

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

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

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРИЛОЖЕНИЕ ДЛЯ РАСЧЕТА
ЗАРАБОТНОЙ ПЛАТЫ СОТРУДНИКОВ
КОММЕРЧЕСКОЙ ОРГАНИЗАЦИИ

Студент 31.05.22 Пронькин Андрей Александрович
(подпись, дата) (ФИО полностью)

Руководитель 31.05.22 А.В. Дубравин
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 31.05.22 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно (протокол заседания от 21 июня 2022 № 19)
(подпись, дата) (фамилия, инициалы)

Секретарь ГЭК Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2022

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» 04 2022г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Пронькин Андрей Александрович, гр.18ВВ2 факультета вычислительной техники.

2. Тема работы: «Приложение для расчета заработной платы сотрудников коммерческой организации»

Тема утверждена приказом ПГУ №357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Дубравин А.В.

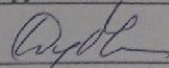
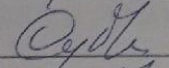
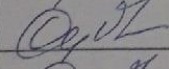
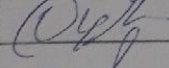
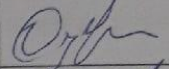
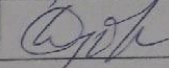
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Приложение предназначено для расчета заработной платы сотрудников коммерческой организации. Оно должно быть интуитивно понятно для бухгалтера. Входные данные вносятся бухгалтером, выходные данные можно сохранить в Excel. Приложение должно автоматизировать работу бухгалтера и исключать возможность ошибки, так как это может понести большие финансовые потери. Приложение должно осуществлять расчет заработной платы согласно нормам ТК РФ. Основные клиенты приложения – бухгалтеры.

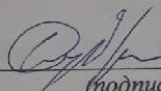
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить аналоги существующих приложений, разработать, изучить основы разработки приложений, выбрать наиболее комфортный дизайн приложения. Также выполнить задание по БЖД.

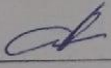
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	50	26.04.2022 – 30.04.2022	
2.	Постановка задачи	10	01.05.2022 – 02.05.2022	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2022 – 07.05.2022	
4.	Описание алгоритмов	40	08.05.2022 – 11.05.2022	
5.	Разработка программного приложения расчета заработной платы сотрудников коммерческой организации	150	12.05.2022 – 29.05.2022	
6.	Оформление пояснительной записки	24	30.05.2022 – 31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы  26.04.22 Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент  26.04.22 Пронькин А.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  17.06.2022 Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит 122 страницы формата А4, 26 рисунков, 23 таблицы, 8 источников, 3 приложения.

БД, БУХГАЛТЕРИЯ, SQL, СУБД, SQLSERVER, MYSQL, C#.

Объект разработки - Приложение для расчета заработной платы сотрудников коммерческой организации.

Целью дипломной работы является разработка приложения для расчета заработной платы сотрудников коммерческой организации, предназначенного для облегчения работы бухгалтера. Это достигается путем заполнения и сбора информации в хранилище данных, введенных сотрудниками предприятия. Необходимо решить задачу автоматизации учета заработной платы коммерческой организации, путем создания приложения.

В результате проведенной работы создано приложение, протестированы программные средства, удовлетворяющие поставленной задаче.

В пояснительной записке приведен анализ предметной области, обзор технологии разработки и отладки программы.

Графическая часть содержит модели БД и UML-диаграммы приложения.

					ПГУ 1.090301.31.001 ПЗ			
Изм.	Лист	№ докум	Подпись	Дата	Приложение для расчета заработной платы сотрудников коммерческой организации Пояснительная записка	Лит.	Лист	Листов
Разраб.		Пронькин А.А.						
Провер.		Дубравин А.В.					2	122
Н. Контр.		Бычков А.С.				ФВТ, каф. ВТ гр. 18ВВ2		
Утверд.		Митрохин М.А.						

Содержание

Введение.....	5
1 Приложение для расчета заработной платы сотрудников коммерческой организации	6
1.1 Анализ предметной области	6
1.1.1 Анализ существующих аналогов.....	8
1.1.2 Постановка задачи.....	13
1.1.2 Выбор среды разработки	16
1.1.3 Выбор СУБД.....	21
1.2 Реализация	20
1.2.1 Администрирование базы данных.	20
1.2.2 Входная и выходная информация.	24
1.2.3 Описание реализации.....	31
1.3 Эксплуатация разработанного приложения.....	34
2 Безопасность жизнедеятельности.....	45
2.1 Анализ опасных и вредных факторов, воздействующих на программиста при разработке данной системы.....	45
2.2 Микроклимат рабочей зоны программиста	46
2.3 Освещение рабочего места	47
2.4 Электробезопасность. Статическое электричество.....	49
2.5 Организация рабочего места программиста	52
2.6 Пожаробезопасность	52
2.7 Профилактика пожара	53
Заключение.	56
Список использованных источников.	57
Приложение А – Листинг программы.....	58
Приложение Б – <i>UML</i> -диаграммы.....	109

Введение

Последние десять лет существенно изменилось программное обеспечение. В настоящее время ПО способно не только выполнять простые математические операции, но и управлять сложными системами предприятия.

Бухгалтерская сфера – это та сфера, где каждый день проходят десятки операций. Бухгалтерам постоянно приходится вести учет начислений зарплат, налогов, расчет больничных листов. Прибавим к этом постоянные изменения в законодательстве. Именно поэтому бухгалтера, ведущие расчет заработной платы сотрудников, заинтересованы в создание автоматизированного приложения, которое значительно упрощает их деятельность.

Актуальность выбранной темы в том, что в настоящее время учет заработной платы не имеет смысла без автоматизации.

Объектом исследования стало клиент-серверное приложение по учету заработной платы, состоящее из клиента с графическим интерфейсом.

1 Приложение для расчета заработной платы сотрудников коммерческой организации

1.1 Анализ предметной области

Сегодня управление бизнесом без компьютера просто невообразимо. Компьютеры уже давно прочно вошли в управленческие области бухгалтерского учета, управления складом и классификации магазинов. Некоторые программы могут обрабатывать большие потоки информации.

Целью любого приложения является обработка данных об объектах и явлениях реального мира и предоставление необходимой информации о них.

Под базой данных (БД) понимают хранилище структурированных данных, при этом данные должны быть непротиворечивы, минимально избыточны и целостны. Базу данных можно определить как унифицированную совокупность данных. Сама БД должна быть хорошо спроектирована, чтобы система БД функционировала как единое целое [1].

Хорошо организованная база данных облегчает доступ к содержащейся в ней информации. Хорошо спроектированные базы данных сокращают время и усилия, необходимые для ввода данных, их изменения и извлечения из баз данных.

Автоматизация – это возложение определенных функций на компьютерную программу. И неважно, больше функций или меньше, даже использование компьютера для выполнения рутинной операции уже автоматизация.

Современный этап развития компьютеризации и автоматизации характеризуется использованием распределенной обработки информации. Наиболее перспективной областью применения концепции распределенной обработки информации является автоматизация функций планирования и управления на базе персональных компьютеров, установленных

непосредственно на рабочих местах специалистов.

Бухгалтерская сфера занятости не является исключением, где в крупной компании ежедневно могут проходить сотни хозяйственных операций, все они должны быть отражены в субъектах учета и для этого должны быть внесены необходимые документы [2].

Таким образом, расчеты заработной платы наемного персонала занимают особое место в наиболее трудоемкой области бухгалтерского учета.

Эта работа требует: высокой степени точности, аккуратности, предельного внимания и, конечно же, своевременного выполнения всех бухгалтерских операций. В реестре могут быть десятки, сотни или даже тысячи людей, и для каждого из них необходимо записать и обработать большой объем данных.

Бухгалтерам необходимо учитывать совокупный доход работников-налогоплательщиков и правильно определять все вычеты и отображать их все в бухгалтерском учете.

Также необходимо большое внимание уделять знанию российского законодательства, поэтому бухгалтеры должны постоянно следить за всеми изменениями, касающимися налоговых льгот, надбавок, выплат, начислений отпускных, расчетов больничных и т.д. Вот почему бухгалтеры, которые производят расчеты заработной платы сотрудников, заинтересованы в использовании автоматизированной системы, которая значительно упрощает работу на этом сайте [3].

Расчет заработной платы – сложный и длительный процесс. Ведь вы должны подсчитывать все выплаты и удержания в структуре заработной платы вашего сотрудника, платить налоговые отчисления и социальные отчисления, а также своевременно сдавать и готовить квартальные отчеты [2].

При ведении учета вручную возможны ошибки и неточности, которые могут привести к огромным последствиям и финансовому краху компании.

При автоматизации учета необходимо, чтобы приложение позволяло значительно сэкономить время и силы работы бухгалтера за счет автоматизации рутинных операций.

Чем больше штат предприятия и чем больше разнообразие начислений и отчислений, тем сложнее решить эти задачи без использования компьютерных технологий. Вот почему многие бухгалтеры, занимающиеся этой частью бухгалтерского процесса, хотят использовать автоматизированную систему, предназначенную для расчета заработной платы, вместо того, чтобы выполнять все расчеты вручную.

Какие еще преимущества дает автоматизация труда? Создание автоматизированного приложения для расчета заработной платы сотрудников в коммерческих компаниях позволит сделать работу предприятий более эффективной, значительно упростив ввод и вывод информации. Один человек может выполнять работу десяти человек. Это значительно увеличит прибыль компании [3].

1.1.1 Анализ существующих аналогов

Рынок информационных технологий представлен довольно большим количеством систем ведения бухгалтерского учета.

"Инфо-Бухгалтер" - это автоматизированная система расчета заработной платы любой сложности, кадровый учет в любом бизнесе, программа расчета заработной платы, программа учета персонала, программа автоматизации склада, это удобная программа автоматизации для владельцев бизнеса.

Основной функцией программы является бухгалтерский и налоговый учет, при этом все бухгалтерские документы, разработанные в строгом соответствии с требованиями ФНС, формируются автоматически.

Программа очень удобна в работе и не потребует много времени на обучение.

Главное отличие от других программ, в программе «Инфо-Бухгалтер» строго соблюдены все принципы теории и практики бухучета, а также учтены требования национальных и международных стандартов бухгалтерского учета и отчетности. Все это обеспечивает высокое качество и эффективность учетного процесса.

Недостатки программы:

- однозадачность;
- отсутствие в стандартной постановке *DOS* сетевых средств и многих других встроенных возможностей.

Цель системы бухгалтерского учета «Парус» - подготовка и отслеживание финансово-хозяйственной деятельности компании, сбор информации о хозяйственной деятельности, осуществляемой на счетах, и получение внутренней и внешней отчетности.

«Парус» обеспечивает: полную и достоверную информацию о финансово-хозяйственной деятельности; оформление первичных документов, на которых основывается бухгалтерский учет; хранение и движение денежных средств или основных средств; отражение операций на основании документов в бухгалтерском учете; внутреннее документирование фактов финансово-хозяйственной деятельности в различных отчетах; подготовку годовых, квартальных, месячных отчетов для бюджетного органа.

Недостатки программы:

- отсутствие возможности синтеза данных, поскольку данные готовятся в формате, определенном разработчиком, который не всегда отвечает потребностям лиц, принимающих решения;
- большое число данных и взаимосвязей между ними предусматривают

большой шанс ошибки, которое может стоить очень много.

Программа «1С: Бухгалтерия» является универсальной бухгалтерской программой и предназначена для ведения синтетического и аналитического бухгалтерского учета по различным разделам.

Преимущества: возможность создавать индивидуальные проекты, встроенный объектно-ориентированный язык программирования, полная открытость программных продуктов 1С.

В системе 1С можно выделить недостатки, рассмотренные ниже.

Универсального решения для всех типов компаний не существует. Основное применение 1С - автоматизация налогового и бухгалтерского учета. Основные продукты "1С" предназначены в первую очередь для автоматизации бухгалтерского и налогового учета для всех видов бизнеса. Эти решения не подходят для предприятий, служб и специализированных помещений. Многие компании, специализирующиеся на разработке 1С, разрабатывают собственные конфигурации для различных сфер бизнеса. Стоимость этих продуктов 1С значительно выше, чем базовой версии 1С, а функциональность необходимо улучшать и платить дополнительные деньги за настройку и поддержку. Связь продукта 1С с другим продуктом 1С (например, 1С:Бухгалтерия и 1С:Управление предприятием) осуществляется путем обмена информацией (выгрузка и/или загрузка документов).

Существенная разница между понятиями «готовая система» и «платформа». На каждом предприятии используется уникальное решение «1С», реализованное не настройкой конфигурации, а запрограммированное на языке программирования 1С, который встроен в платформу.

Настройка, внедрение и запуск должны осуществляться квалифицированным 1С-программистом. Это достаточно дорого обходится предприятию.

Необходимость передачи услуг по поддержке 1С на аутсорсинг. Очень

часто заказчик хочет, чтобы все проблемы и ошибки, возникающие в процессе работы с 1С (исправление ошибок конфигурации, обучение сотрудников работе с программой, установка обновлений программного обеспечения), решались немедленно.

Global — Salary включает несколько специальных интерфейсов для управления различными типами премий. Ввод данных для разовых премий для большого количества сотрудников управляется системой с помощью следующих функций:

- копирования строк с сотрудниками;
- копирования из предыдущих периодов;
- автоматического заполнения списком сотрудников;
- автоматического проставления нужных сумм и процентов.

Процентные премии, а также премии в виде фиксированных сумм могут быть введены бухгалтером индивидуально (например, с электронной карты работника), либо через список-журнал. В то же время можно использовать автоматизированные действия для облегчения массового ввода при премирования. Дополнительно пользователь может установить дату начала и окончания периода в форме, где будет начисляться бонус.

Для процентных премий система диктует настройки начислений. При этом каждый вид премии может иметь свой тип ввода. Кроме того, имеется поддержка ведения «истории» определенных типов записей в базе данных.

Автоматизация работы при присуждении премий большому количеству сотрудников, бухгалтеру важно максимально использовать помощь системы для облегчения и ускорения ввода информации в систему.

Обширная практика использования *Global-Salary* на крупных предприятиях позволила создать набор интерфейсов, максимально автоматизирующих действия пользователей по вводу бонусов за массовые премии.

Система предоставляет пользователю:

- копирование строк с сотрудниками;
- копирование премии с процентами из предыдущих периодов;
- автоматического заполнения списком сотрудников;
- автоматического проставления процентов премии;
- задание шаблона маски для автоматического проставления процентов по категории и подразделению;
- автоматически получать информацию по депримированным сотрудникам;
- возможность «снимать» премии предыдущих периодов, автоматически генерируя корректировочные записи;
- автоматически генерировать корректировки, в случаях изменения базы для премии в прошлых периодах;
- вести аудит действий пользователей в системе;
- лишение и снижение премии.

В *Global — Salary* проработан также и такой момент как депримирование — лишение премии для отдельных сотрудников или уменьшение процента премии. Для такого учёта в системе есть возможность по каждому сотруднику внести необходимую информацию или просто написать в примечании. При этом, данная информация будет учитываться при массовом премировании, возможно также выделение цветом таких записей.

Кроме того, возможен вариант, когда необходимые изменения вносятся в уже сформированный список сотрудников, откуда исключаются полностью депримированные работники или им корректируется процент премии.

Программы для ведения бухгалтерского учета с их достоинствами и недостатками продемонстрированы в таблице 1.1.

					ПГУ 1.090301.31.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 1.1- Аналогии бухгалтерских программ

Название системы	Достоинства	Недостатки
Инфо-бухгалтер	Простота. Высокое качество Упрощенный документооборот.	Однозадачность.
Парус-бухгалтерия	Простой и разнообразный функционал. Быстрое внедрение.	Большое числа данных и взаимосвязей, что увеличивает риск возникновения ошибки.
1:С Бухгалтерия	Возможность создавать и загружать индивидуальные настройки.	Отсутствие возможности интеграции с другими сервисами. Требуется обучение пользователя.
<i>Global Salary</i>	Возможность массового премирования Большой выбор функционала. Эффективная работа с документами. Мощная система отчетности.	Сложность в обучении персонала.

1.1.2 Постановка задачи

Учитывая достоинства и недостатки, а также особенности работы бухгалтера были поставлены следующие задачи:

- изучить Трудовой Кодекс РФ;
- рассмотреть аналогичные программы, провести сравнение, продумать наиболее удобный функционал и дизайн;
- ознакомиться с сетевыми приложениями «клиент-сервер»;
- разработать функционал сервера с клиентами;
- изучить основы разработки приложений.

При выполнении задач должны быть учтены следующие основные требования:

- оптимальное для пользователя представление исходной и конечной информации.
- простота и удобство в эксплуатации программы автоматизации;

Задача дипломного проекта – разработать систему для бухгалтерии, которая позволит:

- автоматизировать процедуру учета заработной платы;
- вводить и хранить данные о заработной плате сотрудников.

1.1.3 Выбор среды разработки

Для разработки приложений доступны различные инструменты программирования и среды. Каждая из этих сред программирования имеет свои преимущества и недостатки.

Java - это язык, о котором слышали почти все, кто пользуется компьютерами и Интернетом. Но мало кто знает, для чего нужна *Java*, где ее можно использовать и что с ней можно делать. Это особенно важно для людей, которые хотят всю жизнь заниматься кодом и не знают, какой язык учить.

Это один из самых популярных языков в истории программирования.

Он был выпущен в 1995 году и с тех пор не потерял своей актуальности. В различных публикациях и статистических данных он по-прежнему входит в пятерку самых популярных языков программирования.

"Java" используют многие популярные и известные компании, такие как *TripAdvisor*, *eBay*, *Intel*, *Google*, *Yandex*, *AliExpress*, *Netflix* и многие другие.

Язык получил свое название от одноименного кофе, а его символом является кофейная чашка.

Java - простой и легкий для изучения язык. Учитывая его сложность, он является умеренно трудным для изучения. Некоторые языки легче выучить: *Ruby*, *Python* и т.д., но есть и более сложные языки: *C#*, *C++* и т.д. Но в целом он подходит даже для новичков, которые никогда раньше не занимались программированием. Хорошее знание *Java* не означает, что вы останетесь без работы.

Как и все известные языки программирования, *Java* имеет преимущества и недостатки, которые влияют на сферу его применения.

Преимущества *Java*:

- код независимый: любая платформа, поддерживающая *Java VMs*, сможет скопировать код;
- надежный код: все платформы, поддерживающие *Java*, поддерживают *Java*, большое преимущество;
- высокая функциональность: на *Java* можно написать практически все
- от простых приложений для смартфонов до программ машинного обучения для беспилотников;
- умеренно сложный синтаксис: подходит для новичков, которые никогда раньше не разрабатывали;
- *Android* - самая популярная операционная система для смартфонов, а *Java* - самый популярный язык приложений для *Android*. Изучение *Java* - это очень широкая область.

Java также имеет недостатки:

- низкая производительность: *Java* в целом медленнее других языков, таких как *C*, *C#*, *C++* и *Python*, из-за особенностей языка *Java*;

- из-за своих особенностей он требует больше памяти, чем многие другие языки сторонних разработчиков;

- оплата: на практике *Java* является платным языком для коммерческих и юридических проектов, но совершенно бесплатным для личного использования.

Python - самый популярный язык, используемый во всех областях информатики: машинное обучение, разработка программных приложений, создание сценариев автоматизации и многое другое.

