(int index)` удаляет действия в системе, для которых было снято выделение.

```

public void removeItemSortedHistory(int index) {
    for (int i=0; i<sortedHistory.size(); i++) {
        if (sortedHistory.get(i) == index) {
            sortedHistory.remove(i);
        }
    }
}

```

```

    }
}

```

Файл SearchEmployees.java хранит информацию о найденных в системе сотрудниках.

```

public class SearchEmployees {
    private String surname;
    private String name;
    private String patronymic;
    private String phone;
    private String currentPosition;

    public SearchEmployees(String surname, String name, String
patronymic, String phone, String currentPosition) {
        this.surname = surname;
        this.name = name;
        this.patronymic = patronymic;
        this.phone = phone;
        this.currentPosition = currentPosition;
    }

    public String getSurname() {
        return surname;
    }

    public String getName() {
        return name;
    }

    public String getPatronymic() {
        return patronymic;
    }

    public String getPhone() {
        return phone;
    }

    public String getCurrentPosition() {
        returncurrentPosition;
    }
}

```

Файл Main.java представляет из себя класс, реализующий графический интерфейс приложения и основную логику работы.

Метод saveTabs() выполняет сохранение вкладок графического интерфейса в коллекцию Map.

```

public void saveTabs() {
    for (inti=0; i< jTabledPane2.getTabCount(); i++) {
        tabS.put(jTabledPane2.getTitleAt(i),
jTabledPane2.getComponentAt(i));
    }
}

```

Метод checkConnection() проверяет наличие подключения к серверу.

```

public void checkConnection() {
    try {
        Socket checkConn = new Socket("localhost", 8000);
        DataOutputStream out = new
DataOutputStream(checkConn.getOutputStream());
        out.writeInt(500);
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, "Отсутствует подключение к
серверу!\nПроверьте подключение и перезапустите программу.",
"Ошибка соединения", JOptionPane.ERROR_MESSAGE);
        System.exit(0);
    }
}

```

Метод auth() осуществляет проведение входа в систему.

```

public String auth(String login, String pass) {
    if(!(login.equals("") || pass.equals(""))) {
//Если все поля заполнены

        try {
            clientSocket = new Socket("localhost", 8000); //Подключение к
серверу
            outputStream = new
DataOutputStream(clientSocket.getOutputStream());
            inputStream = new DataInputStream(clientSocket.getInputStream());

            outputStream.writeInt(10); //Отправка кода запроса
            outputStream.writeUTF(login); //Отправка данных для входа в
систему
            outputStream.writeUTF(pass);

            response = inputStream.readUTF();
//Чтение кода ответа сервера

            switch (response) { //Анализ ответа сервера
                case "authOK" -> {
                    intcount = inputStream.readInt(); //Количество строк по
результатам запроса к БД

                    for (inti = 0; i< count; i++) {
                        requestResult.add(inputStream.readUTF());

```

```

    }

    String[] data = requestResult.get(0).split("/");
    //Данные учетной записи
    String[] dataUpdPass = data[3].split("\\.");
    String[] dataUpdWord = data[7].split("\\.");

    //Сохранение информации о
    текущем пользователе
    currentUser = new
    CurrentUser(Integer.parseInt(data[0]), data[1], data[2],
    dataUpdPass[0],
    data[4], data[5], data[6], dataUpdWord[0]);

    //Аудит входа в систему
    outputStream.writeInt(currentUser.getUserId());

    //Получение информации из профиля сотрудника
    String[] employeeData = inputStream.readUTF().split("/");

    //Получение паспортных данных сотрудника
    String[] passportData = inputStream.readUTF().split("/");

    myProfile = new EmployeeData(Integer.parseInt(employeeData[0]),
    employeeData[1], employeeData[2], employeeData[3];

    return "authSuccess";
    }
    case "userNotExistError" -> {
        return "userNotExistError";
    }
    case "uncorrectLoginOrPasswordError" -> {
return "uncorrectLoginOrPasswordError";
    }
    case "userBlockedError" -> {
        return "userBlockedError";
    }
    default -> {
    }
    }

else {
    return "emptyFieldError";
}
return "unknownError";
}

```

Метод createNewEmployee() осуществляет регистрацию нового сотрудника в системе.

```

        public void createNewEmployee() {
            try {
                clientSocket = new Socket("localhost", 8000); //Подключение к
серверу
                outputStream = new
DataOutputStream(clientSocket.getOutputStream());
                inputStream = new DataInputStream(clientSocket.getInputStream());

                outputStream.writeInt(16); //Отправкакодазапроса
                outputStream.writeInt(currentUser.getUserId());
                //Отправканомераучетнойзаписи

                outputStream.writeUTF(passport); //Отправкаданных
                outputStream.writeUTF(employeeData);
                outputStream.writeUTF(userData);

                response = inputStream.readUTF();
                //Чтениеответаназапрос
                outputStream.writeInt(currentUser.getUserId());
                //Аудитвнесениясведенийосотруднике

                switch(response) {
                    case "loginNotAvailable" -> {
                        JOptionPane.showMessageDialog(this, "Логин @" +
jTextField70.getText().trim() + " ужезанятдругимсотрудником!\n"
+ "Пожалуйста, введите другой свободный логин.",
"Ошибкарегистрации", JOptionPane.ERROR_MESSAGE);
                    }
                    case "passportNotAvailable" -> {
                        String login = inputStream.readUTF();
                        JOptionPane.showMessageDialog(this, "Паспорт с серией " +
jTextField58.getText().trim() + " и №"
+ jTextField59.getText().trim() + " принадлежит сотруднику с
логином @" + login + "!\n"
+ "Пожалуйста, проверьте
корректность введенных данных.",
"Ошибкарегистрации", JOptionPane.ERROR_MESSAGE);
                    }
                    case "phoneNotAvailable" -> {
                        String login = inputStream.readUTF();
                        JOptionPane.showMessageDialog(this, "Номертелефона +7" +
jTextField53.getText().trim() + " ужеиспользуется " +
"и принадлежит сотруднику с логином @" + login + "!\n"
+ "Пожалуйста, введите другой номер
телефона.",
"Ошибкарегистрации", JOptionPane.ERROR_MESSAGE);
                    }
                    case "success" -> {
                        JOptionPane.showMessageDialog(this,
"Сотрудникуспешнозарегистрирован!",
"Регистрациязавершена",
JOptionPane.INFORMATION_MESSAGE);
                    }
                }
            }
        }
    }

```

```

.setText("");
jCheckBox7.setSelected(false);
jComboBox1.setSelectedItem("Активен");
jComboBox2.setSelectedItem("Пользователь");
}

```

Метод `validateEmployeeData()` осуществляет проверку данных сотрудника.

```

public String validateEmployeeData() {
    boolean fieldsNotEmpty =
        jTextField50.getText().trim().isEmpty() ||
        jTextField51.getText().trim().isEmpty() ||
        if(fieldsNotEmpty) { //Если не заполнена хотя бы одно
поле
        return "emptyFieldsError";
        }

        if(jTextField53.getText().trim().length() < 10 ||
jTextField53.getText().trim().length() > 10) { //Проверка
корректности длины номера телефона
        return "phoneLengthError";
        }

        try {
            Integer.parseInt(jTextField53.getText().trim().substring(0,
5)); //В номере присутствуют символы
            Integer.parseInt(jTextField53.getText().trim().substring(5,
10));
        } catch (NumberFormatException ex) {
            return "incorrectPhoneError";
        }

        try {
            Integer.parseInt(jTextField59.getText().trim());
//Проверка номера паспорта
        } catch (NumberFormatException ex) {
            return "passportNumberError";
        }

        return "validateSuccess";
    }
}

```

Метод `restorePassowrd(String secretWord, String login, String newPassword)` реализует функцию сброса пароля:

```

public String restorePassword(String secretWord, String login,
String newPassword) {
    try {

```

```

        clientSocket = new Socket("localhost", 8000); //Подключение к
серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(21); //Отправка кода запроса
        outputStream.writeUTF(login); // Отправка логина
        outputStream.writeUTF(secretWord); //Отправка кодового слова
        outputStream.writeUTF(newPassword); // Отправка нового пароля

        response = inputStream.readUTF(); //Чтение ответа

        switch(response) {
            case "user_not_found" -> {
                return "user_not_found";
            }
            case "incorrectSecretWord" -> {
                return "incorrectSecretWord";
            }
        }
    }
}

```

Метод `createClient(String secretWord, String login, String newPassword)` реализует функцию сброса пароля:

```

public void createClient() {
    if(jCheckBox8.isSelected()) {
        if(jTextField1.getText().trim().isEmpty()) {
            JOptionPane.showMessageDialog(this, "Введите номер телефона",
"Ошибка регистрации", JOptionPane.ERROR_MESSAGE);
            return;
        }
        try {
            clientSocket = new Socket("localhost", 8000); //Подключение к
серверу
            outputStream = new
DataOutputStream(clientSocket.getOutputStream());
            inputStream = new DataInputStream(clientSocket.getInputStream());

            outputStream.writeInt(23); //Отправка кода запроса
            outputStream.writeInt(currentUser.getUserId()); // Отправка номера
сотрудника, выполнившего регистрацию

            data = jTextField1.getText().trim() + "/нет/нет/нет";
//Отправка пользовательских данных
            outputStream.writeUTF(data); //Отправка кода сотрудника, чья
учетная запись будет заблокирована

```

```

        response = inStream.readUTF(); //Получение ответа сервера
        if(response.equals("phoneUsedError")) {
            JOptionPane.showMessageDialog(this, "Номер телефона уже
используется другим клиентом!", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        JOptionPane.showMessageDialog(this, "Клиент успешно
зарегистрирован!", "Регистрация завершена",
JOptionPane.INFORMATION_MESSAGE);
        jTextField1.setText("");
    }
}

```

Метод `getEmployeeList()` позволяет получить список сотрудников, зарегистрированных в системе.

```

public void getEmployeesList() {
    Socket clientSocket = null;
    DataOutputStream outStream = null;
    DataInputStream inStream = null;
    String response;
    int countRows;

    try {
        clientSocket = new Socket("localhost", 8000); //Подключение к
серверу
        outStream = new
DataOutputStream(clientSocket.getOutputStream());
        inStream = new DataInputStream(clientSocket.getInputStream());

        outStream.writeInt(20); //Отправка кода запроса

        countRows = inStream.readInt(); //Чтение количества записей

        searchEmployees.clear(); //Очистка списка найденных сотрудников

        for (inti=0; i<countRows; i++) {
            String result = inStream.readUTF();
            String[] data = result.split("/");
            //Считываем строку с результатами
            searchEmployees.add(new SearchEmployees(data[0], data[1],
data[2], "+7" + data[3], data[4]));
        }

        for (inti=0; i<searchEmployees.size(); i++) {
            DefaultTableModel model = (DefaultTableModel)
jTable3.getModel();

```

```

        model.addRow(new Object[] {searchEmployees.get(i).getSurname(),
searchEmployees.get(i).getName(),
        searchEmployees.get(i).getPatronymic(),
searchEmployees.get(i).getPhone(),
        searchEmployees.get(i).getCurrentPosition()});
    }
}

```

Метод `logoutSystemAudit()` сохраняет историю выходов сотрудника из системы.

```

public void logoutSystemAudit() {
    try {
        clientSocket = new Socket("localhost", 8000); //Подключение к
серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(1); //Отправкакодазапроса

        outputStream.writeInt(currentUser.getUserId());
//Отправканомераучетнойзаписи
    }
}

```

Остальные методы класса `Main.java` приведены в приложении Б (листинг программы).

Описание серверной части

В папке `server/logs` хранится история ошибок, возникших в процессе работы сервера. Это необходимо для повышения стабильности работы сервера в будущем.

Папка `server/program_files` хранятся файлы программы-сервера.

В файле `Server.java` представляет из себя основной класс сервера, который устанавливает соединение с клиентской частью программы, создавая новый поток для обработки каждого подключения. Сервер ожидает подключения клиента на порту 8000.

```

public class Server {
    public static void main(String[] args) {
        ServerSocketserverSocket;
        try {

```

```

serverSocket = new ServerSocket(8000);
serverSocket.setReuseAddress(true);

while (true) {
    Socket clientSocket = serverSocket.accept();
//Ожиданиеподключенияклиента
    System.out.println("New client connected " +
clientSocket.getInetAddress().getHostAddress());

    ClientHandler client = new ClientHandler(clientSocket);

    new Thread(client).start();

}

}
catch(IOException ex) {
System.out.println("Connection error" + ex);
}
}
}

```

Файл ClientHandler.java содержит методы для работы с данными, полученными от клиентской части. Также в методах реализован функционал запросов к БД, получение и обработка результатов.

Метод auth() осуществляет проведение входа в систему.

```

public void auth() {
    try {
        String data;

        String login = inStream.readUTF();
        String password = inStream.readUTF();
        rs = stmt.executeQuery("SELECT * FROM users WHERE login='" +
login + "'");
        int count = 0;

        if(!rs.next()) {
            outputStream.writeUTF("userNotExistError"); //Отправка кода ответа
клиенту, если пользователя не существует
            outputStream.close();
            inStream.close();
            clientSocket.close();
            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();

            return;
        }

        rs.beforeFirst();
    }
}

```

```

        while (rs.next()) {
            if (!(rs.getString("login").equals(login)
&&rs.getString("pass").equals(password))) {
                //Еслилогинилипарольневерные
                outputStream.writeUTF("uncorrectLoginOrPasswordError");
                inputStream.close();
                outputStream.close();
                clientSocket.close();
                return;
            }
            else
if(rs.getString("status").equals("Заблокирован")) {
                //Если учетная запись пользователя временно заблокирована
                outputStream.writeUTF("userBlockedError");
                inputStream.close();
                outputStream.close();
                clientSocket.close();
                return;
            }
            else {
                //Иначе отправляем результат запроса
клиенту
                data = rs.getInt("userId");
                rs.getTimestamp("date_upd_word");
                count++;
            }

            outputStream.writeUTF("authOK");
            outputStream.writeInt(count);
            outputStream.writeUTF(data);
        }

        //Аудит входа в систему
        int userID = inputStream.readInt();
        String username = "";

        rs = stmt.executeQuery("SELECT login FROM users WHERE userId="
+ userID); //Получаемлогинпользователя
        while (rs.next()) {
            username = rs.getString("login");
        }

        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
+ "VALUES ('Вход в систему', 'Пользователь @" + username + "
вошел в систему', "
+ "to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD
HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS'), "
+ userID + ")");

        //Получениеинформации о сотруднике
        rs = stmt.executeQuery("SELECT * FROM employees WHERE user_id="
+ userID);

```

```

        String employeeData = "";
        while (rs.next()) {
            employeeData = rs.getInt("employee_id");
        }
        outputStream.writeUTF(employeeData);

        //Получение паспортных данных сотрудника
        String[] buf = employeeData.split("/");
        String passportId = buf[4];
        String passportData = "";
        rs = stmt.executeQuery("SELECT * FROM employee_passport WHERE
passport_id=" + passportId);
        while (rs.next()) {
            passportData = rs.getInt("passport_id");
            outputStream.writeUTF(passportData);
        }
    }

```

Метод `logoutSystemAudit()` сохраняет запись о выходе сотрудника из системы.

```

public void logoutSystemAudit() {
    try {
        //Аудит выхода из системы
        int userID = inputStream.readInt();
        String username = "";
        rs = stmt.executeQuery("SELECT login FROM users WHERE userID="
+ userID); //Получаем логин пользователя
        while (rs.next()) {
            username = rs.getString("login");
        }
        stmt.executeUpdate("INSERT INTO history (action_type,
action description, action date, user id)"
+ "VALUES ('Выход из системы', 'Пользователь @" + username + "
вышел из системы', "
+ "to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD
HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS'), "
+ userID + ")");
    }
}

```

Метод `getAllHistory()` осуществляет отправку истории всех действий

```

public void getAllHistory() {
    int count = 0;

    String data;

    ArrayList<String> result = new ArrayList<>();

    try {
        int limit = inputStream.readInt(); //Чтение количества отображаемых
записей
    }
}

```

```

        rs = stmt.executeQuery("SELECT history.action_id,
history.action_type, history.action_description,
history.action_date, users.login " +
                                "FROM history, users " +
                                "WHERE history.user_id =
users.userid " +
                                "ORDER BY action_date DESC
LIMIT " + limit);
        while (rs.next()) {
            data = rs.getInt("action_id") + "/" +
rs.getString("action_type") + "/" +
rs.getString("action_description") + "/" +
rs.getString("action_date") + "/" +
rs.getString("login");
            result.add(data);
            count++;
        }
        outStream.writeInt(count); //Записьчислаполученныхстрок
        for(String item : result) {
            outStream.writeUTF(item);
        }
    }

```

Метод `restorePassword()` осуществляет восстановление доступа к системе.

```

public void restorePassword() {
    try {
        String login;
        String secretWord;
        String newPassword;
        String wordFromDB = "";
        int user_id = 0;

        login = inStream.readUTF(); // Чтение логина пользователя,
        // который желает изменить пароль
        secretWord = inStream.readUTF(); // Чтение кодового слова
        newPassword = inStream.readUTF(); // Чтение нового пароля

        rs = stmt.executeQuery("SELECT * FROM users WHERE login='" +
login + "'");
        if(!rs.next()) {
            outStream.writeUTF("user_not_found");
            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();
            return;
        }

        rs.beforeFirst();
        while(rs.next()) {
            wordFromDB = rs.getString("secret_word");
        }
    }
}

```

```

        if(!wordFromDB.equals(secretWord)) {
outStream.writeUTF("incorrectSecretWord");
rs.close(); //Освобождение ресурсов
stmt.close();
conn.close();

        return;
    }
    rs = stmt.executeQuery("SELECT userid FROM users WHERE login='"
+ login + "'");
    while(rs.next()) {
        user_id = rs.getInt("userid");
    }

    stmt.executeUpdate("UPDATE users SET pass='" + newPassword +
    "', date_upd_pass=to_timestamp(to_char(current_timestamp, 'YYYY-MM-
    DD HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS') WHERE login='" + login +
    "'"); //Обновление пароля

```

Остальные методы класса ClientHandler.java приведены в приложении Б.

