Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
Ассистент кафедры ЭИ
_____ Д.И. Булыга
___.____.2023

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:
«СИСТЕМА РАСЧЁТА ЗАРАБОТНОЙ ПЛАТЫ»

Выполнил студент группы 124402
ЧЕРНУШЕВИЧ Артём Анатольевич

_____
(подпись студента)

Курсовая работа представлена на проверку ___.____.2023

_____
(подпись студента)

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные организации и предприятия в условиях цифровой экономики нуждаются в эффективных способах управления персоналом и контроля за выплатой заработной платы. В данном проекте рассматривается возможность создания автоматизированной системы расчета заработной платы, которая повысит результативность использования ресурсов и обеспечит прозрачность трудовых процессов.

Целью курсового проекта является разработать систему расчета заработной платы с использованием клиент-серверного подхода, которая будет обеспечивать эффективность, надежность и простоту использования, а также автоматизирует рутинные операции и снизит временные затраты на управление персоналом.

Для достижения этой цели необходимо решить следующие задачи:

– разработать клиент-серверную архитектуру, обеспечивающую быстрое взаимодействие между пользователем и сервером;

– создать многопоточный сервер для одновременной обработки запросов и повышения скорости работы системы;

– спроектировать базу данных, соответствующую стандартам нормализации и позволяющую эффективно хранить и обрабатывать информацию о зарплате;

– реализовать бизнес-логику на сервере и обеспечить безопасность доступа к данным;

– интегрировать удобный графический интерфейс для пользователя.

Результат этапа программирования представляется в форме проекта в среде разработки приложений на языке Java (IntelliJ IDEA).

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В условиях современного бизнеса эффективное управление персоналом и оптимизация процессов, связанных с заработной платой, являются ключевыми компонентами успешной деятельности предприятий. Предметная область данной курсовой работы охватывает систему расчета заработной платы, предназначенной для управления финансовыми аспектами организации и обеспечения порядка в вопросах оплаты труда.

Основные функции и задачи системы:

– расчёт заработной платы;

– управление сотрудниками организации;

– соблюдение норм и правил оплаты труда;

– автоматизация отчетности;

– обеспечение прозрачности и доступности данных;

– представление данных в виде диаграмм или графиков;

– просмотр сотрудниками актуальной информации.

В итоге, решение задач в предметной области данного проекта направлено на создание современной, надежной и удобной в использовании системы управления заработной платой, способной соответствовать требованиям современного бизнеса и повышать эффективность организации.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ

2.1 Постановка задачи

Постановка задачи для разработки системы учета заработной платы содержит в себе следующие аспекты:

– анализ возможных проблем при создании системы;

– определение бизнес-целей, требований, функциональных потребностей;

– обеспечение безопасности хранения информации о сотрудниках;

– оценка ограничений в функционале для обычных пользователей и администраторов, возможные риски при внедрении системы;

– определение интерфейса программы, организация функционала, создание понятного, удобного и простого расположения кнопок в целях удобства пользования.

2.2 Обзор методов решения поставленной задачи

Для успешной разработки системы учета заработной платы, необходимо внимательное рассмотрение и применение различных методов, обеспечивающих эффективность, безопасность и удобство использования. Обзор методов решения задачи включает следующие аспекты:

– Анализ проблем и рисков;

Целью аспекта является идентификация потенциальных трудностей, которые могут возникнуть в процессе разработки и внедрения, с последующими шагами по их предотвращению или решению.

– Определение бизнес-целей и требований;

Целью аспекта является получение четкого понимания бизнес-целей, функциональных требований и потребностей пользователей для их успешной интеграции в систему.

– Обеспечение безопасности;

Целью аспекта является защита конфиденциальных данных сотрудников от несанкционированного доступа, обеспечение безопасности хранения и передачи информации.

– Оценка ограничений и рисков;

Целью аспекта является оценка ограничений в функционале для различных пользовательских ролей, выявление возможных рисков и разработка стратегий их предотвращения.

– Определения интерфейса и функционала;

Целью аспекта является разработка интуитивно понятного пользовательского интерфейса с удобным распределением элементов, обеспечивающего комфортное взаимодействие пользователей с системой.

– Использование современных технологий.

Целью аспекта является обеспечение высокой производительности, масштабируемости и поддержки системы, а также использование передовых методов разработки.

Обобщая, успешное решение задачи по разработке системы учета заработной платы требует комплексного подхода, объединяющего аналитические, технические и дизайнерские методы для достижения оптимальных результатов.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

3.1 Теоретическая часть

IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

В основе IDEF0-методологии лежат 4 основных понятия:

– функциональный блок;

– интерфейсная дуга (стрелка);

– декомпозиция;

– глоссарий.

Функциональный блок олицетворяет некоторую конкретную функцию или работу в рамках рассматриваемой системы. Каждый блок в рамках единой системы имеет уникальный номер, а каждая его сторона имеет свое назначение. Наименование осуществляется оборотом глагола или существительного.

Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображаемую функциональным блоком.

Графически изображается в виде однонаправленной стрелки.

Каждая дуга должна иметь свое уникальное название, сформулированное оборотом существительного (должно отвечать на вопросы кто? что?). В зависимости от того, к какой стороне блока она подходит, интерфейсная дуга будет являться входящей, выходящей, управления, механизма.

Принцип декомпозиции применяется при разбиении сложных процессов на составляющие его функции. При этом уровень детализации определяется непосредственно разработчиком модели.

Модель IDEFO всегда начинается с рассмотрения системы как единого целого, т.е. одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма называется контекстной, она обозначается идентификатором А-0. Для определения границ системы на контекстной диаграмме обязательно должны быть цель и точка зрения.

Для каждого из элементов в IDEF0 существует стандарт, подразумевающий создание и поддержку набора соответствующих определений, ключевых слов, повествований, изложений, которые характеризуют объект, отраженный данным элементом. Этот набор - глоссарий, являющийся описанием сущности данного элемента.

Точка зрения - позиция, с которой будет строится модель. В качестве точки зрения берется взгляд человека, который видит систему в нужном для моделирования аспекте.

Как правило, выбирается точка зрения человека, ответственного за выполнение моделируемой работы.

Между целью и точкой зрения должно быть жесткое соответствие.

3.2 Практическая часть

Для построения контекстной диаграммы процесса необходимо проанализировать что должно участвовать в процессе, кем он должен выполняться и на что ориентироваться, а также как достичь определенного результата на выходе.

Для формирования группы необходимы данные сотрудника. В ходе процессов данные будут занесены в базу данных, где и будут храниться. У каждого сотрудника будет создано свое уникальное имя пользователя, по которому он будет попадать в свой личный кабинет.

Диаграмма, содержимое которой описано выше, представлена на рисунке 3.1.

Рисунок 3.1 – Контекстная диаграмма процесса «Рассчитать заработную плату»

Декомпозиция процесса «Рассчитать заработную плату» состоит из 5 функций:

– получение данных о пользователе;

– расчёт заработной платы;

– вычет налога;

– конвертация в другие валюты;

– вывод полученной зарплаты.

Декомпозиция процесса «Рассчитать заработную плату» представлена на рисунке 3.2.

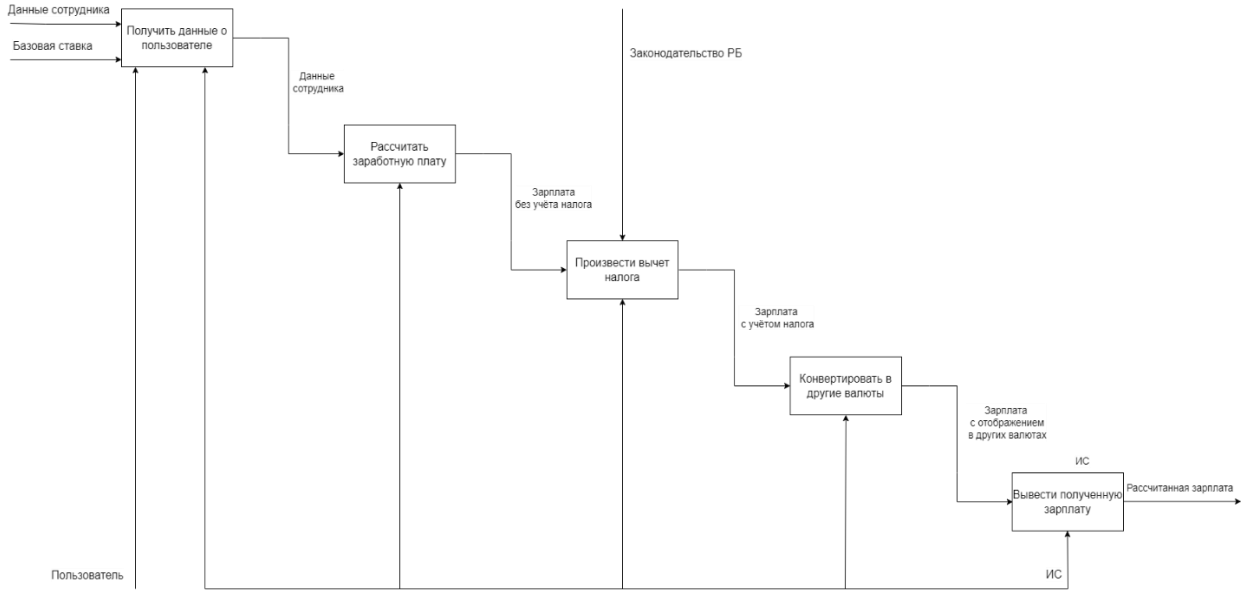

Рисунок 3.2 – Диаграмма декомпозиции процесса «рассчитать заработную плату»

Декомпозиция процесса «Получить данные о пользователе» состоит из 3 функций:

– считать данные сотрудника;

– считать базовую ставку должности сотрудника;

– передать данные в систему.

Декомпозиция процесса «Получить данные о пользователе» представлена на рисунке 3.3.

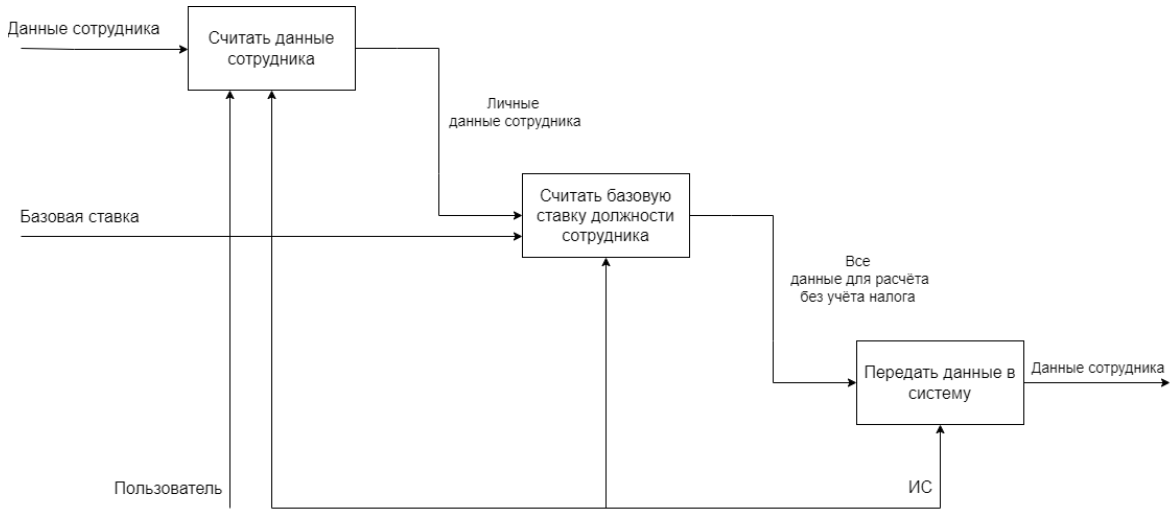

Рисунок 3.3 – Диаграмма декомпозиции процесса «Получить данные о пользователе»

Декомпозиция процесса «Считать данные сотрудника» состоит из 3 функций:

– проверить данные на существование;
– сделать запись в базу данных;
– сделать запрос в базу данных.

Декомпозиция процесса «Считать данные сотрудника» представлена на рисунке 3.4.

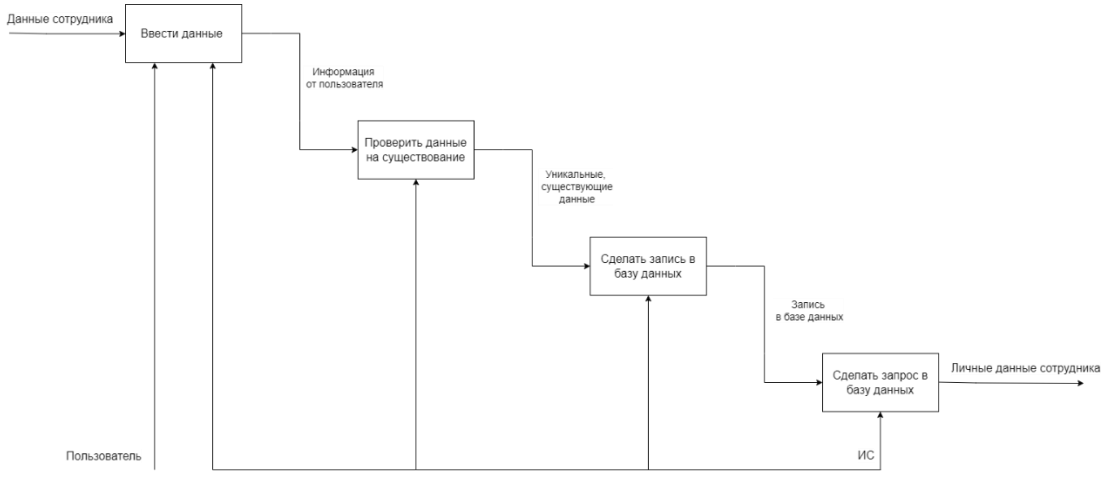

Рисунок 3.4 – Диаграмма декомпозиции процесса «Считать данные сотрудника»

Декомпозиция процесса «Произвести вычет налога» состоит из 3 функций:

– считать страну сотрудника;
– считать налог на зарплату;
– произвести вычет налога.