Python - это язык высокого уровня, который доминирует во многих областях, его можно смело назвать универсальным языком общего назначения. *Python* удалось завоевать рынок веб-разработки.

Благодаря своей универсальности *Python* отличается от других языков и привлекает к себе большое внимание, поклонники *Python* часто называют его языком номер один, а сегодняшние рейтинги и данные от *PYPL*, *TIOBE* и *statista.com* частично подтверждают это, поскольку *Python* входит в тройку лидеров уже несколько лет подряд.

Он был разработан в конце 1980-х годов Гвидо ван Россумом, в то время программистом в Голландском центре компьютерных наук и математики. Он была разработана в его свободное время и основана на языке *ABC*.

Многие эксперты считают, что *Python* облегчает написание кода и ускоряет разработку. Да, его проще и быстрее программировать, но он имеет следующие особенности:

- динамическая типизация: программисту не нужно указывать тип переменной - это делает сам язык, а операторы различных типов,

участвующие в операции, автоматически преобразуются в нужный тип, но по определенным правилам;

- удобно возвращать несколько значений из функции: программист может перечислить значения и разделить их запятыми, и они автоматически преобразуются в список. Например, чтобы вернуть массив из функции, просто введите `"return_array_name"`. Это означает, что функции не нужно выделять память или передавать указатели;

- автоматическое распределение памяти: программисту не нужно самому выделять память. Да, контроль над программой меньше, но есть и преимущество: разработка действительно ускоряется;

- уборщик мусора: когда объект становится ненужным, то есть его больше ничто не трогает, сборщик мусора автоматически удаляет его снова. Это позволяет сборщику мусора оптимально использовать память без необходимости вручную удалять ненужные объекты;

- $a, b = b, a$. Мы просто изменяем значения переменных с помощью этой простой буквенно-цифровой нотации. То, что было *a*, теперь стало *b*, и наоборот. Этот метод удобен и позволяет изменять значения трех, пяти и более переменных,

- тип данных: в *Python* тип данных связан со значением, а не с переменной. Значение - это объект с атрибутами, которые определяют тип и другие свойства этого объекта, а переменная - это просто ссылка на этот объект. Чего достигает такой подход? Типы не нужно определять явно, и это облегчает переназначение значения переменной (что особенно важно, если тип нового значения отличается от исходного).

Плюсы языка *Python*:

- открытый код;
- простота синтаксиса;
- бесплатность;

- простота;
- многоплатформенность.

Минусы:

- невысокая скорость выполнения программ;
- эволюция языка;
- ошибки исполнения.

Для разработки данной программы использовалась среда программирования *Microsoft Visual Studio C# 2019*, а также система управления базами данных *SQL Server 2008*.

Одно из основных отличий *C#* от других подобных средств визуальной разработки приложений — наличие очень быстрого и эффективного компилятора. Компилятор будет выдавать не только сообщения об ошибках, но и множество предупреждений и подсказок, позволяющих создавать более чистый код. Важным преимуществом также является то, что одновременно выдается несколько сообщений об ошибках, что позволяет быстрее их исправить [5].

Не менее важным с точки зрения профессионального программиста является то, что *C#* имеет хорошо развитые инструменты для работы с базами данных. Базы данных позволяют эффективно управлять информацией. *C#* позволяет эффективно создавать приложения, работающие с базами данных.

Преимущества данного языка перед другими:

- улучшенная отладка программ. Интегрированный отладчик имеет много полезных свойств;
- визуальное построение приложений позволяет быстро и качественно создать интерфейс программы;
- высокая производительность;
- актуальность языка в настоящее время [6].

1.4 Выбор СУБД

Oracle 12c - это система управления базами данных, разработанная для облачных вычислений и может быть развернута на одном или нескольких серверах. *Oracle 12c* может управлять базами данных с миллиардами записей. Безопасность *Oracle 12c* находится на самом высоком уровне, поскольку каждая транзакция изолирована от других. Все эти преимущества СУБД компенсируют высокую стоимость программного обеспечения.

MySQL - самая популярная система управления базами данных для веб-приложений. *MySQL* является бесплатным программным обеспечением, но для коммерческого использования доступна платная версия.

Эта СУБД позволяет выбирать различные механизмы хранения данных, что позволяет модифицировать функции инструмента и редактировать данные, хранящиеся в различных типах таблиц. Гибкость *MySQL* связана с количеством поддерживаемых типов таблиц: пользователи могут выбирать между таблицами *MyISAM*, которые поддерживают полнотекстовые запросы, и таблицами *InnoDB*, которые поддерживают транзакции на уровне отдельных записей.

Кроме того, СУБД *MySQL* включает пример конкретного типа таблицы, который показывает, как создать новый тип таблицы. Благодаря открытому исходному коду и лицензии *GPL*, в базы данных *MySQL* постоянно добавляются новые типы таблиц. Он также имеет простой в использовании интерфейс и пакетные команды, которые позволяют удобно обрабатывать большие объемы данных. Система очень надежна и не пытается захватить все доступные аппаратные ресурсы [7].

Система управления *SQL Server 2008* представляет собой последнюю разработку фирмы *Microsoft* в области баз данных и анализа данных для быстрого создания масштабируемых решений электронной коммерции, бизнес приложений и хранилищ данных или для любых других целей, где

вам может потребоваться надежная система управления базами данных (СУБД) [8].

1.2 Реализация

1.2.1 Администрирование базы данных

Для работы в приложение необходимо установить *Microsoft SQL Server 2008 R2*.

Данный пакет можно скачать с официального сайта *Microsoft* (рисунок 1.1).

Он бесплатный.

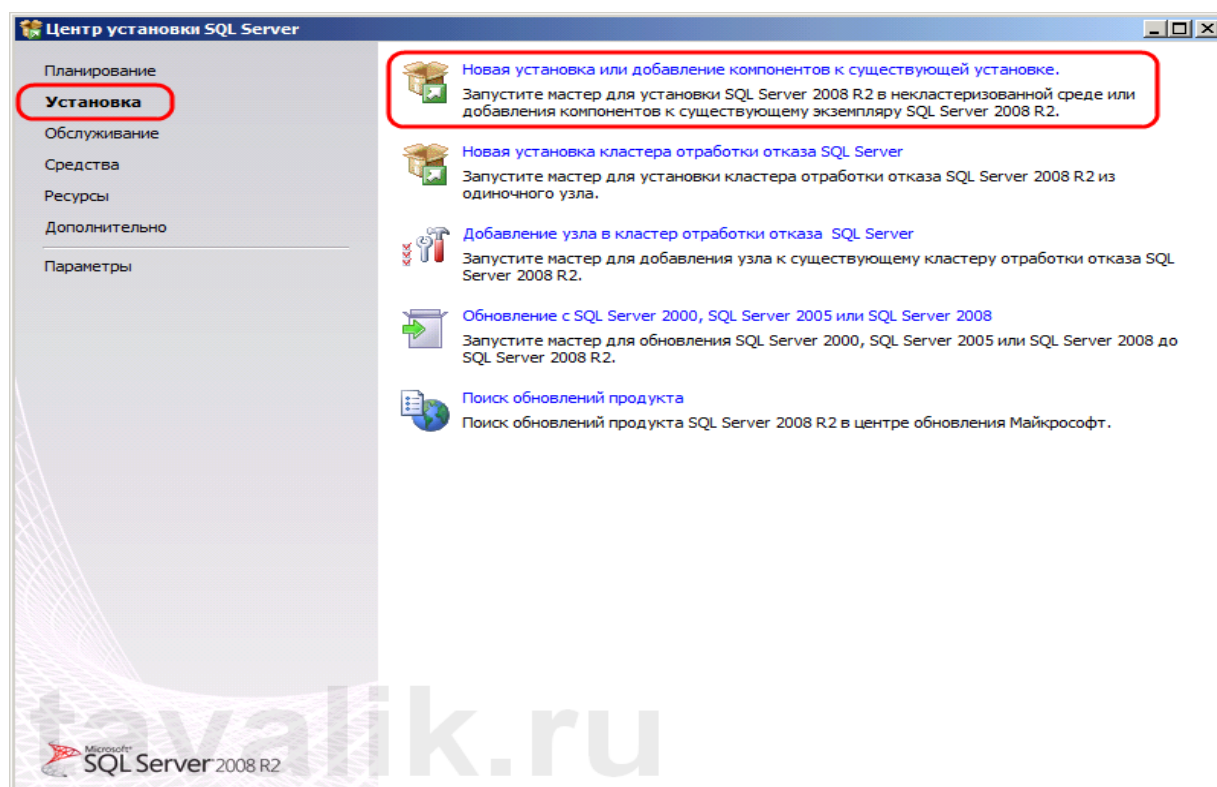


Рисунок 1.1 – Скачивание *Microsoft SQL Server*

Установив программу, копируем файлы базы данных (*ZP.mdf* и *ZP_log.LDF*) в папку *\MSSQL\DATA*).

Открываем «Среда *Microsoft SQL Server Management Studio*» (Пуск -> Все программы-> *Microsoft SQL Server*).

В окне «Соединение с сервером», показаном на рисунке 1.2, нажимаем кнопку «Соединить» и соединяемся с сервером.

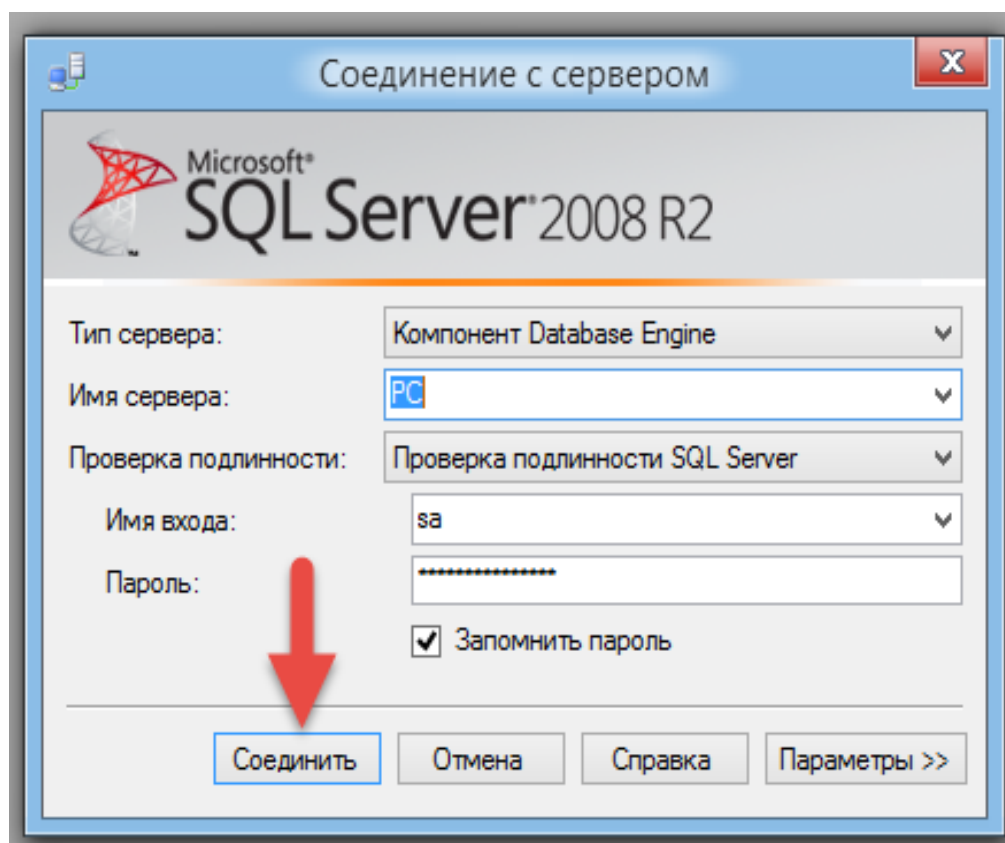


Рисунок 1.2 – Соединение с сервером

Подключившись к серверу, необходимо нажать на строку «Базы данных».

Далее нужно выбрать пункт «Присоединить», что проиллюстрировано на рисунке 1.3.

В открывшемся окне, продемонстрированном на рисунке 1.4, нажимаем на кнопку «Добавить».

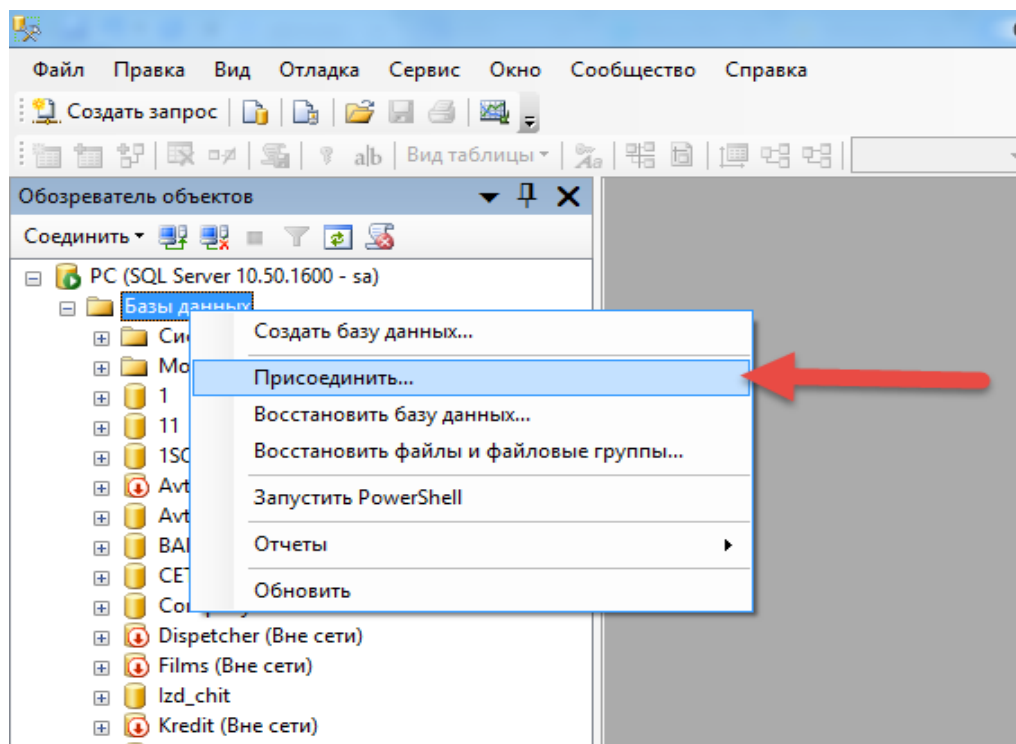


Рисунок 1.3 – Соединение с БД. Часть 1

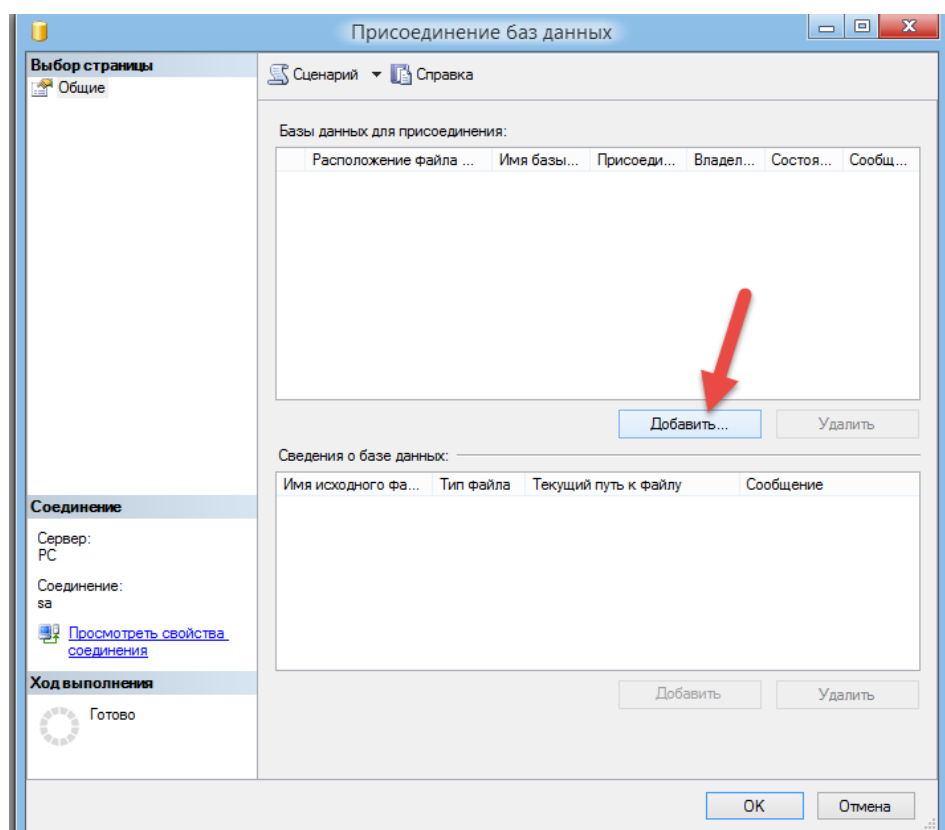


Рисунок 1.4 - Соединение с БД. Часть 2

Выбираем файл *ZP.mdf* (рисунок 1.5).

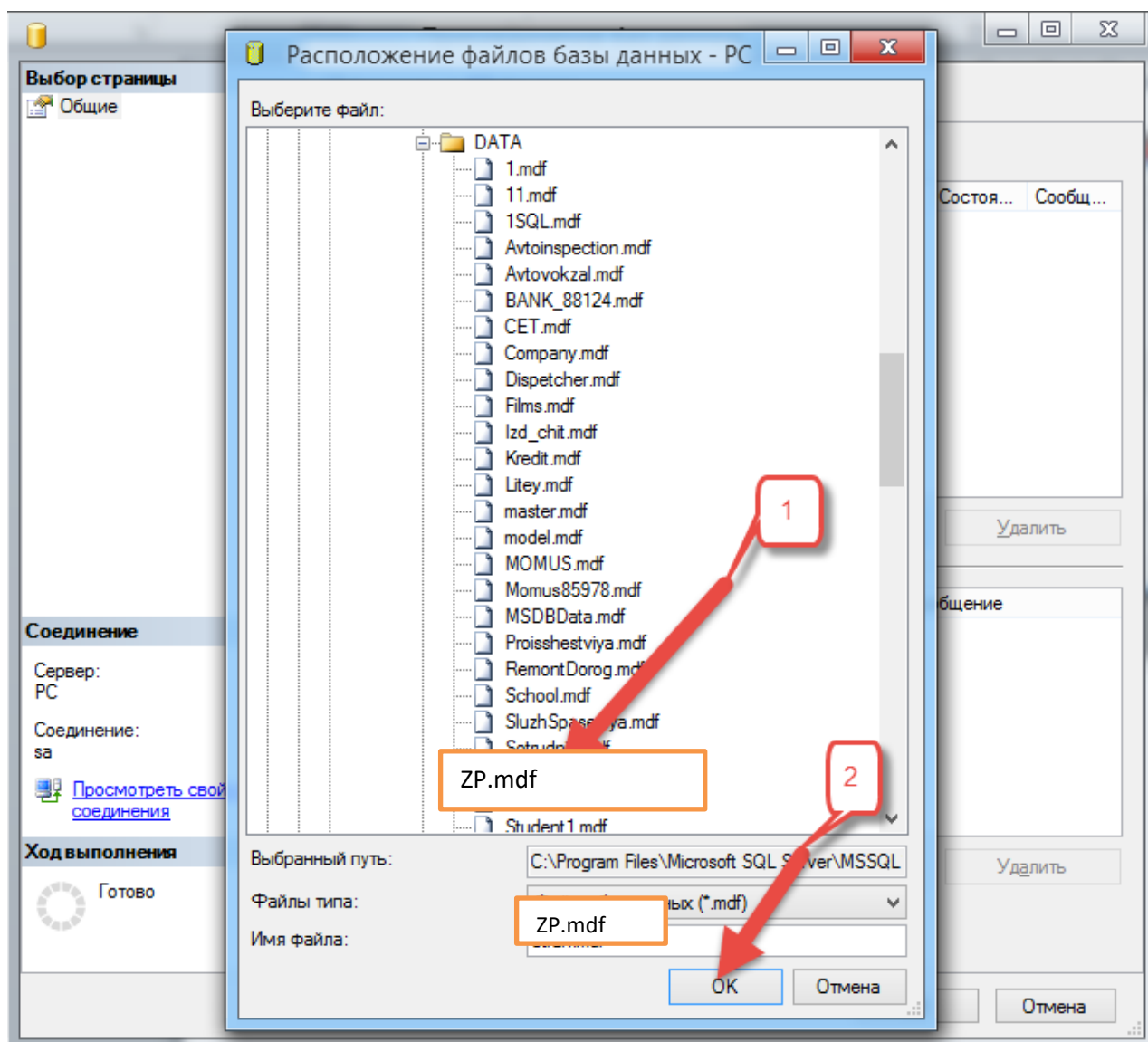


Рисунок 1.5 – Выбор файла

Далее необходимо нажать кнопку «*OK*» (рисунок 1.6).

