

**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное автономное образовательное учреждение  
высшего образования**

**«Южный федеральный университет»**

**Институт высоких технологий и пьезотехники**

**Кафедра прикладной информатики и инноватики**

**Хамадов Константин Константинович**

**Проектирование и разработка десктопного приложения по учету  
заработной платы**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА  
по направлению 09.03.03 — Прикладная информатика**

**Научный руководитель —  
доцент кафедры прикладной информатики и инноватики, к.т.н.  
Архипова Ольга Евгеньевна**

**Ростов-на-Дону — 2023**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ИНСТИТУТ ВЫСОКИХ ТЕХНОЛОГИЙ И ПЬЕЗОТЕХНИКИ  
Кафедра прикладной информатики и инноватики**

**З А Д А Н И Е  
на выпускную квалификационную работу**

**Студент** гр. 4-7 Хамадов К.К.

**1. Тема:** Проектирование и разработка десктопного приложения по учету заработной платы

**2. Срок сдачи законченной работы:** 06.06.2022

**3. Исходные данные:**

Государственные стандарты:

1. ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;
2. ГОСТ 7.82-2001 «Библиографическая запись. Библиографическое описание электронных ресурсов»
3. ГОСТ 19.001-77. Общие положения;
4. ГОСТ 19.781-90. Термины и определения;
5. ГОСТ 19.701-90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения;
6. ГОСТ 19.103-77. Обозначения программ и программных документов;
7. ГОСТ 19.101-77. Виды программ и программных документов;
8. ГОСТ 19.106-78. Требования к программным документам, выполненным печатным способом;

9. ГОСТ 19.104-78. Основные надписи;
10. ГОСТ 19.301-79. Программа и методика испытаний. Требования к содержанию и оформлению;
11. ГОСТ 19.402-78. Описание программы;
12. ГОСТ 19.401-78. Текст программы. Требования к содержанию и оформлению;
13. ГОСТ 19.504-79. Руководство программиста. Требования к содержанию и оформлению;
14. ГОСТ 19.505-79. Руководство оператора. Требования к содержанию и оформлению;
15. ГОСТ 19.202-78. Спецификация. Требования к содержанию и оформлению.

Основное требование — разработать сервис, выполняющий основные требуемые функции.

**4. Перечень вопросов, подлежащих разработке:**

- 1) Анализ предметной области.
- 2) Обоснование технических решений.
- 3) Проектирование элементов системы.
- 4) Реализация компонентов системы.
- 5) Разработка методики тестирования.

**5. Перечень графического материала:**

Подготовка графических материалов для презентации работы.

**6. Консультанты по работе:**

**7. Дата выдачи задания:**

**8. Руководитель** \_\_\_\_\_ **Архипова О.Е.**

*Подпись*

*ФИО*

**9. Задание принято к исполнению**

---

*Дата*

---

*Подпись студента*

## АННОТАЦИЯ

Тема выпускной квалификационной работы – разработка десктопного приложения для учёта заработной платы. Работа включает в себя введение, постановку задачи, обзор существующих решений, проектирование, реализацию, тестирование, заключение, список использованной литературы и приложения.

Во время выполнения работы были рассмотрены все необходимые темы, проанализированы существующие аналоги, составлено техническое задание, выбраны инструментальные средства для разработки, создан весь функционал системы и пользовательский интерфейс, а также проведено тестирование готового продукта.

Результатом выполнения выпускной квалификационной работы является приложение, позволяющее вести учёт заработной платы сотрудников предприятия.

## ABSTRACT

The topic of the final qualifying work is the development of a desktop application for payroll accounting. The work includes an introduction, a statement of the problem, an overview of existing solutions, design, implementation, testing, conclusion, a list of references and applications.

In the course of the work, all the necessary topics were considered, existing analogues were analyzed, technical specifications were compiled, development tools were selected, all the system functionality and user interface were created, and the finished product was tested.

The result of the completion of the final qualification work is an application that allows you to keep records of the salaries of employees of the enterprise.

## РЕФЕРАТ

Выпускная квалификационная работа 48 стр., 14 рис., 13 источников, 1 прил.

### РАЗРАБОТКА ПРИЛОЖЕНИЯ, БАЗЫ ДАННЫХ, ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON, ЗАРАБОТНАЯ ПЛАТА

Объект исследования – расчёт заработной платы сотрудников предприятия.

Предмет исследования – проектирование и разработка программы для расчёта заработной платы сотрудников предприятия.

Цель работы – разработка приложения для учёта заработной платы сотрудников небольших предприятий, работающего на операционных системах «Windows», отличительными чертами которого будут лёгкость в установке и использовании, а также простота в понимании для рядового пользователя.

В результате выполнения выпускной квалификационной работы было создано приложение для учёта заработной платы сотрудников небольших предприятий с использованием языка программирования «Python3», баз данных «SQLite». Тестирование показало полную работоспособность программы и отсутствие серьёзной нагрузки на ресурсы компьютера.

## Оглавление

<b>Введение</b> .....	9
1 Анализ проблемы и постановка задачи .....	11
1.1 Постановка цели .....	11
1.2 Декомпозиция цели .....	11
1.3 Существующие реализации .....	12
1.4 Корректировка цели .....	13
1.5 Изучение нормативно-правовой базы .....	14
2 Проектирование .....	18
2.1 Технического задание .....	18
2.2 Выбор инструментов разработки .....	19
2.3 Создание архитектуры базы данных .....	20
2.4 Создание архитектуры приложения .....	22
2.5 Выводы к главе 2 .....	25
3 Реализация .....	26
3.1 Основные этапы разработки .....	26
3.2 Информация .....	26
3.3 Установка и настройка компонентов .....	26
3.4 Реализация интерфейса .....	27
3.5 Реализация основы программы .....	30
3.6 Реализация кнопок создания, открытия и сохранения .....	31
3.7 Реализация остальных кнопок .....	33
3.8 Реализация событий календаря .....	35
3.9 Реализация выпадающих списков и кнопки расчёта .....	35
3.10 Вывод к главе 3 .....	36
4 Тестирование .....	38
4.1 Компиляция программы .....	38
4.2 Проверка файлов базы данных .....	38
4.3 Методика тестирования готового проекта .....	39
4.4 Тестирование готового приложения .....	40
4.5 Тестирование ресурсозатратности .....	43
4.6 Выводы к главе 4 .....	44

Заключение .....	45
Список использованных источников .....	46
Приложение .....	48



## ***Введение***

Программное обеспечение за всю свою историю претерпело множество изменений. Компьютеры уже давно вошли в области управления самыми разными отраслями, будь то система управления закупками, создание отчёта о количестве учеников в школах или вычисление наиболее выгодного маршрута поездки. Для принятия грамотного решения необходимо учитывать множество факторов, а скорость работы и наличие человеческого фактора двигают людей в сторону создания ещё более простых и удобных систем, позволяющих им лучше и эффективнее выполнять свою работу.

На данный момент даже малое предприятие не обходится без систем, позволяющих вести учёт сотрудников, продукции и других ресурсов. Для того, чтобы грамотно обеспечить работу предприятия, используется разнообразное программное обеспечение, в том числе и программы для бухгалтерского учёта.

Из-за существования обширного количества нормативно-правовых документов широкое распространение получили приложения для учёта зарплаты сотрудников предприятий. Такое программное обеспечение позволяет облегчить работу руководителей или бухгалтеров, избежать ошибок при вычислении, ускорить процесс работы и систематизировать полученный результат.

В первой главе выпускной квалификационной работы были изучены существующие реализации программ для учёта заработной платы и нормативно-правовые документы, регламентирующие процесс начисления заработной платы сотрудникам предприятий. Была сформирована цель разработки.

Во второй главе была построена архитектура приложения, были выбраны инструменты разработки и сформировано техническое задание.

В третьей главе были установлены необходимые для разработки инструменты и разработано приложение.

В четвёртой главе выпускной квалификационной работы было произведено тестирование разработанной программы.

## ***1 Анализ проблемы и постановка задачи***

### ***1.1 Постановка цели***

Цель выпускной квалификационной работы – создание приложения для учёта заработной платы сотрудников. Объектом исследования в рамках выпускной квалификационной работы является расчёт заработной платы сотрудников, предметом исследования – приложение для расчёта заработной платы.

### ***1.2 Декомпозиция цели***

На первом этапе необходимо обосновать целесообразность разработки подобной программы. Для того необходимо изучить уже существующие реализации, отметить их сильные и слабые стороны, выделить основные приоритеты в разработке.

Следующий шаг после корректировки цели – изучение нормативно-правовых документов Российской Федерации, регламентирующих начисление оплаты сотрудникам предприятий и организаций. Это является необходимым условием при создании архитектуры программы.

После изучения существующих реализаций и постановки чётких требований для конечной программы можно приступать к выбору инструментов для создания приложения. Следует разобрать возможные варианты реализации и выбрать наиболее подходящий вариант под поставленную перед приложением цель.