1.4.3 Руководство пользователя

Чтобы начать работу с программой, необходимо пройти авторизацию в системе (рисунок 36). Для этого в поля «Логин» и «Пароль» введем учетные данные пользователя, после чего нажмем кнопку «Войти».

Рисунок 36 – Окно авторизации в системе

При попытке входа с хотя бы одним незаполненным полем на экране появится предупреждающее уведомление (рисунок 37).

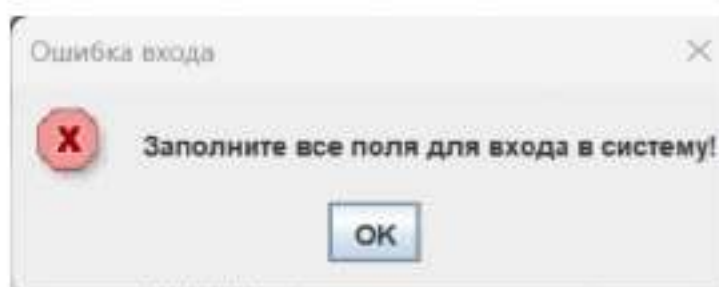


Рисунок 37 – Попытка входа с незаполненными полями

Если попытаться войти в профиль пользователя с несуществующим логином, появится соответствующее окно (рисунок 38).

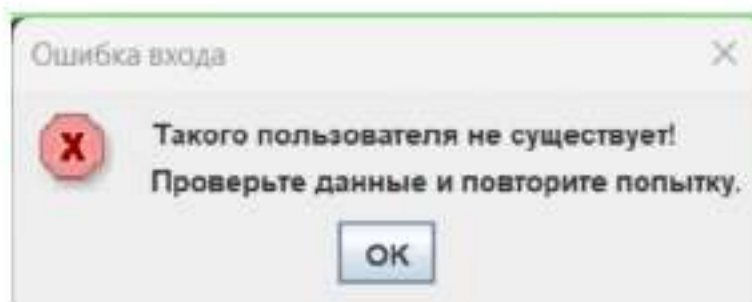


Рисунок 38 – Ошибка входа

Если ввести неправильный логин или пароль, появится уведомление с просьбой проверить данные и предложение воспользоваться функцией восстановления пароля (рисунок 39).

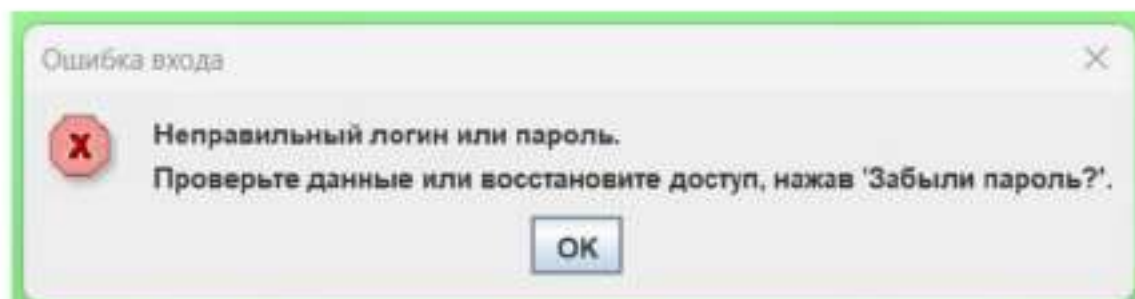


Рисунок 39 – Предложение восстановить пароль

Если авторизация успешно пройдена, на экране появится соответствующее уведомление (рисунок 40).

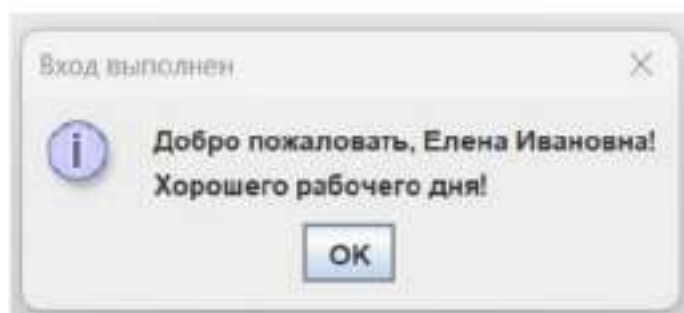


Рисунок 40 – Успешная авторизация

Пользователю доступна возможность восстановления доступа к системе без помощи администратора. Для этого достаточно знать кодовое слово, которое было присвоено учетной записи в момент ее создания.

Чтобы приступить к восстановлению доступа, в поле «Логин» необходимо ввести логин, доступ к которому требуется восстановить, а затем нажать кнопку «Забыли пароль?». Откроется окно, в котором требуется ввести корректное кодовое слово, новый пароль и его подтверждение (рисунок 41).

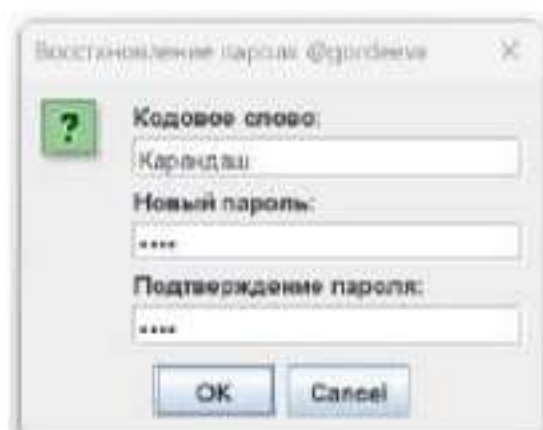


Рисунок 41 – Восстановление пароля

Если кодовое слово, новый и его подтверждение введены корректно, появится уведомление об успешном восстановлении доступа (рисунок 42).



Рисунок 42 – Успешное восстановление доступа

Если кодовое слово неверно, появится сообщение об ошибке восстановления (рисунок 43).



Рисунок 44 – Ошибка восстановления

После успешной авторизации откроется главное меню программы (рисунок 45). Чтобы выполнить нужное действие, необходимо щелкнуть по той или иной кнопке действия.



Рисунок 45 – Главное меню программы

Рассмотрим основные функции и возможности программы. Чтобы внести сведения о новом посетителе заведения, нажмите кнопку «Регистрация клиента». В появившемся окне (рисунок 46) заполните все поля и нажмите кнопку «Зарегистрировать».

Рисунок 47 – Регистрация нового клиента

При желании можно зарегистрировать клиента, указав лишь номер телефона, а внести остальные данные позднее. Для этого нужно поставить галочку в поле «Заполнить ФИО позже» (рисунок 48).

После успешной регистрации появится соответствующее окно.

Рассмотрим возможности раздела «История активности» (рисунок 49). В этом разделе отображается информация обо всех действиях сотрудников в системе. Если совершен вход с учетной записью обычного пользователя, будет доступна история действий только той учетной записи, от имени которой осуществлен вход.

Регистрация нового клиента

Создайте профиль посетителя для учета его активности

Телефон: +7

Описание:

Имя:

Отчество:

☒ Заполнить ФИО позже

Рисунок 48 – Регистрация клиента по номеру телефона

История действий

Здесь хранится информация о ваших действиях в системе

Введите логин сотрудника для просмотра его действий:

Тип действия: ☐ Вход ☐ Выход ☐ Просмотр ☐ Изменение ☐ Анонимизация ☐ Внесение сведений

№ действия	Тип действия	Описание действия	Дата действия	Пользователь
544	Вход в систему	Пользователь @orgname вошел в систему	2024-06-03 14:22:31	admin
543	Изменение сведений	Пользователь @orgname восстановил доступ в систему	2024-06-03 14:22:31	admin
542	Изменение сведений	Пользователь @orgname восстановил доступ в систему	2024-06-03 14:22:31	admin
541	Выход из системы	Пользователь @orgname вышел из системы	2024-06-03 14:14:09	admin
540	Вход в систему	Пользователь @orgname вошел в систему	2024-06-03 14:08:08	admin
539	Выход из системы	Пользователь @orgname вышел из системы	2024-06-24 11:13:54	admin
538	Вход в систему	Пользователь @orgname вошел в систему	2024-06-24 11:10:56	admin
495	Выход из системы	Пользователь @orgname вышел из системы	2024-06-22 11:19:42	admin
494	Вход в систему	Пользователь @orgname вошел в систему	2024-06-22 11:16:21	admin
493	Выход из системы	Пользователь @orgname вышел из системы	2024-06-22 11:16:09	admin
492	Вход в систему	Пользователь @orgname вошел в систему	2024-06-22 11:15:42	admin
325	Выход из системы	Пользователь @orgname вышел из системы	2024-06-19 02:30:31	admin
324	Вход в систему	Пользователь @orgname вошел в систему	2024-06-19 02:30:47	admin
293	Выход из системы	Пользователь @orgname вышел из системы	2024-06-19 10:40:08	admin
289	Просмотр сведений	Пользователь @orgname просмотрел сведения о сотруднике @orgname	2024-06-19 10:40:06	admin
288	Вход в систему	Пользователь @orgname вошел в систему	2024-06-19 10:40:47	admin
287	Просмотр сведений	Пользователь @orgname просмотрел сведения о сотруднике @orgname	2024-06-19 10:37:44	admin
285	Вход в систему	Пользователь @orgname вошел в систему	2024-06-19 00:17:30	admin
283	Вход в систему	Пользователь @orgname вошел в систему	2024-06-19 00:49:51	admin

Рисунок 49 – История действий в системе

Рассмотрим возможности раздела «Мой профиль». В нем собрана вся информация о сотруднике, о его учетной записи, паспортные данные, данные о трудоустройстве и истории принятых заказов (рисунок 50).

Мой профиль

Гордеева Елена Ивановна

Общие сведения

Фамилия: Пашкина

Имя: Елена

Отчество: Ивановна

Телефон: +7 800 123 45 67

ID профиля: 1

Наименование учреждения: 1. Золотое

Типа: ул. Пашкина

№ дома: 8

№ квартиры: 2

Дата регистрации: 2017.10.11

Изменить данные

Паспортные данные

Серия: M24

Номер: 870341

Дата рождения: 11 / 06 / 1978

Место рождения: г. Золотое (ул. Мухоморова) обл. Пензенская

Кем выдан: УФМС России по Пензенской области

Код подразделения: 999 - 011

Дата выдачи: 19 / 10 / 2016

Адрес регистрации: ул. Пензенская для Мухоморов, с. Золотое ул. Пашкина, д. 23, кв. 1

Изменить данные

Сведения об учетной записи

Логин: g0rdeeva

Пароль: 12345678

Кодовое слово: 123456789

Пароль-оповещение: 204-06-01 11-23-45

Дата акт. КС: 2016-05-01 11-34-32

Тип учетной записи: Пользователь

Статус учетной записи: Активен

Изменить пароль

Изменить ID

Изменить МП/участка

Изменить адрес

Сведения о работе

Дата трудоустройства: 2017.10.11

Выполнено заявок: 5

Срок (дней): 0

Занимаемая должность: Владелец

Изменить данные

История принятых заказов

№ заказа	С. заказчика	И.О. заказчика	Телефон заказчика	И.О. исполнителя	Дата заказа	Статус	Сумма
<div> <div>Панель подробнее</div> <div>Отменить заказ</div> <div>Позвонить исполнителю</div> <div>Профиль клиента</div> </div>							

Рисунок 50 – Профиль сотрудника

При желании данные можно отредактировать. Например, давайте изменим кодовое слово сотрудника Гордеева Елена Ивановна. Для этого в блоке «Общие сведения» нажмем кнопку «Изменить данные». Появится уведомление о входе в режим изменения данных (рисунок 51), а поля, доступные для редактирования, станут активными.

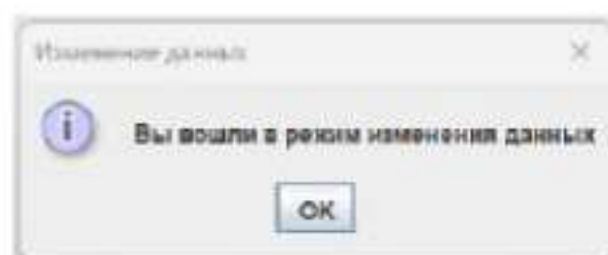


Рисунок 51 – Вход в режим изменения данных

Изменим улицу проживания сотрудника. Для этого введем новое значение в поле «Улица» (рисунок 52). После проверки корректности введенных данных нажмем «Продолжить», появится запрос на подтверждение внесения изменений о

сотруднике (рисунок 53). Подтвердим действие, нажав кнопку «Сохранить». После успешного внесения изменений появится соответствующее сообщение (рисунок 54).

Общие сведения

Фамилия: Населенный пункт:

Имя: Улица:

Отчество: № дома: № квартиры:

Телефон: Дата регистрации:

ID профиля:

Рисунок 52 – Изменение названия улицы проживания сотрудника

Изменение данных

? Вы действительно хотите внести изменения в профиль сотрудника?

Рисунок 53 – Подтверждение внесения изменений

Изменение данных

i Внесенные изменения успешно сохранены

Рисунок 54 – Успешное внесение изменений о сотруднике

Иногда возникает необходимость просмотреть профиль нужного нам сотрудника. Для этого нужно перейти в раздел «Поиск сотрудника» (рисунок 55). В этом разделе по умолчанию в виде таблицы представлен список зарегистрированных в системе сотрудников заведения.

Поиск сотрудника

[Закрыть](#)

Найдите профиль сотрудника и получите всю информацию о нём

Поиск по номеру телефона:

[Найти](#)

Поиск по фамилии:

[Найти](#)

Фамилия	Имя	Отчество	Номер телефона	Должность
Жукова	Мария	Венеславовна	+79074408205	Продавец
Зубов	Евгений	Александрович	+79074408205	Менеджер
Арапчов	Артём	Викторович	+79075652434	Старший
Верховский	Михаил	Александрович	+79074408201	Менеджер
Наваскин	Андрей	Викторович	+79027574311	Охранник
Белогородский	Арина	Викторовна	+79075487790	Кассир
Господкин	Елена	Александровна	+79075442115	Повар

[Открыть профиль](#)

Рисунок 55 – Список зарегистрированных сотрудников

Доступна возможность поиска сотрудника по номеру телефона (рисунок 56) или фамилии (рисунок 57). Для этого достаточно ввести нужные данные в соответствующее поле и нажать кнопку «Найти».

Поиск сотрудника

[Закрыть](#)

Найдите профиль сотрудника и получите всю информацию о нём

Поиск по номеру телефона:

[Найти](#)

Поиск по фамилии:

[Найти](#)

Фамилия	Имя	Отчество	Номер телефона	Должность
Жукова	Мария	Венеславовна	+79074408205	Продавец
Зубов	Евгений	Александрович	+79074408205	Менеджер
Верховский	Михаил	Александрович	+79074408201	Менеджер

[Открыть профиль](#)

Рисунок 56 – Поиск сотрудника по номеру телефона

Поиск сотрудника

Закрыть

найдите профиль сотрудника и получите всю информацию о нем

Поиск по номеру телефона: Найти
Поиск по фамилии: Найти

Фамилия	Имя	Отчество	Номер телефона	Должность
Жукова	Мария	Владиславовна	+7967408285	Продавец

Открыть профиль

Рисунок 57 – Поиск сотрудника по фамилии

Для регистрации нового сотрудника в системе перейдем в разделе «Добавить сотрудника» (рисунок 58).

Создание профиля сотрудника

Отмена

Заполните все поля для внесения сведений о сотруднике

Общие сведения

Фамилия:
Имя:
Отчество:
Телефон: +7

Почтовый адрес:
Почта:
МФ-догов:
МФ-контрагент:
☐ Нет контрагента

Паспортные данные

Серия:
Паспорт:
Дата рождения: /

☒
☐

Место рождения:
Чем занят:
Дата выезда: /

☒
☐
☐

Код подразделения: -
Адрес регистрации:

Сведения об учетной записи

Придумайте логин:
Придумайте пароль:
Придумайте контрольный номер:
Выберите статус: Активен

Выберите тип учетной записи: Пользователь

Сведения о работе

Занимаемая должность:
Дата трудоустройства: /

☒
☐

Завершить регистрацию

Рисунок 58 – Регистрация нового сотрудника

В системе также доступен поиск зарегистрированных клиентов (рисунок 59). При желании можно найти клиента по номеру телефона.

