Оглавление

[Введение 3](#_Toc90423869)

[1 ОИСАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc90423870)

[1.1 Сущность и функция ПРОГРАМНОЙ ПОДДЕРЖКИ УЧЕТА И АНАЛИЗА ДЕЯТЕЛЬНОСТИ ПЕРСОНАЛА НА ПРЕДПРИЯТИИ 5](#_Toc90423871)

[2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ 6](#_Toc90423872)

[2.1 Постановка задачи и обзор методов её решения 6](#_Toc90423873)

[2.1.1 Java 7](#_Toc90423874)

[2.1.2 JavaFX 8](#_Toc90423875)

[2.1.3 MySQL 9](#_Toc90423876)

[2.3 Модели представления системы и их описание 10](#_Toc90423877)

[2.2.1 Диаграмма классов 10](#_Toc90423878)

[2.2.2 Диаграмма последовательности 10](#_Toc90423879)

[2.2.3 Диаграмма состояния 10](#_Toc90423880)

[3 ОПИСАНИЕ ОСНОВНОГО ПРОЦЕССА ФУНКЦИОНИРОВАНИЯ РАБОТЫ СИСТЕМЫ УПРАВЛЕНИЯ НА ОСНОВЕ СТАНДАРТА IDEF0 12](#_Toc90423881)

[4 Построение информационной модели 13](#_Toc90423882)

[5 Спецификация вариантов использования системы 14](#_Toc90423883)

[6 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 16](#_Toc90423884)

[6.1 Алгоритм работы функции начисления заработной платы 16](#_Toc90423885)

[6.2 Алгоритм работы всей программы 17](#_Toc90423886)

[6.3 Алгоритм работы функции авторизации 17](#_Toc90423887)

[7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ. 18](#_Toc90423888)

[7.1 Пользовательское приложение 18](#_Toc90423889)

[7.2 Приложение администратора 25](#_Toc90423890)

[Заключение 41](#_Toc90423891)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 42](#_Toc90423892)

[ПРИЛОЖЕНИЕ A 43](#_Toc90423893)

[ПРИЛОЖЕНИЕ Б 45](#_Toc90423894)

[ПРИЛОЖЕНИЕ В 46](#_Toc90423895)

[ПРИЛОЖЕНИЕ Д 47](#_Toc90423896)

[ПРИЛОЖЕНИЕ Г 48](#_Toc90423897)

[ПРИЛОЖЕНИЕ Е 49](#_Toc90423898)

[**ПРИЛОЖЕНИЕ Ж** 54](#_Toc90423899)

[**ПРИЛОЖЕНИЕ З** 56](#_Toc90423900)

[**ПРИЛОЖЕНИЕ И** 66](#_Toc90423901)

[**ПРИЛОЖЕНИЕ К** 67](#_Toc90423902)

[**ПРИЛОЖЕНИЕ Л** 68](#_Toc90423903)

# Введение

В настоящее время во всех отраслях человеческой деятельности огромное внимание уделяется использованию информационных технологий для наиболее рационального и эффективного использования ресурсов, имеющихся в распоряжении человека.

Среди многообразия поисков путей развития рынка, средств производства, новых направлений деятельности коммерческо-посреднических организаций и предприятий вызывают значительный интерес научные исследования и практические новации. В течение последних лет бурно развиваются основанные на информатике новые технологии.

Актуальность темы заключается в том, что информационные системы (ИС) занимают в этих технологиях центральное положение.

В современных условиях эффективное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными, человеческими и другими ресурсами. Следовательно, повышение эффективности становится одним из направлений совершенствования деятельности в целом. Наиболее очевидным способом повышения эффективности протекания трудового процесса является его автоматизация. Бурное развитие информационных компьютерных технологий, совершенствование технической платформы и появление принципиально новых классов программных продуктов привело к изменению подходов к автоматизации управления предприятиями, автоматизации документооборота, автоматизации различных процессов предприятий.

Начисление амортизации основных средств обычно не представляет никаких методических трудностей, однако на практике обычно представляет собой однообразную, повторяющуюся из месяца в месяц операцию. В отдельных случаях управляющему предприятием могут потребоваться данные по сумме, которую он должен потратить на содержание сотрудников на выплату им заработных плат. Кроме того, необходимо контролировать сумму начисленной заработной платы, чтобы не допустить его излишнее начисление. Так, основным результатом автоматизации этого участка учета должно стать освобождение бухгалтера от однообразной работы. Программа учета и анализа персонала на предприятии должна также позволять формировать различные отчеты и вести аналитический учет в разрезе отдельных объектов основных средств.

Каждое производственная организация имеет в своем распоряжении основные и оборотные средства. Поэтому основной целью является создание базы данных, которая позволит структурировать те огромные объемы

информации, которые накапливаются на предприятиях, и тем самым существенно облегчить работу и доступ к данным практически всем звеньям, участвующим в управлении основными средствами предприятия.

Таким образом, целью данного курсового проекта является сокращение временных затрат управляющих предприятием лиц за счёт автоматизации учета и анализа необходимой информации о персонале: расчета заработных плат, учета дисциплинарных взысканий и премиальных поощрений сотрудникам представления расчётов в виде диаграмм.

Для достижения поставленной цели требуется решить следующие задачи:

* проанализировать заданную предметную область;
* рассмотреть физическую и логическую модель представления данных;
* создать базу данных;
* реализовать серверную часть, имеющую доступ к базе данных и выполняющую ряд необходимых действий, позволяющих достигнуть цель курсового проекта;
* реализовать клиентскую часть с дружественным интерфейсом, обеспечивающий максимально удобное взаимодействие с программой или вычислительной системой.
* Реализовать в виде круговых графиков информацию по каждому отделу.

# 1 ОИСАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

# 1.1 Сущность и функция ПРОГРАМНОЙ ПОДДЕРЖКИ УЧЕТА И АНАЛИЗА ДЕЯТЕЛЬНОСТИ ПЕРСОНАЛА НА ПРЕДПРИЯТИИ

В данной курсовой работе необходимо разработать информационную систему для учета и анализа персонала на предприятии. Главной задачей при разработке данной информационной системы является быстрота нахождения информации, просмотра списка сотрудников, учета их деятельности путем учета отработанных каждым сотрудником часов и учета их дисциплинарных взысканий и поощрений." Программная поддержка учета и анализа деятельности персонала на предприятии ". ИС должна быть проста в использовании и предельно понятна для сотрудников. Клиент не может вносить изменения в структуру таблиц и схему данных.

 " Программная поддержка учета и анализа деятельности персонала на предприятии" должна обеспечивать выполнение следующих основных функций:

* Ведение списка сотрудников на предприятии, учета их деятельности как каждого по отдельности, так и по определенным отделам предприятия;
* Вывод информации о сотрудниках;
* Вывод информации о деятельности сотрудников, наличие дисциплинарных взысканий, премиальных поощрений, количестве отработанных часов каждым сотрудником (или отделом в целом);
* Расчет заработной платы для каждого сотрудника в отдельности в зависимости от наличия у сотрудника дисциплинарного взыскания или поощрения и количестве отработанных сотрудником часов.
* Вывод информации о заработной плате для сотрудников.
* Вывод информации по отделам о количестве дисциплинарных взысканий количестве сотрудников и количестве отработанных часов в круговых диаграммах

# 2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ

# 2.1 Постановка задачи и обзор методов её решения

Задача данного курсового проекта – это создание функционирующего приложения в архитектуре клиент-сервер с организацией взаимодействия с базой данных на объектно-ориентированном языке Java. Серверное приложение может быть реализовано в виде консольного приложения или GUI-приложения. Клиентское приложение: оконное приложение с использованием стандартных библиотек пользовательского интерфейса (JavaFX).

Для успешной реализации системы необходимо, в первую очередь, выделить основные задачи, которые будет решать система «Программной поддержки учета и анализа персонала», а также те задачи, которые необходимо выполнить для правильной работы системы.

Основными задачами являются следующие:

1. Разработка базы данных для MySQL 5.5+.
2. Разработка и использование собственной иерархии классов, расширение базовых классов, предоставляемых JDK.
3. Реализация не менее 2-х паттернов проектирования.
4. Использовать сокрытие данных (инкапсуляция), перегрузку методов, переопределение методов, социализацию, абстрактные типы данных (интерфейсы, абстрактные классы), статические методы, обработку исключительных ситуаций.
5. В разрабатываемом приложении обеспечить добавление, редактирование и удаление записей из базы данных, сохранение табличных результатов в файле (создание текстового отчета), предоставление пользователю аналитической информации (графики, диаграммы).
6. Предусмотреть возможность сохранять информацию в любой момент на сервере и загружать ранее сохраненные данные.
7. В консольном окне серверной части должна отображаться текущая статистика подключений и параметры запуска сервера.
8. Предусмотреть механизм авторизации пользователей (роли Администратор и Пользователь).
9. Система должна предоставлять удобный интерфейс для конечного пользователя.

Рассмотрим некоторые функции, реализующие бизнес-логику:

* Регистрация пользователей.

При регистрации пользователю предоставляется окно регистрации, где необходимо указать имя, фамилию, электронную почту, пол, и местоположение. создать логин и пароль. Данные отправляются на сервер и сохраняются в БД.

* Авторизация.

В данной программе существует два клиентских приложения приложение пользователя и приложение администратора. Пользователь вводит свой логин и пароль. Если введённый логин есть в базе данных, то дальше проверяем, совпадает ли пароль, введённый пользователем с паролем в базе данных. Если логин в БД не найдена или введён не верный пароль, то пользователю будет выведено сообщение об ошибке. Иначе пользователю будет открыто окно с его интерфейсом для дальнейшей работы и в зависимости от роли пользователя, у него будут определённые привилегии.

* Просмотр информации.

Клинт посылает сообщения серверу о получения данные о заработной плате, после чего на сервере формируется SQL-запрос и выводится информация о клиентах.

* Просмотр операций.

Формируется SQL-запрос, и выводится информация, о об операциях.

* Визуализация информации посредством графиков и круговых диаграмм.

С сервера заращивается статистика об операциях и клиентах. После получения необходимых данных строятся графики о клиентах и операциях.

* Добавления работников.

Необходимо заполнить соответствующую форму и нажать кнопку «Добавить», после чего данные занесутся в базу данных.

Добавление учетной информации о работниках предприятия. Необходимо выбрать нужного нам сотрудника добавить его в учет деятельности и далее уже проводить нужные нам операции, такие как назначение дисциплинарного взыскания, снятие дисциплинарного взыскания, добавление отработанных часов, назначение премии.

# 2.1.1 Java

Java – полностью объектно-ориентированный язык программирования. Java используется для разработки программного обеспечения проектов различного назначения: начиная от приложений для мобильного телефона, заканчивая масштабными корпоративными приложениями. Java может использоваться для создания двух типов программ: приложений и апплетов. Приложения Java могут быть непосредственно выполнены, используя интерпретатор Java. Апплет - небольшая программа, работающая с окнами, которые внедрены в страницу HTML.

Встроенная поддержка многопоточности снабжает программистов Java мощным инструментом для улучшения интерактивной работы графических приложений. Потоки иногда также называют легкими процессами или контекстами выполнения. Библиотека Java обеспечивает класс потока, который содержит обширную коллекцию методов запуска, выполнения, и остановки потока, а также проверки его состояния.

Самой выдающейся чертой языка Java является то, что его приложения компилируются в специальный байт-код и выполняются в специальной виртуальной машине в любой операционной системе и на любом оборудовании.

Другим важным преимуществом языка является довольно высокая безопасность. Поскольку все приложения выполняются и, соответственно, контролируются Java-машиной, то при любой попытке несанкционированного доступа или соединения программа просто прерывает свою работу. Некоторые программисты выделяют и более упрощённую саму форму написания программ Java по сравнению с её предшественниками, коими являются С, С+ и С++. Помимо этого, к языку существует огромное количество бесплатных программных библиотек, написанных программистами всего мира.

Также Java обеспечивает автоматическое освобождение неиспользуемой памяти. Наряду с распределением памяти программистом JVM ведет учет всех используемых блоков памяти и указателей на них. Если блок памяти больше не используется, он может быть освобожден. Это обеспечивает процесс, который называется "сборкой мусора". Он периодически вызывается JVM, проверяет все используемые блоки памяти и освобождает те из них, на которые отсутствуют указатели. Сборка мусора очень удобна, но за ее использование приходится расплачиваться большим потреблением памяти и низкой производительностью.

# 2.1.2 JavaFX

JavaFX — платформа на основе [Java](https://ru.wikipedia.org/wiki/Java) для создания [приложений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) с насыщенным [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F). Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений ([RIA](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%8B%D1%89%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)), работающих в [браузерах](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку [Swing](https://ru.wikipedia.org/wiki/Swing" \o "Swing). Платформа JavaFX конкурирует с [Microsoft Silverlight](https://ru.wikipedia.org/wiki/Microsoft_Silverlight" \o "Microsoft Silverlight), [Adobe Flash](https://ru.wikipedia.org/wiki/Adobe_Flash" \o "Adobe Flash) и аналогичными системами.

# 2.1.3 MySQL

MySQL – это система управления реляционными базами данных.

В реляционной базе данные хранятся в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

Система безопасности MySQL основана на привилегиях и паролях с возможностью верификации с удаленного компьютера, за счет чего обеспечивается гибкость и безопасность. Пароли при передаче по сети при соединении с сервером шифруются. Клиенты могут соединяться с MySQL, используя сокеты TCP/IP, сокеты Unix или именованные каналы.

MySQL поддерживает параллельный доступ к базе данных. Приложения могут обращаться к базе данных одновременно, что повышает общую производительность системы. Кроме того, отдельные операции могут «распараллеливаться» для еще большего улучшения производительности.

MySQL помогает восстанавливать информацию в случае непредвиденного сбоя, незаметно для пользователей создавая резервные копии данных. Все изменения, вносимые в базу данных, регистрируются, поэтому многие операции можно отменять и выполнять повторно.

# 2.3 Модели представления системы и их описание

# 2.2.1 Диаграмма классов

Диаграмма классов является центральным звеном объектно-ориентированного подхода. Она является одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграммы классов используются при моделировании ПС наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

# 2.2.2 Диаграмма последовательности

 Диаграммы последовательности визуально моделируют поток логики в системе, позволяя документировать и проверять свою логику, и обычно используются как для анализа, так и для целей проектирования. Диаграммы последовательности являются наиболее популярным артефактом UML для динамического моделирования, которое фокусируется на определении поведения в данной системе. Другие методы динамического моделирования включают в себя диаграмму активности, диаграмму связи, временную диаграмму и диаграмму обзора взаимодействия. Диаграммы последовательности, а также диаграммы классов и физические модели данных, на мой взгляд, являются наиболее важными моделями уровня проектирования для разработки современных Тур-приложений.

# 2.2.3 Диаграмма состояния

Объекты меняют своё состояние в ответ на происходящие события и с течением времени. Диаграмма состояний представляет состояния объекта и переходы между ними, а также начальное и конечное состояние объекта.

Основными элементами диаграммы состояний являются «Состояние» и «Переход». Диаграмма состояний имеет схожую семантику с диаграммой деятельности, только деятельность здесь заменена состоянием, переходы символизируют действия. Таким образом, если для диаграммы деятельности отличие между понятиями «Деятельность» и «Действие» заключается в возможности дальнейшей декомпозиции, то на диаграмме состояний деятельность символизирует состояние, в котором объект находится продолжительное количество времени, в то время как действие моментально.

Цель данной диаграммы является показать поведение одного объекта в течение его жизни, начиная от создания объекта и заканчивая его уничтожением.

В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта.

# 3 ОПИСАНИЕ ОСНОВНОГО ПРОЦЕССА ФУНКЦИОНИРОВАНИЯ РАБОТЫ СИСТЕМЫ УПРАВЛЕНИЯ НА ОСНОВЕ СТАНДАРТА IDEF0

Анализ предметной области является одним из важнейших этапов проектирования информационной системы. На данном этапе необходимо понять и описать бизнес-логику предметной области. Известно, что исправление ошибок, допущенных на данной стадии, обходится примерно в десять раз дороже, чем на следующих, откуда следует, что начальные стадии проекта являются одними из наиболее критических и требуют применения эффективных средства автоматизации.

Для графического описания модели системы предприятии был использован стандарт **IDEF0** (Integration Definition for Function Modeling), который подставляет собой методологию функционального моделирования, предназначенную для описания бизнес-процессов.

**IDEF0** — это методология графического описания систем и процессов деятельности организации как множества взаимосвязанных функций. Она позволяет исследовать функции организации, не связывая их с объектами, обеспечивающими их реализацию.

В стандарте **IDEF0** посредством входа показывают объекты — информационные и материальные потоки, которые преобразуются в бизнес- процессе. С помощью управления показываются объекты — материальные и информационные потоки, которые не преобразуются в процессе, по нужны для его выполнения. Используя механизмы IDEF0 можно отображать инструменты и ресурсы, с помощью которых бизнес-процесс реализуется (например, технические средства, люди, информационные системы и т.д.). Выход бизнес-процесса, описанного в стандарте IDEF0, полностью соответствует по смыслу выходу процесса, описанному с помощью DFD-схемы.

Детальное рассмотрение процесса системы учета и анализа персонала на предприятии представлен в приложение А. Оно реализовано с помощью нотации IDEF0, которая широко применяется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. Нотация IDEF0 является одной из самых популярных нотаций моделирования бизнес-процессов. На первом уровне декомпозиции используется «контекстная диаграмма». Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой.

# 4 Построение информационной модели

В процессе информационного моделирования были выделены следующие сущности:

* Пользователь;
* Администратор;
* Работник (член персонала предприятия можно сказать как сотрудник);
* Учет.

Сущность «Пользователь» необходима для авторизации в пользовательском приложении. Все пользователи хранятся в отдельной таблице пользователей в которую заносится пользователь после прохождения регистрации. Сущность «Пользователь» содержит такие поля как имя, фамилия, логин, пароль, электронная почта, пол (гендер), место нахождения (регион).

Сущность «Администратор» необходима для хранения аккаунтов администраторов программы. При авторизации в администраторском приложении данные из этой сущности проверяются на введенные и есть они совпадают то авторизация считается успешной. Данные в эту сущность заносятся уже существующими администраторами, иначе говоря уже существующий администратор регистрирует новых администраторов, которые в свою очередь могут регистрировать других администраторов. В данной сущности присутствуют такие поля как имя, фамилия, логин, пароль, электронная почта.

Сущности «Работник» и «Учет» связаны между собой по полю id-Работника. У сущности Работник есть такие поля как имя, фамилия, отчество, отдел, должность, почасовая оплата и id, А сущность Учет также имеет поле id- Работника по которому соединятся с сущностью работник эта сущность содержит информацию об отработанных часах о наличии премиальное поощрения и дисциплинарное взыскание.

С учетом выявленных связей между сущностями смоделируем их взаимодействие в формате IDEF1. Информационная модель соответствует условиям третьей нормальной формы – неключевые атрибуты зависят от ключей, причем от ключей целиком и ни от чего другого, кроме как от ключей.

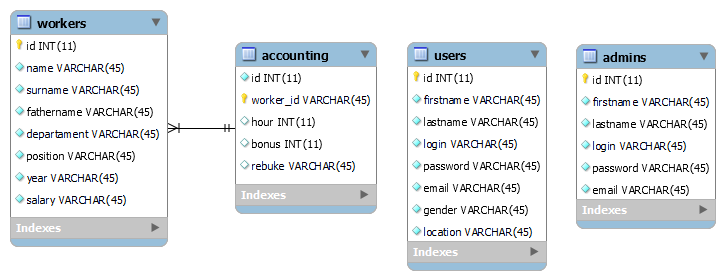


Рисунок 4.1– Информационная модель базы данных

# 5 Спецификация вариантов использования системы

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующих в процессе.

