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  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Лыщик |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Стистема расчета зАработной платы»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 073601  Падневич Владимир Владимирович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_.\_\_.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2022

# **Реферат**

Данная курсовая работа рассматривает процесс расчета заработной платы. По ходу выполнения работы будет проведено исследование процесса расчета заработной платы сотрудников, определены процессы, для которых возможна автоматизация.

Объектом исследования является процесс расчета заработной платы.

Предметом исследования выступают особенности протекая процесса расчета заработной платы внутри фирм, которые не могут позволить услуги штатного бухгалтера.

Целью выполнения курсового проекта является повышение эффективности деятельности бухгалтера, в задачи которого входит расчет заработной платы сотрудников.

По итогам исследования можно отметить что было детально изучена система расчета заработной платы. Разработанное программное средство позволяет автоматизировать процессы расчета заработной платы и ведения учета информации о сотрудниках.

Система расчета зработной платы : Курсовая работа / В.В. Падневич. – Минск : БГУИР, 2022, – п.з. – 44 с., чертежей (плакатов) – 5л. формата А4

# **Содержание**

[Введение 6](#_Toc120979108)

[1 Анализ и моделирование предметной области программного средства 7](#_Toc120979109)

[1.1 Описание предметной области 7](#_Toc120979110)

[1.2 Разработка функциональной модели предметной области. 8](#_Toc120979111)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 14](#_Toc120979112)

[1.4 Разработка информационной модели предметной области 15](#_Toc120979113)

[1.5 UML-модели представления программного средства и их описание 17](#_Toc120979114)

[2 Проектирование и конструирование программного средства 23](#_Toc120979115)

[2.1 Постановка задачи. 23](#_Toc120979116)

[2.2 Архитектурные решения 23](#_Toc120979117)

[2.3 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 24](#_Toc120979118)

[2.4 Проектирование пользовательского интерфейса 26](#_Toc120979119)

[2.5 Обоснование выбора компонентов и технологий для реализации программного средства 27](#_Toc120979120)

[3 Тестирование и проверка работоспособности программного средства 29](#_Toc120979121)

[4 Инструкция по развертыванию приложения и сквозной тестовый пример 30](#_Toc120979122)

[Заключение 36](#_Toc120979123)

[Список использованных источников 37](#_Toc120979124)

[Приложения А Отчет о проверке на заимствования в системе «Антиплагиат» 38](#_Toc120979125)

[Приложение Б Листинг кода алгоритмов, реализующих основную бизнес-логику 39](#_Toc120979126)

[Приложение В Листинг скрипта генерации базы данных 42](#_Toc120979127)

# **Введение**

Расчет заработной платы – сложный процесс, требующий значительных усилий со стороны финансового персонала компании. Достаточно вспомнить, сколько времени и сил уходит на то, чтобы правильно рассчитать все выплаты и удержания в структуре заработной платы сотрудников, точно определить величину налоговых вычетов, начислить и уплатить социальные взносы, своевременно подготовить и сдать обязательную квартальную и годовую отчетность. Кроме того, на бухгалтера ложится дополнительная нагрузка по отслеживанию и правильному применению всех законодательных нововведений, которые непосредственно влияют на расчет заработной платы, будь то новые алгоритмы расчетов или изменившиеся налоговые ставки.

Этот программный продукт значительно облегчит работу сотрудников отдела бухгалтерии. Он позволит собрать информацию о сотрудниках компании в единое целое, сделает удобнее запись, сортировку и анализ данных.

Целью курсовой работы является повышение эффективности процесса расчета заработной платы.

Поставленная цель потребовала решения следующих задач:

* изучить процесс расчета заработной платы;
* разработать систему расчета заработной платы;
* протестировать программу;
* описать разработанное приложение.

Объектом исследования является процесс учета выплат заработной платы сотрудникам.

# **Анализ и моделирование предметной области программного средства**

## **Описание предметной области**

Заработная плата является одним из квалифицирующих признаков трудовых отношений, отличающих их от иных правоотношений и, прежде всего, от гражданско-правовых. Данное разграничение имеет большое практическое значение для квалификации правоотношений как трудовых в условиях, когда работодатели оформляют фактически трудовые отношения гражданско-правовым договором с целью снизить платежи по единому социальному налогу и уйти от гарантий. По трудовому договору вознаграждение за труд выплачивается в форме заработной платы.

Система оплаты труда включает как способ установления соотношения между мерой труда и мерой вознаграждения за него, на основании которого строится порядок исчисления заработка работника (форма заработной платы), так и конкретные размеры тарифных ставок, окладов (должностных окладов). В систему оплаты труда включаются также условия, порядок выплаты и размеры доплат и надбавок компенсационного характера, условия, порядок выплаты и размеры доплат и надбавок стимулирующего характера, премий. В соответствии с законодательством организация самостоятельно разрабатывает и утверждает формы и системы оплаты труда работников, как числящихся в штате, так и привлекаемых со стороны для выполнения работ по договорам. Система оплаты труда, принятая на конкретном предприятии отражается в коллективном договоре, соглашении любого вида либо локальном нормативном акте.

Для начисления заработка работникам необходимо вести учёт их численности, проработанного и не проработанного времени, выполненной или невыполненной работы и выработки. Общая численность работников, состоящих в штате предприятия, составляет списочную численность. Лица, работающие на предприятии по трудовым соглашениям, составляют вне списочный состав.

Среди наиболее трудоемких участков бухгалтерского учета особое место занимают расчеты с персоналом по оплате труда. Для работы на данном участке необходимы высокая точность, аккуратность, максимум внимания и, безусловно, своевременное выполнение всех расчетных операций. В роли объектов учета могут выступать десятки, сотни и даже тысячи человек, по каждому из которых нужно учитывать и обрабатывать достаточно большие объемы данных.

Особенно затрудняют работу бухгалтеров частые изменения правил расчета подоходного налога с физических лиц, где в качестве объекта налогообложения выступает совокупный доход сотрудников. Во-первых, бухгалтеру, нужно учесть все доходы работника, относящиеся к налогооблагаемому периоду. Во-вторых, он должен правильно определить и исключить из налогооблагаемой базы все необходимые вычеты, что требует знания множества нюансов. В-третьих, ему нужно корректно отразить в бухучете начисление, удержание и перечисление данного налога в бюджет.

Оплата труда может быть сдельной или повременной. Рассмотрим основные нюансы применения данных систем.

При сдельной форме общий размер заработанных средств рассчитывается, исходя из объема осуществленной работы или выпущенной продукции.

При повременной оплате труда зарплата определяется от количества фактически отработанных часов, вне зависимости от того, сколько продукции было произведено и какой объем работы был выполнен.

## **Разработка функциональной модели предметной области.**

На рисунке 1.1 приведена контекстная диаграмма процесс расчета заработной платы.

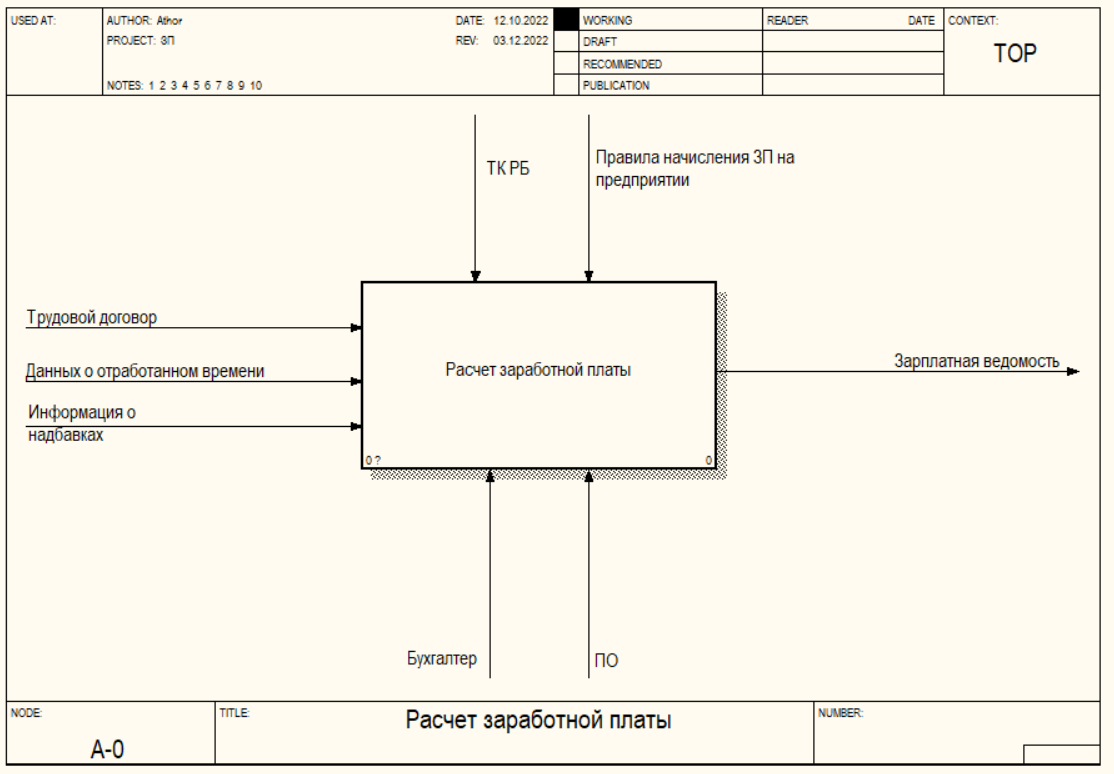


Рисунок 1.1 – Контекстная диаграмма

На рисунке 1.2 представлена декомпозиция контекстной диаграммы.