Декомпозиция процесса «Произвести вычет налога» представлена на рисунке 3.5.

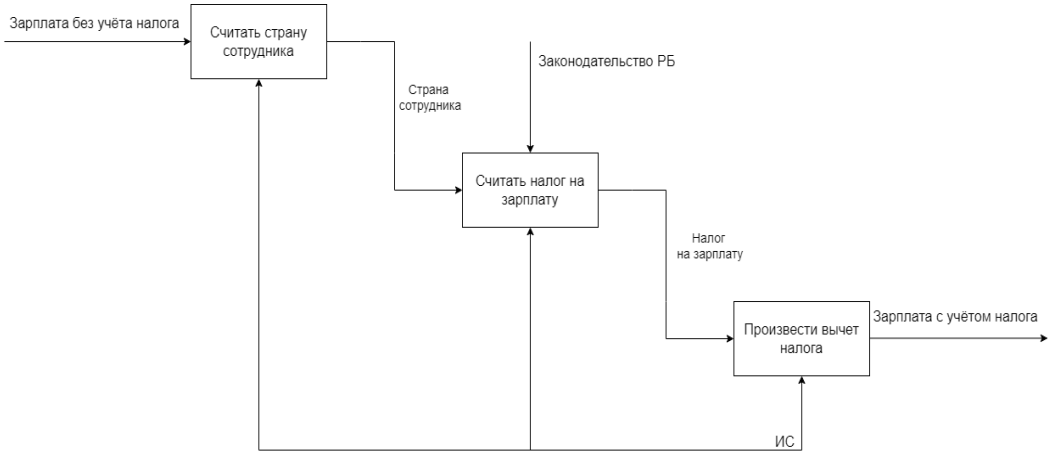

Рисунок 3.5 – Диаграмма декомпозиции процесса «Произвести вычет налога»

Декомпозиция процесса «Конвертировать в другие валюты» состоит из 3 функций:

– рассчитать множители других валют;
– получить зарплату в других валютах;
– вывести полученный результат.

Декомпозиция процесса «Конвертировать в другие валюты» представлена на рисунке 3.6.

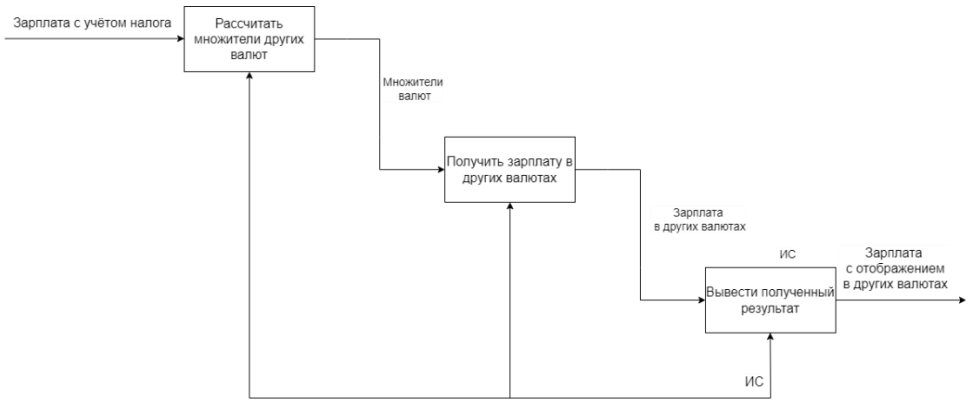

Рисунок 3.6 – Диаграмма декомпозиции процесса «Конвертировать в другие валюты»

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ

Информационная модель системы расчёта заработной платы содержит следующие сущности:

Сущность users (пользователи):

– идентификатор (первичный ключ);

– логин;

– пароль;

– роль.

В данной сущности хранятся все логины, пароли и роли пользователей.

Сущность userdepartments (пользователи-отделы):

– идентификатор (первичный ключ);

– идентификатор пользователя (внешний ключ);

– идентификатор отдела (внешний ключ).

В данной сущности хранятся все идентификаторы сущностей пользователи и отделы.

Сущность salaries (зарплаты):

– идентификатор (первичный ключ);

– название должности (внешний ключ);

– базовая ставка;

– премия;

– количество отработанных часов;

– оплата за час работы;

– дата зачисления зарплаты;

– итоговая сумма зарплаты.

В данной сущности хранятся все параметры расчёта зарплаты и название должности с этой зарплатой.

Сущность positions (должности):

– название должности (первичный ключ);

– обязанности;

– требования.

В данной сущности хранятся все должности, из обязанности и требования.

Сущность penalties (штрафы):

– идентификатор (первичный ключ);

– Логин (внешний ключ);

– сумма штрафа;

– дата получения штрафа;

– причина штрафа.

В данной сущности хранятся все штрафы сотрудников, их сумма и причина получения, а также логины пользователей со штрафами.

Сущность messages (сообщения):

– идентификатор (первичный ключ);

– отправитель;

– получатель (внешний ключ);

– текст сообщения.

В данной сущности хранятся все сообщения сотрудников, текст сообщения, а также имя отправителя и логин получателя.

Сущность employees (сотрудники):

– идентификатор (первичный ключ);

– фамилия;

– имя;

– отчество;

– номер телефона;

– зарплата;

– дата найма;

– логин;

– название отдела;

– должность.

В данной сущности хранится вся информация о сотрудниках компании.

Сущность employeepositions (сотрудники-должности):

– идентификатор (первичный ключ);

– название должности (первичный ключ) (внешний ключ).

В данной сущности хранятся все идентификаторы сотрудников и названия должностей.

Сущность departments (отделы):

– идентификатор (первичный ключ);

– название отдела;

– местоположение отдела.

В данной сущности хранятся все отделы, а также их местоположение.

Связи между таблицами выглядят следующим образом:

– пользователи и пользователи-отделы – связь 1: M;

– пользователи и сотрудники – связь 1:1;

– сотрудники и сотрудники-должности – связь 1: M;

– должности и сотрудники-должности – связь 1: M;

– должности и зарплаты – связь 1: M;

– пользователи и штрафы – связь 1: M;

– пользователи и сообщения – связь 1: M;

– сотрудники и зарплаты – связь 1: M.

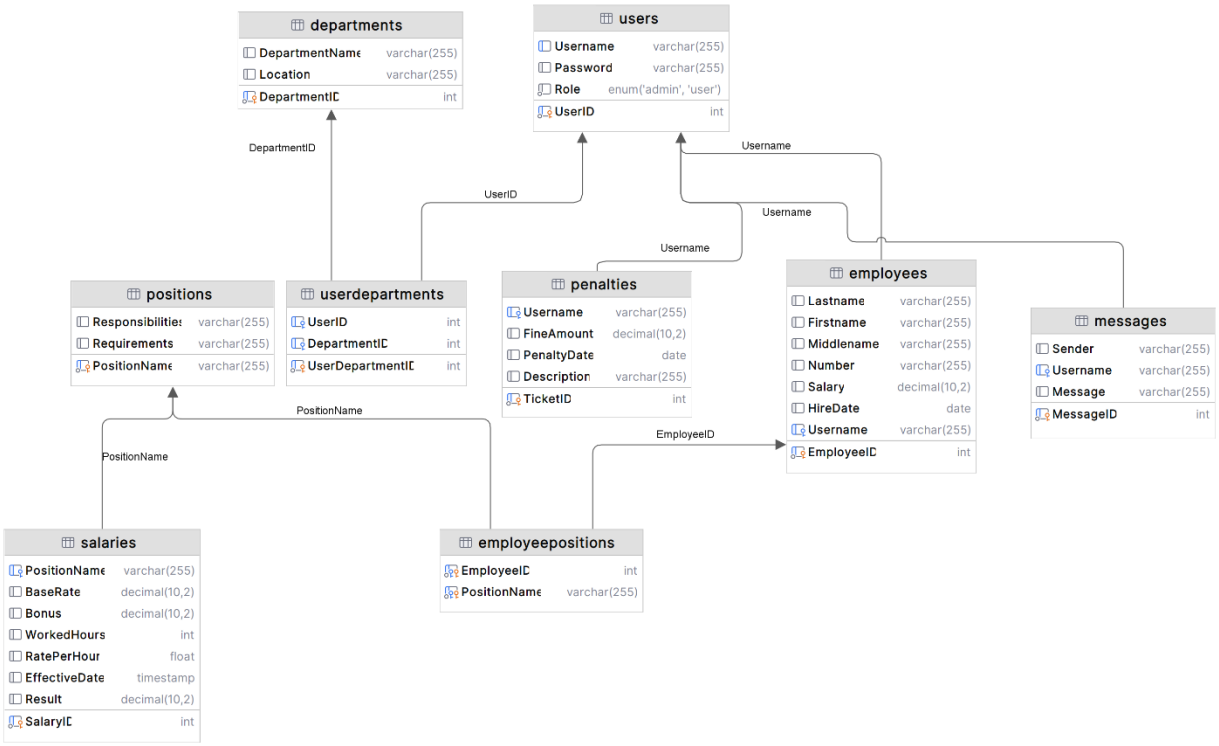

Рисунок 4.1 – Схема базы данных

Данная информационная модель находиться в третьей нормальной форме по причине того, что колонки в таблицах зависят лишь от первичного ключа и не зависят друг от друга, все таблицы имеют ключ, все ячейки хранят лишь одно значение, все данные в каждой колонке одного типа, записи отличаются друг от друга.

5 ОБОСНОВАНИЕ ОРИГИНАЛЬНЫХ РЕШЕНИЙ ПО ИСПОЛЬЗОВАНИЮ ТЕХНИЧЕСКИХ И ПРОГРАММНЫХ СРЕДСТВ

В системе по учёту заработной платы было использовано средство HeidiSQL для работы с базой данных совместно с MySQL. MySQL является одной из наиболее популярных и широко используемых реляционных баз данных, обеспечивающая стабильность и надежность хранения данных. HeidiSQL предоставляет удобный графический интерфейс для управления базой данных MySQL. Это позволяет легко создавать, изменять и администрировать базу данных, упрощая процесс разработки.

Выбор IDE Intellij IDEA обусловлен тем, что она предоставляет богатый набор инструментов для Java-разработки, включая умный рефакторинг кода, автоматическое завершение кода и мощную систему отладки. Это снижает вероятность ошибок и повышает эффективность разработчика. IntelliJ IDEA интегрирована с популярными фреймворками, что упрощает разработку и интеграцию. В данном случае, возможность легкой интеграции с JavaFX обеспечивает удобное создание графического интерфейса.

Выбор JavaFX для графического интерфейса обусловлен тем, что JavaFX обеспечивает возможность создания кроссплатформенных приложений с богатым графическим интерфейсом. Это важно для обеспечения доступа к системе на различных операционных системах. Поскольку JavaFX является частью Java Development Kit (JDK), его использование обеспечивает естественную интеграцию с языком программирования Java, что упрощает разработку и поддержку кода.

Выбранные технологии обеспечивают легкость масштабирования системы в случае необходимости добавления новых функций или изменения требований. Выбор данных технологий позволяет эффективно использовать ресурсы, обеспечивая баланс между производительностью, функциональностью и удобством разработки.

Таким образом, использование IntelliJ IDEA, MySQL с HeidiSQL и JavaFX обосновано исходя из их технических характеристик, удобства разработки и поддержки, что обеспечивает эффективное создание системы расчёта заработной платы.

6 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

На рисунке 6.1 представлена блок-схема листинга кода функции Main на сервере.

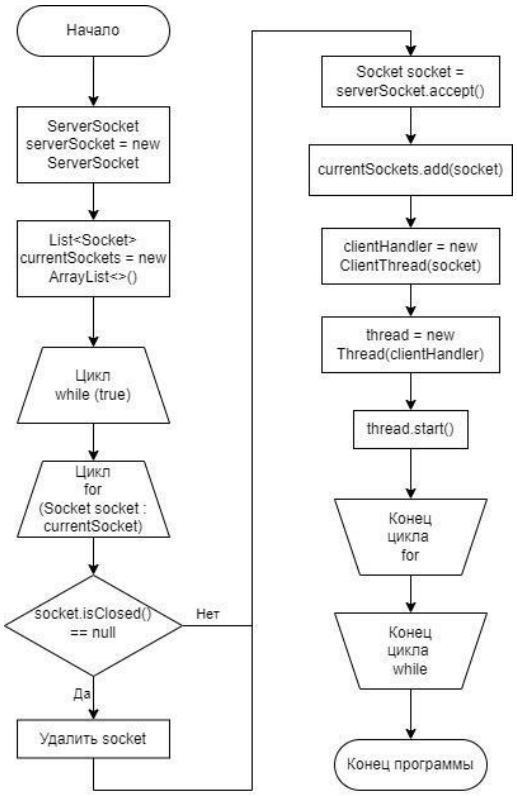

Рисунок 6.1 – Блок-схема функции Main у сервера

При запуске сервера, программа входит в бесконечный цикл, при котором ожидает подключения клиента, после успешного подключения клиента создается отдельный поток, где запускается основная логика программы.