В данном курсовом проекте были определены четыре основных действующих лица: администратор, работник, пользователь и незарегистрированный пользователь.

Последнему предоставлена единственная функция регистрации в системе. Пользователь же уже знает, что он зарегистрирован и он может получить доступ к системе.

Администратору и пользователю представлены некоторые одинаковые функции, такие как:

1. Просмотр списка сотрудников
2. Просмотр аналитики сотрудников
3. Просмотр зарплат каждого сотрудника
4. Просмотр аналитики отделов

Администратору предоставлен более широкий функционал, для него реализованы возможности:

* добавление новых администраторов
* удаление аккаунтов пользователей
* добавление работников;
* редактирование имеющейся информации;
* учет деятельности персонала.

Диаграмма вариантов использования представлена в Use case диаграмме в приложении к записке.

# 6 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы

**Алгоритм** — представляет собой пошаговую процедуру, которая определяет набор инструкций, которые должны быть выполнены в определенном порядке, чтобы получить желаемый результат. Алгоритмы, как правило, создаются независимо от базовых языков, то есть алгоритм может быть реализован на нескольких языках программирования.

# 6.1 Алгоритм работы функции начисления заработной платы

В данной курсовой работе реализован алгоритм начисления заработной платы сотруднику. Заработная плата сотрудника зависит от таких факторов как количество отработанных часов, почасовая оплата труда, наличие дисциплинарных взысканий, наличие премии. Администратор приложения должен добавить сотрудника в учет из уже имеющихся сотрудников, которые находятся в базе данных если же сотрудника нету в базе, то его нужно добавить во вкладке работа с персоналом. Если же сотрудник уже добавлен в учет, то администратор приложения может назначить сотруднику премию в размере процентов от основной заработной платы от 10 до 50 процентов, так же администратор может добавить отработанные часы сотруднику не более 12 часов за сутки. Администратор может назначить дисциплинарное взыскание сотруднику или же снять уже имеющиеся дисциплинарное взыскание. Начисление заработной платы проходит в следующем порядке:

* 1. программа достает данные об учете сотрудника из базы данных (количество часов, размер премии, почасовая зарплата, наличие дисциплинарных взысканий)
  2. Если сотрудник имеет дисциплинарные взыскания, то при расчете заработной платы премия не учитывается.
  3. Программа берет количества отработанных сотрудником часов и умножает на почасовую оплату труда далее при отсутствии дисциплинарных взысканий программа умножает получившуюся сумму на размер премии (в процентах).
  4. Выводит информацию о зарплатах в таблицу и подсчитывает итоговую сумму которое предприятие должно своим сотрудникам

Таким образом работает система начисления заработной платы сотрудникам предприятия на основе данных полученных при учете деятельности сотрудника.

# 6.2 Алгоритм работы всей программы

В данной курсовой работе реализовано два клиентских приложения и одно серверное приложение. При запуске клиентского приложения открывается главное окно с авторизацией в котором пользователь или администратор должен пройти авторизацию в приложении. При успешной авторизации окно с авторизацией скрывается и открывается новое окно с самой рабочей программой. В зависимости от того какое клиентское приложение запустил пользователь будет разный функционал. При авторизации в окне авторизации пользователя будет отображена кнопка регистрации нового аккаунта для дальнейшей работы. Администраторов может регистрировать только другой администратор. Администратор имеет возможность добавлять удалять редактировать просматривать других администраторов и просматривать и удалять аккаунты пользователей также администратору доступны функции учета персонала предприятия а пользователю только функции анализа персонала такие как таблицы и графики дающие частичное представление о проделанной работе на предприятии.

# 6.3 Алгоритм работы функции авторизации

Для проведения авторизации пользователь указывает его логин и пароль, после ввода этих данных производится соединение с БД и на сервер отправляется запрос выборки данных о пользователях. Далее в цикле проверяется совпадение введенных данных с поступающими данными из базы данных. В случае несовпадения вызывается пользователью не будет предоставлен доступ к программе.

# 7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.

# 7.1 Пользовательское приложение

При запуске приложения появляется окно авторизации, в котором мы можем ввести данные аккаунта и параметры подключения к серверу такие как ip с и порт для входа в приложение.

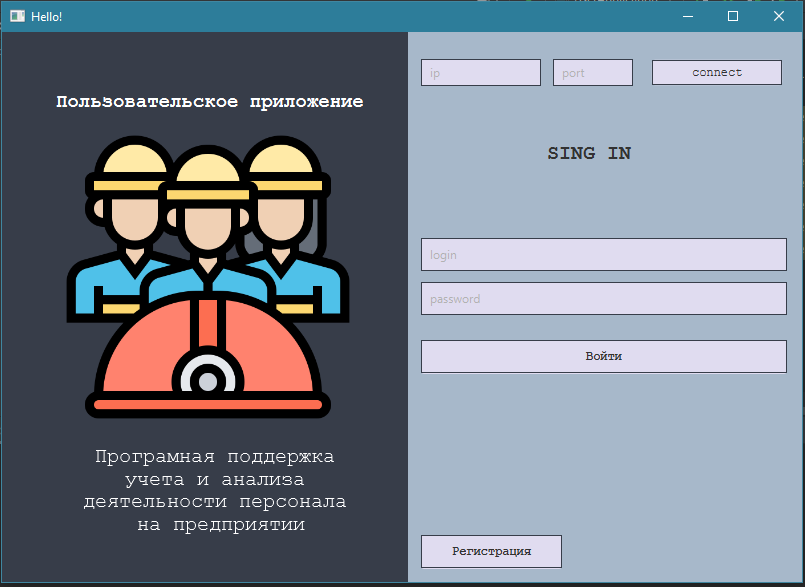


Рис. 7.1

Если же у пользователя нету аккаунта он может нажать на кнопку регистрации и тогда откроется окно регистрации где пользователь может зарегестрировать новый аккаунт.

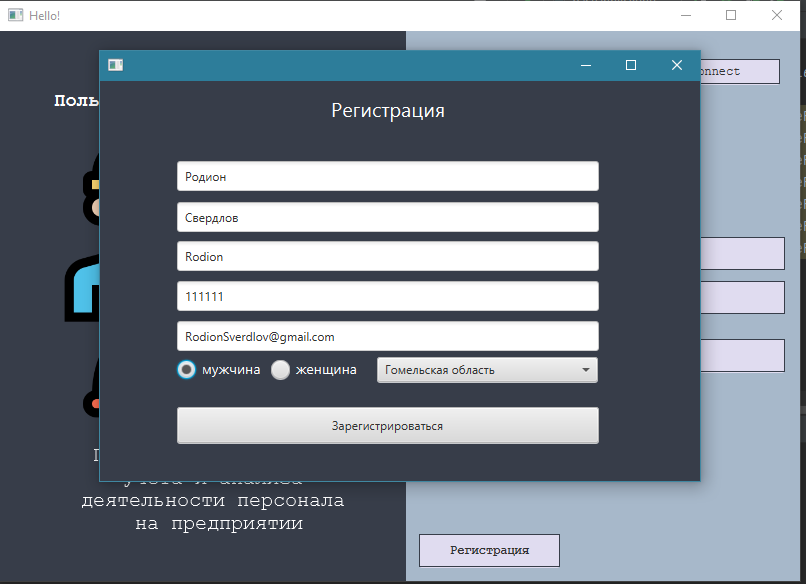


Рис. 7.2

После прохождения авторизации пользователю будет открыто пользовательское приложение.

В пользовательском приложении есть такой пункт как просмотр персонала. В этом пункте есть таблица с персоналом и кнопка обновить которая обновляет таблицу если администраторы внесли изменение в список сотрудников. Рис 7.3



Рис. 7.3

В приложении пользователя есть раздел аналитики деятельности персонала на предприятии, в котором пользователь может получить всю основную информацию об работе предприятия наглядно на круговых диаграммах и таблицах, так же пользователь может рассчитать зарплаты сотрудников и общую итоговую сумму денег которое предприятие должно своим сотрудникам.

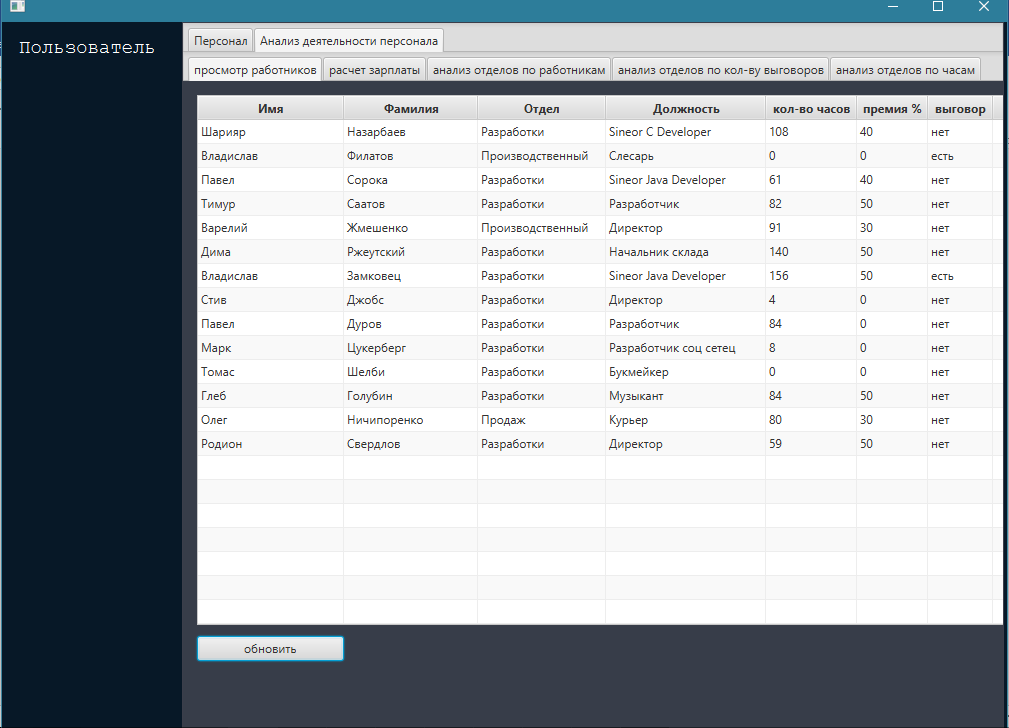


Рис. 7.4

На рисунке 7,4 пользователь может посмотреть всех сотрудников которые внесены в учет, узнать количества отработанных часов, наличие премии, наличие дисциплинарного взыскания. Так же пользователь может обновить статистику если администратор приложения внесет изменения в таблицу.

Далее идет вкладка «Расчет заработных плат сотрудникам» в которой пользователь может получить подробную информацию о заработных платах сотрудников и итоговую сумму зарплат.

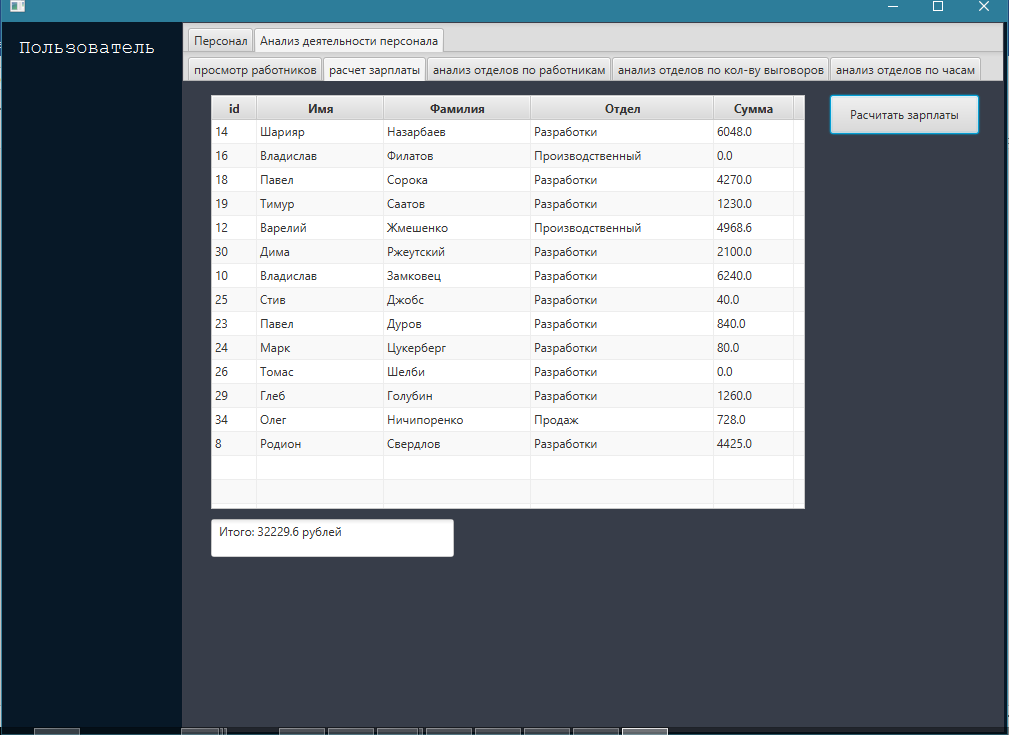


Рис 7.5

В данной таблице предоставлена информация о зарплатах сотрудникам

Далее пользователь может посмотреть более наглядную статистику по работе предприятия на круговых диаграммах.

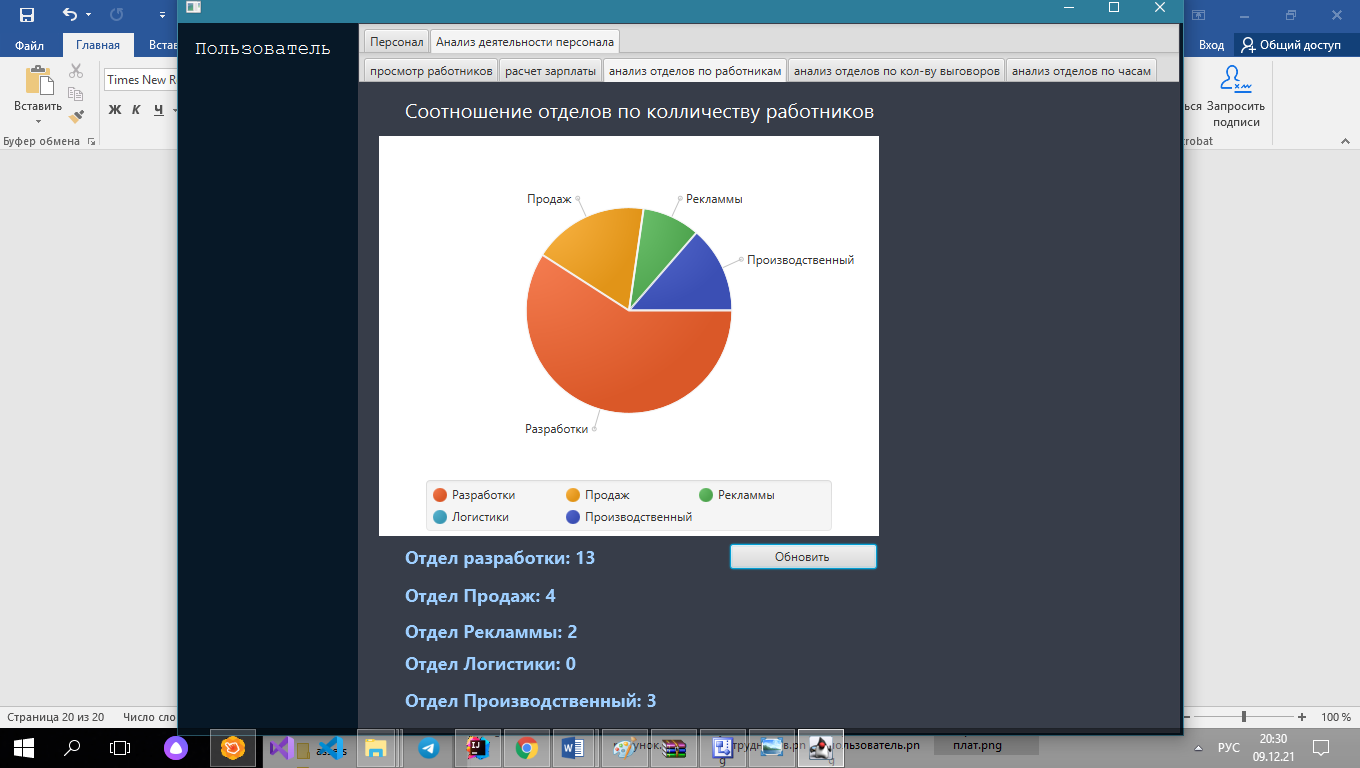


Рис 7.6

На данной круговой диаграмме (Рис. 7.6) представлена статистика по отделам которая показывает сколько работников работает в каждом отделе, другими словами предоставляет информацию о размере отделов. Так же в данной вкладке есть кнопка «Обновить» которая обновляет статистику если администратор приложения внес изменения в таблицу.

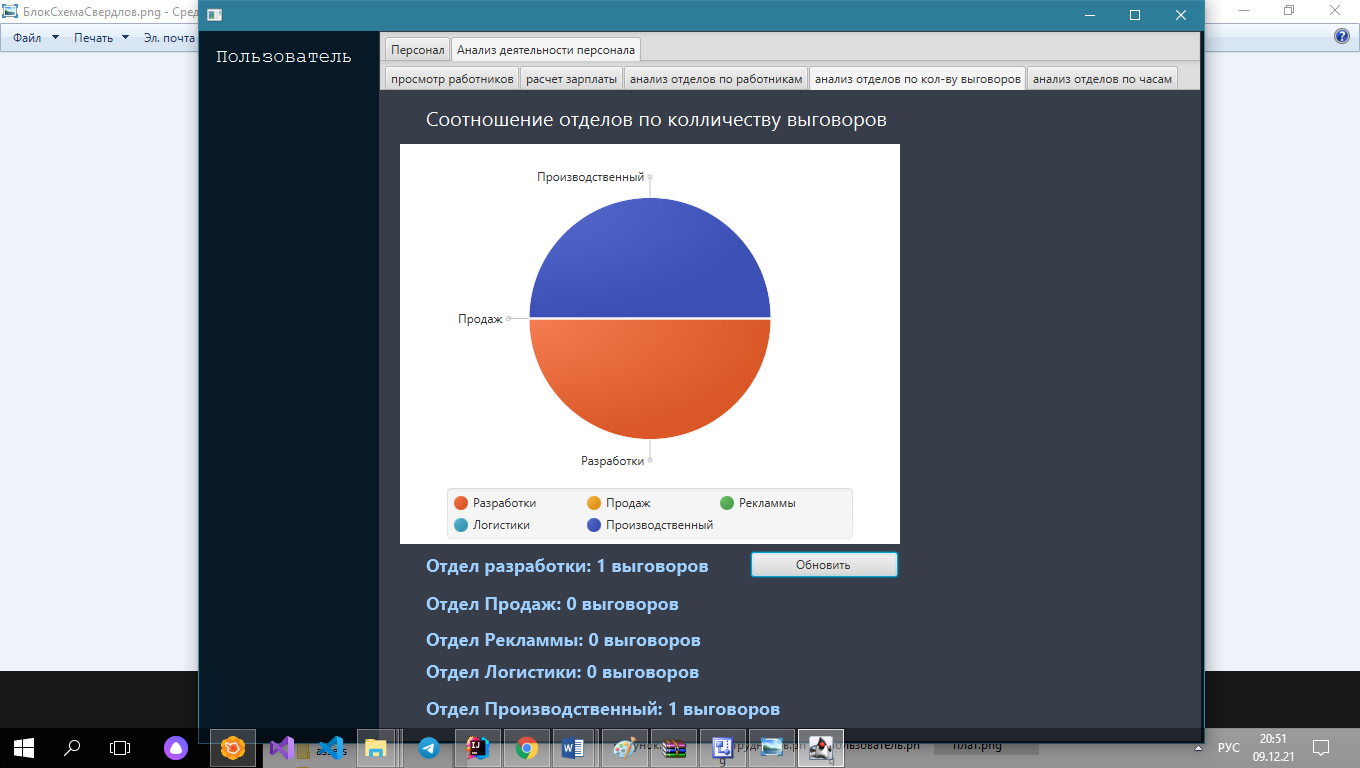


Рис 7.7

На рисунке 7.7 переде нами предстает круговая диаграмма которая показывает информацию о соотношении отделов количеству дисциплинарных взысканий а под данной круговой диаграммой располагается информация в текстовом виде которая отражает количественное соотношение дисциплинарных взысканий на каждый отдел. Также под круговой диаграммой располагается кнопка «обновить» при нажатии на которую будет обновлена статистика если администратор ввел изменения в учет деятельности сотрудников предприятия.