Далее можно приступать к проектированию и разработке. В первую очередь необходимо разобраться с архитектурой приложения, а затем приступить к написанию кода.

Готовое приложение необходимо протестировать. Тестирование также следует проводить на всех этапах разработки, чтобы облегчить написание программы и избежать затрат по времени в будущем на исправление ошибок.

### ***1.3 Существующие реализации***

Из всех приложений, используемых для расчёта заработной платы, наиболее востребованным на данный момент является продукт «1С:Предприятие» компании «1С» [1]. Несколько лет назад компания «1С» предоставляла пользователям приложение под названием «1С:Бухгалтерия», однако в настоящее время все элементы этого приложения вошли в состав продукта «1С:Предприятие», хотя старое название до сих пор используется в неформальных диалогах. Программа предоставляет как объекты для работы (отчёты, документы, регистры), так и инструменты для их использования и работает при помощи «конфигураций», настраиваемых для каждого предприятия отдельно.

Плюсы «1С:Предприятие»:

- гибкость, возможность легко подстроить приложение под своё предприятие;
- большие вычислительные мощности.

Минусы «1С:Предприятие»:

- необходимость обучения персонала и наличие технической поддержки. Далеко не каждый рядовой пользователь или бухгалтер сможет самостоятельно разобраться с работой в программе;
- программа является платной.

Основным конкурентом для компании «1С» является, как ни странно, продукт компании Microsoft «Microsoft Excel». Изначально эта программа не создана для подобных задач и является приложением для работы с таблицами, однако для небольших предприятий с малым количеством сотрудников существующего функционала часто бывает достаточно, чтобы вести учёт для каждого сотрудника отдельно, а встроенные функции позволяют проводить необходимые расчёты со значениями из заполненных полей таблиц.

Плюсы «Microsoft Excel»:

- простота в использовании. Примерно каждый пользователь, когда-либо работавший с компьютером, знаком с «Microsoft Excel»;

- Легко сочетается с другими продуктами компании Microsoft, с которыми тоже зачастую приходится работать.

Минусы «Microsoft Excel»:

- ограниченность. При увеличении количества пользователей будет расти количество листов, в результате чего работать станет проблематично, а небольшие изменения в системе оплаты труда принесут множество хлопот;
- для использования продуктов «Microsoft Excel» в коммерческих целях придётся заплатить.

Существуют альтернативные бесплатные приложения по учёту заработной платы сотрудников небольших предприятий. «ARUS Зарботная плата и Кадры» – приложение для учёта заработной платы сотрудников небольших предприятий. Имеет бесплатную версию, ограниченную в функциональности и способную работать только с тремя сотрудниками.

Плюсы бесплатной версии «ARUS Зарботная плата и Кадры»:

- бесплатная программа;
- широкий функционал с возможностями создания отчётов при помощи приложения «Microsoft Excel».

Минусы бесплатной версии «1С:Предприятие»:

- ограниченное число сотрудников;
- проблемы с установкой под операционными системами «Windows».

После изучения существующих реализаций можно точнее сформулировать цель.

#### ***1.4 Корректировка цели***

Цель разработки переопределим следующим образом. Цель выпускной квалификационной работы – разработка приложения для учёта заработной платы под операционной системой Windows для малых предприятий, основными преимуществами которого будут:

- простота в понимании и изучении для обычного пользователя;
- легкость в использовании;
- возможность подстроить программу под своё предприятие.

### ***1.5 Изучение нормативно-правовой базы***

Учёт заработной платы в условиях современной России включает в себя несколько этапов и подробно описан в нормативно-правовых документах, а именно в Трудовом кодексе Российской Федерации [2] и Налоговом кодексе Российской Федерации [3]. Существует несколько типов заработной платы, регламентированных в упомянутых документах.

Расчёт заработной платы – процесс начисления оплаты нанятым сотрудникам согласно условиям трудового договора и удержания налогов и взносов с учётом вычетов, предусмотренных законодательством Российской Федерации, а также документальное оформление этих операций.

Если рассматривать процесс расчёта заработной платы с точки зрения работодателя, то этот процесс имеет следующие составляющие:

- Регистрация фактов приема, перевода и увольнения работников.;
- Расчет размера оплаты труда сотрудников;
- Расчет и удержание налога на доходы физических лиц (НДФЛ);
- Расчет и начисление страховых взносов;
- Оформление документов на выплату заработной платы;
- Перечисление рассчитанных налогов и взносов в бюджет;
- Выплата ЗП работникам.

При расчёте оплаты необходимо согласовываться со статьёй 133 Трудового Кодекса Российской Федерации.

Налог на доходы физических лиц рассчитывается из начисленной зарплаты, удерживается из неё и перечисляется в бюджет самим работодателем. Размер НДФЛ составляет 13% (Страховые взносы рассчитываются из той же начисленной зарплаты и перечисляются поверх этой суммы без удержания их у работника). Налог НДФЛ можно уменьшить при помощи стандартных налоговых вычетов (льготы, применяемой при расчёте заработной платы).

Страховые взносы уплачиваются работодателем самостоятельно в размерах, равных 22% для пенсионного страхования, 5,1% для медицинского

и 2,9% для ВНиМ. В разрабатываемой программе нет причин учитывать страховые взносы.

Согласно статье 135 Трудового кодекса Российской Федерации заработная плата устанавливается согласно с существующими у работодателя системами оплаты труда. Разберём основные системы оплаты труда, а именно повременную и сдельную.

В случае повременной системы оплаты труда заработок сотрудника напрямую зависит от отработанного времени, то есть существует фиксированная цена за единицу времени. Единицей времени могут быть месяц, час, день или отработанная смена. Если сотрудник работает по стандартному 40-часовому графику, то основным видом его начислений является оклад, сумма, которая платится сотруднику за отработанный месяц. Если сотрудник отработал месяц не полностью, то расчёты выполняются пропорционально отработанному времени. Для расчёта оклада за месяц берётся сумма оклада за месяц, делится на норму рабочего времени за этот месяц и умножается на фактически отработанное время.

Например, если в месяце 18 рабочих дней, а сотрудник с окладом, равным 30 000 рублей работал всего 16 из них, то его заработная плата будет на две восемнадцатые меньше, то есть 26 666.67 рублей. Такую форму расчёта заработной платы удобное использовать при стандартном рабочем графике.

Расчёт по окладу можно использовать и для сотрудников, работающих по сменному графику, однако в таком случае удобнее будет использовать тарифную ставку, установленную за смену или час. Например, работники фирмы по продаже книг имеют два выходных после двух рабочих дней и сменяют друг друга на время выходных. В результате, за месяц один сотрудник отработал 16 смен, а другой 14. Для тарифной оплаты за смену можно легко рассчитать заработную плату сотрудников как тарифную ставку, умноженную на количество смен. Неудобство использования системы расчёта заработной платы по окладу для данной ситуации состоит в том, чтобы рассчитать месячную норму по времени. В законодательстве не описаны

подобные ситуации. Рассчитать временную норму для каждого конкретного сотрудника можно по нормативному календарю пятидневной сорокачасовой рабочей недели, по графику сотрудника, а так же и другими способами. Механизм расчёта при этом не изменится.

Сдельная оплата труда не зависит от отработанного времени. Для расчёта заработной платы при использовании этой системы оплаты труда используется количество произведённой продукции или оказанных услуг. Например, дневная норма обслуженных клиентов за день у массажиста составляет 5 человек, а дневная ставка равна 2500 рублей, что соответствует 500 рублям, полученным в качестве заработной платы за обслуживание одного клиента. За месяц массажист обслужил 100 человек и его месячный заработок составил 50 000 рублей.

Рассмотрим эту же ситуации с немного другими условиями оплаты труда. Теперь вместо системы оплаты труда, зависящей от количества обслуженных клиентов напрямую, введём сдельную оплату по норме времени. Установим массажисту норму времени на обслуживание одного клиента как 120 минут или 2 часа. Обозначим часовую тарифную ставку как 200 рублей за час времени. За месяц массажист обслужил 100 клиентов, что соответствует 200 часам работы. Ему заработная плата за месяц будет равняться количеству часов работы, умноженному на тарифную ставку, то есть 40 000 рублям.

Заработная плата сотрудника при любой системе оплаты труда делится на две части, первая из которых называется авансом. Для системы расчёта зарплаты, использующей оклад сотрудника и норму отработанного времени за период работы, рекомендуется либо использовать 50% от суммы оклада, либо рассчитывать аванс из отработанного за первую часть периода времени. Например, сотрудник имеет оклад в 25 000 рублей, а в месяце имеется 18 рабочих дней. В первую половину рабочего месяца этот сотрудник отработал 8 дней из 18-ти, из-за чего размер его оклада будет равняться окладу в 25 000 рублей, поделённому на 18 рабочих дней в месяце и умноженному на 8 уже отработанных дней, или 11 111.11 рублей. Для сотрудников, работающих по



сдельной системе оплаты труда, следует рассчитывать аванс как 50% от заработной платы по итогам предыдущего месяца работы.