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

7.1 Регистрация

При открытии программы пользователю будет выведено на экран окно, представленное на рисунке 7.1.

Рисунок 7.1 – Окно авторизации

В данном окне пользователь либо вводит свои данные для входа в систему, либо нажимает на кнопку «Зарегистрироваться», кроме этого, пользователь может нажать на восклицательный знак и откроется окно помощи. При нажатии кнопки «Зарегистрироваться» отобразиться новое окно, представленное на рисунке 7.2.

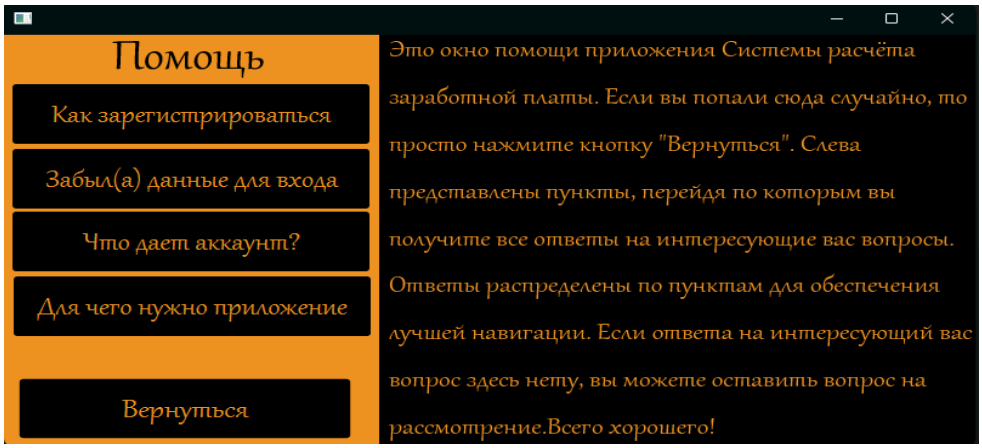

Рисунок 7.2 – Окно помощи

Рисунок 7.3 – Окно регистрации

В данном окне пользователю необходимо ввести свои данные. При вводе всех данных корректно, появляется диалоговое окно, информирующее об успешной регистрации, представленное на рисунке 7.4.

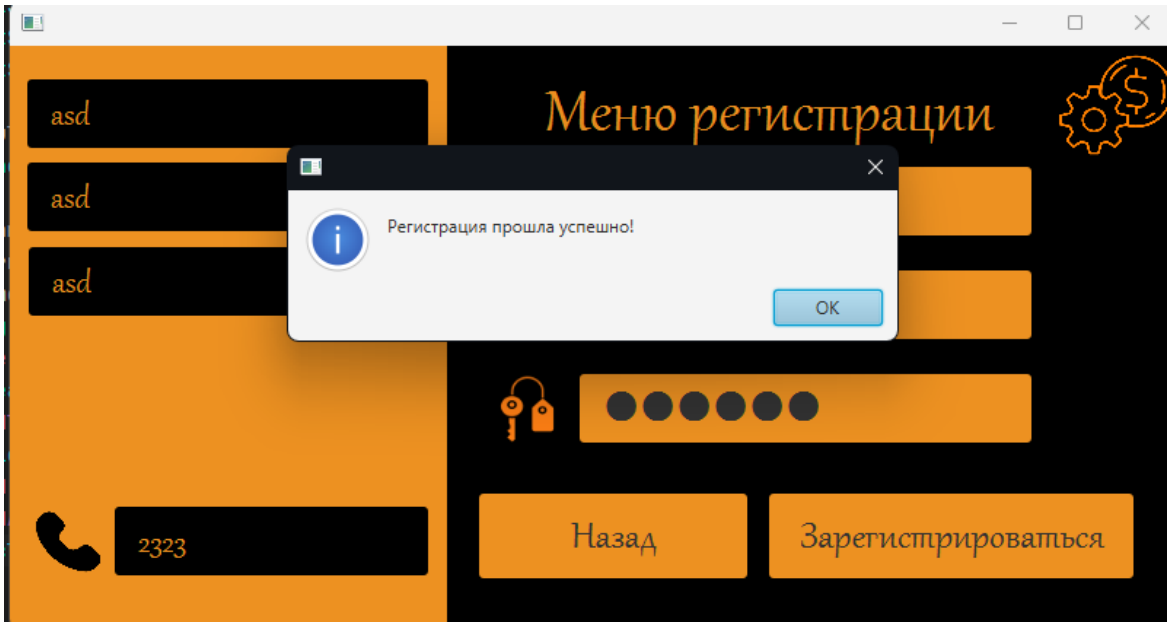

Рисунок 7.4 – Окно успешной регистрации

После регистрации пользователю необходимо войти в аккаунт, введя только что введенные данные.

7.2 Модуль обычного пользователя

При входе за обычного пользователя, ему будут предложены услуги, представленные на рисунке 7.5.

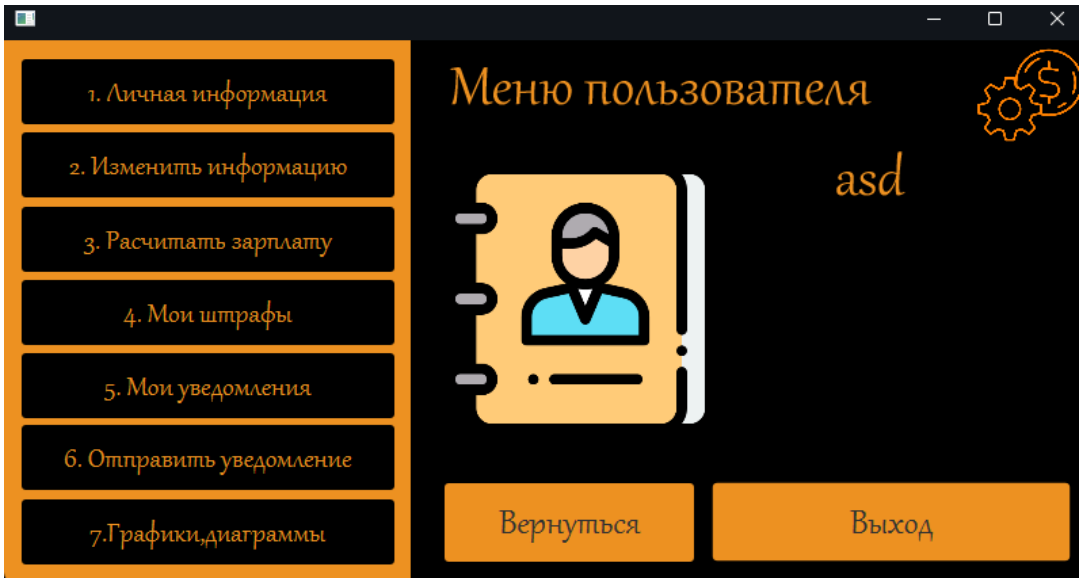

Рисунок 7.5 – Окно обычного пользователя

Окно просмотра личной информации представлено на рисунке 7.6.

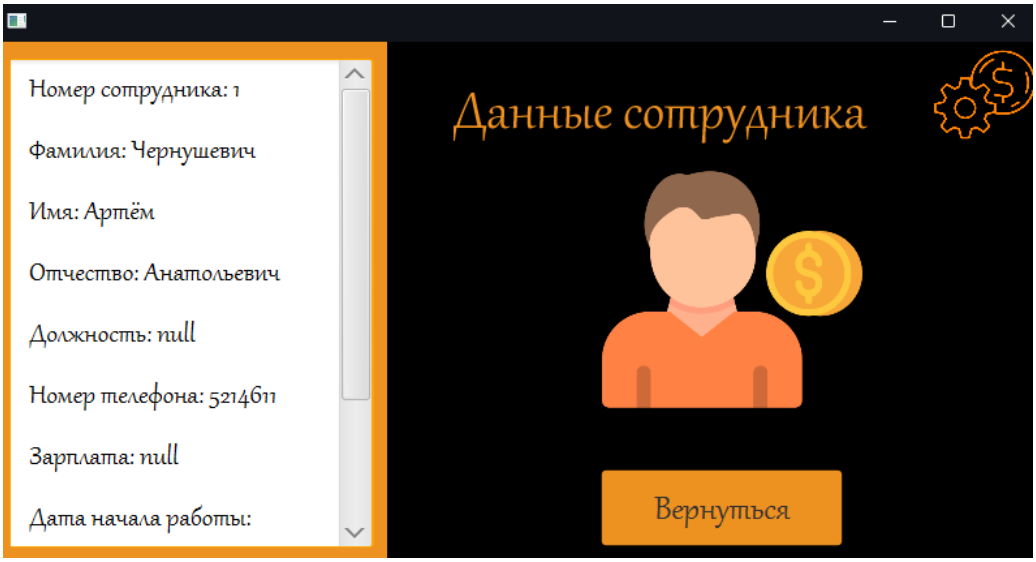

Рисунок 7.6 – Окно просмотра личной информации

Окно изменения личной информации представлено на рисунке 7.7.

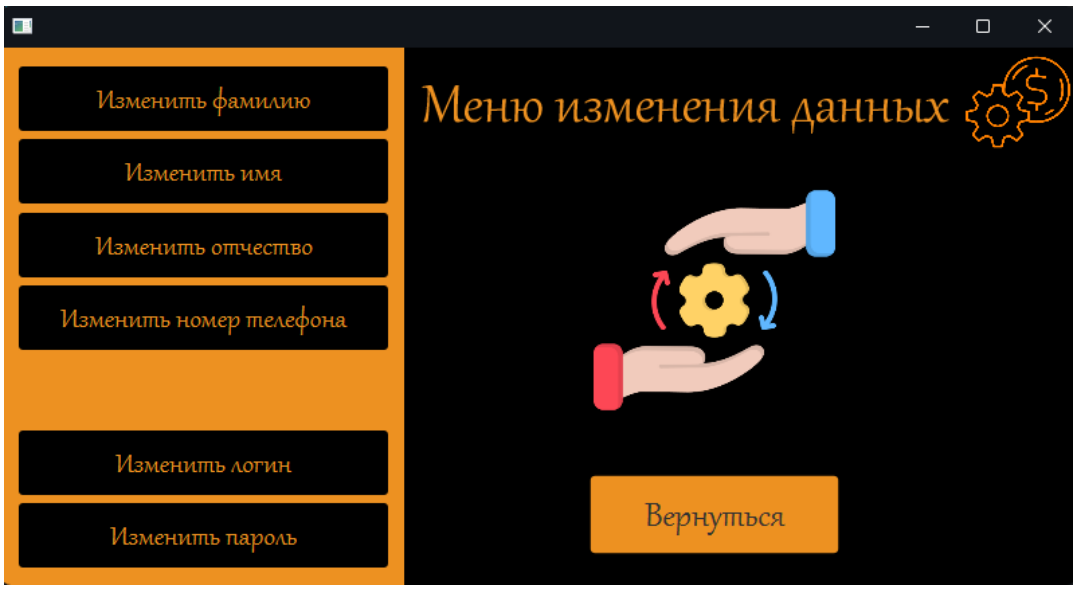

Рисунок 7.7 – Окно изменения личной информации

Здесь можно изменить такие данные как фамилия, имя, отчество, номер телефона логин или пароль, причем при изменении логина или пароля пользователю нужно будет заново авторизироваться.

При нажатии на кнопку «Рассчитать зарплату» открывается окно расчёта заработной платы, которое представлено на рисунке 7.8.

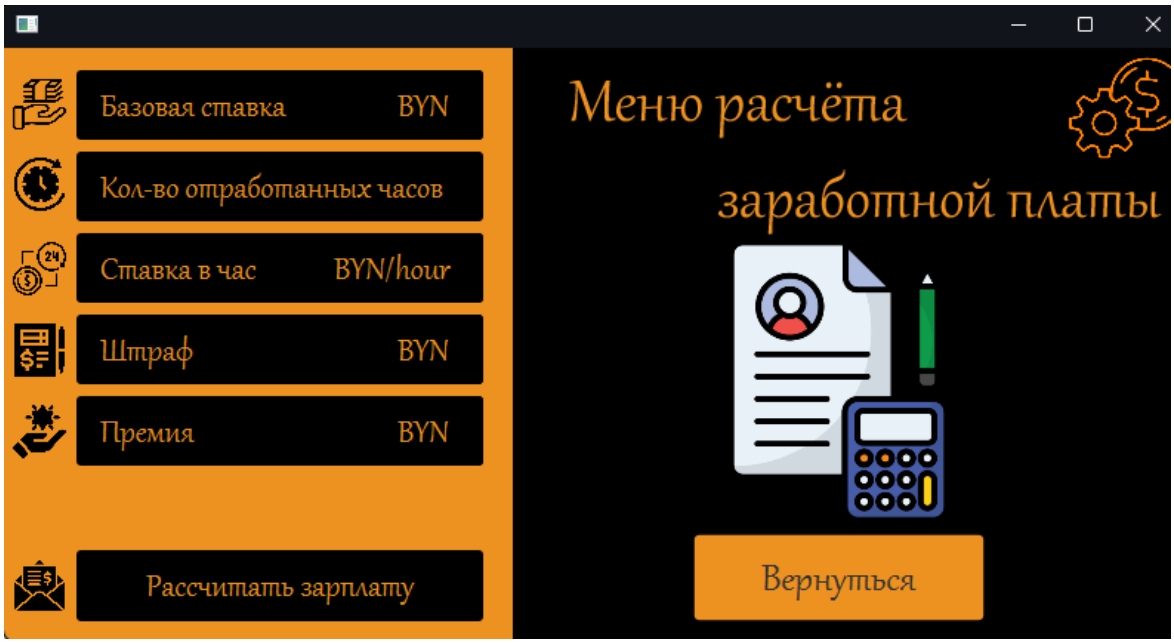

Рисунок 7.8 – Окно расчёта заработной платы

При нажатии на кнопку «Мои штрафы» открывается окно со всеми штрафами пользователя, где находится список, всех актуальных штрафов, с суммой, датой получения и причиной получения штрафа, которое представлено на рисунке 7.9.

