ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАСЧЕТА ТРУДОЗАТРАТ ДЛЯ ПРЕДПРИЯТИЯ ЗАО «ОСКОЛНЕФТЕМАШ»

Выпускная квалификационная работа
обучающегося по направлению подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения,
группы 07001402
Изварина Максима Александровича

Научный руководитель
к.т.н., доцент Чашин Ю.Г.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

С развитием компьютерных технологий появилась возможность автоматизировать большинство процессов нашей жизни и упростить труд множества профессий известных нам. Учитывая степень развития этих технологий люди, с их помощью, могут упростить массу аспектов своей жизни такие как: быт, досуг, работу, развлечения, общение и множество других. Другими словами мы сейчас повсеместно встречаем и используем информационные технологии, а в частности автоматизированные системы и можно сделать простой вывод, что каждый из приведенных до этого аспектов нашей жизни требуют качественного программного сопровождения для простоты и улучшения качества жизнедеятельности.

Целью ВКР будет разработка программного сопровождения для помощи в расчете трудозатрат и заработной платы на производстве. В процессе разработки этого проекта была реализована автоматизированная система для расчета трудозатрат на предприятии ЗАО «ОсколНефтеМаш», занимающегося изготовлением комплектующих для нефтяных насосов.

В ходе реализации этой цели необходимо решить следующие задачи такие как:

- Автоматизированная работа с сотрудниками и их личными данными;
- Контроль и учет премий, штрафов, отпусков, трудозатрат, больничных, пропусков;
- предоставление информации о тарифных ставках и окладах рабочих, их разряде и профессии;
- расчет их заработной платы за месяц, с учетом всех полученных данных.

Все это должно предоставляться в удобном и дружелюбном интерфейсе для сотрудников, работающих с автоматизированной системой.

Для достижения этих задач будет разработано приложение соответствующее всем этим пунктам и упрощающее труд бухгалтера.

При разработке этого приложения необходимо изучить ряд вопросов, в который входят:

- Анализ деятельности организации, для которой разрабатывается программное обеспечение;
- Изучение рабочего процесса отдела, который будет пользоваться программой;
- Обзор аналогичных систем, выполняющих схожие задачи;
- Формулировка необходимых требований к информационной системе;
- Изучение аналогов;
- Проектирование информационной системы;
- Выбор средств реализации;
- Реализация спроектированной системы;
- Тестирование и анализ готовой информационной системы.

В итоге, эта выпускная квалификационная работа будет состоять из: введения, трех глав, заключения, списка литературы и приложения.

Введение состоит из общих сведений, которые необходимы для описания задач работы.

В первой главе описывается сама организация и ее структура, исследование деятельности предприятия и формулировка задач, решение которых ведет к реализации информационной системы.

Во второй главе описывается выбор средств реализации и проектирование базы данных, а так же разработка программного модуля программы.

Третья глава описанию методов тестирования, разработке интерфейса и самому тестированию программного продукта.

В заключении описываются итоги на основании выполненной работы.

Выпускная квалификационная работа состоит из: 51 страницы, 23 рисунков и приложения состоящего из 15 страниц.

ГЛАВА 1 . ОПИСАНИЕ ОРГАНИЗАЦИИ

1.1 Описание организации

Выпускная квалификационная работа будет выполняться для завода производящего запчасти для нефтяных насосов ООО «ОсколНефтеМаш» - «Оскольский завод нефтяного машиностроения» производит и реализует рабочие ступени электропогружных центробежных насосов. Входит в группу Производственная компания «Борец» - одну из крупнейших международных компаний нефтяного машиностроения, специализирующихся на разработке, производстве, реализации и сервисном обслуживании установок электропогружных центробежных насосов. Компания предлагает интересную работу, перспективы профессионального и карьерного развития, конкурентоспособный уровень заработной платы.

Это современное предприятие, основанное в 2014 г., специализирующееся на производстве ступеней сложной пространственной геометрии для высокодебитных центробежных насосов (ЭЦН).

Технологический комплекс завода обладает высокой гибкостью и позволяет изготавливать литые заготовки широкой номенклатуры при высоком уровне автоматизации, значительно сокращающей количество трудоемких операций.

Предприятие на сегодняшний день уже выпускает ступени для габаритной группы ЭЦН 5А производительностью от 140 до 700 $м^3$/сутки и осваивает ступени габаритной группы 6. Литейный комплекс завода предусматривает изготовление литых заготовок из легированного, серого высокопрочного чугуна и материала Ni-resist Type 1.

Основные и вспомогательные участки оснащены современным оборудованием отечественных и иностранных производителей.

Высокое качество выпускаемой продукции достигается за счет широкой автоматизации всех производственных и технологических процессов предприятия, а также автоматизированного контроля основных технологических параметров работы оборудования. Кроме того, на всех ключевых стадиях предусмотрен визуально-измерительный контроль.

Структура предприятия состоит из подразделений, например: служба главного энергетика, металлообрабатывающий цех, литейный цех и тому подобные, во главе которых находится руководящие лица, такие как: главный инженер, главный энергетик, начальник службы охраны и так далее. В свою очередь у них в подчинение находиться, порядка, 5 мастеров, которые уже руководят рабочими.

1.1.1 Методика расчета заработной платы

В рамках наиболее важный момент, как происходит расчет заработной платы. В основе всего лежит то, что форма оплаты делится на 2 типа: оклад и сдельно-премиальная и в зависимости от типа определяется формула расчета месячного заработка работника. Работники, у которых сдельно-премиальная форма оплаты имеют тарифную ставку, которая зависит от профессионального разряда сотрудника, она умножается на количество отработанных часов, ночные часы в свою очередь облагаются двойной тарифной ставкой, но на них не распространяется премия. Премия начисляется как разовая за какие-то особые достижения конкретного сотрудника, так и общая – это когда подразделение выполнило норму производства на подразделение и не превысило норму брака. В этом случае все работники, участвующие в производстве и имеющие сдельно-премиальную форму оплаты получают премию в размере 25%. При невыполнении плана размер премии корректируется на размер выполнения плана. Ниже будут приведены таблицы для лучшего понимания этого

процесса. В таблице 1.1 описана зависимость размера премии от процента выполнения плана

Таблица 1.1

Зависимость размера премии от процента выполнения плана

Процент невыполнения плана-графика, %	Процент снижения размера премии, %	Размер премии для начисления, %
0	0	25
-1.0	-2.5	22.5
-2.0	-5.0	20.0
-3.0	-7.5	17.5
И т.д.	И т.д.	И т.д.
-10.0	-25.0	0.0

Для определения процента премии устанавливается минимальный процент внутреннего брака в размере 7%, предельно-допустимый 9%.

Процент фактического брака определяется с двумя десятичными знаками. Для определения размера увеличения(снижения) процента премии производится округление десятичных знаков процента фактического брака в следующем порядке:

- 0,44 и менее округляется до 0;
- От 0,45 и более – до 1,0;

В таблице 1.2 показан перерасчет премии в случае снижения брака

Таблица 1.2

Перерасчет размера премии в случае снижения процента брака

Процент снижения брака, %	Процент увеличения премии, %	Размер премии для начисления, %
0	0	25,0
- 1,0	+ 2,0	27,0
-2,0	+ 4,0	29,0
и т.д.	и т.д.	и т.д.
-7,0	+ 14,0	39,0

В таблице 1.3 показан перерасчет размера премии в случае увеличения процента брака. За каждый процент произведенной бракованной продукции у сотрудника вычитается 5% премии.

Таблица 1.3

Перерасчет размера премии в случае увеличения процента брака

Процент увеличения брака, %	Процент снижения премии, %	Размер премии для начисления, %
0	0	25,0
+ 1,0	-5,0	20,0
+ 2,0	- 10.0	15.0
Свыше 2	не выплачивается	

Сотрудники, которые работают на форме оплаты оклад, не имеют тарифной ставки, и расчет их заработной платы идет несколько иначе. Они получают фиксированную сумму в месяц, на которую не распространяется общая премия за выполнения нормы производства, как правило, это сотрудники которые не учувствуют непосредственно в производстве, а занимают либо руководящие должности, либо вспомогательные (охранник, уборщица и т.п.).

Помимо лишения премии на предприятии есть система штрафов, но она не отличается как для работников на сдельно-премиальной форме оплаты, так и тех, кто на окладе.

В случае выхода на работу в праздничный день, работник получает двойную ставку, на которую не распространяется премию и может, по усмотрению начальства, получить разовую премию.

При расчете заработной платы так же учитывают доплату за вредность. Методика вычисления этой доплаты происходит следующим образом – на предприятие приезжает определенная комиссия, которая оценивает рабочее место сотрудника и определяет размер дополнительных средств необходимых сотруднику, дабы покрыть некомфортные или вредные условия труда.

1.2 Обзор существующих аналогов

На сегодняшний день самой распространенной программой для автоматизации бухгалтерского учета является «1С: Бухгалтерия». Несмотря на огромную популярность, отношение к данной программе среди пользователей сложилось двойственное: часть из них признает ее лучшим из существующих средств автоматизации бухгалтерского и налогового учета, часть считает довольно слабым продуктом, имеющим множество недочетов и недостатков.

Если же оценивать «1С: Бухгалтерию» объективно, то ее нельзя назвать ни плохой, ни хорошей, поскольку неправильно давать оценочную характеристику данной программе без привязки к конкретному предприятию. Для некоторых организаций «1С: Бухгалтерия» является идеальным вариантом, полностью удовлетворяющим их требования, предъявляемые к автоматизации бухгалтерского учета. А для некоторых компаний данная программа не подходит, поскольку не способна в полной мере решить поставленные перед ней задачи. Поэтому, прежде чем начинать работать с «1С: Бухгалтерией», необходимо оценить все ее достоинства и недостатки и, учитывая их, принимать окончательное решение о выборе компьютерной программы.[1]

Рассмотрим основные достоинства и недостатки «1С: Бухгалтерии». К достоинствам данной программы можно отнести следующее:

- С помощью «1С: Бухгалтерии» можно вести все существующие виды бухгалтерского и налогового учета;
- На сегодняшний день «1С: Бухгалтерия» является одной из самых универсальных бухгалтерских программ, которая может использоваться в самых разных организациях. Данная программа основана на платформе «1С:Предприятие», которую можно модифицировать под нужды

конкретного бизнеса. Подобная гибкость «1С: Бухгалтерии» позволяет решать с ее помощью множество различных задач;

- «1С: Бухгалтерия» идеально приспособлена под российское законодательство и позволяет легко подстраиваться под регулярно меняющиеся в нашей стране законы и требования чиновников. Разработчики «1С» следят за всеми изменениями в налоговом законодательстве и оперативно обновляют формы отчетности в программе;
- Программа «1С: Бухгалтерия» (особенно ее последняя версия – «1С: Бухгалтерия 8») обладает высокой производительностью, что дает возможность решать с ее помощью самые сложные задачи.

К сожалению, «1С: Бухгалтерия» обладает и рядом недостатков, к которым можно отнести следующее:

- В подавляющем большинстве случаев, чтобы «1С: Бухгалтерия» решала все поставленные перед ней задачи, программу приходится дорабатывать. Каждое предприятие уникально, поэтому для эффективной его работы, как правило, требуются индивидуальные решения по автоматизации бизнес-процессов (в том числе и по автоматизации ведения бухгалтерского и налогового учета);
- При переходе на «1С: Бухгалтерию» с другой бухгалтерской программы могут возникнуть серьезные затруднения при переносе информации из одной базы данных в другую (значительную часть информации нередко приходится переносить вручную).
- В «1С: Бухгалтерии» затруднен поиск ошибок, сделанных во время обработки документов;
- Программа «1С: Бухгалтерия» достаточно сложна в освоении и требует специального обучения пользователей;
- «1С: Бухгалтерия» достаточно дорогостоящий продукт.[2]

Так же стоит кратко рассмотреть некоторые менее известные программы для ведения бухгалтерского учета.

«Парус» - это современная программа, которая предназначена, чтобы автоматизировать финансовую деятельность коммерческих и государственных предприятий, и управления. Распространяется в Украине, России и Казахстане. Модуль «Парус – Бухгалтерия» это составная часть системы, которая позволяет более удобно составить бизнес-процесс учета, рассчитать и начислить заработную плату, управлять персоналом и деловыми процессами.

Система работает соответственно Международных стандартов бухгалтерского учета, предоставляя возможность проводить учет по МСБУ, и одновременно поддерживать любую трансформацию данных, которые накапливаются в учетных регистрах, к одному виду учета к другому. Можно выделить следующие задачи:

- Автоматизирование бухгалтерских операций в тех организациях, где источниками финансов выступает собственный капитал, и капитал, который привлекается в предприятие;
- Оптимизация сложных бизнес-процессов, и строение учета «от начального документа»;
- Полная поддержка многих пользовательских видов учета, и обеспечение любого преобразования данных, накопленных в документации по учету, в разных видах учёта;
- Контроль по наличию и движению имущества. Также рассматривается рациональное использование производственных ресурсов, своевременное предупреждение негативных факторов в финансовой деятельности;
- Расчет данных в приложении «Расчет зарплаты», для создания учетного журнала, формирования «Кассовых документов».

Недостатки программы «Парус»:

- «Парус» по своей сути закрытая система и не может быть изменена пользователем;
- Освоение программы также может вызывать некоторые трудности, потому как правильно работать там смогут только высококвалифицированные работники;
- Также стоимость этой программы определяется стоимостью одного рабочего места, и с увеличением количества пользователей соответственно растёт.[3]

1.3 Требования к автоматизированной системе

Для успешной реализации темы диплома, существует необходимость определится с требованиями, которым должна соответствовать автоматизированная система.

Требования к функционалу автоматизированной системы:

- Давать бухгалтеру информацию обо всех сотрудниках, а именно: табельный номер, ФИО, название профессии, разряд сотрудника, дату рождения, банковский счет для перевода денежных средств, телефон и подразделение к которому он относится;
- Предоставление информации о профессиях, форме оплаты для каждой профессии, тарифной ставке и окладе, в зависимости от формы оплаты и доплате за вредность;
- Показывать всех сотрудников, которые были в отпуске, дату начала и конца отпуска, сумму отпускных, тип отпуска и количество рабочих дней, которые сотрудник провел в отпуске;
- Выводить список всех штрафов, их размер, причину и дату;
- Показать данные о тех работниках, которым была выдана разовая премия и дату выдачи;

- Предоставлять перечень всех пропусков и прогулов работы и показывать их дату и причину;
- Показывать все больничные, дату начала и конца, больничные выплаты и количество рабочих дней проведенных на больничном;
- Реализация расчета заработной платы на основе трудозатрат на поставленный план произведенной продукции;
- Так же должна быть система поиска по каждому разделу и реализован экспорт в MS Excel;
- Должен быть реализован вывод отчета в банк для перевода денежных средств;
- Так же должна быть сделана система помощи, для сотрудников, которые впервые пользуются программой;
- Реализована система ролей для разграничения прав доступа;
- Реализована панель для администратора, в которой возможно задавать пользователям пароли и роли и изменять их.

Требования к вводу данных:

- Сотрудник должен иметь возможность вводить, удалять и изменять все записи, которые соответствуют его правам доступа.
- На каждой ячейке должны быть подписи, для более понятного и удобного пользования программой.

Требования к выводу данных:

При выводе данных необходимо чтобы после любой операции интерфейс обновлялся, то есть происходило динамически и для того чтобы увидеть изменения не было необходимости перезапуска программы. Excel файлы, экспортированные из приложения должны сохранятся, в специальной папке на ПК, должен создаваться отчет, который содержит имя сотрудника, его банковский счет, счет предприятия и сумму начислений.

Требования к расчету заработной платы:

При расчете заработной платы процесс должен быть автоматизирован, то есть минимальное количество полей, которые вводит сам сотрудник. Бухгалтер должен вводить: количество дневных часов, количество ночных часов, дату, на сколько процентов выполнена норма производства, а так же брака на подразделение и выбрать работника, для которого производится расчет из выпадающего списка. Остальные данные программа должна учитывать автоматически.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1 Выбор средств разработки и обоснование выбора

Для создания информационной системы был выбран язык программирования Java, на платформе JavaFX в среде разработки Intellij Idea. В случае с работой с базой данных была выбрана СУБД(система управления базой данных) PostgreSql. Рассмотрим все это более подробно.

Java –типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems, которую впоследствии приобрела компания Oracle. Дата официального выпуска - 23 мая 1995 года.

Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор[4].

Достоинства языка Java:

- Кроссплатформенность – то есть код Java можно запустить на любой операционной системе и оборудовании, для которого существует Java машина. Это достигается тем, что Java приложения транслируются в специальный байт-код, который обрабатывается JVM.
- Мощные стандартные библиотеки, которые во многом упрощают работу на этом языке программирования.
- Безопасность – функционирование программы полностью определяется и ограничивается Java машиной.
- Сборщик мусора – освобождение памяти при работе осуществляется автоматически, т.е. работать с динамической памятью стало проще[5].

JavaFX — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing. Платформа JavaFX конкурирует с Microsoft Silverlight, Adobe Flash и аналогичными системами.