## ***2 Проектирование***

### ***2.1 Технического задание***

Разрабатывая техническое задание необходимо ссылаться на поставленные цели и задачи. Главными приоритетами в разработке должны оставаться гибкость, простота в использовании и простота в понимании рядовым пользователем. Должна быть интуитивно понятной, не должна требовать особых навыков работы с компьютером, а также содержать возможности кастомизации, выбора типа зарплаты. В результате было сформировано следующее техническое задание.

Создать программу, которая:

- способна создавать и редактировать файлы с информацией о сотрудниках предприятия;
- способна создавать и редактировать файлы с информацией о количестве труда, произведённого работниками предприятия;
- способна выводить информацию о количестве произведённой работы сотрудником за определённый период;
- способна производить вычисления заработной платы за определённый период, исходя из информации в файлах с данными о сотрудниках и проделанной ими работы;
- способна выводить результат этих вычислений в текстовый файл и в интерфейс пользователя;
- имеет интуитивно понятный и внешне приятный интерфейс;
- имеет интерфейс, позволяющий пользователю взаимодействовать с файлами, содержащими информацию о сотрудниках предприятия и произведённой ими работе в определённые промежутки времени, то есть добавлять и редактировать эту информацию;
- имеет интерфейс, позволяющий пользователю запросить у программы вычисление заработной платы для определённого сотрудника в определённый период;
- не выдаёт непредвиденных ошибок и исключений во время работы;

– корректно отображает пользователю посредством интерфейса информацию о сотрудниках и проделанной ими работе за определённый период.

## ***2.2 Выбор инструментов разработки***

Для разработки приложений на платформе Windows существует значительное количество инструментов и языков программирования. Для грамотного выбора инструментов следует обратиться к техническому заданию.

В первую очередь заметим, что в ходе разработке придётся написать множество строк программного кода. По этой причине следует тестировать приложение ещё на этапах разработки, чтобы избежать сложностей с редактированием кода в будущем. По этой же причине стоит использовать только высокоуровневые языки программирования, чтобы уменьшить время разработки.

Выбор языка программирования произведём из следующего набора:

– Язык «C++» [4].

Объектно-ориентированный статистически типизированный язык программирования. Имеет богатую стандартную библиотеку и часто используется в разработке программного обеспечения, содержит свойства как высокоуровневых, так и низкоуровневых языков программирования. Рассматривается в рамках выпускной квалификационной работы из-за обширной библиотеки, высокой скорости и успешного опыта его внедрения в разработку приложений.

– Язык «Python» [5].

Высокоуровневый язык программирования с динамической строгой типизацией. Является мультипарадигменным языком программирования. Имеет множество сторонних библиотек, позволяющих подстроить его под практически любые задачи. Рассматривается как вариант для языка программы как раз из-за его гибкости и возможности легко адаптироваться под разные задачи, а также динамической типизации, позволяющей облегчить

преобразования при разработке системы, работающей с разными типами данных.

После изучения возможностей обоих языков программирования в области разработки оконного приложения для рабочего стола, был выбран язык программирования «Python», а именно – его версия «Python3». Причиной этого решения стало то, что для этого языка существующий пакет «tkinter» позволяет справиться практически со всеми задачами в техническом задании, а возможности языка дадут возможность выполнить оставшиеся.

Для создания файлов о работниках и проделанной ими работе было принято решение использовать базу данных. Однако данное решение резко конфликтует с целью разработки, а именно простотой в понимании и управлении приложением. Изначально рассмотренные базы данных «MySQL» [6] и «PostgreSQL» [7] требовали от пользователя установки сервера и подключения к нему при помощи программного приложения. Для человека, слабо разбирающегося в базах данных подобные задачи могут стать слишком тяжёлыми. Из-за этого от баз данных, для использования которых необходимо устанавливать сервер, было принято решение отказаться. Решением проблемы, позволяющим использовать базу данных и не усложнять установку и использование приложения, стал пакет «SQLite3» [8] языка «Python3». Данный пакет поставляется вместе со средой разработки по умолчанию и позволяет создавать таблицы однофайловой базы данных «.bd», а также работать с ними при помощи объектов этого же пакета и строк кода языка запросов «SQL» [9].

### ***2.3 Создание архитектуры базы данных***

База данных будет содержать в себе все необходимые данные для приложения. К таким данным относятся:

- имя сотрудника;
- тип зарплаты сотрудника;
- ставка (для повременной оплаты труда);
- количество часов в ставке (для повременной оплаты труда);

- стоимость продукта или услуги (для сдельной оплаты труда);
- количество часов работы для каждого из дней (для повременной оплаты труда);
- количество продуктов или услуг (для сдельной оплаты труда).

Каждому сотруднику можно будет установить один из двух вариантов оплаты труда.

Для базы данных нет смысла делать более двух связанных таблиц, а именно дней и сотрудников. Назовём их «USERS» и «DAYS». Каждая запись из таблицы «USERS», то есть сотрудник, может иметь несколько связанных с ней записей в таблице «DAYS» с днями, но каждая запись из таблицы «DAYS» может относиться только к одному конкретному сотруднику из таблицы «USERS». Такую связь между таблицами называют «один ко многим».

Определим поля будущих таблиц. Таблица «USERS» будет содержать в себе поля:

- user\_id, ключевое поле таблицы, номер сотрудника в базе данных;
- user\_name, имя сотрудника, не может являться ключевым полем из-за возможности одинаковых имён для нескольких сотрудников;
- user\_type, тип зарплаты для сотрудника;
- user\_stake, размер оклада для сотрудника (для повременной оплаты труда);
- user\_hours\_stake, количество часов работы для оклада (для повременной оплаты труда);
- user\_item\_cost, стоимость одного продукта или одной услуги (для сдельной оплаты труда);

Таблица «DAYS» будет содержать в себе следующие поля:

- day\_id, ключевое поле;
- day\_day, номер дня в месяце;
- day\_month, номер месяца в году;
- day\_year, год;
- day\_hours, количество отработанных часов в дне (для повременной

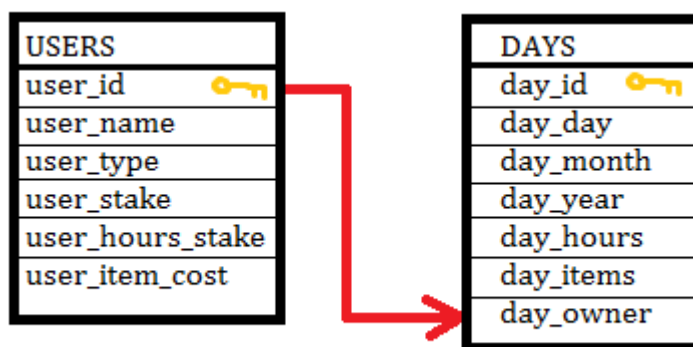
оплаты труда);

– day\_items, количество произведённых продуктов или услуг в дне (для сдельной оплаты труда);

– day\_owner, номер сотрудника, к которому относится информация об этом дне.

Столбец «day\_owner» таблицы «DAYS» будет связан со столбцом «user\_id» таблицы «USERS», что и создаст связь «один ко многим».

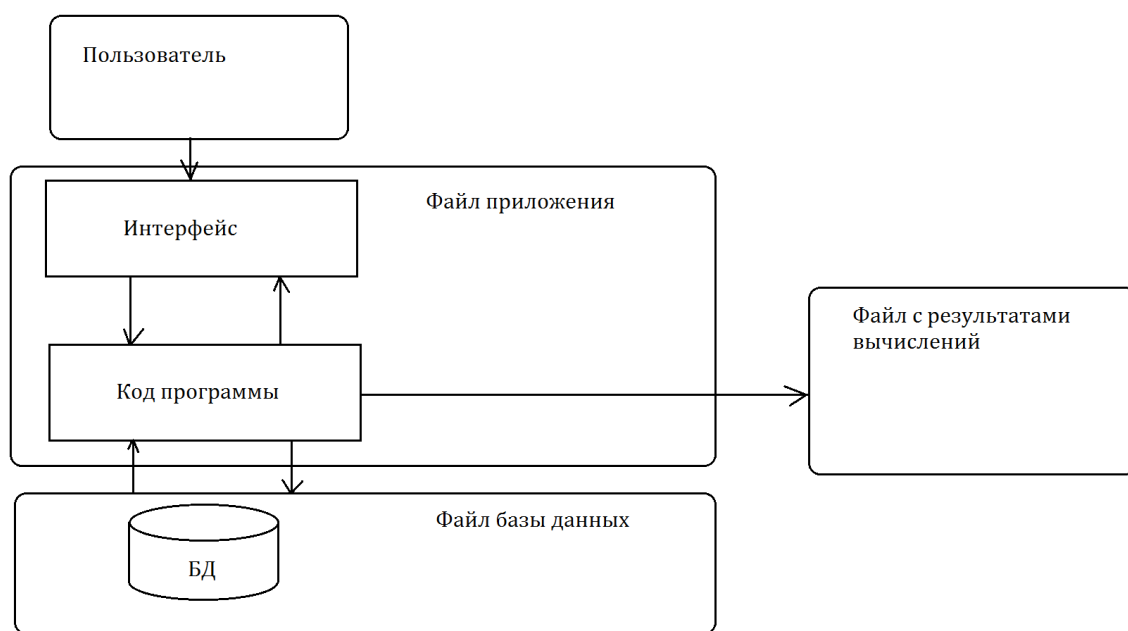
Графически база данных представлена на рисунке 1.