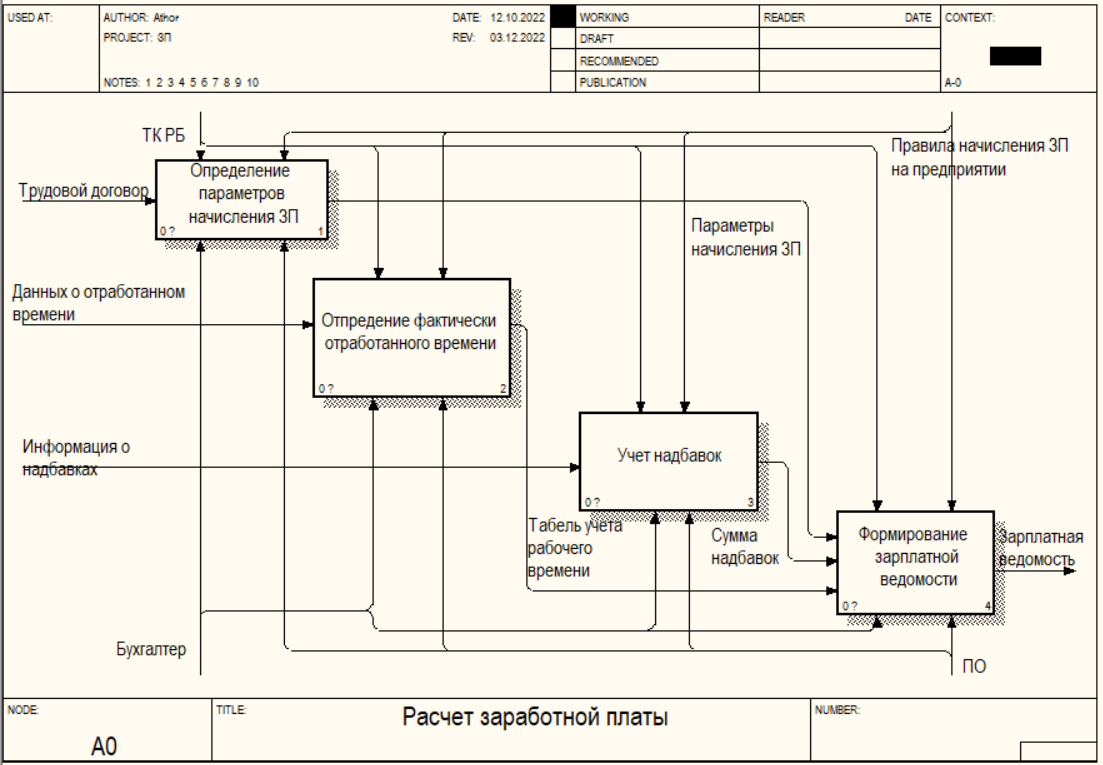


Рисунок 1.2 – Декомпозиция процесса расчета заработной платы

Определение параметров расчета заработной платы состоит из выполнения следующих процессов(рисунок 1.3):

* определение способа начисления заработной платы;
* просмотр графика работы;
* определение размера оклада.

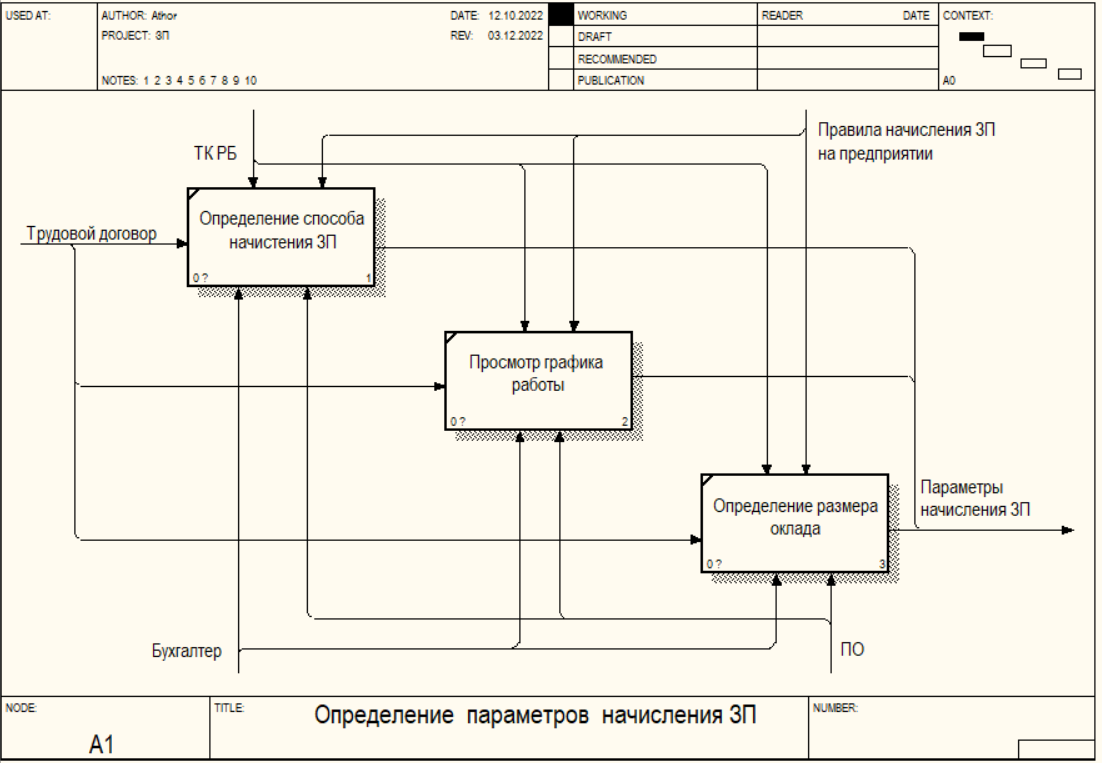


Рисунок 1.3 – Процесс «Определение параметров начисления ЗП»

Определение фактически отработанного времени состоит из выполнения проверки данных о фактически отработанном времени, учета неотработанного времени, включенного в рабочее время и формирования табеля учета рабочего времени(рисунок 1.4).

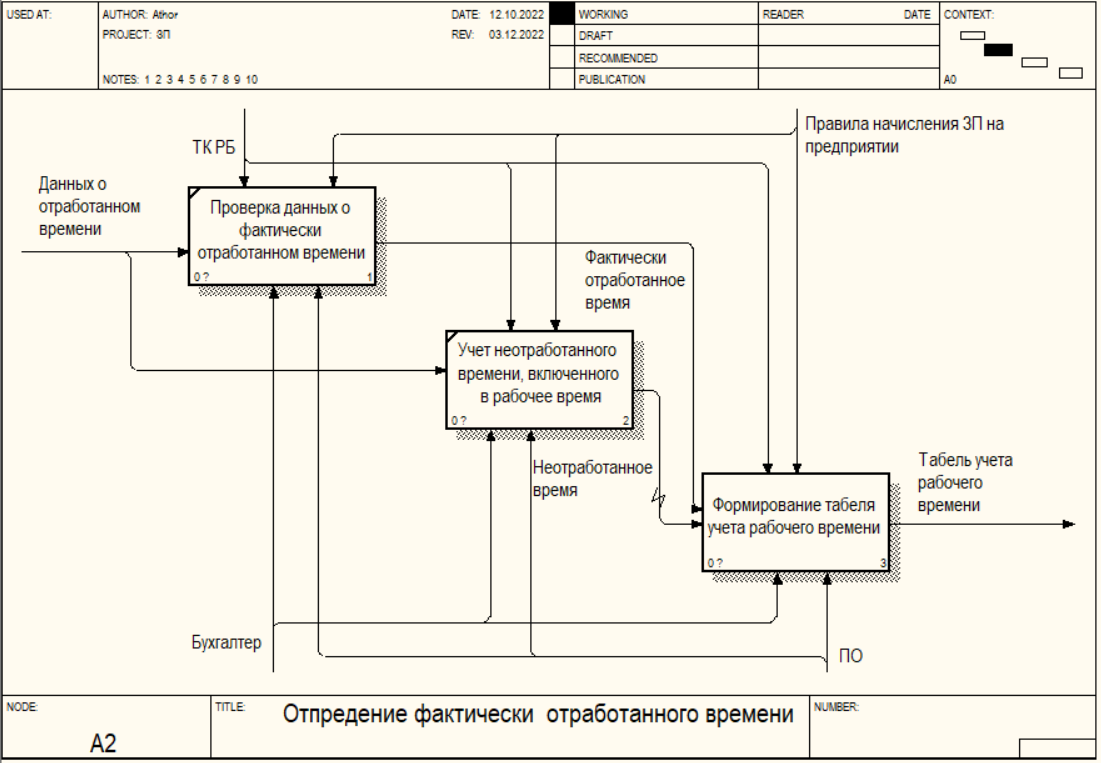


Рисунок 1.4 – Декомпозиция процесса «Определение фактически отработанного времени»

Декомпозиция процесса учета надбавок приведена на рисунке 1.5. При учете надбавок требуется проверить основания для начисления надбавок, определить сумму, на которую будет начислена надбавка, определить период начисления данной надбавки и на основе всего перечисленного рассчитать итоговую сумму заработной платы сотрудника.

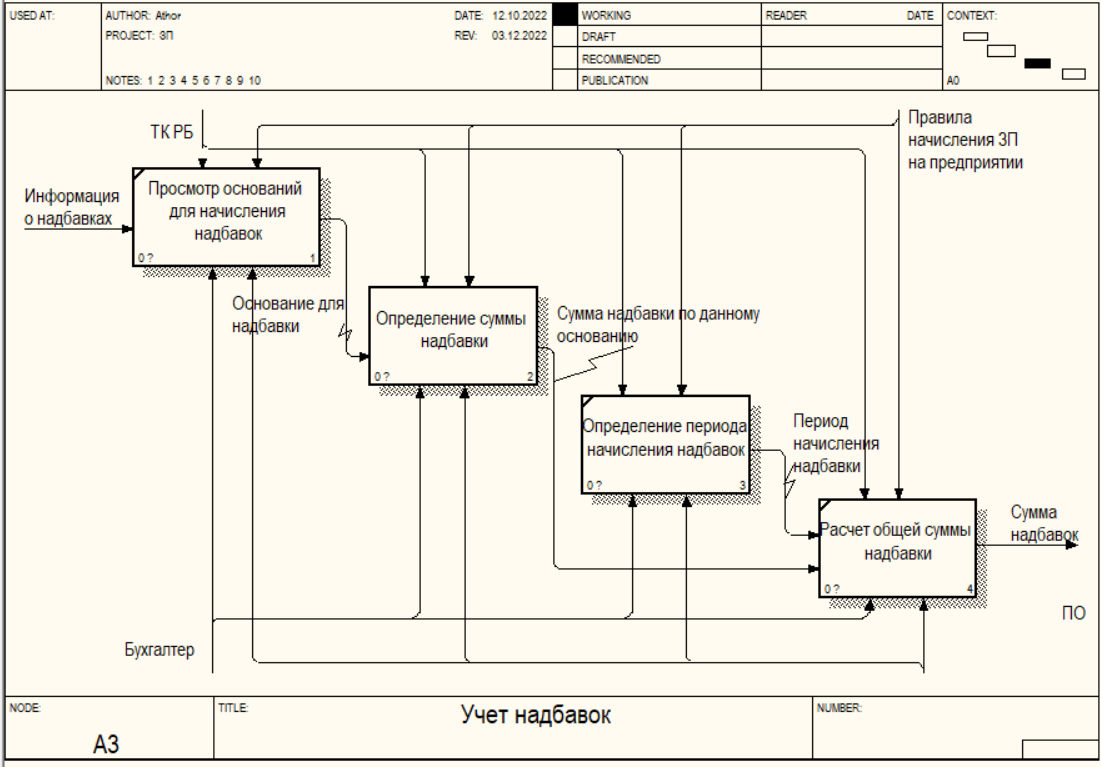


Рисунок 1.5 – Диаграмма процесса учета надбавок

Процесс формирования зарплатной ведомости состоит из выполнения расчета оклада и расчета общей суммы заработной платы(рисунок 1.6). Для формирования зарплатной ведомости требуется наличие табеля учета рабочего времени, рассчитанной суммы надбавок и определенных параметров начисления заработной платы.