В далеком 2007 году на JavaOne компания SunMicrosystems представила миру первую версию платформы JavaFX. Для того чтобы иметь возможность писать на этой платформе программы, вам было нужно ставить компилятор, и учить новый язык – JavaFXScript. Но все изменилось, когда Oracle купила SunMicrosystems. Компания решила прекратить развитие JavaFXScript, но сообщила, что 2 версия JavaFX будет портирована на Java. И уже 10 октября 2011 года была выпущена JavaFX версии 2.0, в виде библиотеки для Java. А с выпуском восьмой версии Java, она была добавлена в состав JDK и получила символический номер 8.

Возможности платформы

Среди возможностей этой платформы можно отметить: кроссплатформенность, поддержка каскадных таблицей стилей, поддержка анимации компонентов, возможность работы и отображение 3D графики, поддержка тачей и сенсоров и очень многое другое[6].

В ходе разработки приложения будет использована модель MVC(Model-View-Controller) – разделение данных приложения, интерфейса и логики приложения на 3 отдельных сегмента, таким образом, что каждый из них может редактироваться независимо от других. Один из главных плюсов использования этого шаблона – это разделение логики приложения и минимальное дублирование кода, что делает код гораздо более читабельным и понятным. Несмотря на то, что изначально данная модель разрабатывалась для веб приложений, мы сможем удачно использовать ее и в нашем приложении. Итак, разберем каждый из 3 сегментов более подробно[7].

Модель(Model) - представляет собой объектную модель некой предметной области, включает в себя данные и методы работы с этими данными, реагирует на запросы из контроллера, возвращая данные и/или изменяя своё состояние, при этом модель не содержит в себе информации, как данные можно визуализировать, а также не «общается» с пользователем напрямую. В нашем приложении мы, грубо говоря, разбили модель на 2 части это: сама модель и DAO. В модели все поля класса соответствуют полям таблицы в базе данных так же в нее входят: конструктор, который нужен для создания нового объекта базы данных и так называемые методы чтения геттеры и сеттеры[8].

Геттер - специальный метод, позволяющий получить данные, доступ к которым напрямую ограничен. Это один из методов объектно-ориентированного программирования, который помогает реализовать гибкий механизм инкапсуляции[8].

Сеттер - метод, используемый в объектно-ориентированном программировании для того, чтобы присвоить какое-либо значение инкапсулированному полю[8].

DAO(Data Access Object) – говоря простыми словами объект доступа к данным базы данных. В ней описаны все методы, которые служат для доступа к базе данных такие как: добавление данных, изменение, удаление, поиск, получение данных по идентификационному номеру записи и вывод всех данных из таблицы по средствам SQL запросов. Для каждой из таблиц базы данных были созданы модель и DAO[9].

Представление(view) - отвечает за отображение информации (визуализацию). В нашем случае представление это – fxml файл, который генерируется при помощи программы Scene Builder. JavaFX Scene Builder это - визуальный инструмент дизайна, позволяющий вам быстро создавать интерфейс приложения способом перетаскивания компонентов на рабочую область программы. И код создается в виде XML[8].

Контроллер(Controller) - обеспечивает связь между пользователем и системой, использует модель и представление для реализации необходимой реакции на действия пользователя, как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация (проверяются права пользователя на выполнение действий или получение информации)[8].

На рисунке 2.1 представлена общая схема работы модели MVC.

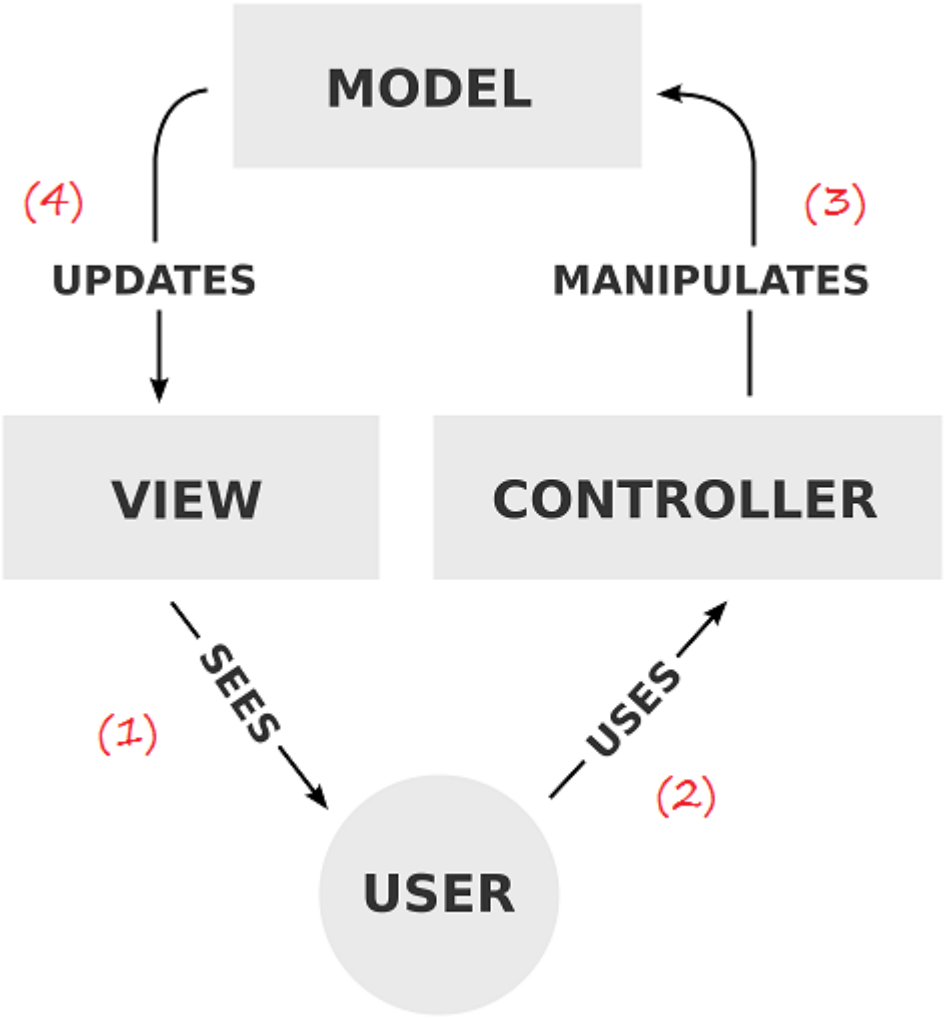


Рис. 2.1. схема работы MVC

Пользователь сначала видит Представление(view) то есть fxml файл, который был создан в Scene Builder и дает команду Контроллеру(Controller) на какое либо действие. После этого Контроллер проводит манипуляции с

Моделью(Model) например: обновить, изменить, добавить и т.д. и возвращает в Представление измененную Модель.

Теперь перейдем к обоснованию выбора СУБД. Нами была выбрана PostgreSQL. PostgreSQL - это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных[10].

PostgreSQL свободно распространяемая и максимально соответствует стандартам SQL. PostgreSQL или Postgres стараются полностью применять ANSI/ISO SQL стандарты своевременно с выходом новых версий[14].

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID)). Благодаря мощным технологиям Postgre очень производительна. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие ACID. PostgreSQL очень легко расширять своими процедурами, которые называются хранимые процедуры. Эти функции упрощают использование постоянно повторяемых операций[10].

Хотя PostgreSQL и не может похвастаться большой популярностью в отличие от MySQL, существует довольно большое число приложений облегчающих работу с PostgreSQL, несмотря на всю мощность функционала. Сейчас довольно легко установить эту СУБД используя, стандартные менеджеры пакетов операционных систем.

Подведем итоги и выделим плюсы PostgreSQL

- Открытое программное обеспечение, соответствующее стандарту SQL.

- PostgreSQL - бесплатное программное обеспечение с открытым исходным кодом. Эта СУБД является очень мощной системой.
- Большое количество дополнений - несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими.
- Расширения - существует возможность расширения функционала за счет сохранения своих процедур.
- Объектность - PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого[10].

2.2 Проектирование базы данных

В основе любого приложения лежит база данных поэтому к этому вопросу нужно подойти максимально ответственно. То есть база данных должна удовлетворять, как минимум трем нормальным формам баз данных.

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Первая нормальная форма - Отношение находится в первой нормальной форме, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице[11].

Вторая нормальная форма - Отношение находится во 2НФ, если оно находится в первой нормальной форме, и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(Primary Key). Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость[11].

Третья нормальная форма - Отношение находится в третьей нормальной форме, когда находится во второй нормальной форме, и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы[11].

Как говорилось выше, для реализации базы данных использовалась СУБД PostgreSQL. Ниже будет представлен рисунок логической модели базы данных, которая была построена в программе Allfusion ERwin Data Modeler.

На рисунке 2.2 изображена логическая модель базы данных

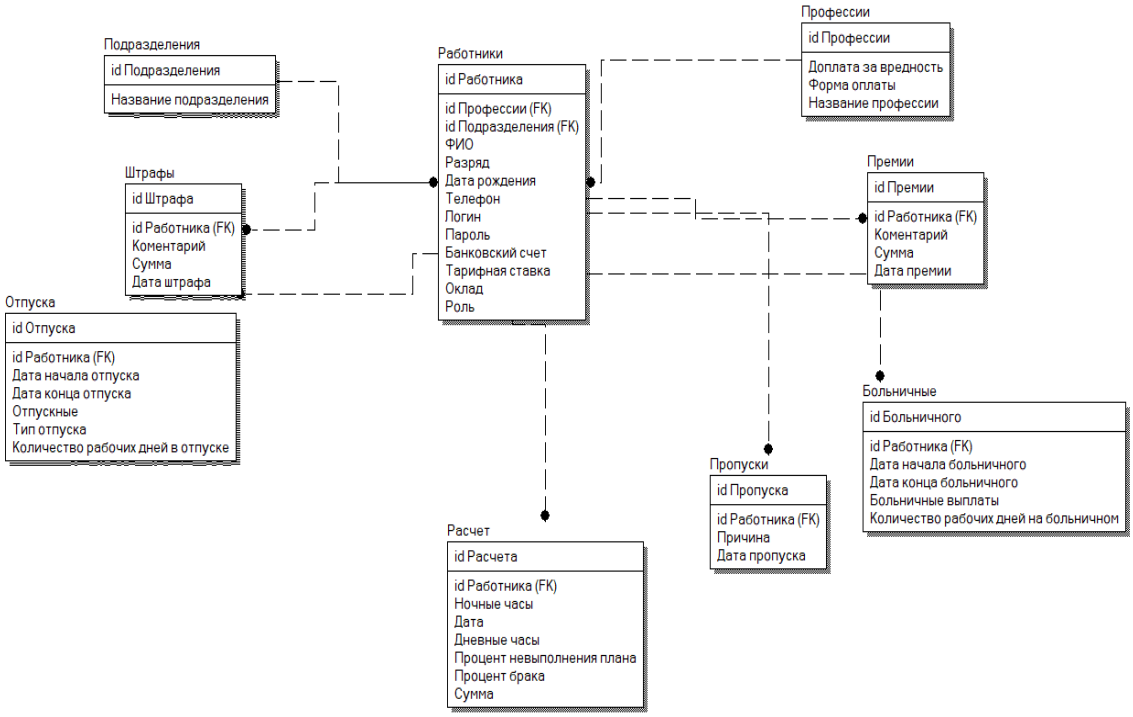


Рис. 2.2. Логическая модель базы данных

Теперь пройдемся по каждой сущности отдельно

Таблица «Подразделения» - создана с целью перечисления всех существующих подразделений или цехов, а так же в ней содержаться нормы

производства и брака на каждое подразделение, для дальнейшего расчета премии.

Таблица «Профессии» служит для перечисления всех профессий, а так же определения формы оплаты сдельно-премиальная или оклад и указания доплаты за вредность, в зависимости от условий труда.

Таблица «Работники» - одна из основных сущностей базы данных в ней перечислена почти вся информации о сотрудниках фирмы, такая как: табельный номер, профессия сотрудника и его подразделение, ФИО, разряд от которого зависит тарифная ставка сотрудников работающих на сдельно-премиальной форме оплаты, дата рождения, телефон, банковский счет для перечисления средств, тарифная ставка и оклад. А так же логин и пароль для авторизации, но он выдается только бухгалтерам т.к. остальным сотрудником кроме бухгалтера, главного бухгалтера и администратора приложения запрещено иметь доступ к данным о заработной плате.

Таблица «Штрафы» содержит список всех штрафов, их причину и сумму штрафа, а так же при помощи вторичного ключа(foreign key) связана с таблицей работников, для определения какой именно работник получил штраф.

«Премии» создана для учета всех разовых премий их размера и заслуг, за которые была выдана премия и аналогичным образом, как и все следующие таблицы, связана с таблицей «Работники».

Таблица «Пропуска» фиксирует все пропуски, независимо от того по уважительной причине или нет. В случае если по неуважительной причине, то сотруднику выписывается штраф за прогул работы.

В таблице «Больничные» учтены все больничные, дата их начала и конца, а также больничные выплаты и количество рабочих дней, которые сотрудник провел на больничном это понадобится для расчета заработной платы.

Таблица «Отпуска» фиксирует отпуска сотрудников и их тип, например: за свой счет, декретный и т.д. а так же отпускные выплаты, дату начала и конца отпуска и количество рабочих дней в отпуске.

И последняя таблица базы данных «Расчет» собирает в себе все данные о доходах и убытках работника и служит для расчета заработной платы.

2.3 Программная реализация автоматизированной системы

При реализации информационной системы вся структура программы была разбита на пакеты(package), каждый пакет предоставляет уникальное пространство имен для своего содержимого. Как правило, в пакеты объединяют классы одной и той же категории или схожие по функционалу.

Первый пакет называется «configuration» он содержит в себе класс, который отвечает за подключение к базе данных. Для начала мы задаем приватные поля класса с модификатором final т.к. эти поля будут неизменны на протяжении всей работы программы и поля connection и DBconnection.

На листинге 2.1 указаны значения для подключения к базе данных

Листинг 2.1: Значения для подключения к БД

```
private static final String url = "jdbc:postgresql://127.0.0.1:5432/diplom";
private static final String user = "postgres";
private static final String password = "qwe";
private Connection connection = null;
private static DBConnection dbConnection;
```

Листинг 2.1: Конец

После этого идет метод непосредственно для подключения к базе данных. В блоке try описано подключение к базе данных, при помощи полей, которые мы инициализировали выше, а в блоке catch e – экземпляр класса Exception. printStackTrace () - метод, определенный в классе Exception и

используется, чтобы печатать информацию относительно исключения, то есть, как оно произошло и какой строке кода.

На листинге 2.2 показан метод, который реализует подключение к БД

Листинг 2.2: Метод реализующий подключение к БД

```
private DBConnection () {
      try {
         Class.forName("org.postgresql.Driver");
         connection = DriverManager.getConnection(url, user, password);
      } catch (ClassNotFoundException | SQLException e) {
         e.printStackTrace();
      }
   }
```

Листинг 2.2: Конец

Следующий пакет «DAO». При проектировании информационной системы выявляются некоторые слои, которые отвечают за взаимодействие различных модулей системы. Соединение с базой данных является одной из важнейшей составляющей приложения. Всегда выделяется часть кода, модуль, отвечающий за передачу запросов в БД и обработку полученных от неё ответов. В общем случае, определение DAO(data access object) описывает его как прослойку между БД и системой. Вершиной иерархии DAO является абстрактный класс или интерфейс с описанием общих методов, которые будут использоваться при взаимодействии с базой данных. Как правило, это методы поиска, удаление по ключу, обновление и т.д. На листинге 2.3 изображен этот самый интерфейс. Методы, которые есть в этом интерфейсе необходимо будет описывать в каждом классе, который является наследником интерфейса DAO.

Листинг 2.3: Методы интерфейса DAO

```
package dao;
import java.util.List;
public interface Dao {
    List getAll();
    void insert(Object entity);
    void update(Object entity);
    void delete(long id);
    Object getById(long id);
}
```

Листинг 2.3: Конец

И для обзора возьмем только один класс из пакета «DAO» т.к. все остальные сделаны по образу и подобию.

Следующий метод сделан для получения всех записей из таблицы «Absence». В боке try создается ArrayList в который заносятся все записи из таблицы при помощи сеттеров, которые реализована в модели «Absence». В блоке catch вывод ошибки и места где она была, в случае ее наличия, при помощи объекта класса Exception и функции printStackTrace().

На листинге 2.4 изображен метод, который отвечает за получение всех записей из таблицы базы данных

Листинг 2.4: Получение всех записей из таблицы БД

```
public List getAll() {

    Dao workerDao = WorkerDao.getDao();
    List<Absence> absences = new ArrayList<>();
    String sql = "SELECT * FROM absence";
    try {
        Statement statement = connection.createStatement();

    ResultSet resultSet = statement.executeQuery(sql);
        while (resultSet.next()) {
            Absence absence = new Absence();
            absence.setId_absence(resultSet.getLong(1));

absence.setCause(resultSet.getString(2));
```

Листинг 2.4: Продолжение

```
absence.setDate_absence(resultSet.getDate(3));;
absence.setWorker((Worker) workerDao.getById(resultSet.getLong(4)));

absences.add(absence);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return absences;
    }
```

Листинг 2.4: Конец

Добавление записей реализовано при помощи следующего метода, который показан на листинге 2.5. В блоке try инициализирована переменная sql, в которой записан sql запрос для добавления записей в таблицу и при помощи сеттеров заносятся записи в сущность.