Таким образом, на этом подключение базы данных окончено.

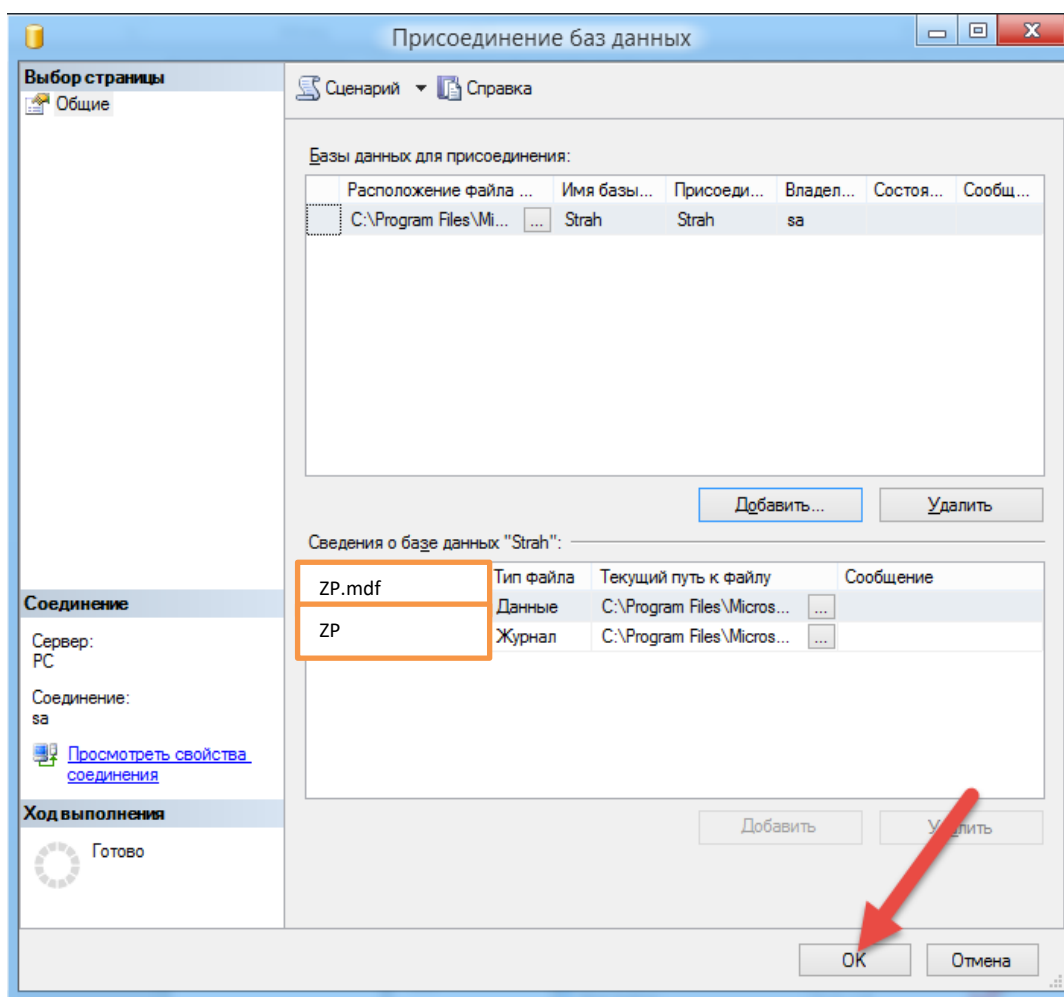


Рисунок 1.6 – Последний этап присоединения БД

1.2.2 Входная и выходная информация

На рисунке 1.7 представлена контекстная диаграмма процесса «Автоматизированный расчёт заработной платы».

Входная информация – это информация, поступающая в организацию. В разрабатываемым приложением входной информацией являются: данные о явке и неявке работника, данные о выполненной работе, штатное расписание и жалобы на работу работника.

Вся информация хранится в базе данных.



Рисунок 1.7– Схема процесса «Автоматизированный расчёт заработной платы»

Определим ключевые сущности приложения и их свойства (таблицы 1.2 – 1.11).

Таблица 1.2. – Сущность «КТУ»

Наименование поля	Тип данных	Описание
<i>ID</i> КТУ	Число	Идентификатор КТУ
Название	Строка	Название КТУ
Описание	Строка	Описание КТУ

Таблица 1.3. – Сущность «Должность»

Наименование поля	Тип данных	Описание
<i>ID</i> должности	Число	Идентификатор должности
Название	Строка	Название должности
Оклад	Число	Оклад должности

Таблица 1.4 – Сущность «Отделы»

Наименование поля	Тип данных	Описание
<i>ID</i> отдела	Число	Идентификатор отдела
Название	Строка	Название отдела

Таблица 1.5 – Сущность «Параметры КТУ»

Наименование поля	Тип данных	Описание
<i>ID</i> параметра	Число	Идентификатор параметра
Min	Число	Минимальный порог для получения процента
Max	Число	Максимальный порог для получения процента
Процент премии	Число	Процент премии

Таблица 1.6 – Сущность «Причины неявки»

Наименование поля	Тип данных	Описание
<i>ID</i> причины	Число	Идентификатор причины
Название	Строка	Название причины
Обозначение	Строка	Обозначение причины

Таблица 1.7 – Сущность «Сотрудник»

Наименование поля	Тип данных	Описание
<i>ID</i> сотрудника	Число	Идентификатор сотрудника
ФИО	Строка	ФИО сотрудника
Телефон	Строка	Телефон сотрудника
Уволен	Логический	Флаг увольнения

Таблица 1.8 – Сущность «Неявки»

Наименование поля	Тип данных	Описание
ID неявки	Число	Идентификатор неявки
Дата	Дата	Дата неявки

Таблица 1.9 – Сущность «Данные о работе»

Наименование поля	Тип данных	Описание
ID данных	Число	Идентификатор данных
Значение	Число	Значение КТУ для сотрудника
Год	Строка	Год данных КТУ для сотрудника
Месяц	Строка	Месяц данных КТУ для сотрудника

Таблица 1.10 – Сущность «Претензия»

Наименование поля	Тип данных	Описание
ID претензии	Число	Идентификатор претензии
Дата	Дата	Дата претензии
Описание	Строка	Описание претензии

Таблица 1.11 – Сущность «Значение КТУ»

Наименование поля	Тип данных	Описание
ID значения	Число	Идентификатор значения
Месяц	Число	Месяц значения
Год	Число	Год значения
Общий процент	Число	Общий процент значения

Результатной информацией, разрабатываемой АИС являются: табель рабочего времени, отчёт по КТУ, представление о премировании, аналитика.

Данная информация расположена в таблице 1.12

Таблица 1.12 – Сущность «Табель»

Наименование поля	Тип данных
Код табеля	<i>Integer</i>
Код сотрудника	<i>Integer</i>
Дата	<i>Date</i>
Отметка	<i>Char</i>
Примечание	<i>Char</i>

Документ «Табель рабочего времени» (рисунок 1.8) позволяет просмотреть отметки, которые получили сотрудники на протяжении указанного месяца.

Месяц _____ Год _____

ФИО	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Всего

Рисунок 1.8 – Табель рабочего времени

Документ «Отчёт по КТУ» позволяет просмотреть эффективность работы каждого сотрудника по каждому показателю.

Данный документ продемонстрирован на рисунке 1.9.

ФИО	Должность	КТУ N		КОНЕЧНЫЙ ПОКАЗАТЕЛЬ КТУ
		Значение	% премии	

Рисунок 1.9 – Отчёт по КТУ

Документ «Представление о премировании», показанный на рисунке 1.10, позволяет просмотреть суммарную информацию о выработке каждого сотрудника.

ФИО	Должность	Оклад в рублях	Конечный показатель КТУ за месяц	Кол-во рабочих дней в месяце	Количество неявок	Сумма премии

Рисунок 1.10 – Представление о премировании

1.2.3 Описание реализации.

В рамках данной программы все данные хранятся в базе данных *Microsoft SQL Server*.

Для обращения к базе используется класс *SqlConnection*, представляющий уникальный сеанс для *SQL Server* источника данных. После чего формируется текст запроса к базе данных. Далее сформированный текст передается объект *SqlCommand*, представляющий набор команд данных и подключение к базе данных. Данные, полученные из таблицы, хранятся в объекте *DataSet* с помощью использования специальных адаптеров – *SqlDataAdapter*. Такой способ работы с базами данных дает возможность работать с данными даже в случае отсутствия подключения [8].

В таблице неограниченное количество записей, но по мере поступления информации данные вносятся в таблицу.

Каждая таблица содержит идентификатор и ключевое поле, которые позволяют выполнять поиск.

Схема базы данных расположена на рисунке 1.11.

Ниже, в таблицах 1.13-1.23 проиллюстрированы сущности представленной таблицы.

Продолжительность хранения информации в базе данных соответствует требованиям отрасли экономики, а также требованиям к хранению информации.

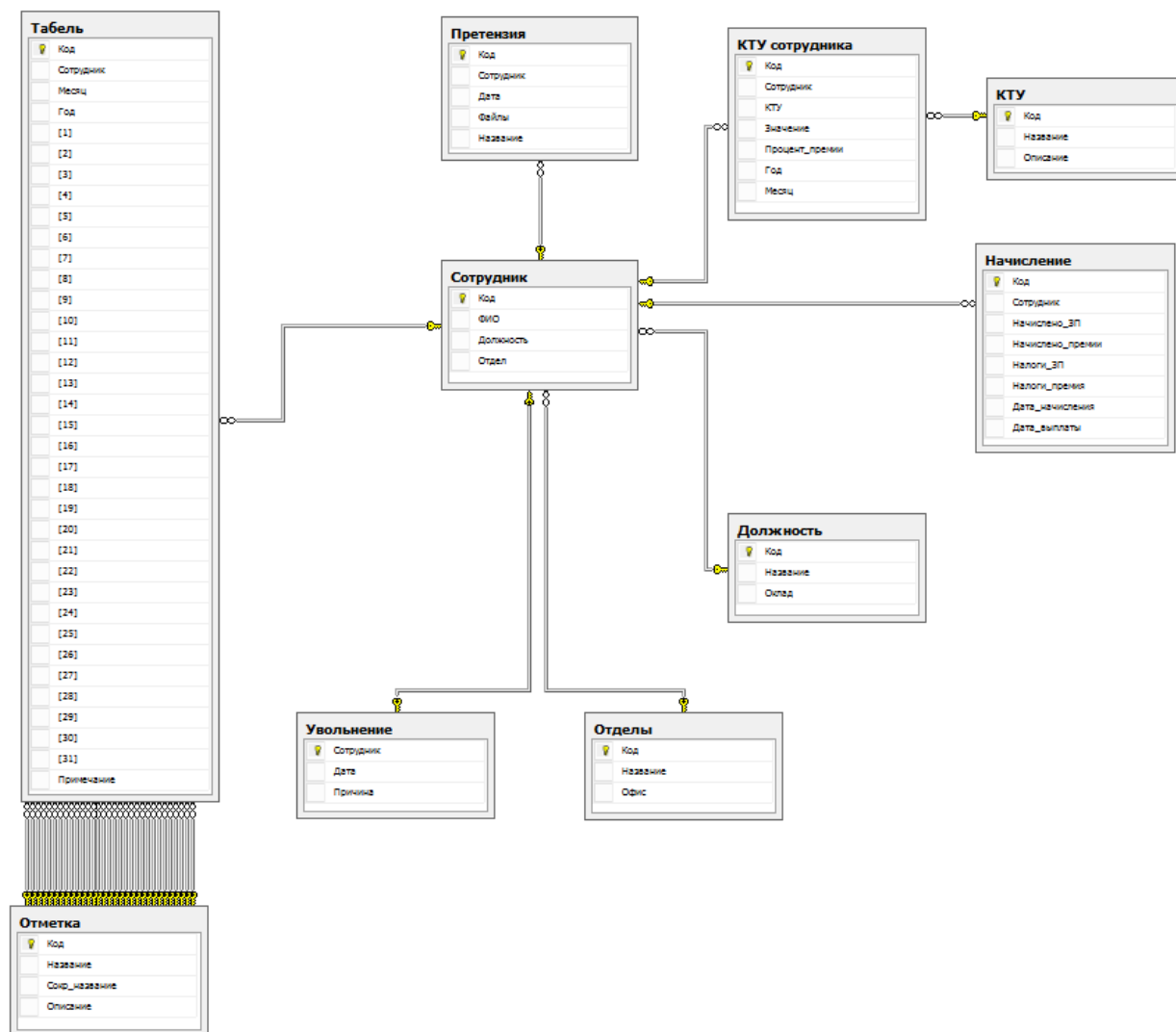


Рисунок 1.11– Схема базы данных

Таблица 1.13 – Структура таблицы «Отметки»

Имя поля	Тип данных	Длина	Ключевое поле
Код отметки	<i>Int</i>		Первичный ключ
Название	<i>nvarchar</i>	255	
Сокращённое название	<i>nvarchar</i>	255	
Описание	<i>nvarchar</i>	255	

Таблица 1.14 – Структура таблицы «Отделы»

Имя поля	Тип данных	Длина	Ключевое поле
Код отдела	<i>Int</i>		Первичный ключ
Название	<i>nvarchar</i>	255	
Офис	<i>nvarchar</i>	255	

Таблица 1.15 – Структура таблицы «Должности»

Имя поля	Тип данных	Длина	Ключевое поле
Код должности	<i>Int</i>		Первичный ключ
Название	<i>nvarchar</i>	255	
Оклад	<i>money</i>		

Таблица 1.16 – Структура таблицы «Увольнение»

Имя поля	Тип данных	Длина	Ключевое поле
Сотрудник	<i>Int</i>		Первичный ключ, Внешний ключ
Дата	<i>Datetime</i>		
Причина	<i>nvarchar</i>	255	

Таблица 1.17 – Структура таблицы «Претензия»

Имя поля	Тип данных	Длина	Ключевое поле
Код	<i>Int</i>		Первичный ключ
Сотрудник	<i>Int</i>		Внешний ключ
Дата	<i>Datetime</i>		
Файл	<i>Varbinary</i>	MAX	
Название	<i>Nvarchar</i>	255	

Таблица 1.18 – Структура таблицы «КТУ»

Имя поля	Тип данных	Длина	Ключевое поле
Код КТУ	<i>Int</i>		Первичный ключ
Название	<i>nvarchar</i>	255	
Описание	<i>nvarchar</i>	255	

Таблица 1.19 – Структура таблицы «КТУ сотрудника»

Имя поля	Тип данных	Длина	Ключевое поле
Код	<i>Int</i>		Первичный ключ
Сотрудник	<i>Int</i>		Внешний ключ
КТУ	<i>Int</i>		Внешний ключ
Значение	<i>Int</i>		
Процент премии	<i>Int</i>		
Год	<i>Int</i>		
Месяц	<i>nvarchar</i>	50	

Таблица 1.20 – Структура таблицы «Налоги»

Имя поля	Тип данных	Длина	Ключевое поле
Код налога	<i>Int</i>		Первичный ключ
Название	<i>nvarchar</i>	50	
Процент	<i>float</i>		
Применить на оклад	<i>bit</i>		
Применить на премию	<i>bit</i>		

Таблица 1.21 – Структура таблицы «Начисления»

Имя поля	Тип данных	Длина	Ключевое поле
Код	Int		Первичный ключ
Сотрудник	int		Внешний ключ
Начислено_ЗП	money		
Начислено_премии	money		
Налоги_ЗП	money		
Налоги_премия	money		
Дата_начисления	date		

Таблица 1.22 – Структура таблицы «Табель»

Имя поля	Тип данных	Длина	Ключевое поле
1	2	3	4
Код	Int		Первичный ключ
Сотрудник	int		Внешний ключ
Месяц	nvarchar	50	
Год	int		

Таблица 1.23 – Структура таблицы «Табель»

1	2	3	4
1	int		Внешний ключ
2	int		Внешний ключ
3	int		Внешний ключ
4	int		Внешний ключ
5	int		Внешний ключ
6	int		Внешний ключ
7	int		Внешний ключ

Продолжение таблицы 1.23

1	2	3	4
8	int		Внешний ключ
9	int		Внешний ключ
18	int		Внешний ключ
19	int		Внешний ключ
20	int		Внешний ключ
21	int		Внешний ключ
22	int		Внешний ключ
23	int		Внешний ключ
24	int		Внешний ключ
25	int		Внешний ключ
26	int		Внешний ключ
27	int		Внешний ключ
28	int		Внешний ключ
29	int		Внешний ключ
30	int		Внешний ключ
31	int		Внешний ключ

1.3 Эксплуатация разработанного приложения

Полный листинг приложения расположен в приложении А.

Главное окно программы позволяет пользователю осуществлять все необходимые операции с приложением.

Оно продемонстрировано на рисунке 1.12.

Рисунок 1.12 – Главное окно программы

В начале работы с приложением пользователь должен заполнить справочники (рисунок 1.13).

Справочник «Должности» (рисунок 1.14) позволяет пользователю указать данные о должностях, которые занимают сотрудники.