Рисунок 59 – Поиск клиента в системе

1.4.4 Тестирование и отладка

В процессе работы с программой пользователь может допустить некоторые ошибки, которые могут привести к её некорректной работе. Например, пользователь может случайно забыть заполнить некоторые обязательные поля в момент регистрации клиента, либо неправильно ввести его номер телефона, превысив его общую длину или нарушив формат ввода. Для таких случаев в программе предусмотрены обработчики исключительных ситуаций, работу которых требуется проверить путем составления тест-кейсов (таблица 17).

Таблица 17 – Тест-кейсы

Наименование	Описание	Шаги для воспроизведения	Ожидаемый результат
1. Проверка наличия пустых полей во время авторизации в системе	Во время входа в систему пользователю необходимо заполнить поля «Логин» и «Пароль»	1. Запустить программу; 2. Не заполнив хотя бы одно из полей («Логин» или «Пароль»), нажать на кнопку «Войти»	На экране отобразилось уведомление: «Заполните все поля для входа в систему!»
2. Ввод несуществующего имени пользователя	Во время авторизации пользователь должен указать корректный логин и пароль для входа в систему	1. Запустить программу; 2. В поле «Логин» ввести несуществующее имя пользователя; 3. Нажать кнопку «Войти».	На экране появилось уведомление: «Такого пользователя не существует! Проверьте данные и повторите попытку.»
3. Ввод неправильного пароля для входа в систему	Необходимо ввести корректный логин и пароль для успешной авторизации в системе.	1. Запустить программу; 2. В поле «Логин» или «Пароль» ввести некорректные данные; 3. Нажать кнопку «Войти».	На экране появилось уведомление: «Неправильный логин или пароль. Проверьте данные и повторите попытку.»
4. Попытка восстановить пароль без заполнения поля «Логин»	Сначала требуется заполнить поле «Логин», затем можно приступить к восстановлению доступа, нажав кнопку «Забыли пароль?»	1. Запустить программу; 2. Оставить поле «Логин» пустым; 3. Нажать кнопку «Забыли пароль?».	На экране появится уведомление: «Введите логин учетной записи, доступ к которой Вы хотите восстановить!»
5. Ввод неверного кодового слова в момент восстановления доступа к системе	Для восстановления доступа к системе необходимо ввести корректное кодовое слово	1. Запустить программу; 2. Заполнить поле «Логин»; 3. Нажать кнопку «Забыли	Появилось уведомление об ошибке: «Кодовое слово указано неверно! Пожалуйста, повторите

		пароль?»; 4. В появившемся окне ввести неверное кодовое слово.	попытку или обратитесь к администратору.»
6. Ошибка подтверждения нового пароля при попытке восстановить доступ в систему	Для восстановления доступа к системе требуется указать кодовое слово учетной записи, затем нужно ввести новый пароль, после чего ввести его еще раз для подтверждения корректности ввода	1. Запустить программу; 2. Заполнить поле «Логин»; 3. Нажать кнопку «Забыли пароль?»; 4. В появившемся окне ввести в поле «Подтверждение пароля» такой пароль, который будет отличаться от того, что указан в поле «Новый пароль»	На экране появилось уведомление: «Введенные пароли не совпадают! Проверьте данные и повторите попытку.»
7. Попытка зарегистрировать нового клиента с уже используемым номером телефона	Для регистрации нового клиента требуется указать номер телефона, который не используется другим клиентом	1. Запустить программу; 2. Осуществить вход в систему; 3. В главном меню нажать кнопку «Регистрация клиента»; 4. В поле «Номер телефона» ввести уже занятый номер телефона.	На экране появилось уведомление: «Этот номер телефона уже используется другим клиентом! Пожалуйста, укажите другой телефон»
8. Регистрация нового клиента при наличии хотя бы одного незаполненного поля	Для регистрация нового клиента с указанием ФИО требуется заполнить все поля, после чего нажать кнопку «Зарегистрировать»	1. Запустить программу; 2. Осуществить вход в систему; 3. В главном меню нажать кнопку «Регистрация клиента»;	На экране появилось уведомление: «Заполните все поля для регистрации клиента!»

		4. Оставить хотя бы одно поле пустым.	
9. Попытка поиска клиента в системе с указанием номера телефона в неверном формате	Для поиска клиента в системе требуется ввести номер телефона в формате: 9XXXXXXXXX, после чего нажать кнопку «Найти»	1. Запустить программу; 2. Осуществить вход в систему; 3. В главном меню нажать кнопку «Поиск клиента»; 4. В поле поиска по телефону указать номер телефона в неверном формате	На экране появилось уведомление: «Укажите номер телефона в формате 9XXXXXXXXX»
10. Некорректное заполнение данных сотрудника (общие сведения, паспортные данные, сведения об учетной записи, сведения о трудоустройстве) в момент регистрации	Для регистрации сотрудника требуется корректно заполнить все поля, после чего нажать кнопку «Зарегистрировать»	1. Запустить программу; 2. Осуществить вход в систему; 3. В главном меню нажать кнопку «Добавить сотрудника»; 4. Допустить ошибку в заполнении какого-либо поля	На экране появится уведомление: «Проверьте корректность заполнения всех полей и повторите регистрацию»
11. Попытка зарегистрировать сотрудника, будучи авторизованным с правами обычного пользователя	Чтобы получить доступ к разделу «Добавить сотрудника» требуется вход в систему от имени администратора	1. Запустить программу; 2. Осуществить вход в систему с учетной записи обычного пользователя; 3. В главном меню нажать кнопку «Добавить сотрудника».	На экране появилось уведомление: «Для доступа к этому разделу войдите с учетной записью администратора»
12. Попытка найти сотрудника с правами обычного пользователя	Чтобы получить доступ к разделу «Найти сотрудника»,	1. Запустить программу; 2. Осуществить вход в систему с	На экране появилось уведомление: «Для доступа к

	требуется вход в систему от имени администратора	учетной записи обычного пользователя; 3. В главном меню нажать кнопку «Найти сотрудника».	этому разделу войдите с учетной записью администратора»
13. Попытка оформления пустого заказа	Чтобы оформить заказ, необходимо добавить в него хотя бы одну позицию	1. Запустить программу; 2. Выполнить вход в систему; 3. В главном меню нажать кнопку «Принять заказ». 4. Оставив список позиций пустым, нажать кнопку «Принять заказ»	На экране появилось уведомление: «Добавьте в заказ хотя бы одну позицию»
14. Попытка в момент принятия заказа списать у клиента больше бонусов, чем у него есть на бонусном балансе	В момент оформления заказа требуется количество бонусов к списанию, не превышающее сумму доступных бонусных средств	1. Запустить программу; 2. Выполнить вход в систему; 3. В главном меню нажать кнопку «Принять заказ». 4. Выбрать клиента для привязки заказа к его профилю. 5. Нажать кнопку «Списать бонусы» 6. Ввести число, превышающее размер доступных к списанию бонусов	На экране появится уведомление «Недостаточно бонусов для списания! Введите число, не превышающее сумму доступных средств.»

2 Охрана труда

Современное законодательство в области охраны труда при работе с электронно-вычислительными машинами (ЭВМ) выдвигает следующие требования [12]:

Помещение, в котором осуществляется работа с ПК, должно соответствовать следующим требованиям:

- освещение должно соответствовать стандартам, а в случае отсутствия естественного освещения должны быть предоставлены расчеты, доказывающие безопасность работы без него;

- оконные проемы должны иметь регулируемые устройства, такие как жалюзи или занавеси;

- компьютеры не должны размещаться в местах, где есть риск попадания влаги.

- рабочие места с компьютерами должны быть оборудованы защитным заземлением;

- в помещениях с компьютерами следует проводить ежедневную влажную уборку и проветривание каждый час;

- рекомендуемая температура воздуха в помещении с компьютерами составляет $+19^{\circ}\text{C}$ - $+21^{\circ}\text{C}$;

- рабочий стол должен быть спроектирован таким образом, чтобы обеспечить оптимальное расположение используемого оборудования в соответствии с выполняемой работой, разрешается использование различных типов рабочих столов, соответствующих современным стандартам эргономики;

- в помещениях с компьютерами следует проводить ежедневную влажную уборку и проветривание каждый час.

Рабочие места с персональными компьютерами должны быть организованы следующим образом:

- при организации рабочих мест с компьютерами необходимо учитывать, что расстояние между рабочими столами, на которых установлены

видеомониторы, должно быть не менее 2,0 метров в направлении между ними и не менее 1,2 метра между боковыми поверхностями видеомониторов;

– размещение компьютеров должно предусматривать расстояние не менее 1 метра от источников тепла. Следует избегать прямого попадания солнечного света на них;

– запрещается размещать рабочие места с компьютерами рядом с силовыми кабелями, высоковольтными трансформаторами и другим оборудованием, способным создавать помехи в работе компьютеров;

– при установке компьютеров следует избегать прохождения интерфейсных и электропитания кабелей по полу. В случае необходимости, их следует закрыть защитными коробками;

– экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов;

– системный блок и монитор следует разместить так, чтобы расстояние от вентиляционных отверстий этих устройств до стены (или стенки мебели) составляло не менее 10 сантиметров;

– искусственное освещение в помещениях для эксплуатации компьютеров должно осуществляться системой общего равномерного освещения. Освещенность на поверхности рабочего стола в зоне размещения ПК должна быть 300-500 люкс. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 люкс.

В случае работы на компьютере в течение 8-часовой смены следует предусматривать перерывы в соответствии с категорией работ:

- для работников I категории перерывы устанавливаются через 2 часа после начала смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;

- работники II категории должны иметь перерыв через 2 часа после начала смены и через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы;

- для работников III категории рекомендуется устанавливать перерыв через 1,5 - 2,0 часа после начала смены и также через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

При 12-часовой смене регламентированные перерывы должны быть предусмотрены в первые 8 часов работы так же, как и при 8-часовой смене. В течение последних 4 часов работы, независимо от категории и вида работ, необходимо предусматривать перерыв каждый час продолжительностью 15 минут (таблица 18).

Таблица 18 – Время регламентированных перерывов в зависимости от категории работы

Категория работы	Уровень нагрузки за рабочую смену при видах работ с компьютером			Суммарное время регламентированных перерывов, мин.	
	группа А, количество знаков	группа Б, количество знаков	группа В, час.	при 8-часовой смене	при 12-часовой смене
I	До 20000	До 15000	До 2,0	30	70
II	До 40000	До 30000	До 4,0	50	90
III	До 60000	До 40000	До 6,0	70	120

Заключение

В ходе выполнения данной выпускной квалификационной работы была спроектирована и разработана система автоматизации работы заведений общественного питания.

В результате был проведен глубокий анализ работы сотрудников заведений общественного питания малого масштаба, а также их взаимодействие с посетителями. Были определены основные недостатки и пути их устранения у действовавших на тот момент способах хранения и обработки информации. Перед началом разработки продукта был проведен анализ предметной области, проанализированы популярные аналоги уже разработанных систем по автоматизации работы кафе и ресторанов, выбраны необходимые средства и инструменты для разработки собственного приложения.

В итоге было принято решение разработать десктопное приложение на языке программирования Java, графический интерфейс реализован с помощью библиотеки SwingGUI. Для хранения и обработки пользовательских данных была применена СУБД PostgreSQL в связке с графическим клиентом pgAdmin4, упрощающим процесс написания SQL-запросов и оценки корректности их работы. В качестве среды разработки выбрана ApacheNetBeans.

На основе приведенных выше технологий, удалось спроектировать, разработать и протестировать приложение, со следующим функционалом:

- управление профилями посетителей заведения, включая регистрацию, изменение данных, удаление профилей и просмотр информации в них;
- обработка заказов, включая создание, изменение и отмену заказов, выбор профиля клиента для списания и начисления бонусов, а также сохранение информации в истории заказов;
- настройка меню заведения, добавление, изменение и удаление позиций меню;
- работа с заявками на приобретение продукции, включая просмотр, создание, закрытие, отклонение и принятие заявок;

- управление учетными записями сотрудников, включая просмотр, внесение данных, изменение учетных данных, создание и аннулирование учетных записей;
- взаимодействие с поставщиками продуктов, добавление, изменение и удаление информации о поставщиках, прием товаров.

Разработанное в данной выпускной квалификационной работе приложение на текущий момент содержит весь необходимый базовый функционал для начала комфортной работы. Однако, как и большинство других программных продуктов, нуждается в доработке в плане реализации новых функций и улучшения работы действующих. Также планируется реализация полноценного взаимодействия программы с внешними устройствами. Например, имеет место быть возможность печати списка приобретенных позиций напрямую в чек, который выдается посетителю, с отправкой всех необходимых в Федеральную налоговую службу РФ.

Созданная автоматизированная система, призванная упростить и улучшить выполнение повседневных задач работников кафе и ресторанов, полностью готова к эксплуатации.

Список использованных источников

1. Сиддхартха Рао. Освой самостоятельно C++ за 21 день. 7-е издание. – М.: Издательство «Вильямс», 2013. – 685 с.
2. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
3. Хорстмани, Кей С., Корнелл, Гари. Java – Библиотека профессионала, том 1. Основы. 9-е изд.: Пер. с англ. – М: ООО «И.Д. Вильямс», 2014. – 864 с.
4. Лафорс Р. Структуры данных и алгоритмы в Java. 2-е изд. – СПб: Питер, 2013. – 704 с.
5. Рогов Е. В. PostgreSQL 16 изнутри. – М.: ДМК Пресс, 2024. – 664 с.
6. Официальная документация Apache NetBeans 21. [Электронный ресурс] / Режим доступа: <https://netbeans.apache.org/front/main/about/>, свободный (дата обращения 17.04.2024).
7. Официальная документация по среде IntelliJ IDEA. [Электронный ресурс] / Режим доступа: <https://www.jetbrains.com/help/idea/javadocs.html/>, свободный (дата обращения 10.05.2024).
8. Иван Портянкин. Swing: Эффективные пользовательские интерфейсы. – М.: Издательство «Лори», 2011. – 591 с.
9. Официальная документация JavaFX. [Электронный ресурс] / Режим доступа: <https://www.oracle.com/java/technologies/javase/javafx-docs.html/>, свободный (дата обращения 15.05.2024).
10. Официальная документация Introducing JSON. [Электронный ресурс] / Режим доступа: <https://www.json.org/json-en.html/>, свободный (дата обращения 07.05.2024).
11. Описание шаблона проектирования MVC. [Электронный ресурс] / Режим доступа: https://pub.dev/documentation/mvc_pattern/latest/, свободный (дата обращения 11.05.2024).
12. Инструкция по охране труда при работе с ЭВМ. [Электронный ресурс] / Режим доступа: https://kpfu.ru/portal/docs/F_1086113514/1815.pdf/, свободный (дата обращения 12.05.24).