Листинг 2.5: Добавление записи

```
public void insert(Object entity) {

        Absence absence = (Absence) entity;
        String sql = "INSERT INTO public.absence(\n" +
                "            cause, date_absence, id_worker)\n" +
                "    VALUES (?, ?, ?);\n";
        try {
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, absence.getCause());
statement.setDate(2,  new java.sql.Date(absence.getDate_absence().getTime()));
statement.setLong(3, absence.getWorker().getId_worker());
            statement.execute();
        } catch (SQLException e) {

            e.printStackTrace();

        }
```

Листинг 2.5: Конец

Метод для удаления записей показан на листинге 2.6. В нем удаляется запись с id, который равен выбранному. Каким образом это сделано будет описано ниже

Листинг 2.6: Удаление записей

```
        public void delete(long id) {
      String sql = "DELETE FROM absence WHERE id_absence = ?";
      try {
         PreparedStatement statement =  connection.prepareStatement(sql);

statement.setLong(1, id);
         statement.execute();
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
```

Листинг 2.6: Конец

На этом рассмотрения класса AbsenceDao окончено. Следующий пакет Model. В классах пакета Model реализованы так называемые методы чтения и записи, то есть геттеры и сеттеры. Приводить примеры их листингов нет смысла т.к. это вполне стандартные методы для объектно-ориентированных программ.

Теперь рассмотрим пакет View. В этом пакете хранятся fxml файлы, которые построены с помощью визуального инструмента дизайна Scene Builder. Примеры этих файлов подробно рассматривать нет смысла, их содержание можно увидеть в приложении.

Последний пакет – это Controller, он отвечает за взаимодействие между пользователем и программой. Начинается все с инициализации элементов, которые созданы при помощи Scene Builder в fxml файле. На листинге 2.7 показана инициализация компонентов Scene Builder.

Листинг 2.7: Инициализация компонентов Scene Builder

```
public TableView absenceTable;
    public ComboBox absenceList;
    public TextField cause;
    public DatePicker date_absence;
    public TextField searchAbsence;
```

Листинг 2.7: Конец

В классах пакета Controller реализован поиск по таблице. При вводе в поле для поиска searchAbsence программа ищет по нескольким полям таким как: причина и имя работника, для фильтрации записей по основным параметрам. Поиск происходит динамически это значит, что не нужно обновлять программу для того что бы увидеть результаты поиска. На листинге 2.8 показана реализация поиска.

Листинг 2.8: Поиск по таблице

```
searchAbsence.textProperty().addListener((observable, oldValue, newValue) -> {
        List absenceForSearch = absenceDao.getAll();

        Object[] filtered = absenceForSearch.stream().filter((o) ->
((Absence)o).getCause().toLowerCase().contains(newValue) ||
((Absence)o).getWorker().getFio().toLowerCase().contains(newValue)).toArray();
        absenceTable.setItems(FXCollections.observableArrayList(filtered));
    });
```

Листинг 2.8: Конец

Экспорт в Microsoft Office Excel реализован при помощи библиотеки Apache POI - это набор средств для Java, предоставляющих доступ к чтению файлов и записи в них в форматах таких офисных приложений Microsoft Office, как Word, PowerPoint, Excel, Outlook, Visio и Publisher. Исходный код Apache POI распространяется под лицензией Apache[17]. На листинге 2.9 показана реализация экспорта в Excel.

Листинг 2.9: Обработчик кнопки для экспорта в Excel

```
public void export_absence_excel() {
      try{
         String filename="e:/absence.xls" ;
         HSSFWorkbook hwb = new HSSFWorkbook();
         HSSFSheet sheet =  hwb.createSheet("Премии");
         HSSFRow rowhead=   sheet.createRow(0);
         rowhead.createCell((short) 0).setCellValue("Работник");
 rowhead.createCell((short) 1).setCellValue("Причина");
         rowhead.createCell((short) 2).setCellValue("Дата пропуска");
         //sheet.setZoom(150);
         int i = 1;
         ObservableList absenceTableItem = absenceTable.getItems();
         for (Object item : absenceTableItem) {
            Absence absence = (Absence) item;
            HSSFRow row=   sheet.createRow(i);
            row.createCell(0).setCellValue(absence.getWorker().getFio());
            row.createCell(1).setCellValue(absence.getCause());
            row.createCell(2).setCellValue(new
java.sql.Date(absence.getDate_absence().getTime()));
         }
         for(int j = 0; j <= 6; j++)
            sheet.autoSizeColumn(j);
         FileOutputStream fileOut =  new FileOutputStream(filename);
         hwb.write(fileOut);
         fileOut.close();
         Alert alert = new Alert(Alert.AlertType.INFORMATION);
         alert.setTitle("Окно оповещения");
         alert.setContentText("Данные были экспортированы в Excel");
         alert.showAndWait();

      } catch ( Exception ex ) {
         Alert alert = new Alert(Alert.AlertType.INFORMATION);
         alert.setTitle("Окно оповещения");
         alert.setContentText("Ошибка при экспорте данных");
         alert.showAndWait();
```

Листинг 2.9: Конец

В переменной filename хранится путь для сохранения Excel файла. Далее создается объект класса HSSFWorkbook поля и методы, которого описаны в самой библиотеке ApachePOI. После этого задаем название полей

в таблице Excel. И создаем ObservableList, который содержит все записи таблицы Absence. Далее в цикле for заполняем поля таблицы Excel. И при помощи функции autoSizeColumn() подстраиваем размер ячейки Excel под размер ее содержимого. В случае успеха будет выведено окно с оповещением о результате. В блоке catch реализован вывод сообщения в случае ошибки экспорта.

Далее рассмотрим, каким образом идет расчет заработной платы для работников со сдельно-премиальной формамой оплаты. На листинге 2.10 показана формула расчета заработной платы.

Листинг 2.10: Формула расчета заработной платы

```
if (worker.getProfession().getForm_pay().equals("Сдельно-премиальная")) {
        double dayCost = worker.getTarrif_rate() * dayHours;
        double nightCost = worker.getTarrif_rate() * 2 * nightHours;
        double permiumPercent = 25 - 2.5 * proisvPercent;
        if (brakPercent <= 7) {
           permiumPercent += (7 - brakPercent) * 2;
        } else {
           permiumPercent += (brakPercent - 7) * 5;
        }
        double premium = (worker.getTarrif_rate() * (dayHours + nightHours) * (permiumPercent/100));
        double tax = (dayCost + nightCost)/100*13;
        cost = dayCost + nightCost + premium + premiumSum - finesSum + vacationSum + hospitalSum - tax;
}
```

Листинг 2.10: Конец

Переменные premumSum, finesSum, vacationSum, hospitalsSum содержат сумму всех премий, штрафов, отпускных и больничных за месяц, за который выполняется расчет соответственно.

Так же стоить уделить внимание контроллеру, который отвечает за авторизацию и разграничение прав доступа, то есть за роли. Приложение создает объект Worker и проверяет наличие введенных в окне авторизации данных и их правильность ввода. В случае если поля для ввода заполнены и

заполнены правильно, тогда пользователь переходит на основной экран программы. Если нет, то будет выведено сообщение об ошибке. На листинге 2.11 представлен метод, отвечающий за авторизацию.

Листинг 2.11: Метод для авторизации

```
private void authorization(String loginField, String passField) throws IOException
{
        final String login = loginField;
        final String password = passField;
        Dao workerDao = WorkerDao.getDao();
        List<Worker> workers = (List<Worker>)
workerDao.getAll().parallelStream()
                .filter((o) -> {
                    Worker w = (Worker) o;
                    return w.getLogin() != null && w.getPassword_worker() != null &&
w.getLogin().equals(login) && w.getPassword_worker().equals(password);
                })
                .collect(Collectors.toCollection(ArrayList<Worker>::new));
        if (workers.isEmpty()) {
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Окно оповещения");
            alert.setContentText("Неверный логин или пароль");
            alert.showAndWait();
}else {
            currentUser = workers.get(0);
            authorization_button.getScene().getWindow().hide();
            Parent root = FXMLLoader.load(new
File("src/main/java/view/main.fxml").toURL());
            Stage stage = new Stage();
            stage.setTitle("Расчет заработной платы");
            stage.setScene(new Scene(root, 1400, 950));
            stage.show();
        }
```

Листинг 2.11 Конец

И рассмотрим, каким образом реализовано разграничение ролей и определение прав доступа при авторизации. В приложении создана последовательность(Enum), в которой записаны все 3 роли, при авторизации значение поля role из базы данных и значение из последовательности

сравниваются и в зависимости от результата пользователь видит определенный список работников с данными, которых он может работать.

На листинге 2.12 показано, каким образом формируется список доступных сотрудников.

Листинг 2.12: Проверка на права доступа

```
if (AuthorizationController.currentUser.getRole().equals(UserRole.PART)) {
        partList.setVisible(false);
        workers = (List) workers.stream().filter((w -> ((Worker) w).getPart().equals(AuthorizationController.currentUser.getPart()))).collect(Collectors.toList());
    }
```

Листинг 2.12: Конец

На этом рассмотрение кода приложения будет закончено, полный код будет в Приложении 1.

ГЛАВА 3. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Программа и методика тестирования

Объектом испытаний, рассматриваемым в данной главе, является ранее спроектированная и разработанная «Автоматизированная система расчета трудозатрат для предприятия ЗАО «ОсколНефтеМаш»».

Целью испытаний информационной системы являются проверка ее работоспособности, а так же ее соответствие указанным в первой главе требованиям к автоматизированной системе.

Полный перечень требований к автоматизированной системе перечислен в третьем разделе первой главы, исходя из этих требований, процесс тестирования будет проведен по следующему алгоритму:

- Тестирование авторизации;
- Тестирование системы помощи сотрудникам;
- Тестирование функционала приложения в зависимости от роли, авторизовавшегося сотрудника;
- Тестирование добавления, удаление и изменения записей;
- Тестирование поиска по таблицам;
- Тестирование экспорта данных в Excel;
- Проверка расчетов, выполняемых программой на правильность;
- Тестирование функционала вкладки «Администрирование».

3.2 Разработка интерфейса

Разработка интерфейса осуществлялась при помощи инструмента графического дизайна - Scene Builder, подробнее про данный инструмент можно прочитать во второй главе, пункте 2.1. В этом разделе будут представлен интерфейс основных вкладок в автоматизированной системе.

Интерфейс каждой вкладки описывается в отдельном fxml файле это сделано для более удобного и понятного редактирования и чтения fxml файлов. Основным элементом каждой вкладки (Tab) является TableView, этот компонент создан для отображения данных в виде таблиц, в том числе и сущностей базы данных. Структура каждого Tab состоит из HBox – это контейнер для компонентов, которые будут размещены по горизонтали и VBox – в нем размещаются компоненты по вертикали. В HBox и VBox помещаются компоненты для управления и взаимодействия с пользователем такие как: TextField, DatePicker, ComboBox, Button и т.д. На рисунке 3.1 будет представлен worker.fxml – fxml файл отвечающий за интерфейс вкладки «Сотрудники», открытый с помощью Scene Builder.

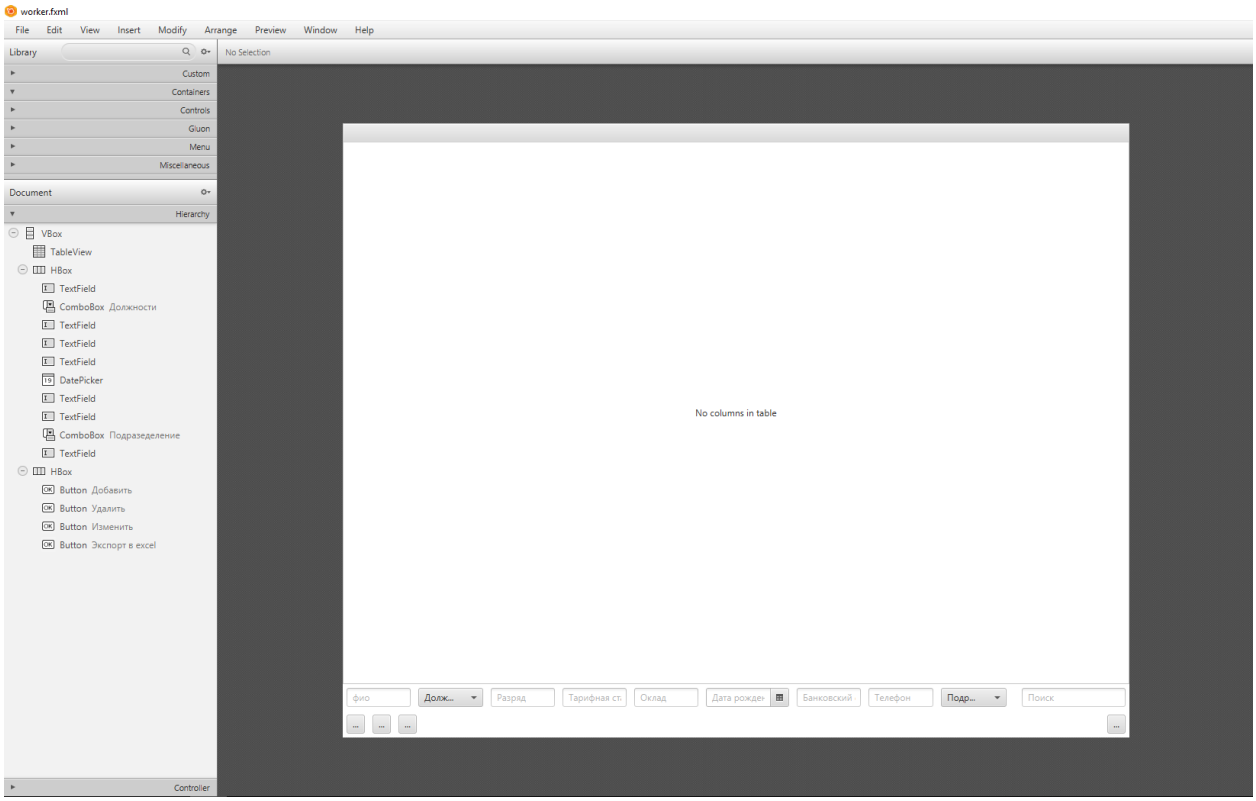


Рис. 3.1. Иерархия компонентов worker.fxml

Таким же образом построены все fxml файлы, отвечающие за отображение таблиц из базы данных. Теперь рассмотрим main.fxml, к которому подключается все сформированные до этого fxml файлы. Каждый

Tab компонента TabPane представляет собой отдельный fxml файл. TabPane “обернут” в AnchorPane, на этом компоненте выполнена шапка с названием предприятия и прочие элементы оформления приложения. На рисунке 3.2 будет представлен main.fxml.

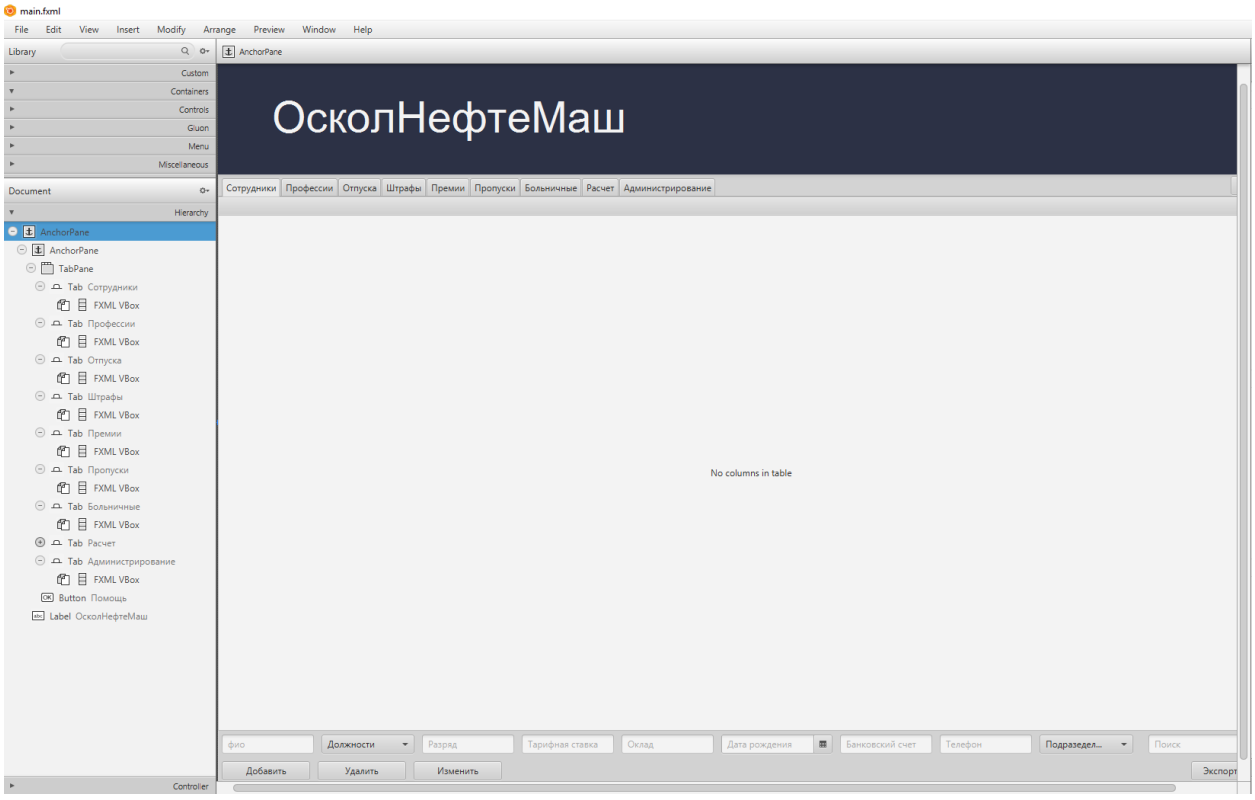


Рис. 3.2. Структура main.fxml

3.3 Тестирование автоматизированной системы

Тестирование стоит начать с авторизации, для проверки правильности разграничения прав доступа авторизируем пользователя имеющего доступ ко всем записям.