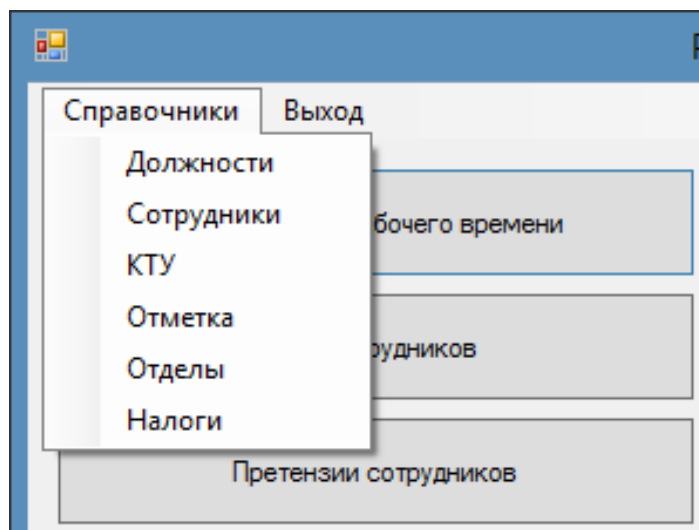


Рисунок 1.13 – Меню «Справочники»

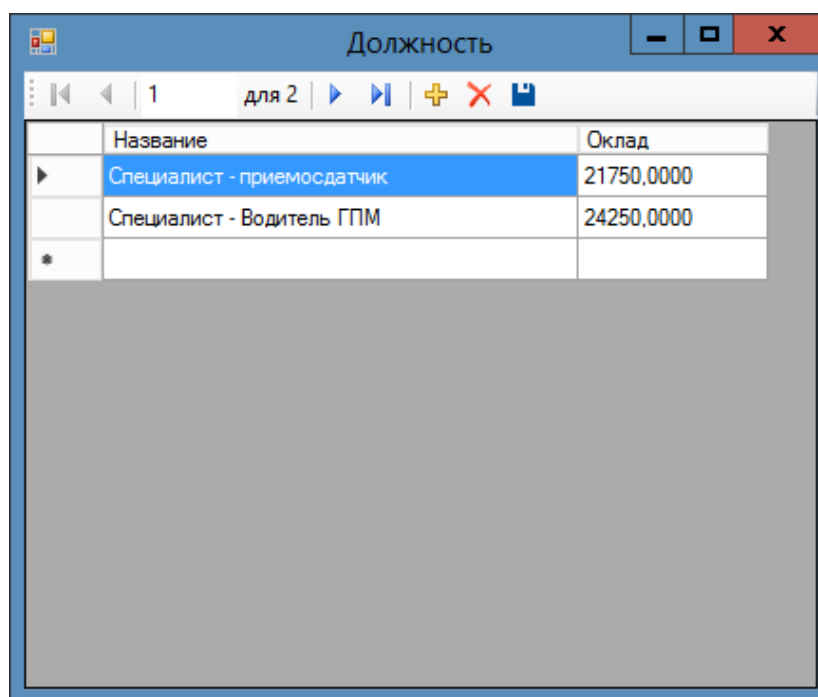


Рисунок 1.14 – Форма «Должность»

Справочник «Отделы» (рисунок 1.15) позволяет пользователю указать данные об отделах, которые есть на предприятие.

	Название	Офис
▶	Главный	306
	Филиал 1	54
*		

Рисунок 1.15 – Форма «Отделы»

Справочник «Сотрудники» (рисунок 1.16) позволяет пользователю указать данные о сотрудниках.

При необходимости можно произвести поиск нужного сотрудника по ФИО.

Справочник «КТУ» (рисунок 1.17) позволяет пользователю указать данные о КТУ, которые применяются к сотрудникам.

Сотрудники

1 для 5

ФИО	Должность	Отдел
Зайченков М.И.	Специалист - приемосдатчик	Главный
Федотов М.Г.	Специалист - приемосдатчик	Главный
Иванов А.К.	Специалист - приемосдатчик	Главный
Кручинин Р.А.	Специалист - приемосдатчик	Главный
Пигарин А.В.	Специалист - Водитель ГПМ	Главный
*		

Уволить

Поиск

Рисунок 1.16 – Форма «Сотрудники»

Справочник КТУ

1 для 5

Название
КТУ1 (ошибки при сборе заказа)
КТУ2 (ошибки при проверке заказа)
КТУ3 (скорость набора и проверки)
КТУ4 (отношение к работе)
КТУ5 (Кол-во объяснительных)
*

Описание

Рисунок 1.17 – Форма «Редактирование показателя КТУ»

Форма «Табель учёта рабочего времени» (рисунок 1.18) позволяет пользователю просмотреть данные о посещениях работы, а также об отметках неявки. При необходимости пользователь может распечатать табель, нажав на кнопку «Экспорт в Excel» (рисунок 1.19).

Рисунок 1.18 – Форма «Табель»

Месяц - Февраль Год - 2020

ФИО	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Всего
Зайченков М.И.	0																															0
Федотов М.Г.																																0
Иванов А.К.		11	бл																													11
Кручинин Р.А.			11																													22
Пигарин А.В.																																22

Рисунок 1.19 – Отчёт «Табель»

Форма «Данные о работе сотрудника за месяц» позволяет зарегистрировать данные согласно показателям КТУ для каждого сотрудника в указанный месяц.

Данная форма расположена на рисунок 1.20.

КТУ сотрудников

Назад

Укажите месяц: Март

Укажите год: 2020

	Сотрудник	КТУ	Значение	Процент премии
▶	Зайченков М.И.	КТУ1 (ошибки при сборе заказа)	7	20
	Зайченков М.И.	КТУ2 (ошибки при проверке за...	5	15
	Зайченков М.И.	КТУ3 (скорость набора и прове...	4	20
	Зайченков М.И.	КТУ4 (отношение к работе)	3	20
	Зайченков М.И.	КТУ5 (Кол-во объяснительных)	2	20
	Федотов М.Г	КТУ1 (ошибки при сборе заказа)	4	20
	Федотов М.Г	КТУ2 (ошибки при проверке за...	5	20
	Федотов М.Г	КТУ3 (скорость набора и прове...	7	20
	Федотов М.Г	КТУ4 (отношение к работе)	5	20
	Федотов М.Г	КТУ5 (Кол-во объяснительных)	0	20
*				

1 для 10

Отчет по КТУ

Представление о премировании

Рисунок 1.20 – Форма «Данные о работе сотрудника за месяц»

Добавить данные о претензиях к сотрудникам можно с помощью одноименной формы.

Форма, отвечающая за добавление претензии от сотрудников продемонстрирована на рисунке 1.21.

Претензии сотрудников

Список сотрудников

ФИО	Должность	Отдел
Зайченков М.И.	1	1
Федотов М.Г.	1	1
Иванов А.К.	1	1
Кручинин Р.А.	1	1
Пигарин А.В.	2	1

Дата	Название
29.02.2020	xarakt.txt

Открыть

Добавить Удалить

Рисунок 1.21 – Форма «Претензии»

На основании введенных данных можно сформировать отчеты (рисунки 1.22 – 1.24).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1			КТУ1 (ошибки при сб		КТУ2 (ошибки при пр		КТУ3 (скорость набор		КТУ4 (отношение к ра		КТУ5 (Кол-во объясн		КОНЕЧНЫЙ
2	ФИО	Должность	Значение	% премии	Значение	% премии	Значение	% премии	Значение	% премии	Значение	% премии	ПОКАЗАТЕЛЬ КТУ
3	Зайченков М.И.	Специалист - приемосдатчи	7	20	5	15	4	20	3	20	2	20	95
4	Федотов М.Г.	Специалист - приемосдатчи	4	20	5	20	7	20	5	20	0	20	100

Рисунок 1.22 – Отчёт «Отчёт по КТУ»

ФИО	Должность	Оклад в рублях	Конечный показатель КТУ за месяц	Кол-во рабочих дней в	Количество неявок	Сумма премии
Зайченков М.И.	Специалист - приемосдатчик	21750	95	16	1	19371,09
Федотов М.Г	Специалист - приемосдатчик	21750	100	16	1	20390,62

Рисунок 1.23 – Отчёт «Представление о премировании»

Код	ФИО	Полная смена	Неявка	больничный
1	Зайченков М.И.	5	1	1
2	Федотов М.Г	2	2	
3	Иванов А.К.	1	3	1
4	Кручинин Р.А.	1		1
5	Пигарин А.В.	1	1	

Рисунок 1.24– Отчёт «Выработка за месяц»

Таким образом, разработанное приложение выполняет все заявленные функции: позволяет вводить данные о сотрудниках, вести таблицы учета рабочего времени, указывать показатели КТУ по каждому сотруднику, формировать отчеты.

В ходе опытной эксплуатации приложения установлено, что количество задач, выполняемых в разработанном программном обеспечении, соответствует технологии начисления премий работникам на основе КТУ, и может быть принято решение о принятии разработанного программного продукта в эксплуатацию.

Также для каждого сотрудника можно произвести расчёт заработной платы с учётом налогом.

Это продемонстрировано на рисунке 1.25.

Сотрудник	Начислено ЗП	Начислено премии	Налоги ЗП	Налоги премия	Дата начисления	Дата выплаты
Зайченков М.И.	21750,0000	19371,0900	7612,5000	5850,0700	31.03.2020	31.03.2020
Федотов М.Г.	21750,0000	20390,6200	7612,5000	6157,9700	18.03.2020	18.03.2020
Иванов А.К.	21750,0000	0,0000	7612,5000	0,0000	30.03.2020	16.04.2020
Кручинин Р.А.	21750,0000	0,0000	7612,5000	0,0000	30.03.2020	

Добавить
Редактировать
Удалить

Рисунок 1.25 – Форма «Расчёт ЗП»

Расчёт заработной платы производится автоматически, с учётом премии, оклада и начисленных налогов (рисунок 1.26).

Сотрудник	Дата начисления	Дата выплаты
Зайченков М.И.	☒ 31 марта 2020 г.	☒ 31 марта 2020 г.

Перерасчет

Начислено ЗП	21750,00	Налоги ЗП	7612,5	К оплате 27658,52
Начислено премии	19371,09	Налоги премии	5850,07	
Всего начислено	41121,09	Всего налогов	13462,57	

Название	Процент	Сумма налога ЗП	Сумма налога премии	Сумма налога
Социальное страхование	30,2	6568,5	5850,07	12418,57
Пенсионный фонд	4,8	1044	0	1044

Рисунок 1.26 – Форма «Расчёт ЗП»

Поэтому разработанный программный инструмент выполняет все предписанные функции: позволяет вносить данные о сотрудниках, вести таблицы учета рабочего времени, указывать показатели КТУ по каждому сотруднику, а также формировать отчеты и рассчитывать заработную плату с удержанием налогов.

В ходе опытной эксплуатации программного продукта установлено, что объем задач, реализованных в разработанном программном обеспечении, соответствует технологии оплаты труда, и может быть принято решение о принятии разработанного программного продукта в эксплуатацию.

					ПГУ 1.090301.31.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

2 Безопасность жизнедеятельности

Безопасность человека должна решаться на всех этапах жизненного цикла, будь то разработка программы, ее реализация или эксплуатация.

Обеспечение безопасной жизни во многом зависит от правильной оценки опасных и вредных факторов, связанных с работой. Даже серьезные травмы человеческого тела могут быть вызваны рядом факторов. Они могут включать все аспекты рабочей среды, чрезмерные физические и умственные нагрузки, нервно-эмоциональный стресс и различные комбинации этих факторов.

2.1 Анализ опасных и вредных факторов, воздействующих на программиста при разработке данной системы

Опасные и вредные факторы производства по характеру своего возникновения подразделяют на следующие группы:

- физические;
- химические;
- психофизиологические;
- биологические.

В помещении на программистов могут негативно воздействовать следующие физические факторы:

- максимальная и минимальная температура;
- чрезмерное загрязнение воздуха пылью и газами;
- высокая или низкая влажность воздуха;
- недостаточное освещение на рабочем месте;
- шум выше допустимых пределов;
- чрезмерное ионизирующее излучение;
- повышенный уровень электромагнитного излучения;

- повышенный заряд статического электричества;
- опасность поражения электрическим током;
- выцветание экрана дисплея.

Химическая опасность, которая постоянно воздействует на программистов, - это наличие активных частиц, вызванных ионизацией воздуха во время работы за компьютером.

Биологические вредные производственные факторы в данном помещении отсутствуют.

Психологические факторы, которые влияют на операторов во время смены, следующие.

- стресс и эмоциональные перегрузки;
- психологический стресс.

2.2 Микроклимат рабочей зоны программиста

Микроклимат рабочей зоны - это экологический климат этих зон, определяемый сочетанием температуры, влажности и скорости движения воздуха, который воздействует на организм человека.

Место работы программиста является помещением I категории (выполняются легкие физические работы), поэтому должны соблюдаться следующие требования:

- оптимальная температура воздуха- 22°C (допустимая - $20-24^{\circ}\text{C}$);
- оптимальная относительная влажность- 40 -60% (допустимая - не более 75%) ;
- скорость движения воздуха не более 0.1м/с.

На рабочем месте используется нагрев горячей воды в холодное время года и охлаждение в теплое время года для автоматического создания оптимальной температуры, влажности, чистоты и воздушного потока независимо от внешних условий. Кондиционер - это система вентиляции,

которая поддерживает заданные параметры воздуха в помещении с помощью автоматического управления.

2.3 Освещение рабочего места

Работа, выполняемая с использованием вычислительной техники, имеют следующие недостатки:

- вероятность появления прямой блескости;
- ухудшенная контрастность между изображением и фоном;
- отражение экрана.

В связи с тем, что естественное освещение слабое, на рабочем месте должно применяться также искусственное освещение. Далее будет произведен расчет искусственного освещения.

Размещение светильников определяется следующими размерами:

- $H = 3$ м. - высота помещения;
- $h_c = 0,25$ м. - расстояние светильников от перекрытия;
- $h_n = H - h_c = 3 - 0,25 = 2,75$ м. - высота светильников над полом;
- $h_p = 0,7$ м - высота расчетной поверхности (для помещений, связанных с работой ПЭВМ);
- $h = h_n - h_p = 2,75 - 0,7 = 2,05$ - расчетная высота.

Светильника типа ЛДР (2х40 Вт). Длина 1,24 м, ширина 0,27 м, высота 0,10 м.

L - расстояние между соседними светильниками (рядами люминесцентных светильников), L_a (по длине помещения) = 1,76 м, L_b (по ширине помещения) = 3 м.

l - расстояние от крайних светильников или рядов светильников до стены, $l = 0,3 - 0,5L$.

$$l_a = 0,5 L_a, l_b = 0,3 L_b,$$

$$l_a = 0,88 \text{ м}, l_b = 0,73 \text{ м}.$$

Светильники с люминесцентными лампами в помещениях для работы рекомендуют устанавливать рядами.

Метод коэффициента использования светового потока предназначен для расчета общего равномерного освещения горизонтальных поверхностей при отсутствии крупных затемняющих предметов. Потребный поток ламп в каждом светильнике.

$$\Phi = E * r * S * z / N * h,$$

где Φ - поток ламп одного ряда;

E - заданная минимальная освещенность = 300 лк., т.к. разряд зрительных работ=3;

r - коэффициент запаса = 1,3 (для помещений, связанных с работой ПЭВМ);

S - освещаемая площадь = 30 м²;

z - характеризует неравномерное освещение, $z = E_{\text{ср}} / E_{\text{min}}$ - зависит от отношения $l = L/h$, $l_a = L_a/h = 0,6$, $l_b = L_b/h = 1,5$ т.к. l превышают допустимых значений, то $z = 1,1$ (для люминесцентных ламп);

N - число светильников, намечаемое до расчета. Первоначально намечается число рядов n , которое подставляется вместо N ;

h - коэффициент использования. Для его нахождения выбирают индекс помещения i и предположительно оцениваются коэффициенты отражения поверхностей помещения $r_{\text{пот.}}$ (потолка) = 70%, $r_{\text{ст.}}$ (стены) = 50%, $r_{\text{п.}}$ (пола) = 30%.

$$N = \Phi / \Phi_1,$$

где Φ_1 - поток ламп в каждом светильнике.

$$\Phi = 300 * 1,3 * 25 * 1,1 / 2 * 0,3 = 21450 \text{ лм.}$$

2.4 Электробезопасность. Статическое электричество

В зависимости от риска поражения электрическим током помещение может быть отнесено к классу 1, т.е. неопасное помещение (сухое, непыльное, с нормальной температурой воздуха, изолированными полами и низким заземлением).

На рабочем месте программиста металлическим является только корпус системного блока компьютера, но здесь используется IBM-совместимый системный блок, который, помимо рабочей изоляции, содержит элемент заземления и кабель с заземляющим проводником, подключенный к источнику питания. Поэтому оборудование в данном случае относится к категории 1 (ПУЭ).

Опасное и вредное воздействие электрического тока, электрической дуги и электромагнитных полей на человека проявляется в виде электротравм и профессиональных заболеваний.

От этого зависит степень опасного и вредного воздействия электрических токов, электрических дуг и электромагнитных полей на человека:

- рода и величины напряжения и тока;
- частоты электрического тока;
- пути тока через тело человека;
- продолжительности воздействия на организм человека.

Электробезопасность оборудования должна обеспечиваться техническими методами и средствами защиты, а также организационными и техническими мерами.

Главные причины для поражения электрического шока человека:

- контакт с токопроводящими металлическими частями (шкафы, периферийные устройства компьютера), которые могут находиться под напряжением из-за поврежденной изоляции;

- нерегулируемое использование электроприборов;
- отсутствие инструктажа сотрудников по правилам электробезопасности.

Во время использования в корпусе компьютера накапливается статическое электричество. На расстоянии 5-10 см от экрана напряженность электростатического поля составляет 60-280 кВ/м, что в 10 раз превышает обычные 20 кВ/м.

Кроме того, если компьютер не защищен от сбоев, может произойти отключение питания, что может привести к поражению электрическим током или электротравме. Чтобы избежать этого, я рекомендую убедиться, что металлический корпус устройства соединен с проводом заземления.

Электробезопасность обеспечивается по ГОСТ 12.1.030.- 81. Опасное и вредное воздействие электрических токов на людей в виде электротравм и профессиональных заболеваний.

Электробезопасность оборудования должна обеспечиваться техническими методами и средствами защиты, а также организационными и техническими мерами.

Рассмотрим основные причины поражения программиста электрическим током на рабочем месте:

- не прикасайтесь к металлическим частям системного блока компьютера, так как они не являются токопроводящими и могут стать активными при нарушении изоляции;
- использование электрических приборов, таких как электрические плиты, чайники, обогреватели.

2.5 Организация рабочего места программиста

Работа программистов на производстве требует от них длительного сидения, что приводит к постоянной гиподинамии и мешает им быть активными. Плечевые ремни важны при сидячей работе. Движение рук в пространстве влияет не только на лопатки и мышцы спины, но и на положение позвоночника, таза и даже ног.

Чтобы не заболеть, вы должны иметь возможность свободно менять позу. Для поддержания базовой рабочей позы необходимо соблюдать схему работы/отдыха в тех частях опорно-двигательного аппарата, где нет "отвлекающего" мышечного напряжения.

Антропологические особенности человека определяют габаритно-планировочные параметры его рабочего места, а также свободные параметры отдельных его элементов.

По условиям труда под рабочим местом программиста понимается рабочее место человека, на котором он сидит и работает.