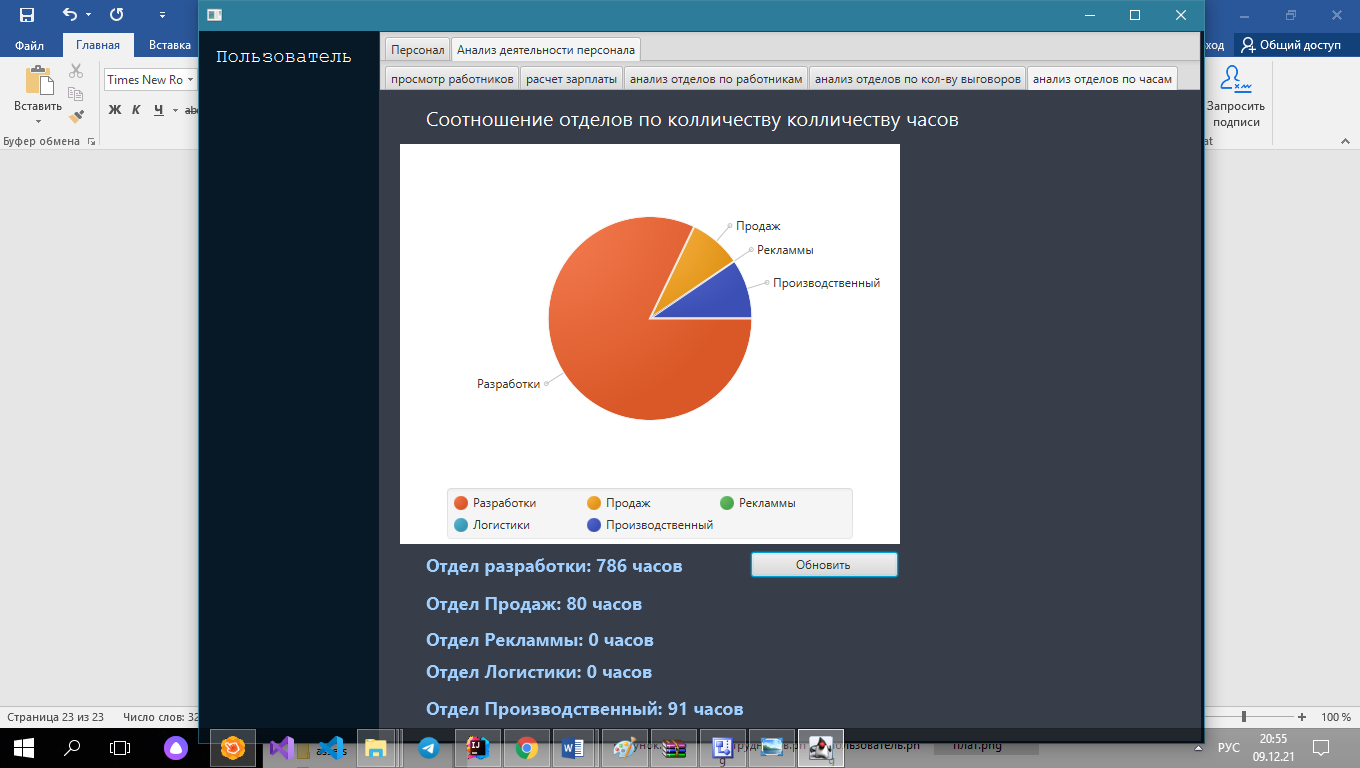


Рис 7.8

Во вкладке «анализ отделов по часам» мы можем посмотреть соотношение отделов по количеству отработанных сотрудниками часов. Мы можем увидеть круговую диаграмму, которая визуально показывает нам соотношение отделов, а под данной круговой диаграммой мы можем увидеть суммарное количество отработанных часов всеми сотрудниками определенного отдела, которых администратор приложения добавил в учет. Также под диаграммой есть кнопка «обновить» которая обновляет нам статистику, если администратор приложения внес изменения в учет деятельности сотрудников.

# 7.2 Приложение администратора

При запуске приложения администратора нас встречает окно авторизации в котором администратор должен указать логин и пароль, а также данные для подключения к серверу такие как ip адрес и порт. В данном окне отсутствует кнопка регистрации т.к. регистрировать администратора могут только другие администраторы. (Рис 7.9)



Рис. 7.9

После успешной авторизации в приложении администратору открывается администраторское приложение. В котором администратора встречает вкладка «Работа с администраторами».

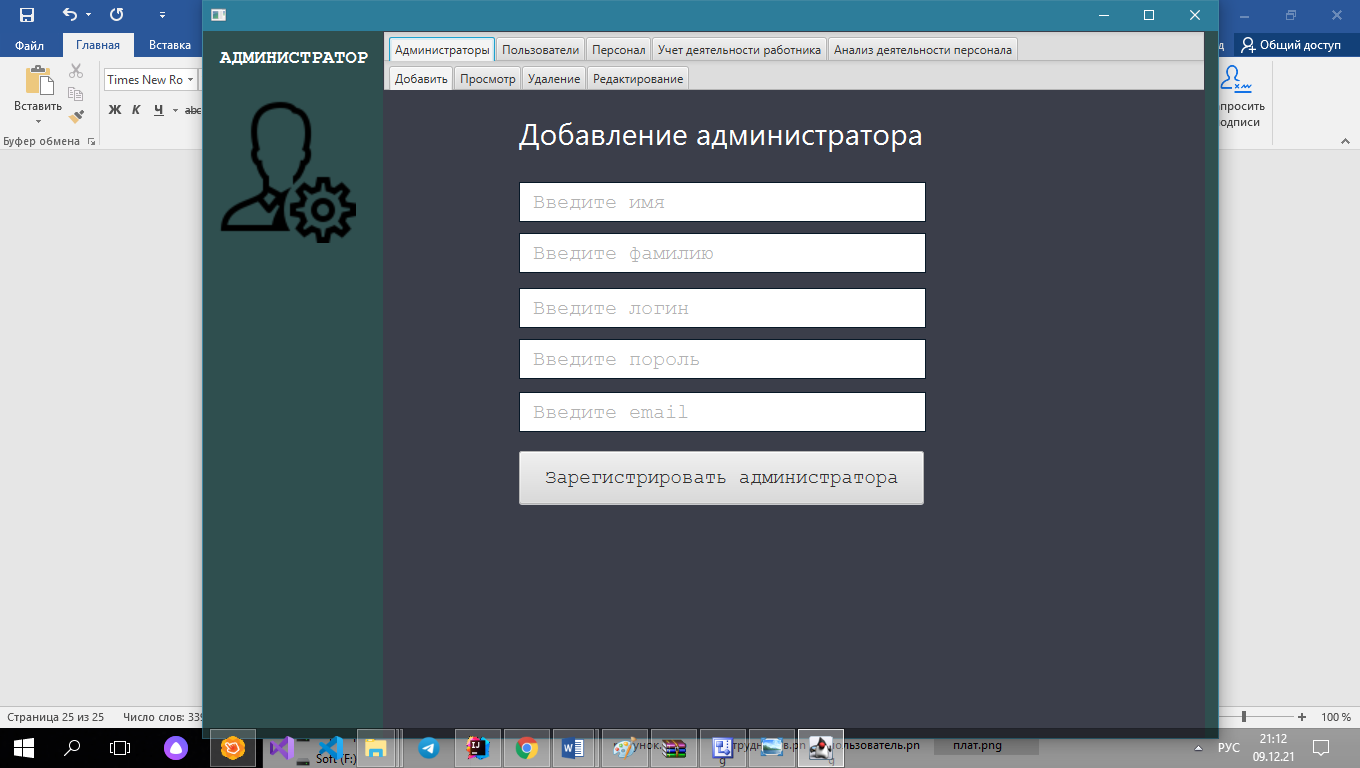


Рис 7.10

Во вкладке «Администраторы» имеется интерфейс добавления нового администратора. Для добавления нового администратора нужно ввести имя, фамилию, логин, пароль, и электронную почту будущего администратора. Далее нужно нажать на кнопу «Зарегистрировать администратора» и после этого новый администратор будет добавлен в базу данных администраторов.

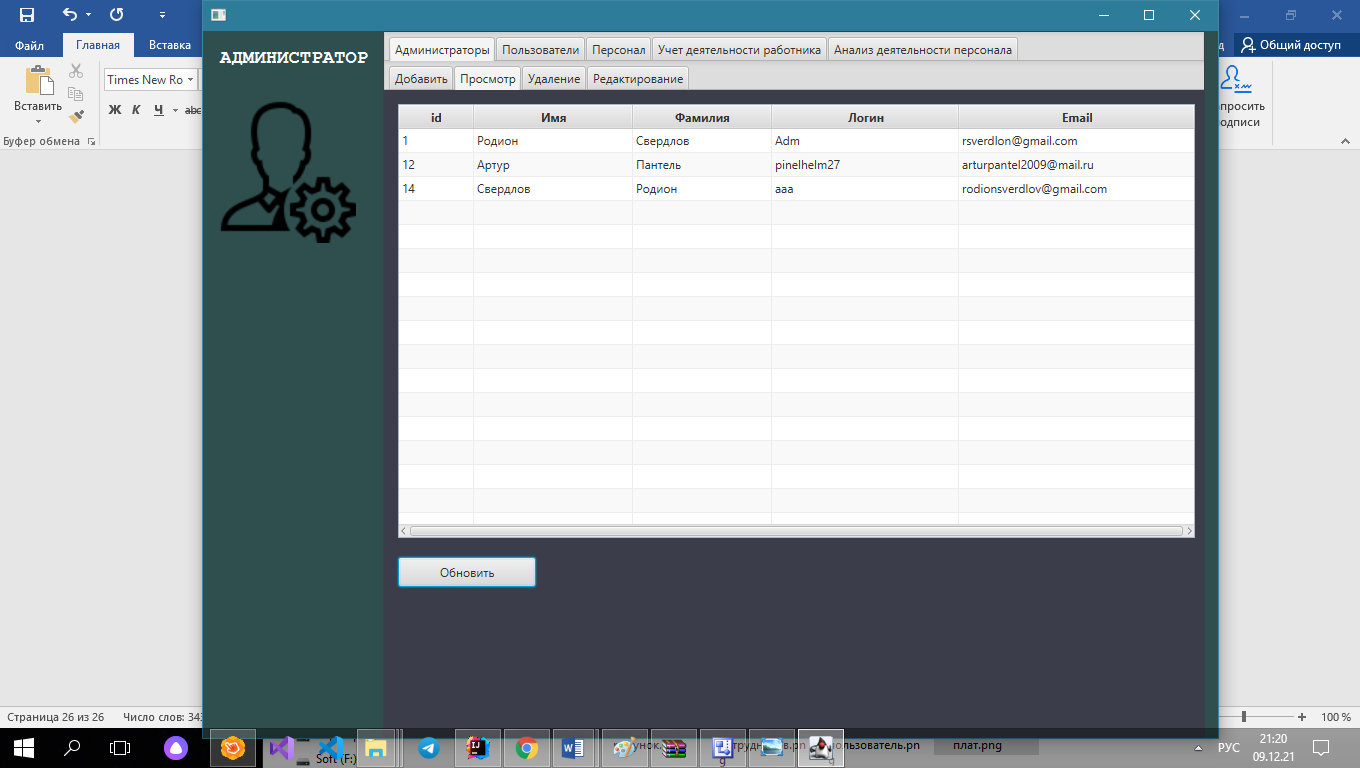


Рис 7.11

Окно «просмотр» предназначено для просмотра уже существующих администраторов здесь можно узнать имя, фамилию, логин, и электронную почту других администраторов. При нажатии на кнопку обновить таблица администраторов будет обновлена если в нее внесены соответствующие изменения.

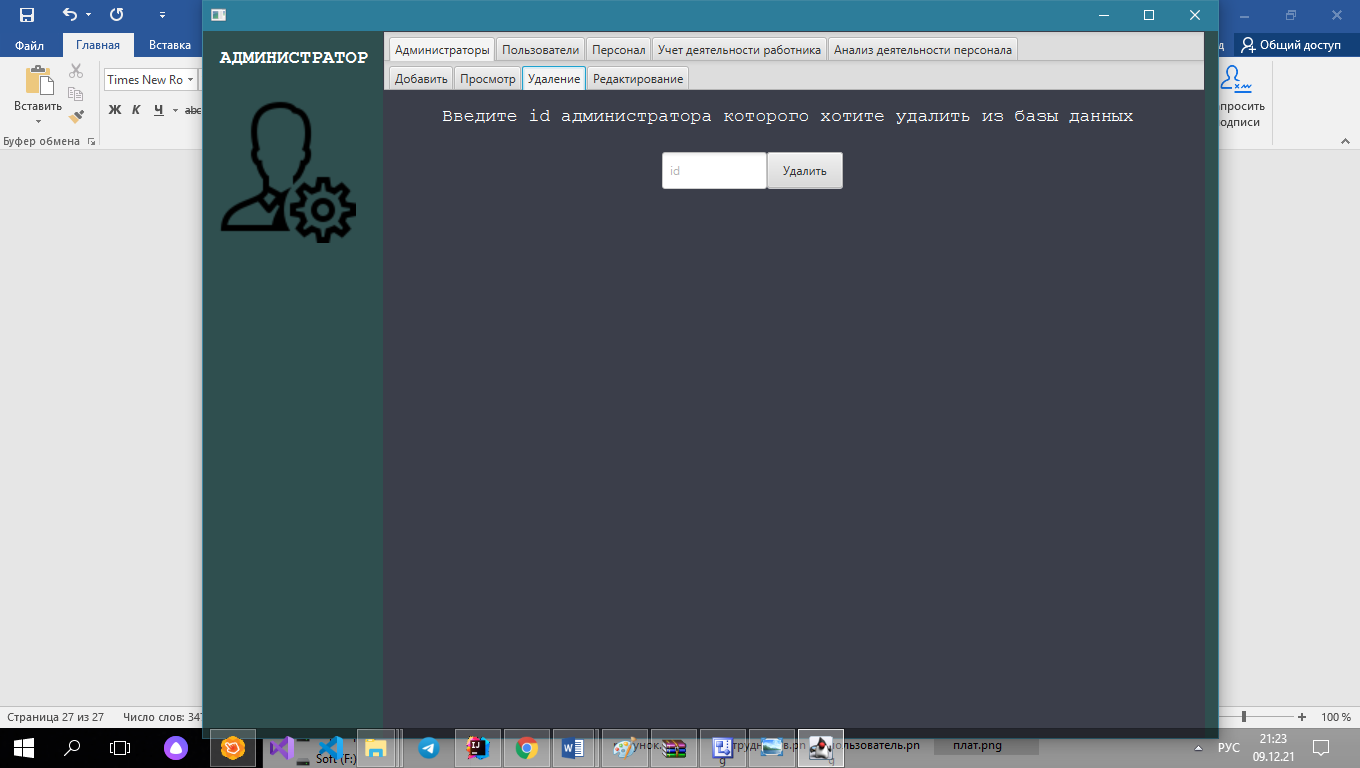


Рис 7.12

Во вкладке «Удаление администратора» можно удалить других администраторов и самого себя, но при этом зайти в приложение уже будет нельзя. Для удаления администратора необходимо ввести идентификационный номер администратора и нажать на кнопку «Удалить» и если администратор с таким идентификационным номером существует, то он будет удален, чтобы узнать результат удаления администратора нужно зайти во вкладку просмотр администраторов (Рис 7.11) и нажать на кнопку «Обновить».

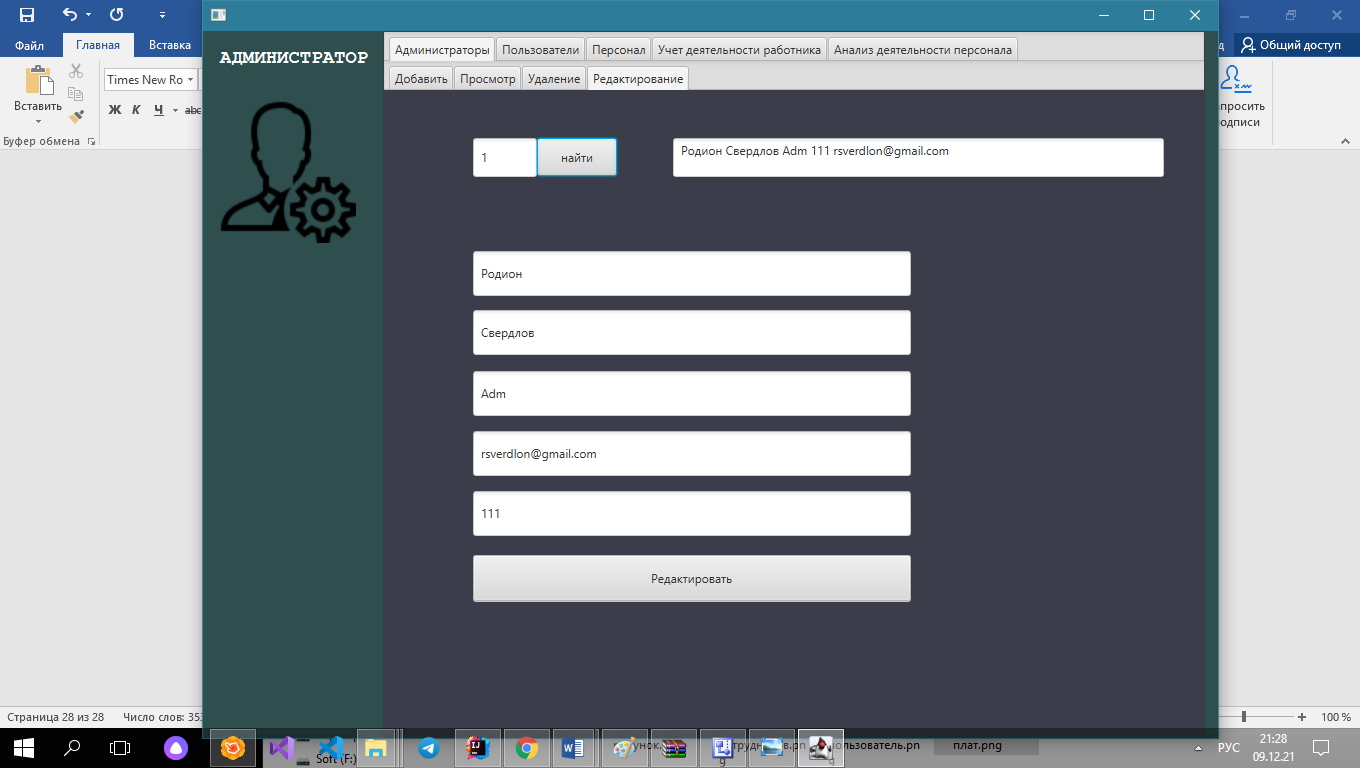


Рис 7.13

Во вкладке редактирование администратора нам доступна функция редактирования администратора в которой администратору необходимо ввести идентификационный номер администратора, которого он хочет отредактировать и нажать кнопку найти далее будет предоставлена текущая информация, об администраторе которую мы можем отредактировать. Также администратор может редактировать и себя. После нажатия на кнопку администратора администратор будет отредактирован, чтобы узнать результат редактирования необходимо перейти во вкладку просмотр администраторов (Рис. 7.11) и нажать на кнопку обновить.