На рисунке 3.3 изображено окно авторизации. Оно включает в себя 2 поля для ввода логина и пароля, поле для ввода пароля замещает введенные символы на точки. В результате действий произведенных в этом окне будет осуществляться доступ в приложение или наоборот отказ в доступе и распределение по ролям.

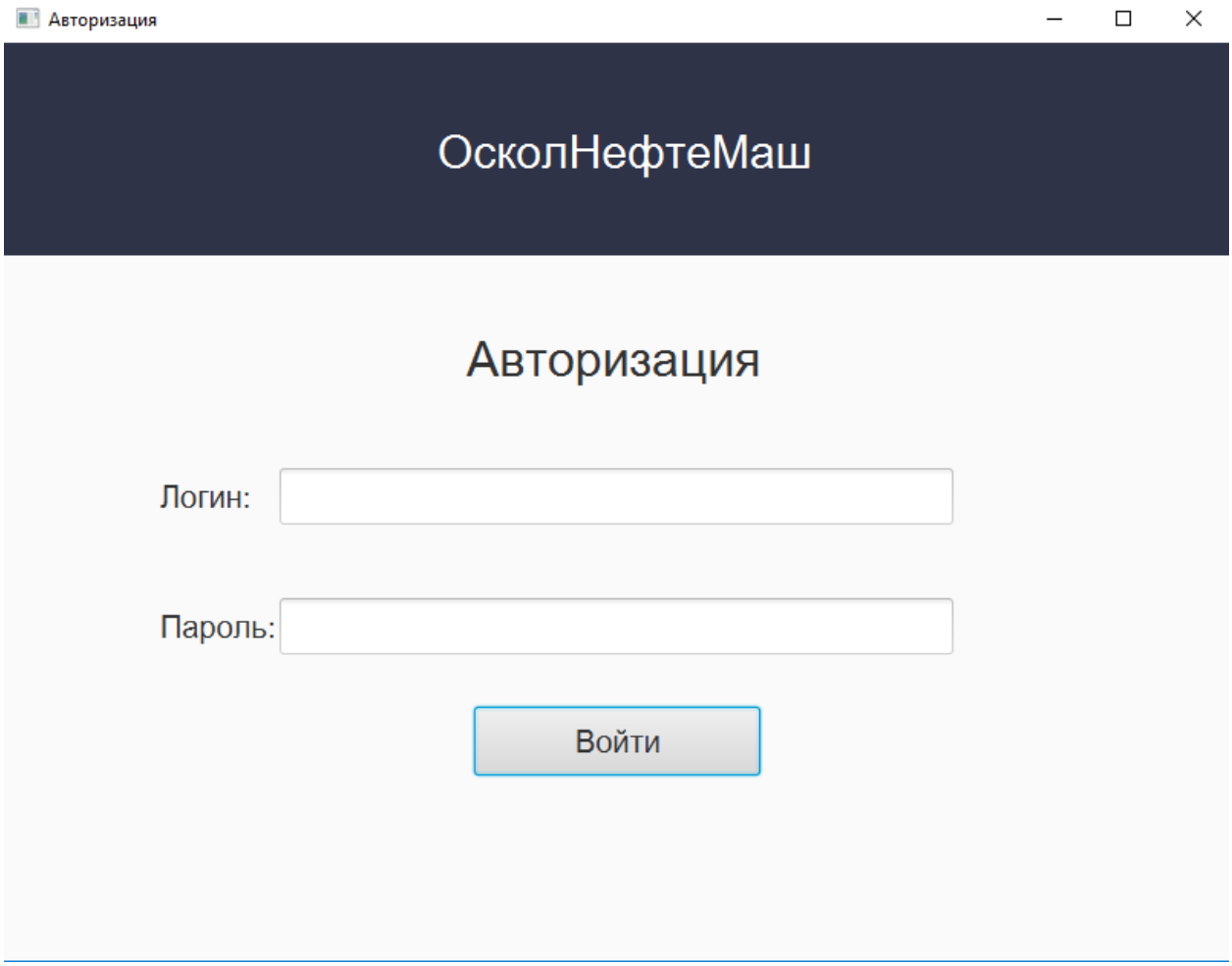


Рис 3.3. Окно авторизации.

На рисунке 3.4 изображено сообщение, которое выводиться в случае ввода неправильного логина или пароля.

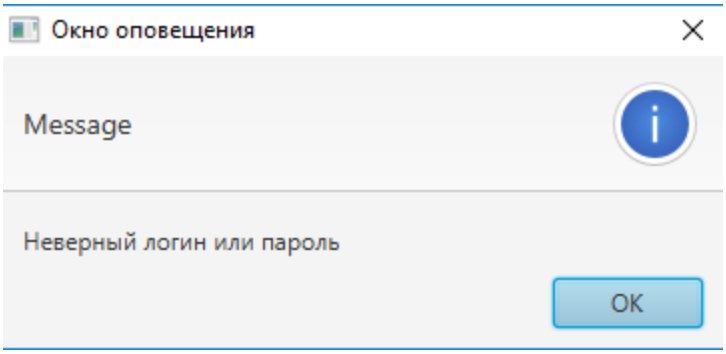


Рис. 3.4. Ошибка при вводе данных.

После входа в приложение рекомендуется ознакомиться с инструкцией по использованию, для этого нужно нажать на кнопку «Помощь». На рисунке 3.5 изображено окно помощи сотрудникам.

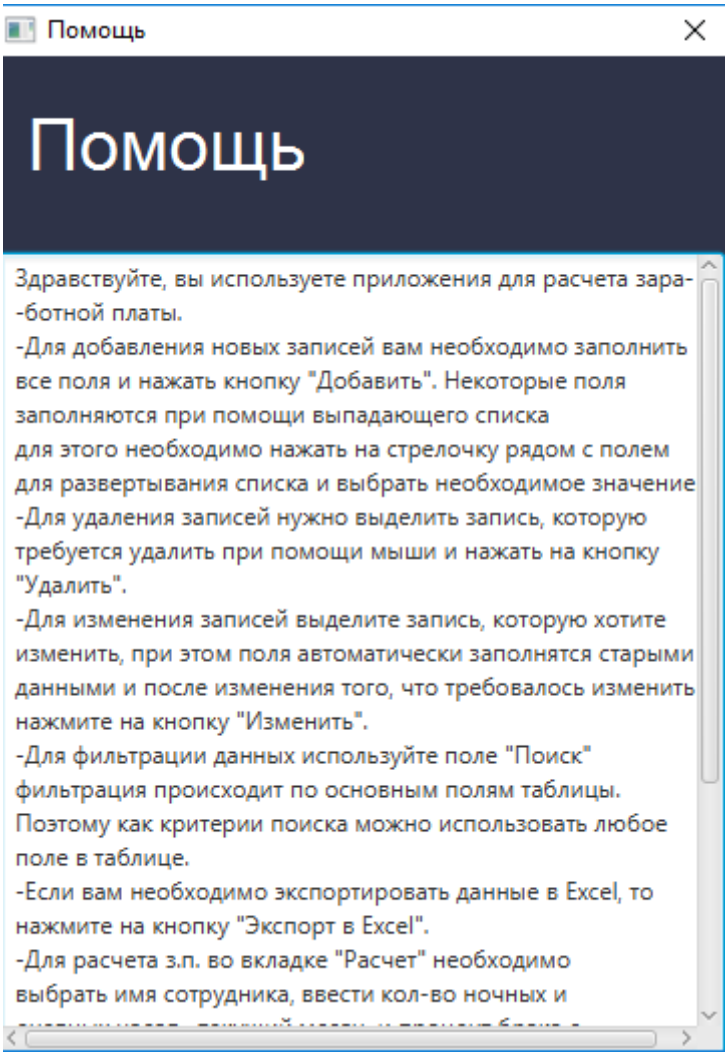


Рисунок 3.5. Окно помощи.

На рисунке 3.6 изображено вкладка «Сотрудники» для пользователя с правами просмотра всех данных (главный бухгалтер), которая служит для вывода всех сотрудников, которые входят в права доступа пользователя, а так же добавления, удаления, изменения и поиска. Поиск возможно осуществить по нескольким полям вкладки, для более удобной фильтрации. В данном случае это поля: ФИО, Название профессии и Подразделение. Так же есть кнопка для экспорта таблицы в Excel, если пользователю нужно

перед экспортом отфильтровать данные, то экспорт будет произведен по результатам поиска, т.е. экспортированы только те поля, которые удовлетворяют условиям фильтрации.

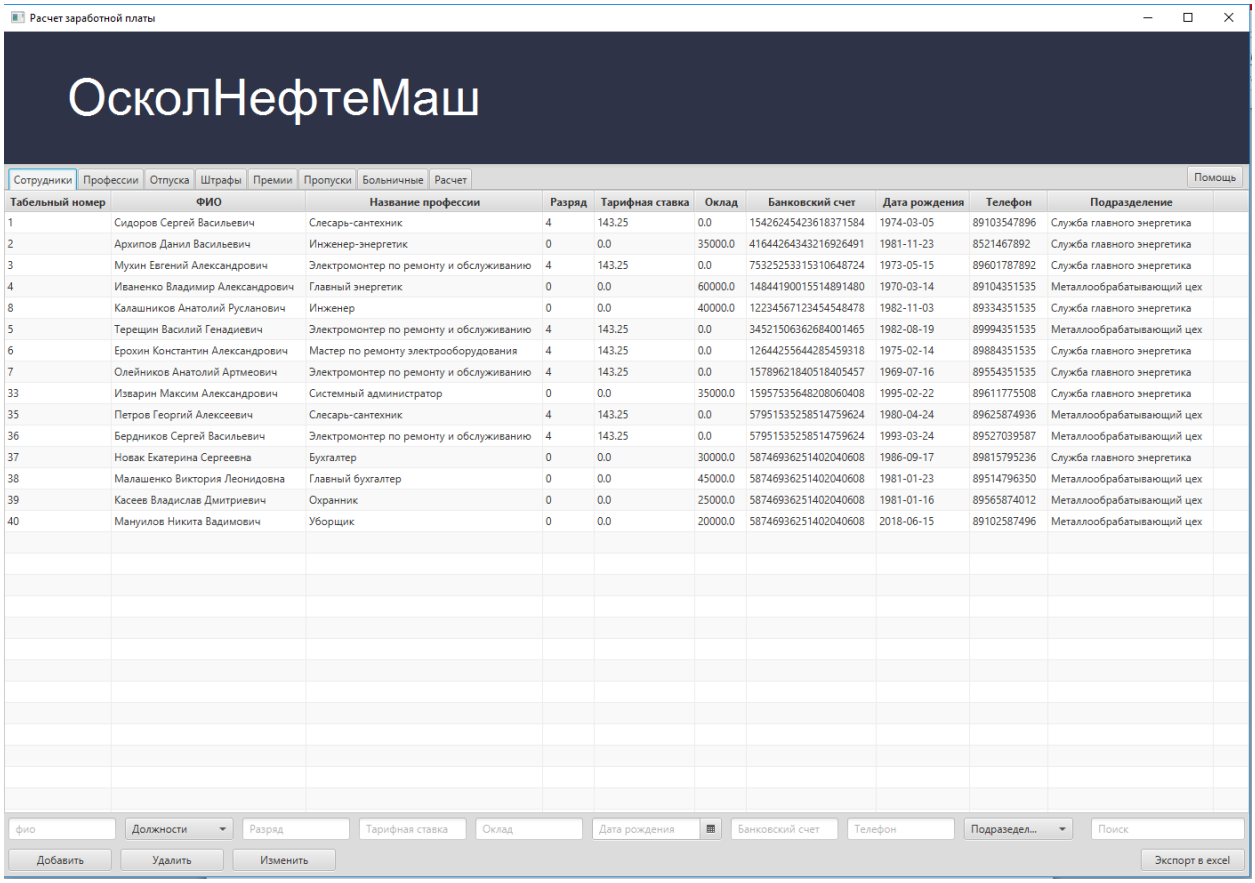
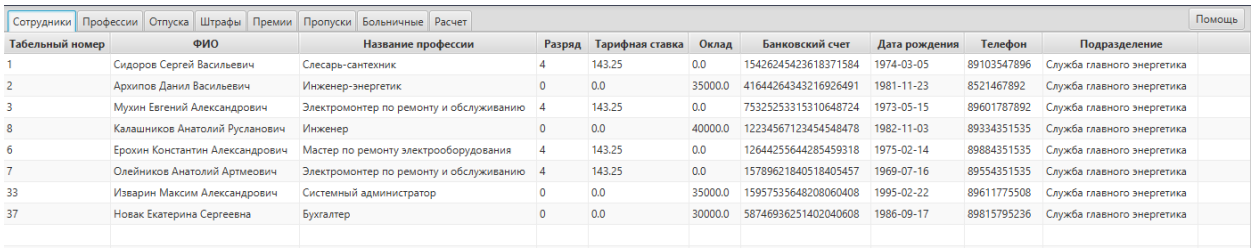


Рис 3.6. Вкладка «Сотрудники» с правами просмотра всех записей.

Теперь протестируем отображение той же вкладки сотрудники, только для частичных прав доступа (бухгалтер). Как видно на рисунке 3.7 теперь отображаются только сотрудники, которые входят в подразделение «Служба главного энергетика» т.е. бухгалтер видит только сотрудников того подразделения к которому относиться и он сам.

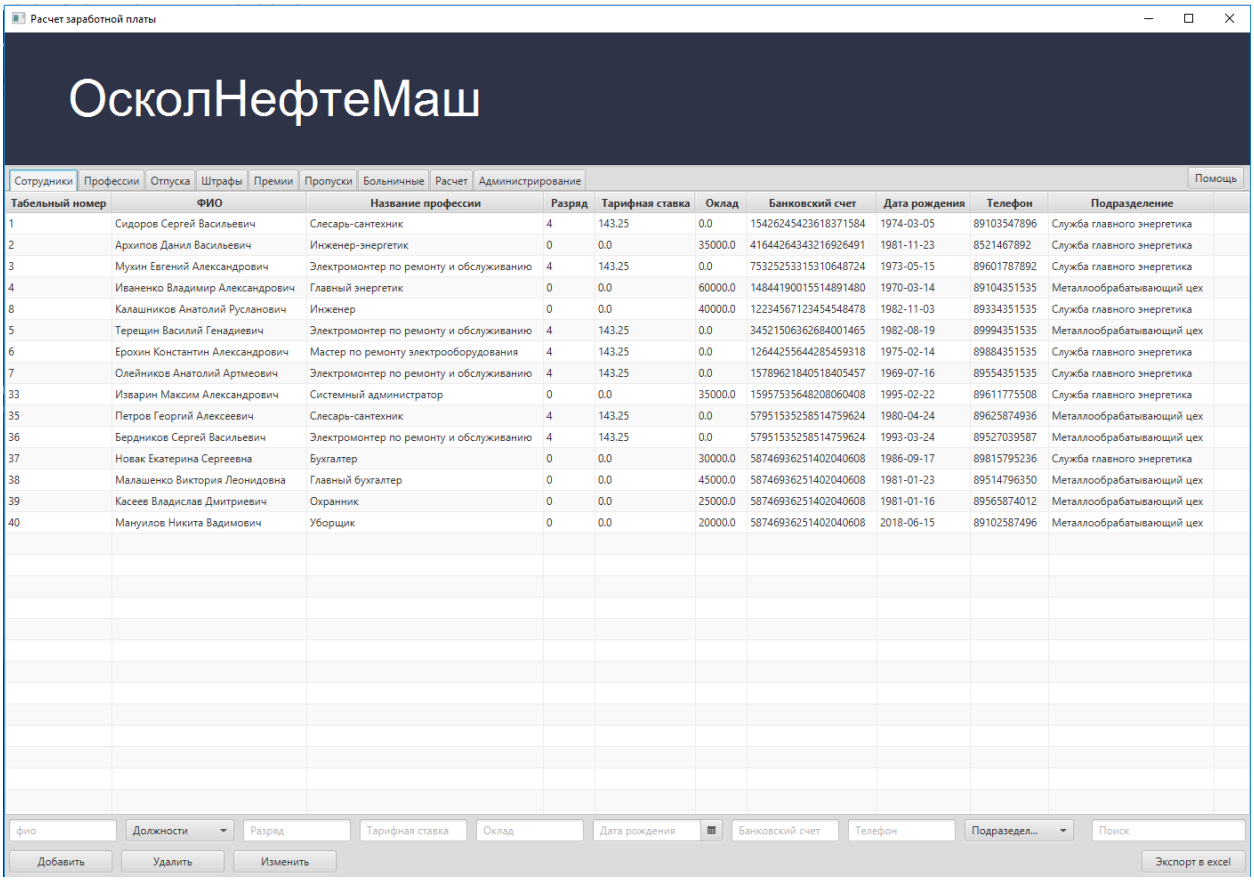


Сотрудники | Профессии | Отпуска | Штрафы | Премии | Пропуски | Больничные | Расчет | Помощь

Табельный номер	ФИО	Название профессии	Разряд	Тарифная ставка	Оклад	Банковский счет	Дата рождения	Телефон	Подразделение
1	Сидоров Сергей Васильевич	Слесарь-сантехник	4	143.25	0.0	15426245423618371584	1974-03-05	89103547896	Служба главного энергетика
2	Архипов Данил Васильевич	Инженер-энергетик	0	0.0	35000.0	41644264343216926491	1981-11-23	8521467892	Служба главного энергетика
3	Мухин Евгений Александрович	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	75325253315310648724	1973-05-15	89601787892	Служба главного энергетика
8	Калашников Анатолий Русланович	Инженер	0	0.0	40000.0	12234567123454548478	1982-11-03	89334351535	Служба главного энергетика
6	Ерохин Константин Александрович	Мастер по ремонту электрооборудования	4	143.25	0.0	12644255644285459318	1975-02-14	89884351535	Служба главного энергетика
7	Олейников Анатолий Артмеович	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	15789621840518405457	1969-07-16	89554351535	Служба главного энергетика
33	Изварин Максим Александрович	Системный администратор	0	0.0	35000.0	15957535648208060408	1995-02-22	89611775508	Служба главного энергетика
37	Новак Екатерина Сергеевна	Бухгалтер	0	0.0	30000.0	58746936251402040608	1986-09-17	89815795236	Служба главного энергетика

Рис 3.7. Отображение таблицы «Сотрудники» для частичных прав доступа.