Приложение А

(Обязательное)

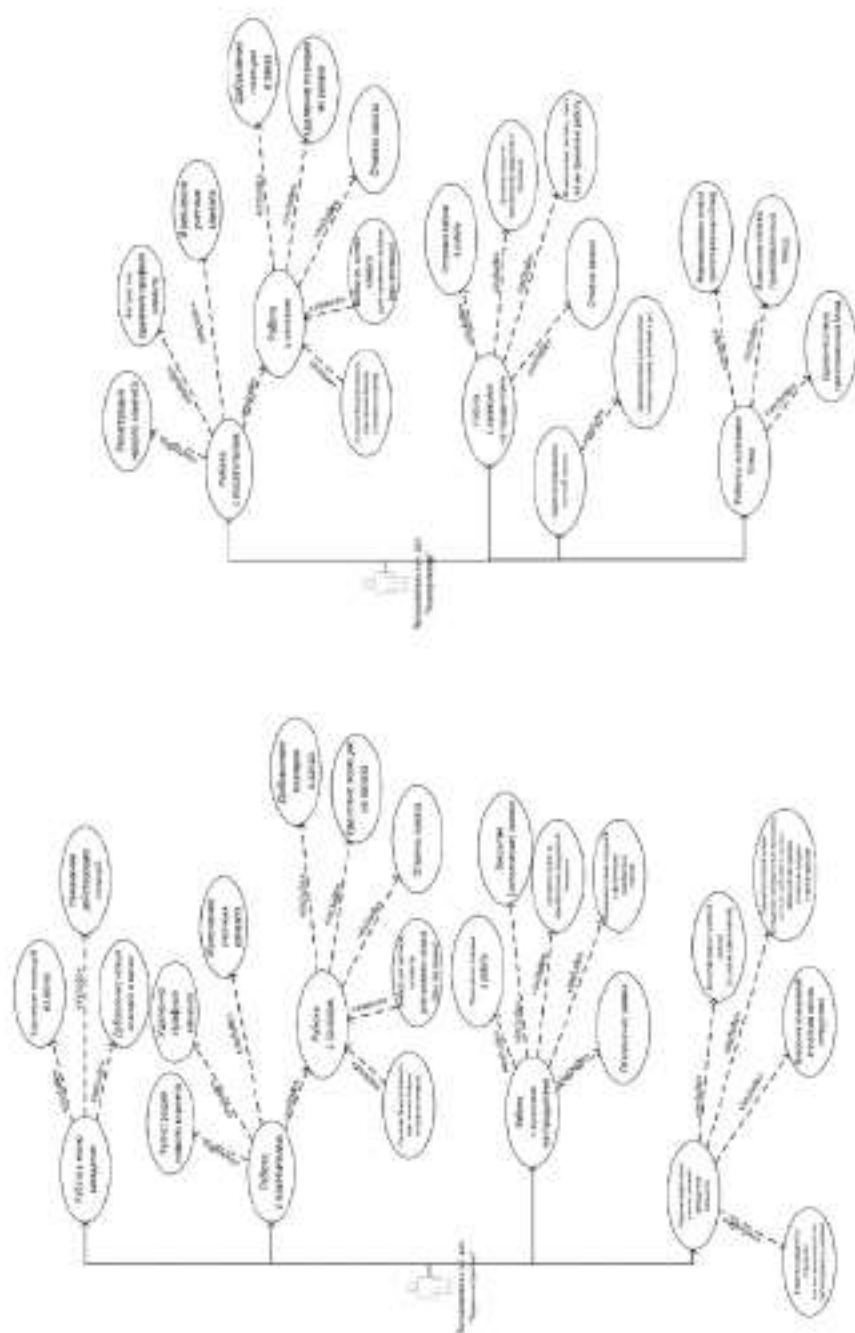
Система автоматизации работы заведений общественного питания

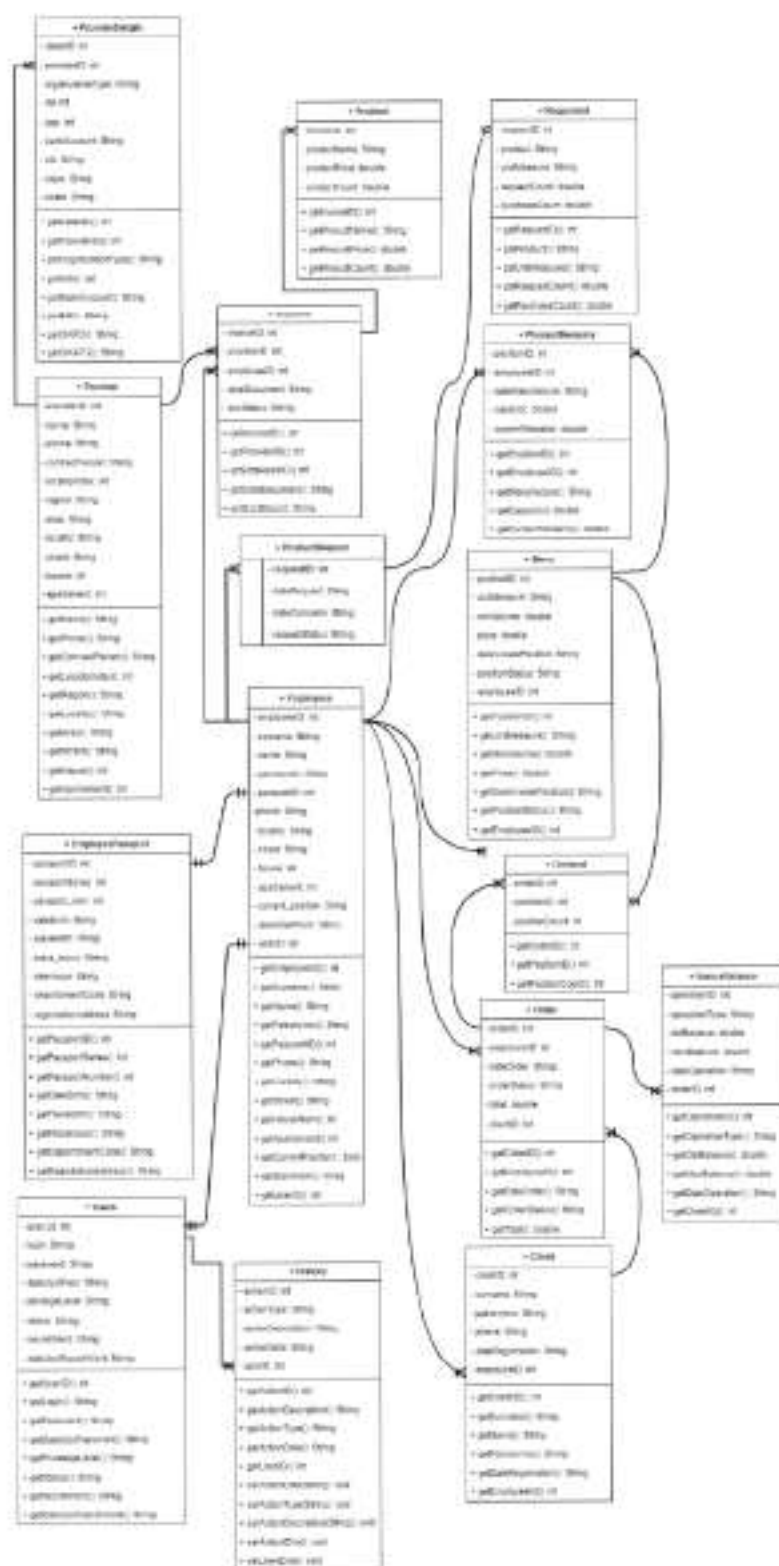
UML-диаграммы

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		100



ПУ 1.090301.17.005.Д14			
Изм.	Лист	№ докум.	Подпись
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10

[illegible]



ПУУ 1.090301.17.001.Д11			
Изм.	Лист	№ докум.	Подпись
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10
11	11	11	11
12	12	12	12
13	13	13	13
14	14	14	14
15	15	15	15
16	16	16	16
17	17	17	17
18	18	18	18
19	19	19	19
20	20	20	20
21	21	21	21
22	22	22	22
23	23	23	23
24	24	24	24
25	25	25	25
26	26	26	26
27	27	27	27
28	28	28	28
29	29	29	29
30	30	30	30
31	31	31	31
32	32	32	32
33	33	33	33
34	34	34	34
35	35	35	35
36	36	36	36
37	37	37	37
38	38	38	38
39	39	39	39
40	40	40	40
41	41	41	41
42	42	42	42
43	43	43	43
44	44	44	44
45	45	45	45
46	46	46	46
47	47	47	47
48	48	48	48
49	49	49	49
50	50	50	50
51	51	51	51
52	52	52	52
53	53	53	53
54	54	54	54
55	55	55	55
56	56	56	56
57	57	57	57
58	58	58	58
59	59	59	59
60	60	60	60
61	61	61	61
62	62	62	62
63	63	63	63
64	64	64	64
65	65	65	65
66	66	66	66
67	67	67	67
68	68	68	68
69	69	69	69
70	70	70	70
71	71	71	71
72	72	72	72
73	73	73	73
74	74	74	74
75	75	75	75
76	76	76	76
77	77	77	77
78	78	78	78
79	79	79	79
80	80	80	80
81	81	81	81
82	82	82	82
83	83	83	83
84	84	84	84
85	85	85	85
86	86	86	86
87	87	87	87
88	88	88	88
89	89	89	89
90	90	90	90
91	91	91	91
92	92	92	92
93	93	93	93
94	94	94	94
95	95	95	95
96	96	96	96
97	97	97	97
98	98	98	98
99	99	99	99
100	100	100	100

Приложение Б

(Обязательное)

Система автоматизации работы заведений общественного питания

Листинги программы

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		106

Файл MainClass.java:

```
package MyProgramm;
import java.awt.Color;
import java.awt.Component;
import java.io.IOException;
import static java.lang.Thread.sleep;
import java.net.Socket;
import javax.swing.SwingUtilities;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import javax.swing.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Date;
import java.util.Iterator;
import javax.swing.table.DefaultTableModel;

public class NewJFrame extends javax.swing.JFrame {

    //Переменные, хранящие пользовательские данные
    Map<String, Component> tabS = new HashMap<String, Component>();
    CurrentUser currentUser;
    EmployeeData myProfile;
    ArrayList<History> history = new ArrayList<>();
    History countItems = new History();
    ArrayList<SearchEmployees> searchEmployees = new ArrayList<>();

    public void saveTabs() {
        for (int i=0; i < jTabledPane2.getTabCount(); i++) {
            tabS.put(jTabledPane2.getTitleAt(i),
jTabledPane2.getComponentAt(i));
        }
    }

    public void checkConnection() {
        try {
            Socket checkConn = new Socket("localhost", 8000);
            DataOutputStream out = new
DataOutputStream(checkConn.getOutputStream());
            out.writeInt(500);
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this, "Отсутствует
подключение к серверу!\nПроверьте подключение и перезапустите
программу.",

```

```

        "Ошибка соединения",
        JOptionPane.ERROR_MESSAGE);
        System.exit(0);
    }
}

public String auth(String login, String pass) {
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;
    String response;
    ArrayList<String> requestResult = new ArrayList<>();

    if(!(login.equals("") || pass.equals(""))){ //Если все поля
заполнены

        try {
            clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
            outputStream = new
DataOutputStream(clientSocket.getOutputStream());
            inputStream = new
DataInputStream(clientSocket.getInputStream());

            outputStream.writeInt(10); //Отправка кода запроса
            outputStream.writeUTF(login); //Отправка данных для
входа в систему
            outputStream.writeUTF(pass);

            response = inputStream.readUTF(); //Чтение кода ответа
сервера

            switch (response) { //Анализ ответа сервера
                case "authOK" -> {
                    int count = inputStream.readInt(); //Количество
строк по результатам запроса к БД

                    for (int i = 0; i < count; i++) {
                        requestResult.add(inputStream.readUTF());
                    }

                    String[] data =
requestResult.get(0).split("/"); //Данные учетной записи
                    String[] dataUpdPass = data[3].split("\\.");
                    String[] dataUpdWord = data[7].split("\\.");

                    //Сохранение информации о текущем
пользователе
                    currentUser = new
CurrentUser(Integer.parseInt(data[0]),data[1], data[2],
dataUpdPass[0],
data[4],
data[5], data[6], dataUpdWord[0]);

```

```

        //Аудит входа в систему
        outputStream.writeInt(currentUser.getUserId());

        //Получение информации из профиля сотрудника
        String[] employeeData =
inStream.readUTF().split("/");

        //Получение паспортных данных сотрудника
        String[] passportData =
inStream.readUTF().split("/");

        myProfile = new
EmployeeData(Integer.parseInt(employeeData[0]), employeeData[1],
employeeData[2], employeeData[3],

Integer.parseInt(employeeData[4]), employeeData[5], employeeData[6],
employeeData[7],

Integer.parseInt(employeeData[8]),
Integer.parseInt(employeeData[9]), employeeData[10],

employeeData[11], Integer.parseInt(employeeData[12]), passportData);

        return "authSuccess";
    }
    case "userNotExistError" -> {
        return "userNotExistError";
    }
    case "uncorrectLoginOrPasswordError" -> {
        return "uncorrectLoginOrPasswordError";
    }
    case "userBlockedError" -> {
        return "userBlockedError";
    }
    default -> {
    }
}

} catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);

}
finally {
    try {
        if (outStream != null) {
            outputStream.close();
            clientSocket.close();
        }
        if (inStream != null) {
            inStream.close();
            clientSocket.close();
        }
    }
}

```

```

        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
else {
    return "emptyFieldError";
}

return "unknownError";
}

public String getHistoryByLogin(String login) {
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;
    String response;
    int countRows;

    try {
        clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(15); //Отправка кода запроса
        outputStream.writeUTF(login); //Отправка логина на сервер
        outputStream.writeInt(countItems.getCountActionsDisplay());
//Отправка числа отображаемых записей, по умолчанию - 30

        response = inputStream.readUTF(); // Чтение ответа на запрос

        if (response.equals("not_found")) { //Если запрос пустой
            return "not_found";
        }

        countRows = inputStream.readInt(); //Чтение количества
записей

        history.clear();
        DefaultTableModel model2 = (DefaultTableModel)
jTable2.getModel();
        model2.setRowCount(0);

        for (int i=0; i < countRows; i++) {
            String result = inputStream.readUTF();
            String[] data = result.split("/"); //Считываем
строчку с результатами

```

```

        history.add(new History(Integer.parseInt(data[0]),
data[1], data[2], data[3], data[4]));
    }

    if(! (jCheckBox1.isSelected() || jCheckBox2.isSelected()
|| jCheckBox3.isSelected()
|| jCheckBox4.isSelected() ||
jCheckBox5.isSelected() || jCheckBox6.isSelected())) { //Если все
галочки отжаты

        for (int i=0; i < history.size(); i++) {
            DefaultTableModel model = (DefaultTableModel)
jTable2.getModel();
            model.addRow(new Object[]
(history.get(i).getActionId(), history.get(i).getActionType(),
history.get(i).getActionDescription(),
history.get(i).getActionDate(),
history.get(i).getLogin()));
        }
    }
    else {
        if (jCheckBox1.isSelected()) {
            for(int i=0; i < history.size(); i++) {
if(history.get(i).getActionType().equals("Вход в систему")) {
countItems.addNewItemIntoSortedHistory(i);
            }
        }

        if (jCheckBox2.isSelected()) {
            for(int i=0; i < history.size(); i++) {
if(history.get(i).getActionType().equals("Выход из системы")) {
countItems.addNewItemIntoSortedHistory(i);
            }
        }

        if (jCheckBox3.isSelected()) {
            for(int i=0; i < history.size(); i++) {
if(history.get(i).getActionType().equals("Просмотр сведений")) {
countItems.addNewItemIntoSortedHistory(i);
            }
        }
    }
}

```

```

        if (jCheckBox4.isSelected()) {
            for(int i=0; i < history.size(); i++) {
                if(history.get(i).getActionType().equals("Изменение сведений")) {
                    countItems.addNewItemIntoSortedHistory(i);
                }
            }
        }

        if (jCheckBox5.isSelected()) {
            for(int i=0; i < history.size(); i++) {
                if(history.get(i).getActionType().equals("Аннулирование")) {
                    countItems.addNewItemIntoSortedHistory(i);
                }
            }
        }

        if (jCheckBox6.isSelected()) {
            for(int i=0; i < history.size(); i++) {
                if(history.get(i).getActionType().equals("Внесение сведений")) {
                    countItems.addNewItemIntoSortedHistory(i);
                }
            }
        }

        ArrayList<Integer> sortedHistory =
countItems.getSortedHistory(); //Получаем ранее отсортированную
историю
        DefaultTableModel model = (DefaultTableModel)
jTable2.getModel();
        for(Integer item : sortedHistory) {
            model.addRow(new Object[]
{history.get(item).getActionId(), history.get(item).getActionType(),
history.get(item).getActionDescription(),
history.get(item).getActionDate(),
history.get(item).getLogin()});
        }

        if(history.size() < countItems.getCountActionsDisplay())
        {
            jButton16.setEnabled(false);
        }

    } catch (IOException ex) {

```

```

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    return "ok";
}

public void getAllHistory() {
    Socket clientSocket = null;
    DataOutputStream outStream = null;
    DataInputStream inStream = null;
    String response;
    int countRows;

    try {
        clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
        outStream = new
DataOutputStream(clientSocket.getOutputStream());
        inStream = new
DataInputStream(clientSocket.getInputStream());

        outStream.writeInt(14); //Отправка кода запроса
        outStream.writeInt(countItems.getCountActionsDisplay());
//Отправка числа отображаемых записей, по умолчанию - 30

        countRows = inStream.readInt(); //Чтение количества
записей

        history.clear();

        for (int i=0; i < countRows; i++) {
            String result = inStream.readUTF();
            String[] data = result.split("/"); //Считываем
            строку с результатами

```

```

        history.add(new History(Integer.parseInt(data[0]),
data[1], data[2], data[3], data[4]));
    }

    if(! (jCheckBox1.isSelected() || jCheckBox2.isSelected()
|| jCheckBox3.isSelected()
|| jCheckBox4.isSelected() ||
jCheckBox5.isSelected() || jCheckBox6.isSelected())) { //Если все
галочки отжаты

        for (int i=0; i < history.size(); i++) {
            DefaultTableModel model = (DefaultTableModel)
jTable2.getModel();
            model.addRow(new Object[]
(history.get(i).getActionId(), history.get(i).getActionType(),
history.get(i).getActionDescription(),
history.get(i).getActionDate(),
history.get(i).getLogin()));
        }
    }
    else {
        if (jCheckBox1.isSelected()) {
            for(int i=0; i < history.size(); i++) {
if(history.get(i).getActionType().equals("Вход в систему")) {
countItems.addNewItemIntoSortedHistory(i);
            }
        }

        if (jCheckBox2.isSelected()) {
            for(int i=0; i < history.size(); i++) {
if(history.get(i).getActionType().equals("Выход из системы")) {
countItems.addNewItemIntoSortedHistory(i);
            }
        }

        if (jCheckBox3.isSelected()) {
            for(int i=0; i < history.size(); i++) {
if(history.get(i).getActionType().equals("Просмотр сведений")) {
countItems.addNewItemIntoSortedHistory(i);
            }
        }
    }
}