Рабочее пространство программиста должно быть не менее 6 м². Высота помещения должна быть не менее 4 м, а его объем - не менее 20 м³ на человека. Анализ рабочего места программиста в лаборатории показал, что рабочее место имеет площадь поверхности 4 м² и объем 12 м³ и не соответствует вышеуказанным требованиям. Кроме того, анализ выявил аномалии в организации самого рабочего места программиста. В связи с этим специалисты предлагают организовать рабочее место программиста следующим образом. Высота над рабочей поверхностью, на которой работает оператор, должна составлять 720 мм. При необходимости высоту рабочей поверхности оператора можно регулировать в диапазоне от 680 до 780 мм. Оптимальный размер рабочей поверхности составляет 1600х1000 квадратных метров. Под столом должно быть предусмотрено пространство для ног глубиной 650 мм. Стол оператора также должен иметь подставку для ног,

расположенную под углом 15° к поверхности стола. Подставка для ног должна иметь длину 400 мм и ширину 350 мм. Клавиатура должна находиться на расстоянии не более 300 мм от края стола, чтобы обеспечить удобную опору для предплечья оператора. Расстояние между глазами оператора и видеоэкраном должно составлять 40-80 см.

Стол программиста должен оснащен подъемно-поворотным механизмом. Высота сиденья должна регулироваться в диапазоне 400-500 мм. Глубина сиденья должна быть не менее 380 мм, а ширина - не менее 400 мм. Поверхность спинки должна быть не менее 300 мм в высоту и 380 мм в ширину. Угол наклона спинки к плоскости сиденья должен составлять $90-110^\circ$.

2.6 Пожаробезопасность

Пожарные характеристики здания зависят от его назначения, класса пожарной опасности, этажности и площади пожарной зоны.

Конструкция зданий, в котором расположено рабочее место программиста, классифицируется как К1 (низкая пожарная опасность), поскольку в нем находятся горючие материалы (книги, документы, мебель, офисное оборудование и т.д.) и трудногорючие материалы (сейфы, разное оборудование и т.д.), которые могут гореть без взрыва при воздействии огня.

С конструктивной точки зрения здания можно разделить на несущие и облицовочные, выполненные из природного или искусственного камня, бетона или железобетона и дерева, защищенные раствором или твердым покрытием.

Поэтому огнестойкость здания можно определить как третью (III).

Помещения лаборатории функциональной пожарной опасности относятся к категории ф 4.2 - высшие учебные заведения, учреждения повышения квалификации.

Пожар на рабочем месте может иметь очень негативные последствия (потеря ценной информации, материальный ущерб, гибель людей и т.д.), поэтому необходимо выявить и устранить все причины пожара, подготовить план действий по борьбе с пожаром в здании и план эвакуации людей, находящихся в здании.

Причиной возгорания может быть:

- поврежденные кабели, розетки и выключатели могут стать причиной короткого замыкания или проблем с изоляцией;
- использование неисправных/неработающих электроприборов;
- использование электрообогревателей с открытыми нагревательными элементами в помещениях;
- пожары, вызванные ударами молнии в здания;
- пожары в зданиях, вызванные внешними воздействиями;
- небрежное использование пожарных извещателей и несоблюдение мер пожарной безопасности.

2.7 Профилактика пожара

Пожарная безопасность - это комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, предотвращение пожаров, ограничение их распространения и создание условий для эффективного тушения пожаров. Для обеспечения пожарной безопасности важно правильно оценить пожарный риск в здании, выявить факторы риска и пересмотреть методы и средства противопожарной защиты.

Одним из условий обеспечения пожарной безопасности является устранение возможных источников возгорания.

На рабочем месте источниками воспламенения могут быть:

- повреждение электрооборудования, проводки, розеток и выключателей. Для предотвращения пожаров по этим причинам необходимо немедленно обнаруживать и устранять неисправности, проводить регулярные осмотры и немедленно устранять любые дефекты;

- электрические неисправности. Меры по предотвращению пожаров включают своевременный ремонт электроустановок, устранение дефектов качества и недопущение использования неисправных электроустановок;

- отопление помещений электрообогревателями с открытыми нагревательными элементами. Открытые нагретые поверхности могут стать причиной пожара из-за находящихся в помещении книг, бумаг и пособий; бумага - легковоспламеняющийся материал. Во избежание пожаров я рекомендую не использовать в лаборатории открытые нагревательные приборы;

- короткое замыкание. Накройте кабели, чтобы снизить риск возгорания из-за короткого замыкания;

- молния ударяет в здание. Во время летних гроз молния может ударить и вызвать пожар. Чтобы предотвратить это, рекомендуется установить молниеотвод на крыше здания;

- отсутствие пожарной безопасности и задымленность в доме также могут стать причиной пожара. Для предотвращения пожаров, вызванных задымлением в лаборатории, рекомендуется полностью запретить курение и разрешить его только в строго разграниченных зонах.

Для предотвращения пожаров рекомендуется провести инструктаж по пожаротушению с программистами, работающими в помещении, чтобы ознакомить персонал с правилами пожарной безопасности и обучить применению основных методов пожаротушения.

В случае пожара отключите питание, вызовите пожарную бригаду, эвакуируйтесь из помещения в соответствии с планом эвакуации и потушите

огонь огнетушителем. При наличии небольшого пламени не допускайте попадания воздуха на объект возгорания.

Заключение

В процессе работы над ВКР было разработано приложения для расчёта заработной платы. Использование разработки значительно упрощает работу бухгалтеру. В процессе работы были автоматизированы следующие процессы: графики удержания, контроль качества сотрудников, определение КТУ по каждому показателю за месяц, написание отчетов.

В разделе анализа состояния вопроса дипломного проекта проведён анализ работы бухгалтера, выявлена экономическая сущность расчёта заработной платы, обоснована необходимость и цели использования приложения для решения задач, проведён анализ существующих разработок.

В разделе проектирования информационной системы построена информационная модель системы, которая является обобщённой схемой движения информационных потоков предприятия.

В разделе реализации информационной системы было разработано программное обеспечение с учётом устранения выявленных при анализе недостатков, дано описание реализации клиентской части и базы данных, описана технология работы с приложением.

В разделе безопасность жизнедеятельности были учтены и разработаны требования к безопасному использованию приложения.

Разработанное приложение имеет понятный графический интерфейс, что позволяет значительно сэкономить время при вводе, обработке, просмотре и редактировании данных.

Построены основные *UML*-диаграммы: диаграммы классов, диаграммы деятельности, диаграмма развертывания и диаграмма вариантов использования.

На основании вышесказанного можно сделать вывод о том, что разработка приложения расчёта заработной платы является целесообразной.

Список использованных источников

1. Белоногов, Г. Г. Автоматизация процессов накопления, поиска и обобщения информации. [Текст] / Г. Г. Белоногов, А. П. Новоселов. – М.: Наука, 2017. - 256 с.
2. Вдовин, В. М. Предметно-ориентированные экономические информационные системы: Учебное пособие. [Текст] / В. М. Вдовин, Л. Е. Суркова и др. – М.: Дашков и К, 2016. – 388 с.
3. Балдин, К. В Информационные системы в экономике: Учебник. [Текст] / К. В Балдин, В. Б. Уткин. – М.: Дашков и К, 2015. – 395 с.
4. Баушев, С. В. Удостоверяющие автоматизированные информационные системы и средства. Введение в теорию и практику. [Текст] / С. В. Баушев. – СПб.: BHV, 2016. – 304 с.
5. Гросс, К. C# 2008 и платформа .NET 3.5 Framework. [Текст] / К. Гросс. – М.: Вильямс, 2016. – 480 с.
6. Вагнер, Билл C# Эффективное программирование. [Текст] / Билл Вагнер. – М.: ЛОРИ, 2017. – 320 с.
7. Грэй, П. Логика, алгебра и базы данных. [Текст] / П. Грэй. – М.: Машиностроение, 2015. – 368 с.
8. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. [Текст] / К. Дж. Дейт. – М.: Символ-плюс, 2017. – 480 с.

Приложение А

(Обязательное)

Приложение для расчета заработной платы сотрудников коммерческой
организации

Листинг программы

					ПГУ 1.090301.31.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Файл EditRaschetZP.cs

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ZP

{

    public partial class EditRaschetZP : Form

    {

        string ID;

        public EditRaschetZP(string id)

        {

            InitializeComponent();

            ID = id;

        }

        private void EditRaschetZP_Load(object sender, EventArgs e)

        {

            // TODO: данная строка кода позволяет загрузить данные в таблицу

            "zPDataSet.Сотрудник". При необходимости она может быть перемещена или

            удалена.

            this.сотрудникTableAdapter.Fill(this.zPDataSet.Сотрудник);

        }

    }

}
```

```

        if (int.Parse(ID) > 0)
        {
            DataTable tres = new DataTable();

            string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

            using (SqlConnection conn = new
SqlConnection(connectionString))
            {
                using (SqlDataAdapter da = new SqlDataAdapter())
                {
                    da.SelectCommand = new SqlCommand("SELECT *"
                        + " FROM Начисление"
                        + " WHERE Код = " + ID, conn); ;

                    DataSet ds = new DataSet();

                    da.Fill(ds, "result_name");

                    tres = ds.Tables["result_name"];

                    textBox1.Text =
Math.Round(decimal.Parse(tres.Rows[0]["Начислено_ЗП"].ToString()),
2).ToString();

                    textBox2.Text =
Math.Round(decimal.Parse(tres.Rows[0]["Начислено_премии"].ToString()),
2).ToString();

                    textBox4.Text =
Math.Round(decimal.Parse(tres.Rows[0]["Налоги_ЗП"].ToString()),
2).ToString();

                    textBox3.Text =
Math.Round(decimal.Parse(tres.Rows[0]["Налоги_премия"].ToString()),
2).ToString();

                    textBox5.Text =
(Math.Round(decimal.Parse(tres.Rows[0]["Налоги_ЗП"].ToString()), 2)+

```

```

Math.Round(decimal.Parse(tres.Rows[0]["Налоги_премия"].ToString()),
2)).ToString();

        textBox6.Text =
(Math.Round(decimal.Parse(tres.Rows[0]["Начислено_ЗП"].ToString()), 2) +
Math.Round(decimal.Parse(tres.Rows[0]["Начислено_премии"].ToString()),
2)).ToString();

        textBox7.Text =
((Math.Round(decimal.Parse(tres.Rows[0]["Начислено_ЗП"].ToString()), 2) +
Math.Round(decimal.Parse(tres.Rows[0]["Начислено_премии"].ToString()), 2)) -
(Math.Round(decimal.Parse(tres.Rows[0]["Налоги_ЗП"].ToString()), 2) +
Math.Round(decimal.Parse(tres.Rows[0]["Налоги_премия"].ToString()),
2))).ToString();

        dateTimePicker1.Value =
DateTime.Parse(tres.Rows[0]["Дата_начисления"].ToString());

        dateTimePicker2.Value =
DateTime.Parse(tres.Rows[0]["Дата_выплаты"].ToString());

    }

    }

    setZP_Premiya();

    calcNalogi();

}

else

{
    textBox1.Text = "0"; textBox2.Text = "0";
    dateTimePicker1.Checked = false; dateTimePicker2.Checked = false;
}

}

private void setZP_Premiya()
{
    textBox1.Text = "0";

    textBox2.Text = "0";

    DataTable tres = new DataTable();

    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

```

```

using (SqlConnection conn = new SqlConnection(connectionString))
{
    using (SqlDataAdapter da = new SqlDataAdapter())
    {
        da.SelectCommand = new SqlCommand("SELECT Оклад"
            + " FROM Должность d INNER JOIN Сотрудник s ON
s.Должность = d.Код"
            + " WHERE s.Код =
"+comboBox1.SelectedValue.ToString(), conn);;

        DataSet ds = new DataSet();
        da.Fill(ds, "result_name");

        tres = ds.Tables["result_name"];

        textBox1.Text =
Math.Round(decimal.Parse(tres.Rows[0][0].ToString()), 2).ToString();
    }
}

string fullMonthName = new DateTime(dateTimePicker1.Value.Year,
dateTimePicker1.Value.Month, 1).ToString("MMMM",
CultureInfo.CreateSpecificCulture("ru"));

DataTable dt, dt2;

using (SqlConnection conn = new SqlConnection(connectionString))
{
    using (SqlDataAdapter da = new SqlDataAdapter())
    {

```

```

        da.SelectCommand = new SqlCommand("SELECT Сотрудник.Код,
Сотрудник.ФИО, Должность.Название, Должность.Оклад, Sum([КТУ
сотрудника].Процент_премии) AS [SumПроцент], [КТУ сотрудника].Месяц, [КТУ
сотрудника].Год"

        + " FROM Должность INNER JOIN (КТУ INNER
JOIN (Сотрудник INNER JOIN [КТУ сотрудника] ON Сотрудник.Код = [КТУ
сотрудника].Сотрудник) ON КТУ.Код = [КТУ сотрудника].КТУ) ON Должность.Код =
Сотрудник.Должность"

        + " WHERE [КТУ сотрудника].Год = " +
dateTimePicker1.Value.Year.ToString() + " and [КТУ сотрудника].Месяц = '" +
fullMonthName + "' and Сотрудник.Код = "+ comboBox1.SelectedValue.ToString()

        + " GROUP BY Сотрудник.Код, Сотрудник.ФИО,
Должность.Название, Должность.Оклад, [КТУ сотрудника].Месяц, [КТУ
сотрудника].Год", conn);

        DataSet ds = new DataSet();

        da.Fill(ds, "result_name");

        dt2 = ds.Tables["result_name"];

    }

}

int rabd;

if (System.DateTime.DaysInMonth(dateTimePicker1.Value.Year,
dateTimePicker1.Value.Month) == 30) rabd = 15; else rabd = 16;

int i = 0;

string sqlExpression2 = "";

foreach (DataRow row in dt2.Rows)
{

    sqlExpression2 = "SELECT Табель.* FROM Табель"

        + " WHERE Сотрудник = " + row["Код"].ToString()

        + " and Месяц = '" + fullMonthName + "'"

        + " and Год = " +
dateTimePicker1.Value.Year.ToString();

```

```

        int sk1 = 0;

        using (SqlConnection conn = new
SqlConnection(connectionString))
        {
            using (SqlDataAdapter da = new SqlDataAdapter())
            {
                da.SelectCommand = new SqlCommand(sqlExpression2,
conn);

                DataSet ds = new DataSet();
                da.Fill(ds, "result_name");

                dt = ds.Tables["result_name"];

                sk1 = 0;
                for (int j = 1; j < dt.Columns.Count - 4; j++)
                {
                    if (dt.Columns[3 + j].ToString() == "2") sk1++;
                }
            }
        }

        double oklad = double.Parse(row["Оклад"].ToString());
        int SumПроцент = (int)row["SumПроцент"];

        textBox2.Text = Math.Round((oklad * ((double)SumПроцент /
100)) / rabd * (rabd - sk1), 2).ToString();

        i++;

```

```

    }

}

private void calcsum()
{
    decimal nalog_zp = 0;
    decimal nalog_premiya = 0;
    decimal nalog_summa = 0;
    for (int i = 0; i < dataGridView1.Rows.Count; ++i)
    {
        nalog_zp +=
Convert.ToDecimal(dataGridView1.Rows[i].Cells[2].Value);

        nalog_premiya +=
Convert.ToDecimal(dataGridView1.Rows[i].Cells[3].Value);
    }

    nalog_summa += nalog_zp + nalog_premiya;

    textBox4.Text = Math.Round(nalog_zp, 2).ToString();
    textBox3.Text = Math.Round(nalog_premiya, 2).ToString();
    textBox5.Text = Math.Round(nalog_summa, 2).ToString();

    textBox6.Text = (decimal.Parse(textBox1.Text) +
decimal.Parse(textBox2.Text)).ToString();

    textBox7.Text = (decimal.Parse(textBox6.Text) -
decimal.Parse(textBox5.Text)).ToString();
}

private void calcNalogi()
{
    DataTable tres = new DataTable();

```

```

        string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            using (SqlDataAdapter da = new SqlDataAdapter())
            {
                da.SelectCommand = new SqlCommand("SELECT
Название,Процент,round([Сумма налога ЗП],2) as [Сумма налога ЗП],round([Сумма
налога премии],2) as [Сумма налога премии],round([Сумма налога ЗП]+[Сумма
налога премии],2) as [Сумма налога]"

                + " FROM("

                + " SELECT Название, Процент,"

                + " case "

                + "      when Применять_на_оклад = 1 then Процент * " +
textBox1.Text.Replace(",",".") + " / 100 else 0"

                + " end as [Сумма налога ЗП],"

                + " case "

                + "      when Применять_на_премию = 1 then Процент* " +
textBox2.Text.Replace(",",".") + " / 100 else 0"

                + " end as [Сумма налога премии]"

                + " FROM dbo.Налоги n) t", conn);

                DataSet ds = new DataSet();

                da.Fill(ds, "result_name");

                tres = ds.Tables["result_name"];

                dataGridView1.DataSource = tres;

                dataGridView1.Columns[0].Width = 200;

                calcsun();

```

```

    }

    }

}

private void button1_Click(object sender, EventArgs e)
{
    if (!dateTimePicker1.Checked) { MessageBox.Show("Укажите дату
расчета начисления!"); dateTimePicker1.Focus();return; }

    if (comboBox1.Text.Length == 0) { MessageBox.Show("Укажите
сотрудника!"); comboBox1.Focus(); return; }

    setZP_Premiya();

    calcNalogi();

}

private void EditRaschetZP_FormClosing(object sender,
FormClosingEventArgs e)
{
    string sql;

    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

    SqlConnection conn = new SqlConnection(connectionString);

    if (int.Parse(ID) > 0)
    {
        sql = "UPDATE Начисление SET Сотрудник =
@Сотрудник, Начислено_ЗП=@Начислено_ЗП, Начислено_премии=@Начислено_премии, "
        + " Налоги_ЗП=@Налоги_ЗП, Налоги_премия =
@Налоги_премия, Дата_начисления=@Дата_начисления, Дата_выплаты=@Дата_выплаты"
        + " WHERE Код = @Код";

        SqlCommand cmd = new SqlCommand();

```

```

cmd.Connection = conn;

cmd.CommandText = sql;

cmd.Parameters.AddWithValue ("Сотрудник",
comboBox1.SelectedValue.ToString());

cmd.Parameters.AddWithValue ("Начислено_ЗП",
decimal.Parse(textBox1.Text));

cmd.Parameters.AddWithValue ("Начислено_премии",
decimal.Parse(textBox2.Text));

cmd.Parameters.AddWithValue ("Налоги_ЗП",
decimal.Parse(textBox4.Text));

cmd.Parameters.AddWithValue ("Налоги_премия",
decimal.Parse(textBox3.Text));

if (dateTimePicker1.Checked)
{
    cmd.Parameters.AddWithValue ("Дата_начисления",
DateTime.Parse(dateTimePicker1.Value.ToString()));
}
else
{
    cmd.Parameters.AddWithValue ("Дата_начисления",
DBNull.Value);
}

if (dateTimePicker2.Checked)
{
    cmd.Parameters.AddWithValue ("Дата_выплаты",
DateTime.Parse(dateTimePicker2.Value.ToString()));
}
else
{
    cmd.Parameters.AddWithValue ("Дата_выплаты",
DBNull.Value);
}

cmd.Parameters.AddWithValue ("Код", int.Parse(ID));

conn.Open();

```

```

        cmd.ExecuteNonQuery();

        conn.Close();

    }

    else

    {

        sql = "INSERT INTO Начисление
(Сотрудник, Начислено_ЗП, Начислено_премии, Налоги_ЗП, Налоги_премия, Дата_начисле
ния, Дата_выплаты) "