Теперь протестируем вход с правами доступа администратора. Как видно на рисунке 3.8 при авторизации администратора появляется дополнительная вкладка «Администрирование» ее вид и функционал будет описан ниже.



Расчет заработной платы

ОсколНефтеМаш

Сотрудники | Профессии | Отпуска | Штрафы | Премии | Пропуски | Больничные | Расчет | Администрирование | Помощь

Табельный номер	ФИО	Название профессии	Разряд	Тарифная ставка	Оклад	Банковский счет	Дата рождения	Телефон	Подразделение
1	Сидоров Сергей Васильевич	Слесарь-сантехник	4	143.25	0.0	154262454236183715 84	1974-03-05	89103547896	Служба главного энергетика
2	Архипов Данил Васильевич	Инженер-энергетик	0	0.0	35000.0	41644264343216926491	1981-11-23	8521467892	Служба главного энергетика
3	Мухин Евгений Александрович	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	75325253315310648724	1973-05-15	89601787892	Служба главного энергетика
4	Иваненко Владимир Александрович	Главный энергетик	0	0.0	60000.0	14844190015514891480	1970-03-14	89104351535	Металлообрабатывающий цех
8	Калашников Анатолий Русланович	Инженер	0	0.0	40000.0	12234567123454548478	1982-11-03	89334351535	Служба главного энергетика
5	Терещин Василий Генадиевич	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	34521506362684001465	1982-08-19	89994351535	Металлообрабатывающий цех
6	Ерохин Константин Александрович	Мастер по ремонту электрооборудования	4	143.25	0.0	12644255644285459318	1975-02-14	89884351535	Служба главного энергетика
7	Олейников Анатолий Артмеович	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	15789621840518405457	1969-07-16	89554351535	Служба главного энергетика
33	Изварин Максим Александрович	Системный администратор	0	0.0	35000.0	15957535648208060408	1995-02-22	89611775508	Служба главного энергетика
35	Петров Георгий Алексеевич	Слесарь-сантехник	4	143.25	0.0	57951535258514759624	1980-04-24	89625874936	Металлообрабатывающий цех
36	Бердников Сергей Васильевич	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	57951535258514759624	1993-03-24	89527039587	Металлообрабатывающий цех
37	Новак Екатерина Сергеевна	Бухгалтер	0	0.0	30000.0	58746936251402040608	1986-09-17	89815795236	Служба главного энергетика
38	Малашенко Виктория Леонидовна	Главный бухгалтер	0	0.0	45000.0	58746936251402040608	1981-01-23	89514796350	Металлообрабатывающий цех
39	Касеев Владислав Дмитриевич	Охранник	0	0.0	25000.0	58746936251402040608	1981-01-16	89565874012	Металлообрабатывающий цех
40	Мануилов Никита Вадимович	Уборщик	0	0.0	20000.0	58746936251402040608	2018-06-15	89102587496	Металлообрабатывающий цех

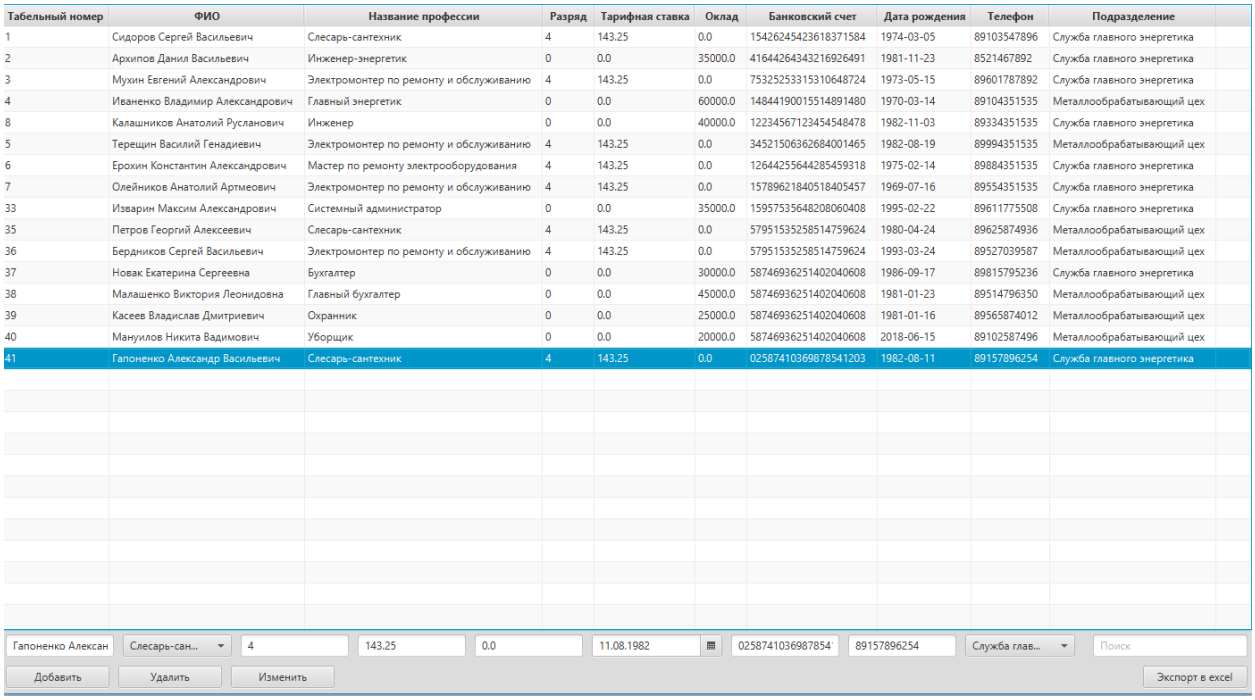
фио | Должности | Разряд | Тарифная ставка | Оклад | Дата рождения | Банковский счет | Телефон | Подразедел... | Поиск

Добавить | Удалить | Изменить | Экспорт в excel

Рис. 3.8. Вход с правами доступа администратора.

На вкладке «Сотрудники» можно добавлять, изменять, удалять записи, а так же производить поиск записей и экспорт в MS Excel. Необходимо протестировать каждую из этих возможностей.

Для добавления записи нужно заполнить все поля и нажать на кнопку «Добавить». На рисунке 3.9 будут изображены последствия добавления нового сотрудника.



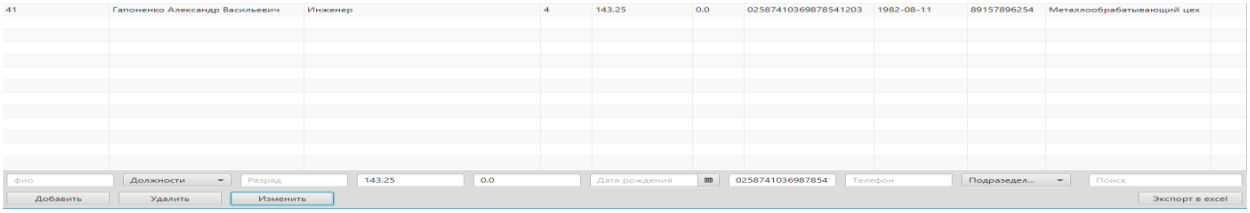
Табельный номер	ФИО	Название профессии	Разряд	Тарифная ставка	Оклад	Банковский счет	Дата рождения	Телефон	Подразделение
1	Сидоров Сергей Васильевич	Слесарь-сантехник	4	143.25	0.0	15426245423618371584	1974-03-05	89103547896	Служба главного энергетика
2	Архипов Данил Васильевич	Инженер-энергетик	0	0.0	35000.0	41644264343216926491	1981-11-23	8521467892	Служба главного энергетика
3	Мухин Евгений Александрович	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	75325253315310648724	1973-05-15	89601787892	Служба главного энергетика
4	Иваненко Владимир Александрович	Главный энергетик	0	0.0	60000.0	14844190015514891480	1970-03-14	89104351535	Металлообрабатывающий цех
8	Калашников Анатолий Русланович	Инженер	0	0.0	40000.0	12234567123454548478	1982-11-03	89334351535	Служба главного энергетика
5	Терещин Василий Генадиевич	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	34521506362684001465	1982-08-19	89994351535	Металлообрабатывающий цех
6	Ерохин Константин Александрович	Мастер по ремонту электрооборудования	4	143.25	0.0	12644255644285459318	1975-02-14	89884351535	Служба главного энергетика
7	Олейников Анатолий Артмеович	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	15789621840518405457	1969-07-16	89554351535	Служба главного энергетика
33	Изварин Максим Александрович	Системный администратор	0	0.0	35000.0	15957535648208060408	1995-02-22	89611775508	Служба главного энергетика
35	Петров Георгий Алексеевич	Слесарь-сантехник	4	143.25	0.0	57951535258514759624	1980-04-24	89625874936	Металлообрабатывающий цех
36	Бердников Сергей Васильевич	Электромонтер по ремонту и обслуживанию	4	143.25	0.0	57951535258514759624	1993-03-24	89527039587	Металлообрабатывающий цех
37	Новак Екатерина Сергеевна	Бухгалтер	0	0.0	30000.0	58746936251402040608	1986-09-17	89815795236	Служба главного энергетика
38	Малашенко Виктория Леонидовна	Главный бухгалтер	0	0.0	45000.0	58746936251402040608	1981-01-23	89514796350	Металлообрабатывающий цех
39	Касеев Владислав Дмитриевич	Охранник	0	0.0	25000.0	58746936251402040608	1981-01-16	89565874012	Металлообрабатывающий цех
40	Мануилов Никита Вадимович	Уборщик	0	0.0	20000.0	58746936251402040608	2018-06-15	89102587496	Металлообрабатывающий цех
41	Гапоненко Александр Васильевич	Слесарь-сантехник	4	143.25	0.0	02587410369878541203	1982-08-11	89157896254	Служба главного энергетика

Гапоненко Алексан | Слесарь-сан... | 4 | 143.25 | 0.0 | 11.08.1982 | 0258741036987854' | 89157896254 | Служба глав... | Поиск

Добавить | Удалить | Изменить | Экспорт в excel

Рис. 3.9. Добавление сотрудника.

Далее изменим профессию и подразделение добавленного сотрудника, дабы протестировать изменение записей. На рисунке 3.10 показано изменение данных сотрудника.



41	Гапоненко Александр Васильевич	Инженер	4	143.25	0.0	02587410369878541203	1982-08-11	89157896254	Металлообрабатывающий цех

фио | Должности | Разряд | 143.25 | 0.0 | Дата рождения | 0258741036987854' | Телефон | Подразедел... | Поиск

Добавить | Удалить | Изменить | Экспорт в excel

Рис. 3.10. Изменение записи.

И теперь удалим измененную запись для проверки работы удаления записей. На рисунке 3.11. будут изображены последствия удаления записей.

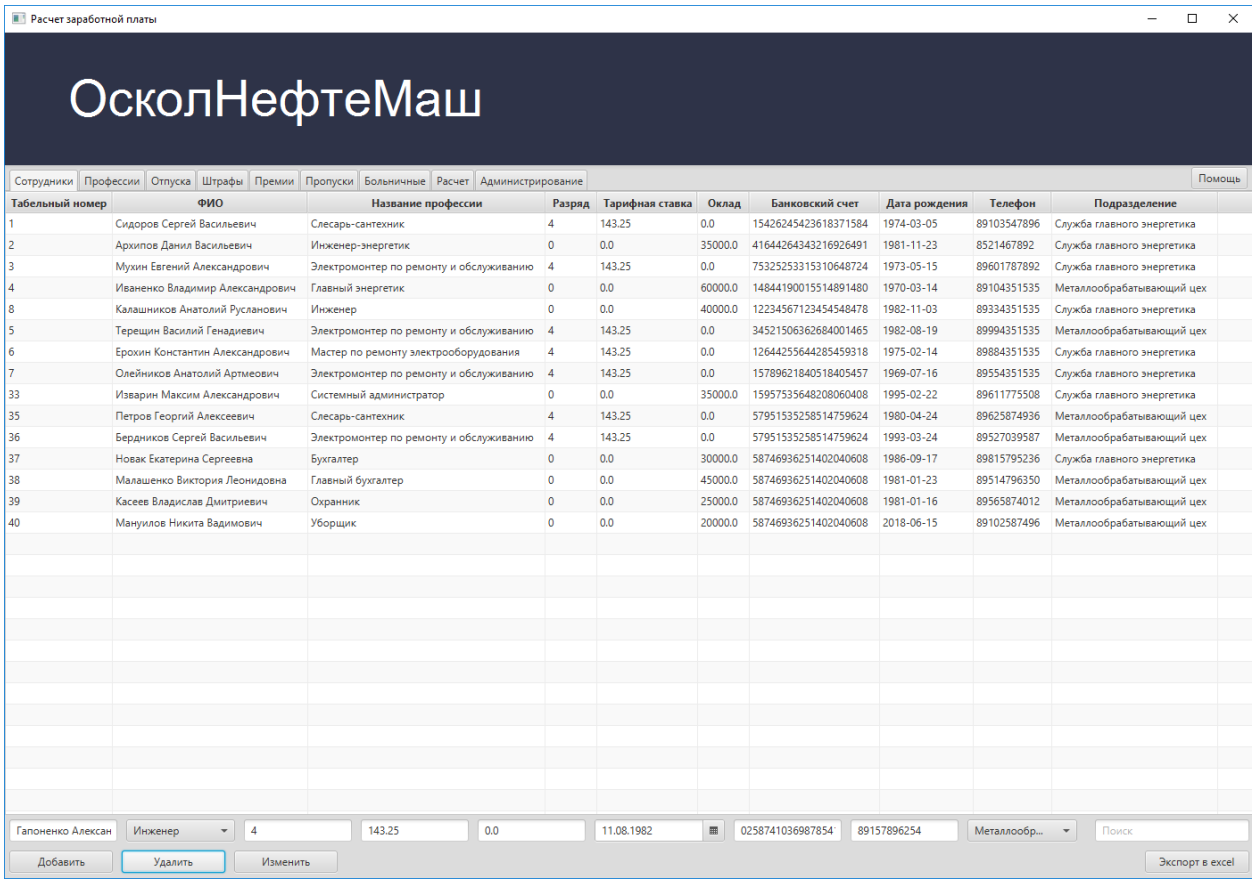


Рис. 3.11. Удаление записей

Так же необходимо протестировать работу поиск. Поиск в программе реализован таким образом, что он ведется по нескольким полям таблицы для более удобной фильтрации, при вводе текста в поле поиска программа “понимает” как маленький, так и большой регистр. В данном случае это поля: «ФИО», «Название профессии» и «Подразделение». Проверим работу поиска по каждому из полей.

На рисунке 3.12 будет показан поиск по полю «ФИО».

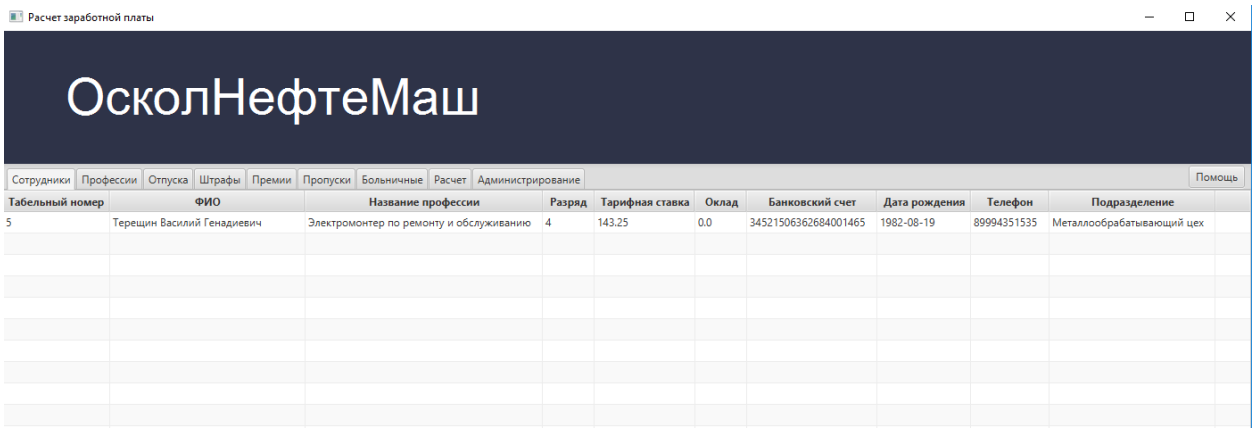


Рис. 3.12. Поиск по полю «ФИО».