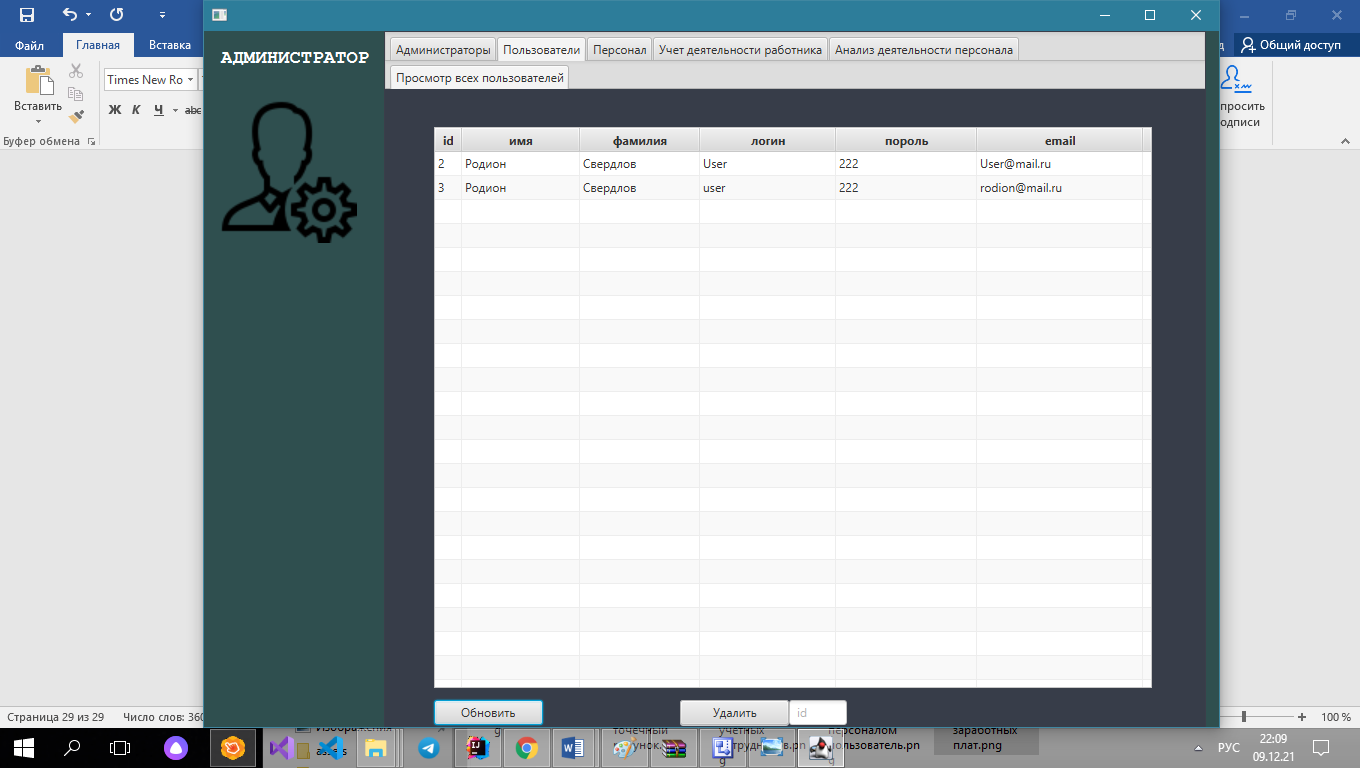


Рис 7.14

Раздел «Пользователи» включает в себя только одну вкладку «Просмотр всех пользователей» которая объединила в себе функции просмотра и удаления аккаунтов пользователей. Перед администратором предстает таблица где можно увидеть информацию об аккаунтах пользователей, под данной таблицей находится две кнопки «обновить» и кнопка «удалить». При нажатии на кнопку обновить информация об аккаунтах пользователей будет обновлена если администратор удалил пользователя либо же новый пользователь зарегистрировался. Для удаления аккаунта пользователя администратору необходимо ввести идентификационный номер аккаунта пользователя в поле ввода идентификационного номера аккаунта пользователя, далее администратору необходимо нажать на кнопку «удалить» и тогда аккаунт выбранного администратором пользователя будет удален.

Одним из самых важных разделов данного приложения является раздел, «Работники» который предоставляет администратору возможность добавлять нового работника в базу данных, просматривать всех уже добавленных работников, удалить работника из базы данных и редактировать информацию о работнике.

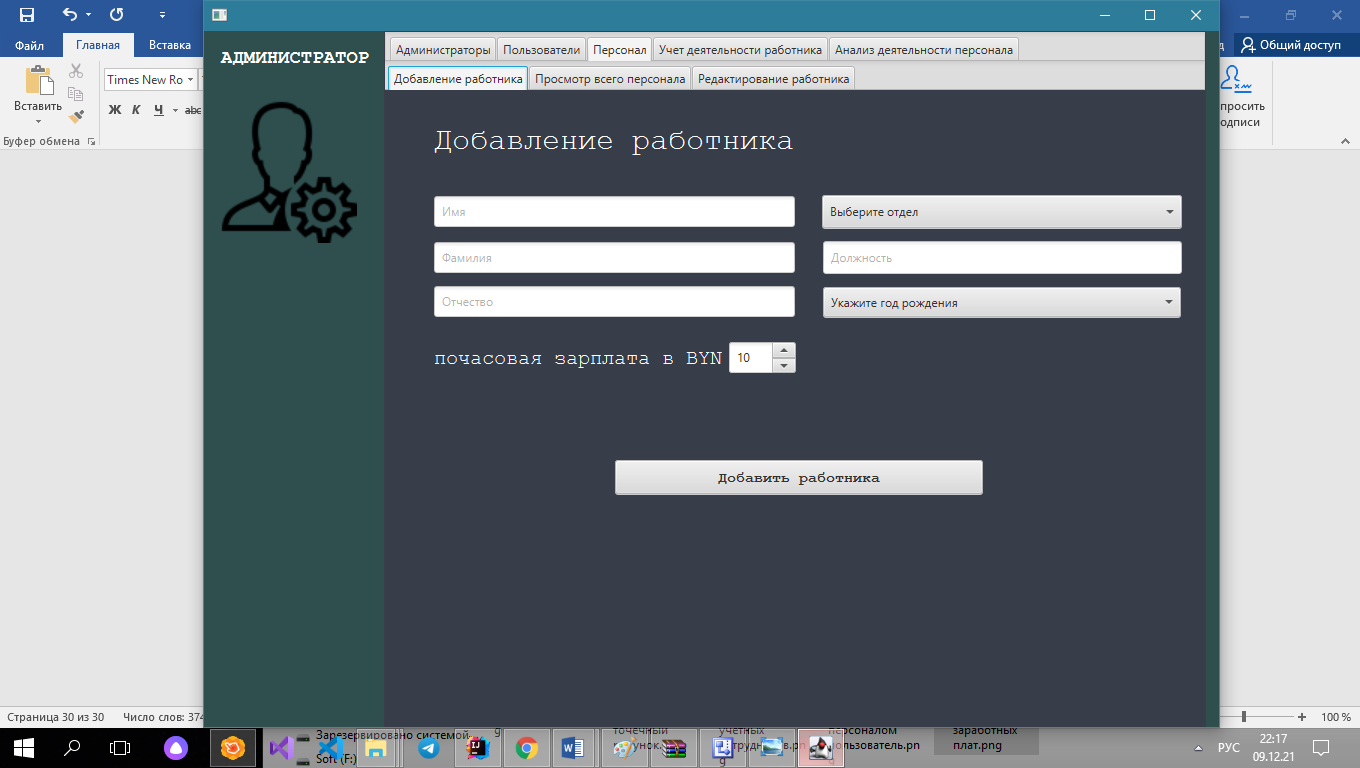


Рис 7.15

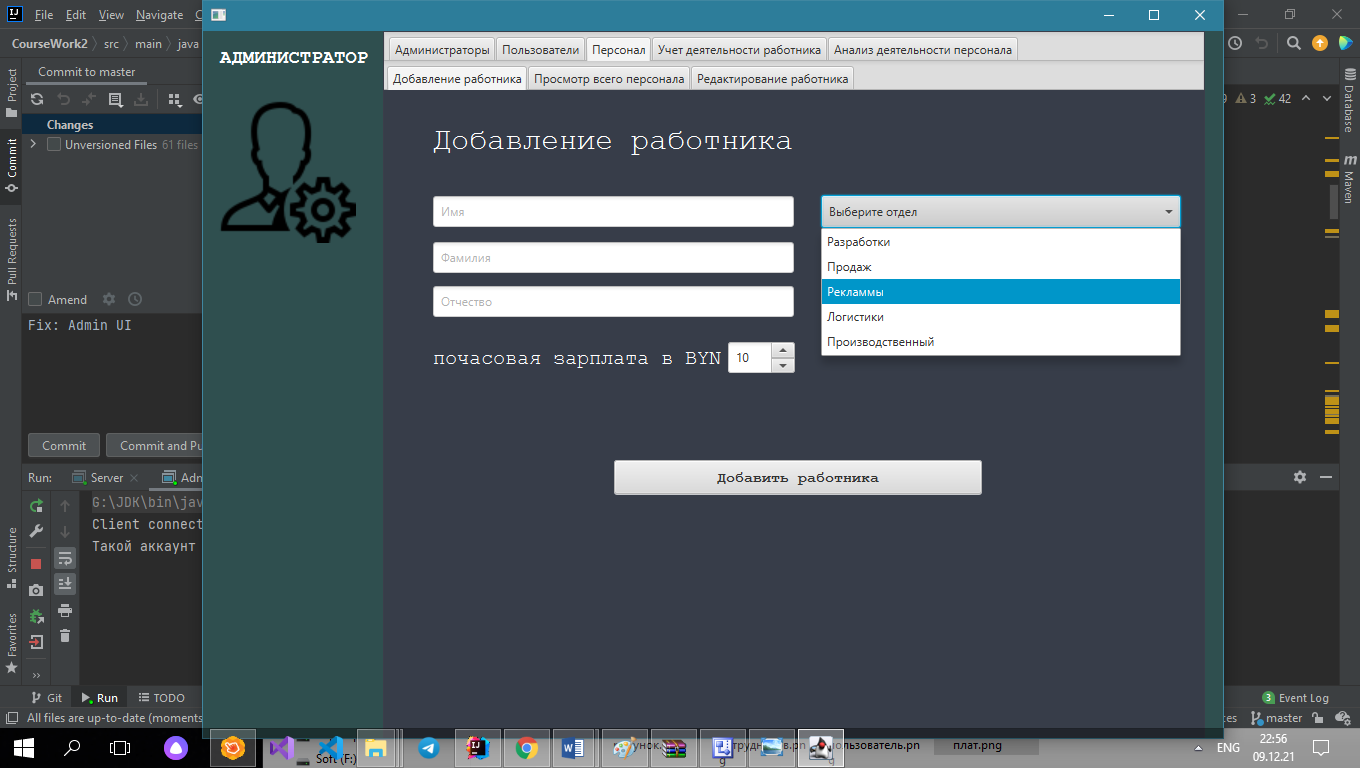


Рис 7.16



Рис 7.17

Для добавления работника администратор должен указать имя, фамилию, отчество, выбрать отдел из пяти предложенных (Разработки, Продаж, Рекламы, Логистики, Производственный) Рис.7.16. Далее администратору необходимо указать должность и год рождения работника. Далее нужно указать заработную плату работника в час которая в дальнейшем будет использована для расчета заработной платы. После того как администратор введет все параметры работнику необходимо нажать на кнопку «Добавить работника» после чего работник будет добавлен в базу данных.

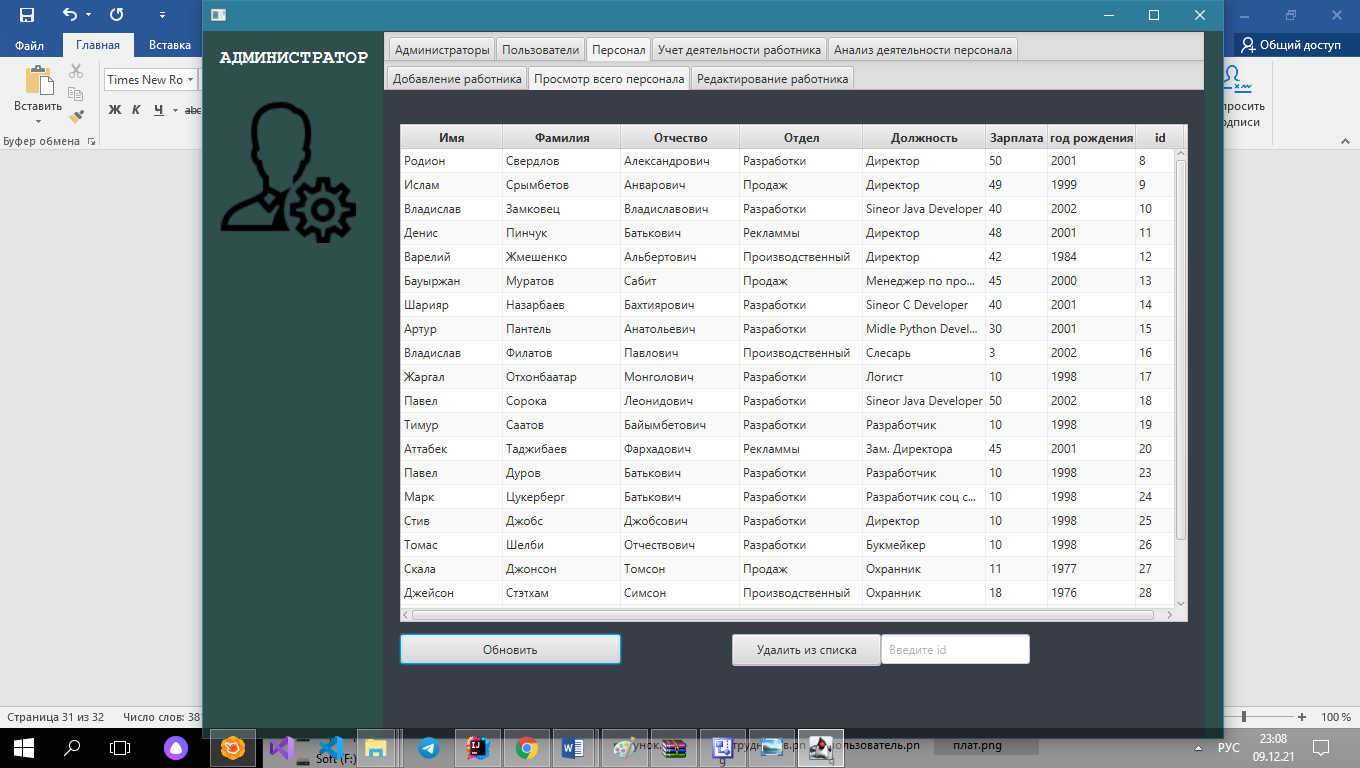


Рис 7.18

Во вкладке просмотр всего персонала можно получить подробную информацию о всех работниках, добавленных в базу данных, для этого необходимо нажать на кнопку обновить которая обновляет таблицу работников если данный либо же другой администратор внес в нее изменения. Под таблицей находится кнопка «Удалить из списка». Для удаления работника из таблицы необходимо ввести идентификационный номер работника в поле ввода идентификационного номера работника и нажать кнопку «удалить работника из списка» после чего работник будет удален, для проверки успешности удаления необходимо обновить таблицу работников нажав на кнопку «обновить».

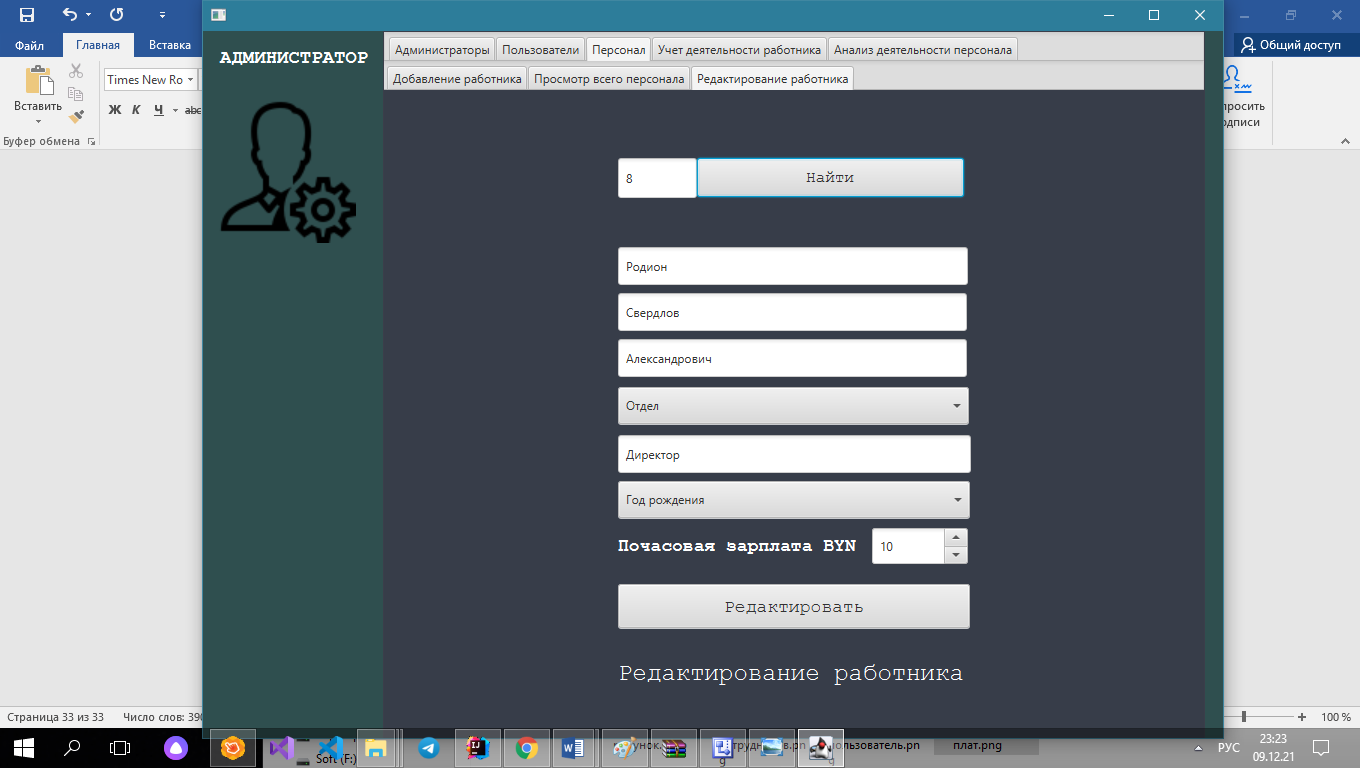


Рис. 7.19

Во вкладке «Редактирование работника» администратору нужно выбрать работника которого он хочет редактировать, для этого нужно ввести идентификационный номер работника в поле ввода идентификационного поля работника и нажать на кнопку «Найти» после чего в полях редактирования появится текущая информация о работнике которую мы можем отредактировать, после редактирования полей информации о работниках необходимо нажать на кнопку «Редактирование» после чего информация о работнике будет отредактирована и перезаписана в базу данных.