```

```

        if (jCheckBox4.isSelected()) {
            for(int i=0; i < history.size(); i++) {
                if(history.get(i).getActionType().equals("Изменение сведений")) {
                    countItems.addNewItemIntoSortedHistory(i);
                }
            }
        }

        if (jCheckBox5.isSelected()) {
            for(int i=0; i < history.size(); i++) {
                if(history.get(i).getActionType().equals("Аннулирование")) {
                    countItems.addNewItemIntoSortedHistory(i);
                }
            }
        }

        if (jCheckBox6.isSelected()) {
            for(int i=0; i < history.size(); i++) {
                if(history.get(i).getActionType().equals("Внесение сведений")) {
                    countItems.addNewItemIntoSortedHistory(i);
                }
            }
        }

        ArrayList<Integer> sortedHistory =
countItems.getSortedHistory(); //Получаем ранее отсортированную
историю
        DefaultTableModel model = (DefaultTableModel)
jTable2.getModel();
        for(Integer item : sortedHistory) {
            model.addRow(new Object[]
{history.get(item).getActionId(), history.get(item).getActionType(),
history.get(item).getActionDescription(),
history.get(item).getActionDate(),
history.get(item).getLogin()});
        }

        if(history.size() < countItems.getCountActionsDisplay())
        {
            jButton16.setEnabled(false);
        }

    } catch (IOException ex) {

```

```

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

}

public void logoutSystemAudit() {
    Socket clientSocket = null;
    DataOutputStream outStream = null;
    DataInputStream inStream = null;

    try {
        clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
        outStream = new
DataOutputStream(clientSocket.getOutputStream());
        inStream = new
DataInputStream(clientSocket.getInputStream());

        outStream.writeInt(1); //Отправка кода запроса

        outStream.writeInt(currentUser.getUserId()); //Отправка
номера учетной записи

    } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
            }
        }
    }
}

```

```

        clientSocket.close();
    }
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

public void viewProfileEmployeeAudit() {
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;

    try {
        clientSocket = new Socket("localhost", 8000);
        //Подключение к серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(2); //Отправка кода запроса

        outputStream.writeInt(currentUser.getUserId()); //Отправка
номера учетной записи

    } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outputStream != null) {
                outputStream.close();
                clientSocket.close();
            }
            if (inputStream != null) {
                inputStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void createNewEmployee() {
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;

```

```

String response; //Код ответа сервера
String passport; //Паспортные данные сотрудника
String employeeData; //Общие сведения о сотруднике
String userData; //Данные учетной записи

try {
    clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
    outputStream = new
DataOutputStream(clientSocket.getOutputStream());
    inputStream = new
DataInputStream(clientSocket.getInputStream());

    outputStream.writeInt(16); //Отправка кода запроса
    outputStream.writeInt(currentUser.getUserId()); //Отправка
номера учетной записи

    passport = jTextField58.getText().trim() + "/" +
        jTextField59.getText().trim() + "/" +
        jTextField62.getText().trim() + "-" +
jTextField61.getText().trim() + "-" + jTextField60.getText().trim()
+ "/" +
        jTextField66.getText().trim() + "/" +
        jTextField67.getText().trim() + "/" +
        jTextField65.getText().trim() + "-" +
jTextField64.getText().trim() + "-" + jTextField63.getText().trim()
+ "/" +
        jTextField68.getText().trim() + "-" +
jTextField69.getText().trim() + "/" +
        jTextField16.getText().trim();

    employeeData = jTextField50.getText().trim() + "/" +
        jTextField51.getText().trim() + "/" +
        jTextField52.getText().trim() + "/" +
        jTextField53.getText().trim() + "/" +
        jTextField54.getText().trim() + "/" +
        jTextField55.getText().trim() + "/" +
        jTextField56.getText().trim() + "/" +
        jTextField57.getText().trim() + "/" +
        jTextField73.getText().trim() + "/" +
        jTextField43.getText().trim() + "-" +
jTextField42.getText().trim() + "-" + jTextField41.getText().trim();

    userData = jTextField70.getText().trim() + "/" +
        jTextField71.getText().trim() + "/" +
        jTextField72.getText().trim() + "/" +
        jComboBox1.getSelectedItem() + "/" +
        jComboBox2.getSelectedItem();

    outputStream.writeUTF(passport); //Отправка данных
    outputStream.writeUTF(employeeData);
    outputStream.writeUTF(userData);
}

```

```

        response = inStream.readUTF(); //Чтение ответа на запрос
        outputStream.writeInt(currentUser.getUserId()); //Аудит
        внесения сведений о сотруднике

        switch(response) {
            case "loginNotAvailable" -> {
                JOptionPane.showMessageDialog(this, "Логин @" +
jTextField70.getText().trim() + " уже занят другим сотрудником!\n"
                + "Пожалуйста, введите другой свободный
логин.",
                "Ошибка регистрации", JOptionPane.ERROR_MESSAGE);
            }
            case "passportNotAvailable" -> {
                String login = inStream.readUTF();
                JOptionPane.showMessageDialog(this, "Паспорт с
серией " + jTextField58.getText().trim() + " и №"
                + jTextField59.getText().trim() + "
принадлежит сотруднику с логином @" + login + "!\n"
                + "Пожалуйста, проверьте корректность
введенных данных.",
                "Ошибка регистрации", JOptionPane.ERROR_MESSAGE);
            }
            case "phoneNotAvailable" -> {
                String login = inStream.readUTF();
                JOptionPane.showMessageDialog(this, "Номер
телефона +7" + jTextField53.getText().trim() + " уже используется "
+
                "и принадлежит сотруднику с логином @" +
login + "!\n"
                + "Пожалуйста, введите другой номер
телефона.",
                "Ошибка регистрации", JOptionPane.ERROR_MESSAGE);
            }
            case "success" -> {
                JOptionPane.showMessageDialog(this, "Сотрудник
успешно зарегистрирован!",
                "Регистрация завершена",
JOptionPane.INFORMATION_MESSAGE);

                jTextField50.setText("");
                jTextField51.setText("");
                jTextField52.setText("");
                jTextField53.setText("");
                jTextField54.setText("");
                jTextField55.setText("");
                jTextField56.setText("");
                jTextField57.setText("");
                jTextField58.setText("");
                jTextField59.setText("");
                jTextField60.setText("");
                jTextField61.setText("");
                jTextField62.setText("");
                jTextField66.setText("");

```

```

        jTextField67.setText("");
        jTextField63.setText("");
        jTextField64.setText("");
        jTextField65.setText("");
        jTextField68.setText("");
        jTextField69.setText("");
        jTextField16.setText("");
        jTextField70.setText("");
        jTextField71.setText("");
        jTextField72.setText("");
        jTextField73.setText("");
        jTextField41.setText("");
        jTextField42.setText("");
        jTextField43.setText("");

        jCheckBox7.setSelected(false);
        jComboBox1.setSelectedItem("Активен");
        jComboBox2.setSelectedItem("Пользователь");
    }
}

} catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
}
finally {
    try {
        if (outStream != null) {
            outStream.close();
            clientSocket.close();
        }
        if (inStream != null) {
            inStream.close();
            clientSocket.close();
        }
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

}

public String validateEmployeeData() {
    boolean fieldsNotEmpty =
        jTextField50.getText().trim().isEmpty() ||
        jTextField51.getText().trim().isEmpty() ||
        jTextField52.getText().trim().isEmpty() ||
        jTextField53.getText().trim().isEmpty() ||
        jTextField54.getText().trim().isEmpty() ||
        jTextField55.getText().trim().isEmpty() ||
        jTextField56.getText().trim().isEmpty() ||
        jTextField57.getText().trim().isEmpty() ||

```

```

jTextField58.getText().trim().isEmpty() ||
jTextField59.getText().trim().isEmpty() ||
jTextField60.getText().trim().isEmpty() ||
jTextField61.getText().trim().isEmpty() ||
jTextField62.getText().trim().isEmpty() ||
jTextField66.getText().trim().isEmpty() ||
jTextField67.getText().trim().isEmpty() ||
jTextField63.getText().trim().isEmpty() ||
jTextField64.getText().trim().isEmpty() ||
jTextField65.getText().trim().isEmpty() ||
jTextField68.getText().trim().isEmpty() ||
jTextField69.getText().trim().isEmpty() ||
jTextField16.getText().trim().isEmpty() ||
jTextField70.getText().trim().isEmpty() ||
jTextField71.getText().trim().isEmpty() ||
jTextField72.getText().trim().isEmpty() ||
jTextField73.getText().trim().isEmpty() ||
jTextField41.getText().trim().isEmpty() ||
jTextField42.getText().trim().isEmpty() ||
jTextField43.getText().trim().isEmpty();

if(fieldsNotEmpty) { //Если не заполнена хотя бы одно поле
    return "emptyFieldsError";
}

if(jTextField53.getText().trim().length() < 10 ||
jTextField53.getText().trim().length() > 10) { //Проверка
корректности длины номера телефона
    return "phoneLengthError";
}

try {

Integer.parseInt(jTextField53.getText().trim().substring(0, 5)); //В
номере присутствуют символы

Integer.parseInt(jTextField53.getText().trim().substring(5, 10));
} catch (NumberFormatException ex) {
    return "incorrectPhoneError";
}

if(!jTextField53.getText().trim().substring(0,
1).equals("9")) { // Номер начинается не с 9-ки
    return "incorrectStartCharacterError";
}

try {
    Integer.parseInt(jTextField56.getText().trim());
//Проверка номера дома
} catch (NumberFormatException ex) {
    return "incorrectHouseError";
}

```

```

    }

    try {
        Integer.parseInt(jTextField57.getText().trim());
//Проверка номера квартиры
    } catch (NumberFormatException ex) {
        return "incorrectApartmentError";
    }

    if(jTextField58.getText().trim().length() < 4 ||
jTextField58.getText().trim().length() > 4) {
        return "passportSeriesLengthError";
    }

    try {
        Integer.parseInt(jTextField58.getText().trim());
//Проверка серии паспорта
    } catch (NumberFormatException ex) {
        return "passportSeriesError";
    }

    if(jTextField59.getText().trim().length() < 6 ||
jTextField59.getText().trim().length() > 6) {
        return "passportNumberLengthError";
    }

    try {
        Integer.parseInt(jTextField59.getText().trim());
//Проверка номера паспорта
    } catch (NumberFormatException ex) {
        return "passportNumberError";
    }

    //Проверка правильности длины введенной даты рождения
    if(jTextField60.getText().trim().length() < 2 ||
jTextField60.getText().trim().length() > 2
        || jTextField61.getText().trim().length() < 2 ||
jTextField61.getText().trim().length() > 2
        || jTextField62.getText().trim().length() < 4 ||
jTextField62.getText().trim().length() > 4) {

        return "birthdayError";
    }

    try {
        Integer.parseInt(jTextField60.getText().trim());
//Проверка наличия символов в дате рождения
        Integer.parseInt(jTextField61.getText().trim());
        Integer.parseInt(jTextField62.getText().trim());
    } catch (NumberFormatException ex) {
        return "birthdayError";
    }

```

```

int date1 = Integer.parseInt(jTextField60.getText().trim());
int date2 = Integer.parseInt(jTextField61.getText().trim());
int date3 = Integer.parseInt(jTextField62.getText().trim());

// Получаем текущую дату
LocalDate currentDate = LocalDate.now();
// Получаем текущий год из текущей даты
int currentYear = currentDate.getYear();

if(date1 < 1 || date1 > 31 || date2 < 1 || date2 > 12 ||
date3 < 1900 || date3 > currentYear) {
    return "birthdayError";
}

//Проверка правильности длины введенной ДАТЫ ВЫДАЧИ ПАСПОРТА
if(jTextField63.getText().trim().length() < 2 ||
jTextField63.getText().trim().length() > 2
|| jTextField64.getText().trim().length() < 2 ||
jTextField64.getText().trim().length() > 2
|| jTextField65.getText().trim().length() < 4 ||
jTextField65.getText().trim().length() > 4) {

    return "dateIssueError";
}

try {
    Integer.parseInt(jTextField63.getText().trim());
//Проверка наличия символов в дате выдачи паспорта
    Integer.parseInt(jTextField64.getText().trim());
    Integer.parseInt(jTextField65.getText().trim());
} catch (NumberFormatException ex) {
    return "dateIssueError";
}

date1 = Integer.parseInt(jTextField63.getText().trim());
date2 = Integer.parseInt(jTextField64.getText().trim());
date3 = Integer.parseInt(jTextField65.getText().trim());

// Получаем текущую дату
currentDate = LocalDate.now();
// Получаем текущий год из текущей даты
currentYear = currentDate.getYear();

if(date1 < 1 || date1 > 31 || date2 < 1 || date2 > 12 ||
date3 < 1900 || date3 > currentYear) {
    return "dateIssueError";
}

//Проверка корректности введенного кода подразделения
if(jTextField68.getText().trim().length() < 3 ||
jTextField68.getText().trim().length() > 3
|| jTextField69.getText().trim().length() < 3 ||
jTextField69.getText().trim().length() > 3) {

```

```

        return "departamentCodeError";
    }

    try {
        Integer.parseInt(jTextField68.getText().trim());
        //Проверка наличия символов в коде подразделения
        Integer.parseInt(jTextField69.getText().trim());
    } catch (NumberFormatException ex) {
        return "departamentCodeError";
    }

    //Проверка длины пароля
    if(jTextField71.getText().trim().length() < 5) {
        return "passwordLengthError";
    }

    //Проверка правильности длины введенной даты трудоустройства
    if(jTextField41.getText().trim().length() < 2 ||
jTextField41.getText().trim().length() > 2
        || jTextField42.getText().trim().length() < 2 ||
jTextField42.getText().trim().length() > 2
        || jTextField43.getText().trim().length() < 4 ||
jTextField43.getText().trim().length() > 4) {

        return "dateExperienceError";
    }

    try {
        Integer.parseInt(jTextField41.getText().trim());
        //Проверка наличия символов в дате рождения
        Integer.parseInt(jTextField42.getText().trim());
        Integer.parseInt(jTextField43.getText().trim());
    } catch (NumberFormatException ex) {
        return "dateExperienceError";
    }

    date1 = Integer.parseInt(jTextField41.getText().trim());
    date2 = Integer.parseInt(jTextField42.getText().trim());
    date3 = Integer.parseInt(jTextField43.getText().trim());

    // Получаем текущую дату
    currentDate = LocalDate.now();
    // Получаем текущий год из текущей даты
    currentYear = currentDate.getYear();

    if(date1 < 1 || date1 > 31 || date2 < 1 || date2 > 12 ||
date3 < 1900 || date3 > currentYear) {
        return "dateExperienceError";
    }

    return "validateSuccess";
}

```

```

public String updateEmployeeData() {
    //Валидация данных
    boolean fieldsNotEmpty =
        jTextField5.getText().trim().isEmpty() ||
        jTextField6.getText().trim().isEmpty() ||
        jTextField7.getText().trim().isEmpty() ||
        jTextField9.getText().trim().isEmpty() ||
        jTextField10.getText().trim().isEmpty() ||
        jTextField11.getText().trim().isEmpty() ||
        jTextField13.getText().trim().isEmpty() ||
        jTextField14.getText().trim().isEmpty();

    if(fieldsNotEmpty) { //Если не заполнена хотя бы одно поле
        return "emptyFieldsError";
    }

    if(jTextField9.getText().trim().length() < 10 ||
    jTextField9.getText().trim().length() > 10) { //Проверка
корректности длины номера телефона
        return "phoneLengthError";
    }

    try {

Integer.parseInt(jTextField9.getText().trim().substring(0, 5)); //В
номере присутствуют символы

Integer.parseInt(jTextField9.getText().trim().substring(5, 10));
    } catch (NumberFormatException ex) {
        return "incorrectPhoneError";
    }

    if(!jTextField9.getText().trim().substring(0,
1).equals("9")) { // Номер начинается не с 9-ки
        return "incorrectStartCharacterError";
    }

    try {
        Integer.parseInt(jTextField13.getText().trim());
//Проверка номера дома
    } catch (NumberFormatException ex) {
        return "incorrectHouseError";
    }

    try {
        Integer.parseInt(jTextField14.getText().trim());
//Проверка номера квартиры
    } catch (NumberFormatException ex) {
        return "incorrectApartmentError";
    }
}

```

```

//Проверка доступности номера телефона
Socket clientSocket = null;
DataOutputStream outputStream = null;
DataInputStream inputStream = null;
String response; //Код ответа сервера
String passport; //Паспортные данные сотрудника
String employeeData; //Общие сведения о сотруднике
String userData; //Данные учетной записи

try {
    clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
    outputStream = new
DataOutputStream(clientSocket.getOutputStream());
    inputStream = new
DataInputStream(clientSocket.getInputStream());

    outputStream.writeInt(17); //Отправка кода запроса
    outputStream.writeInt(currentUser.getUserId()); //Отправка
номера учетной записи

    employeeData = jTextField5.getText().trim() + "/" +
        jTextField6.getText().trim() + "/" +
        jTextField7.getText().trim() + "/" +
        jTextField9.getText().trim() + "/" +
        jTextField10.getText().trim() + "/" +
        jTextField11.getText().trim() + "/" +
        jTextField13.getText().trim() + "/" +
        jTextField14.getText().trim() + "/" +
        jTextField8.getText().trim();

    outputStream.writeUTF(employeeData); //Отправка измененных
данных

    response = inputStream.readUTF(); //Чтение ответа от
сервера

    if(response.equals("phoneUsedError")) {
        return "phoneUsedError";
    }

} catch (IOException ex) {

    Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
}
finally {
    try {
        if (outputStream != null) {
            outputStream.close();
            clientSocket.close();
        }
        if (inputStream != null) {