Теперь проверим фильтрацию записей по полю «Название профессии». Как видно на рисунке 3.13 программа выводит всех сотрудников с профессией, введенной в поле поиска.

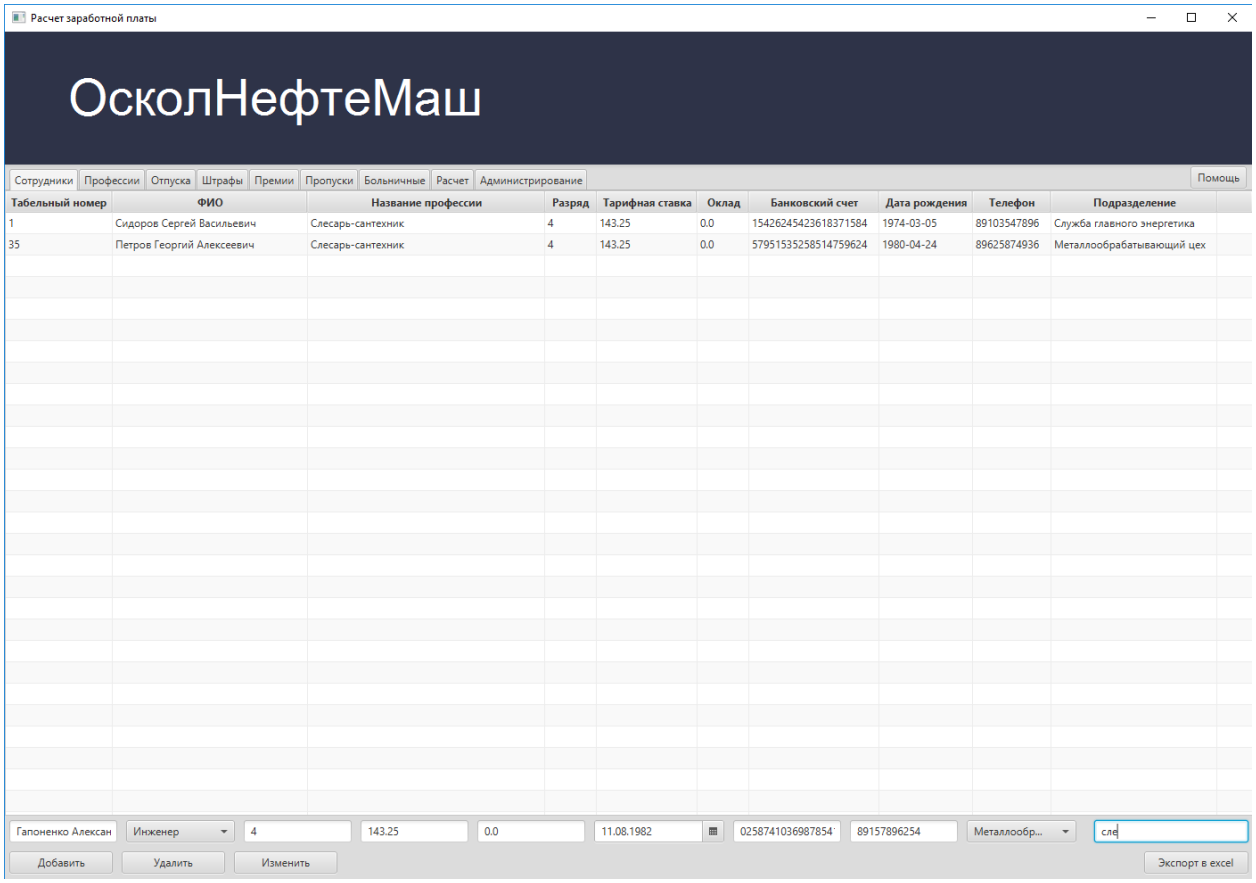


Рис. 3.13. Поиск по полю «Название профессии».

Осталось проверить поиск по полю «Подразделение». На рисунке 3.14 показаны результаты поиска по полю «Подразделение».

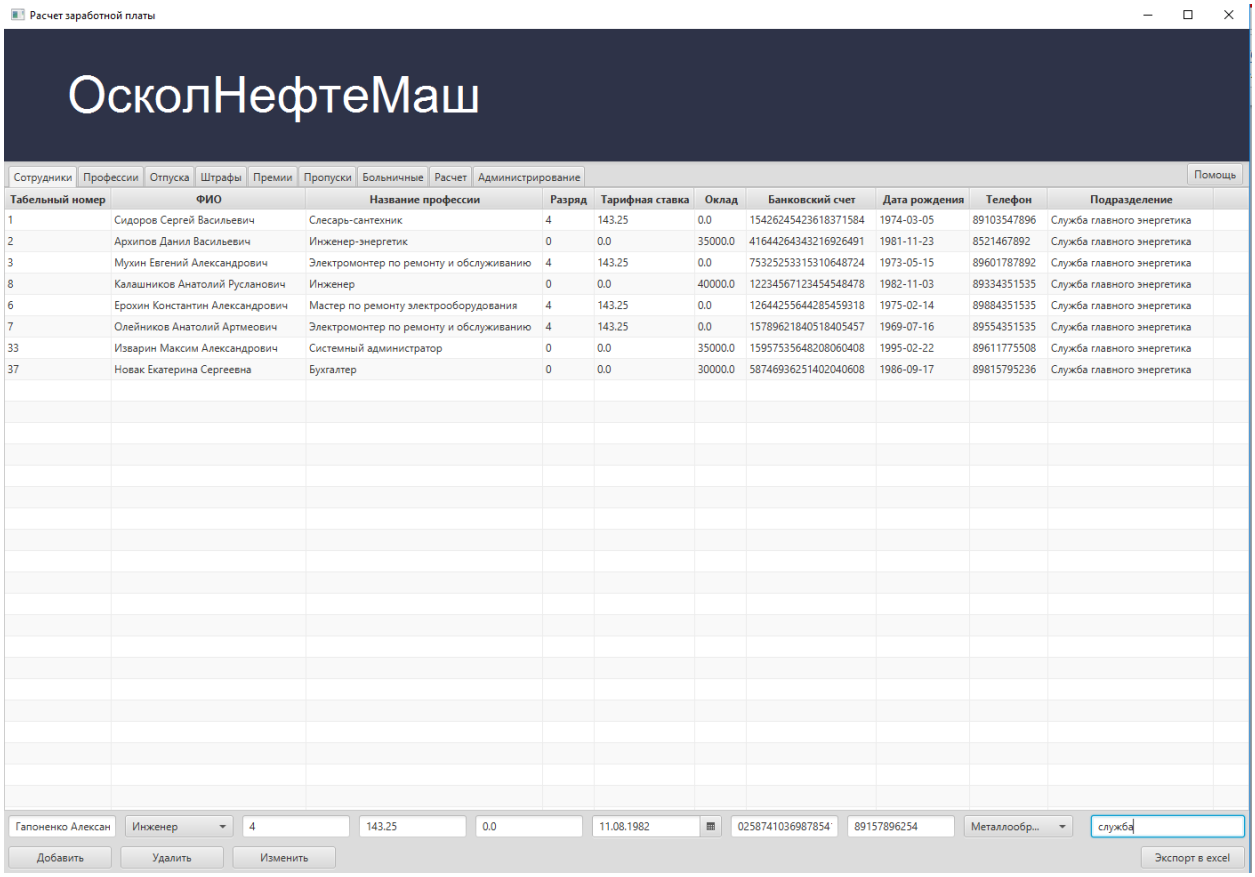


Рис. 3.14. Поиск по полю «Подразделение»

С тестированием поиска закончено. Теперь перейдем к проверке экспорта в MS Excel. На 3.15 будет показано сообщение, которое выводиться при ошибке экспорта.

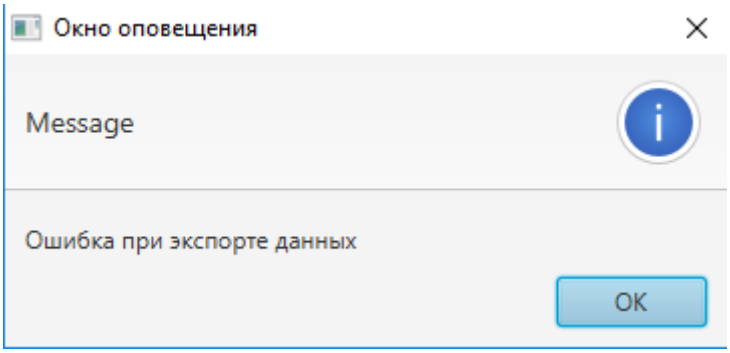
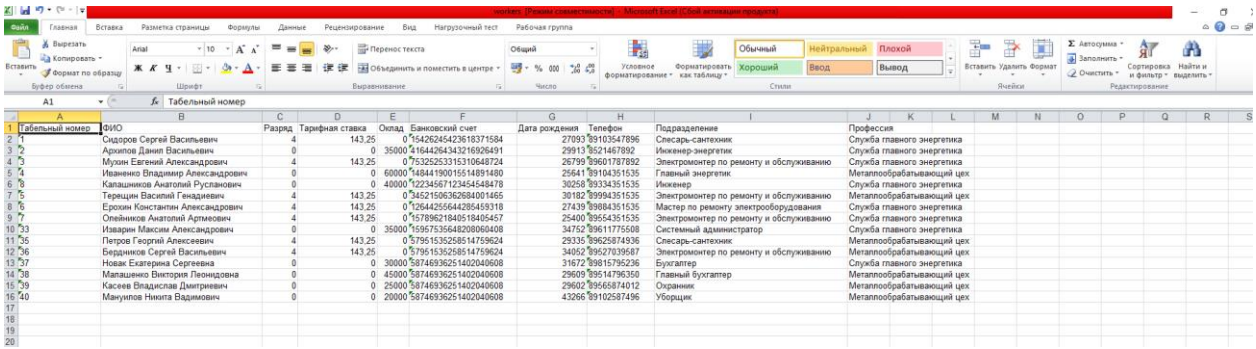


Рис 3.15. Сообщение при ошибке экспорта.

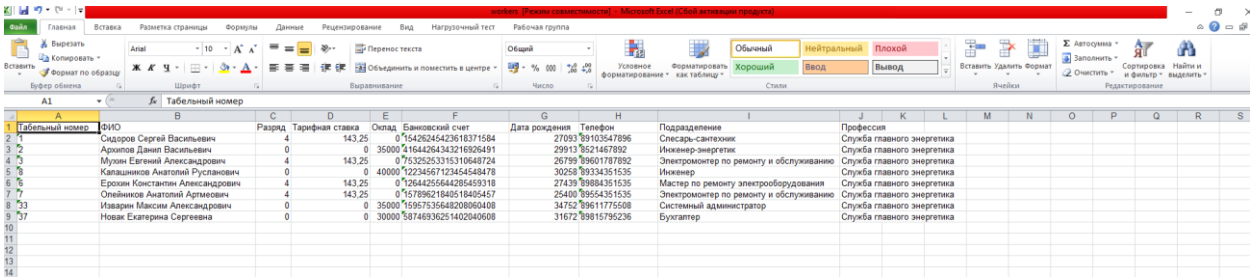
Теперь приведем пример документа, который получился в результате экспорта в MS Excel. На рисунке 3.16 будет приведен Excel документ, который получился в результате экспорта.



Табельный номер	ФИО	Разряд	Тарифная ставка	Оклад	Банковский счет	Дата рождения	Телефон	Подразделение	Профессия
1	Сидоров Сергей Васильевич	4	143,25	0	15426245423618371584	27093	89103547896	Слесарь-сантехник	Служба главного энергетика
2	Архипов Данил Васильевич	0	0	35000	41644264343216926491	29913	8521467892	Инженер-энергетик	Служба главного энергетика
5	Мухин Евгений Александрович	4	143,25	0	75325253315310648724	26799	89601787892	Электромонтер по ремонту и обслуживанию	Служба главного энергетика
4	Иваненко Владимир Александрович	0	0	60000	14844190015514891480	25641	89104351535	Главный энергетик	Металлообрабатывающий цех
8	Капашников Анатолий Русланович	0	0	40000	12234567123454548478	30258	89334351535	Инженер	Служба главного энергетика
5	Терещин Василий Генадиевич	4	143,25	0	34521506362684001465	30182	89994351535	Электромонтер по ремонту и обслуживанию	Металлообрабатывающий цех
6	Ерохин Константин Александрович	4	143,25	0	12644255644285459318	27439	89884351535	Мастер по ремонту электрооборудования	Служба главного энергетика
7	Олейников Анатолий Артмеович	4	143,25	0	15789621840518405457	25400	89554351535	Электромонтер по ремонту и обслуживанию	Служба главного энергетика
33	Изварин Максим Александрович	0	0	35000	15957535648208060408	34752	89611775508	Системный администратор	Служба главного энергетика
35	Петров Георгий Алексеевич	4	143,25	0	57951535258514759624	29335	89625874936	Слесарь-сантехник	Металлообрабатывающий цех
36	Бердников Сергей Васильевич	4	143,25	0	57951535258514759624	34052	89527039587	Электромонтер по ремонту и обслуживанию	Металлообрабатывающий цех
37	Новак Екатерина Сергеевна	0	0	30000	58746936251402040608	31672	89815795236	Бухгалтер	Служба главного энергетика
38	Малащенко Виктория Леонидовна	0	0	45000	58746936251402040608	29609	89614796350	Главный бухгалтер	Металлообрабатывающий цех
39	Касеев Владислав Дмитриевич	0	0	25000	58746936251402040608	29602	89668740412	Охранник	Металлообрабатывающий цех
40	Макулов Никита Вадимович	0	0	20000	58746936251402040608	43266	89102587496	Уборщик	Металлообрабатывающий цех

Рис. 3.16. Экспорт в Excel.

Так же в программе существует возможность экспортировать лишь результат поиска, если пользователю нет нужды экспортировать всю таблицу. На рисунке 3.17 будет показан экспорт сотрудников, которые состоят в подразделении «Служба главного энергетика».



Табельный номер	ФИО	Разряд	Тарифная ставка	Оклад	Банковский счет	Дата рождения	Телефон	Подразделение	Профессия
1	Сидоров Сергей Васильевич	4	143,25	0	15426245423618371584	27093	89103547896	Слесарь-сантехник	Служба главного энергетика
2	Архипов Данил Васильевич	0	0	35000	41644264343216926491	29913	8521467892	Инженер-энергетик	Служба главного энергетика
5	Мухин Евгений Александрович	4	143,25	0	75325253315310648724	26799	89601787892	Электромонтер по ремонту и обслуживанию	Служба главного энергетика
8	Капашников Анатолий Русланович	0	0	40000	12234567123454548478	30258	89334351535	Инженер	Служба главного энергетика
6	Ерохин Константин Александрович	4	143,25	0	12644255644285459318	27439	89884351535	Мастер по ремонту электрооборудования	Служба главного энергетика
7	Олейников Анатолий Артмеович	4	143,25	0	15789621840518405457	25400	89554351535	Электромонтер по ремонту и обслуживанию	Служба главного энергетика
33	Изварин Максим Александрович	0	0	35000	15957535648208060408	34752	89611775508	Системный администратор	Служба главного энергетика
37	Новак Екатерина Сергеевна	0	0	30000	58746936251402040608	31672	89815795236	Бухгалтер	Служба главного энергетика

Рис. 3.17. Экспорт результатов поиска.

Выше были описаны общие функции для всех таблиц, которые служат для отображения базы данных. Поэтому описывать и тестировать каждую из них не имеет смысла. Теперь перейдем к тестированию расчета заработной платы на основе трудозатрат. Для начала проверим правильность расчета заработной платы для сотрудников со сдельно-премиальной формой оплаты.

Подробно методика расчета заработной платы описана в 1 главе, пункте 1.1.1. Ниже будет приведен расчет заработной платы для сотрудника Терещин Василий Геннадиевич.

Cost1 = 143.25 * 80 * 1.25 + 1000 – 600 + 2400 + 2121 = 19246 (рублей). (1)

Это сумма без учета налогового вычета, который равен 13%. Теперь нужно вычислить размер налогового вычета.

Tax = 19246 / 100 * 13 = 2501.98 (рублей). (2)

И для получения итоговой суммы необходимо вычесть из Cost1 значение Tax.

Cost = 19246 – 2501.98 = 16744.02 (рублей). (3)

Далее сравним результаты расчетов, с результатами расчетов программы. На рисунке 3.18 будут изображены результаты расчетов программы.