**Рисунок 1 – структура базы данных**

## ***2.4 Создание архитектуры приложения***

Пользователь будет работать с базой данных и создавать текстовый файл с результатом расчётов при помощи приложения. Разделим приложение на две части. Первая часть, интерфейс, будет отвечать за взаимодействие с пользователем. Интерфейс будет отлавливать события, создаваемые пользователем и реагировать на них, отправляя команду во вторую часть приложения, программный код. Вторая часть приложения будет получать команду через интерфейс и, в зависимости от команды, решать, что делать дальше: выдать сообщение об ошибке, обратиться с запросом к базе данных или создать текстовый файл и передать в интерфейс новые данные. Архитектура работы программы представлена на рисунке 2.



**Рисунок 2 – архитектура программы**

Сразу определим возможные системы расчёта заработной платы сотрудников. В приложении будет реализовано 2 способа подсчёта заработной платы. Первый способ учитывает оклад сотрудника, количество рабочих часов в неделю, количество рабочих дней в месяце и фактически отработанное сотрудником рабочее время за период. Эту окладную повременную систему расчёта заработной платы назовём в приложении «Повременная». Вторым способом станет сдельная система расчёта оплаты труда. В этой системе будет учитываться сумма, получаемая работником за производство одного продукта или оказание одной услуги, а также количество продукции, произведённой сотрудником за период работы (или количество оказанных услуг). Назовём в приложении такую систему «Сдельная». Для способа расчёта заработной платы «Повременная» аванс будет рассчитываться как половина итоговой заработной платы сотрудника за период, а для системы «Сдельная» аванс будет браться как сумма, равная половине произведённой сотрудником продукции (или половине от количества оказанных услуг) умноженной на текущую стоимость оплаты труда сотрудника за производство единицы

продукции (или оказание одной услуги).

Разберём подробнее необходимые элементы интерфейса. В первую очередь следует дать пользователю возможность создать новый файл базы данных или открыть уже существующий, а также сохранить базу данных под другим именем. Во вторую очередь необходимо создать интерфейс пользователя, выводящий ему получить информацию из файла базы данных. Далее нужно позволить пользователю редактировать эти данные и добавлять новые. После чего необходимо дать пользователю возможность производить расчёты из имеющихся в базе данных строк, а также выводить информацию в файл или интерфейс.

Для работы с данными из таблицы «USERS» следует использовать обычные поля с текстом для значений ставки, часов ставки и стоимости продукта (услуги), а также всплывающие списки с выбором сотрудника из базы данных и данных о типе зарплаты для него. Для работы с днями было принято решение использовать пакет «tkcalendar» языка программирования «Python3». При помощи этого пакета можно создать календарь и отлавливать события смены дня и месяца для этого календаря. Помимо календаря следует создать поля для ввода значений строк таблицы базы данных для выбранного дня, а именно для количества часов работы за выбранный день и для количества созданного продукта (оказанных услуг).

Для создания ввода данных нового сотрудника следует добавить в приложение кнопку «Добавить сотрудника», в текстовое поле вводятся данные – фамилия, имя сотрудника, для возможности сменить данные полного имени сотрудника необходимо выбрать сотрудника из списка и отредактировать данные. Для того, чтобы подтвердить информацию о вводе данных в таблицу можно создать две кнопки, одну для сотрудника, другую для дня (если записи с днём не существует, создастся новая). При нажатии этих кнопок будет производиться проверка текста в полях, и либо будет выводиться сообщение об ошибке в интерфейс пользователя, либо будет вноситься информация в базу данных.



Для получения расчёта заработной платы придётся создать дополнительные текстовые поля для вывода информации, одно редактируемое текстовое поле для выбора дня начала периода и кнопку, при нажатии на которую будут производиться вычисления и записываться результат в текстовый файл и интерфейс программы.

### ***2.5 Выводы к главе 2***

Во время подготовки программы было создано техническое задание, были изучены возможности языков программирования «C++» и «Python» в разработке десктопных приложений, были изучены возможности базы данных «SQLite», были разработаны архитектура программы и базы данных.

### ***3 Реализация***

#### ***3.1 Основные этапы разработки***

Написание приложения разделим на несколько ключевых этапов:

- выбор компьютера для работы и его характеристик;
- установка необходимых компонентов;
- реализация компонентов программы.

В разработку компонентов программы входят:

- реализация интерфейса;
- реализация базы данных;
- реализация кода программы.

#### ***3.2 Информация***

Для разработки использовался ноутбук «DELL Inspiron 17r 5737» 2014 года выпуска со следующими характеристиками:

- Операционная система «Windows8.1»;
- Процессор «Intel® Core™ i7-4500 CPU @ 1.80GHz 2.40G»;
- Оперативная память 8 Гб;
- Тип системы 64-разрядная;
- Встроенный в процессор графический чип «HD Graphics»;
- Дискретная видеокарта «AMD Radeon HD 8870M».

Графический чип и видеокарта в разработке не использовались.

#### ***3.3 Установка и настройка компонентов***

Работа с кодом языка программирования «Python3» производилась при помощи среды разработки «IDLE Python3.10 64-bit»

Вместе со средой разработки в автоматическом режиме был установлен компонент «pip», в дальнейшем использовавшийся для установки прочих компонентов. Также были по умолчанию загружены многие другие необходимые в дальнейшей разработке пакеты, среди них «os», «tkinter», «time» и некоторые другие.

«tkinter» – пакет языка программирования «Python3», позволяющий создавать диалоговые окна, окно приложения и объекты интерфейса, а также

предоставляющий инструменты для работы с ними.

«os» – пакет языка программирования «Python3», предоставляющий дополнительные возможности для работы с файлами.

«time» – пакет языка программирования «Python3», дающий возможность работать со временем.

Из неустановленных пакетов необходимо загрузить пакет «tkcalendar», использующийся для создания календаря

Чтобы установить этот пакет используем «pip»:

- откроем командную строку;
- введём команду «pip install tkcalendar»;
- дождёмся завершения работы программы.

Если командная строка выдаёт сообщение о том, что «pip» не является командой, значит его необходимо добавить в «PATH».

Результат работы «pip» представлен на рисунке 3.

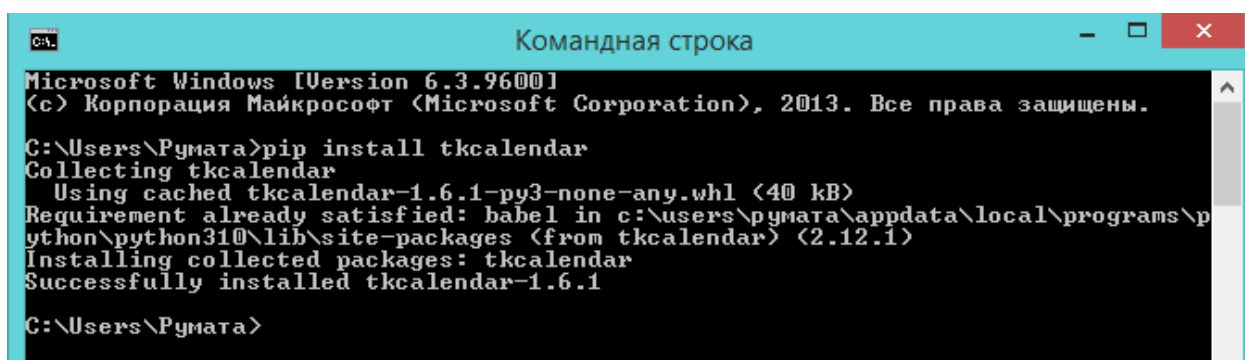


Рисунок 3 – результат работы «pip»

### 3.4 Реализация интерфейса

Начнём написание приложения. Импортируем необходимые библиотеки следующим образом:

- «import tkinter»;
- «import tkinter.filedialog as fd»;
- «from tkinter import ttk»;
- «from tkinter import messagebox»;

- «from tkcalendar import Calendar»;
- «import os.path»;
- «import shutil», библиотека, позволяющая легко создать копию файла;
- «import sqlite3».

Создание интерфейса программы производится при помощи библиотеки «tkinter».