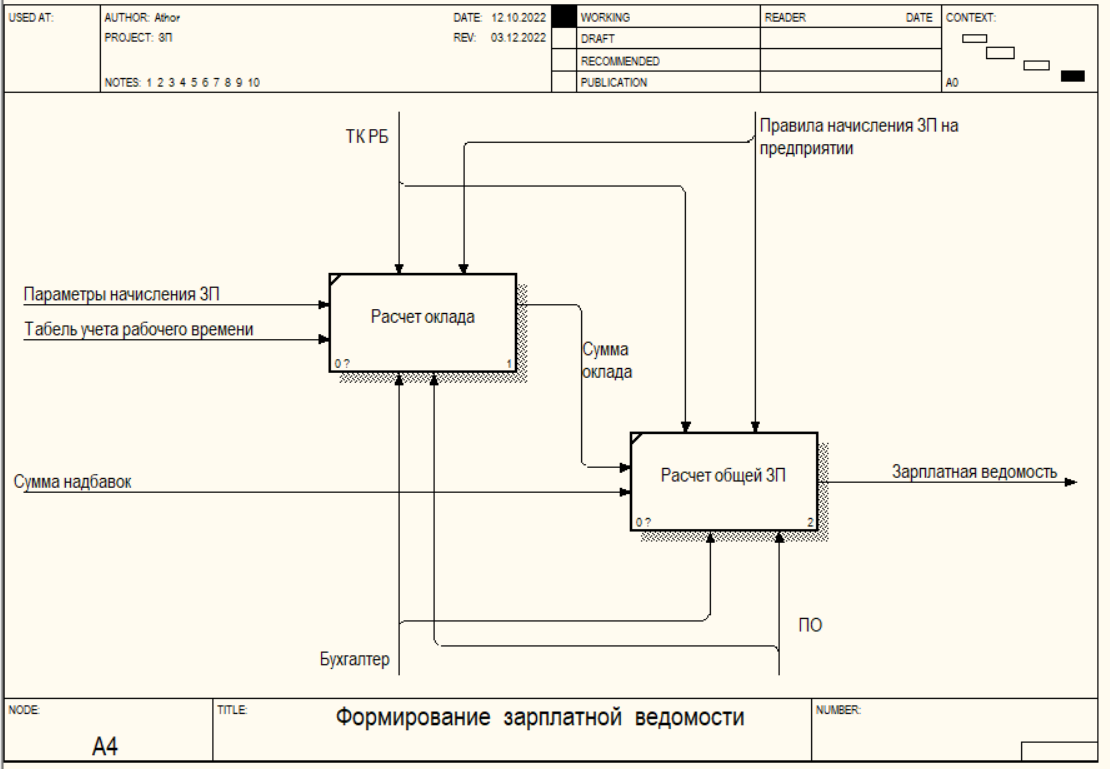


Рисунок 1.6 – Процесс формирования зарплатной ведомости

Процесс формирования табеля учета рабочего времени не полностью раскрыт. Данный процесс состоит из выполнения(рисунок 1.7):

* заполнение отработанных дней;
* заполнение кодов причин отсутствия;
* окончательное заполнение.

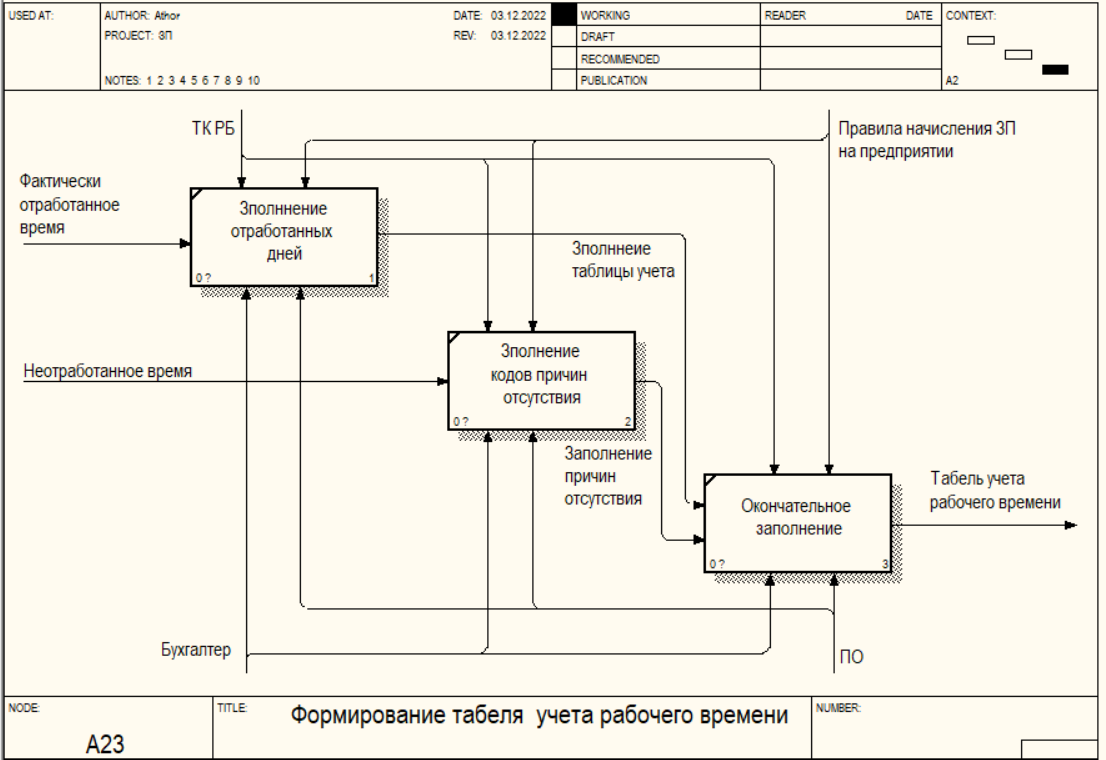


Рисунок 1.7 – Декомпозиция процесса формирования табеля учета рабочего времени

Таким образом, было проведено функциональное моделирование процесса расчета заработной платы. Достигнуто понимание функционирования процесса в действительности, что позволяет приступить к дальнейшей разработке программного средства.

## **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Приложение должно быть выполнено в архитектуре клиент-сервер с многопоточным сервером с организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

Серверное приложение может быть реализовано в виде GUI-приложения. Настройки сервера должны меняться без изменения исходного кода (аргументы командной строки, конфиг-файлы и т.д.). Клиентское приложение – оконное приложение с использованием стандартных библиотек пользовательского интерфейса.

Бизнес-логика системы должна быть реализована только на серверной части приложения. На сервере должна быть предусмотрена возможность параллельной обработки запросов. Доступ к данным в СУБД должен осуществляться через драйвер, предоставляемый производителем СУБД или через использование специальных технологий.

В приложении должны быть определены следующие роли:

* бухгалтер;
* администратор;
* сотрудник.

Администратор регистрирует сотрудников, дает им право заходить в личный кабинет, редактирует информацию о сотрудниках, может также уволить сотрудников. Функциональные возможности даминистратора:

1. Просмотр всех сотрудников предприятия.
2. Добавление/Редактирование/Удаление сотрудников.
3. Просмотр сотрудников, которые хотят уволиться, разрешает им увольняться или нет.
4. Имеет возможность записать в файл данные о сотрудниках предприятия.
5. Статистика (просмотр посещения предприятия).

Бухгалтер рассчитывает з/п сотрудников, может редактировать з/п. Функциональные возможности бухгалтера:

1. Просмотр всех з/п сотрудников.
2. Расчёт з/п сотрудниках (может начислить премию, так же указывает налог).
3. Проверяет рассчитана ли сотруднику зарплата, и если да, то сотрудник просматривает расчётный лист (то есть, даёт возможность просмотреть их если выставлена зарплата).

Сотрудник входит в систему только если его зарегистрировали, может просматривать информацию о компании, редактировать свои данные и уволиться. Функциональные возможности сотрудника:

1. Просмотр всех своих з/п.
2. Так же используя расчетный лист может просмотреть более конкретную информацию о начислении з/п.
3. Есть возможность уволиться.
4. Просмотр информации о компании.
5. Редактирование/Просмотр профиля.

## **Разработка информационной модели предметной области**

Методология моделирования IDEF1X, являясь расширением стандарта IDEF1, предназначена для описания данных (информации). В ее основе лежит язык семантического моделирования, основанного на концепции "сущность — связь", позволяющей определять данные и связи между ними. Методология используется для создания информационной модели предметной области с помощью идентификации ее сущностей и связей между ними.

На рисунке 1.8 приведена информационная модель системы в нотации IDEF1X.

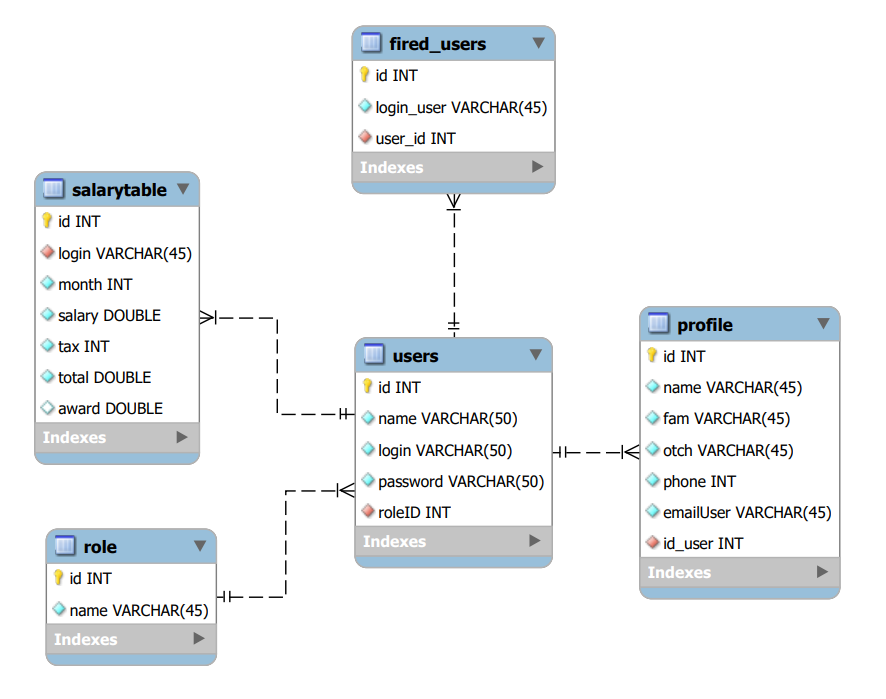


Рисунок 1.8 – Информационная модель

Отношения находятся в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

Отношения находятся во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо зависит от её потенциального ключа.