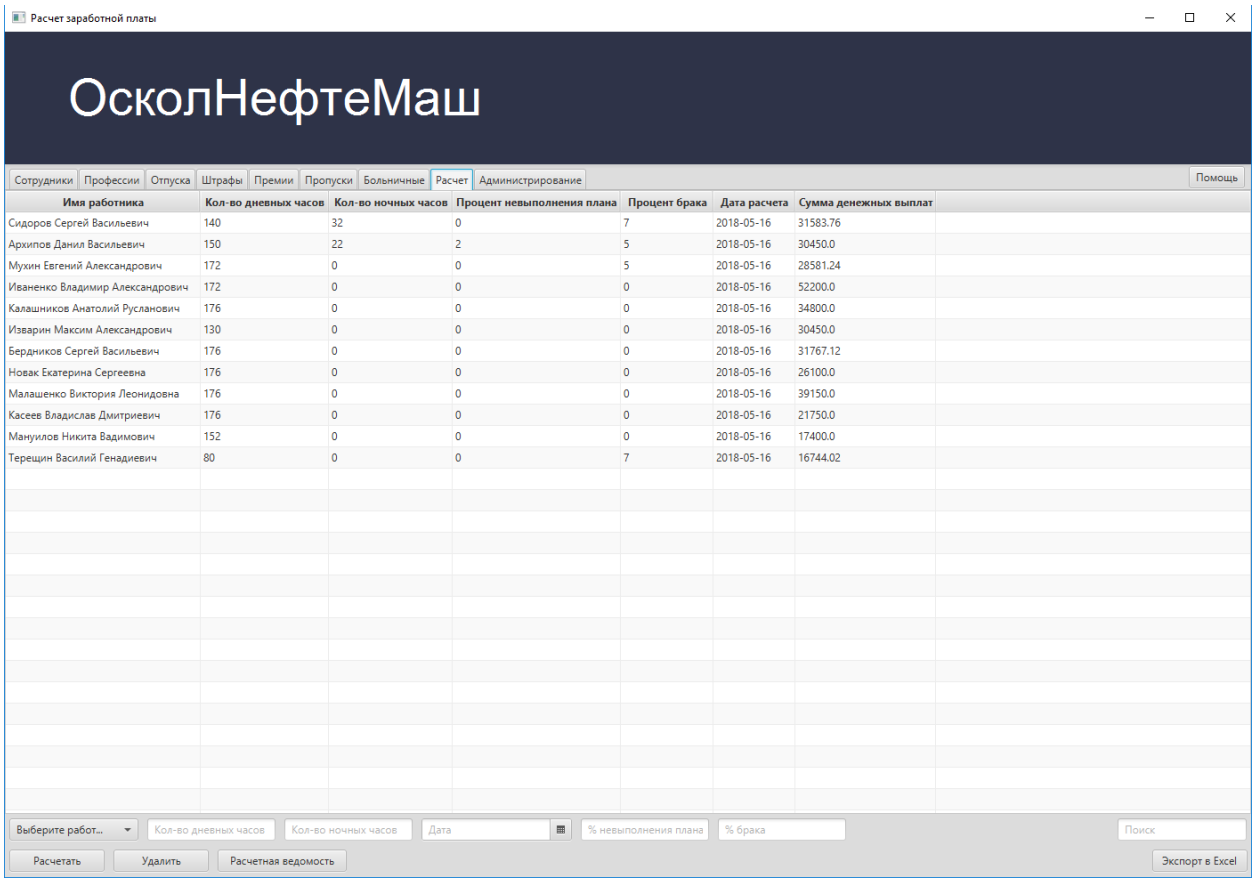
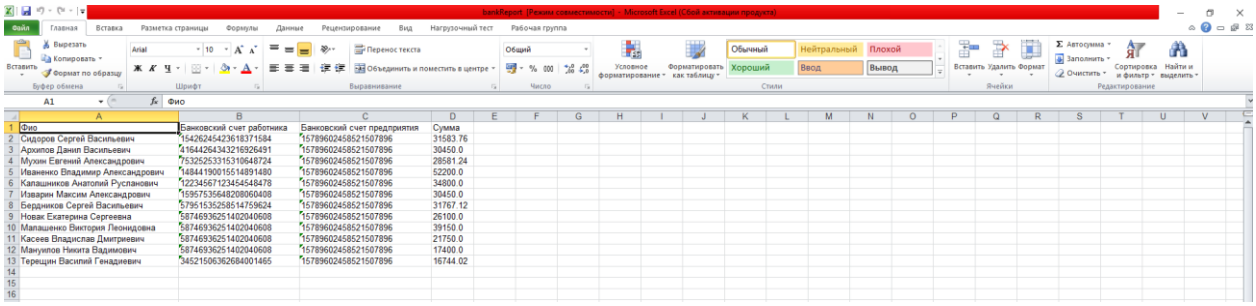


Рис. 3.18. Результаты расчетов программы.

Как мы видим, результаты расчета заработной платы для Терещина Василия Геннадиевича совпадают. Из чего можно сделать вывод, что программа считает правильно.

Во вкладке «Расчет» реализована возможность формирования расчетной ведомости в MS Excel. Она содержит ФИО сотрудника, его банковский счет, банковский счет предприятия и сумму для начисления. При формировании этого отчета так же существует возможность при помощи поиска формировать содержимое отчета. На рисунке 3.19 будет показан документ, который формируется при нажатии на кнопку «Расчетная ведомость».



Фио	Банковский счет работника	Банковский счет предприятия	Сумма
Сидоров Сергей Васильевич	15426245423618371584	15789602458521507896	31583.76
Архипов Данил Васильевич	41644264343216926491	15789602458521507896	30450.0
Мухин Евгений Александрович	75325253315310648724	15789602458521507896	28581.24
Иваненко Владимир Александрович	14844190015514891480	15789602458521507896	52200.0
Калашников Анатолий Русланович	12234567123454548478	15789602458521507896	34800.0
Изварин Максим Александрович	15957535648208060408	15789602458521507896	30450.0
Бердников Сергей Васильевич	57951535258514759624	15789602458521507896	31767.12
Новак Екатерина Сергеевна	58746936251402040608	15789602458521507896	26100.0
Малашенко Виктория Леонидовна	58746936251402040608	15789602458521507896	39150.0
Касеев Владислав Дмитриевич	58746936251402040608	15789602458521507896	21750.0
Мануилов Никита Вадимович	58746936251402040608	15789602458521507896	17400.0
Терещин Василий Геннадиевич	34521506362684001465	15789602458521507896	16744.02

Рис 3.19. Расчетная ведомость

Далее протестируем функционал вкладки «Администрирование». Эта вкладка создана для администратора информационной системы и основная ее функция назначение ролей и установка логина и пароля для пользователей. Роль PART соответствует должности бухгалтер, то есть пользователи с этой ролью видят только сотрудников своего подразделения. FULL_PART эта роль для главного бухгалтера и его заместителя, она позволяет просматривать сотрудников всех подразделений. Роль ADMIN предназначена для администратора информационной системы, и только он имеет доступ к этой вкладке. И NOT_USER это сотрудники, которые не имеют доступа к информационной системе. На рисунке 3.20 будет представлен внешний вид вкладки «Администрирование».

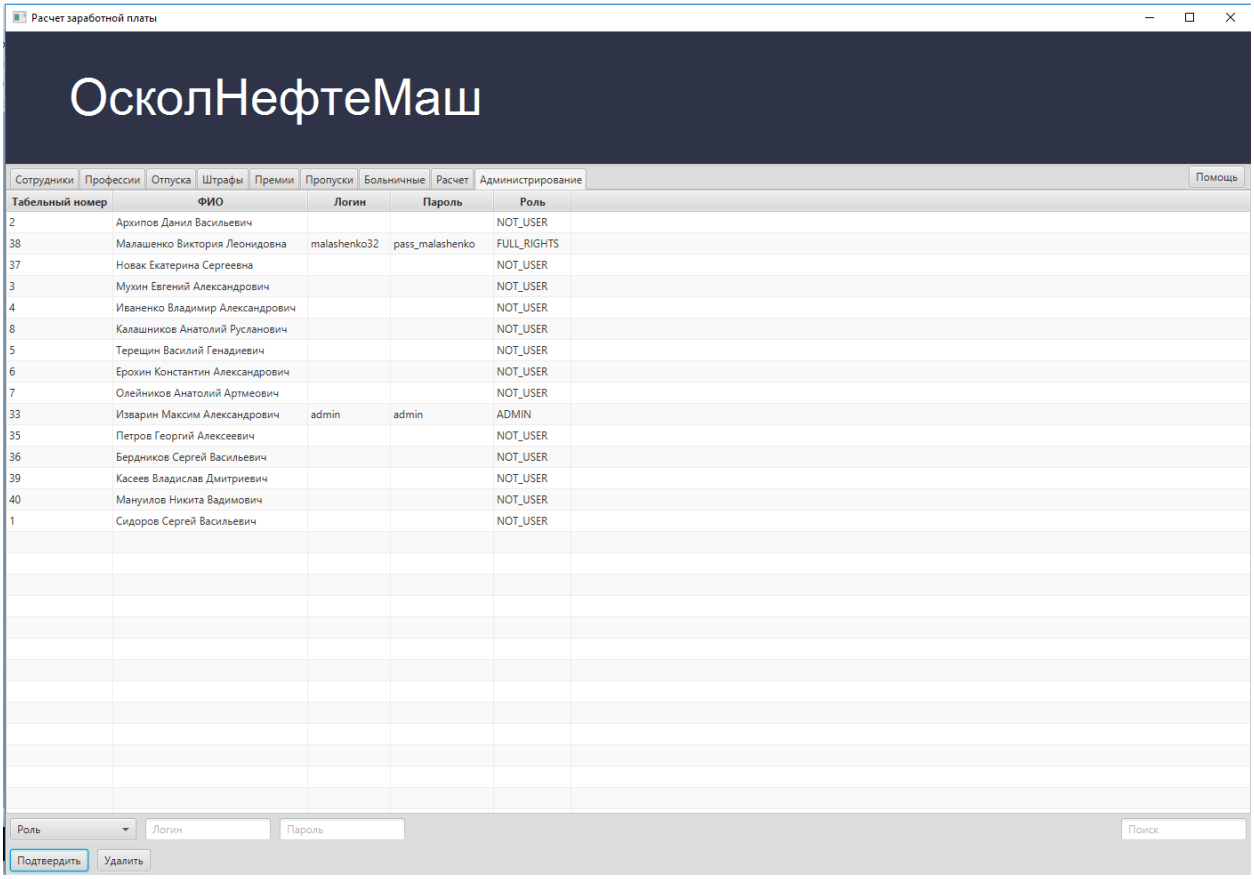


Рис. 3.20. Вкладка «Администрирование».

Протестируем на примере пользователя Новак Екатерина Сергеевна, которая занимает должность бухгалтер, которая соответствует роли PART, изменения роли и задания логина и пароля. На рисунке 3.21 будут показаны результаты изменения роли и задания логина и пароля. Как можно увидеть значения роль, пароль и логин для выбранного пользователя были изменены, и теперь он имеет доступ к системе, в соответствии со своей ролью.

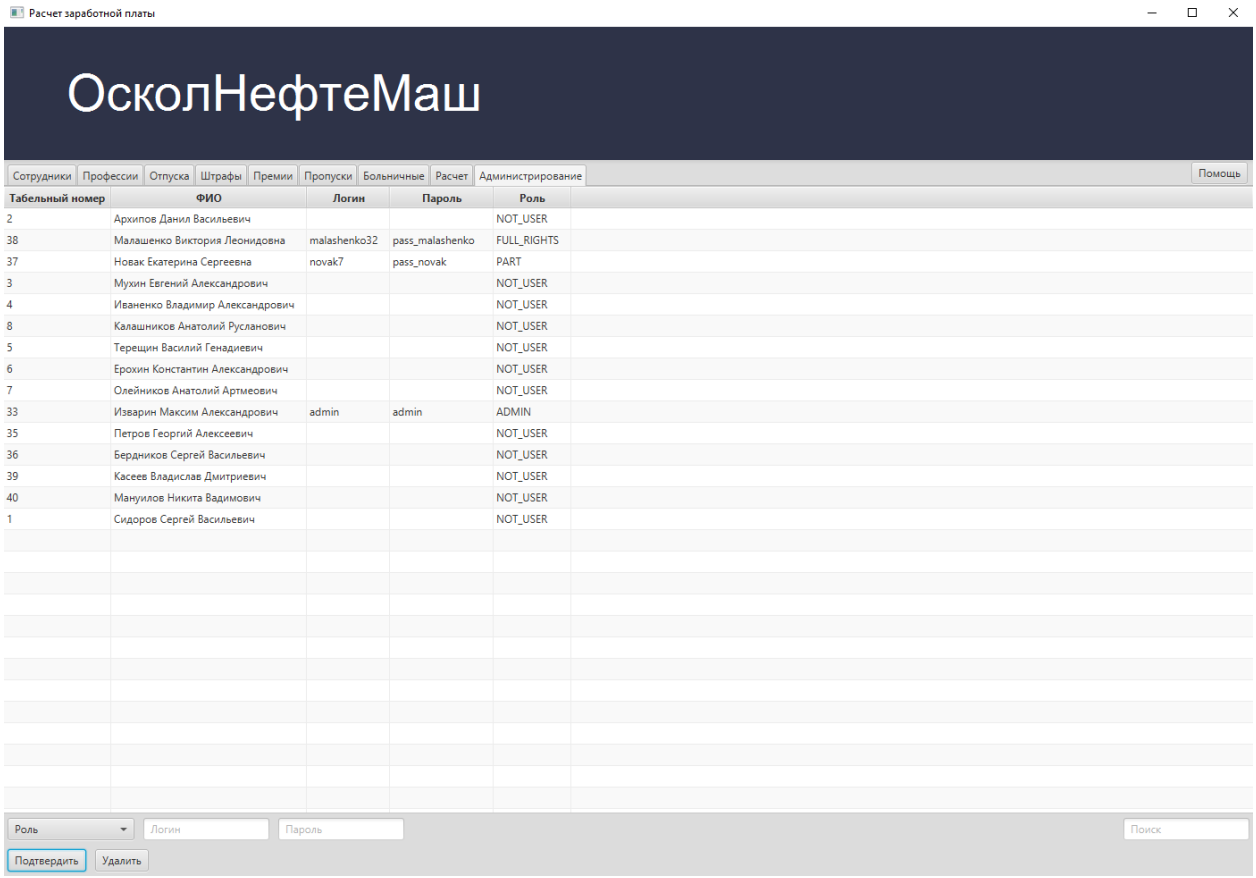


Рис. 3.21. Изменение роли, логина и пароля.

На этом тестирование информационной системы можно считать законченным. По результатам тестирования можно сделать вывод, что автоматизированная система соответствует всем заданным требованиям.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была спроектирована и разработана автоматизированная система расчета трудозатрат на предприятии ЗАО «ОсколНефтеМаш».

Данная автоматизированная система реализована в виде Desktop приложения, которое служит для работы с данными сотрудников предприятия, фиксирования отпусков, больничных, их дат начала и конца и денежных средств, которые необходимо выплатить сотруднику за этот промежуток времени. А так же запись всех штрафов и премий, их размера и даты. Основной функцией этой системы является расчет заработной платы на основе трудозатрат на производство запланированного количества продукции.

После анализа деятельности предприятия и изучения аналогов были сформулированы требования, на основе которых спроектирована и разработана автоматизированная система. По результатам тестирования системы, следует сделать вывод – разработанное приложение полностью удовлетворяет всем требования, предъявленным к системе на этапе постановки задачи. К выпускной квалификационной работе будет приложен акт апробации от предприятия, для которого разрабатывается автоматизированная система.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Преимущества и недостатки программы «1С: Бухгалтерия»// Стимул: учебный центр – 2013 [Электронный ресурс]. – URL: https://stimul.kiev.ua/articles.htm?a=preimuschestva_i_nedostatki_1s_buhgalteriya (Дата обращения: 07.05.2018).

2. Достоинства и недостатки программ компании «1С» (1С: Бухгалтерия, 1С: Предприятие)// UNIPRO: промышленные технологии – 2012 [Электронный ресурс]. – URL: https://unipro.com.ua/ru/dostoinstva-i-nedostatki-programm-kompanii--1s----1s--buhgalteriya--1s--predpriyatie/(Дата обращения: 07.05.2018).

3. Что такое «Парус»? Области применения, задачи и нюансы. Преимущества и недостатки// Стимул: учебный центр – 2014 [Электронный ресурс]. – URL: https://stimul.kiev.ua/articles.htm?a=chto_takoe_parus_oblast_primeneniya_zadachi_i_nyuansy_preimushchestva_i_nedostat(Дата обращения: 07.05.2018).

4. Java// Википедия: свободная энциклопедия – 2015 [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Java (Дата обращения: 14.05.2018).

5. Java// Прогопедия: энциклопедия языков программирования – 2014 [Электронный ресурс]. – URL: http://progopedia.ru/language/java/ (Дата обращения: 14.05.2018).

6. JavaFX// Википедия: свободная энциклопедия – 2016 [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/JavaFX (Дата обращения: 14.05.2018).

7. Шаблон проектирования «Модель Представление Контроллер» // Википедия: свободная энциклопедия - 2012 [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 17.05.2018)

8. Ильичев С. MVC: что это такое и какое отношение имеет к пользовательскому интерфейсу // Типичный программист – 2015 [Электронный ресурс]. – URL: https://tproger.ru/articles/mvc/ (дата обращения: 18.05.2018).