Сперва создаём окно приложения при помощи команды «tkinter.Tk()» и присвоим полученный объект окна приложения переменной «tk». Задаём этому объекту параметры «geometry», «title» и «resizable». В результате получаем окно приложения с названием и фиксированным размером.

Командой «os.path.exists("pick.ico")» проверим наличие файла с картинкой в директории приложения. В случае успешной проверки пропишем команду «tk.iconbitmap(default="pick.ico")». Теперь, если положить рядом с файлом программы изображение «pick.ico», это изображение станет иконкой приложения.

В конце всего кода необходимо прописать команду «tk.mainloop()». Эта команда зацикливает главное окно приложения, благодаря чему программы, созданные при помощи пакета «tkinter» не закрываются сразу после открытия. Эту команду до конца разработки будем оставлять в конце кода приложения.

Приступаем к созданию элементов. В ходе разработки понадобятся следующие элементы интерфейса:

- «tkinter.Label», заголовок;
- «tkinter.Entry», текстовое поле;
- «tkinter.Button», кнопка;
- «ttk.Combobox», выпадающий список;
- «Calendar», календарь.

Поочерёдно создаём все необходимые элементы на главном окне. После создание каждого элемента будем прикреплять его к главному окну командой «object.place». Для создания заголовков используем команду «tkinter.Label()», для создания кнопок вызовем команду «tkinter.Button()», для текстового окна

– «tkinter.Entry». Чтобы создать выпадающие списки используем команду «ttk.Combobox()», а также вызовем создание календаря функцией «Calendar()».

Прикрепляем к выпадающим спискам события выбора нового элемента при помощи команды «object.bind()». Теперь интерфейс готов. На нём присутствуют все необходимые элементы, а также прикреплены все события взаимодействия пользователя с элементами, а именно нажатие на кнопки и выбор новых элементов в выпадающих списках.

После размещения элементов интерфейса внешний вид приложения приобрёл вид рисунка 4.

БыстрыйСчёт C:/Users/Рымата/Desktop/dist/Новая папка/newdb....

Открыть Новый файл Новый сотрудник Имя Roman Сохранить Сохранить сотрудника

Оклад 20000 Тип зарплаты Повременн

Часов в рабочей неделе 40

Рабочих дней за период 12

Часов работы за период

Начисленно Вычтено: НДФЛ Итого за месяц

З.п. 1 (аванс) З.п. 2 Сохранить данные и рассчитать

January 2023

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
52	26	27	28	29	30	31	1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29
5	30	31	1	2	3	4	5

Часов работы за день 3 Записать

Рисунок 4 – внешний вид приложения

### ***3.5 Реализация основы программы***

Прежде, чем двигаться дальше, следует разобраться с переменными в языке программирования «Python3». Во-первых, в этом языке программирования отсутствуют константы, а также ссылочный тип данных в привычном понимании. Во-вторых, в отличие от большинства языков программирования, в «Python3» нет деления на функции и процедуры. Из-за динамической типизации просто отсутствует целесообразность такого деления. Любая функция в «Python3» может вернуть любой тип данных и объявляется как «def name(atr\*):». Ввиду отсутствия ссылочного типа данных, нельзя передать в функцию ссылку на переменную, а значит, функция не может изменить значение переданного ей атрибута. Это приводит к тому, что для изменения вспомогательных переменных (например, текущие день и месяц) внутри функции, определяющей, какой сегодня день, можно использовать только 2 подхода.

Первый – передавать функции необходимые данные в виде параметров функции, а возвращаемое значение установить как список («list»), содержащий в том числе новые значения, которые необходимо присвоить атрибутам.

Второй подход – использование глобальных переменных. Когда функция в «Python» не находит нужную переменную в своей локальной области видимости, она не начинает искать эту переменную в глобальной области, а просто создаёт новую локальную переменную. Из-за этого необходимо в начале функции указывать, что переменную следует искать в глобальной области. Для этого используется команда «global name».

Второй способ реализации выглядит гораздо привычнее для языков программирования высокого уровня, и этот способ был использован в выпускной квалификационной работе.

Также стоит упомянуть, что расположение функций в коде «Python3» не влияет на их функциональность: любая функция может быть вызвана из любой точки кода.

Опишем необходимые программе глобальные переменные в начале всего кода приложения. Эти переменные:

- «numbers», список с текстовыми элементами. В коде используется для функции «is\_number()», проверяющей, можно ли преобразовать строку в число с плавающей точкой;
- «user\_id\_for\_combobox\_number», список с целочисленными элементами, используется для связи индексов элементов выпадающего списка имён сотрудников с реальными номерами сотрудников в базе данных;
- «current\_filepath», путь к текущей базе данных;
- «current\_day», текущий выбранный день в календаре;
- «current\_month», текущий выбранный месяц в календаре;
- «current\_year», текущий выбранный год в календаре;
- «current\_user\_id», номер текущего выбранного сотрудника в базе данных;
- «con», объект типа «соединение» для базы данных;
- «cur», объект типа «курсор» для базы данных;
- «max\_user\_id», максимальный номер сотрудника в базе данных;
- «max\_day\_id», максимальный номер дня в базе данных.

### ***3.6 Реализация кнопок создания, открытия и сохранения базы данных***

Начнём прикреплять код к событиям. Для нажатия кнопки «Новый файл» прописываем создание нового файла базы данных. Сперва используем команду «fd.asksaveasfilename», вызывающую окно сохранения файла. Если пользователь создал файл, вызываем команду «create\_new\_database». Эта команда создаёт новый файл базы данных «.bd», объявляет в нём таблицы «USERS» и «DAYS», а также добавляет одного сотрудника с пустыми полями данных в таблицу «USERS». Информацию о сотруднике можно изменить как после создания нового сотрудника, так и после выбора другого сотрудника при помощи выпадающего списка, который будет создан в дальнейшем.

Для события нажатия кнопки «Открыть файл» прописываем функцию

открытия базы данных. Сперва вызываем диалоговое окно открытия файла. Если пользователь выбрал файл, открываем файл базы данных.

Обеим кнопкам добавляем действие по обновлению списка сотрудников. Для этого объявим функцию «set\_name\_values», которая будет написана на следующем этапе разработки. Выбираем первого сотрудника в базе данных и выводим информацию о нём в интерфейс.

Команда «create\_new\_database» создаёт новый объект файла базы данных командой «sqlite3.connect()», новый объект типа курсор командой «object.cursor()», а также при помощи кода языка запросов «SQL» создаёт базу данных по умолчанию. Запросы к базе данных выглядят следующим образом:

- «string = “SQL code”», строка запроса;
- «cur.execute(string)», обращение к базе данных;
- «con.commit()», сохранение базы данных;

При использовании выборки из базы данных также используется команда «cur.fetchall()», позволяющая получать результат выборки и записать его в переменную в виде списка.

В событии кнопки сохранения вызовем функцию сохранения файла. Функция сперва отключается от базы данных, после чего создаёт копию файла командой «shutil.copyfile» и подключается к этой копии.

На протяжении написания кода создаём сообщения для консоли, чтобы проверять, что программа работает корректно. Функция сохранения новой базы данных представлена на рисунке 5.



```

def command_save_sql_btn():
    global current_filepath
    global con
    global cur
    filetypes = (("База данных SQLite", "*.db"), ("Любой", "*"))
    filename = fd.asksaveasfilename(title="Сохранить файл",
                                     defaultextension="*.db", filetypes=filetypes)
    if filename:
        con.close()
        shutil.copyfile(current_filepath, filename)
        current_filepath=filename
        con = sqlite3.connect(current_filepath)
        sqlite3.enable_callback_tracebacks(True)
        tk.title("БыстрыйСчёт "+str(current_filepath))
        cur = con.cursor()
        print("saved")
        print(current_filepath)

```

**Рисунок 5 – код кнопки сохранения**

### ***3.7 Реализация кнопок записи в базе данных***

Для события нажатия на кнопку «Создать сотрудника» создаём новую функцию. После запроса на создание нового сотрудника в базе данных, указываем в ней вызов уже созданной функции «set\_name\_values». В ней создаём один необязательный параметр, по умолчанию разный «False». Функция будет обновлять список сотрудников. Дополнительный параметр будет указывать на то, что после обновления необходимо переключиться на последнего сотрудника, для нашего вызова этой функции передадим его как «True». Команда для выбора сотрудника приведёт к срабатыванию события по обновлению данных для нового сотрудника.

Кнопка «Записать» обновляет или добавляет информацию для календаря выбранного сотрудника. Для события нажатия на кнопку создаём функцию, сначала проверяющую наличие записи с подобной датой для текущего работника. В зависимости от результата проверки обновляем или добавляем запись с новыми данными о работе, произведённой сотрудником в текущий день.