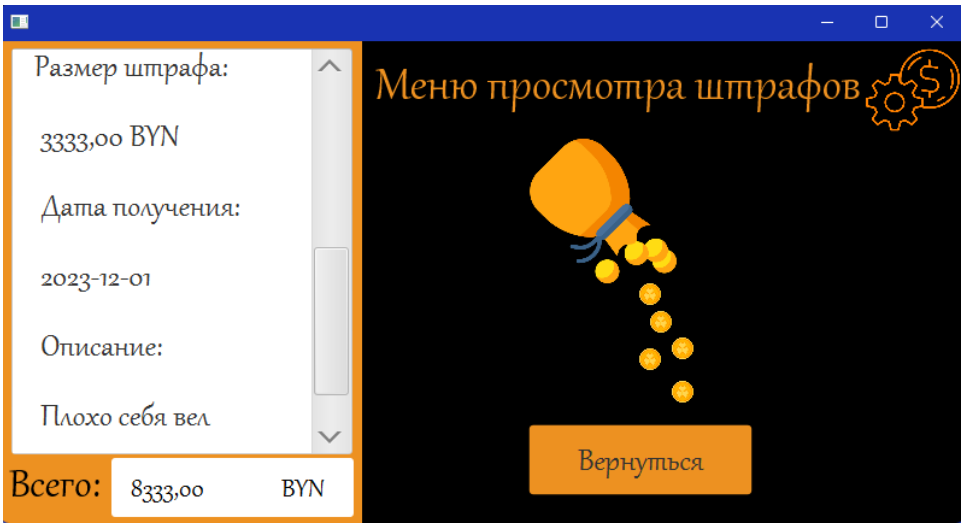

Рисунок 7.9 – Окно просмотра штрафов

При нажатии на кнопку «Мои уведомления» открывается окно, где можно либо прочитать все уведомления, которые принадлежат этому пользователю, либо очистить уведомления, также можно отправить уведомлению любому пользователю, с учётом того, что вы знаете его логин, имя отправителя же можно указать любое. Окно представлено на рисунке 7.10.

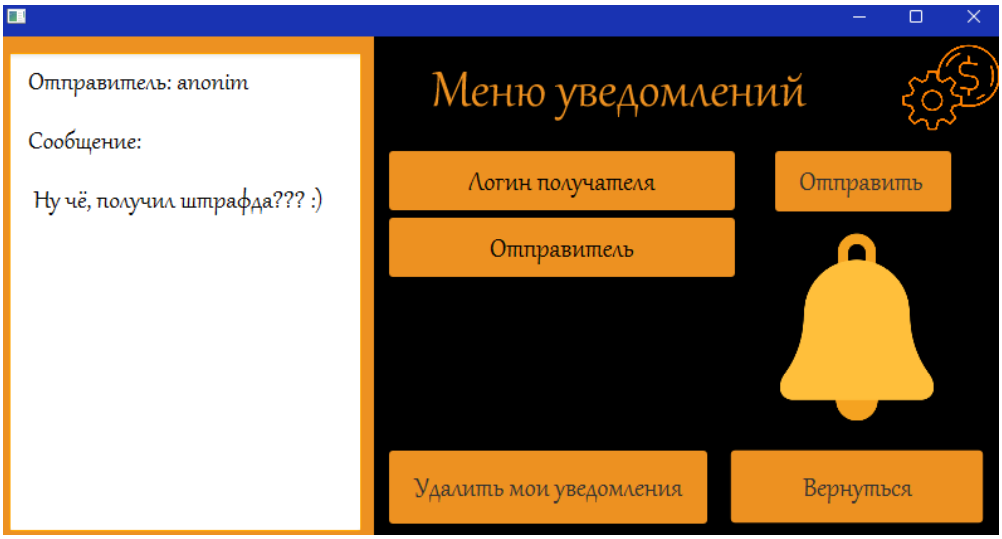

Рисунок 7.10 – Окно просмотра, отправки либо удаления сообщений

При нажатии на кнопку с диаграммой открывается меню диаграмм конкретного пользователя, изображенной на рисунке 7.11.

Рисунок 7.11 – Окно диаграммы зарплаты пользователя

7.3 Модуль администратора

После ввода данных, которые принадлежат администратору открывается его меню, представленное на рисунке 7.12.

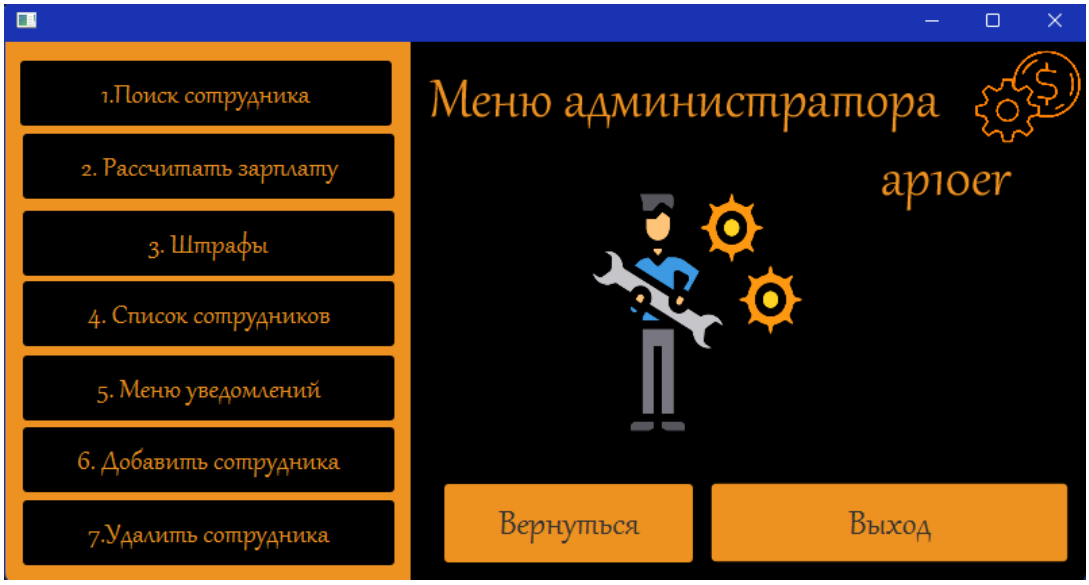

Рисунок 7.12 – Окно администратора

При нажатии на кнопку «Поиск сотрудника» появляется диалоговое окно, в которое нужно ввести полное сотрудника. Диалоговое окно представлено на рисунке 7.13.

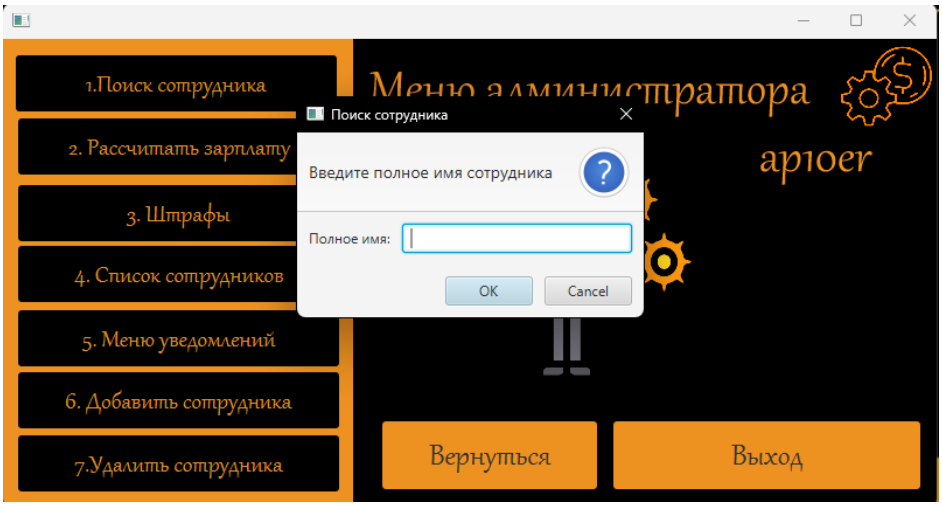

Рисунок 7.13 – Окно ввода данных для поиска сотрудника

При нажатии на кнопку «Рассчитать зарплату» открывается окно расчёта зарплаты, однако, в отличие от окна обычного пользователя, здесь расширен функционал, появлсь возможность добавления зарплаты в базу данных, окно представленно на рисунке 7.14.

Рисунок 7.14 – Окно расчёта заработной платы

При нажатии на кнопку «Меню штрафов» открывается окно, где можно либо оштрафовать сотрудника, либо просмотреть все актуальные штрафы, лиибо анулировать штраф. Окно представлено на рисунке 7.15.

Рисунок 7.15 – Окно меню штрафов

При нажатии на кнопку «Список сотрудников» открывается окно, в котором находится список всех сотрудников компании, представленный на рисунке 7.16.

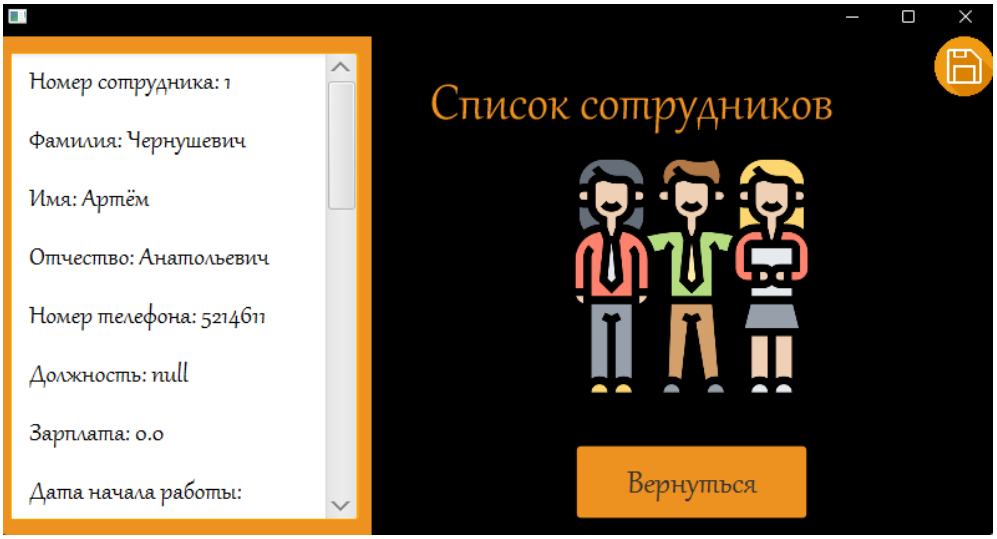

Рисунок 7.16 – Окно добавления записи в медицинскую карту пациенту

При нажатии на кнопку «Меню уведомлений» открывается меню, в котором можно отправить, просмотреть либо удалить уведомления любого пользователя, окно изображено на рисунке 7.17.

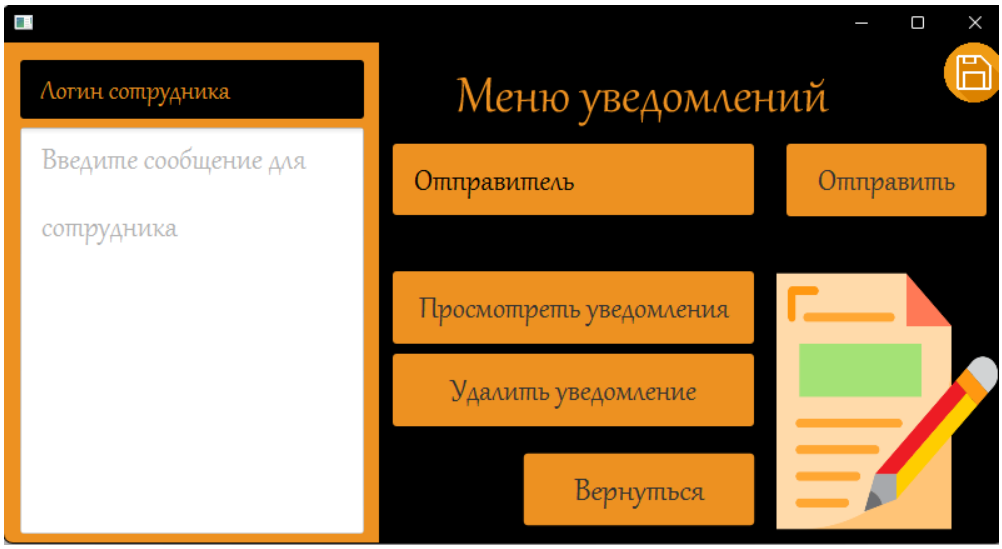

Рисунок 7.17 – Окно просмотра, отправления, удаления уведомлений

При нажатии на кнопку «Добавить сотрудника» открывается окно добавления сотрудника, которое изображено на рисунке 7.18.