        + "
VALUES (@Сотрудник, @Начислено_ЗП, @Начислено_премии, @Налоги_ЗП, @Налоги_премия, @
Дата_начисления, @Дата_выплаты) ";

        SqlCommand cmd = new SqlCommand();

        cmd.Connection = conn;

        cmd.CommandText = sql;

        cmd.Parameters.AddWithValue("Сотрудник",
comboBox1.SelectedValue.ToString());

        cmd.Parameters.AddWithValue("Начислено_ЗП",
decimal.Parse(textBox1.Text));

        cmd.Parameters.AddWithValue("Начислено_премии",
decimal.Parse(textBox2.Text));

        cmd.Parameters.AddWithValue("Налоги_ЗП",
decimal.Parse(textBox4.Text));

        cmd.Parameters.AddWithValue("Налоги_премия",
decimal.Parse(textBox3.Text));

        if (dateTimePicker1.Checked)

        {

            cmd.Parameters.AddWithValue("Дата_начисления",
DateTime.Parse(dateTimePicker1.Value.ToString()));

        }

        else

        {

            cmd.Parameters.AddWithValue("Дата_начисления",
DBNull.Value);

```

```

        }

        if (dateTimePicker2.Checked)
        {
            cmd.Parameters.AddWithValue("Дата_выплаты",
DateTime.Parse(dateTimePicker2.Value.ToString()));

        }

        else
        {

            cmd.Parameters.AddWithValue("Дата_выплаты",
DBNull.Value);

        }

        conn.Open();

        cmd.ExecuteNonQuery();

        conn.Close();

    }

}

}

```

Файл RaschetZP.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

```

					ПГУ 1.090301.31.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

```

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ZP
{
    public partial class RaschetZP : Form
    {
        public RaschetZP()
        {
            InitializeComponent();

            private void начислениеBindingNavigatorSaveItem_Click(object sender,
            EventArgs e)
            {
                this.Validate();

                this.начислениеBindingSource.EndEdit();

                this.tableAdapterManager.UpdateAll(this.zPDataSet);

            }

            private void RaschetZP_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу
                "zPDataSet.Сотрудник". При необходимости она может быть перемещена или
                удалена.

                this.сотрудникTableAdapter.Fill(this.zPDataSet.Сотрудник);

                // TODO: данная строка кода позволяет загрузить данные в таблицу
                "zPDataSet.Начисление". При необходимости она может быть перемещена или
                удалена.

                this.начислениеTableAdapter.Fill(this.zPDataSet.Начисление);

```

```

    }

    private void button1_Click(object sender, EventArgs e)
    {
        EditRaschetZP f = new EditRaschetZP("0");

        f.ShowDialog();

        this.начислениеTableAdapter.Fill(this.zPDataSet.Начисление);
    }

    private void button2_Click(object sender, EventArgs e)
    {
        EditRaschetZP f = new
        EditRaschetZP(начислениеDataGridView.Rows[начислениеDataGridView.CurrentRow.
        RowIndex].Cells[0].Value.ToString());

        f.ShowDialog();

        this.начислениеTableAdapter.Fill(this.zPDataSet.Начисление);
    }

    private void button3_Click(object sender, EventArgs e)
    {
        DialogResult dialogResult = MessageBox.Show("Вы уверены что
        хотите удалить?", "Удаление", MessageBoxButtons.YesNo);

        if (dialogResult == DialogResult.Yes)
        {
            SqlConnection myConnection;

            myConnection = new
            System.Data.SqlClient.SqlConnection(ConfigurationManager.ConnectionStrings["Z
            P.Properties.Settings.ZPConnectionString"].ConnectionString);

            SqlCommand cmd = new SqlCommand("Empty", myConnection);

```

```

        cmd.CommandText = "DELETE FROM Начисление WHERE Код = @Код;";

        cmd.Parameters.AddWithValue("Код",
начислениеDataGridView.Rows[начислениеDataGridView.CurrentRow.Index].Cells[0].Value.ToString());

        myConnection.Open();

        cmd.ExecuteNonQuery();

        cmd.Parameters.Clear();

        myConnection.Close();

        this.начислениеTableAdapter.Fill(this.zPDataSet.Начисление);
    }
}
}
}

```

Файл Form1.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace ZP
{

```

					ПГУ 1.090301.31.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void должникиToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        Dolzhnost f = new Dolzhnost();
        f.ShowDialog();
    }

    private void сотрудникиToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Sotrudniki f = new Sotrudniki();
        f.ShowDialog();
    }

    private void кТУToolStripMenuItem_Click(object sender, EventArgs e)
    {
        KTU f = new KTU();
        f.ShowDialog();
    }

    private void выходToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
}

```

```

e) private void отметкаToolStripMenuItem_Click(object sender, EventArgs
{
    Otmetka f = new Otmetka();
    f.ShowDialog();
}

```

```

e) private void отделиToolStripMenuItem_Click(object sender, EventArgs
{
    Otdeli f = new Otdeli();
    f.ShowDialog();
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    Pretenzii f = new Pretenzii();
    f.ShowDialog();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    KTU_Sotrud f = new KTU_Sotrud();
    f.ShowDialog();
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    Tabel f = new Tabel();
    f.ShowDialog();
}

```

```

    }

    private void Form1_Load(object sender, EventArgs e)
    {
        int curmonth = DateTime.Now.Month;

        string[] months = { "Январь", "Февраль", "Март", "Апрель", "Май",
            "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь" };

        comboBox1.SelectedIndex = curmonth - 1;

        textBox1.Text = DateTime.Now.Year.ToString();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        // создаем пустую книгу и объявляем переменные
        Microsoft.Office.Interop.Excel.Application xlexcel;
        Microsoft.Office.Interop.Excel.Workbook xlWorkBook;
        Microsoft.Office.Interop.Excel.Worksheet xlWorkSheet;
        object misValue = System.Reflection.Missing.Value;

        xlexcel = new Excel.Application();
        xlexcel.Visible = true;

        xlWorkBook = xlexcel.Workbooks.Add(misValue);

        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

        DataTable tres = new DataTable();

        string connectionString =
            System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
            ttings.ZPConnectionString"].ConnectionString;

        using (SqlConnection conn = new SqlConnection(connectionString))
        {

```

```

        using (SqlDataAdapter da = new SqlDataAdapter())
        {
            da.SelectCommand = new SqlCommand("SELECT Код, ФИО FROM
Сотрудник", conn);

            DataSet ds = new DataSet();
            da.Fill(ds, "result_name");

            tres = ds.Tables["result_name"];
        }
    }

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        using (SqlDataAdapter da = new SqlDataAdapter())
        {
            da.SelectCommand = new SqlCommand("SELECT Название FROM
Отметка", conn);

            DataSet ds = new DataSet();
            da.Fill(ds, "result_name");

            DataTable dt = ds.Tables["result_name"];

            foreach (DataRow row in dt.Rows)
            {
                tres.Columns.Add(row["Название"].ToString(),
typeof(System.Int32));
            }
        }
    }

```

```

string sql = "Список_выработки";

using (SqlConnection conn = new SqlConnection(connectionString))
{
    using (SqlDataAdapter da = new SqlDataAdapter())
    {
        da.SelectCommand = new SqlCommand(sql, conn);

        da.SelectCommand.CommandType =
CommandType.StoredProcedure;

        DataSet ds = new DataSet();

        da.Fill(ds, "result_name");

        DataTable dt = ds.Tables["result_name"];

        foreach (DataRow row in dt.Rows)
        {
            DataRow dr = tres.Select("Код="+
row["Код_сотрудника"].ToString()).FirstOrDefault();

            if (dr != null)
            {
                dr[row["Отметка"].ToString()] =
row["Количество"]; //changes the Product_name
            }
        }
    }

    foreach (DataColumn col in tres.Columns)

```

```

        {

            xlWorkSheet.Cells[1, tres.Columns.IndexOf(col) + 1] =
col.ColumnName.ToString();

        }

        foreach (DataRow row in tres.Rows)

        {

            foreach (DataColumn col in tres.Columns)

            {

                xlWorkSheet.Cells[tres.Rows.IndexOf(row)+2,
tres.Columns.IndexOf(col)+1] = row[tres.Columns.IndexOf(col)];

            }

        }

        xlWorkSheet.Columns["A:A"].Hidden = true;

        //автовывравнивание колонок

        xlWorkSheet.Columns["A:AA"].AutoFit();

        //границы таблицы

        Excel.Range xlWorkSheet_rng = xlWorkSheet.get_Range("A1", "E" +
(tres.Rows.Count+1).ToString());

        xlWorkSheet_rng.Borders.ColorIndex = 0;

        xlWorkSheet_rng.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;

        xlWorkSheet_rng.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlThin;

        //сделать первую строку жирной

        xlWorkSheet.Cells[1, 1].EntireRow.Font.Bold = true;

    }

```

```

        private void налогриToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            Nalogi f = new Nalogi();
            f.ShowDialog();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            RaschetZP f = new RaschetZP();
            f.ShowDialog();
        }
    }
}

```

Файл Nalogi.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ZP
{
    public partial class Nalogi : Form

```

```

{
    public Nalogi()
    {
        InitializeComponent();
    }

    private void налогиBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
    {
        this.Validate();
        this.налогиBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.zPDataSet);
    }

    private void Nalogi_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "zPDataSet.Налоги". При необходимости она может быть перемещена или удалена.
        this.налогиTableAdapter.Fill(this.zPDataSet.Налоги);
    }
}

```

Файл KTU_Sotrud.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

```

```

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace ZP
{
    public partial class KTU_Sotrud : Form
    {
        public KTU_Sotrud()
        {
            InitializeComponent();
        }

        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void кТУ_сотрудникаBindingNavigatorSaveItem_Click(object
sender, EventArgs e)
        {
            this.Validate();

            this.кТУ_сотрудникаBindingSource.EndEdit();

            this.tableAdapterManager.UpdateAll(this.zPDataSet);
        }
    }
}

```

```

private void KTU_Sotrud_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные в таблицу
    "zPDataSet.KTY". При необходимости она может быть перемещена или удалена.

    this.кТУTableAdapter.Fill(this.zPDataSet.KTY);

    // TODO: данная строка кода позволяет загрузить данные в таблицу
    "zPDataSet.Сотрудник". При необходимости она может быть перемещена или
    удалена.

    this.сотрудникTableAdapter.Fill(this.zPDataSet.Сотрудник);

    // TODO: данная строка кода позволяет загрузить данные в таблицу
    "zPDataSet.KTY_сотрудника". При необходимости она может быть перемещена или
    удалена.

    this.кТУ_сотрудникаTableAdapter.Fill(this.zPDataSet.KTY_сотрудника);

    int curmonth = DateTime.Now.Month;

    string[] months = { "Январь", "Февраль", "Март", "Апрель", "Май",
    "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь" };

    comboBox1.SelectedIndex = curmonth - 1;

    textBox1.Text = DateTime.Now.Year.ToString();

    кТУ_сотрудникаBindingSource.Filter = "Месяц = '" + comboBox1.Text
    + "' and Год = " + textBox1.Text;
}

private void кТУ_сотрудникаDataGridView_DefaultValuesNeeded(object
sender, DataGridViewRowEventArgs e)
{
    e.Row.Cells["dataGridViewTextBoxColumn7"].Value = comboBox1.Text;

    e.Row.Cells["dataGridViewTextBoxColumn6"].Value = textBox1.Text;
}

private void button1_Click(object sender, EventArgs e)

```

```

{

    // создаем пустую книгу и объявляем переменные
    Microsoft.Office.Interop.Excel.Application xlexcel;
    Microsoft.Office.Interop.Excel.Workbook xlWorkBook;
    Microsoft.Office.Interop.Excel.Worksheet xlWorkSheet;

    object misValue = System.Reflection.Missing.Value;

    xlexcel = new Excel.Application();
    xlexcel.Visible = true;
    xlWorkBook = xlexcel.Workbooks.Add(misValue);
    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
    int ii = 0;
    DataTable dt,dt2;

    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

    string sqlExpression = "SELECT Count(KTY.Код) AS [Count-Код] FROM
КТУ";

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        using (SqlDataAdapter da = new SqlDataAdapter())
        {
            da.SelectCommand = new SqlCommand(sqlExpression, conn);

            DataSet ds = new DataSet();
            da.Fill(ds, "result_name");

            dt = ds.Tables["result_name"];

```

```

        foreach (DataRow row in dt.Rows)
        {
            ii = int.Parse(row[0].ToString());
        }
    }

    sqlExpression = "SELECT Сотрудник.ФИО, Должность.Название,
КТУ.Название AS КТУ, [КТУ сотрудника].Значение, "
                    +" [КТУ сотрудника].Процент_премии, [КТУ
сотрудника].Месяц FROM (Должность INNER JOIN Сотрудник"
                    +" ON Должность.Код = Сотрудник.Должность)
INNER JOIN (КТУ INNER JOIN [КТУ сотрудника] ON КТУ.Код = "
                    +" [КТУ сотрудника].КТУ) ON Сотрудник.Код =
[КТУ сотрудника].Сотрудник "
                    +" WHERE ((([КТУ
сотрудника].Год)="+textBox1.Text+")) "
                    +" AND ((([КТУ сотрудника].Месяц)='"+
comboBox1.Text + "')) "
                    +" ORDER BY Сотрудник.ФИО, КТУ.Название;";

    //MessageBox.Show(sqlExpression);

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        using (SqlDataAdapter da = new SqlDataAdapter())
        {
            da.SelectCommand = new SqlCommand(sqlExpression, conn);

            DataSet ds = new DataSet();
            da.Fill(ds, "result_name");
        }
    }

```

```

        dt2 = ds.Tables["result_name"];

    }

}

xlWorkSheet.Columns[1].ColumnWidth = 20;
xlWorkSheet.Columns[2].ColumnWidth = 30;

//задаем шапку таблицы
xlWorkSheet.Cells[1, 1] = "ФИО";

xlWorkSheet.Range[xlWorkSheet.Cells[1, 1], xlWorkSheet.Cells[2,
1]].Merge();

xlWorkSheet.Cells[1, 2] = "Должность";

xlWorkSheet.Range[xlWorkSheet.Cells[1, 2], xlWorkSheet.Cells[2,
2]].Merge();

int i = 0;
int j = 0;
while(j < ii * 2)
{
    xlWorkSheet.Columns[3 + j].ColumnWidth = 20;
    xlWorkSheet.Columns[4 + j].ColumnWidth = 20;

    xlWorkSheet.Cells[1, 3 + j] = dt2.Rows[i]["KTU"].ToString();

    xlWorkSheet.Range[xlWorkSheet.Cells.Item[1, 3 + j],
xlWorkSheet.Cells.Item[1, 3 + j + 1]].MergeCells = true;

    xlWorkSheet.Cells[2, 3 + j] = "Значение";

    xlWorkSheet.Cells[2, 4 + j] = "% премии";

    j += 2;

    i++;

}

```

```

xlWorkSheet.Columns[3 + j].ColumnWidth = 25;

xlWorkSheet.Cells[1, 3 + j] = "КОНЕЧНЫЙ ПОКАЗАТЕЛЬ КТУ";

xlWorkSheet.Cells[1, 3 + j].WrapText = true;

xlWorkSheet.Range[xlWorkSheet.Cells.Item[1, 3 + j],
xlWorkSheet.Cells.Item[2, 3 + j]].MergeCells = true;


int index = 3;

int numRows = dt2.Rows.Count;


//MessageBox.Show((numRows/ii).ToString());


for (i = 0; i<= numRows-1; i++)
{

    xlWorkSheet.Cells[index, 1] =
dt2.Rows[i].Field<string>("ФИО");

    xlWorkSheet.Cells[index, 2] =
dt2.Rows[i].Field<string>("Название");


    j= 0;

    int proc = 0;


    while (j < ii * 2)
    {

        xlWorkSheet.Cells[index, 3 + j] =
dt2.Rows[i].Field<int>("Значение");

        xlWorkSheet.Cells[index, 4 + j] =
dt2.Rows[i].Field<int>("Процент_премии");


        proc += dt2.Rows[i].Field<int>("Процент_премии");

        j += 2;

        if (j < ii * 2) i++;
    }
}

```

```

    }

    xlWorkSheet.Cells[index, 3 + j] = proc;

    index++;

}

//автовыравнивание колонок

//xlWorkSheet.Columns["A:H"].AutoFit();

//границы таблицы

Excel.Range xlWorkSheet_rng = xlWorkSheet.get_Range("A1", "M" +
(index - 1).ToString());

xlWorkSheet_rng.Borders.ColorIndex = 0;

xlWorkSheet_rng.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;

xlWorkSheet_rng.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlThin;

//сделать первую строку жирной

xlWorkSheet.Cells[1, 1].EntireRow.Font.Bold = true;

xlWorkSheet.Cells[2, 1].EntireRow.Font.Bold = true;

}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)

{

    if ((comboBox1.Text.Length>0) && (textBox1.Text.Length>0))

        кТУ_сотрудникаBindingSource.Filter = "Месяц = '" + comboBox1.Text
+ "' and Год = " + textBox1.Text;

}

private void textBox1_TextChanged(object sender, EventArgs e)

```

```

    {
        if ((comboBox1.Text.Length > 0) && (textBox1.Text.Length > 0))
            кТУ_сотрудникаBindingSource.Filter = "Месяц = '" +
comboBox1.Text + "' and Год = " + textBox1.Text;
    }

private void button2_Click(object sender, EventArgs e)
{
    // создаем пустую книгу и объявляем переменные
    Microsoft.Office.Interop.Excel.Application xlexcel;
    Microsoft.Office.Interop.Excel.Workbook xlWorkBook;
    Microsoft.Office.Interop.Excel.Worksheet xlWorkSheet;

    object misValue = System.Reflection.Missing.Value;

    xlexcel = new Excel.Application();
    xlexcel.Visible = true;
    xlWorkBook = xlexcel.Workbooks.Add(misValue);
    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
    int ii = 0;
    DataTable dt, dt2;

    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

    string sqlExpression = "SELECT Сотрудник.ФИО, Должность.Название,
Должность.Оклад, Sum([КТУ сотрудника].Процент_премии) AS [SumПроцент], [КТУ
сотрудника].Месяц, [КТУ сотрудника].Год"

        + " FROM Должность INNER JOIN (КТУ INNER
JOIN (Сотрудник INNER JOIN [КТУ сотрудника] ON Сотрудник.Код = [КТУ
сотрудника].Сотрудник) ON КТУ.Код = [КТУ сотрудника].КТУ) ON Должность.Код =
Сотрудник.Должность"

        + " WHERE [КТУ сотрудника].Год = " +
textBox1.Text + " and [КТУ сотрудника].Месяц = '" + comboBox1.Text + "'"

```

```

        + " GROUP BY Сотрудник.ФИО,
Должность.Название, Должность.Оклад, [КТУ сотрудника].Месяц, [КТУ
сотрудника].Год";

```

```

xlWorkSheet.Columns[1].ColumnWidth = 20;

```

```

xlWorkSheet.Columns[2].ColumnWidth = 30;

```

```

xlWorkSheet.Cells[1, 1]= "ФИО";

```

```

xlWorkSheet.Cells[1, 2]= "Должность";

```

```

xlWorkSheet.Cells[1, 3]= "Оклад в рублях";

```

```

xlWorkSheet.Cells[1, 4]= "Конечный показатель КТУ за месяц";