```

```

        inStream.close();
        clientSocket.close();
    }
}
catch (IOException e) {
    e.printStackTrace();
}
}

return "success";
}

public void updateSecretWord(String secretWord) {
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;

    try {
        clientSocket = new Socket("localhost", 8000);
        //Подключение к серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(18); //Отправка кода запроса
        outputStream.writeUTF(currentUser.getLogin()); //Отправка
номера учетной записи
        outputStream.writeUTF(secretWord); //Отправка логина
        outputStream.writeInt(currentUser.getUserId());

    } catch (IOException ex) {

        Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outputStream != null) {
                outputStream.close();
                clientSocket.close();
            }
            if (inputStream != null) {
                inputStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}

```

```

    public void updatePassword(String newPassword) {
        Socket clientSocket = null;
        DataOutputStream outputStream = null;
        DataInputStream inputStream = null;

        try {
            clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
            outputStream = new
DataOutputStream(clientSocket.getOutputStream());
            inputStream = new
DataInputStream(clientSocket.getInputStream());

            outputStream.writeInt(19); //Отправка кода запроса
            outputStream.writeUTF(currentUser.getLogin()); //Отправка
логина сотрудника, у которого меняется пароль
            outputStream.writeUTF(newPassword); //Отправка нового пароля
            outputStream.writeInt(currentUser.getUserId());

        } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
        }
        finally {
            try {
                if (outputStream != null) {
                    outputStream.close();
                    clientSocket.close();
                }
                if (inputStream != null) {
                    inputStream.close();
                    clientSocket.close();
                }
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public void getEmployeesList() {
        Socket clientSocket = null;
        DataOutputStream outputStream = null;
        DataInputStream inputStream = null;
        String response;
        int countRows;

        try {
            clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
            outputStream = new
DataOutputStream(clientSocket.getOutputStream());

```

```

        inStream = new
DataInputStream(clientSocket.getInputStream());

        outStream.writeInt(20); //Отправка кода запроса

        countRows = inStream.readInt(); //Чтение количества
записей

        searchEmployees.clear(); //Очистка списка найденных
сотрудников

        for (int i=0; i < countRows; i++) {
            String result = inStream.readUTF();
            String[] data = result.split("/"); //Считываем
            строку с результатами
            searchEmployees.add(new SearchEmployees(data[0],
data[1], data[2], "+7" + data[3], data[4]));
        }

        for (int i=0; i < searchEmployees.size(); i++) {
            DefaultTableModel model = (DefaultTableModel)
jTable3.getModel();
            model.addRow(new Object[]
{searchEmployees.get(i).getSurname(),
searchEmployees.get(i).getName(),

searchEmployees.get(i).getPatronymic(),
searchEmployees.get(i).getPhone(),

searchEmployees.get(i).getCurrentPosition()});
        }

    } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    public String restorePassword(String secretWord, String login,
String newPassword) {
        Socket clientSocket = null;
        DataOutputStream outputStream = null;
        DataInputStream inputStream = null;
        String response;

        try {
            clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
            outputStream = new
DataOutputStream(clientSocket.getOutputStream());
            inputStream = new
DataInputStream(clientSocket.getInputStream());

            outputStream.writeInt(21); //Отправка кода запроса
            outputStream.writeUTF(login); // Отправка логина
            outputStream.writeUTF(secretWord); //Отправка кодового
слова
            outputStream.writeUTF(newPassword); // Отправка нового
пароля

            response = inputStream.readUTF(); //Чтение ответа

            switch(response) {
                case "user_not_found" -> {
                    return "user_not_found";
                }
                case "incorrectSecretWord" -> {
                    return "incorrectSecretWord";
                }
            }

        } catch (IOException ex) {

            Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
        }
        finally {
            try {
                if (outputStream != null) {
                    outputStream.close();
                    clientSocket.close();
                }
                if (inputStream != null) {
                    inputStream.close();
                    clientSocket.close();
                }
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

        }
    }

    return "success";
}

public void blockUnlockEmployee() {
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;
    String response;

    try {
        clientSocket = new Socket("localhost", 8000);
        //Подключение к серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(22); //Отправка кода запроса
        outputStream.writeInt(currentUser.getUserId()); // Отправка
номера пользователя

        outputStream.writeInt(Integer.parseInt(jTextField8.getText().trim()));
        //Отправка кода сотрудника, чья учетная запись будет заблокирована

    } catch (IOException ex) {

        Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outputStream != null) {
                outputStream.close();
                clientSocket.close();
            }
            if (inputStream != null) {
                inputStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void createClient() {
    if(jCheckBox8.isSelected()) {

        if(jTextField1.getText().trim().isEmpty()) {

```

```

        JOptionPane.showMessageDialog(this, "Введите номер
телефона", "Ошибка регистрации", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(jTextField1.getText().trim().length() < 10 ||
jTextField1.getText().trim().length() > 10) {
        JOptionPane.showMessageDialog(this, "Введите номер
телефона в формате 9XXXXXXXX", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {

Integer.parseInt(jTextField1.getText().trim().substring(0, 5)); //В
номере присутствуют символы

Integer.parseInt(jTextField1.getText().trim().substring(5, 10));
    }
    catch(NumberFormatException ex) {
        JOptionPane.showMessageDialog(this, "Проверьте
корректность заполнения номера телефона", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(!jTextField1.getText().trim().substring(0,
1).equals("9")) {
        JOptionPane.showMessageDialog(this, "Номер телефона
должен начинаться с цифры 9", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    //Отправка данных на сервер
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;
    String response;
    String data = "";

    try {
        clientSocket = new Socket("localhost", 8000);
//Подключение к серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(23); //Отправка кода запроса
        outputStream.writeInt(currentUser.getUserId()); //
Отправка номера сотрудника, выполнившего регистрацию
    }

```

```

        data = jTextField1.getText().trim() +
"/нет/нет/нет"; //Отправка пользовательских данных
        outputStream.writeUTF(data); //Отправка кода
сотрудника, чья учетная запись будет заблокирована

        response = inputStream.readUTF(); //Получение ответа
сервера

        if(response.equals("phoneUsedError")) {
            JOptionPane.showMessageDialog(this, "Номер
телефона уже используется другим клиентом!", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        JOptionPane.showMessageDialog(this, "Клиент успешно
зарегистрирован!", "Регистрация завершена",
JOptionPane.INFORMATION_MESSAGE);
        jTextField1.setText("");

    } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);

    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
else {

    if(jTextField1.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Введите номер
телефона", "Ошибка регистрации", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(jTextField1.getText().trim().length() < 10 ||
jTextField1.getText().trim().length() > 10) {

```

```

        JOptionPane.showMessageDialog(this, "Введите номер
телефона в формате 9XXXXXXXX", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {

Integer.parseInt(jTextField1.getText().trim().substring(0, 5)); //В
номера присутствуют символы

Integer.parseInt(jTextField1.getText().trim().substring(5, 10));
    }
    catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this, "Проверьте
корректность заполнения номера телефона", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(!jTextField1.getText().trim().substring(0,
1).equals("9")) {
        JOptionPane.showMessageDialog(this, "Номер телефона
должен начинаться с цифры 9", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(jTextField2.getText().trim().isEmpty() ||
jTextField3.getText().trim().isEmpty() ||
jTextField2.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Заполните все
поля для регистрации клиента", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    //Отправка данных на сервер
    Socket clientSocket = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;
    String response;
    String data = "";

    try {
        clientSocket = new Socket("localhost", 8000);
        //Подключение к серверу
        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

        outputStream.writeInt(23); //Отправка кода запроса
    }

```

```

        outputStream.writeInt(myProfile.getEmployeeId()); //
Отправка номера сотрудника, выполнившего регистрацию

        data = jTextField1.getText().trim() + "/" +
//Отправка пользовательских данных
        jTextField2.getText().trim() + "/" +
        jTextField3.getText().trim() + "/" +
        jTextField46.getText().trim();

        outputStream.writeUTF(data); //Отправка кода
сотрудника, чья учетная запись будет заблокирована

        response = inputStream.readUTF(); //Получение ответа
сервера

        if(response.equals("phoneUsedError")) {
            JOptionPane.showMessageDialog(this, "Номер
телефона уже используется другим клиентом!", "Ошибка регистрации",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        JOptionPane.showMessageDialog(this, "Клиент успешно
зарегистрирован!", "Регистрация завершена",
JOptionPane.INFORMATION_MESSAGE);
        jTextField1.setText("");

    } catch (IOException ex) {

        Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null,
ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public NewJFrame() {
    initComponents();
    saveTabs(); //Сохранение вкладок
    checkConnection(); //Проверка подключения к серверу
}

```

```

        jTabbedPane2.removeAll(); //Очистка всех вкладок
        jTabbedPane2.addTab("Авторизация", new
ImageIcon("src/Icons/login.png"), tabS.get("Авторизация"));

    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NewJFrame().setVisible(true);
            }
        });
    }
}

```

Файл CurrentUser.java:

```

package MyProgramm;
import java.util.ArrayList;

public class CurrentUser {
    private int userId;
    private String login;
    private String pass;
    private String dateUpdPass;
    private String privelegeLevel;
    private String status;
    private String secretWord;
    private String dateUpdWord;

    public CurrentUser(int userId, String login, String pass, String
dateUpdPass,
                        String privelegeLevel, String status,
String secretWord, String dateUpdWord)
    {
        this.userId = userId;
        this.login = login;
        this.pass = pass;
        this.dateUpdPass = dateUpdPass;
        this.privelegeLevel = privelegeLevel;
        this.status = status;
        this.secretWord = secretWord;
        this.dateUpdWord = dateUpdWord;
    }

    public String getPrivelegeLevel() {
        return privelegeLevel;
    }
}

```

```

    public int getUserId() {
        return userId;
    }

    public String getLogin() {
        return login;
    }

    public String getPassword() {
        return pass;
    }

    public String getDateUpdPass() {
        return dateUpdPass;
    }

    public String getStatus() {
        return status;
    }

    public String getSecretWord() {
        return secretWord;
    }

    public String getDateUpdWord() {
        return dateUpdWord;
    }

    public void setSecretWord(String word) {
        secretWord = word;
    }

    public void setNewPassword(String pass) {
        this.pass = pass;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}

```

Файл EmployeeData.java:

```

package MyProgramm;
import java.util.ArrayList;

public class EmployeeData {
    private int employeeId;
    private String surname;
    private String name;
    private String patronymic;
    private int passportId;

```

```

private String phone;
private String locality;
private String street;
private int houseNumber;
private int apartmentNumber;
private String currentPosition;
private String dateWorkExperience;
private int userId;
private String[] passportData;

public EmployeeData(int employeeId, String surname, String name,
String patronymic, int passportId,
String phone, String locality, String street, int
houseNumber, int apartmentNumber,
String currentPosition, String dateWorkExperience, int
userId, String[] passportData)
{
    this.employeeId = employeeId;
    this.surname = surname;
    this.name = name;
    this.patronymic = patronymic;
    this.passportId = passportId;
    this.phone = phone;
    this.locality = locality;
    this.street = street;
    this.houseNumber = houseNumber;
    this.apartmentNumber = apartmentNumber;
    this.currentPosition = currentPosition;
    this.dateWorkExperience = dateWorkExperience;
    this.userId = userId;
    this.passportData = passportData;
}

public void setSurname(String surname) {
    this.surname = surname;
}

public void setName(String name) {
    this.name = name;
}

public void setPatronymic(String patronymic) {
    this.patronymic = patronymic;
}

public void setPhone(String phone) {
    this.phone = phone;
}

public void setLocality(String locality) {
    this.locality = locality;
}

```

```

public void setStreet(String street) {
    this.street = street;
}

public void setHouseNumber(int house) {
    houseNumber = house;
}

public void setApartamentNumber(int apart) {
    apartamentNumber = apart;
}

public void setSurname(int apartament) {
    apartamentNumber = apartament;
}

public void setCurrentPosition(String position) {
    currentPosition = position;
}

public void setDateWorkExperience(String date) {
    dateWorkExperience = date;
}

public int getEmployeeId() {
    return employeeId;
}

public String getName() {
    return name;
}

public String getSurname() {
    return surname;
}

public String getPatronymic() {
    return patronymic;
}

public int getPassportId() {
    return passportId;
}

public String getPhone() {
    return phone;
}

public String getLocality() {
    return locality;
}

public String getStreet() {

```

```

        return street;
    }

    public int getHouseNumber() {
        return houseNumber;
    }

    public int getApartamentNumber() {
        return apartamentNumber;
    }

    public String getCurrentPosition() {
        return currentPosition;
    }

    public String getDateWorkExperience() {
        return dateWorkExperience;
    }

    public int getUserId() {
        return userId;
    }

    public String[] getPassportData() {
        return passportData;
    }
}

```

Файл History.java:

```

package MyProgramm;

import java.util.ArrayList;

public class History {
    private int actionId;
    private String actionType;
    private String actionDescription;
    private String actionDate;
    private String login;
    private int countActionsDisplay;
    private ArrayList<Integer> sortedHistory; //Индексы
отсортированных по типу действий

    public History(int actionId, String actionType, String
actionDescription, String actionDate, String login) {
        this.actionId = actionId;
        this.actionType = actionType;
        this.actionDescription = actionDescription;
        this.actionDate = actionDate;
        this.login = login;
    }
}

```

```

    }
    public History() {
        countActionsDisplay = 30;
        sortedHistory = new ArrayList<>();
    }

    public int getActionId() {
        return actionId;
    }

    public String getActionType() {
        return actionType;
    }

    public String getActionDescription() {
        return actionDescription;
    }

    public String getActionDate() {
        return actionDate;
    }

    public String getLogin() {
        return login;
    }

    public void setCountActionsDisplay(int count) {
        countActionsDisplay = count;
    }

    public int getCountActionsDisplay() {
        return countActionsDisplay;
    }

    public ArrayList<Integer> getSortedHistory() {
        return sortedHistory;
    }

    public void addNewItemIntoSortedHistory(int index) {
        sortedHistory.add(index);
    }

    public void clearSortedHistory() {
        sortedHistory.clear();
    }

    public void removeItemSortedHistory(int index) {
        for (int i=0; i < sortedHistory.size(); i++) {
            if(sortedHistory.get(i) == index) {
                sortedHistory.remove(i);
            }
        }
    }
}

```

}

Файл SearchEmployees.java:

```
package MyProgramm;
```

```
public class SearchEmployees {  
    private String surname;  
    private String name;  
    private String patronymic;  
    private String phone;  
    private String currentPosition;  
  
    public SearchEmployees(String surname, String name, String  
patronymic, String phone, String currentPosition) {  
        this.surname = surname;  
        this.name = name;  
        this.patronymic = patronymic;  
        this.phone = phone;  
        this.currentPosition = currentPosition;  
    }  
  
    public String getSurname() {  
        return surname;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public String getPatronymic() {  
        return patronymic;  
    }  
  
    public String getPhone() {  
        return phone;  
    }  
  
    public String getCurrentPosition() {  
        return currentPosition;  
    }  
}
```

Файл Client.java:

```
package MyProgramm;
```

```
import java.util.ArrayList;
```

```
public class Client {  
    private String surname;
```

```

private String name;
private String patronymic;
private String phone;

public Client(String surname, String name, String patronymic,
String phone) {
    this.surname = surname;
    this.name = name;
    this.patronymic = patronymic;
    this.phone = phone;
}

public String getSurname() { return surname; }
public String getName() { return name; }
public String getPatronymic () { return patronymic; }
public String getPhone() { return phone; }

public void setSurname(String surname) { this.surname = surname;
}

public void setName(String name) { this.name = name; }
public void setPatronymic (String patronymic) { this.patronymic =
patronymic; }
public void setPhone(String phone) { this.phone = phone; }
}

```

Файл Server.java:

```

package server;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {

    public static void main(String[] args) {
        ServerSocket serverSocket;
        try {
            serverSocket = new ServerSocket(8000);
            serverSocket.setReuseAddress(true);

            while (true) {
                Socket clientSocket = serverSocket.accept();
                //Ожидание подключения клиента
                System.out.println("New client connected " +
clientSocket.getInetAddress().getHostAddress());

                ClientHandler client = new
ClientHandler(clientSocket);

```

```

        new Thread(client).start();
    }
}
catch(IOException ex) {
    System.out.println("Connection error" + ex);
}
}
}

```

Файл ClientHandler.java:

```

package server;

import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.*;

public class ClientHandler implements Runnable {
    private final Socket clientSocket;
    private Connection conn;
    private Statement stmt;
    private String urlDB;
    private String usernameDB;
    private String passDB;
    private ResultSet rs;
    private DataInputStream inStream = null;
    private DataOutputStream outStream = null;

    public ClientHandler(Socket socket) {
        this.clientSocket = socket;
        urlDB = "jdbc:postgresql://localhost:5432/cafe";
        usernameDB = "postgres";
        passDB = "tedebe56";

        try {
            conn = DriverManager.getConnection(urlDB, usernameDB,
passDB);
            stmt =
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