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых. Разработанная модель соответствует вышеперечисленным требованиям, следовательно, находится в третьей нормальной форме.

Таким образом, мы разработали информационную модель, соответствующую поставленным требованиям. На этом можно завершить процесс моделирования и приступить к разработке системы расчета заработной платы.

## **UML-модели представления программного средства и их описание**

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Конечно, если разрабатывается простая программа, которая может выполняться локально на компьютере пользователя, не задействуя никаких периферийных устройств и ресурсов, то в этом случае нет необходимости в разработке дополнительных диаграмм. Однако при разработке корпоративных приложений ситуация представляется совсем по-другому.

На рисунке 1.9 приведена диаграмма развёртывания программного продукта.

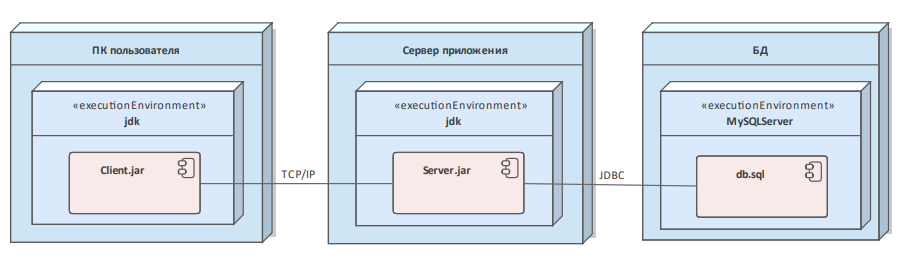


Рисунок 1.9 – Диаграмма развертывания

Диаграмма последовательности – это способ описать поведение системы "на примерах". Фактически, диаграмма последовательности – это запись протокола конкретного сеанса работы системы (или фрагмента такого протокола). В объектно-ориентированном программировании самым существенным во время выполнения является посылка сообщений взаимодействующими объектами. Именно последовательность посылки сообщений отображается на данной диаграмме, отсюда и название.

На рисунке 1.10 изображена диаграмма последовательности разрабатываемой системы. Она отражает последовательность взаимодействия пользователя с приложением клиента, также показывает необходимость общения между клиентской и серверной частью приложения. И в то же время на ней показан процесс обращения сервера к используемой базе данных и получение ответа на запрос от нее.

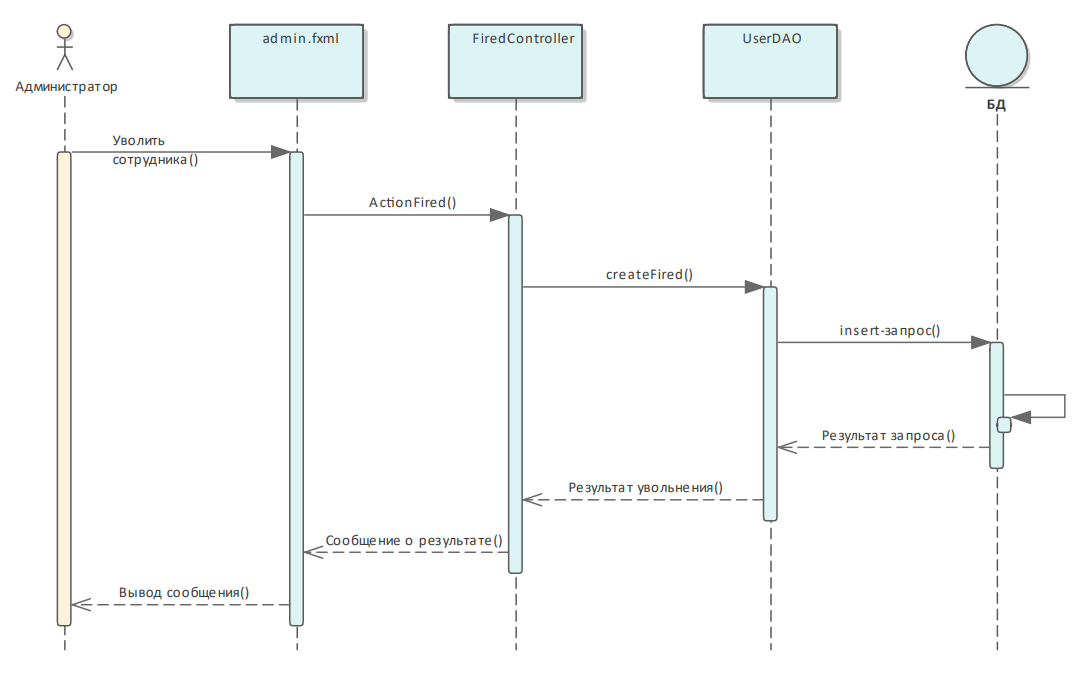


Рисунок 1.10 – Диаграмма последовательности

Диаграмма состояний показывает автомат, сосредотачивая внимание на потоке управления от одного состояния к другому. Изображается в виде графа с вершинами и дугами (ребрами).

На рисунке 1.11 представлена диаграмма состояний расчета заработной платы .

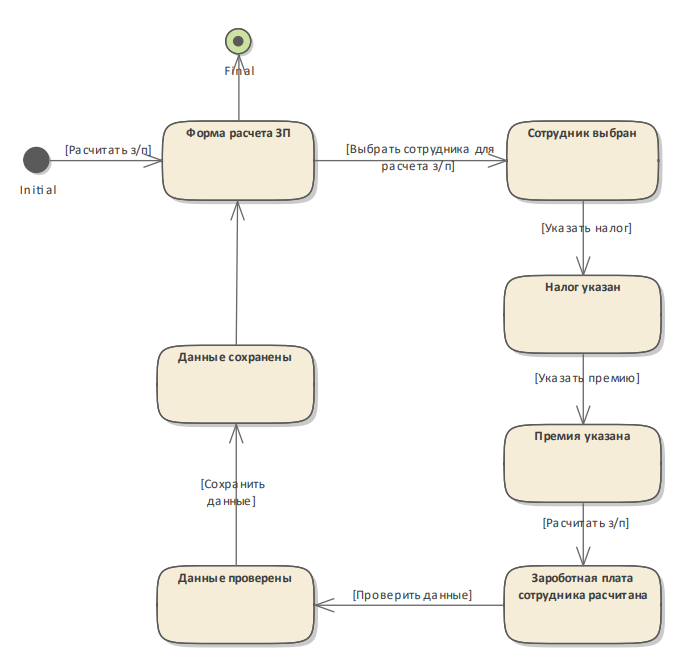


Рисунок 1.11 –Диаграмма состояний

В состав языка UML входят условные обозначения диаграмм классов, которые можно использовать для отображения классов, интерфейсов, а также ассоциаций между ними. Диаграммы классов предназначены для статического моделирования объектов.

На рисунке 1.12 представлена диаграмма классов клиентского приложения.

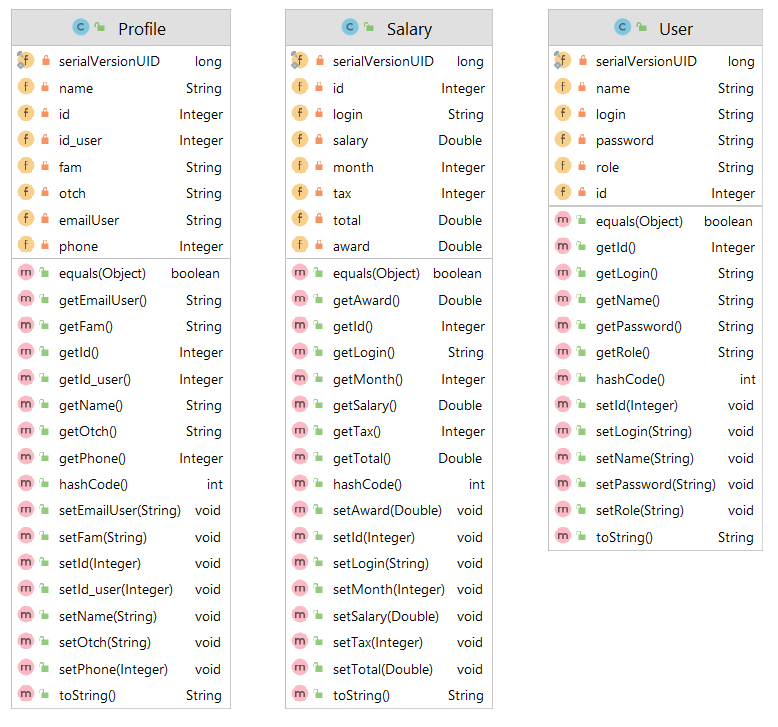


Рисунок 1.12 – Диаграмма классов клиентского приложения

На рисунке 1.13 приводится диаграмма классов серверной части приложения.

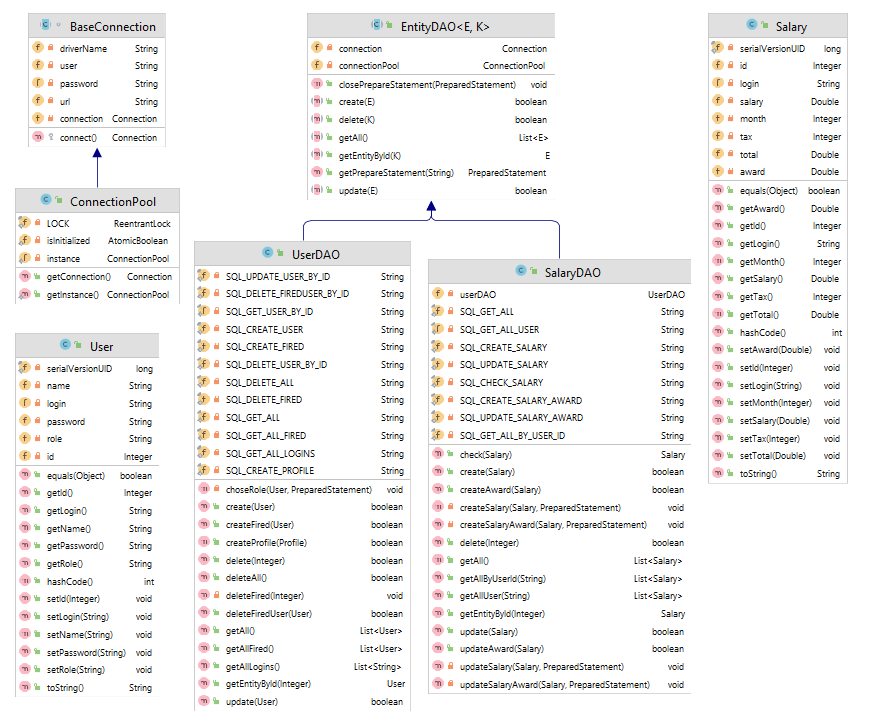


Рисунок 1.13 – Диаграмма классов сервера

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой и отражает представление о поведении системы с точки зрения пользователя. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать, или целей, которые он преследует по отношению к разрабатываемой системе.