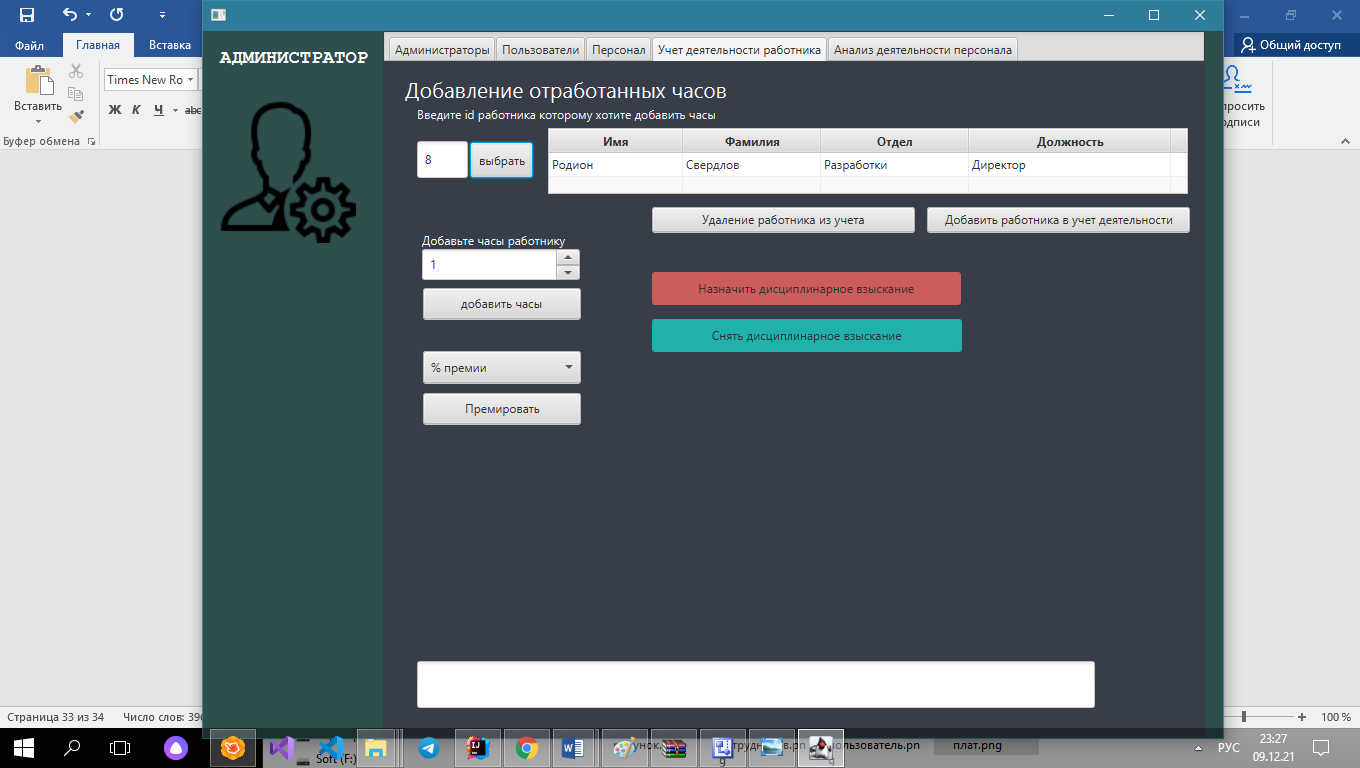


Рис. 7.20

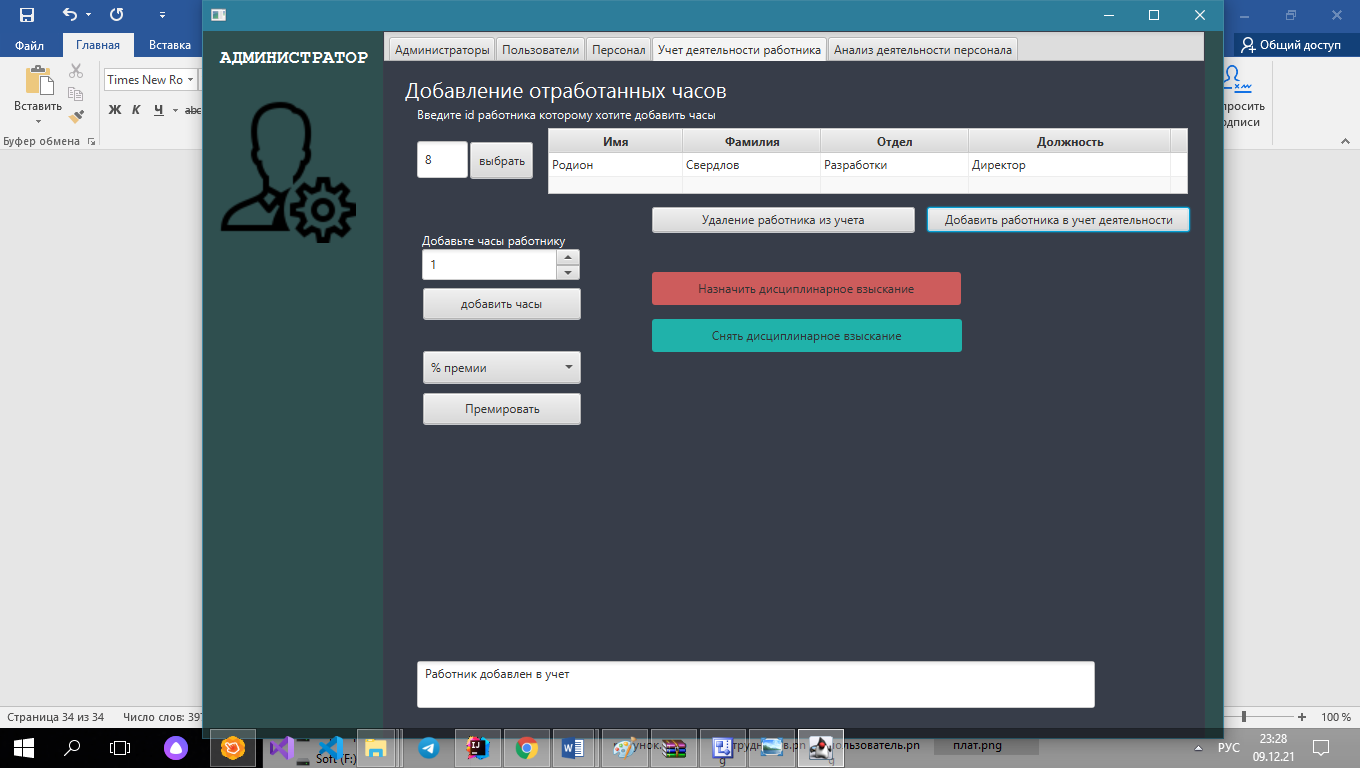


Рис 7.21



Рис.7.22

Вкладка «Учет деятельности работника» является ключевой в этом приложении. В данной вкладке ведется учет деятельности работника, на основании которого будет вестись расчет заработной платы сотрудникам и анализ деятельности сотрудников на предприятии. Эта вкладка включает в себя такие функции как добавление работника в учет, удаление работника из учета, добавление часов работнику, назначение премии, назначение дисциплинарного взыскания работнику и снятие уже имеющегося дисциплинарного взыскания. Для работы с учетом деятельности работника в первую очередь администратору необходимо ввести идентификационный номер работника в поле ввода идентификационного номера работника после чего необходимо нажать на кнопку «выбрать» после чего в таблице будет показана текущая информация о работнике имя, фамилия, отдел и должность. После выбора работника администратор может добавить работнику отработанные часы. Для этого необходимо выбрать количество часов, которое он хочет назначить работнику от 1 до 12, и нажать на кнопку добавить часы после чего в поле вывода текущего состояния внизу экрана будет выведено текущее состояние. Кнопка «премировать» предназначена для начисления премии сотруднику, для этого необходимо выбрать размер премии (в процентах) Рис 7.22. и нажать на кнопку премировать. Также в данной вкладке присутствует две кнопки красная и зеленая. При нажатии на красную кнопку работнику будет выдано дисциплинарное взыскание, а при нажатии на зеленую с работника будет снято текущее дисциплинарное взыскание если такое имеется.

Вкладка «анализ деятельности персонала на предприятии» аналогична той которая имеется в пользовательском приложении.