Рисунок 7.18 – Окно добавления сотрудника

При нажатии на кнопку «Удалить сотрудника» появляется окно для ввода логина сотрудника, если такой логин есть в базе данных, то все данные

пользователя удалятся. Окно удаления пользователя изображено на рисунке 7.19.

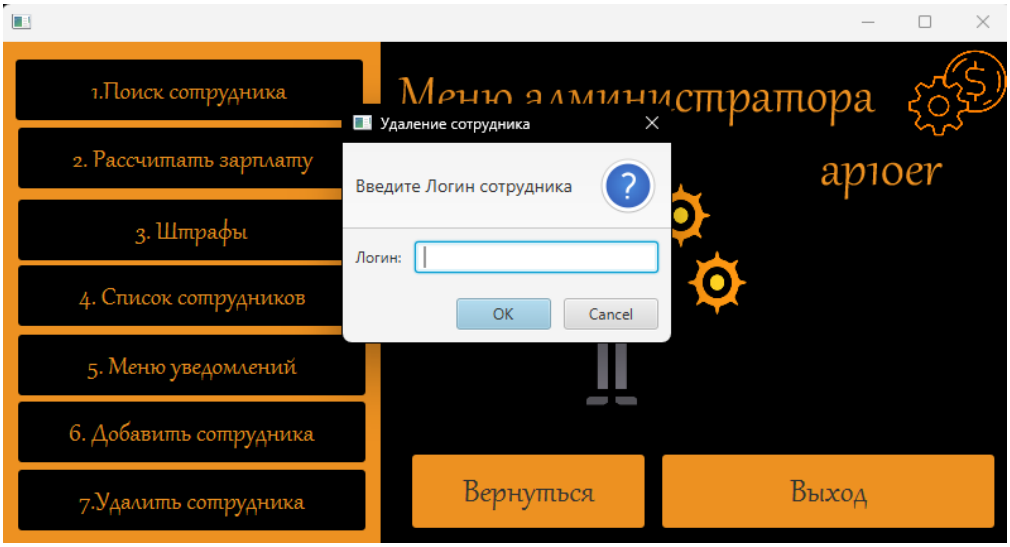

Рисунок 7.19 – Окно удаления сотрудника

Кроме этого, в некоторых окнах справа вверху есть иконка дискеты, при нажатии на которую из базы данных считается вся актуальная информация выбранной таблицы и запишется в файл.

8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

Далее будут приведены примеры проверок на неправильный ввод данных, проверки на возможность совершения данного действия.

При регистрации, добавлении или редактировании данных пользователь вводит множество данных, и неправильный ввод может быть как специальным, так и абсолютно случайным.

Если клиент ввел данные для входа неверно или оставил определенные поля пустыми, ему высветится сообщение об этом, представленное на рисунке 8.1.

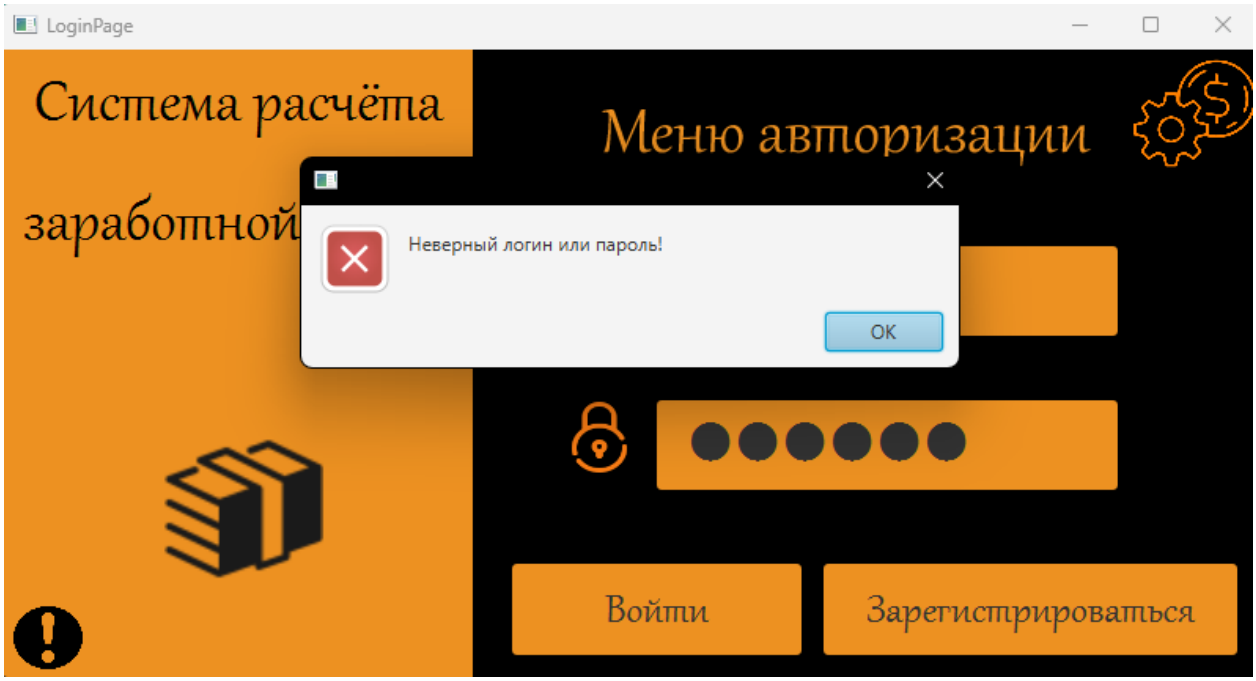

Рисунок 8.1 – Предупреждение о несуществующем аккаунте

Также если пользователь введет пароль, состоящий из менее 4 символов, программа выдаст ошибку, показанную на рисунке 8.2.

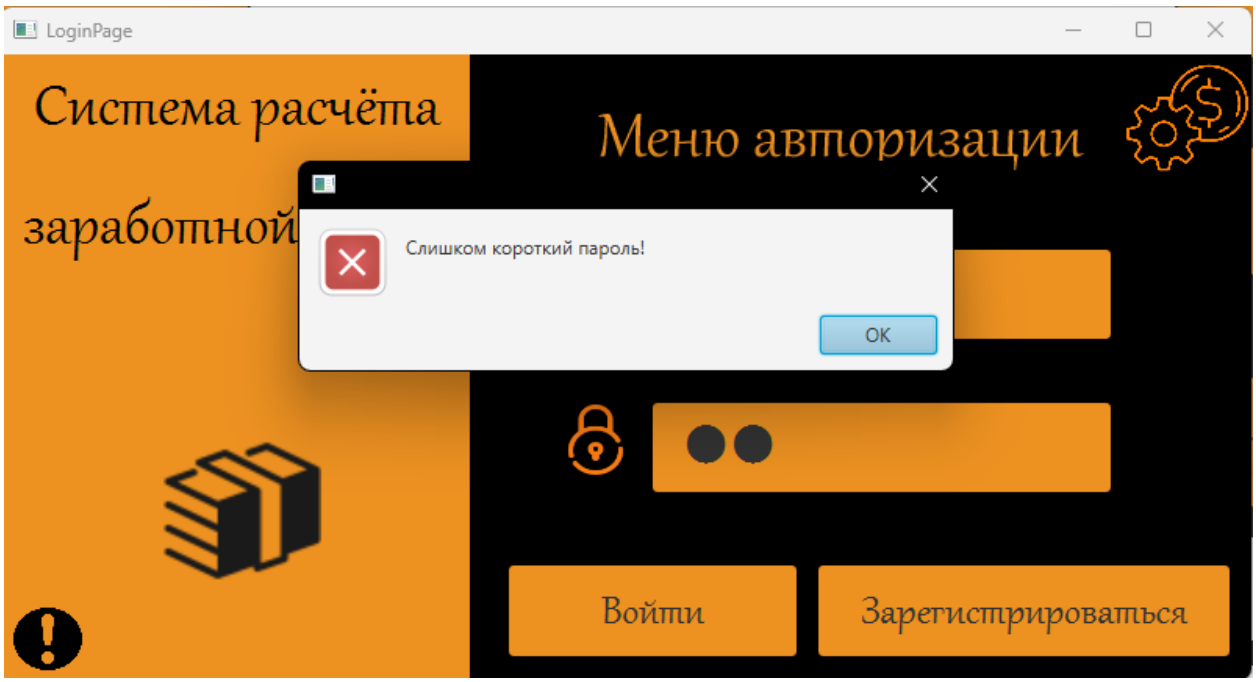

Рисунок 8.2 – Сообщение о слишком коротком пароле

Если при расчёте заработной платы в поля, где предусмотрены цифры вводить не числовые значения, программа выдаст ошибку, изображенную на рисунке 8.3.

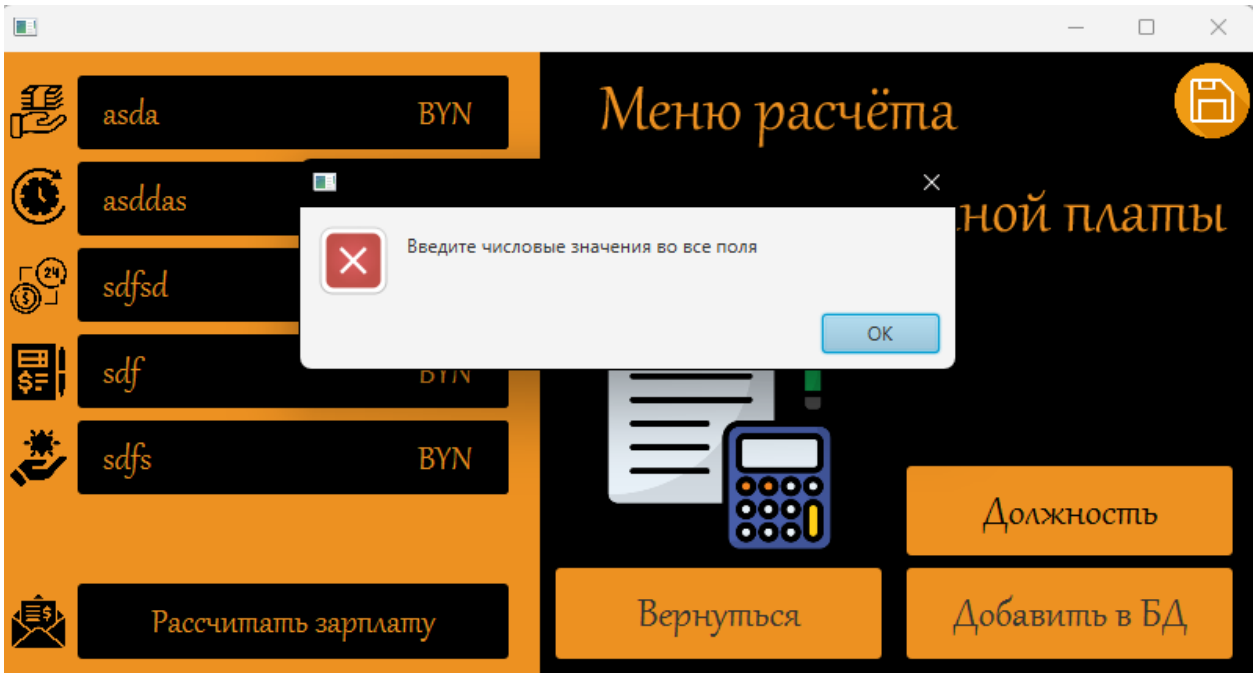

Рисунок 8.3 – Окно о неверном типе ввода

При попытке оштрафовать несуществующего пользователя, появится диалоговое окно с ошибкой, показанное на рисунке 8.4.

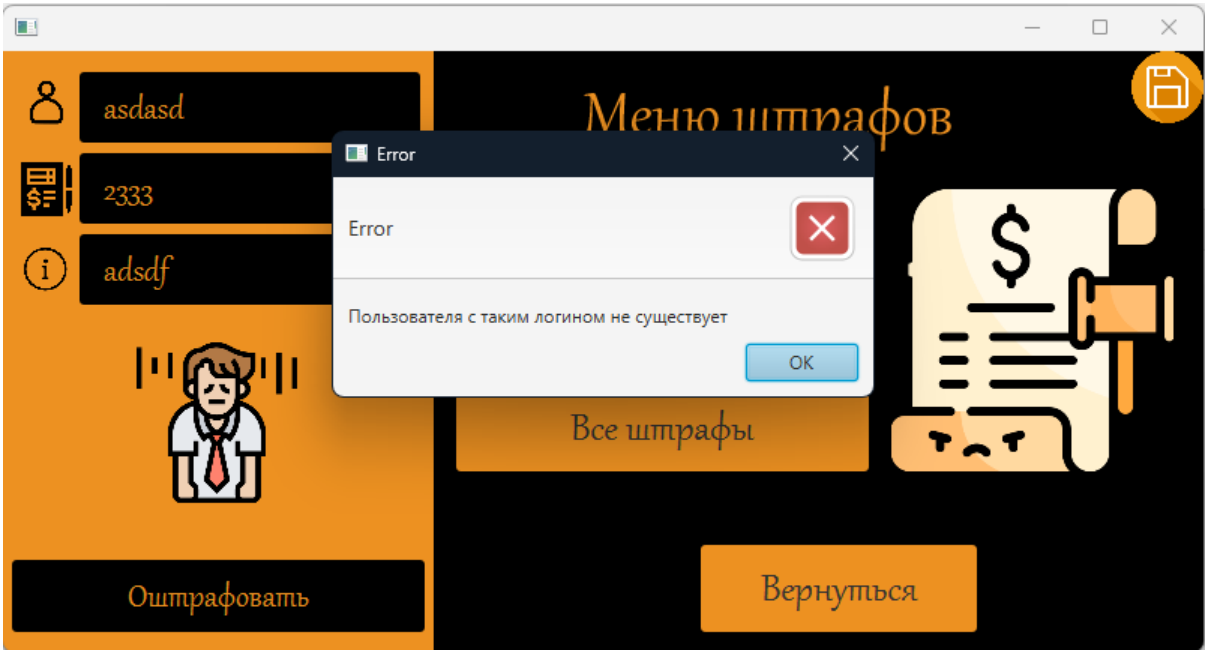

Рисунок 8.4 – Сообщение о удалении аккаунта третьим лицом

При попытке удалить либо просмотреть штрафы пользователя, у которого их нету, появится диалоговое окно с ошибкой, показанное на рисунке 8.5.