Диаграмма вариантов использования показывают отдельные прецеденты в системе некоторых действующих лиц и отношения между ними. Демонстрируется высокоуровневое описание системы, когда действующим актером становится все элементы предполагаемой информационной системы, которые взаимодействуют в исследуемой задаче.

На рисунке 1.14 приведена диаграмма вариантов использования. Как видно из диаграммы пользователи системы с разными ролями имеют разный набор прав доступа.

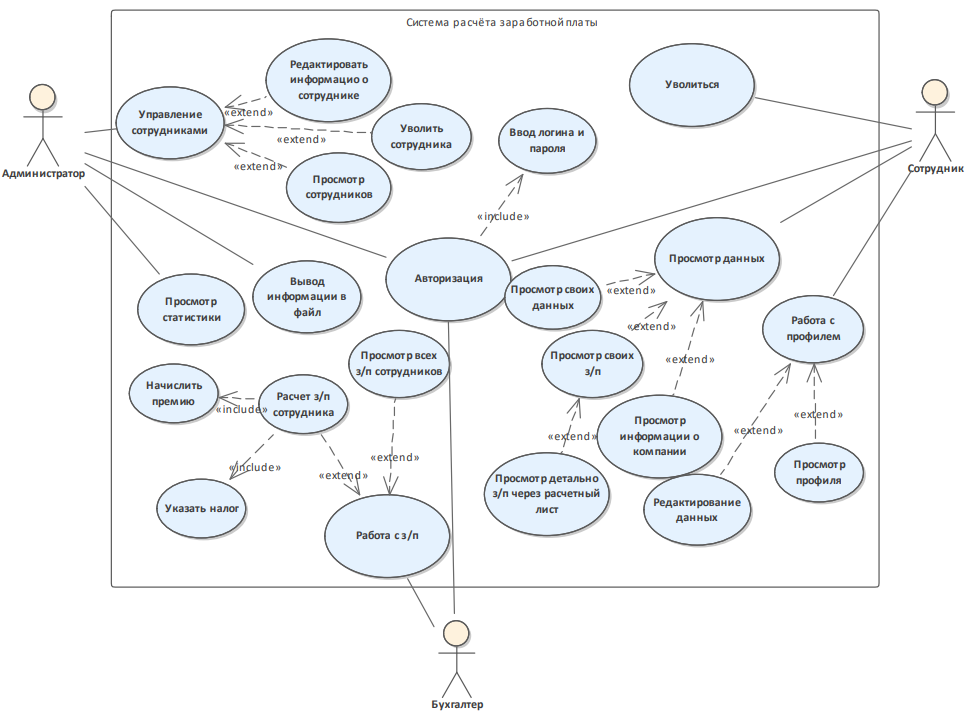


Рисунок 1.14 – Диаграмма вариантов использования

Таким образом, по результатам UML-моделирование составлено полной представление о функционировании будущего программного средства расчета заработной платы.

# **Проектирование и конструирование программного средства**

## **Постановка задачи.**

Поставленная цель потребовала решения следующих задач:

* изучить процесс расчета заработной платы;
* разработать систему расчета заработной платы;
* протестировать программу;
* описать разработанное приложение.

## **Архитектурные решения**

При написании приложения было решено следовать концепции MVC. Принцип MVC, позволяет разделить реализацию логики приложения, внешний вид и взаимодействие с пользователем. Это приводит к более структурированном коде, позволяет работать над проектом более специализированным людям, упрощает поддержку кода, делает его более логичным и понятным. Изменение в одном из компонентов минимально влияет на остальные. Можно к одной модели подключать разные виды, разные контроллеры.

Архитектура MVC предлагает множество преимуществ для программиста при разработке приложений, которые включают в себя:

1. Несколько разработчиков могут работать с тремя слоями (Модель, Вид и Контроллер) одновременно.
2. Обеспечивает улучшенную масштабируемость, которая дополняет способность приложения расти.
3. Поскольку компоненты имеют низкую зависимость друг от друга, их легко поддерживать.
4. Модель может быть повторно использована несколькими представлениями, что обеспечивает возможность повторного использования кода.
5. Принятие MVC делает приложение более выразительным и простым для понимания.
6. Расширение и тестирование приложения становится легким.

## **Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

На рисунке 2.1 приведен алгоритм работы все программы.

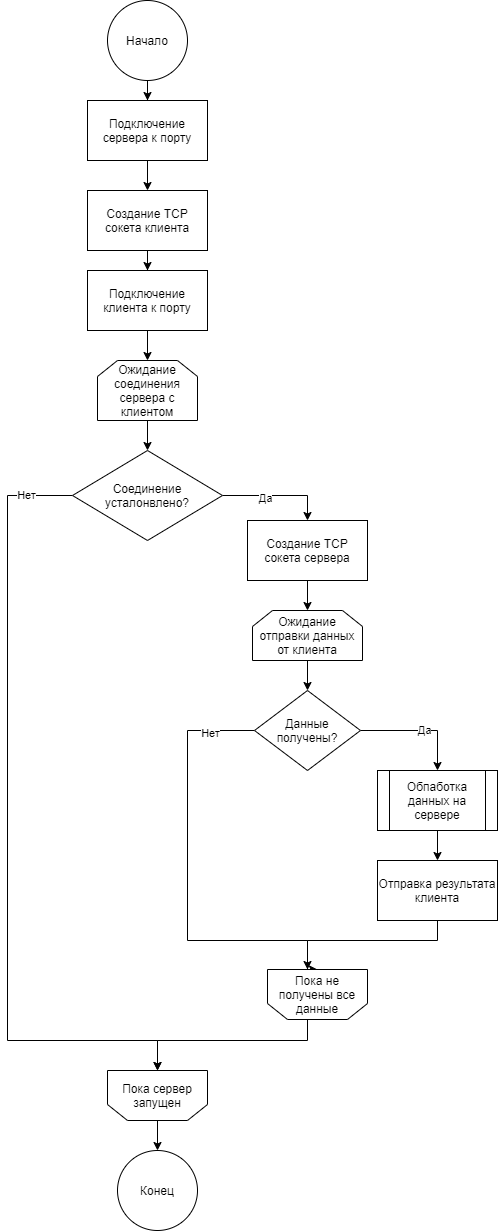


Рисунок 2.1 – Алгоритм работы всей программы

На рисунке 2.2 приведена блок-схема работы основного алгоритма приложения – расчет заработной платы.

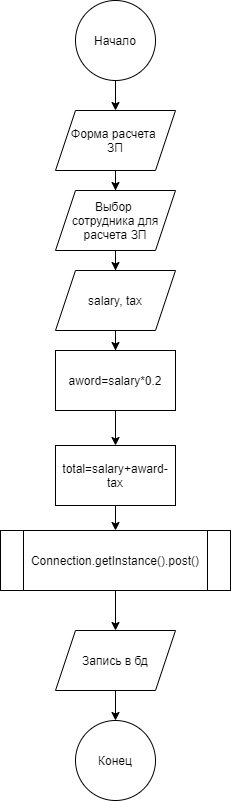


Рисунок 2.2 –Алгоритм работы расчет заработной платы

Таким образом, с помощью блок-схем рассмотрены основные алгоритмы, используемые в приложении.

## **Проектирование пользовательского интерфейса**

Будет разработано GUI-приложение с проектированием графического интерфейса с помощью JavaFX.

JavaFX – это набор инструментов для создания кроссплатформенных приложений на основе языка программирования Java. Этот инструментарий был создан с целью упростить написание приложений. Другими словами, JavaFX — это декларативный язык сценариев со статической типизацией.

Особенности JavaFX:

1. Его структура и API-интерфейсы предназначены для создания анимации, 2D- и 3D-геометрии, диаграмм, специальных эффектов, цветовых градиентов, графических элементов управления и простых манипуляций с мультимедиа, включая аудио, видео и изображения.
2. Визуализация графики JavaFX пользуется возможностями аппаратного ускорения. Это обеспечивает хорошую производительность рендеринга графики и анимации.
3. Вы можете встраивать содержимое JavaFX в панели Swing с помощью JFXPanel. Магия этого специализированного компонента Swing позволяет создавать сложные приложения платформы NetBeans.

Как правило, JavaFX состоит из трех основных компонентов: Stage, Scene и Nodes,

Stage – это вышестоящее окно, в котором мы можем вводить все элементы графического интерфейса. Он содержит все объекты приложения JavaFX и представлен классом Stage пакета javafx.stage. По сути он и является точкой входа.

Scene – это физическое содержимое приложения JavaFX. В нем находится все содержимое графы Stage. Класс Scene пакета javafx.scene представляет объект сцены. В одном экземпляре объект сцены добавляется только к одному этапу.

Node – визуальное содержимое Scene. Сюда могут входить:

* геометрические объекты (как 2D, так и 3D), а также такие фигуры, как прямоугольник, круг, многоугольник;
* элементы, с которыми взаимодействуют пользователи: кнопки, поля выборов, разнообразные флажки и многое другое;
* панели – сюда относятся различные панели сетки и границ;
* объекты мультимедиа – изображения, видео и аудио.

## **Обоснование выбора компонентов и технологий для реализации программного средства**

Для разработки приложения используется объектно-ориентированный язык программирования Java.

У Java масса достоинств, благодаря которым многие разработчики выбирают именно его.

Простота – первое техническое достоинство Java. У него чёткие синтаксические правила и понятная семантика. Рациональность и краткость очень полезны для обработки кода машинами с ограниченным объёмом ресурсов. Для встроенных устройств создана специальная платформа Java Micro Edition.