```

```

        outputStream = new
DataOutputStream(clientSocket.getOutputStream());
        inputStream = new
DataInputStream(clientSocket.getInputStream());

    } catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }

}

public void auth() {
    try {
        String data;

        String login = inputStream.readUTF();
        String password = inputStream.readUTF();
        rs = stmt.executeQuery("SELECT * FROM users WHERE
login='" + login + "'");
        int count = 0;

        if(!rs.next()) {
            outputStream.writeUTF("userNotExistError"); //Отправка
кода ответа клиенту, если пользователя не существует
            outputStream.close();
            inputStream.close();
            clientSocket.close();
            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();
            return;
        }

        rs.beforeFirst();
        while (rs.next()) {
            if (!(rs.getString("login").equals(login) &&
rs.getString("pass").equals(password))) {
                //Если логин или пароль неверные

            outputStream.writeUTF("uncorrectLoginOrPasswordError");
                inputStream.close();
                outputStream.close();
                clientSocket.close();
                return;
            }
            else
            if(rs.getString("status").equals("Заблокирован")) {
                //Если учетная запись пользователя временно
заблокирована
                outputStream.writeUTF("userBlockedError");
            }
        }
    }
}

```

```

        inStream.close();
        outStream.close();
        clientSocket.close();
        return;
    }
    else {
        //Иначе отправляем результат запроса клиенту
        data = rs.getInt("userId") + "/" +
            rs.getString("login") + "/" +
            rs.getString("pass") + "/" +
            rs.getTimestamp("date_upd_pass") + "/" +
            rs.getString("privilege_level") + "/" +
            rs.getString("status") + "/" +
            rs.getString("secret_word") + "/" +
            rs.getTimestamp("date_upd_word");

        count++;
    }

    outStream.writeUTF("authOK");
    outStream.writeInt(count);
    outStream.writeUTF(data);
}

//Аудит входа в систему
int userID = inStream.readInt();
String username = "";

rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + userID); //Получаем логин пользователя
while (rs.next()) {
    username = rs.getString("login");
}

stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action date, user id)"
+ "VALUES ('Вход в систему', 'Пользователь @" +
username + " вошел в систему', "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ userID + ")");

//Получение информации о сотруднике
rs = stmt.executeQuery("SELECT * FROM employees WHERE
user_id=" + userID);
String employeeData = "";
while (rs.next()) {
    employeeData = rs.getInt("employee_id") + "/" +
        rs.getString("surname") + "/" +
        rs.getString("name") + "/" +
        rs.getString("patronymic") + "/" +
        rs.getInt("passport_id") + "/" +
        rs.getString("phone") + "/" +

```

```

        rs.getString("locality") + "/" +
        rs.getString("street") + "/" +
        rs.getInt("house_number") + "/" +
        rs.getInt("apartment_number") + "/" +
        rs.getString("current_position") + "/" +
        rs.getDate("date_work_experience") + "/" +
        rs.getInt("user_id");
    }
    outputStream.writeUTF(employeeData);

    //Получение паспортных данных сотрудника
    String[] buf = employeeData.split("/");
    String passportId = buf[4];
    String passportData = "";
    rs = stmt.executeQuery("SELECT * FROM employee_passport
WHERE passport_id=" + passportId);
    while (rs.next()) {
        passportData = rs.getInt("passport_id") + "/" +
            rs.getInt("passport_series") + "/" +
            rs.getInt("passport_number") + "/" +
            rs.getDate("date_birthday") + "/" +
            rs.getString("place_birth") + "/" +
            rs.getString("place_issue") + "/" +
            rs.getDate("date_issue") + "/" +
            rs.getString("departament_code") + "/" +
            rs.getString("registration_address");
    }
    outputStream.writeUTF(passportData);

    rs.close(); //Освобождение ресурсов
    stmt.close();
    conn.close();

    } catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}

public void logoutSystemAudit() {
    try {
        //Аудит выхода из системы
        int userID = inStream.readInt();
        String username = "";

        rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + userID); //Получаем логин пользователя
        while (rs.next()) {
            username = rs.getString("login");
        }

        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
+ "VALUES ('Выход из системы', 'Пользователь @"
+ username + " вышел из системы', "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ userID + ")");

        rs.close(); //Освобождение ресурсов
        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {

        Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
        null, ex);
    }
    finally {
        try {
            try {
                if (outStream != null) {
                    outStream.close();
                    clientSocket.close();
                }
                if (inStream != null) {
                    inStream.close();
                    clientSocket.close();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void getEmployeeData() {
}

```

```

public void viewProfileEmployeeAudit() {
    try {
        //Аудит выхода из системы
        int userID = inStream.readInt();
        String username = "";

        rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + userID); //Получаем логин пользователя
        while (rs.next()) {
            username = rs.getString("login");
        }

        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
+ "VALUES ('Просмотр сведений', 'Пользователь @"
+ username + " просмотрел сведения о сотруднике @" + username + "',
"
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ userID + ")");

        rs.close(); //Освобождение ресурсов
        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {
        Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void getAllHistory() {
    int count = 0;
    String data;
    ArrayList<String> result = new ArrayList<>();

```

```

        try {
            int limit = inStream.readInt(); //Чтение количества
отображаемых записей

            rs = stmt.executeQuery("SELECT history.action_id,
history.action_type, history.action_description,
history.action_date, users.login " +
                                "FROM history, users " +
                                "WHERE history.user_id =
users.userid " +
                                "ORDER BY action_date DESC LIMIT
" + limit);
            while (rs.next()) {
                data = rs.getInt("action_id") + "/" +
                    rs.getString("action_type") + "/" +
                    rs.getString("action_description") + "/" +
                    rs.getString("action_date") + "/" +
                    rs.getString("login");
                result.add(data);
                count++;
            }

            outStream.writeInt(count); //Запись числа полученных
строк

            for(String item : result) {
                outStream.writeUTF(item);
            }

            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();

        } catch (SQLException | IOException ex) {

            Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
            null, ex);
        }
        finally {
            try {
                if (outStream != null) {
                    outStream.close();
                    clientSocket.close();
                }
                if (inStream != null) {
                    inStream.close();
                    clientSocket.close();
                }
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

    }

    public void getHistoryByLogin(String login) {

        int count = 0;
        String data;
        ArrayList<String> result = new ArrayList<>();

        try {
            int limit = inStream.readInt(); //Чтение количества
            отображаемых записей

            rs = stmt.executeQuery("SELECT history.action_id,
            history.action_type, history.action_description,
            history.action_date, users.login " +
                                "FROM history, users " +
                                "WHERE history.user_id =
            users.userid AND users.login LIKE '%" + login + "%' " +
                                "ORDER BY action_date DESC LIMIT
            " + limit);

            if(!rs.next()) {
                outputStream.writeUTF("not_found"); //По запросу ничего
            не найдено

            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();
            outputStream.close();
            inStream.close();

            return;
        }

        outputStream.writeUTF("ok");
        rs.beforeFirst();
        while (rs.next()) {
            data = rs.getInt("action_id") + "/" +
                    rs.getString("action_type") + "/" +
                    rs.getString("action_description") + "/" +
                    rs.getString("action_date") + "/" +
                    rs.getString("login");
            result.add(data);
            count++;
        }

        outputStream.writeInt(count); //Запись числа полученных
        строк

        for(String item : result) {
            outputStream.writeUTF(item);
        }

        rs.close(); //Освобождение ресурсов
    }

```

```

        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {

        Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
            null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void createNewEmployee() {
    int userID;
    String login = "";
    String[] passport; //Паспортные данные сотрудника
    String[] employeeData; //Общие сведения о сотруднике
    String[] userData; //Данные учетной записи

    try {
        userID = inStream.readInt(); //Чтение кода сотрудника
        отправившего запрос

        passport = inStream.readUTF().split("/"); //Чтение
        паспортных данных
        employeeData = inStream.readUTF().split("/"); //Чтение
        общих данных о сотруднике
        userData = inStream.readUTF().split("/"); // Чтение
        данных об учетной записи

        if (checkPassport(Integer.parseInt(passport[0]),
            Integer.parseInt(passport[1])) == -1) {
            outStream.writeUTF("passportNotAvailable");

            rs = stmt.executeQuery("SELECT login FROM users,
            employees, employee_passport " +
                                "WHERE
            users.userid=employees.user_id AND
            employees.passport_id=employee_passport.passport_id " +

```

```

        "AND
employee_passport.passport_series=" + passport[0] +
        " AND
employee_passport.passport_number=" + passport[1]);
        while (rs.next()) {
            login = rs.getString("login");
        }
        outputStream.writeUTF(login);

        outputStream.close();
        inputStream.close();
        clientSocket.close();
        return;
    }

    if(checkPhone(employeeData[3]) == -1) {
        outputStream.writeUTF("phoneNotAvailiable");

        rs = stmt.executeQuery("SELECT login FROM users,
employees " +
                                "WHERE
users.userid=employees.user_id AND employees.phone='" +
employeeData[3] + "'");
        while (rs.next()) {
            login = rs.getString("login");
        }
        outputStream.writeUTF(login);

        outputStream.close();
        inputStream.close();
        clientSocket.close();
        return;
    }

    if(checkLoginAvailibale(userData[0]) == -1) {
        outputStream.writeUTF("loginNotAvailiable");
        outputStream.close();
        inputStream.close();
        clientSocket.close();
        return;
    }

    //Добавление паспортных данных в систему
    stmt.executeUpdate("INSERT INTO employee_passport
(passport_series, " +
                                "passport_number, " +
                                "date_birthday, " +
                                "place_birh, " +
                                "place_issue, " +
                                "date_issue, " +
                                "departament_code, " +
                                "registration_address) " +

```

```

        "VALUES (" + passport[0] + ", " +
passport[1] + ", '" + passport[2] + "', '" +
        passport[3] + "', '" +
passport[4] + "', '" + passport[5] + "', '" + passport[6] + "', '" +
passport[7] + "')";);

```

```

//Добавление данных об учетной записи
stmt.executeUpdate("INSERT INTO users (login, pass,
date_upd_pass, privilege_level, status, secret_word, date_upd_word)
" +
        "VALUES ('" + userData[0] + "', '" +
userData[1] + "', to_timestamp(to_char(current_timestamp, 'YYYY-MM-
DD HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS'), "
        + "'" + userData[4] + "', '"
+ userData[3] + "', '" + userData[2] + "',
to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'))"););

```

```

//Добавление данных о сотруднике
int user_id = 0;
int passport_id = 0;
rs = stmt.executeQuery("SELECT userid FROM users WHERE
login='" + userData[0] + "'");
while(rs.next()) {
    user_id = rs.getInt("userid");
}

```

```

rs = stmt.executeQuery("SELECT passport_id FROM
employee_passport WHERE passport_series=" + passport[0] + " AND
passport_number=" + passport[1]);
while (rs.next()) {
    passport_id = rs.getInt("passport_id");
}

```

```

stmt.executeUpdate("INSERT INTO employees (surname,
name, patronymic, passport_id, phone, locality, street, " +
"house_number, apartament_number, " +
"current_position, date_work_experience, user_id) " +
        "VALUES ('" + employeeData[0] + "',
'" + employeeData[1] + "', '" + employeeData[2] + "', "
        + passport_id + ", '" +
employeeData[3] + "', '" + employeeData[4] + "', '"
        + employeeData[5] +
"', " + employeeData[6] + ", " + employeeData[7] + "', '" +
        employeeData[8] +
"', '" + employeeData[9] + "', " + user_id + ")"););

```

```

outStream.writeUTF("success"); //Отправка ответа клиенту

```

```

//Аудит внесения сотрудника в систему

```

```

        user_id = inStream.readInt(); //Чтение номера
сотрудника, внесшего изменения
        String username = "";

        rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + user_id); //Получаем логин пользователя
        while (rs.next()) {
            username = rs.getString("login");
        }

        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
            + "VALUES ('Внесение сведений', 'Пользователь @"
+ username + " зарегистрировал нового сотрудника с логином @" +
userData[0] + "', "
            +
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
            + user_id + ")");

    } catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public int checkPhone(String phone) {
    try {
        rs = stmt.executeQuery("SELECT * FROM employees WHERE
phone='" + phone + "'");
        if(rs.next()) {
            return -1;
        }
    }

    } catch (SQLException ex) {

```

```

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }

    return 0;
}

public int checkPassport(int series, int number) {
    try {
        rs = stmt.executeQuery("SELECT * FROM employee_passport
WHERE passport_series=" + series + " AND passport_number=" +
number);
        if(rs.next()) {
            return -1;
        }

    } catch (SQLException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    return 0;
}

public int checkLoginAvailibale(String login) {
    try {
        rs = stmt.executeQuery("SELECT * FROM users WHERE
login='" + login + "'");
        if(rs.next()) {
            return -1;
        }

    } catch (SQLException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    return 0;
}

public void updateEmployeeData() { //Обновление общих сведений
сотрудника
    try {
        int user_id;
        String[] employeeData;
        String phone;
        String oldPhone = "";

        user_id = inStream.readInt(); // Чтение ID пользователя,
выполнившего изменение
        employeeData = inStream.readUTF().split("/"); //Чтение
изменяемых данных
    }
}

```

```

        phone = employeeData[3]; //Получение номера телефона
сотрудника

        //Получение старого номера телефона
        rs = stmt.executeQuery("SELECT phone FROM employees
WHERE user_id=" + employeeData[8]);
        while(rs.next()) {
            oldPhone = rs.getString("phone");
        }

        rs = stmt.executeQuery("SELECT phone FROM employees
WHERE phone='" + phone + "'");
        if(rs.next()) {
            if(!oldPhone.equals(phone)) {
                outputStream.writeUTF("phoneUsedError");
                rs.close(); //Освобождение ресурсов
                stmt.close();
                conn.close();
                outputStream.close();
                inputStream.close();
                clientSocket.close();
                return;
            }
        }

        //Обновление номера телефона
        stmt.executeUpdate("UPDATE employees SET surname='" +
employeeData[0] + "', name='" + employeeData[1] + "', patronymic='"
+
            employeeData[2] + "', phone='" + employeeData[3]
+ "', locality='" + employeeData[4] + "', street='" +
            employeeData[5] + "', house_number=" +
employeeData[6] + ", apartament_number=" + employeeData[7] + " WHERE
phone='" +
            oldPhone + "'");

        outputStream.writeUTF("success"); //Успешное изменение
данных

        //Аудит внесения изменений в профиль пользователя
        String username = "";
        String empl = "";

        rs = stmt.executeQuery("SELECT users.login FROM users,
employees WHERE employees.user_id=users.userid "
            + "AND employees.phone='" + employeeData[3] +
            "'");
        while (rs.next()) {
            empl = rs.getString("login");
        }

```

```

        rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + user_id); //Получаем логин пользователя
        while (rs.next()) {
            username = rs.getString("login");
        }

        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
+ "VALUES ('Изменение сведений', 'Пользователь
@" + username + " изменил общие сведения о сотруднике с логином @" +
empl + "', "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");

        rs.close(); //Освобождение ресурсов
        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void updateSecretWord() { //Обновление кодового слова
сотрудника
    try {
        String login;
        String newSecretWord;
        int user_id;
        String usnm = "";

        login = inStream.readUTF(); // Чтение логина
пользователя, выполнившего изменение

```

```

        newSecretWord = inStream.readUTF(); //Чтение нового
кодowego слова
        user_id = inStream.readInt(); // Чтение ID пользователя,
выполнившего изменение кодowego слова

        stmt.executeUpdate("UPDATE users SET secret_word='" +
newSecretWord + "'",
date_upd_word=to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD
HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS')"
+ " WHERE login='" + login + "'");

        //Аудит изменения кодowego слова
        rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + user_id); //Получаем логин пользователя
        while (rs.next()) {
            usnm = rs.getString("login");
        }

        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
+ "VALUES ('Изменение сведений', 'Пользователь
@" + usnm + " изменил кодговое слово сотрудника с логином @" + login
+ "', "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");

        rs.close(); //Освобождение ресурсов
        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {

        Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    public void updatePassword() { //Обновление пароля сотрудника
        try {
            String login;
            String newPassword;
            int user_id;
            String usrm = "";

            login = inStream.readUTF(); // Чтение логина
пользователя, над которым выполняются изменения
            newPassword = inStream.readUTF(); //Чтение нового
кодированного слова
            user_id = inStream.readInt(); // Чтение ID пользователя,
выполнившего изменение кодированного слова

            stmt.executeUpdate("UPDATE users SET pass='" +
newPassword + "',
date_upd_pass=to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD
HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS')"
+ " WHERE login='" + login + "'");

            //Аудит изменения кодированного слова
            rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + user_id); //Получаем логин пользователя, выполнившего
изменение
            while (rs.next()) {
                usrm = rs.getString("login");
            }

            stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
+ "VALUES ('Изменение сведений', 'Пользователь
@" + usrm + " изменил пароль сотрудника с логином @" + login + "',
"
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");

            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();

        } catch (SQLException | IOException ex) {

            Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
            null, ex);
        }
        finally {
            try {
                if (outStream != null) {