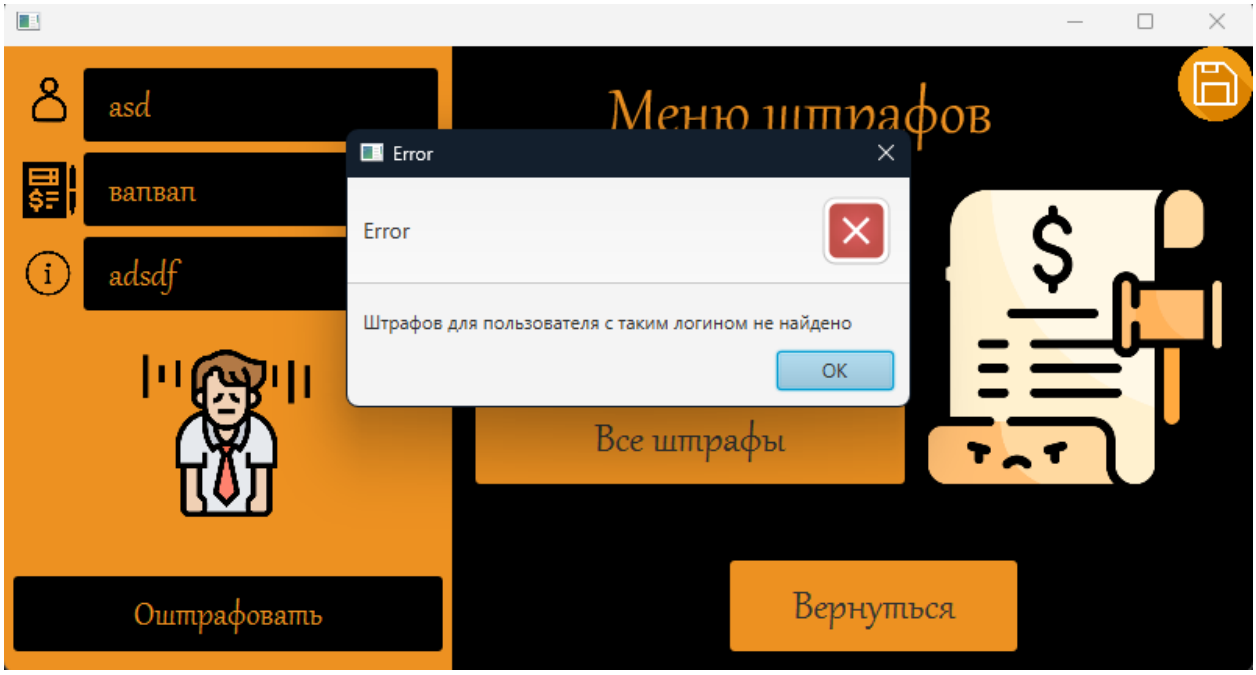

Рисунок 8.5 – Сообщение об отсутствии штрафов у пользователя

При попытке отправить сообщение несуществующему пользователю программа выдаст ошибку, изображенную на рисунке 8.6.

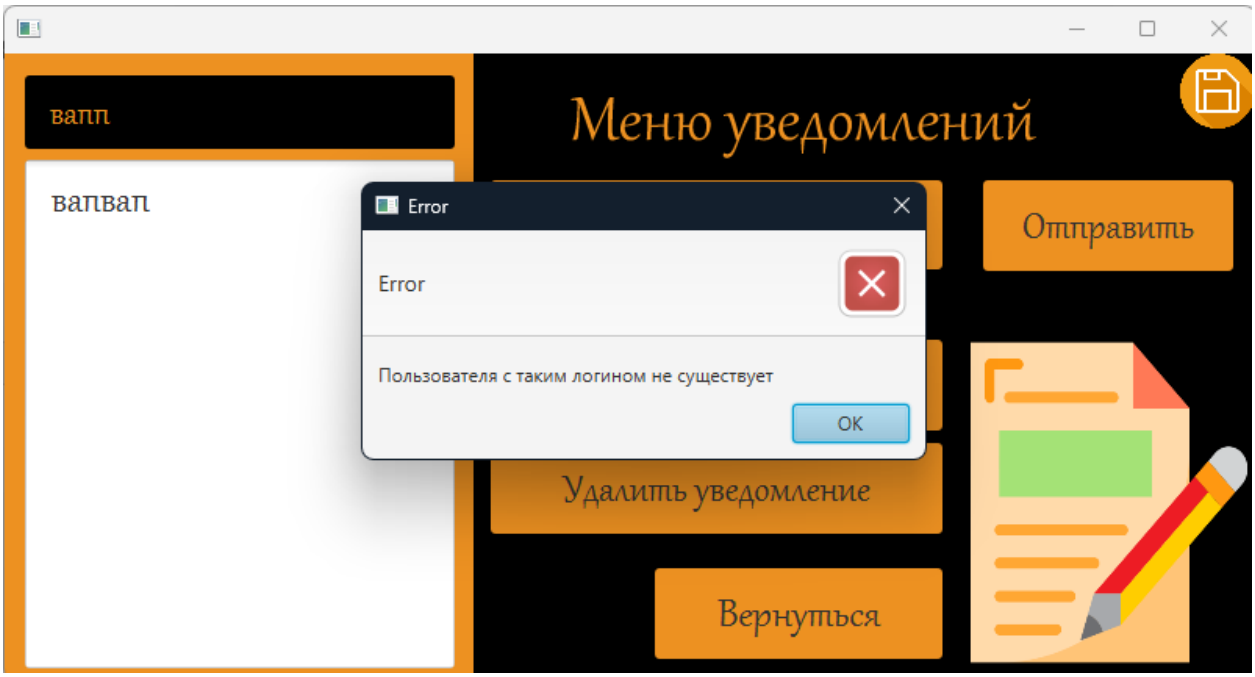

Рисунок 8.6 – Отправка сообщения несуществующему пользователю

При попытке удалить сотрудника, которого нету в базе данных программа выдаст ошибку, изображенную на рисунке 8.7.

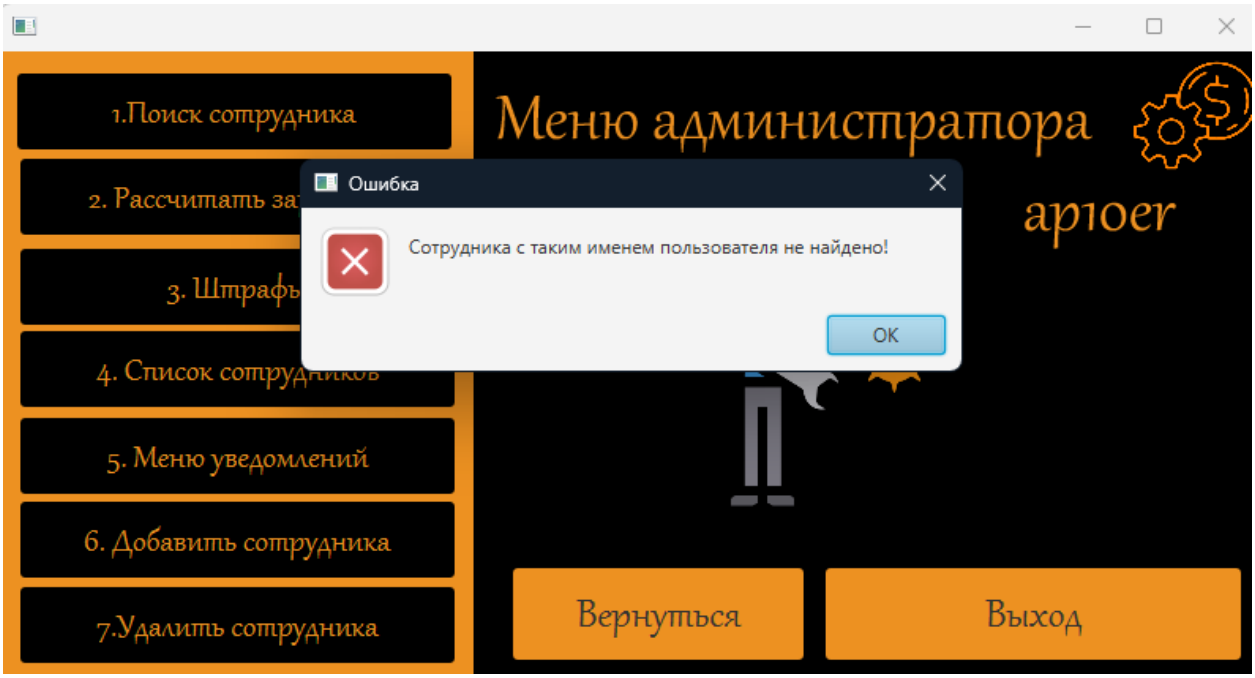

Рисунок 8.7 – Сотрудника с введенном именем не существует

ЗАКЛЮЧЕНИЕ

В процессе разработки системы расчета заработной платы, которая должна соответствовать предъявленным требованиям, были применены передовые методы и технологии с целью обеспечения высокой эффективности, надежности и удобства использования. В ходе выполнения проекта активно использовались основные приемы программирования, такие как инкапсуляция, перегрузка методов, переопределение методов, сериализация, абстрактные типы данных, статические методы и обработка исключительных ситуаций. Этот подход способствовал формированию четкой структуры кода, повышению его читаемости и облегчению поддержки.

В ходе разработки применялись различные техники моделирования в соответствии со стандартами IDEFO, IDEFIX и UML 2.0. Это позволило провести функциональное и информационное моделирование, создав ясное представление о процессах и структуре системы.

Основной целью проекта было создание современной, надежной и удобной в использовании системы управления заработной платой. В результате успешно были решены различные задачи, включая разработку клиент-серверной архитектуры, создание многопоточного сервера, проектирование эффективной базы данных, реализацию безопасной бизнес-логики и интеграцию удобного графического интерфейса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] ГОСТ 19.701-90. Схемы алгоритмов, программ и систем.

[2] Маклаков, С. В. Моделирование бизнес-процессов с AIIFusion Process Modeler. – М.: Диалог-МИФИ, 2008. – 90 с.

[3] Bloch, Joshua. Effective Java. [Электронный ресурс]. – Режим доступа: https://www.oreilly.com/library/view/effective-java-3rd/9780134686097/.

[4] itteach.ru [Электронный ресурс] – Режим доступа: https://itteach.ru.

[5] MySQL 8.0 Reference Manual [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/.

[6] JFreeChart Developer Guide [Электронный ресурс]. – Режим доступа: https://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html.

[7] Беляев Д.А. «Функции служб управления экономическими процессами в вузах» [Электронный ресурс]. – Режим доступа: https://www.cfin.ru/management/practice/university_management.shtml.

[8] Доманов, А.Т. Стандарт предприятия СТП 01-2017 / А. Т. Доманов, Н. И. Сорока. – Минск: БГУИР, 2017. – 169 с.

[9] jetbrains.com [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/help/idea/getting-started.html.

[10] metanit.com [Электронный ресурс] – Режим доступа: https://metanit.com/java/javafx/3.1.php.

[11] javarush.com [Электронный ресурс] – Режим доступа: https://javarush.com/.

[12] JavaFX [Электронный ресурс] – Режим доступа: https://controlsfx.github.io/.