Объектно-ориентированный подход. За 3 десятилетия он доказал свою эффективность. Суть состоит в том, что в центре внимания находятся данные (объекты), интерфейсы и алгоритмы вторичны. Другими словами, мы отталкиваемся от результата при выборе инструментов, способов их применения.

Безопасность. Важнейший критерий, учитывая применение языка в сетевых / распределенных средах. Разработчики проделали большую работу по защите платформы Java. И она продолжается. Обойти или взломать механизмы защиты крайне сложно. Пример: использование классов, имеющих цифровую подпись. Полные права предоставляются только при полном доверии автору класса.

Производительность. Изначально она вызывала вопросы. Новые версии динамических компиляторов Java не уступают традиционным из других платформ. Мощный прирост скорости обработки даёт оптимизация тех фрагментов кода, которая исполняется чаще. При необходимости те или иные приёмы оптимизации включаются или отменяются JIT-компилятором.

Надёжность – одно из важнейших преимуществ. Программы на Java стабильно работают в любых условиях. Компилятор способен выявить ошибки ещё до выполнения кода, то есть на ранних стадиях. Контроль выполнения позволяет предотвратить сбои в памяти (например, из-за неточного указателя). Сами указатели можно применять не везде, а только там, где это необходимо (скажем, в работе со связными списками).

Независимость от аппаратной части и ОС. Важно лишь наличие исполняющей среды и JVM. А компьютерная архитектура в целом значения не имеет. Байт-код легко интерпретируется на любой машине. Подход доказал свою состоятельность во многом благодаря динамической компиляции. Кроссплатформенностью отличается также интерфейс, реализованный в системных библиотеках.

Динамичность и адаптируемость. Эта особенность позволяет Java не теряться в постоянно изменяющейся среде. При необходимости можно добавить в библиотеки новые объекты, методы. При этом трогать приложения, использующие данные библиотеки не нужно. Отследить информацию о структуре объектов, их поведении, о ходе выполнения программы очень легко.

Удобные и эффективные сетевые возможности.

Не надо забывать, что Java – это трио, состоящее из языка программирования, мощного универсального обработчика и внушительной библиотеки. Все эти наработки доступны программистам. Им не нужно разрабатывать с нуля многие необходимые процедуры (доступ к сети, базам данных и т.п). Это тоже весомый аргумент в пользу Java.

Для хранения данных используется база данных MySQLServer.

MySQL – это система управления базами данных (СУБД), распространяемая как свободное программное обеспечение (пользователи имеют право на неограниченную установку, запуск, свободное использование). Преимущества MySQL:

Открытый исходный код. Распространяется бесплатно для домашнего применения.

Простота. MySQL легко устанавливается, имеет понятный интерфейс, а разнообразие плагинов и дополнительных приложений упрощает работу с БД.

Функционал. Включает в себя практически весь необходимый набор инструментов, который может пригодиться при разработке любого проекта.

Безопасность. Многие системы безопасности уже встроены и работают по умолчанию.

Масштабируемость. Может использоваться в работе как с малым, так и с большим объемом данных.

Скорость. Является одной из самых быстрых среди имеющихся на современном рынке.

По функциональной оснащенности и надежности MySQL давно конкурирует с другими известными продуктами. Чаще всего ее используют при разработке веб-решений, что объясняется тесной интеграцией с популярными языками программирования, высокими показателями скорости и, конечно, ее бесплатностью.

Таким образом, на MySQL обращают внимание как небольшие компании, так и крупные корпорации. Данная СУБД привлекает своей надежностью и свободным распространением. Если дело касается веб-разработки, то MySQL практически всегда будет лучшим вариантом (с учетом потребностей, специфики проекта и экономической целесообразности).

# **Тестирование и проверка работоспособности программного средства**

Приложение полностью обрабатывает возникающие исключения и ошибки пользователя.

На рисунке 3.1 приведено сообщение при неверно введенном логине или пароле при авторизации.

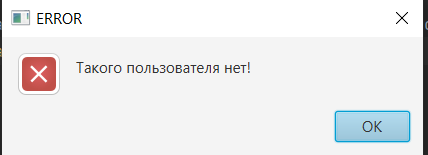


Рисунок 3.1 – Ошибка авторизации

На рисунке 3.2 приведено сообщение об ошибке при сохранении профиля сотрудника.

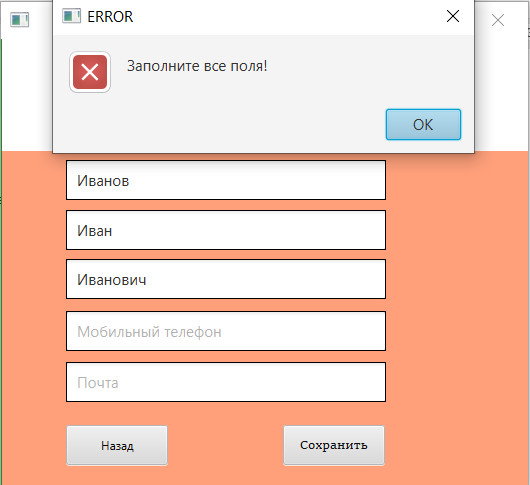


Рисунок 3.2 – Ошибка при сохранении профиля

Таким образом, приложение полностью проверяет введенные данные и сообщает об ошибках пользователю.

# **Инструкция по развертыванию приложения и сквозной тестовый пример**

После запуска приложения появляется окно авторизации(рисунок 4.1).

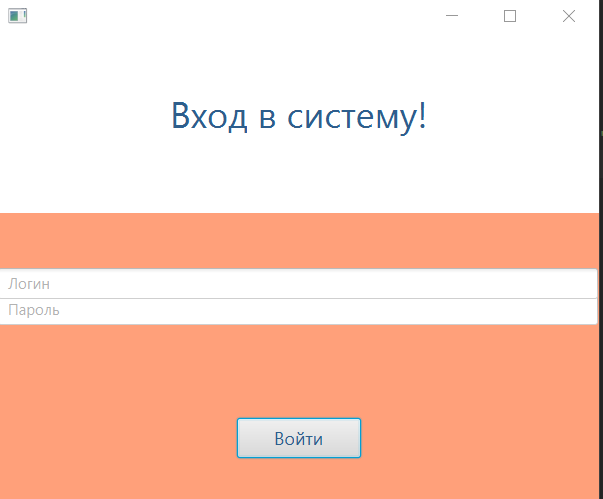


Рисунок 4.1 – Окно входа в систему

Для авторизации необходимо ввести логин и пароль учетной записи, после чего будет открыта окно соответствующей роли пользователя.

На рисунке 4.2 приведено окно администратора.

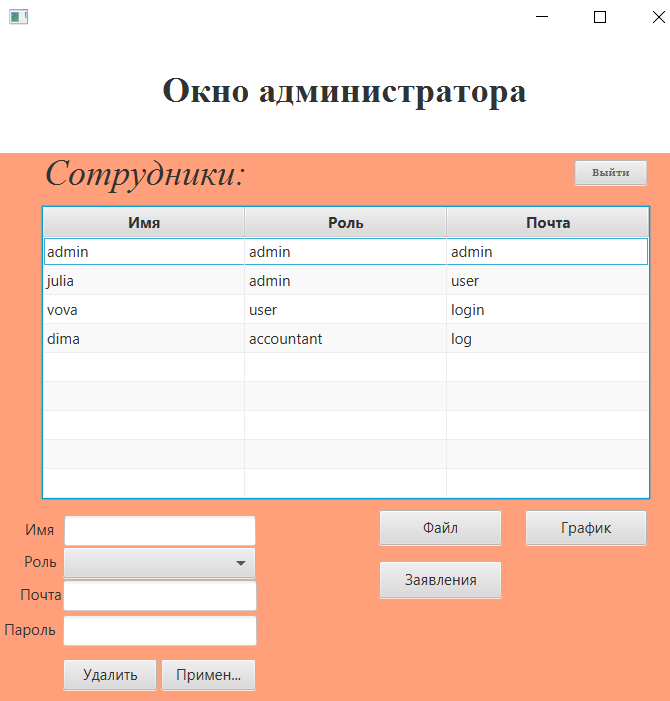


Рисунок 4.2 – Окно администратора

Администратор имеет возможность просматривать таблицу зарегистрированных пользователей, а также изменять данные и удалять учетные записи. Для этого необходимо выбрать учетную запись из таблицы, после чего информация о сотруднике отобразится в текстовых полях ниже(рисунок 4.3). Можно изменить данные в текстовых полях и нажать кнопку «изменить», после чего новые данные будут сохранены, тут же нажав на кнопку удаления, можно полностью ужалить учетную запись выбранного пользователя.

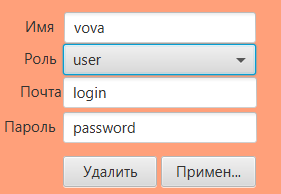


Рисунок 4.3 – Изменение данных пользователя

Также администратор может вывести данные в файл, нажав на советующую кнопку. При успешном создании файла, администратор увидит сообщение об успешности операции(рисунок 4.4).

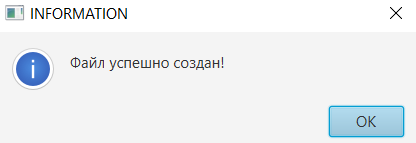


Рисунок 4.4 –Сообщение об успешности вывода информации в файл

Пример вывод информации в файл приведен на рисунке 4.5.

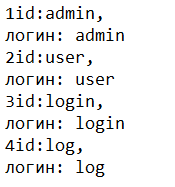


Рисунок 4.5 – Пример сформированного файла

У администратора также есть возможность просмотра статистики по подключениям пользователей. Диаграмма посещений приведена на рисунке 4.6.

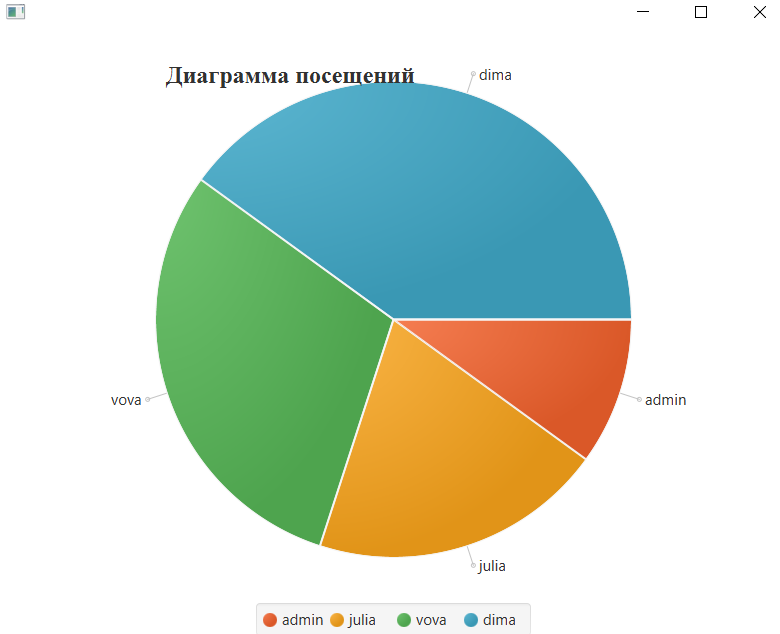


Рисунок 4.6 – Диаграмма посещений

Следующая существующая роль в системе – роль бухгалтера. На рисунке 4.7 приведено окно бухгалтера.