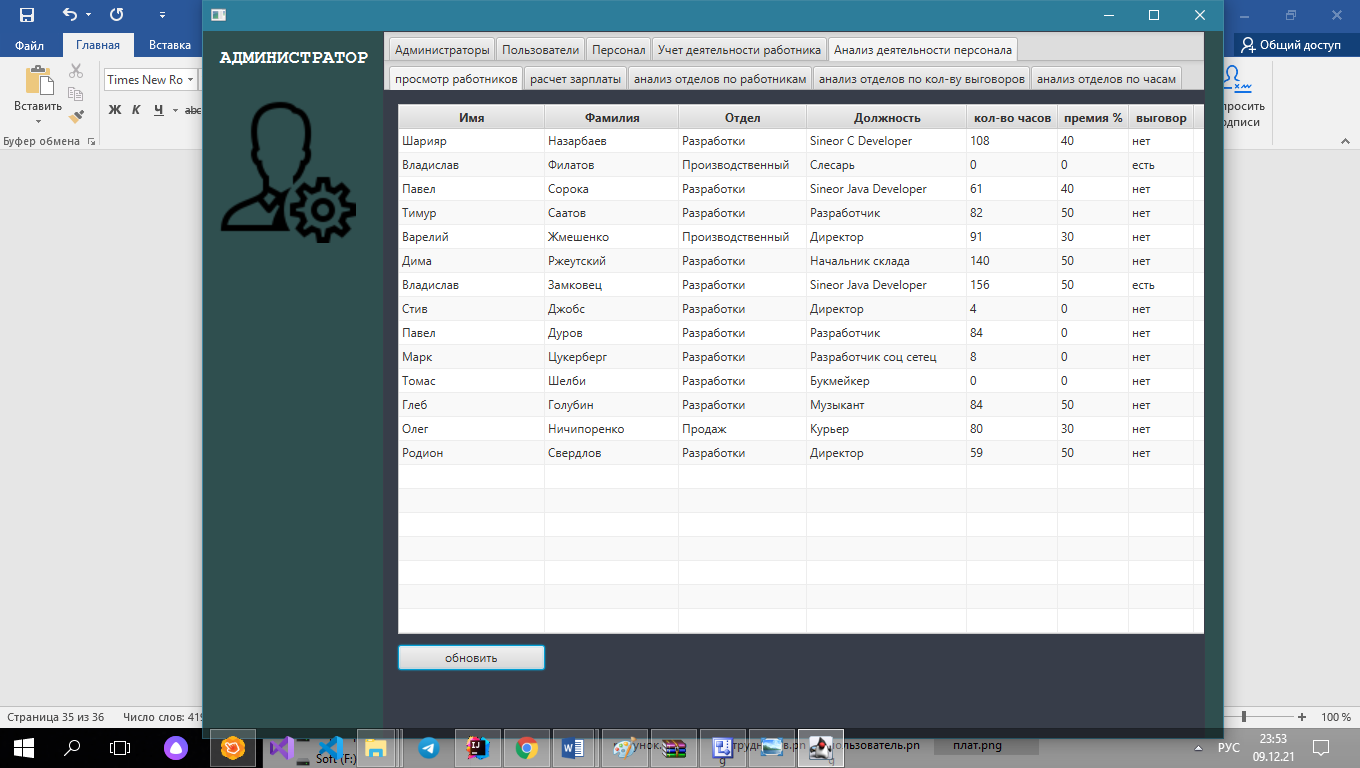


Рис 7.23

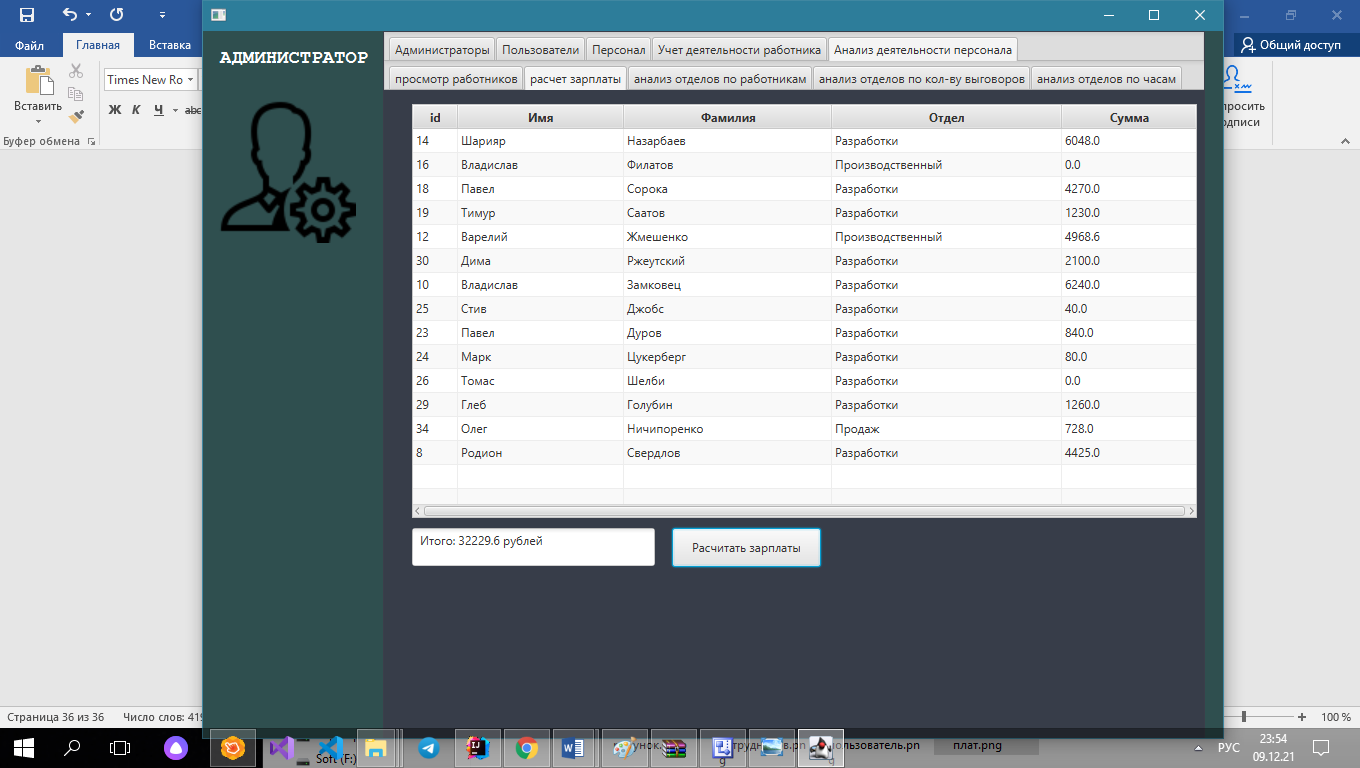


Рис. 7.24

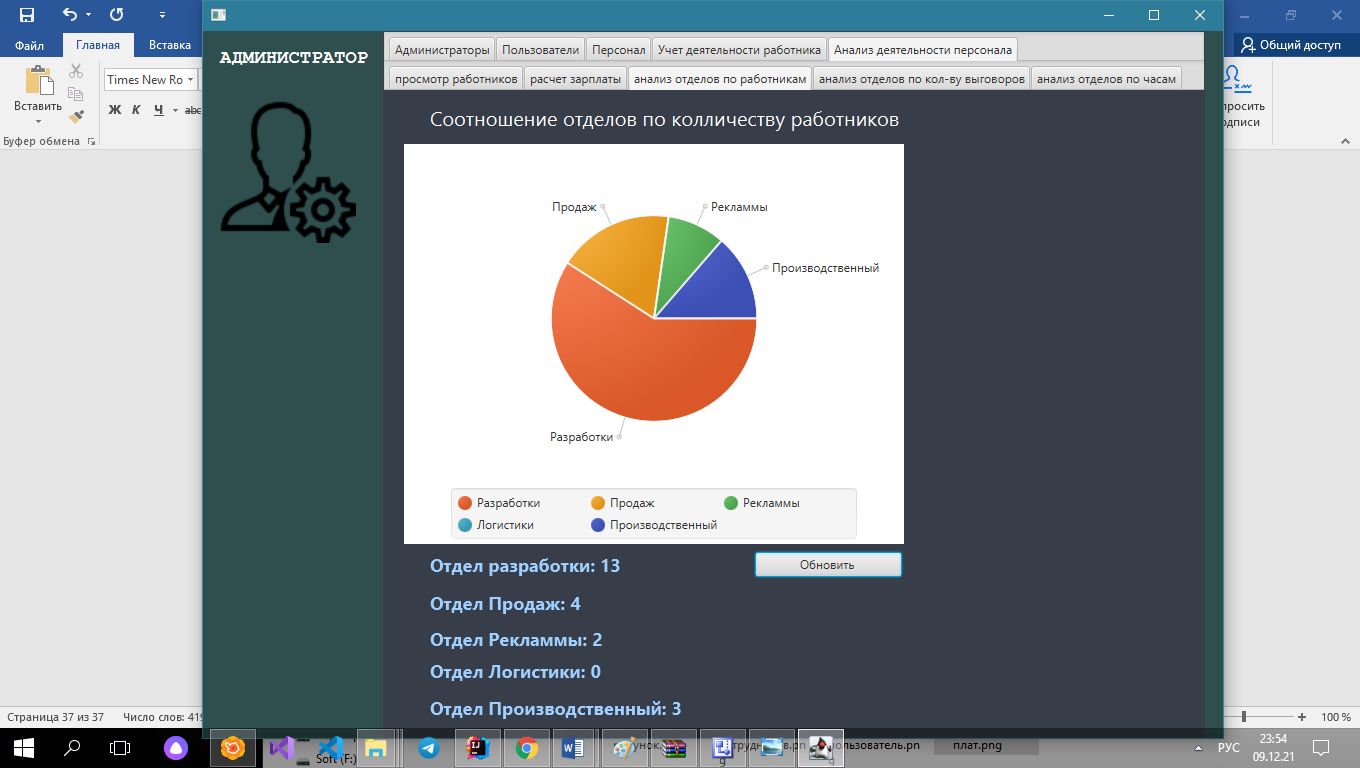


Рис 7.25

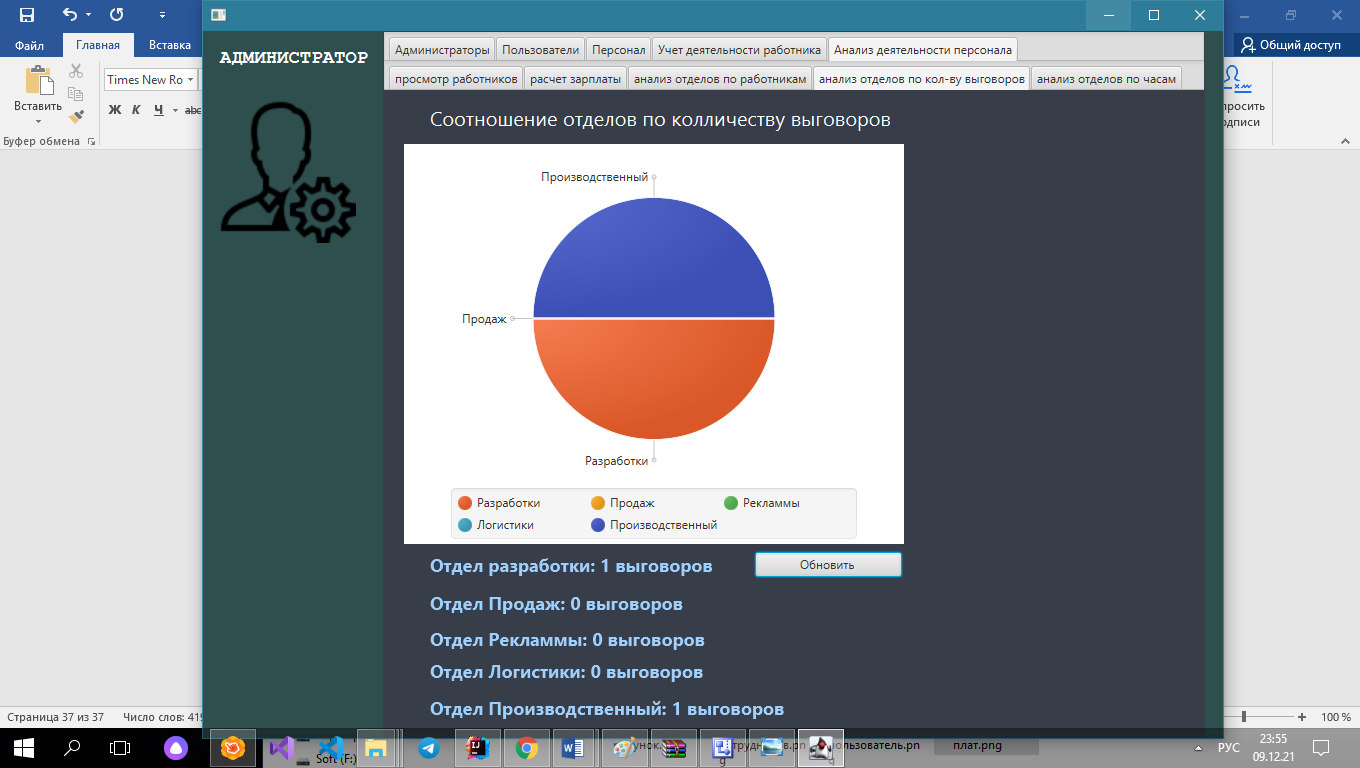


Рис 7.26

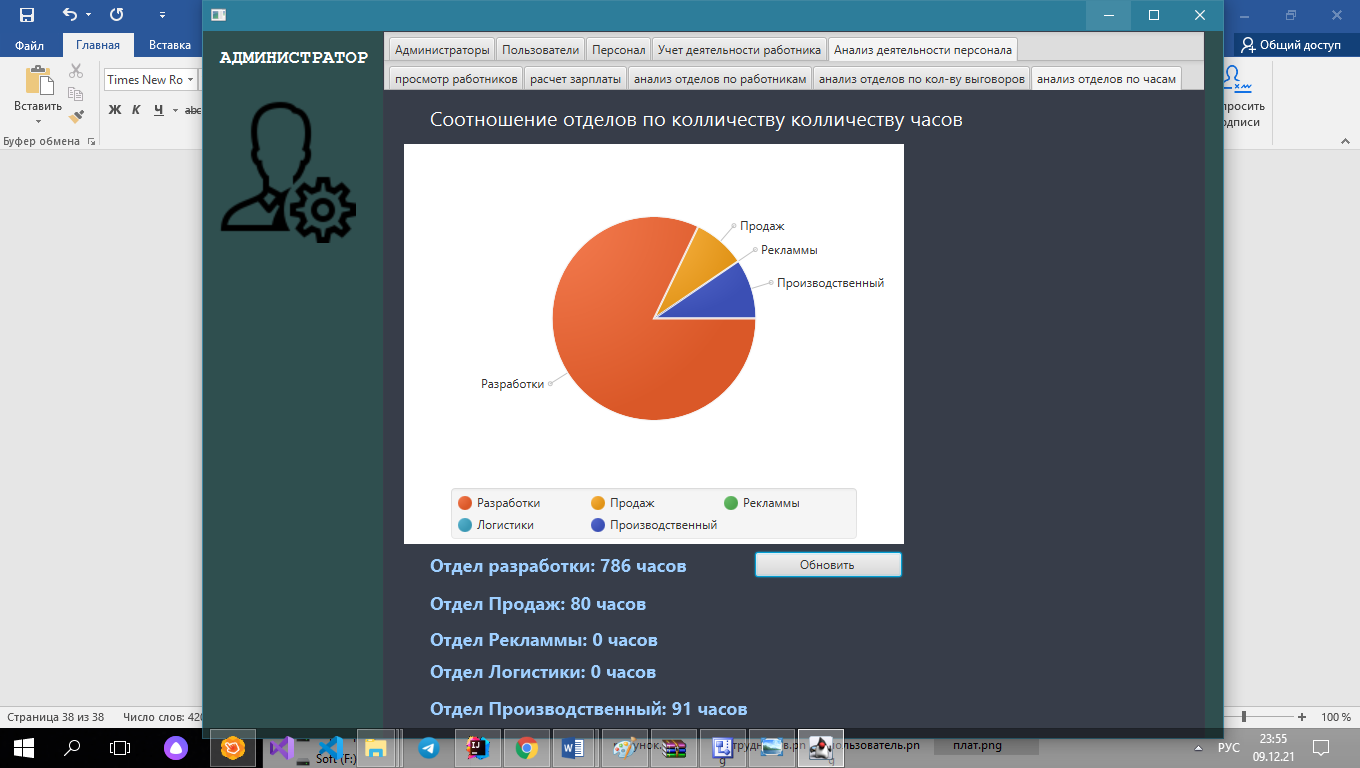


Рис 7.27

# Заключение

Прогресс не стоит на месте, поэтому всегда актуально создавать что-то новое, например, новые писать программы, которые упростят и ускорят работу человека и у сведут ручную работ к минимуму.

Автоматизация различных сфер жизни человека является очень значимой и важной, так как сокращает время обработки различной документации и освобождает сотрудников от рутинной работы. Автоматизация учёта и анализа деятельности персонала на предприятии требует использования специального программного продукта. Так как специфика деятельности в данной сфере требует хранение больших массивов информации, а также их систематизацию, необходимо создание баз данных и систем по их управлению, чтобы обрабатывать максимальное количество информации за минимальное время.

Подводя итоги, можно сказать, что созданная программа разработана для облегчения и удобства в работе с базами данных в бухгалтерских и управляющих отделах предприятий и организациях. Приложение приводит в систему данные, позволяет обеспечить централизованное хранение информации, уменьшает затраты времени на ввод данных и их удаление.

Таким образом, можно сделать вывод, что поставленные в начале работы цели и задачи были выполнены, программа соответствует необходимым нормам и готова к своему использованию на практике.

В дальнейшем приложение может быть усовершенствовано в соответствии с новыми идеями или требованиями. Для этого в коде программы созданы специальные функции и переменные который при желании можно реализовать в новый функционал для данного приложения. Поддержка кода и обновления приложения при необходимости будет занимать меньше времени за счет удобного и понятного кода приложения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] [Электронный ресурс]. – Электронные данные. - Режим доступа:    <https://en.wikipedia.org/wiki/Java>

[2] Документация по архитектуре клиент-сервер [Электронный ресурс] – Режим доступа:   
<https://www.youtube.com/watch?v=b-gJxonPcok&list=LL&index=6>

https://www.youtube.com/watch?v=Q61ZrmOAsrs&list=LL&index=5

[3] Комличенко, В.Н. Компьютерные сети:лаб. практикум, Минск: БГУИР, 2012.

[4] Живицкая, Е.Н., Комаровский А.О., Швед О.И. Системный анализ и проектирование информационных систем. Лабораторный практикум: учеб-метод. Пособие – Минск: БГУИР, 2011.

 [5] [Электронный ресурс]. – Электронные данные. - Режим доступа:   <http://www.ru.wikipedia.org>;

[7] [Электронный ресурс]. – Электронные данные. - Режим доступа: http://qt-doc.ru - “Модель "клиент-сервер";

[8] [Электронный ресурс]. – Электронные данные. - Режим доступа: https://openjfx.io/

[9] [Электронный ресурс]. – Электронные данные. - Режим доступа: https://www.mysql.com/products/workbench/

[10] [Электронный ресурс]. – Электронные данные. - Режим доступа:   <https://valor-software.com/ngx-bootstrap/#/>

[11] [Электронный ресурс]. – Электронные данные. - Режим доступа:   <https://fontawesome.com/>;

[12] [Электронный ресурс]. – Электронные данные. - Режим доступа:   <https://getbootstrap.com/>;

[13] [Электронный ресурс]. – Электронные данные. - Режим доступа:   <https://www.mysql.com/>;

# ПРИЛОЖЕНИЕ A

**(обязательное)**

**Функциональная модель процессов функционирования системы управления туристической фирмы**

# 

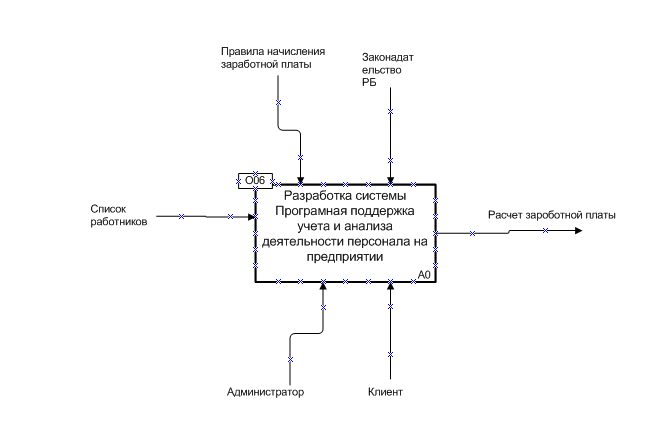


Рисунок А.1 - Контекстная диаграмма функциональной модели

Продолжение приложения А

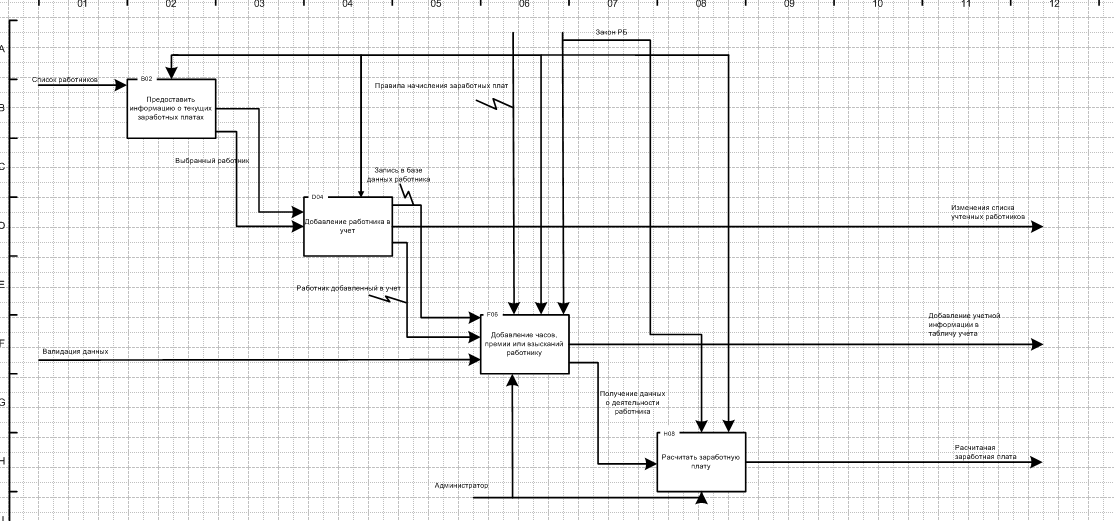


Рисунок А.2 – Декомпозиция контекстной диаграммы верхнего уровня

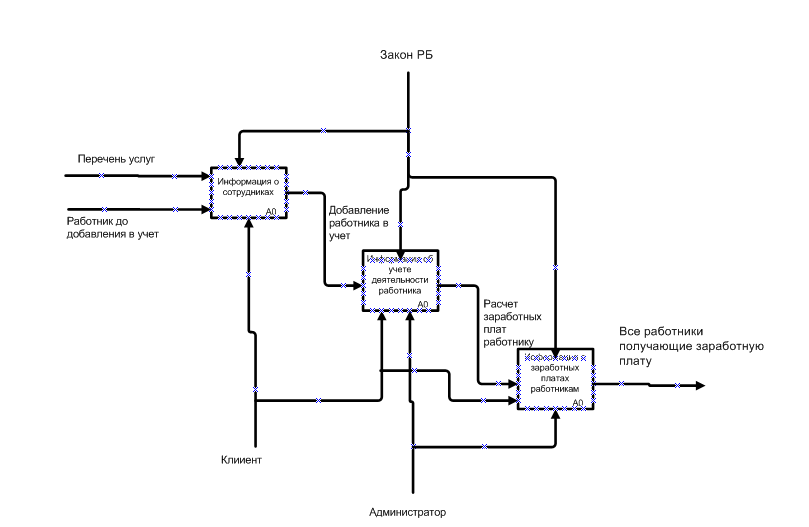


Рисунок А.3 - Декомпозиция процесса туристической фирмы

# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Информационная модель программной поддержки учета и анализа деятельности персонала на предприятии**

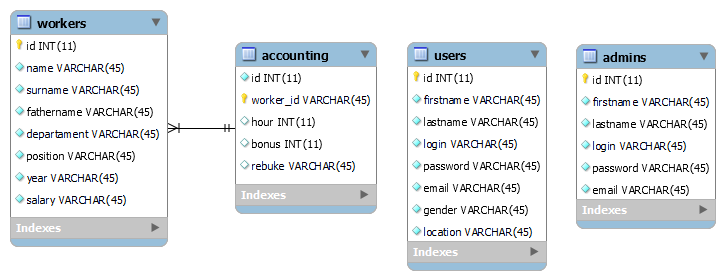


Рисунок Б.1 - Информационная модель системы управление туристической фирмы

# ПРИЛОЖЕНИЕ В

**(обязательное)**

**Диаграмма вариантов использования программной поддержки учета и анализа деятельности персонала на предприятии**

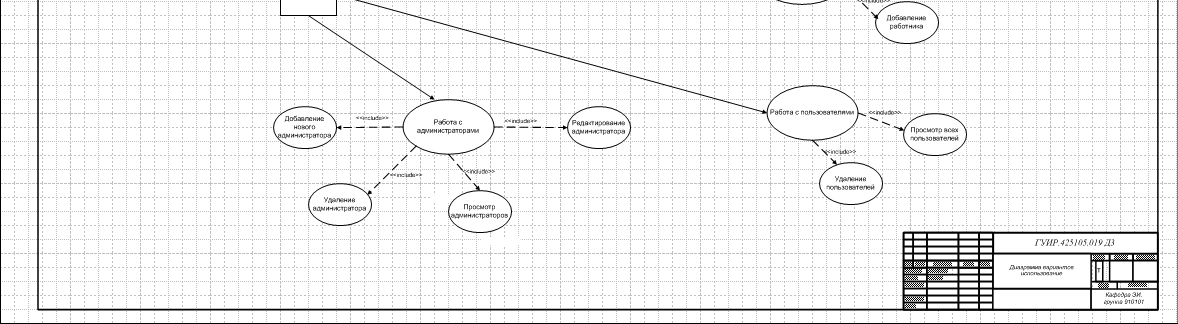
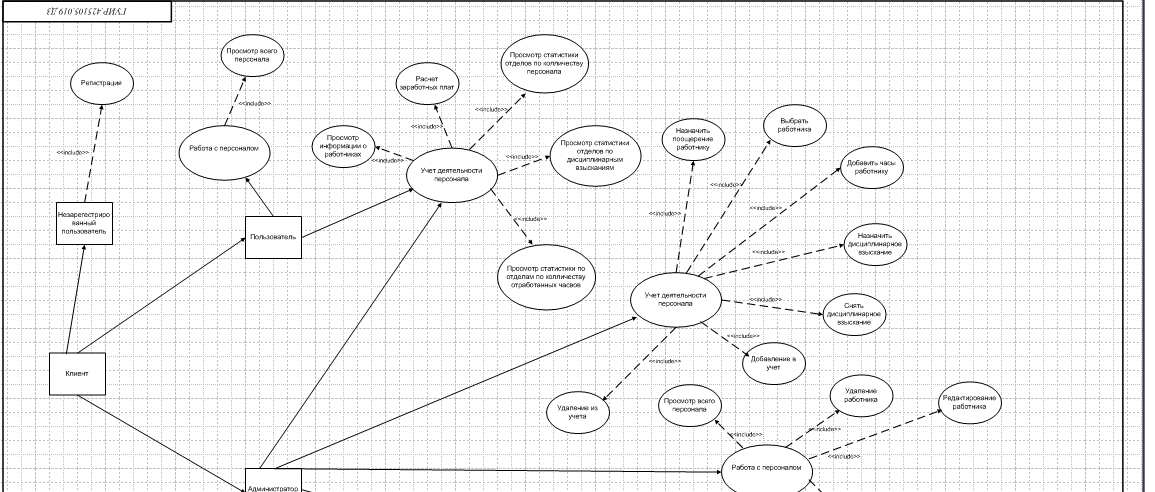


Рисунок В.1 Диаграмма вариантов использования.