Для события нажатия на кнопку «Сохранить сотрудника» создаём событие, проверяющее значение всех полей интерфейса связанных с сотрудником. В случае, если программа не нашла ошибок в тексте, обновляем значения строки текущего сотрудника в базе данных. Для проверки

используем функцию «is\_number», сравнивающую каждый символ строки с точкой и цифрами по определённому правилу. Для удобства отладки и тестирования выводим в консоль сообщения о результатах работы функции. Код функции для проверки соответствия строки неотрицательному числу представлен на рисунке 6.

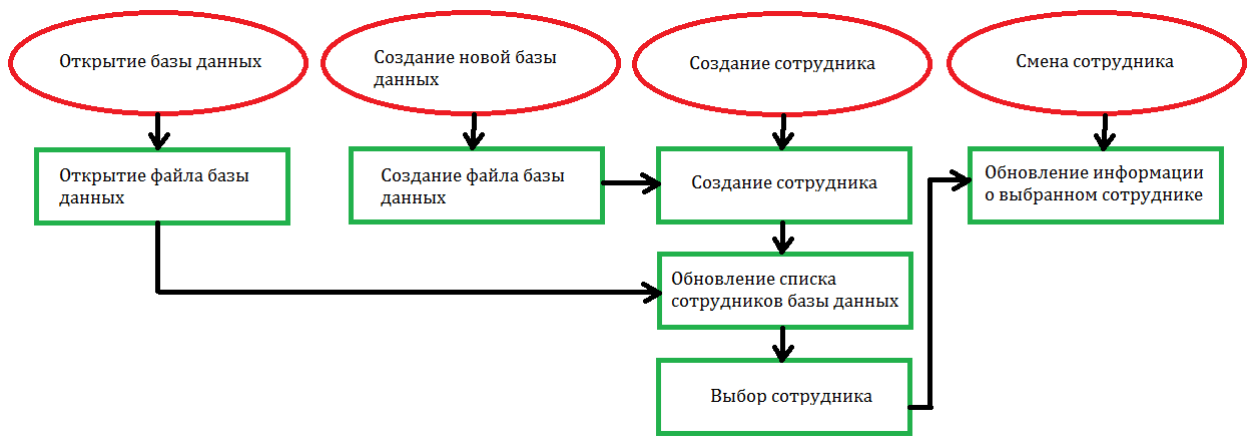
```
def is_number(value):
    num=str(value)
    print("num="+num)
    print("type="+str(type(num)))
    if num=="":
        return True
    else:
        if (num[0] in numbers) and (num[-1] in numbers):
            boolcheck=False
            for cell in num[1:-1]:
                if cell=='.':
                    if boolcheck:
                        print("exit2")
                        return False
                    else:
                        boolcheck=True
            elif not (cell in numbers):
                print("exit3")
                return False

            return True
        else:
            print("exit1")
            return False
```

**Рисунок 6 – функция «is\_number»**

Событие выбора другого сотрудника при помощи выпадающего списка соединим с действием по обновлению информации о выбранном сотруднике в интерфейсе программы.

Схема связи действий для событий открытия и создания базы данных, а также создания и выбора нового сотрудника представлена на рисунке 7.



**Рисунок 7 – схема связи между событиями и действиями**

### ***3.8 Реализация событий календаря***

Для события изменения месяца в календаре создаём функцию, которая очищает переменные, отвечающие за текущую дату в календаре.

Для события выбора новой даты напишем функцию, сравнивающую новую дату с предыдущей и, если те отличаются, обновляющую вспомогательные переменные и обновляющую данные в интерфейсе на результат запроса к базе данных по текущему работнику и текущей дате. Если же результат запроса не содержит запись, то в поле с количеством проделанной работы за текущий день пишем пустую строку.

### ***3.9 Реализация выпадающих списков и кнопки расчёта***

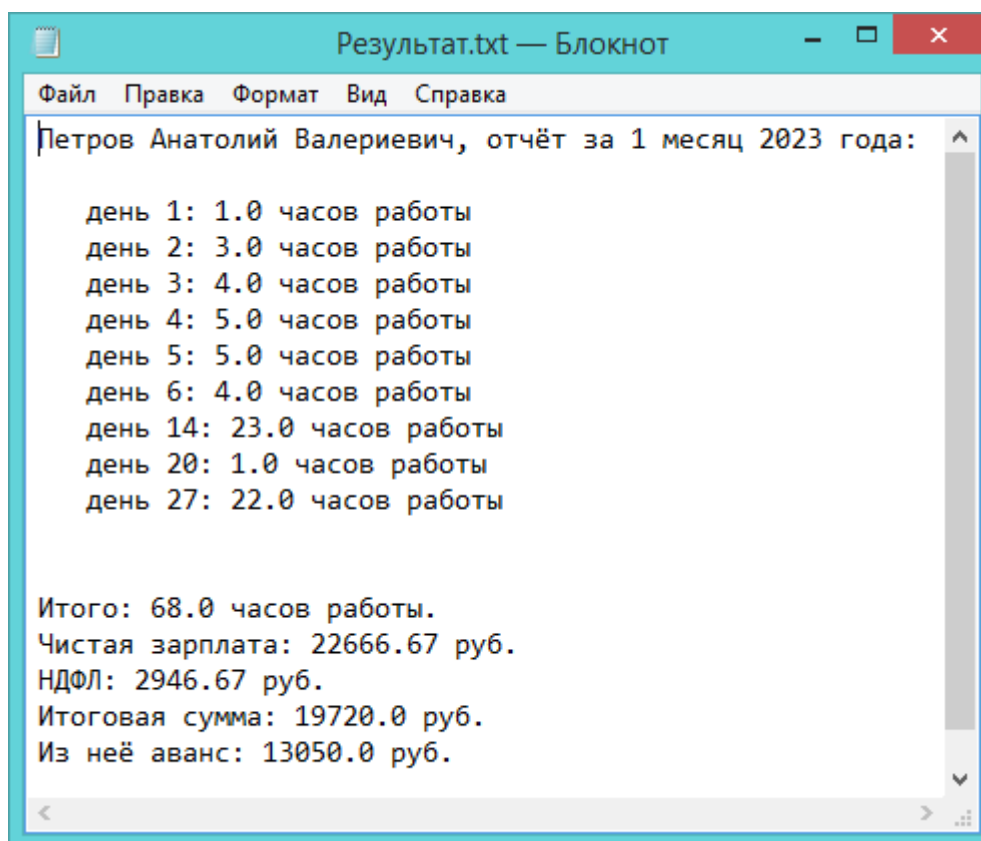
Для события «выбор нового сотрудника» из выпадающего списка работников создаём функцию, обновляющую интерфейс и вспомогательные переменные. Из переменной «user\_id\_for\_combobox\_number», получаем настоящий номер пользователя в зависимости от индекса нового выбранного сотрудника в списке. При помощи запроса к базе данных вводим новые значение в интерфейс пользователя.

Для выпадающего списка типа оплаты обновляем данные в базе данных для текущего работника и поля интерфейса пользователя.

Последней создаём функцию расчёта заработной платы. Эта функция должна заполнять поля интерфейса и создавать текстовый файл. Сначала задаём в ней переменную, отвечающую за минимальную заработную плату,

после чего обращаемся к обеим таблицам поочерёдно. Из таблицы «USERS» получаем значения полей для стоимости одного продукта или одной услуги, оклад и часы оклада для текущего сотрудника. Из таблицы «DAYS» получаем список дней работы в необходимом промежутке времени. Из этого списка получаем суммарное количество проделанной работы за период, будь то часы работы или количество продуктов (услуг).

Исходя из полученных данных и типа заработной платы текущего сотрудника высчитываем результат в виде НДФЛ, итоговой зарплаты за месяц, оклада и прочих значений. Выводим полученный результат в интерфейс и предлагаем пользователю создать текстовый файл для записи полученных значений. В этот файл записываем всю информацию из интерфейса, а также всю информацию по дням работы. Пример результата работы функции представлен на рисунке 8.



**Рисунок 8 – текстовый файл**

### **3.10 Вывод к главе 3**

В третьей главе выпускной квалификационной работы было получено

работоспособное приложение на платформе «Windows», использующее язык программирования «Python3», почти полностью удовлетворяющее ТЗ. Для полной функциональности необходимо провести компиляцию и тестирование программы.