```

```

xlWorkSheet.Cells[1, 5]= "Кол-во рабочих дней в месяце";

```

```

xlWorkSheet.Cells[1, 6]= "Количество неявок";

```

```

xlWorkSheet.Cells[1, 7]= "Сумма премии";

```

```

using (SqlConnection conn = new SqlConnection(connectionString))

```

```

{

```

```

    using (SqlDataAdapter da = new SqlDataAdapter())

```

```

    {

```

```

        da.SelectCommand = new SqlCommand(sqlExpression, conn);

```

```

        DataSet ds = new DataSet();

```

```

        da.Fill(ds, "result_name");

```

```

        dt2 = ds.Tables["result_name"];

```

```

    }

```

```

}

```

```

int rabd;

```

```

        if (System.DateTime.DaysInMonth(int.Parse(textBox1.Text),
        comboBox1.SelectedIndex + 1) == 30) rabd = 15; else rabd= 16;

        int i = 0;

        string sqlExpression2 = "";

        foreach (DataRow row in dt2.Rows)
        {

            xlWorkSheet.Cells[2 + i, 1] = row["ФИО"].ToString();

            xlWorkSheet.Cells[2 + i, 2] = row["Название"].ToString();

            xlWorkSheet.Cells[2 + i, 3] = row["Оклад"];

            xlWorkSheet.Cells[2 + i, 4] = row["SumПроцент"];

            xlWorkSheet.Cells[2 + i, 5] = rabd.ToString();

            sqlExpression2 = "SELECT Табель.* FROM Табель"

                + " WHERE Сотрудник = (SELECT Код FROM Сотрудник WHERE"
        ФИО = '" + row["ФИО"].ToString() + "'"

                + " and Месяц = '" + comboBox1.Text + "'"

                + " and Год = '" + textBox1.Text;

            int skl = 0;

            using (SqlConnection conn = new
        SqlConnection(connectionString))

            {

                using (SqlDataAdapter da = new SqlDataAdapter())

                {

                    da.SelectCommand = new SqlCommand(sqlExpression2,
        conn);

                    DataSet ds = new DataSet();

                    da.Fill(ds, "result_name");

```

```

        dt = ds.Tables["result_name"];

        sk1 = 0;

        for (int j = 1; j < dt.Columns.Count - 4; j++)
        {
            if (dt.Columns[3 + j].ToString() == "2") sk1++;
        }
    }

    xlWorkSheet.Cells[2 + i, 6] = sk1;

    double oklad = double.Parse(row["Оклад"].ToString());

    int SumПроцент = (int)row["SumПроцент"];

    xlWorkSheet.Cells[2 + i, 7] = Math.Round((oklad *
((double)SumПроцент / 100)) / rabd * (rabd - sk1), 2);

    i++;
}

//автовыравнивание колонок

xlWorkSheet.Columns["A:G"].AutoFit();

//границы таблицы

Excel.Range xlWorkSheet_rng = xlWorkSheet.get_Range("A1", "G" +
(dt2.Rows.Count+1).ToString());

xlWorkSheet_rng.Borders.ColorIndex = 0;

xlWorkSheet_rng.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;

```

```

xlWorkSheet_rng.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlThin;

//сделать первую строку жирной
xlWorkSheet.Cells[1, 1].EntireRow.Font.Bold = true;
}
}
}

```

Файл Pretenzii.cs

```

using System;
using System.Activities.Expressions;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ZP
{
    public partial class Pretenzii : Form
    {
        public Pretenzii()
        {
            InitializeComponent();

```

```

    }

    private void претензияBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
    {
        this.Validate();

        this.претензияBindingSource.EndEdit();

        this.tableAdapterManager.UpdateAll(this.zPDataSet);
    }

    private void Pretenzii_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "zPDataSet.Сотрудник". При необходимости она может быть перемещена или
        удалена.

        this.сотрудникTableAdapter.Fill(this.zPDataSet.Сотрудник);

        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "zPDataSet.Претензия". При необходимости она может быть перемещена или
        удалена.

        this.претензияTableAdapter.Fill(this.zPDataSet.Претензия);

        претензияBindingSource.Filter = "Сотрудник = " +
        сотрудникDataGridView.Rows[сотрудникDataGridView.CurrentRow.Index].Cells[
        0].Value.ToString();
    }

    private void сотрудникDataGridView_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        претензияBindingSource.Filter = "Сотрудник = " +
        сотрудникDataGridView.Rows[сотрудникDataGridView.CurrentRow.Index].Cells[
        0].Value.ToString();
    }

```

```

private void button2_Click(object sender, EventArgs e)
{
    var fileContent = string.Empty;

    var filePath = string.Empty;

    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        connection.Open();

        SqlCommand command = new SqlCommand();

        command.Connection = connection;

        //command.CommandText = @"INSERT INTO Претензия VALUES
(@Название, @Дата, @Файлы, @Сотрудник)";

        command.CommandText = @"INSERT INTO Претензия (Название,
Дата, Файлы, Сотрудник) VALUES (@Название, @Дата, @Файлы, @Сотрудник)";

        command.Parameters.Add("@Название", SqlDbType.NVarChar, 250);

        command.Parameters.Add("@Дата", SqlDbType.DateTime, 0);

        command.Parameters.Add("@Файлы", SqlDbType.Image, 1000000);

        command.Parameters.Add("@Сотрудник", SqlDbType.Int, 0);

        using (OpenFileDialog openFileDialog = new OpenFileDialog())
        {
            openFileDialog.InitialDirectory = "c:\\";

            openFileDialog.Filter = "txt files (*.txt)|*.txt|All
files (*.*)|*.*";

            openFileDialog.FilterIndex = 2;

            openFileDialog.RestoreDirectory = true;

```

```

if (openFileDialog.ShowDialog() == DialogResult.OK)
{
    //Get the path of specified file
    filePath = openFileDialog.FileName;

    //Read the contents of the file into a stream
    var fileStream = openFileDialog.OpenFile();

    using (StreamReader reader = new
StreamReader(fileStream))
    {
        fileContent = reader.ReadToEnd();
    }

    // путь к файлу для загрузки
    string filename = filePath;
    // заголовок файла
    string title = openFileDialog.SafeFileName;
    // массив для хранения бинарных данных файла
    byte[] imageData;

    using (System.IO.FileStream fs = new
System.IO.FileStream(filename, FileMode.Open))
    {
        imageData = new byte[fs.Length];
        fs.Read(imageData, 0, imageData.Length);
    }

    // передаем данные в команду через параметры
    command.Parameters["@Название"].Value = title;
    command.Parameters["@Дата"].Value = DateTime.Today;
    command.Parameters["@Файлы"].Value = imageData;

```

```

        command.Parameters["@Сотрудник"].Value =
(Int32)сотрудникDataGridView.Rows[сотрудникDataGridView.CurrentRow.Index]
.Cells[0].Value;

        command.ExecuteNonQuery();

    }

}

}

this.претензияTableAdapter.Fill(this.zPDataSet.Претензия); //обновляем данные
в таблице

}

private void button1_Click(object sender, EventArgs e)
{
    ReadFileFromDatabase();
}

private void ReadFileFromDatabase()
{
    string connectionString =
System.Configuration.ConfigurationManager.ConnectionStrings["ZP.Properties.Se
ttings.ZPConnectionString"].ConnectionString;

    List<Image> images = new List<Image>();

    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        connection.Open();

        string sql = "SELECT Код, Название, Название, Файлы FROM
Претензия WHERE Код = " +
претензияDataGridView.Rows[претензияDataGridView.CurrentRow.Index].Cells[
0].Value.ToString();

        SqlCommand command = new SqlCommand(sql, connection);

        SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())

```

```

        {

            int id = reader.GetInt32(0);

            string filename = reader.GetString(1);

            string title = reader.GetString(2);

            byte[] data = (byte[])reader.GetValue(3);


            Image image = new Image(id, filename, title, data);

            images.Add(image);

        }

    }

    // сохраним первый файл из списка

    if (images.Count > 0)

    {

        using (System.IO.FileStream fs = new
System.IO.FileStream(images[0].FileName, FileMode.OpenOrCreate))

        {

            fs.Write(images[0].Data, 0, images[0].Data.Length);

            Console.WriteLine("Изображение '{0}' сохранено",
images[0].Title);

            Process.Start(Directory.GetCurrentDirectory()+"\\"+
images[0].FileName);

        }

    }

}

private void button3_Click(object sender, EventArgs e)

{

    претензияBindingSource.RemoveCurrent();

    this.Validate();

    this.претензияBindingSource.EndEdit();

    this.tableAdapterManager.UpdateAll(this.zPDataSet);

}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.31.001 ПЗ

Лист

100

```

    }

    public class Image
    {
        public Image(int id, string filename, string title, byte[] data)
        {
            Id = id;
            FileName = filename;
            Title = title;
            Data = data;
        }

        public int Id { get; private set; }
        public string FileName { get; private set; }
        public string Title { get; private set; }
        public byte[] Data { get; private set; }
    }
}

```

Файл Otdeli.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ZP
{
    public partial class Otdeli : Form

```

```

{

    public Otdeli()

    {

        InitializeComponent();

    }


    private void отделиBindingNavigatorSaveItem_Click(object sender,
EventArgs e)

    {

        this.Validate();

        this.отделиBindingSource.EndEdit();

        this.tableAdapterManager.UpdateAll(this.zPDataSet);

    }


    private void Otdeli_Load(object sender, EventArgs e)

    {

        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "zPDataSet.Отделы". При необходимости она может быть перемещена или удалена.

        this.отделыTableAdapter.Fill(this.zPDataSet.Отделы);

    }

}

```

Файл Otmetka.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

```

```

namespace ZP
{
    public partial class Otmetka : Form
    {
        public Otmetka()
        {
            InitializeComponent();

            private void otmetkaBindingNavigatorSaveItem_Click(object sender,
            EventArgs e)
            {
                this.Validate();

                this.otmetkaBindingSource.EndEdit();

                this.tableAdapterManager.UpdateAll(this.zPDataSet);

            }

            private void Otmetka_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу
                "zPDataSet.Отметка". При необходимости она может быть перемещена или удалена.

                this.otmetkaTableAdapter.Fill(this.zPDataSet.Отметка);

            }

        }
    }
}

```

Файл KTU.cs

```

using System;

using System.Collections.Generic;

```

```

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ZP

{

    public partial class KTU : Form

    {

        public KTU()

        {

            InitializeComponent();

        }

        private void κTVBindingNavigatorSaveItem_Click(object sender,
EventArgs e)

        {

            this.Validate();

            this.κTVBindingSource.EndEdit();

            this.tableAdapterManager.UpdateAll(this.zPDataSet);

        }

        private void KTU_Load(object sender, EventArgs e)

        {

            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "zPDataSet.KTU". При необходимости она может быть перемещена или удалена.

            this.κTVTableAdapter.Fill(this.zPDataSet.KTU);

        }

    }

}

```

Файл Uvolnenie.cs

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ZP

{

    public partial class Uvolnenie : Form

    {

        public Uvolnenie()

        {

            InitializeComponent();

        }

        private void Uvolnenie_Load(object sender, EventArgs e)

        {

            // TODO: данная строка кода позволяет загрузить данные в таблицу

            "zPDataSet.Увольнение". При необходимости она может быть перемещена или

            удалена.

            this.увольнениеTableAdapter.Fill(this.zPDataSet.Увольнение);

        }

        private void button2_Click(object sender, EventArgs e)

        {

            Close();

        }

    }

}
```

					ПГУ 1.090301.31.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		

```

private void button1_Click(object sender, EventArgs e)
{
    this.Validate();
    this.увольнениеBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.zPDataSet);
    Close();
}
}
}

```

Файл Sotrudniki.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ZP
{
    public partial class Sotrudniki : Form
    {
        public Sotrudniki()
        {
            InitializeComponent();
        }
    }
}

```

```

        private void сотрудникBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            this.сотрудникBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.zPDataSet);
        }

        private void Sotrudniki_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "zPDataSet.Отделы". При необходимости она может быть перемещена или удалена.

            this.отделыTableAdapter.Fill(this.zPDataSet.Отделы);

            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "zPDataSet.Должность". При необходимости она может быть перемещена или
            удалена.

            this.должностьTableAdapter.Fill(this.zPDataSet.Должность);

            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "zPDataSet.Сотрудник". При необходимости она может быть перемещена или
            удалена.

            this.сотрудникTableAdapter.Fill(this.zPDataSet.Сотрудник);
        }
    }
}

```

Файл Dolzhnost.cs

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

```

```

using System.Windows.Forms;

namespace ZP
{
    public partial class Dolzhnost : Form
    {
        public Dolzhnost()
        {
            InitializeComponent();
        }

        private void должностьBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            this.должностьBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.zPDataSet);
        }

        private void Dolzhnost_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "zPDataSet.Должность". При необходимости она может быть перемещена или
            удалена.

            this.должностьTableAdapter.Fill(this.zPDataSet.Должность);
        }
    }
}

```

Файл Program.cs

```

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

```

```

using System.Windows.Forms;

namespace ZP
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

Файл AssemblyInfo.cs

using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// Общие сведения об этой сборке предоставляются следующим набором
// набора атрибутов. Измените значения этих атрибутов для изменения сведений,
// связанных со сборкой.

[assembly: AssemblyTitle("ZP")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]

```

```

[assembly: AssemblyProduct("ЗР")]

[assembly: AssemblyCopyright("Copyright © 2020")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

// Установка значения False для параметра ComVisible делает типы в этой
// сборке невидимыми
// для компонентов COM. Если необходимо обратиться к типу в этой сборке через
// COM, следует установить атрибут ComVisible в TRUE для этого типа.

[assembly: ComVisible(false)]

// Следующий GUID служит для идентификации библиотеки типов, если этот проект
// будет видимым для COM

[assembly: Guid("8f1a889a-615c-415c-b4db-28f520588a86")]

// Сведения о версии сборки состоят из указанных ниже четырех значений:
//
//      Основной номер версии
//      Дополнительный номер версии
//      Номер сборки
//      Редакция
//
// Можно задать все значения или принять номера сборки и редакции по
// умолчанию
// используя "*", как показано ниже:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]

```

Приложение Б

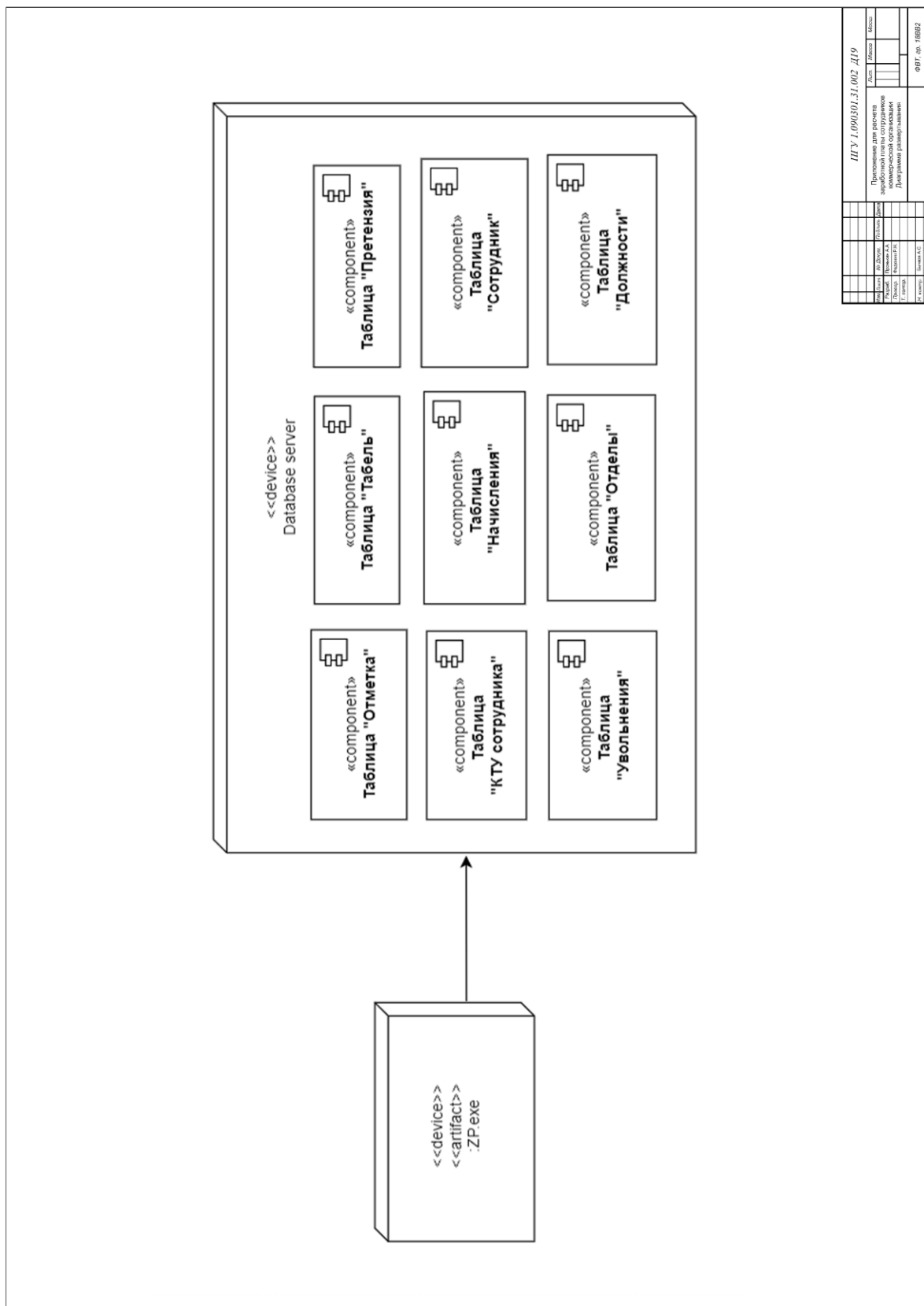
(Обязательное)

Приложение для расчета заработной платы сотрудников коммерческой
организации

UML-диаграммы

					ПГУ 1.090301.31.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Б.1 - Диаграмма развертывания