# ПРИЛОЖЕНИЕ Д

**(обязательное)**

**Диаграмма последовательностей**

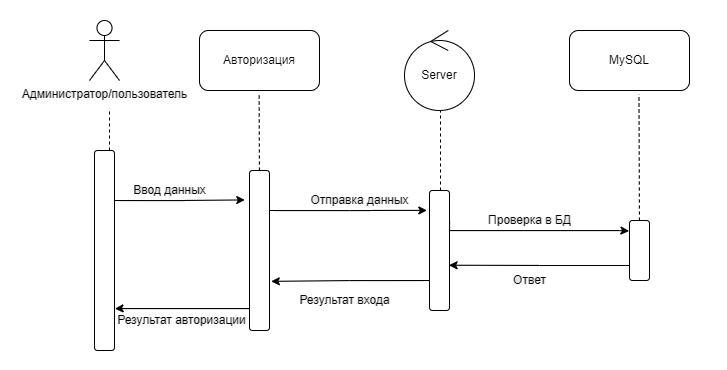


Рисунок Д.1 – Диаграмма последовательностей регистрации пользователей

# ПРИЛОЖЕНИЕ Г

**(обязательное)**

**Диаграмма состояний системы учеты и анализа деятельности персонала на предприятии**

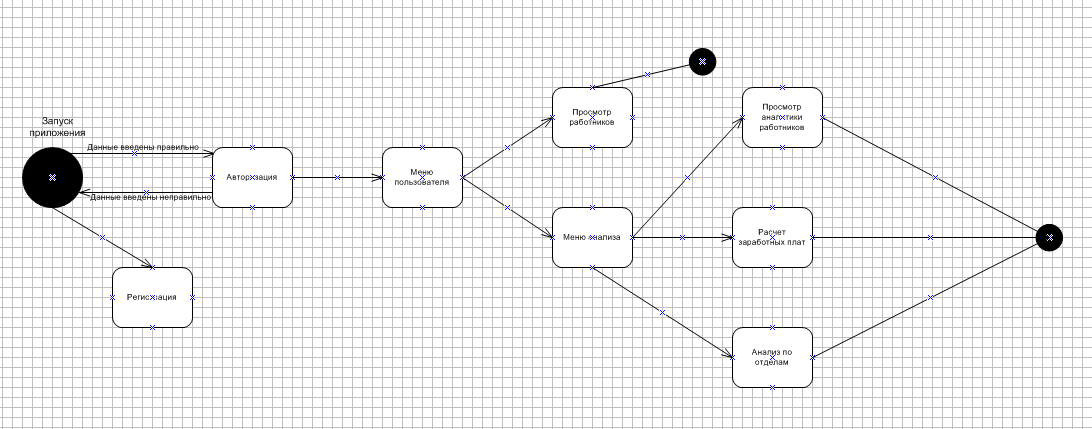


Рисунок Г.1 – Диаграмма состояний пользовательского приложения.

# ПРИЛОЖЕНИЕ Е

**(обязательное)**

**Диаграмма классов программной поддержки учета и анализа деятельности персонала на предприятии**

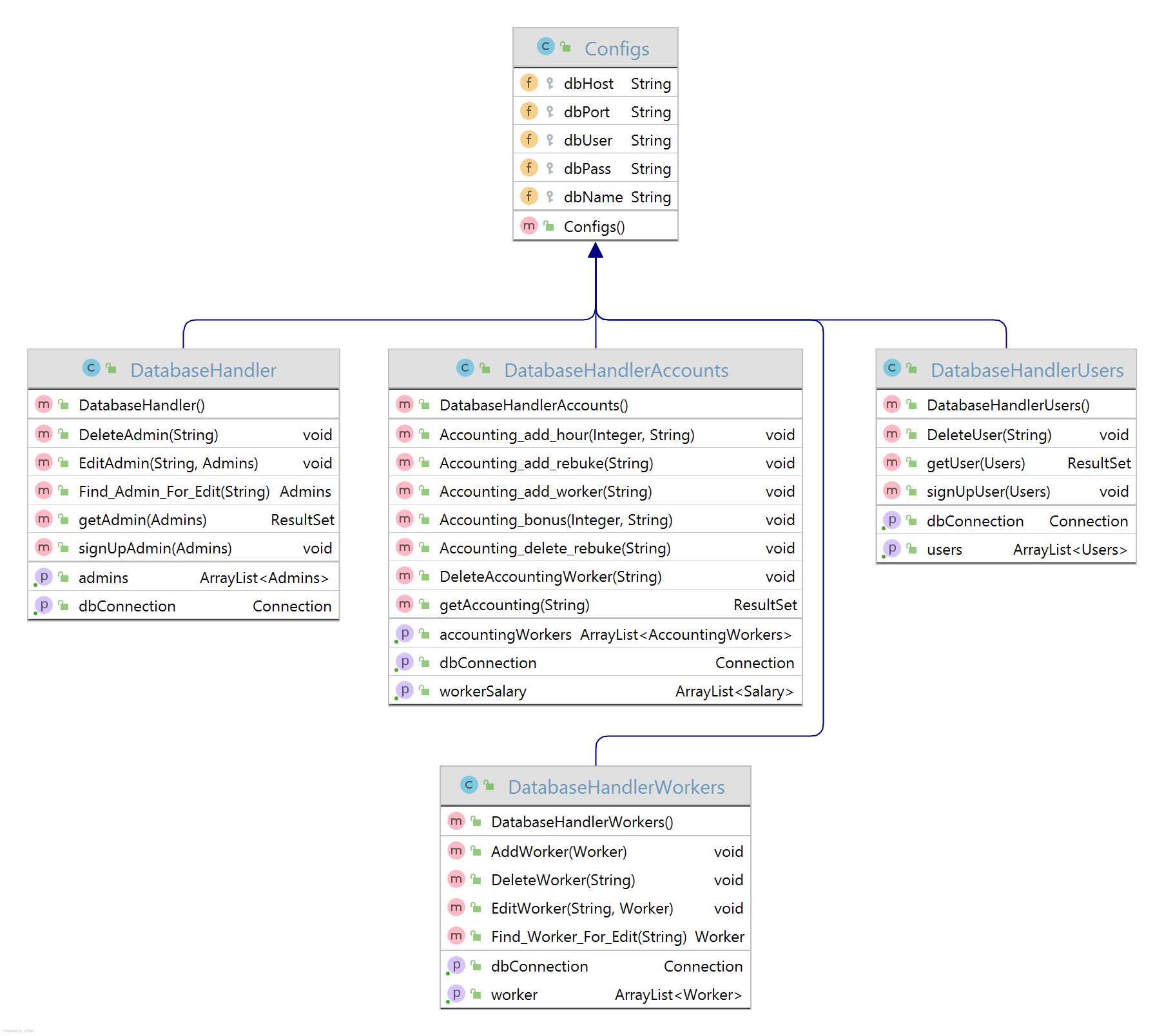


Рисунок Е.2 – Диаграмма классов пакета DatabaseHandler

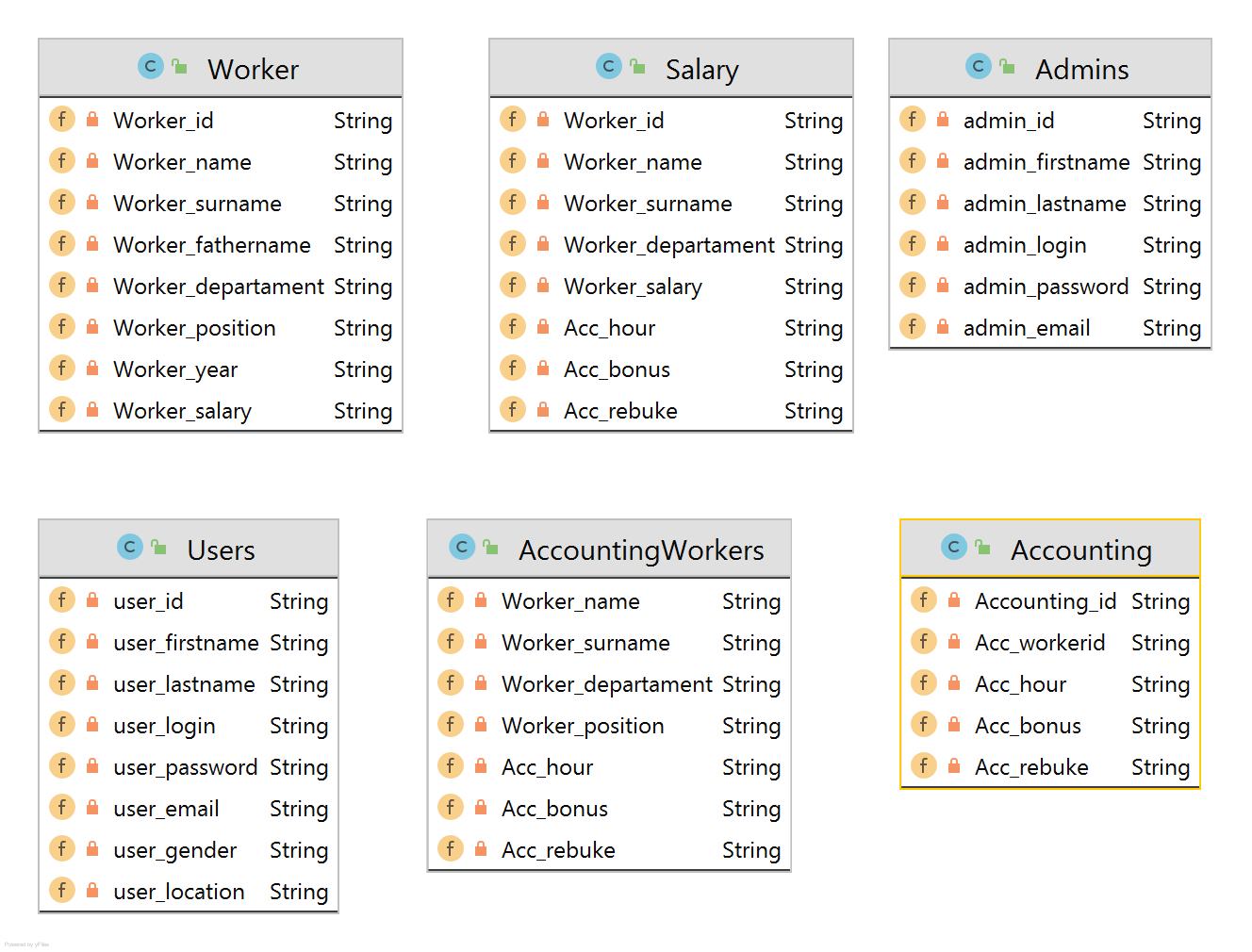


Рисунок Е.3 – Диаграмма классов пакета Clasess

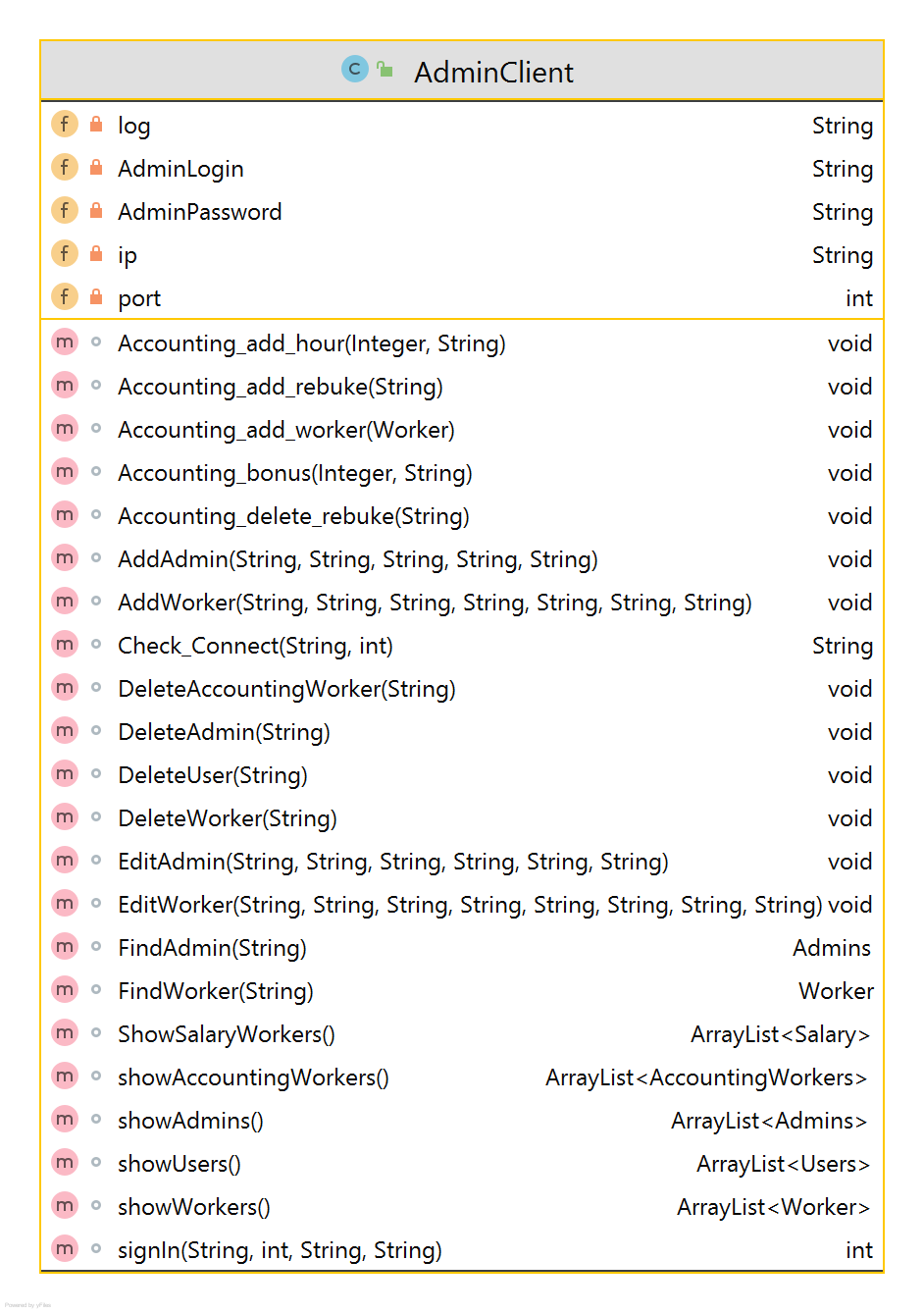
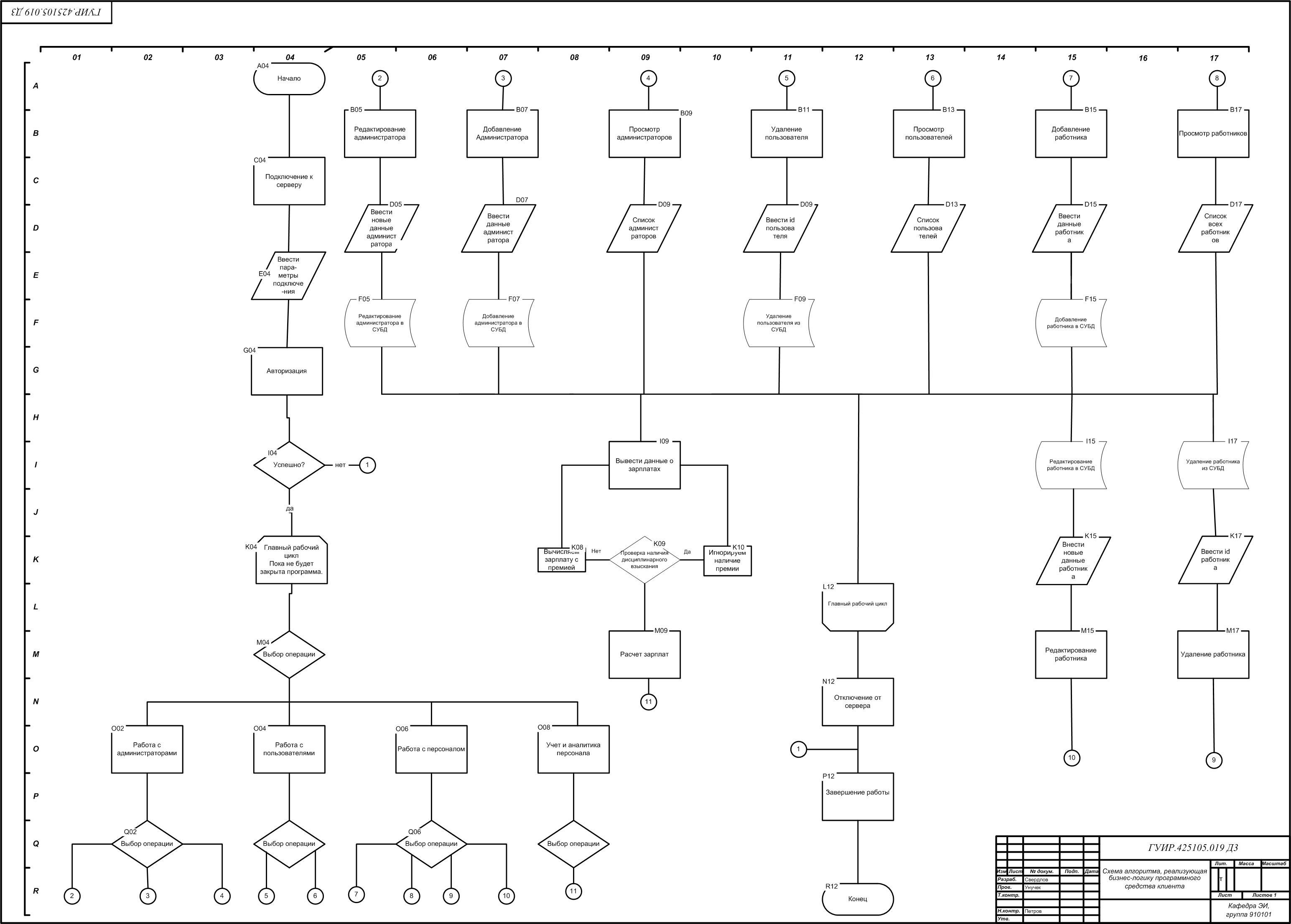


Рисунок Е.3 – Диаграмма класса пакета AdminClient



Рисунок Е.4 – Диаграмма класса пакета Const

Рисунок Е.5 – Продолжение блок-схемы работы программы



## **ПРИЛОЖЕНИЕ Ж**

**(обязательное)**

Скрипт генерации БД

CREATE TABLE `admins` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`firstname` varchar(45) NOT NULL,

`lastname` varchar(45) NOT NULL,

`login` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`email` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=17 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

CREATE TABLE `accounts` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`worker\_id` varchar(45) NOT NULL,

`hour` int(11) DEFAULT NULL,

`bonus` int(11) DEFAULT NULL,

`rebuke` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

CREATE TABLE `users` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`firstname` varchar(45) NOT NULL,

`lastname` varchar(45) NOT NULL,

`login` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`email` varchar(45) NOT NULL,

`gender` varchar(45) NOT NULL,

`location` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

CREATE TABLE `workers` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`surname` varchar(45) NOT NULL,

`fathername` varchar(45) NOT NULL,

`departament` varchar(45) NOT NULL,

`position` varchar(45) NOT NULL,

`year` varchar(45) NOT NULL,

`salary` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=36 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

## **ПРИЛОЖЕНИЕ З**

**(обязательное)**

Код анализа по отделам

RELOAD\_PIECHART\_REBOK.setOnAction(event->{  
 AdminClient adminClient2 = new AdminClient();  
 ArrayList<AccountingWorkers> Workers2 = adminClient2.showAccountingWorkers();  
 int dev,sell,add,log,man;  
 dev=sell=add=log=man = 0;  
 for(AccountingWorkers p : Workers2){  
 if(p.getWorker\_departament().length()==10 && p.getAcc\_rebuke().length()==4){  
 dev++;  
 }  
 if(p.getWorker\_departament().length() == 6&& p.getAcc\_rebuke().length()==4){  
 sell++;  
 }  
 if(p.getWorker\_departament().length()==8&& p.getAcc\_rebuke().length()==4){  
 add++;  
 }  
 if(p.getWorker\_departament().length()==9&& p.getAcc\_rebuke().length()==4){  
 log++;  
 }  
 if(p.getWorker\_departament().length()==16&& p.getAcc\_rebuke().length()==4){  
 man++;  
 }  
 }  
 DEP\_11.setText("Отдел разработки: "+dev + " выговоров");  
 DEP\_21.setText("Отдел Продаж: "+sell+" выговоров");  
 DEP\_31.setText("Отдел Рекламмы: "+add+" выговоров");  
 DEP\_41.setText("Отдел Логистики: "+log+" выговоров");  
 DEP\_51.setText("Отдел Производственный: "+man+" выговоров");  
  