ПРИЛОЖЕНИЕ А
(обязательное)

Проверка в системе «Антиплагиат»

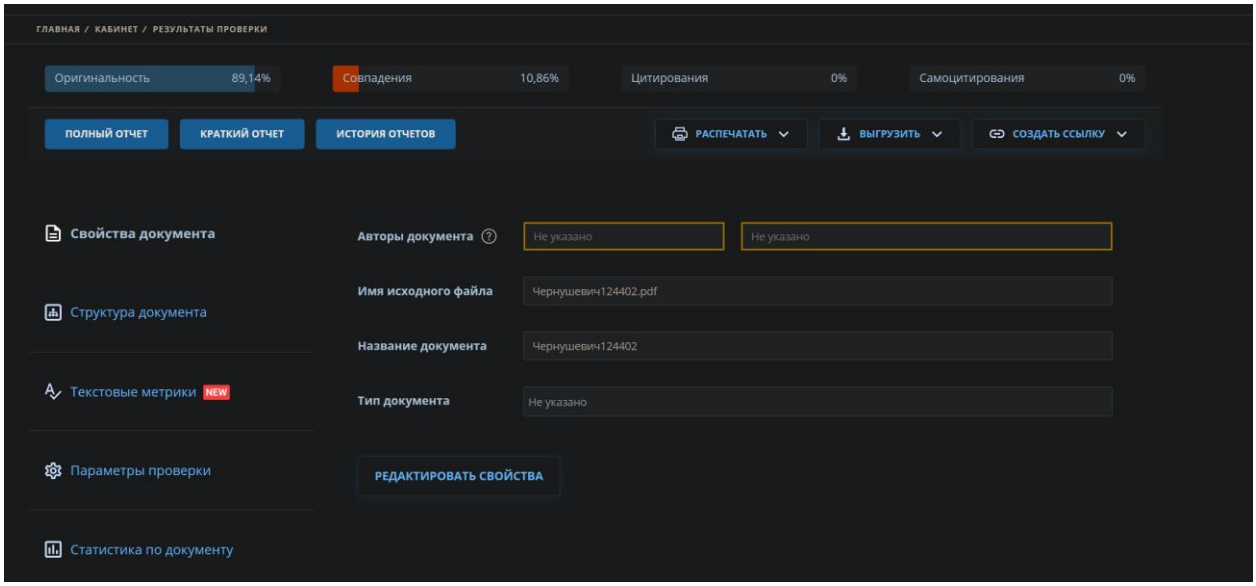

Рисунок А.1 – Оригинальность курсового проекта

ПРИЛОЖЕНИЕ Б
(обязательное)