## ***4 Тестирование***

### ***4.1 Компиляция программы***

Для компиляции программы на языке «Python3» используем библиотеку «pyinstaller». Скачаем её при помощи «pip» так же, как скачивали «tkcalendar». Далее при помощи командной строки скомпилируем приложение следующими командами:

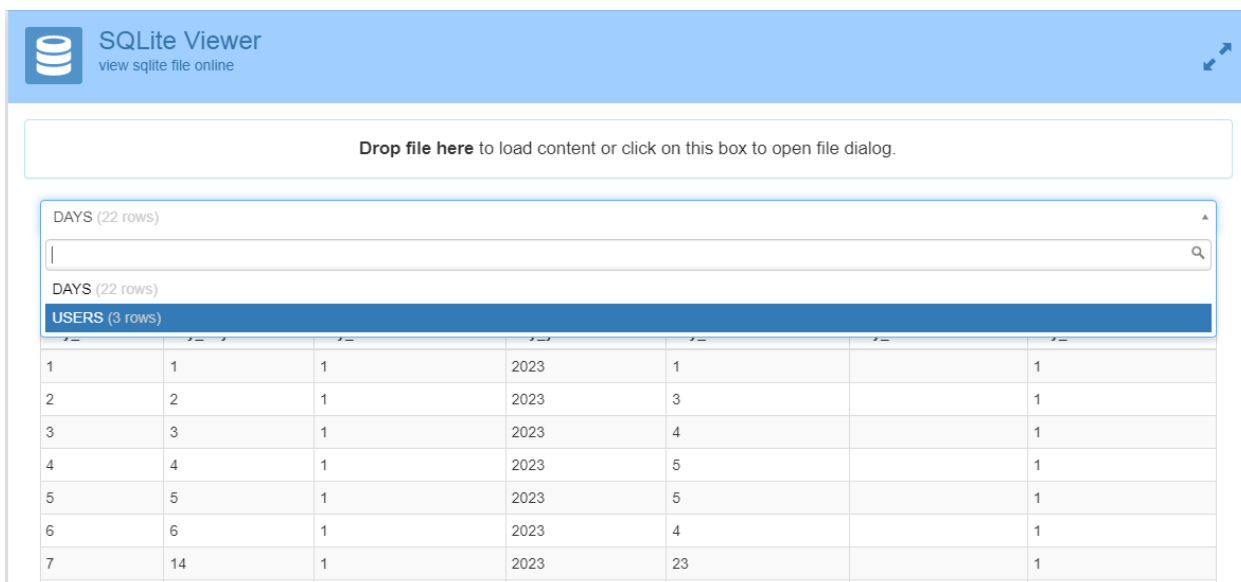
- «cd C:\Users\Румата\Desktop\vkr», меняем текущую директорию на директорию с файлом для компиляции;
- «pyinstaller --onefile --noconsole VKR.py», команда для «pyinstaller» на компиляцию файла «VKR.py» в один файл в режиме оконного приложения без консоли.

Результатом выполнения команды являются две директории «BUILD» и «DIST» и файл с расширением «.spec». Готовый «.exe» файл находится в директории «DIST».

### ***4.2 Проверка файлов базы данных***

Базы данных «SQLite» пользуются довольно большой популярностью. Существует большое количество программ и сервисов для работы с ними, эти же базы данных можно легко импортировать в файлы баз данных других программ.

В результате разработки был создан файл с названием «wetg2.db». Проверим этот файл на помощи онлайн-портала «SQLite Viewer» [10]. В результате увидим, что в базе данных существует две таблицы, увидим количество записей в каждой и что конкретно находится в каждой из строк. Содержимое файла «wetg2.db» изображено на рисунке 9.



1	1	1	2023	1	1
2	2	1	2023	3	1
3	3	1	2023	4	1
4	4	1	2023	5	1
5	5	1	2023	5	1
6	6	1	2023	4	1
7	14	1	2023	23	1

**Рисунок 9 – содержимое файла базы данных**

Исходя из проведённой проверки, файл существует именно в таком виде, в каком задумывался, а также функционирует даже вне приложения, что позволяет пользователю при возникшей необходимости перенести базу данных в другую программу.

#### ***4.3 Методика тестирования готового проекта***

В качестве объекта тестирования создадим базу данных «TEST\_BD.bd». Большая часть необходимых тестов уже была проведена во время разработки приложения путём вывода сообщений в консоль и исправления возникающих ошибок сразу при их появлении. Поэтому на текущем этапе разработки будет проведена дополнительная проверка работоспособности всех элементов приложения после компиляции, а также при специфических данных сотрудников.

Необходимо проверить:

- точность расчёта заработной платы;
- взаимодействие кнопок создания и открытия приложения;
- результат работы при отсутствии данных;

#### 4.4 Тестирование готового приложения

Создадим пользователя «Иванов Иван Иванович» с повременным типом оплаты труда. Укажем ему оклад в 20 000 рублей, 40-часовую рабочую неделю и 19 рабочих дней в феврале. Занесём в календарь данные на все не выходные дни как 8 часов работы за день, но 24 февраля оставим пустым. В результате получим 19 дней по 8 часов работы, что точно сходится с необходимой нормой по времени работы за период. В результате получаем заработную плату, равную окладу сотрудника. Результат представлен на рисунке 10.

The screenshot shows the 'БыстрыйСчёт' application window. The title bar indicates the file path: C:/Users/Рымата/Desktop/dist/Новая папка/newdb.... The interface includes several input fields and buttons for calculating wages.

**Buttons:** Открыть, Новый файл, Новый сотрудник, Сохранить, Сохранить сотрудника, Сохранить данные и рассчитать.

**Input Fields:**

- Иванов Иван Иванович (dropdown menu)
- ФИО (text field)
- Оклад: 20000
- Тип зарплаты: Повременн (dropdown menu)
- Часов в рабочей неделе: 40
- Рабочих дней за период: 19
- Часов работы за период: 152.0
- Начисленно: 20000.0
- Вычтено: НДФЛ: 2600.0
- Итого за месяц: 17400.0
- З.п. 1 (аванс): 8700.0
- З.п. 2: 8700.0
- Часов работы за день: 8

**Calendar:** A calendar for February 2023 is displayed. The date 28 is highlighted in blue. The calendar shows days 1 through 12, with some days having numbers in the background (e.g., 30, 31, 6, 7, 13, 14, 20, 21, 27, 6, 7, 8, 9, 10, 11, 12).

Рисунок 10 – результат расчёта с повременным типом оплаты

Теперь изменим поле «Тип зарплаты» на значение «Сдельная»,



заполним поле «Плата за один продукт (услугу)» как 1000, внесём данные в поле «Продуктов (услуг) за день» как 2 для дат с 1-го по 5-ое января и как 3 для дат с 1-го по 5-ое февраля и запустим подсчёт оплаты для февраля. Создадим текстовый файл «1.txt» для записи результатов. Полученные данные сходятся с расчётами.

Создадим в базе данных нового пользователя, укажем его ставку и количество часов в рабочей неделе, запустим вычисление заработной платы. Из-за проверки на количество отработанных дней ничего не произошло. Результат первого теста представлен на рисунке 11.

The application window 'БыстрыйСчёт' contains the following elements:

- File Operations:** Buttons for 'Открыть' (Open), 'Новый файл' (New file), 'Новый сотрудник' (New employee), 'Сохранить' (Save), and 'Сохранить сотрудника' (Save employee).
- Employee Information:** A dropdown menu for 'Имя' (Name) and a 'Сохранить сотрудника' button.
- Calculation Parameters:**
  - 'Оклад' (Salary): 1200
  - 'Тип зарплаты' (Salary type): 'Повременн:' (Hourly) with a dropdown arrow.
  - 'Часов в рабочей неделе' (Hours in work week): 12
  - 'Рабочих дней за период' (Working days for period): empty field
  - 'Часов работы за период' (Hours of work for period): empty field
- Financial Summary:**
  - 'Начисленно' (Accrued): empty field
  - 'Вычтено: НДФЛ' (Deducted: Income Tax): empty field
  - 'Итого за месяц' (Total for month): empty field
  - 'З.п. 1 (аванс)' (Salary 1 (advance)): empty field
  - 'З.п. 2' (Salary 2): empty field
  - 'Сохранить данные и рассчитать' (Save data and calculate) button.
- Calendar:** A calendar for January 2023. The date '1' is selected and highlighted in blue.
- Additional Fields:**
  - 'Часов работы за день' (Hours of work for day): 12
  - 'Записать' (Record) button.

**Рисунок 11 – результат первого теста**

Используем кнопку «сохранить». В новой базе данных «егор.bd» поменяем значения имён обоих пользователей на «Егор». Первому добавим 8 часов работы за 1 января 2023 года, а второму Егору изменим тип зарплаты. Откроем предыдущую базу данных. Изменения обоих пользователей из базы данных «егор.bd» никак не повлияли на информацию в базе данных «TEST\_BD.bd». Вернёмся в базу данных «егор.bd» при помощи кнопки «Открыть». Информация сохранилась, результат представлен на рисунке 12.

БыстрыйСчёт C:/Users/Рымата/Desktop/dist/Новая папка/newdb....