 ObservableList<PieChart.Data> piechartdata =  
 FXCollections.*observableArrayList*(  
 new PieChart.Data("Разработки",dev),  
 new PieChart.Data("Продаж",sell),  
 new PieChart.Data("Рекламмы",add),  
 new PieChart.Data("Логистики",log),  
 new PieChart.Data("Производственный",man));  
 REBUKE\_PIECHART.getData().clear();  
 REBUKE\_PIECHART.getData().addAll(piechartdata);  
});  
  
  
HOUR\_PIECHART\_BUTTON.setOnAction(event->{  
 AdminClient adminClient2 = new AdminClient();  
 ArrayList<AccountingWorkers> Workers2 = adminClient2.showAccountingWorkers();  
 int dev,sell,add,log,man;  
 dev=sell=add=log=man = 0;  
 for(AccountingWorkers p : Workers2){  
 if(p.getWorker\_departament().length()==10){  
 dev+=Integer.*parseInt*(p.getAcc\_hour());  
 }  
 if(p.getWorker\_departament().length() == 6){  
 sell+=Integer.*parseInt*(p.getAcc\_hour());  
 }  
 if(p.getWorker\_departament().length()==8){  
 add+=Integer.*parseInt*(p.getAcc\_hour());  
 }  
 if(p.getWorker\_departament().length()==9){  
 log+=Integer.*parseInt*(p.getAcc\_hour());  
 }  
 if(p.getWorker\_departament().length()==16){  
 man+=Integer.*parseInt*(p.getAcc\_hour());  
 }  
 }  
 DEP\_313.setText("Отдел разработки: "+dev + " часов");  
 DEP\_32.setText("Отдел Продаж: "+sell+" часов");  
 DEP\_33.setText("Отдел Рекламмы: "+add+" часов");  
 DEP\_34.setText("Отдел Логистики: "+log+" часов");  
 DEP\_35.setText("Отдел Производственный: "+man+" часов");  
  
 ObservableList<PieChart.Data> piechartdata =  
 FXCollections.*observableArrayList*(  
 new PieChart.Data("Разработки",dev),  
 new PieChart.Data("Продаж",sell),  
 new PieChart.Data("Рекламмы",add),  
 new PieChart.Data("Логистики",log),  
 new PieChart.Data("Производственный",man));  
 HOUR\_PIECHART.getData().clear();  
 HOUR\_PIECHART.getData().addAll(piechartdata);  
});  
  
ANALITIC\_RELOAD\_BUTTON.setOnAction(event->{  
 AdminClient adminClient2 = new AdminClient();  
 ArrayList<Worker> Workers2 = adminClient2.showWorkers();  
 int dev,sell,add,log,man;  
 dev=sell=add=log=man = 0;  
 for(Worker p : Workers2){  
 if(p.getWorker\_departament().length()==10){  
 dev++;  
 }  
 if(p.getWorker\_departament().length() == 6){  
 sell++;  
 }  
 if(p.getWorker\_departament().length()==8){  
 add++;  
 }  
 if(p.getWorker\_departament().length()==9){  
 log++;  
 }  
 if(p.getWorker\_departament().length()==16){  
 man++;  
 }  
 }  
 DEP\_1.setText("Отдел разработки: "+dev);  
 DEP\_2.setText("Отдел Продаж: "+sell);  
 DEP\_3.setText("Отдел Рекламмы: "+add);  
 DEP\_4.setText("Отдел Логистики: "+log);  
 DEP\_5.setText("Отдел Производственный: "+man);  
  
 ObservableList<PieChart.Data> piechartdata =  
 FXCollections.*observableArrayList*(  
 new PieChart.Data("Разработки",dev),  
 new PieChart.Data("Продаж",sell),  
 new PieChart.Data("Рекламмы",add),  
 new PieChart.Data("Логистики",log),  
 new PieChart.Data("Производственный",man));  
 DEPARTAMENT\_DIAGRAM.getData().clear();  
 DEPARTAMENT\_DIAGRAM.getData().addAll(piechartdata);  
  
});

Расчет зароботной платы

ArrayList<Salary> ShowSalaryWorkers(){  
 try(Socket clientSocket = new Socket("127.0.0.1",8081);  
 BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));  
 BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream())))  
 {  
 log = "Вы подключились к серверу";  
 writer.write("ShowWorkersSalary");  
 writer.newLine();  
 writer.flush();  
 ArrayList<Salary> arrayList = new ArrayList<Salary>();  
 ArrayList<Salary> salaryworkers = new ArrayList<>();  
 try {  
 ObjectInputStream objectInputStream = new ObjectInputStream(clientSocket.getInputStream());  
 try {  
 Object object = objectInputStream.readObject();  
 arrayList = (ArrayList<Salary>) object;  
 for(Salary p : arrayList){  
 int kolvo = p.getAcc\_rebuke().length();  
 if(kolvo == 3){  
 double sal = 0;  
 Integer bonus = Integer.*parseInt*(p.getAcc\_bonus());  
 double percentbonus = bonus\*0.01;  
 double persent = (percentbonus+1.0);  
 sal = Integer.*parseInt*(p.getWorker\_salary()) \* Integer.*parseInt*(p.getAcc\_hour()) \* persent;  
 p.setWorker\_salary(String.*valueOf*(sal));  
 }  
 if(kolvo==4){  
 double sal = 0;  
 sal = Integer.*parseInt*(p.getWorker\_salary()) \* Integer.*parseInt*(p.getAcc\_hour());  
 p.setWorker\_salary(String.*valueOf*(sal));  
 }  
 salaryworkers.add(p);  
 }  
  
 return salaryworkers;  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 log = "Не удалось подключится к серверу";  
 }  
 return null;  
}

Просмотр работников и пользователей

showUsersButton.setOnAction(event->{  
 AdminClient admin\_client = new AdminClient();  
 admin\_client.showUsers();  
 ArrayList<Users> userArrayList = admin\_client.showUsers();  
 this.UserArrayList = userArrayList;  
 ObservableList<Users> observableList = FXCollections.*observableArrayList*(admin\_client.showUsers());  
 SHOWUSERTABLE.setItems(observableList);  
 SHOWUSERTABLE.getColumns().get(0).setCellValueFactory(new PropertyValueFactory("user\_id"));  
 SHOWUSERTABLE.getColumns().get(1).setCellValueFactory(new PropertyValueFactory("user\_firstname"));  
 SHOWUSERTABLE.getColumns().get(2).setCellValueFactory(new PropertyValueFactory("user\_lastname"));  
 SHOWUSERTABLE.getColumns().get(3).setCellValueFactory(new PropertyValueFactory("user\_login"));  
 SHOWUSERTABLE.getColumns().get(4).setCellValueFactory(new PropertyValueFactory("user\_password"));  
 SHOWUSERTABLE.getColumns().get(5).setCellValueFactory(new PropertyValueFactory("user\_email"));  
});  
  
ShowWorkersButton.setOnAction(event->{  
 AdminClient admin\_client = new AdminClient();  
 admin\_client.showWorkers();  
 ArrayList<Worker> workerArrayList = admin\_client.showWorkers();  
 this.WorkerArrayList = workerArrayList;  
 ObservableList<Worker> observableList = FXCollections.*observableArrayList*(admin\_client.showWorkers());  
 ShowWorkerTable.setItems(observableList);  
 ShowWorkerTable.getColumns().get(0).setCellValueFactory(new PropertyValueFactory("Worker\_name"));  
 ShowWorkerTable.getColumns().get(1).setCellValueFactory(new PropertyValueFactory("Worker\_surname"));  
 ShowWorkerTable.getColumns().get(2).setCellValueFactory(new PropertyValueFactory("Worker\_fathername"));  
 ShowWorkerTable.getColumns().get(3).setCellValueFactory(new PropertyValueFactory("Worker\_departament"));  
 ShowWorkerTable.getColumns().get(4).setCellValueFactory(new PropertyValueFactory("Worker\_position"));  
 ShowWorkerTable.getColumns().get(5).setCellValueFactory(new PropertyValueFactory("Worker\_salary"));  
 ShowWorkerTable.getColumns().get(6).setCellValueFactory(new PropertyValueFactory("Worker\_year"));  
 ShowWorkerTable.getColumns().get(7).setCellValueFactory(new PropertyValueFactory("Worker\_id"));  
});

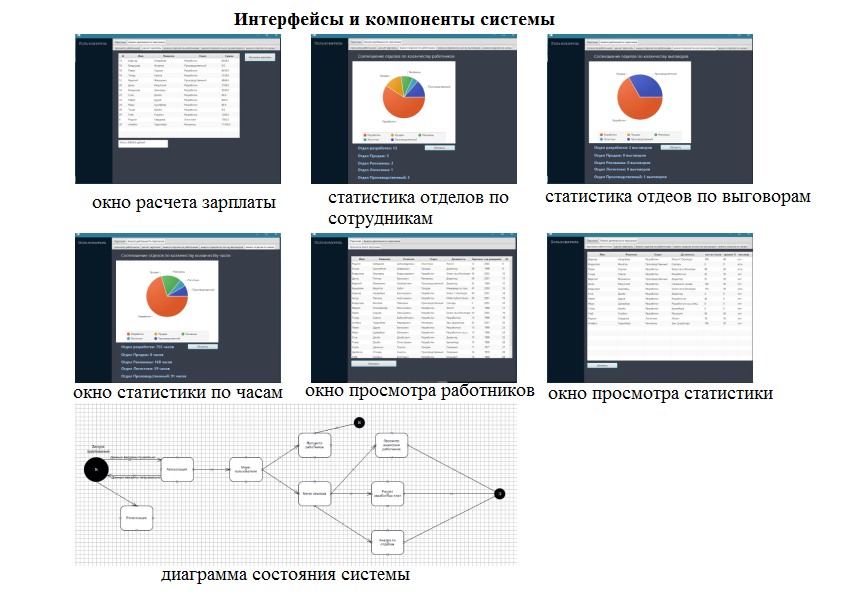
DataBaseHandler класс работы с БД для администраторов

package Database;  
  
import AdminServer.Configs;  
import Clasess.Admins;  
import Const.Const;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.SQLException;  
import java.sql.ResultSet;  
import java.util.ArrayList;  
  
public class DatabaseHandler extends Configs {  
 Connection dbConnection;  
  
 public Connection getDbConnection()  
 throws ClassNotFoundException, SQLException {  
 String connectionString = "jdbc:mysql://" + dbHost + ":"  
 + dbPort + "/" + dbName;  
  
 Class.*forName*("com.mysql.jdbc.Driver");  
  
 dbConnection = DriverManager.*getConnection*(connectionString, dbUser, dbPass);  
 return dbConnection;  
 }  
  
 public void signUpAdmin(Admins admin) {  
 String insert = "INSERT INTO " + Const.*USER\_TABLE* + "(" +  
 Const.*FIRST\_NAME* + "," + Const.*LAST\_NAME* + "," +  
 Const.*LOGIN* + "," + Const.*PASSWORD* + "," +  
 Const.*EMAIL* + ")" +  
 "VALUES(?,?,?,?,?)";  
  
 try {  
 PreparedStatement prSt = getDbConnection().prepareStatement(insert);  
  
 prSt.setString(1, admin.getAdmin\_firstname());  
 prSt.setString(2, admin.getAdmin\_lastname());  
 prSt.setString(3, admin.getAdmin\_login());  
 prSt.setString(4, admin.getAdmin\_password());  
 prSt.setString(5, admin.getAdmin\_email());  
  
  
 prSt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public ResultSet getAdmin(Admins admin) {  
 ResultSet resSet = null;  
  
 String select = "SELECT \* FROM " + Const.*USER\_TABLE* + " WHERE " +  
 Const.*LOGIN* + "=? AND "+Const.*PASSWORD* + "=?";  
 try {  
 PreparedStatement prSt = getDbConnection().prepareStatement(select);  
 prSt.setString(1,admin.getAdmin\_login());  
 prSt.setString(2,admin.getAdmin\_password());  
  
 resSet = prSt.executeQuery();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 return resSet;  
 }  
  
 public ArrayList<Admins> getAdmins() {  
 ArrayList<Admins> products = new ArrayList<Admins>();  
 String select = "SELECT \* FROM " + Const.*USER\_TABLE*;  
 try {  
 PreparedStatement prSt = getDbConnection().prepareStatement(select);  
 ResultSet resSet = prSt.executeQuery();  
 while (resSet.next()) {  
 Admins p = new Admins();  
 p.setAdmin\_firstname(resSet.getString("firstname"));  
 p.setAdmin\_lastname(resSet.getString("lastname"));  
 p.setAdmin\_login(resSet.getString("login"));  
 p.setAdmin\_password(resSet.getString("password"));  
 p.setAdmin\_email(resSet.getString("email"));  
 p.setAdmin\_id(resSet.getString("id"));  
  
 products.add(p);  
 }  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return products;  
 }  
  
 public void DeleteAdmin(String id){  
 String delete = "DELETE FROM " + Const.*USER\_TABLE* + " WHERE " + Const.*ADMIN\_ID* + "='" + id + "'; ";  
 try {  
 PreparedStatement prSt = getDbConnection().prepareStatement(delete);  
 prSt.executeUpdate();  
 System.*out*.println("id принятое на сервер = " + id);  
 System.*out*.println("Админ успешно удален");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public Admins Find\_Admin\_For\_Edit(String id){  
 Admins admin = new Admins();  
 String findAdmin = "SELECT \* FROM " + Const.*USER\_TABLE* + " WHERE "+Const.*ADMIN\_ID* + "=?";  
 try {  
  
 PreparedStatement prSt = getDbConnection().prepareStatement(findAdmin);  
 prSt.setString(1,id);  
 ResultSet resSet = prSt.executeQuery();  
 while (resSet.next()) {  
 Admins p = new Admins();  
 p.setAdmin\_firstname(resSet.getString("firstname"));  
 p.setAdmin\_lastname(resSet.getString("lastname"));  
 p.setAdmin\_login(resSet.getString("login"));  
 p.setAdmin\_password(resSet.getString("password"));  
 p.setAdmin\_email(resSet.getString("email"));  
 p.setAdmin\_id(resSet.getString("id"));  
 admin = p;  
 }  
 System.*out*.println("id принятое на сервер = " + id);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return admin;  
 }  
  
 public void EditAdmin(String id, Admins admin){  
 try{  
 String update = "UPDATE " + Const.*USER\_TABLE* +  
 " SET " + Const.*FIRST\_NAME* + "=?, " + Const.*LAST\_NAME* + "=?, "  
 + Const.*LOGIN* + "=?, " + Const.*PASSWORD* + "=?, " + Const.*EMAIL* + "=? " +  
 " WHERE " + Const.*ADMIN\_ID* + "=?";  
 PreparedStatement preparedStatement = getDbConnection().prepareStatement(update);  
  
 //System.out.println(admin.getAdmin\_email());  
 //int port\_ = Integer.parseInt(port.getText());  
 //Integer intid = Integer.parseInt(id);  
  
 System.*out*.println(id + " " + admin.toString());  
 preparedStatement.setString(1, admin.getAdmin\_firstname());  
 preparedStatement.setString(2, admin.getAdmin\_lastname());  
 preparedStatement.setString(3, admin.getAdmin\_login());  
 preparedStatement.setString(4, admin.getAdmin\_password());  
 preparedStatement.setString(5, admin.getAdmin\_email());  
 preparedStatement.setInt(6, Integer.*parseInt*(id));  
 System.*out*.println("Admin otredacktirovan");  
  
 preparedStatement.executeUpdate();  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
  
 }  
  
}

## **ПРИЛОЖЕНИЕ И**

**(обязательное)**

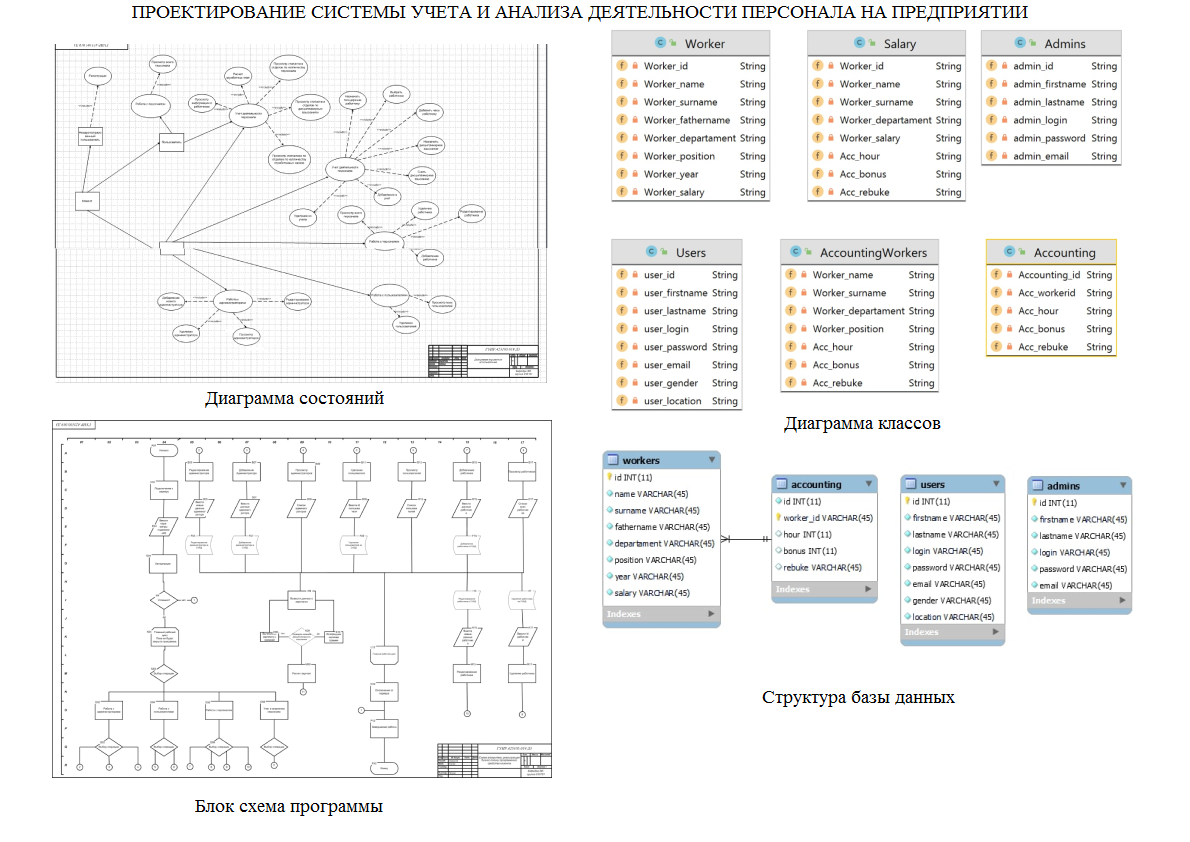
**Плакат 1**



## **ПРИЛОЖЕНИЕ К**

**(обязательное)**

**Плакат 2**



## **ПРИЛОЖЕНИЕ Л**

**(обязательное)**

**Плакат 3**