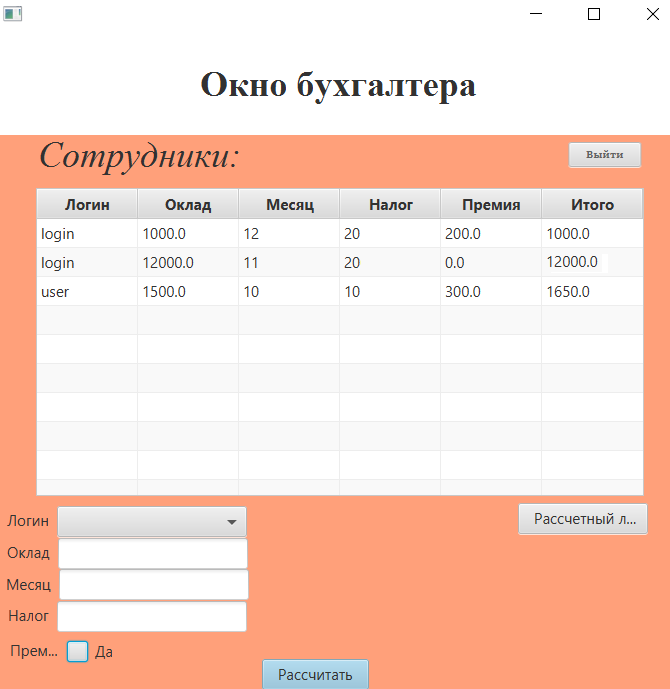


Рисунок 4.7 – Окно бухгалтера

Бухгалтер имеет возможность просматривать начисляемую заработную плату. А также провести расчет заработной платы для отдельного сотрудника. Для расчета заработной платы необходимо выбрать сотрудника из выпадающего списка, ввести размер оклада, месяц начисления зарплаты, процент налога и наличие премии(рисунок 4.8).

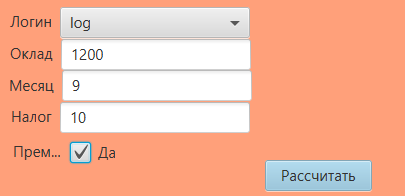


Рисунок 4.8 – Расчет заработной платы

Оставшаяся роль пользователя системы – роль сотрудника. Окно системы для сотрудника приведено на рисунке 4.9.

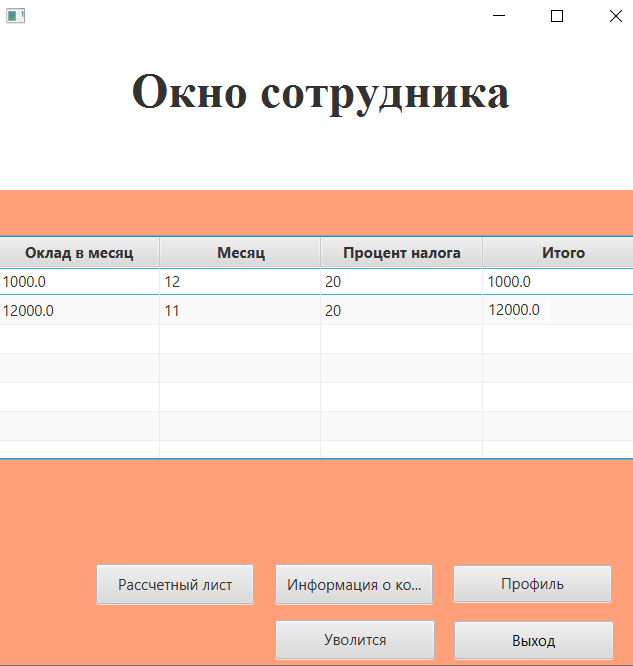


Рисунок 4.9 – Окно сотрудника

Для просмотра информации о компании, сотруднику необходимо нажать на соответствующую кнопку, после чего информация будет выведена на экран(рисунок 4.10).

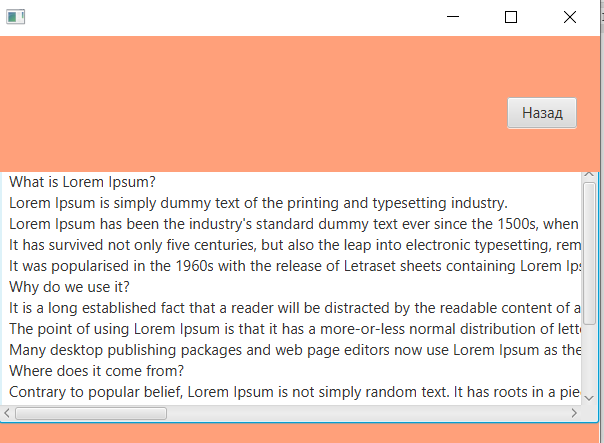


Рисунок 4.10 – Просмотр информации о компании

Сотрудник также может редактировать информацию профиля о себе(рисунок 4.11).

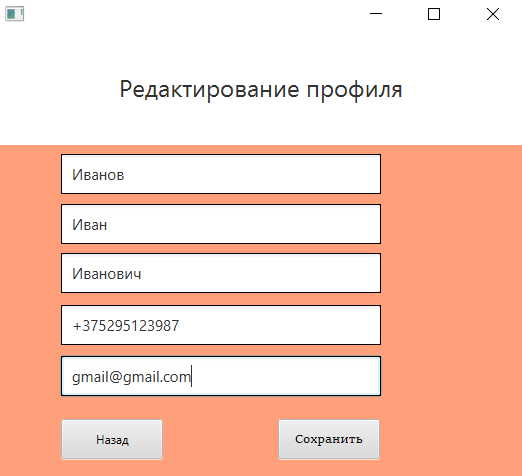


Рисунок 4.11 – Редактирование профиля

Таким образом, было описано подробная работа программы, позволяющая пользователем быстро разобраться в работе системы.

# **Заключение**

По окончанию работы над данным проектом можно сказать, что все поставленные задачи выполнены успешно, а цель достигнута. В ходе выполнения данного проекта было спроектировано и программно реализована система начисления заработной платы сотрудникам. В ходе разработки курсового проекта был детально изучен теоретический материал по данной предметной области, разработаны структуры хранимых данных, пользовательские функции приложения и алгоритмы работы функций.

Данное приложение позволяет осуществлять добавление, изменение, удаление и просмотр данных. Программа выполняет обработку всех исключительных ситуаций и сообщает пользователю об его ошибочных действиях. Систематизированные и обработанные данные, а также отчеты, графики будут очень наглядны при анализе введенных данных.

Основные функции программного средства реализованы в соответствии с выявленными особенностями предметной области. Был разработан довольно широкий функционал. Стиль интерфейса программы создавался с упором на массовость потребления и использования, который позволит любому пользователю легко и просто использовать данное программное средство без лишних временных затрат.

В ходе создания программного средства были подробно изучены особенности расчета заработной платы. Построение диаграммы вариантов использования, функциональной модели IDEF0 – это этапы изучения предметной области, которые очень важны для успешного выполнения поставленной цели. Также была рассмотрена архитектура созданного программного средства. Помимо этого, в ходе выполнения проекта было составлено руководство пользователю, где понятным и доступным языком описывается принцип работы программы.

Благодаря хорошей проектировке данной программы в дальнейшем она может легко может быть дополнена и усовершенствована

# **Список использованных источников**

[1] Цуканова О.А. Методология и инструментарий моделирования

бизнес-процессов: учебное пособие–СПб.: Университет ИТМО, 2015. – 100 с.

[2] Habr [Электронный ресурс]. — Режим доступа: https://habr.com/ru/post/508710/

[3] Вендров А.М. В29 Проектирование программного обеспечения экономических информационных систем: Учебник. - 2-е изд., перераб. и доп. — М.: Финансы и статистика, 2006. — 544 с: ил

[4] Куклина И.Г, Информационные технологии и платформы разработки информационных систем: Куклина И.Г. – 2018. – 283с.

[5] Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание/ Крэг Ларман. — СПб.: Вильямс, 2019-736с.

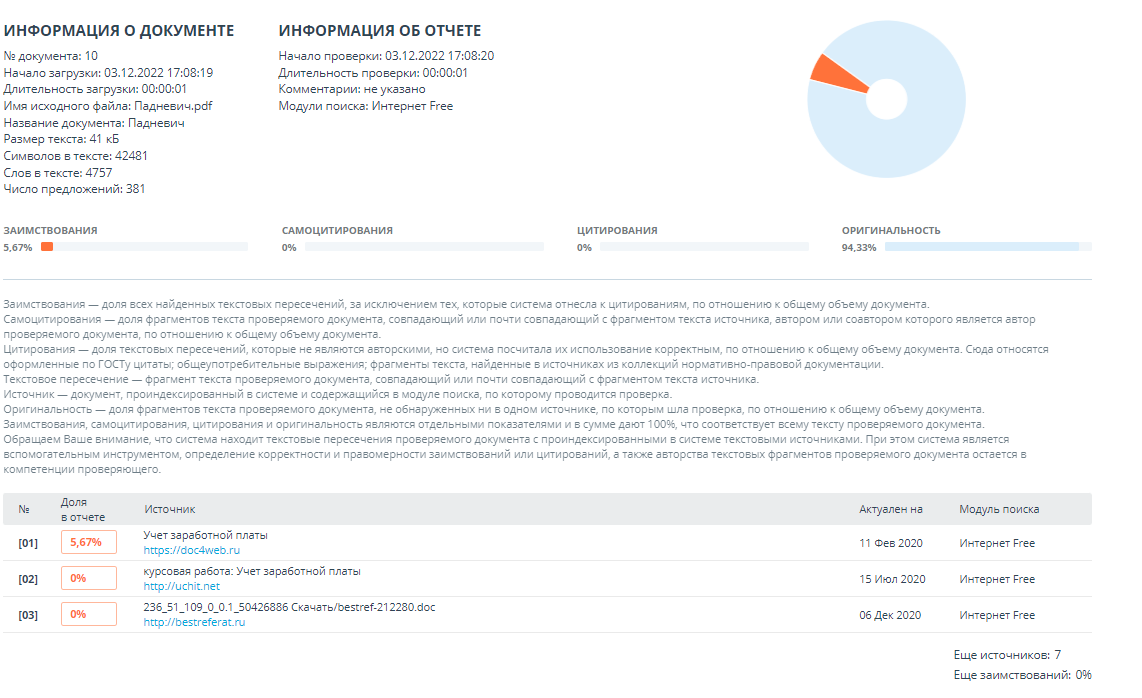
[6] Studme [Электронный ресурс]. — Режим доступа: https://studme.org/87187/ekonomika/metodologiya\_idef1x.