```

```

        outStream.close();
        clientSocket.close();
    }
    if (inStream != null) {
        inStream.close();
        clientSocket.close();
    }
}
catch (IOException e) {
    e.printStackTrace();
}
}

public void restorePassword() {
    try {
        String login;
        String secretWord;
        String newPassword;
        String wordFromDB = "";
        int user_id = 0;

        login = inStream.readUTF(); // Чтение логина
        пользователь, который желает изменить пароль
        secretWord = inStream.readUTF(); // Чтение кодового
        слова
        newPassword = inStream.readUTF(); // Чтение нового
        пароля

        rs = stmt.executeQuery("SELECT * FROM users WHERE
login='" + login + "'");
        if(!rs.next()) {
            outStream.writeUTF("user_not_found");
            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();
            return;
        }

        rs.beforeFirst();
        while(rs.next()) {
            wordFromDB = rs.getString("secret_word");
        }
        if(!wordFromDB.equals(secretWord)) {
            outStream.writeUTF("incorrectSecretWord");
            rs.close(); //Освобождение ресурсов
            stmt.close();
            conn.close();
            return;
        }

        rs = stmt.executeQuery("SELECT userid FROM users WHERE
login='" + login + "'");

```

```

        while(rs.next()) {
            user_id = rs.getInt("userid");
        }

        stmt.executeUpdate("UPDATE users SET pass='" +
newPassword + "',
date_upd_pass=to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD
HH24:MI:SS'), 'YYYY-MM-DD HH24:MI:SS') WHERE login='" + login +
"'"); //Обновление пароля

        //Аудит восстановления пароля
        stmt.executeUpdate("INSERT INTO history (action_type,
action_description, action_date, user_id)"
            + "VALUES ('Изменение сведений', 'Пользователь
@" + login + " восстановил доступ в систему', "
            +
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
            + user_id + ")");

        outputStream.writeUTF("success");

        rs.close(); //Освобождение ресурсов
        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            try {
                if (outStream != null) {
                    outStream.close();
                    clientSocket.close();
                }
                if (inStream != null) {
                    inStream.close();
                    clientSocket.close();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void getEmployeesList() {
    try {
        //String data = "";
        int countRows = 0;
        ArrayList<String> data = new ArrayList<>();
    }
}

```

```

rs = stmt.executeQuery("SELECT * FROM employees");
while (rs.next()) {
    data.add(rs.getString("surname") + "/" +
            rs.getString("name") + "/" +
            rs.getString("patronymic") + "/" +
            rs.getString("phone") + "/" +
            rs.getString("current_position"));
    countRows++;
}

outStream.writeInt(countRows); //Отправка числа
считанных строк
for(String item : data) {
    outStream.writeUTF(item); //Отправка полученных
данных
}

rs.close(); //Освобождение ресурсов
stmt.close();
conn.close();

} catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
}
finally {
    try {
        if (outStream != null) {
            outStream.close();
            clientSocket.close();
        }
        if (inStream != null) {
            inStream.close();
            clientSocket.close();
        }
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

}

public void blockUnlockEmployee() {
    try {
        String login = "";
        int user_id = 0;
        int userByBlockUnlock = 0;
        String username = "";
        String oldStatus = "";

```

```

        user_id = inStream.readInt(); // Чтение номера
пользователя, выполнившего блокировку или разблокировку
        userByBlockUnlock = inStream.readInt(); // Чтение номера
пользователя, чья учетная запись будет заблок./разблок.

        rs = stmt.executeQuery("SELECT status, login FROM users
WHERE userid=" + userByBlockUnlock); //Получение текущего статуса
пользователя
        while(rs.next()) {
            oldStatus = rs.getString("status");
            username = rs.getString("login");
        }

        //Аудит блокировки или разблокировки пользователя
        rs = stmt.executeQuery("SELECT login FROM users WHERE
userId=" + user_id); //Получаем логин пользователя, выполнившего
изменение
        while (rs.next()) {
            login = rs.getString("login");
        }

        if(oldStatus.equals("Активен")) {
            stmt.executeUpdate("UPDATE users SET
status='Заблокирован' WHERE userid=" + userByBlockUnlock);

            //Аудит блокировки учетной записи
            stmt.executeUpdate("INSERT INTO history
(action_type, action_description, action_date, user_id) "
+ "VALUES ('Изменение сведений', 'Пользователь
@" + login + " заблокировал учетную запись сотрудника @" + username +
", "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");
        }
        else {
            stmt.executeUpdate("UPDATE users SET
status='Активен' WHERE userid=" + userByBlockUnlock);

            //Аудит блокировки учетной записи
            stmt.executeUpdate("INSERT INTO history
(action_type, action_description, action_date, user_id) "
+ "VALUES ('Изменение сведений', 'Пользователь
@" + login + " разблокировал учетную запись сотрудника @" + username
+ ", "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");
        }

        outStream.writeUTF("success");

```

```

        rs.close(); //Освобождение ресурсов
        stmt.close();
        conn.close();

    } catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
    }
    finally {
        try {
            if (outStream != null) {
                outStream.close();
                clientSocket.close();
            }
            if (inStream != null) {
                inStream.close();
                clientSocket.close();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void createClient() {
    try {
        String login = "";
        int user_id = 0;
        int userByBlockUnlock = 0;
        String username = "";
        String oldStatus = "";

        user_id = inStream.readInt(); // Чтение номера
        пользователя, выполнившего блокировку или разблокировку
        userByBlockUnlock = inStream.readInt(); // Чтение номера
        пользователя, чья учетная запись будет заблок./разблок.

        rs = stmt.executeQuery("SELECT status, login FROM users
WHERE userid=" + userByBlockUnlock); //Получение текущего статуса
        пользователя
        while(rs.next()) {
            oldStatus = rs.getString("status");
            username = rs.getString("login");
        }

        //Аудит блокировки или разблокировки пользователя
        rs = stmt.executeQuery("SELECT login FROM users WHERE
userid=" + user_id); //Получаем логин пользователя, выполнившего
        изменение
        while (rs.next()) {

```

```

        login = rs.getString("login");
    }

    if(oldStatus.equals("Активен")) {
        stmt.executeUpdate("UPDATE users SET
status='Заблокирован' WHERE userid=" + userByBlockUnlock);

        //Аудит блокировки учетной записи
        stmt.executeUpdate("INSERT INTO history
(action_type, action_description, action_date, user_id) "
+ "VALUES ('Изменение сведений', 'Пользователь
@" + login + " заблокировал учетную запись сотрудника @" + username +
"', "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");
    }
    else {
        stmt.executeUpdate("UPDATE users SET
status='Активен' WHERE userid=" + userByBlockUnlock);

        //Аудит блокировки учетной записи
        stmt.executeUpdate("INSERT INTO history
(action_type, action_description, action_date, user_id) "
+ "VALUES ('Изменение сведений', 'Пользователь
@" + login + " разблокировал учетную запись сотрудника @" + username
+ ", "
+
"to_timestamp(to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS'),
'YYYY-MM-DD HH24:MI:SS'), "
+ user_id + ")");
    }

    outputStream.writeUTF("success");

    rs.close(); //Освобождение ресурсов
    stmt.close();
    conn.close();

} catch (SQLException | IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
}
finally {
    try {
        if (outStream != null) {
            outStream.close();
            clientSocket.close();
        }
        if (inStream != null) {
            inStream.close();
        }
    }
}

```

```

        clientSocket.close();
    }
}
catch (IOException e) {
    e.printStackTrace();
}
}

@Override
public void run() {
    try {
        int requestCode = inStream.readInt(); // Чтение кода
запроса

        switch(requestCode) { //Анализ кода ответа
            case 10 -> {
                auth(); //Вызов функции авторизации
            }
            case 11 -> {
                //Вызов функции регистрации нового сотрудника
            }
            case 12 -> {
                //Вызов функции изменения данных действующего
сотрудника
            }
            case 13 -> {
                //Запрос полных сведений о сотруднике
                getEmployeeData();
            }
            case 14 -> {
                //Получение истории действий в системе
                getAllHistory();
            }
            case 15 -> {
                //Получение истории действий в системе по ЛОГИНУ
                getHistoryByLogin(inStream.readUTF()); //Чтение
введенного логина
            }
            case 16 -> {
                //Регистрация нового сотрудника в системе
                createNewEmployee();
            }
            case 17 -> {
                //Изменение общих сведений о сотруднике
                updateEmployeeData();
            }
            case 18 -> {
                //Изменение кодового слова сотрудника
                updateSecretWord();
            }
            case 19 -> {
                //Обновление пароля сотрудника

```

```

        updatePassword();
    }
    case 20 -> {
        //Получение списка зарегистрированных
сотрудников
        getEmployeesList();
    }
    case 21 -> {
        //Восстановление доступа к системе (до
авторизации)
        restorePassword();
    }
    case 22 -> {
        //Блокировка или разблокировка учетной записи
сотрудника
        blockUnlockEmployee();
    }
    case 23 -> {
        //Создание профиля клиента
        createClient();
    }
    case 1 -> {
        //Выход из системы
        logoutSystemAudit();
    }
    case 2 -> {
        //Аудит просмотра профиля сотрудника
        viewProfileEmployeeAudit();
    }
    case 500 -> {
        clientSocket.close();
    }
    default -> {

    }
}
} catch (IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
null, ex);
}

}
}

```

Приложение В

(Обязательное)

Система автоматизации работы заведений общественного питания

Презентация

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		169

Стартап-проект. Система автоматизации работы заведений общественного питания

БАКАЛАВРСКАЯ РАБОТА

Выполнил: Мартынов Н.В.,
Студент гр. 20ВВП1
Руководитель: Карамышева Н.С.,
к.т.н., доцент каф. ВТ

Пенза,
17 июня 2024 г.

Актуальность разработки

Каждый день сотрудникам заведений общественного питания малого масштаба приходится пручую, если отсутствуют какие-либо средства автоматизации, решить немаловажные задачи: учет числа совершенных заказов, подсчет выручки заведения в конце рабочего дня, фиксирование сведений об объемах приготовленных блюд, прием продукции от поставщиков и документирование данного процесса в виде накладной, осуществление контроля за исполнением обязанностей сотрудников предприятия и др. Значительная информация, связанная с выполнением вышеперечисленных задач, хранится и обрабатывается сотрудниками заведения в бумажной форме.

Разработанное мною решение призвано решить вышеперечисленные проблемы. Автоматизированная система будет иметь невысокую стоимость по сравнению с конкурентами, устанавливаться под любую платформу, недорогое сопровождение и обслуживание, настройку под сторонние внешние устройства (принтеры, сканеры штрих-кодов, фискальный накопитель). Также система имеет АС имеет возможность работы без сети Интернет при помощи хранения всех данных локально. Немаловажным является тот факт, что программа будет обладать набором базовых функций, которые удовлетворят большую часть потребностей по автоматизации работы малого заведения общепита.

Изм.	Лист	№ докум.	Подпись	Дата

Цель проекта и постановка задач

Цель проекта: разработать систему для автоматизации процесса выполнения повседневных и административных задач заведений общественного питания малого масштаба.

Функциональные требования, выдвигаемые к разрабатываемой системе:

- возможность работы с профилями посетителей заведения (регистрация новых клиентов, изменение учетных данных, удаление профиля клиента, просмотр информации, хранящейся в профиле клиента);
- прием и работа с заказами (создание нового заказа путем добавления в него нужных клиенту позиций меню, изменение добавленных в заказ позиций, отмена совершенного заказа, выбор профиля посетителя для возможности списания, начисления бонусных баллов, а также для сохранения информации в истории заказов);
- конфигурирование меню заведения (добавление новых позиций в меню, изменение действующих позиций меню, аннулирование действующих позиций);
- возможность работы с заявками на приобретение нужной продукции (просмотр списка заявок, создание заявок со списком необходимых продуктов, закрытие заявки как результат закупки продукции, отклонение заявки, принятие заявки в работу);

Цель проекта и постановка задач

Функциональные требования, выдвигаемые к разрабатываемой системе:

- администрирование учетных записей сотрудников (просмотр информации в профиле сотрудника, внесение персональных данных о сотруднике, изменение учетных данных профиля, автоматическое создание учетной записи сотрудника для входа в систему в момент первичной регистрации, аннулирование учетной записи сотрудника при его увольнении);
- работа с поставщиками продуктов (добавление новых поставщиков, изменение сведений о действующих поставщиках, удаление информации о них, прием товаров путем формирования соответствующей накладной);
- анализ данных, связанных с работой заведения (просмотр числа заказов за выбранный период, формирование краткой выписки с итогами работы заведения или сотрудника за выбранный период, получение информации об активности сотрудников, выявление популярности (актуальности) той или иной позиции меню заведения);

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.17.001 ПЗ

Лист

171

Цель проекта и постановка задач

- Технические требования, выдвигаемые к разрабатываемой системе:
- удобный и интуитивно понятный интерфейс для работы с программой;
 - расположение графических элементов должно предоставлять пользователю быстрое ориентирование в интерфейсе;
 - возможность масштабирования интерфейса в процессе работы;
 - разработка программы является десктопным решением;
 - наличие возможности сохранения данных локально, пока отсутствует подключение к серверу;
 - операционная система: Windows 7 и более поздние версии;
 - оперативная память: не менее 512 МБ ОЗУ;
 - независимое дисковое пространство: не менее 1,5 ГБ ПЗУ;
 - процессор: Intel Pentium III 800 MHz;
 - разрешение экрана: 1024 x 728.

Диаграмма вариантов использования



Логическая схема БД



Физическая схема БД



Развертывание системы

Вся работа с приложением будет осуществляться в виде диалога с пользователем. Диалоговая система состоит из трех ключевых компонентов:

- клиентская часть, которая обеспечивает пользовательский интерфейс для взаимодействия с системой, включая кнопки, поля ввода, выпадающие списки, таблицы, чекбоксы и др.;
- серверная часть, которая обрабатывает данные, отправленные пользователем, выполняет необходимые алгоритмы и задачи;
- база данных, где хранятся данные с определенными правилами и принципами управления данными.

Система управления базами данных (СУБД) представляет собой набор программных инструментов, которые позволяют создавать, управлять и администрировать базы данных. Другими словами, СУБД предоставляет возможности для организации, контроля и администрирования данных в базах данных.

Диаграмма развертывания приведена на следующем слайде.

Диаграмма развертывания



Источники дохода проекта

Автоматизированная система будет иметь следующие источники доходов:

- продажа лицензий: можно продавать лицензии на использование системы автоматизации кафе и ресторанов различным заведениям в виде на ежемесячную или ежегодную плату за пользование;
- подписка на обновления и поддержку: предложение услуги по обновлению программного обеспечения и технической поддержке системы за ежемесячную плату;
- установка и настройка: предоставление услуг по установке и настройке системы автоматизации в кафе и ресторанах за отдельную плату;
- обучение и консультации: предложение обучающих курсов и консультаций по использованию системы для персонала кафе и ресторанов за отдельную плату.

Исходя из данных о количестве заведений общественного питания в Пензенской области, а согласно источнику 2ГИС их около 270, и предположения, что около 50% из них готовы приобрести информационную систему, можно оценить, что примерно 135 организаций станут будущими клиентами.

Детализация источников дохода приведена на следующем слайде.

Источники дохода

Доход высчитывается по формуле

$$D = K \cdot C \cdot (t)$$

где D – доход;

K – количество потребителей;

C – установленная цена;

Доход от продажи лицензий

$$135 \text{ лр} \cdot 15000 \text{ р} = 3375000 \text{ р}$$

Доход от поддержки и подписки на обновления

$$1000 \text{ р} \cdot 135 \text{ лр} = 135000 \text{ р/мес}$$

Доход от установки и настройки программного обеспечения

$$3000 \text{ р} \cdot 135 \text{ лр} = 405000 \text{ р}$$

Доход от обучения и консультаций

$$10000 \text{ р} \cdot 135 \text{ лр} = 1350000 \text{ р}$$

Таким образом, общий ожидаемый доход в Пензенской области составит 5 265 000 рублей единовременно и 135 000 рублей ежемесячно.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.17.001 ПЗ

Лист

175

Интерфейс программы



Окно авторизации в программе

Интерфейс программы



Просмотр истории действий в системе

Интерфейс программы

Создание профиля сотрудника

Общие сведения

Имя: Фамилия:
 Дата рождения: Пол:
 Контакт: Место: Инициалы:
 Телефон:

Детальная информация

Пол: Место: Дата рождения:
 Место работы:
 Должность: Дата приема: На испытательный:
 Адрес проживания:

Информация о сотруднике

Подпись: Подпись:
 Подпись: Подпись:
 Подпись: Подпись:

Удостоверение

Инициалы: Дата:

Регистрация нового сотрудника в системе

Заключение

Цель работы – разработка собственной системы автоматизации работы заведений общественного питания малого масштаба с целью упрощения процесса выполнения повседневных и управленческих задач.

Для достижения поставленной цели было разработано настольное приложение, отвечающее следующим поставленным заданием: работа в системе под учетной записью сотрудника, управление учетными записями сотрудников, регистрация и ведение истории заказов посетителей, работа с меню заведения, отслеживание истории действий в системе, прием и обработка заказов клиентов, внесение сведений о блюдах на остаток, управление заявками на продукты.

В результате проведенной работы разработана автоматизированная система с графическим интерфейсом Swing GUI, клиент-серверной архитектурой на языке Java. Для хранения данных применена СУБД PostgreSQL в связке с графическим клиентом pgAdmin4.

Спасибо за внимание!

					ПГУ 1.090301.17.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		179