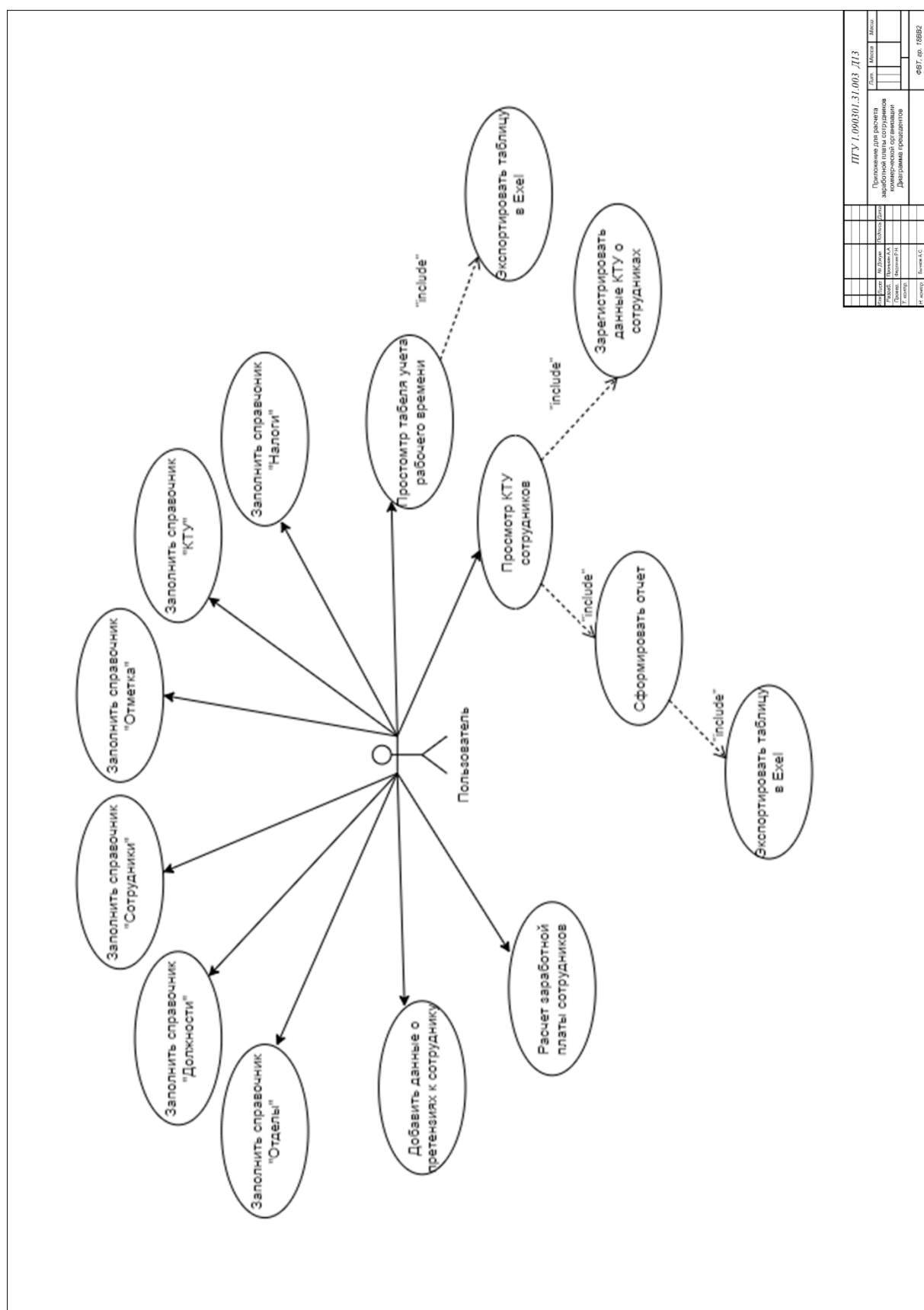


ПГУ 1.090301.31.002 Д19			
Изм.	Лист	Всего	Дата
1	1	1	
Приложение для расчета заработной платы сотрудников			
Диаграмма развертывания			
ФОРМ № 100002			

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.31.001 ПЗ

Приложение Б.2 - Диаграмма прецедентов

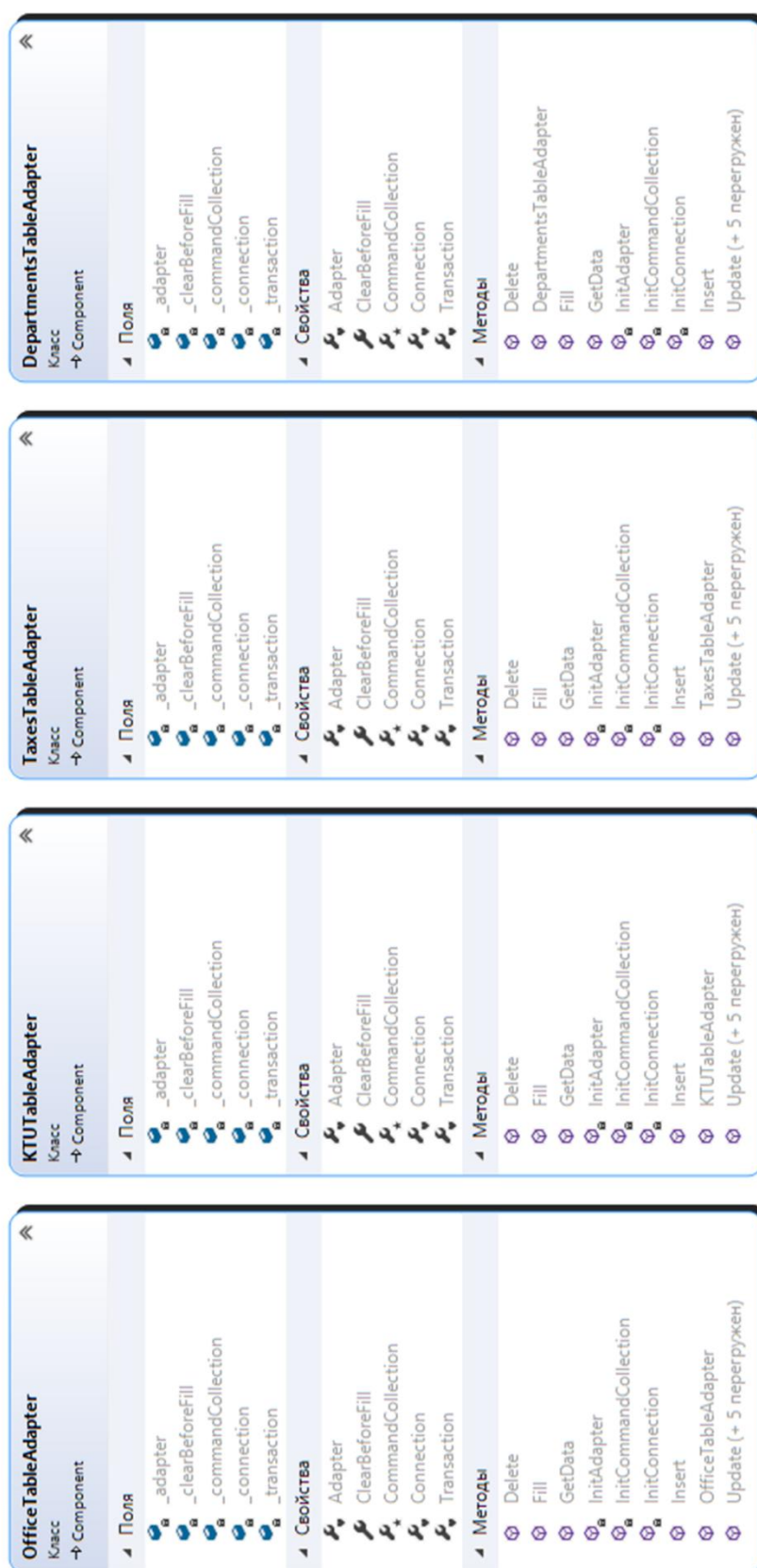


Приложение Б.3 - Диаграмма классов

Nalogi	Klacc	→ Form	
↓ Поля			
bindingNavigatorAddNewItem	a		
bindingNavigatorCountItem	a		
bindingNavigatorDeleteItem	a		
bindingNavigatorMoveFirstItem	a		
bindingNavigatorMoveLastItem	a		
bindingNavigatorMoveNextItem	a		
bindingNavigatorMovePreviousItem	a		
bindingNavigatorPositionItem	a		
bindingNavigatorSeparator1	a		
bindingNavigatorSeparator2	a		
components	a		
dataGridViewCheckBoxColumn1	a		
dataGridViewCheckBoxColumn2	a		
dataGridViewTextBoxColumn1	a		
dataGridViewTextBoxColumn2	a		
dataGridViewTextBoxColumn3	a		
tableAdapterManager	a		
TaxBindingNavigator	a		
TaxBindingNavigatorSaveItem	a		
TaxBindingSource	a		
TaxDataGridview	a		
TaxTableAdapter	a		
zpDataSet	a		
↓ Методы			
Dispose	a		
InitializeComponent	a		
Nalogi	a		
Nalogi_Load	a		
TaxBindingNavigatorSaveItem_Click	a		

[illegible]

Приложение Б.4 - Диаграмма классов



ПТУ 1.090301.31.005.211									
Изм.	Лист	№ докум.	Подпись	Дата	Приложение Б.4 - Диаграмма классов				
					Диаграмма классов				
					ФОРМ. от 1.0902				

Изм.	Лист	№ докум.	Подпись	Дата
------	------	----------	---------	------

ПГУ 1.090301.31.001 ПЗ

Лист

115

Приложение В

(Обязательное)

Приложение для расчета заработной платы сотрудников коммерческой
организации

Презентация

					ПГУ 1.090301.31.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение для расчета заработной платы сотрудников коммерческой организации

Автор работы: ст.гр.18ВВ2

Пронькин Андрей Александрович

Научный к.т.н., доцент каф. ВТ ПГУ

руководитель: Дубравин Алексей Викторович

Актуальность.

- Бухгалтерская сфера – это та сфера, где каждый день проходят десятки операций. Бухгалтерам постоянно приходится вести учет начислений зарплат, налогов, расчет больничных листов. Прибавим к этому постоянные изменения в законодательстве. Именно поэтому бухгалтера, ведущие расчет заработной платы сотрудников, заинтересованы в создании автоматизированного приложения, которое значительно упрощает их деятельность.
- Актуальность выбранной темы в том, что в настоящее время учет заработной платы не имеет смысла без автоматизации.

Обзор аналогов.

Название системы	Достоинства	Недостатки
Инфо-бухгалтер	Простота. Высокое качество Упрощенный документооборот.	Однозадачность.
Парус-бухгалтерия	Простой и разнообразный функционал. Быстрое внедрение.	Большое числа данных и взаимосвязей, что увеличивает риск возникновения ошибки.
1:С Бухгалтерия	Возможность создавать и загружать индивидуальные настройки.	Отсутствие возможности интеграции с другими сервисами. Требуется обучение пользователя.
Global Salary	Возможность массового премирования Большой выбор функционала. Эффективная работа с документами. Мощная система отчетности.	Сложность в обучении персонала.

Постановка задачи.

Учитывая достоинства и недостатки, а также особенности работы бухгалтера были поставлены следующие задачи:

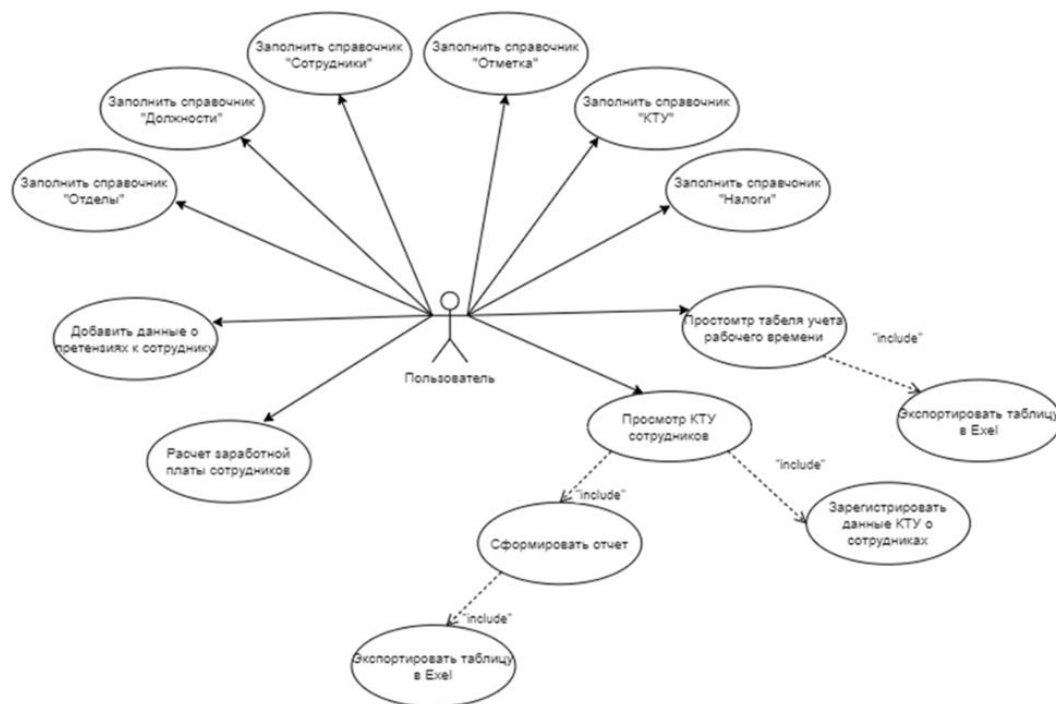
- - изучить Трудовой Кодекс РФ;
- - рассмотреть аналогичные программы, провести сравнение, продумать наиболее удобный функционал и дизайн;
- - ознакомиться с сетевыми приложениями «клиент-сервер»;
- - изучить основы разработки приложений.

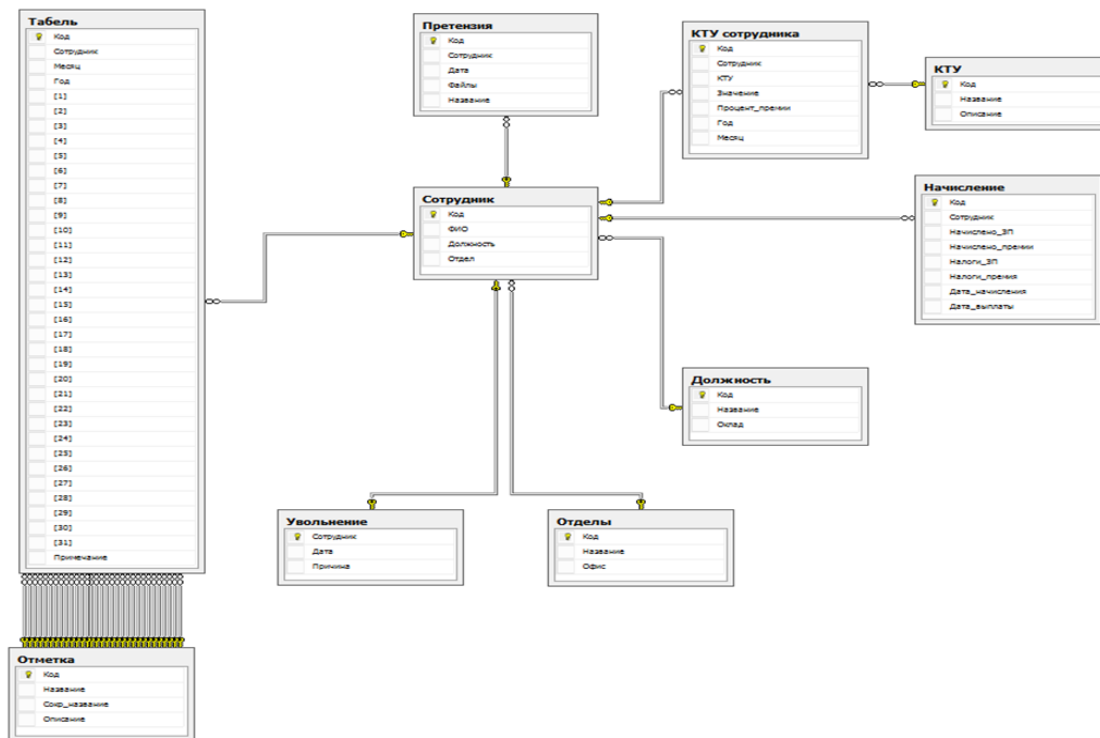
При выполнении задач должны быть учтены следующие основные требования:

- - оптимальное для пользователя представление исходной и конечной информации.
- - простота и удобство в эксплуатации программы автоматизации;

Задача дипломного проекта – разработать систему для бухгалтерии, которая позволит:

- - автоматизировать процедуру учета заработной платы;
- - вводить и хранить данные о заработной плате сотрудников.





Главное меню приложения.

Расчет заработной платы

Справочники Выход

Табель учета рабочего времени

КТУ сотрудников

Претензии сотрудников

Расчет ЗП

Выработка за месяц

Укажите месяц: Июнь

Укажите год: 2022

Сформировать

Окно «Должности»

Название	Оклад
Специалист - приемосдатчик	21750.0000
Специалист - Водитель ГПМ	24250.0000
Специалист- Охранник	15000.0000
*	

Окно «Сотрудники»

Сотрудники

для 7

ФИО	Должность	Отдел
Зайченков М.И.	Специалист - приемосдатчик	Главный
Федотов М.Г.	Специалист - приемосдатчик	Главный
Иванов А.К.	Специалист - приемосдатчик	Главный
Кручинен Р.А.	Специалист - приемосдатчик	Главный
Пигарин А.В.	Специалист - Водитель ГПМ	Главный
Рудakov П.Д.	Специалист- Охранник	Главный
*		

Уволить

Поиск

Окно «Налоги»

Налоги

для 2

Название	Процент	Применять на оклад	Применять на премию
Социальное страхование	30,2	☒	☒
Пенсионный фонд	4,8	☒	☐
*		☐	☐

Окно «Табель учета рабочего времени»

Табель

Назад

Укажите месяц:

Укажите год:

Обновить Экспорт в Excel

Сотрудник	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Примечание
Зайченко М.И.	11	11	11	11	11																											
Федотов М.Г.	11	0	11																													
Иванов А.К.					11																											
Кручинин Р.А.								11																	11							
Пигарин А.В.							11																					11				

«Выработка за месяц» формирование отчета в Excel

формат по образцу					
Буфер обмена		Шрифт			
fx Код					
	В	С	Д	Е	Ф
1	ФИО	Полная смена	Неявка	больничный	
2	Зайченков М.И.	7	2	2	
3	Федотов М.Г	4	4	1	
4	Иванов А.К.	2	3	1	
5	Кручинин Р.А.	2		1	
6	Пигарин А.В.	2	1		
7	Рудаков П.Д.				
8					
9					
10					
11					
12					

Окно «Расчет заработной платы»

Расчет ЗП

Сотрудник	Начислено ЗП	Начислено премии	Налоги ЗП	Налоги премия	Дата начисления	Дата выплаты
Федотов М.Г.	21750,0000	6117,1900	7612,5000	1847,3900	31.05.2022	02.06.2022
Зайченков М.И.	21750,0000	12234,3800	7612,5000	3694,7800	31.05.2022	03.06.2022
Иванов А.К.	21750,0000	2039,0600	7612,5000	615,8000	31.05.2022	03.04.2020
Кручинин Р.А.	21750,0000	6117,1900	7612,5000	1847,3900	31.05.2022	17.06.2022

Добавить Редактировать Удалить

Окно «Редактирование»

Расчет ЗП

Сотрудник: Федотов М.Г.

Дата начисления: 31 мая 2022 г. Дата выплаты: 2 июня 2022 г.

Перерасчет

Начислено ЗП	21750,00	Налоги ЗП	7612,5	К оплате 22677,10
Начислено премии	12234,38	Налоги премии	3694,78	
Всего начислено	33984,38	Всего налогов	11307,28	

Налоги

Название	Процент	Сумма налога ЗП	Сумма налога премии	Сумма налога
Социальное страхование	30,2	6568,5	3694,78	10263,28
Пенсионный фонд	4,8	1044	0	1044

Заключение.

В результате данной работы спроектирована, разработана и отлажено приложение для расчета заработной платы сотрудникам коммерческой организации.

Данное приложение:

- использует базу данных (хранилище данных);
- имеет дружественный и интуитивно-понятный интерфейс;
- обладает возможностью функционального расширения в связи с нуждами предприятия.