[7] Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.: ил.

[8] Дж. Рамбо, М.Блаха UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд.—СПБ.: Питер, 2007.—544с.:ил.

[9] А. Леоненков. Самоучитель UML—2015, СПБ

# **Приложения А Отчет о проверке на заимствования в системе «Антиплагиат»**



# **Приложение Б Листинг кода алгоритмов, реализующих основную бизнес-логику**

public class StartServer extends Thread {

private ServerSocket serverSocket;

private Socket clientSocket;

private ArrayList<ClientThread> clientsConnected = new ArrayList<ClientThread>();

private ResultSet result = null;

public static void main(String[] arg) {

StartServer server = new StartServer();

server.start();

}

@Override

public void run() {

startServer();

}

private void startServer() {

try {

serverSocket = new ServerSocket(Const.PORT);

System.out.println("start started in Thread - " + Thread.currentThread().getName() + "\n" +

"Waiting for connection....");

while (!serverSocket.isClosed()) {

clientSocket = serverSocket.accept();

System.out.println("connection established...");

ClientThread client = new ClientThread(clientSocket);

clientsConnected.add(client);

client.start();

}

} catch (IOException e) {

e.printStackTrace();

System.err.println("IOException in StartServer.run()");

}

}

class ClientThread extends Thread {

Socket clientSocket;

UserDAO userDAO = new UserDAO();

SalaryDAO salaryDAO = new SalaryDAO();

ObjectInputStream input = null;

ObjectOutputStream output = null;

ClientThread(Socket clientSocket) {

this.clientSocket = clientSocket;

try {

output = new ObjectOutputStream(clientSocket.getOutputStream());

input = new ObjectInputStream(clientSocket.getInputStream());

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void run() {

while (clientSocket.isConnected()) {

try {

String clientMessage = (String) input.readObject();

System.out.println(clientMessage);

String[] message = clientMessage.split(" ", 2);

String commandNumberStr = message[0];

String command = message[1];

switch (commandNumberStr) {

case "usersTable":

post(userDAO.getAll());

break;

case "usersFiredTable":

post(userDAO.getAllFired());

break;

case "usersLogins":

post(userDAO.getAllLogins());

break;

case "salaryTable":

post(salaryDAO.getAll());

break;

case "salaryTableUser": {

post(salaryDAO.getAllUser(command));

break;

}

case "logIn": {

String[] values = command.split(" ", 2);

List<User> users = userDAO.getAll();

User user = null;

for (User temp : users) {

if (temp.getLogin().equals(values[0]) && temp.getPassword().equals(values[1])) {

user = temp;

break;

}

}

post(user);

break;

}

case "sighProfile": {

String[] valuesSighUp = command.split(" ", 6);

Profile profile = new Profile();

profile.setName(valuesSighUp[0]);

profile.setFam(valuesSighUp[2]);

profile.setOtch(valuesSighUp[1]);

profile.setPhone(Integer.valueOf(valuesSighUp[3]));

profile.setEmailUser(valuesSighUp[4]);

profile.setId\_user(Integer.valueOf(valuesSighUp[5]));

userDAO.createProfile(profile);

break;

}

case "addUser":

String[] valuesSighUp = command.split(" ", 5);

User user = null;

Boolean test = null;

if (valuesSighUp[0].equals("null")) {

user = new User(0, valuesSighUp[1], valuesSighUp[2], valuesSighUp[3], valuesSighUp[4]);

test = userDAO.create(user);

} else {

user = new User(Integer.parseInt(valuesSighUp[0]), valuesSighUp[1], valuesSighUp[2], valuesSighUp[3], valuesSighUp[4]);

test = userDAO.update(user);

}

post(test);

break;

case "deleteFiredUser":

valuesSighUp = command.split(" ", 2);

user = null;

test = null;

if (!valuesSighUp[0].equals("null")) {

user = new User(Integer.parseInt(valuesSighUp[0]), valuesSighUp[1]);

test = userDAO.deleteFiredUser(user);

}

post(test);

break;

case "addSalary":

valuesSighUp = command.split(" ", 5);

Salary salary = null;

test = null;

if (valuesSighUp[0].equals("null")) {

salary = new Salary(0, valuesSighUp[1], Double.parseDouble(valuesSighUp[2]), Integer.parseInt(valuesSighUp[3]), Integer.parseInt(valuesSighUp[4]));

test = salaryDAO.create(salary);

} else {

salary = new Salary(0, valuesSighUp[1], Double.parseDouble(valuesSighUp[2]), Integer.parseInt(valuesSighUp[3]), Integer.parseInt(valuesSighUp[4]));

test = salaryDAO.create(salary);

}

post(test);

break;

case "addSalaryAward":

valuesSighUp = command.split(" ", 6);

salary = null;

test = null;

if (valuesSighUp[0].equals("null")) {

salary = new Salary(0, valuesSighUp[1], Double.parseDouble(valuesSighUp[2]), Integer.parseInt(valuesSighUp[3]), Integer.parseInt(valuesSighUp[4]),Double.parseDouble(valuesSighUp[5]));

test = salaryDAO.createAward(salary);

} else {

salary = new Salary(0, valuesSighUp[1], Double.parseDouble(valuesSighUp[2]), Integer.parseInt(valuesSighUp[3]), Integer.parseInt(valuesSighUp[4]),Double.parseDouble(valuesSighUp[5]));

test = salaryDAO.createAward(salary);

}

post(test);

break;

case "deleteUser": {

post(userDAO.delete(Integer.parseInt(command)));

break;

}

case "deleteFired": {

post(userDAO.deleteAll());

post(userDAO.delete(Integer.parseInt(command)));

break;

}

case "addFired": {

valuesSighUp = command.split(" ", 1);

user = null;

test = null;

user = new User(Integer.parseInt(valuesSighUp[0]));

test = userDAO.createFired(user);

post(test);

break;

}

case "allSalaryUser": {

post(salaryDAO.getAllByUserId(command));

break;

}

}

} catch (IOException | ClassNotFoundException e) {

try {

clientSocket.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

}

void post(Object obj) {

try {

output.writeObject(obj);

} catch (Exception e) {

e.printStackTrace();

}

}

Object get() {

try {

return input.readObject();

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

return null;

}

}

}

# **Приложение В Листинг скрипта генерации базы данных**

-- MySQL Script generated by MySQL Workbench

-- Sun Nov 6 13:46:47 2022

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema system\_account

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema system\_account

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `system\_account` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;

USE `system\_account` ;

-- -----------------------------------------------------

-- Table `system\_account`.`role`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `system\_account`.`role` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 4

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `system\_account`.`users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `system\_account`.`users` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

`login` VARCHAR(50) NOT NULL,

`password` VARCHAR(50) NOT NULL,

`roleID` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `login\_UNIQUE` (`login` ASC) VISIBLE,

INDEX `roleID` (`roleID` ASC) INVISIBLE,

INDEX `id` (`id` ASC) VISIBLE,

CONSTRAINT `users\_ibfk\_1`

FOREIGN KEY (`roleID`)

REFERENCES `system\_account`.`role` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 49

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `system\_account`.`fired\_users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `system\_account`.`fired\_users` (

`id` INT NOT NULL AUTO\_INCREMENT,

`login\_user` VARCHAR(45) NOT NULL,

`user\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_idx` (`user\_id` ASC) VISIBLE,

CONSTRAINT `fk`

FOREIGN KEY (`user\_id`)

REFERENCES `system\_account`.`users` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

AUTO\_INCREMENT = 25

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `system\_account`.`profile`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `system\_account`.`profile` (

`id` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

`fam` VARCHAR(45) NOT NULL,

`otch` VARCHAR(45) NOT NULL,

`phone` INT NOT NULL,

`emailUser` VARCHAR(45) NOT NULL,

`id\_user` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_idx` (`id\_user` ASC) VISIBLE,

CONSTRAINT `fk1`

FOREIGN KEY (`id\_user`)

REFERENCES `system\_account`.`users` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

-- -----------------------------------------------------

-- Table `system\_account`.`salarytable`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `system\_account`.`salarytable` (

`id` INT NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(45) NOT NULL,

`month` INT NOT NULL,

`salary` DOUBLE NOT NULL,

`tax` INT NOT NULL,

`total` DOUBLE NOT NULL,

`award` DOUBLE NULL DEFAULT NULL,

PRIMARY KEY (`id`),

INDEX `login` (`login` ASC) VISIBLE,

CONSTRAINT `fk2`

FOREIGN KEY (`login`)

REFERENCES `system\_account`.`users` (`login`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

AUTO\_INCREMENT = 27

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

**Ведомость документов курсового проекта**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнитель-ные сведения* | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР КР 1-40 05 01-08 073 ПЗ | | | | | Пояснительная записка | | | | 44 с. | |
|  | | | | | Отзыв руководителя | | | |  | |
|  | | | | | Рецензия | | | |  | |
|  | | | | | Акт о внедрении | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.502528.073 Д2 | | | | | IDEF0-модель процессов предметной области | | | | Формат А4 | |
| ГУИР.502528.073 Д3 | | | | | Схема алгоритма подбора  поставок | | | | Формат А4 | |
| ГУИР.502528.073 Д5 | | | | | Плакаты, отражающие результаты курсового проектирования | | | | Формат А4  3 листа | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | БГУИР КР 1-40 05 01-08 073 Д1 | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л.* | *№ докум.* | *Подп.* | *Дата* | *Система расчета заработной платы*  *Ведомость курсового проекта* | *Лист* | | | *Лист* | *Листов* |
| *Разраб.* | | *Падневич* |  |  |  | Т |  | 44 | 44 |
| *Пров.* | | *Лыщик* |  |  | Кафедра ЭИ,  гр. 073601 | | | | |
| *Т.контр.* | |  |  |  |
| *Н.контр.* | |  |  |  |
| *Утв.* | |  |  |  |