Схемы алгоритмов

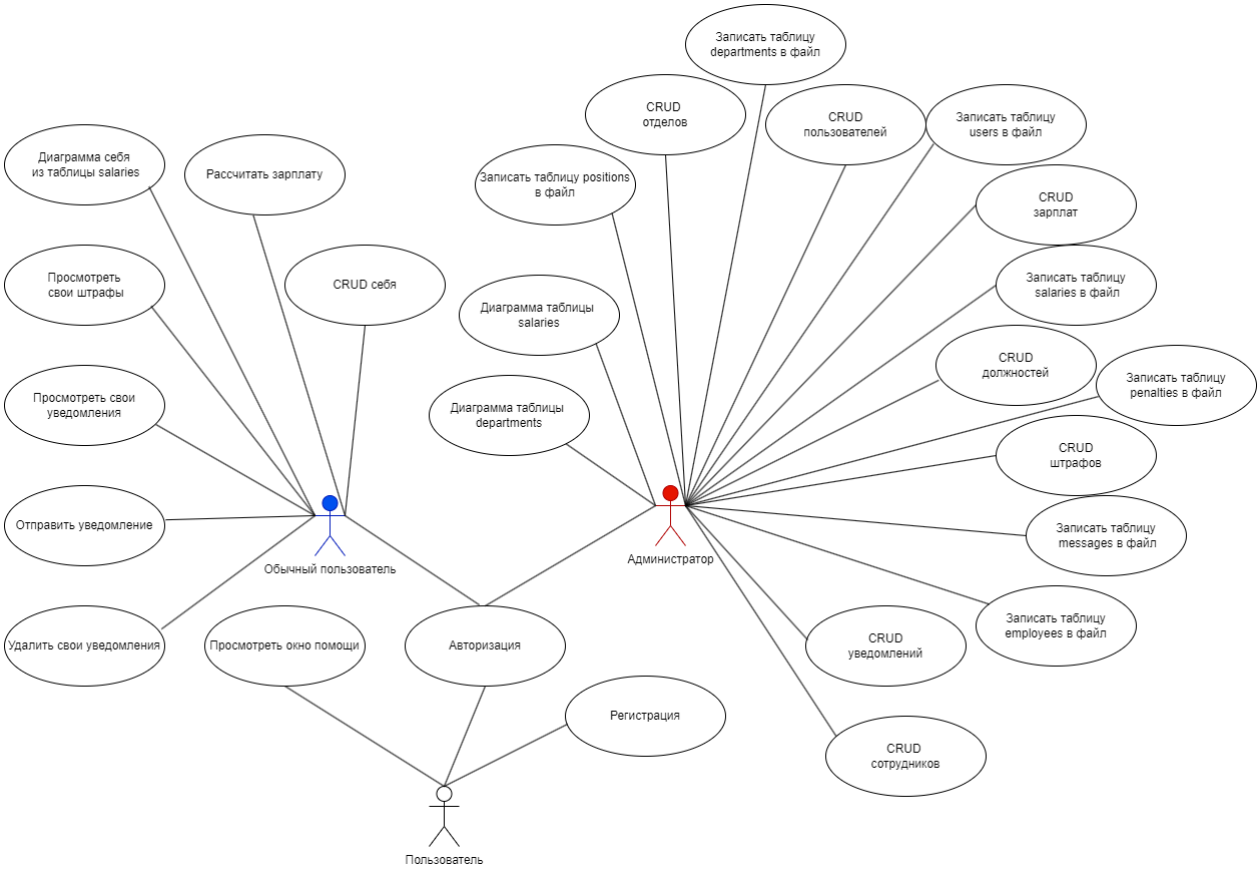

Рисунок Б.1 – Диаграмма вариантов использования

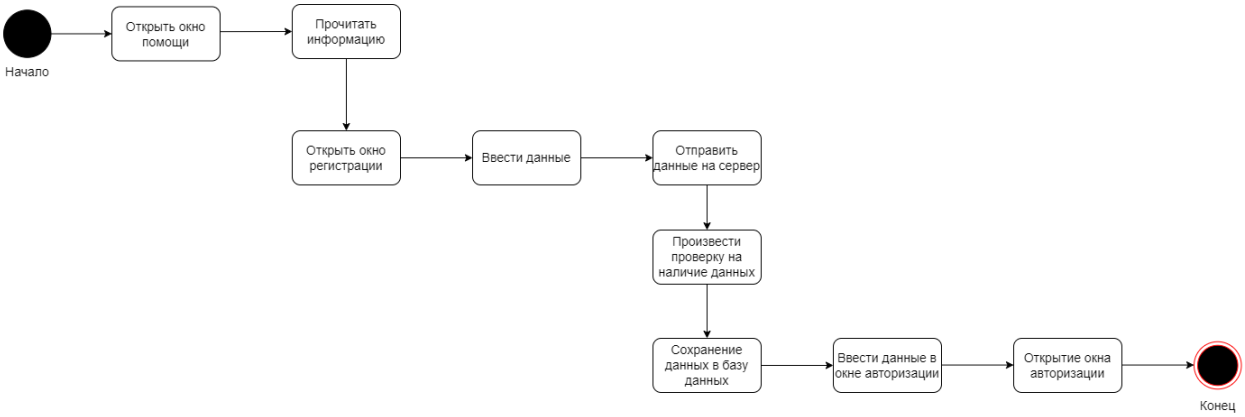

Рисунок Б.2 – Диаграмма состояний

Рисунок Б.3 – Диаграмма состояний

Рисунок Б.4 – Диаграмма последовательности

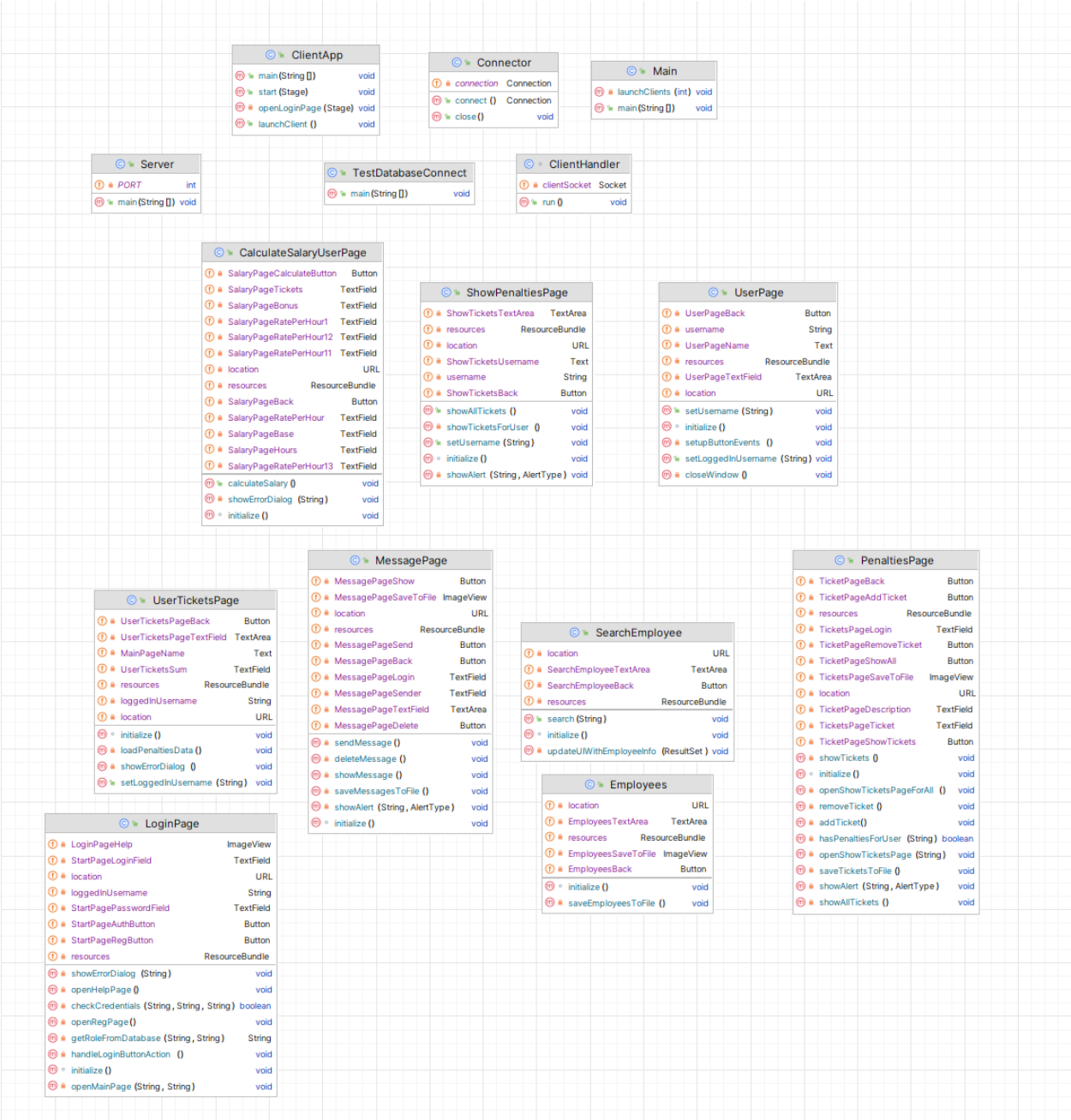

Рисунок Б.5.1 – Диаграмма классов

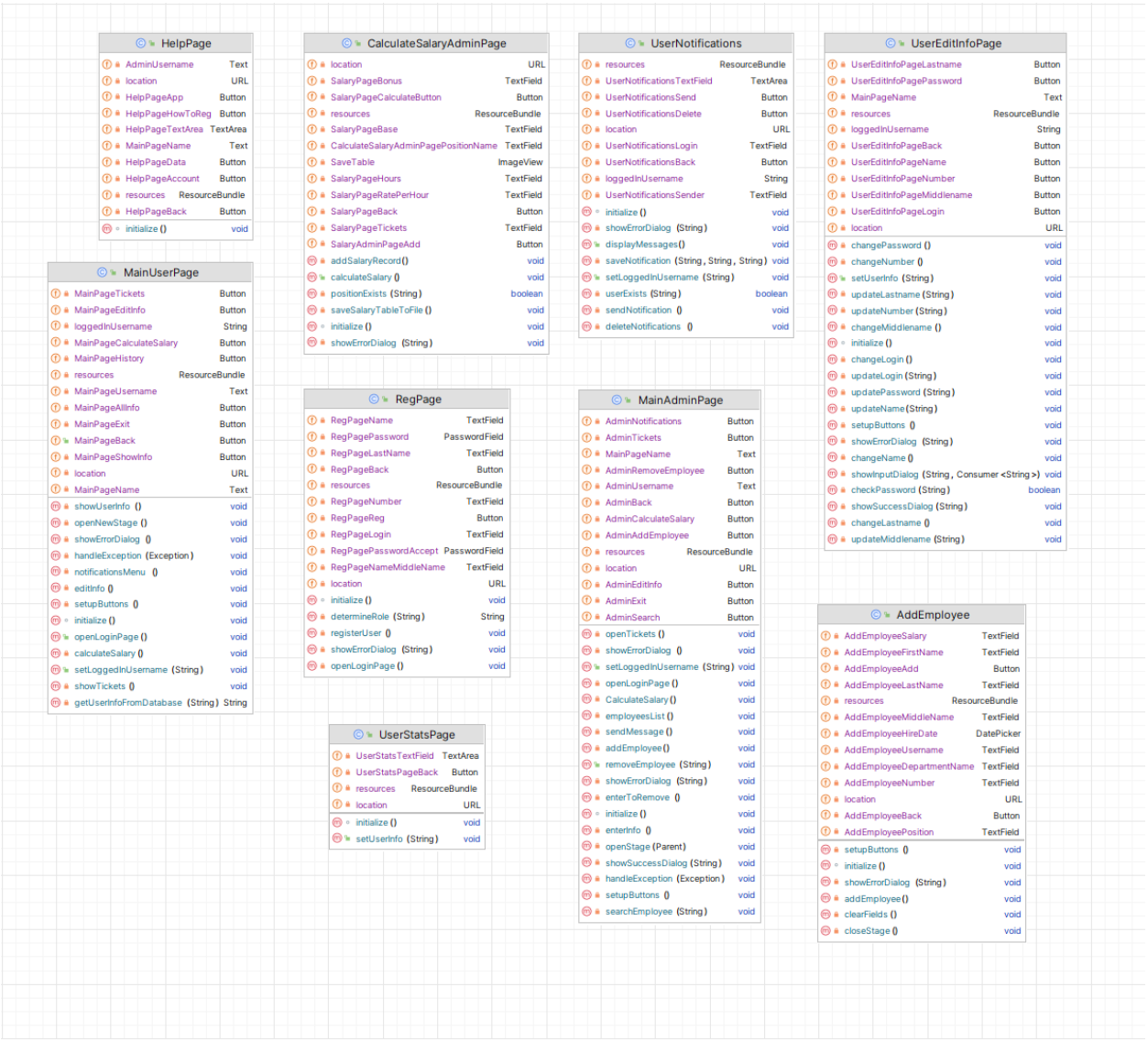

Рисунок Б.5.2 – Продолжение диаграммы классов

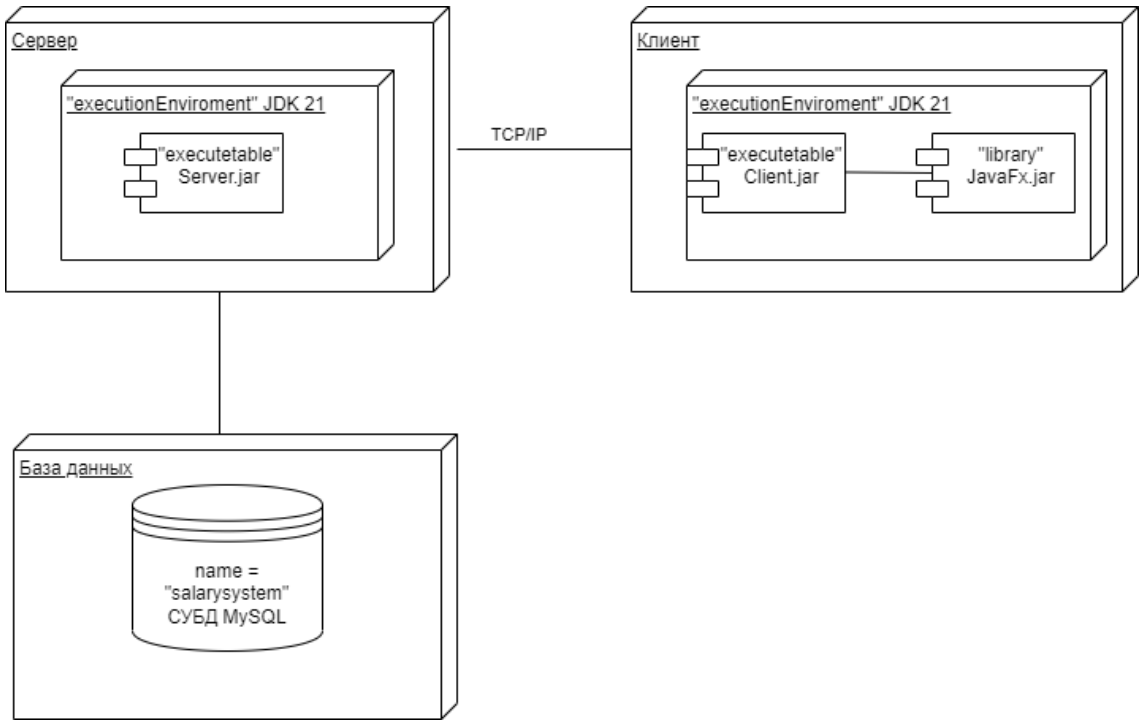

Рисунок Б.6 – Диаграмма развертывания

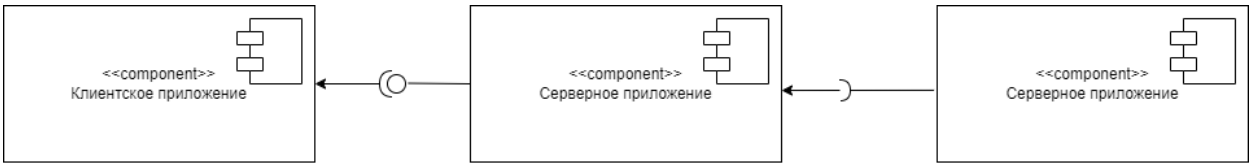

Рисунок Б.7 – Диаграмма компонентов

ПРИЛОЖЕНИЕ В
(обязательное)

Листинг скрипта генерации базы данных

```
-- Создание базы данных
CREATE DATABASE IF NOT EXISTS SalarySystem;
USE SalarySystem;

-- Создание таблицы "Users"
CREATE TABLE IF NOT EXISTS Users (
    UserID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(255) UNIQUE,
    Password VARCHAR(255),
    Role ENUM('Admin', 'User') NOT NULL
);

INSERT INTO Users (Username, Password, Role) VALUES ('ap10er', 'asdasd', 'Admin');

-- Создание таблицы "Departments"
CREATE TABLE IF NOT EXISTS Departments (
    DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
    DepartmentName VARCHAR(255),
    Location VARCHAR(255)
);

-- Создание таблицы "UserDepartments" для хранения связей между пользователями и
департаментами
CREATE TABLE IF NOT EXISTS UserDepartments (
    UserDepartmentID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    DepartmentID INT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID) ON UPDATE CASCADE,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID) ON UPDATE CASCADE
);

-- Создание таблицы "Employees"
CREATE TABLE IF NOT EXISTS Employees (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    Lastname VARCHAR(255),
    Firstname VARCHAR(255),
    Middlename VARCHAR(255),
    Number VARCHAR(255),
    Salary DECIMAL(10,2),
    HireDate DATE,
    Username VARCHAR(255),
    DepartmentName VARCHAR(255),
    Position VARCHAR(255),
    FOREIGN KEY (Username) REFERENCES Users(Username) ON UPDATE CASCADE
);

INSERT INTO Employees (Lastname, Firstname, Middlename, Number, HireDate, Username)
VALUES ('Чернушевич', 'Артём', 'Анатольевич', '5214611', '2023-11-23', 'ap10er');

-- Создание таблицы "Departments"
CREATE TABLE IF NOT EXISTS Departments (
    DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
    DepartmentName VARCHAR(255),
```

```
Location VARCHAR(255)
);

-- Создание таблицы "Positions"
CREATE TABLE IF NOT EXISTS Positions (
    PositionName VARCHAR(255) PRIMARY KEY,
    Responsibilities VARCHAR(255),
    Requirements VARCHAR(255)
);

-- Создание таблицы "EmployeePositions"
CREATE TABLE IF NOT EXISTS EmployeePositions (
    EmployeeID INT,
    PositionName VARCHAR(255),
    PRIMARY KEY (EmployeeID, PositionName),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID) ON UPDATE CASCADE,
    FOREIGN KEY (PositionName) REFERENCES Positions(PositionName) ON UPDATE CASCADE
);

-- Создание таблицы "Salaries"
CREATE TABLE IF NOT EXISTS Salaries (
    SalaryID INT PRIMARY KEY AUTO_INCREMENT,
    PositionName VARCHAR(255),
    BaseRate DECIMAL(10,2),
    Bonus DECIMAL(10,2),
    WorkedHours INT,
    RatePerHour FLOAT,
    EffectiveDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Result DECIMAL(10,2),
    FOREIGN KEY (PositionName) REFERENCES Positions(PositionName) ON UPDATE CASCADE
);

-- Создание таблицы "Penalties"
CREATE TABLE IF NOT EXISTS Penalties (
    TicketID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(255),
    FineAmount DECIMAL(10,2),
    PenaltyDate DATE,
    Description VARCHAR(255),
    FOREIGN KEY (Username) REFERENCES Users(Username) ON UPDATE CASCADE
);

-- Создание таблицы "Messages"
CREATE TABLE IF NOT EXISTS Messages (
    MessageID INT PRIMARY KEY AUTO_INCREMENT,
    Sender VARCHAR(255),
    Username VARCHAR(255),
    Message VARCHAR(255),
    FOREIGN KEY (Username) REFERENCES Users(Username) ON UPDATE CASCADE
);
```

ПРИЛОЖЕНИЕ Г
(обязательное)

Листинг кода

```
package client;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class ClientApp extends Application {

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        openLoginPage(primaryStage);
    }

    private void openLoginPage(Stage primaryStage) {
        try {
            FXMLLoader loader = new FXML-
Loader(getClass().getResource("/Pages/LoginPage.fxml"));
            Parent root = loader.load();

            Scene scene = new Scene(root);
            primaryStage.setScene(scene);
            primaryStage.setTitle("LoginPage");
            primaryStage.show();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void launchClient() {
        launch();
    }
}

package Database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Connector {

    private static Connection connection = null;

    public static Connection connect() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
```

```
System.out.println("MySQL JDBC Driver Registered!");

} catch (ClassNotFoundException e) {

 System.out.println("Where is your MySQL JDBC Driver?");
            e.printStackTrace();
            throw new SQLException("MySQL JDBC Driver not found.");
        }

        connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/salary-
system?useUnicode=true&serverTimezone=UTC", "root", "13979");

        if (connection == null) {
            throw new SQLException("Failed to establish connection to the data-
base.");
        } else {
            System.out.println("Successfully connected!");
        }

        return connection;
    }

    public static void close() {
        try {
            if (connection != null) {
                connection.close();
                System.out.println("Closing connection!\n\n");
            }
        } catch (SQLException e) {
            System.out.println("Failed to close connection!\n\n");
            e.printStackTrace();
        }
    }
}

package Database;

import java.sql.Connection;
import java.sql.SQLException;

public class TestDatabaseConnect {

    public static void main(String[] argv) {
        System.out.println("-------- MySQL JDBC Connection Testing ------------");

        try {
            Connection connection = Connector.connect();

        } catch (SQLException e) {
            System.out.println("Connection Failed!");
            e.printStackTrace();
        } finally {
            Connector.close();
        }
    }
}

package Pages;
```

```
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ResourceBundle;
import Database.Connector;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class AddEmployee {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button AddEmployeeBack;

    @FXML
    private Button AddEmployeeAdd;

    @FXML
    private TextField AddEmployeeLastName;

    @FXML
    private TextField AddEmployeeFirstName;

    @FXML
    private TextField AddEmployeeMiddleName;

    @FXML
    private TextField AddEmployeeNumber;

    @FXML
    private TextField AddEmployeePosition;

    @FXML
    private TextField AddEmployeeSalary;

    @FXML
    private DatePicker AddEmployeeHireDate;

    @FXML
    private TextField AddEmployeeUsername;

    @FXML
    private TextField AddEmployeeDepartmentName;

    @FXML
    void initialize() {
        setupButtons();
```

```
    }

    private void setupButtons() {
        AddEmployeeBack.setOnAction(actionEvent -> closeStage());
        AddEmployeeAdd.setOnAction(actionEvent -> {
            try {
                addEmployee();
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        });
    }

    private void closeStage() {
        Stage stage = (Stage) AddEmployeeBack.getScene().getWindow();
        stage.close();
    }

    private void addEmployee() throws SQLException {
        String lastName = AddEmployeeLastName.getText();
        String firstName = AddEmployeeFirstName.getText();
        String middleName = AddEmployeeMiddleName.getText();
        String number = AddEmployeeNumber.getText();
        String position = AddEmployeePosition.getText();
        String salary = AddEmployeeSalary.getText();
        String hireDate = AddEmployeeHireDate.getValue().toString();
        String departmentName = AddEmployeeDepartmentName.getText();

        try {
            Connection connection = Connector.connect();
            String insertQuery = "INSERT INTO Employees (Lastname, Firstname, Middle-
name, Number, Position, Salary, HireDate, DepartmentName) " +
                    "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
            try (PreparedStatement insertStatement = connection.prepareStatement(in-
sertQuery)) {
                insertStatement.setString(1, lastName);
                insertStatement.setString(2, firstName);
                insertStatement.setString(3, middleName);
                insertStatement.setString(4, number);
                insertStatement.setString(5, position);
                insertStatement.setString(6, salary);
                insertStatement.setString(7, hireDate);
                insertStatement.setString(8, departmentName);

                insertStatement.executeUpdate();
            }

            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("");
            alert.setHeaderText(null);
            alert.setContentText("Сотрудник успешно добавлен!");
            alert.showAndWait();

            clearFields();

        } catch (SQLException e) {
            e.printStackTrace();
```

```
            showErrorDialog("Ошибка при работе с базой данных.");

} finally {
            Connector.close();
        }
    }

    private void showErrorDialog(String message) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("");
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

    private void clearFields() {
        AddEmployeeLastName.clear();
        AddEmployeeFirstName.clear();
        AddEmployeeMiddleName.clear();
        AddEmployeeNumber.clear();
        AddEmployeePosition.clear();
        AddEmployeeSalary.clear();
        AddEmployeeHireDate.getEditor().clear();
        AddEmployeeDepartmentName.clear();
    }
}
```