9. Data Access Object// Википедия: свободная энциклопедия – 2015 [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Data_Access_Object (Дата обращения: 18.05.2018).

10. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом// habr – 2016[Электронный ресурс] URL: https://habr.com/post/282764/ (Дата обращения: 18.05.2018).

11. Нормализация отношений. Шесть нормальных форм// habr – 2015[Электронный ресурс] URL: https://habr.com/post/254773/ (Дата обращения: 20.05.2018).

12. 5 лучших материалов по PostgreSQL// ProgLib: Библиотека программиста - 2016 [Электронный ресурс] URL: https://proglib.io/p/postgresql/ (Дата обращения: 14.03.2018).

13. Учебник по JavaFX// code.markery – 2016 [Электронный ресурс] URL: http://code.makery.ch/library/javafx-8-tutorial/ru/part1/ (Дата обращения: 20.02.2018).

Приложение 1

Код класса DBConnection, пакета Configuration:

```
package configuration;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBConnection {
    private static final String url = "jdbc:postgresql://127.0.0.1:5432/diplom";
    private static final String user = "postgres";
    private static final String password = "qwe";
    private Connection connection = null;
    private static DBConnection dbConnection;
    private DBConnection () {
        try {
            Class.forName("org.postgresql.Driver");
            connection = DriverManager.getConnection(url, user, password);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }
    public static DBConnection getDbConnection() {
        if (dbConnection == null) {
            dbConnection = new DBConnection();
        }
        return dbConnection;
    }
    public void close() {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

```
    public Connection getConnection() {
        return connection;
    }
}
```

Пример реализации класса пакета Model на примере таблицы Absence:

```
package dao;
import configuration.DBConnection;
import model.Absence;
import model.Worker;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class AbsenceDao implements  Dao {
    private Connection connection;
    private static Dao dao;
    private AbsenceDao () {
        connection = DBConnection.getDbConnection().getConnection();
    }
    public static Dao getDao() {
        if (dao == null) {
            dao = new AbsenceDao();
        }
        return dao;
    }
    @Override
    public List getAll() {
        Dao workerDao = WorkerDao.getDao();
        List<Absence> absences = new ArrayList<>();
        String sql = "SELECT * FROM absence";
        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            while (resultSet.next()) {
                Absence absence = new Absence();
```

```
            absence.setId_absence(resultSet.getLong(1));
            absence.setCause(resultSet.getString(2));
            absence.setDate_absence(resultSet.getDate(3));;
            absence.setWorker((Worker) workerDao.getById(resultSet.getLong(4)));
            absences.add(absence);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return absences;
}

@Override
public void insert(Object entity) {

    Absence absence = (Absence) entity;
    String sql = "INSERT INTO public.absence(\n" +
            "            cause, date_absence, id_worker)\n" +
            "    VALUES (?, ?, ?);\n";
    try {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setString(1, absence.getCause());
        statement.setDate(2,  new java.sql.Date(absence.getDate_absence().getTime()));
        statement.setLong(3, absence.getWorker().getId_worker());
        statement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }

}
@Override
public void update(Object entity) {
    Absence absence = (Absence) entity;
    String sql = "UPDATE public.absence\n" +
```

```
            "   SET cause=?, date_absence=?, id_worker=?\n" +
            " WHERE id_absence = ?;\n";
    try {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setString(1, absence.getCause());
        statement.setDate(2,  new java.sql.Date(absence.getDate_absence().getTime()));
        statement.setLong(3, absence.getWorker().getId_worker());
        statement.setLong(4, absence.getId_absence());
        statement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@Override
public void delete(long id) {
    String sql = "DELETE FROM absence WHERE id_absence = ?";
    try {
        PreparedStatement statement =  connection.prepareStatement(sql);
        statement.setLong(1, id);
        statement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
@Override
public Object getById(long id) {
    Dao workerDao = WorkerDao.getDao();
    Absence absence = new Absence();
    String sql = "SELECT * FROM absence WHERE id_absence=" + id;
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(sql);
        while (resultSet.next()) {
            absence.setId_absence(resultSet.getLong(1));
```

```
            absence.setCause(resultSet.getString(2));
            absence.setDate_absence(resultSet.getDate(3));;
            absence.setWorker((Worker) workerDao.getById(resultSet.getLong(4)));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return absence;
}
```

Пример реализации класса пакета DAO на примере таблицы Absence:

```
package dao;
import configuration.DBConnection;
import model.Absence;
import model.Worker;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class AbsenceDao implements  Dao {
    private Connection connection;
    private static Dao dao;

    private AbsenceDao () {
        connection = DBConnection.getDbConnection().getConnection();
    }

    public static Dao getDao() {
        if (dao == null) {
            dao = new AbsenceDao();
        }
        return dao;
    }
    @Override
```

```
public List getAll() {
    Dao workerDao = WorkerDao.getDao();
    List<Absence> absences = new ArrayList<>();
    String sql = "SELECT * FROM absence";
    try {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(sql);
        while (resultSet.next()) {
            Absence absence = new Absence();
            absence.setId_absence(resultSet.getLong(1));
            absence.setCause(resultSet.getString(2));
            absence.setDate_absence(resultSet.getDate(3));;
            absence.setWorker((Worker) workerDao.getById(resultSet.getLong(4)));
            absences.add(absence);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return absences;
}

@Override
public void insert(Object entity) {

    Absence absence = (Absence) entity;
    String sql = "INSERT INTO public.absence(\n" +
            "            cause, date_absence, id_worker)\n" +
            "    VALUES (?, ?, ?);\n";
    try {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setString(1, absence.getCause());
        statement.setDate(2,  new java.sql.Date(absence.getDate_absence().getTime()));
        statement.setLong(3, absence.getWorker().getId_worker());
        statement.execute();
```

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

}

@Override
public void update(Object entity) {
    Absence absence = (Absence) entity;
    String sql = "UPDATE public.absence\n" +
            "   SET cause=?, date_absence=?, id_worker=?\n" +
            " WHERE id_absence = ?;\n";
    try {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setString(1, absence.getCause());
        statement.setDate(2,  new java.sql.Date(absence.getDate_absence().getTime()));
        statement.setLong(3, absence.getWorker().getId_worker());
        statement.setLong(4, absence.getId_absence());
        statement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
@Override
public void delete(long id) {
    String sql = "DELETE FROM absence WHERE id_absence = ?";
    try {
        PreparedStatement statement =  connection.prepareStatement(sql);
        statement.setLong(1, id);
        statement.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
@Override
```

```
    public Object getById(long id) {
        Dao workerDao = WorkerDao.getDao();
        Absence absence = new Absence();
        String sql = "SELECT * FROM absence WHERE id_absence=" + id;
        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            while (resultSet.next()) {
                absence.setId_absence(resultSet.getLong(1));
                absence.setCause(resultSet.getString(2));
                absence.setDate_absence(resultSet.getDate(3));;
                absence.setWorker((Worker) workerDao.getById(resultSet.getLong(4)));
            }

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

Пример реализации класса пакета Controller на примере таблицы Absence:

```
package controller;

import dao.AbsenceDao;
import dao.Dao;
import dao.ProfessionDao;
import dao.WorkerDao;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import model.*;
```

```
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import java.io.FileOutputStream;
import java.net.URL;
import java.time.Instant;
import java.time.ZoneId;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;
import java.util.stream.Collectors;

public class AbsenceController implements Initializable {
    public TableView absenceTable;
    public ComboBox absenceList;
    public TextField cause;
    public DatePicker date_absence;
    public TextField searchAbsence;
    @SuppressWarnings("unchecked")

    public void initialize(URL location, ResourceBundle resources) {
        Dao absenceDao = AbsenceDao.getDao();
        List absence = absenceDao.getAll();
        TableColumn<Absence, Worker> worker_name_absence = new TableColumn<>("Имя работника");
        worker_name_absence.setCellValueFactory(new PropertyValueFactory<>("worker"));
        worker_name_absence.setCellFactory(col -> new TableCell<Absence, Worker>() {
            @Override
            protected void updateItem(Worker item, boolean empty) {
                super.updateItem(item, empty);
                if (empty) {
                    setText("");
                } else {
                    setText(item.getFio());
```

```
            }
        }
    });
    TableColumn<Absence, String> cause = new TableColumn<>("Причина пропуска");
    cause.setCellValueFactory(new PropertyValueFactory<>("cause"));
    TableColumn<Absence, Date> date_absence = new TableColumn<>("Дата пропуска");
    date_absence.setCellValueFactory(new PropertyValueFactory<>("date_absence"));
    absenceList.setItems(FXCollections.observableArrayList(WorkerDao.getDao().getAll()));
    absenceList.setCellFactory(c -> new ListCell<Worker>() {
        @Override
        protected void updateItem(Worker item, boolean empty) {
            super.updateItem(item, empty);
            if (empty) {
                setText("");
            } else {
                setText(item.getFio());
            }
        }
    });
    absenceTable.getColumns().addAll(worker_name_absence, cause, date_absence);
    absenceTable.setItems(FXCollections.observableArrayList(absence));
    searchAbsence.textProperty().addListener((observable, oldValue, newValue) -> {
        List absenceForSearch = absenceDao.getAll();

        Object[] filtered = absenceForSearch.stream().filter((o) ->
((Absence)o).getCause().toLowerCase().contains(newValue) ||
                ((Absence)o).getWorker().getFio().toLowerCase().contains(newValue)).toArray();
        absenceTable.setItems(FXCollections.observableArrayList(filtered));
    });
    absenceTable.setRowFactory(tv -> {
        TableRow<Absence> tableRow = new TableRow<>();
        tableRow.setOnMouseClicked(event -> {
            if (!tableRow.isEmpty()) {
                Absence absence1 = tableRow.getItem();
                absenceList.getSelectionModel().select(absence1.getWorker());
```

```
                this.cause.setText(absence1.getCause());

this.date_absence.setValue(Instant.ofEpochMilli(absence1.getDate_absence().getTime()).atZone
(ZoneId.systemDefault()).toLocalDate());
            }
        });
        return tableRow;
    });

}

public void refreshAbsenceTable () {

absenceTable.setItems(FXCollections.observableArrayList(AbsenceDao.getDao().getAll()));
}
public void add_absence() {
    Absence absence = new Absence();
    absence.setCause(cause.getText());

absence.setDate_absence(Date.from(Instant.from(date_absence.getValue().atStartOfDay(ZoneId
.systemDefault()))));
    absence.setWorker((Worker) absenceList.getSelectionModel().getSelectedItem());
    Dao dao = AbsenceDao.getDao();
    dao.insert(absence);
    refreshAbsenceTable();
}

public void del_absence() {
    Dao dao = AbsenceDao.getDao();
    dao.delete(((Absence)
absenceTable.getSelectionModel().getSelectedItem()).getId_absence());
    refreshAbsenceTable();
}

public void change_absence() {
```

```
        Absence absence = new Absence();
        absence.setId_absence(((Absence)
absenceTable.getSelectionModel().getSelectedItem()).getId_absence());
        absence.setCause(cause.getText());

absence.setDate_absence(Date.from(Instant.from(date_absence.getValue().atStartOfDay(ZoneId
.systemDefault()))));
        absence.setWorker((Worker) absenceList.getSelectionModel().getSelectedItem());
        Dao dao = AbsenceDao.getDao();
        dao.update(absence);
        refreshAbsenceTable();
    }

    public void export_absence_excel() {
        try{
            String filename="e:/absence.xls" ;
            HSSFWorkbook hwb = new HSSFWorkbook();
            HSSFSheet sheet =  hwb.createSheet("Премии");
            HSSFRow rowhead=   sheet.createRow(0);
            rowhead.createCell((short) 0).setCellValue("Работник");
            rowhead.createCell((short) 1).setCellValue("Причина");
            rowhead.createCell((short) 2).setCellValue("Дата пропуска");
            //sheet.setZoom(150);
            int i = 1;
            ObservableList absenceTableItem = absenceTable.getItems();
            for (Object item : absenceTableItem) {
                Absence absence = (Absence) item;
                HSSFRow row=   sheet.createRow(i);
                row.createCell(0).setCellValue(absence.getWorker().getFio());
                row.createCell(1).setCellValue(absence.getCause());
                row.createCell(2).setCellValue(new
java.sql.Date(absence.getDate_absence().getTime()));
                i++;
            }
            for(int j = 0; j <= 6; j++)
```

```
            sheet.autoSizeColumn(j);
        FileOutputStream fileOut =  new FileOutputStream(filename);
        hwb.write(fileOut);
        fileOut.close();
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Окно оповещения");
        alert.setContentText("Данные были экспортированы в Excel");
        alert.showAndWait();

    } catch ( Exception ex ) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Окно оповещения");
        alert.setContentText("Ошибка при экспорте данных");
        alert.showAndWait();

    }
  }
}
```

Метод отвечающий за расчет:

```
public void calculation_button() {
    double dayHours = Double.valueOf(amount_day_hours.getText());
    double nightHours = Double.valueOf(amount_night_hours.getText());

    int proisvPercent = Integer.valueOf(proc_proizv.getText());
    int brakPercent = Integer.valueOf(proc_braka.getText());
    Worker worker = (Worker) workersComboBox.getSelectionModel().getSelectedItem();
    int monthValue = date_calc.getValue().getMonthValue() - 1;
    int yearValue = date_calc.getValue().getYear();
    Double premiumSum = ((List<Premium>) PremiumDao.getDao().getAll()).stream()
            .filter(v ->
                    v.getWorker().getId_worker() == worker.getId_worker() &&
                            v.getDate_premium().getMonth() == monthValue &&
```

```
                    Integer.parseInt(new
SimpleDateFormat("yyyy").format(v.getDate_premium())) == yearValue)
            .map(Premium::getAmount_premium).mapToDouble(Double::doubleValue).sum();

        Double finesSum = ((List<Fines>) FinesDao.getDao().getAll()).stream()
            .filter(p ->
                p.getWorker().getId_worker() == worker.getId_worker() &&
                    p.getDate_fines().getMonth() == monthValue &&
                    Integer.parseInt(new   SimpleDateFormat("yyyy").format(p.getDate_fines()))
== yearValue)
            .map(Fines::getAmount_fines).mapToDouble(Double::doubleValue).sum();

        Double vacationSum = ((List<Vacation>) VacationDao.getDao().getAll()).stream()
            .filter(v ->
                v.getWorker().getId_worker() == worker.getId_worker() &&
                    v.getDate_end_vacation().getMonth() == monthValue &&
                    Integer.parseInt(new
SimpleDateFormat("yyyy").format(v.getDate_end_vacation())) == yearValue)
            .map(Vacation::getPay_for_vacation).mapToDouble(Double::doubleValue).sum();

        Double hospitalSum = ((List<Hospitals>) HospitalsDao.getDao().getAll()).stream()
            .filter(v ->
                v.getWorker().getId_worker() == worker.getId_worker() &&
                    v.getDate_end_hospitals().getMonth() == monthValue &&
                    Integer.parseInt(new
SimpleDateFormat("yyyy").format(v.getDate_end_hospitals())) == yearValue)
            .map(Hospitals::getPay_for_hospitals).mapToDouble(Double::doubleValue).sum();

        double pay_for_harm = worker.getProfession().getPay_for_harm();
        if (worker.getProfession().getForm_pay().equals("Сдельно-премиальная")) {

            double dayCost = worker.getTarrif_rate() * dayHours;
            double nightCost = worker.getTarrif_rate() * 2 * nightHours;

            double permiumPercent = 25 - 2.5 * proisvPercent;
```

```
            if (brakPercent <= 7) {
                permiumPercent += (7 - brakPercent) * 2;
            } else {
                permiumPercent += (brakPercent - 7) * 5;
            }
            double premium = (worker.getTarrif_rate() * (dayHours + nightHours) *
(permiumPercent/100));

            double tax = (dayCost + nightCost + premium + premiumSum - finesSum + vacationSum
+ hospitalSum + pay_for_harm)/100*13;

            cost = dayCost + nightCost + premium + premiumSum - finesSum + vacationSum +
hospitalSum + pay_for_harm - tax;
        } else{

            double salary = worker.getSalary();

            double tax = (salary + premiumSum + hospitalSum + vacationSum - finesSum +
pay_for_harm) / 100 * 13;

            cost = salary + premiumSum + hospitalSum + vacationSum - finesSum - tax +
pay_for_harm;
        }

        Calculation calculation = new Calculation();
        calculation.setNight_hours(Integer.valueOf(amount_night_hours.getText()));

calculation.setDate_calc(Date.from(Instant.from(date_calc.getValue().atStartOfDay(ZoneId.syst
emDefault()))));
        calculation.setWorker((Worker)
workersComboBox.getSelectionModel().getSelectedItem());
        calculation.setDay_hours(Integer.valueOf(amount_day_hours.getText()));
        calculation.setProizv(Integer.valueOf(proc_proizv.getText()));
        calculation.setBrak(Integer.valueOf(proc_braka.getText()));
        calculation.setAmount(cost);
```

```
    Dao dao = CalculationDao.getDao();
    dao.insert(calculation);
    refreshCalculationTable();

    amount_day_hours.setText("");
    amount_night_hours.setText("");
    proc_proizv.setText("");
    proc_braka.setText("");
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*