Открыть Новый файл Новый сотрудник Сохранить сотрудника

Сохранить

ФИО Егор

Тип зарплаты Сдельная

Оплата за один продукт(услугу)

Продуктов (услуг) за период

Начисленно Вычтено: НДФЛ Итого за месяц

З.п. 1 (аванс) З.п. 2 Сохранить данные и рассчитать

January 2023

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
52	26	27	28	29	30	31	1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29
5	30	31	1	2	3	4	5

Продуктов (услуг) за день Записать

**Рисунок 12 – база данных «егор.bd»**

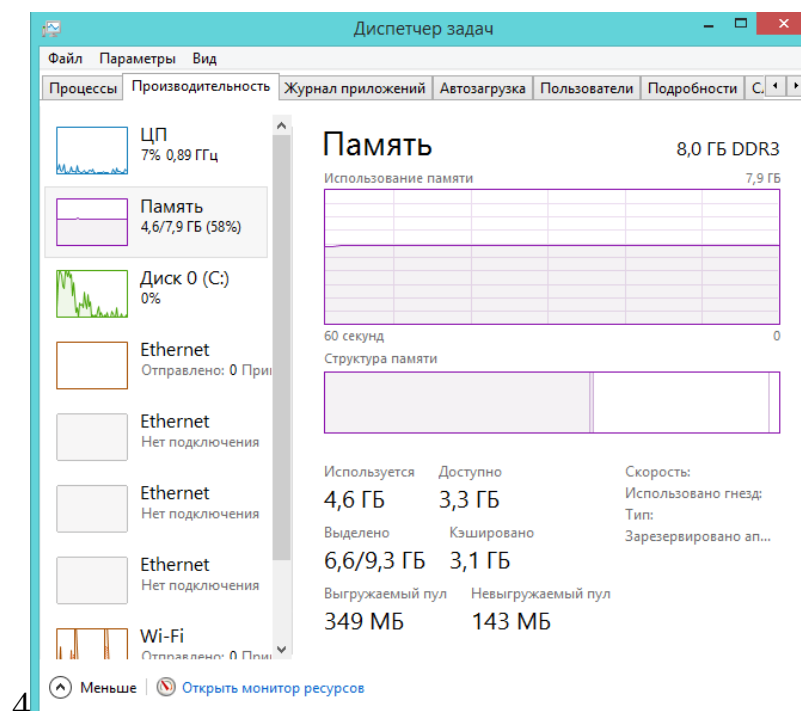
В ходе разработки программы также было создано несколько баз данных и файлов с результатами, эти файлы представлены на рисунке 13.

Этот компьютер > Рабочий стол > dist > Новая папка				
Имя	Дата изменения	Тип	Размер	
12.txt	04.06.2023 19:58	Текстовый докум...	1 КБ	
ASE.db	04.06.2023 20:16	Data Base File	20 КБ	
ASEbackup.db	04.06.2023 20:17	Data Base File	20 КБ	
TEST_BD.db	05.06.2023 19:01	Data Base File	20 КБ	
wetg2.db	04.06.2023 19:57	Data Base File	20 КБ	
wetq.db	04.06.2023 19:55	Data Base File	20 КБ	
Глюкова.txt	04.06.2023 19:57	Текстовый докум...	1 КБ	
egop.db	05.06.2023 19:06	Data Base File	20 КБ	
Катя.txt	04.06.2023 20:16	Текстовый докум...	1 КБ	
цya.txt	04.06.2023 19:59	Текстовый докум...	1 КБ	

**Рисунок 13 – файлы тестирования при разработке**

#### **4.5 Тестирование ресурсозатратности**

Откроем диспетчер задач. Замерим текущее значение для загрузки процессора, оперативной памяти и диска. Откроем приложение, создадим в базе данных пользователя, дадим ему имя, сохраним, вернёмся на первого пользователя, укажем ему все необходимые данные и нажмём кнопку расчёта заработной платы. Закроем приложение. Результаты отладки представлены на рисунке 14.



**Рисунок 14 – диспетчер задач во время тестирования**

Таким образом можно сделать вывод, что приложение почти не использует ресурсы процессора и оперативной памяти, но в значительной мере использует ресурсы жёсткого диска компьютера при открытии программы. Во время работы приложения нагрузка на жёсткий диск заметно уменьшается. Тестирование программы проводилось на компьютерах с операционными системами «Windows 10», «Windows 8.1» и «Windows 7». Программа будет работать на любой системе «Windows», способной работать с языком «Python3».

#### ***4.6 Выводы к главе 4***

В текущей главе в результате тестирования не было выявлено ошибок в работе программы, была исследована нагрузка на оперативную память, процессор и жёсткий диск, был подробнее изучен файл базы данных. В результате тестирования были выполнены последние пункты технического задания.

### ***Заключение***

Во время работы над выпускной квалификационной работой были изучены существующие системы расчёта заработной платы, сформированы необходимые требования к конечному продукту, разработана архитектура приложения, изучены библиотеки языка программирования «Python3», необходимые для реализации проекта.

В результате выпускной квалификационной работы было разработано и протестировано приложение, работающее на операционных системах «Windows», написанное на языке программирования «Python3», способное вести учёт заработной платы сотрудников небольших предприятий, имеющее возможности выбора типа заработной платы, способное взаимодействовать с файлами базы данных «SQLite», основными преимуществами которого являются лёгкость в использовании и установке, и простота в понимании для рядового пользователя.

В ходе преддипломной практики и подготовки выпускной квалификационной работы реализованы следующие профессиональные компетенции в соответствии с ФГОС 09.03.03 «Прикладная информатика» [11] и видами деятельности, предусмотренными основной образовательной программой бакалавриата 09.03.03 Прикладная информатика (уровень бакалавриата) «Современные методы разработки приложений» [12]:

- способен оценивать и выбирать вариант реализации архитектуры программного средства (ПК-1);
- способен разработать тестовые случаи, а также осуществлять проведение тестирования и исследование результатов (ПК-2);
- способен выполнять работы по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы (ПК-3).

### ***Список использованных источников***

1. Компания «1С». Официальный сайт / URL: <https://1c.ru/> (дата обращения 15.05.2023 г.).
2. КонсультантПлюс. Онлайн портал. Трудовой кодекс Российской Федерации / URL: [https://www.consultant.ru/document/cons\\_doc\\_LAW\\_34683/](https://www.consultant.ru/document/cons_doc_LAW_34683/) (дата обращения 16.05.2023 г.).
3. КонсультантПлюс. Онлайн портал. Налоговый кодекс Российской Федерации / URL: [https://www.consultant.ru/document/cons\\_doc\\_LAW\\_19671/](https://www.consultant.ru/document/cons_doc_LAW_19671/) (дата обращения 16.05.2023 г.).
4. Википедия. Язык программирования «C++». Онлайн портал / URL: <https://ru.wikipedia.org/wiki/C%2B%2B> (дата обращения 18.05.2023 г.).
5. Язык программирования «Python». Официальный сайт / URL: <https://www.python.org/> (дата обращения 18.05.2023 г.).
6. База данных «MySQL». Официальный сайт / URL: <https://www.mysql.com/> (дата обращения 19.05.2023 г.).
7. База данных «PostgreSQL». Официальный сайт / URL: <https://www.postgresql.org/> (дата обращения 19.05.2023 г.).
8. Работа с «SQLite3» в языке программирования «Python3» / URL: <https://docs.python.org/3/library/sqlite3.html> (дата обращения 21.05.2023 г.).
9. Учебник по языку запросов «SQL» / URL: <http://www.sql-tutorial.ru/> (дата обращения 21.05.2023 г.).
10. Онлайн портал для просмотра файлов базы данных «SQLite Viewer» / URL: <https://inloop.github.io/sqlite-viewer/> (дата обращения 21.05.2023 г.).
11. Приказ об утверждении федерального государственного образовательного стандарта высшего образования – бакалавриат по направлению подготовки 09.03.03 Прикладная информатика №48531 от 12.10.2017 г. / URL: [https://sfedu.ru/pls/rsu/docs/dir/SPEC\\_DOCS15\\_DIR/Std\\_25156\\_25.06.19.pdf?340924.pdf|application/pdf](https://sfedu.ru/pls/rsu/docs/dir/SPEC_DOCS15_DIR/Std_25156_25.06.19.pdf?340924.pdf|application/pdf) (дата обращения: 19.03.2023г.).

12. Основная образовательная программа высшего образования по направлению 09.03.03 Прикладная информатика (уровень бакалавриата) «Современные методы разработки приложений» / URL: [https://sfedu.ru/pls/rsu/docs/dir/SPEC\\_DOCS15\\_DIR/OOP\\_25156\\_25.06.19.pdf?340510.pdf|application/pdf](https://sfedu.ru/pls/rsu/docs/dir/SPEC_DOCS15_DIR/OOP_25156_25.06.19.pdf?340510.pdf|application/pdf) (дата обращения: 19.03.2023г.).

13. Гитхаб. Служба размещения в интернете репозитория Git. Код готового приложения / URL: <https://github.com/EneRumata/VKR> (дата обращения: 19.03.2023г.)

В данном приложении к выпускной квалификационной работе рассматриваются инструкция к работе с приложением.

Программа состоит из файла «БыстрыйСчёт.exe» и «pick.ico». Для установки приложения разместите файл расширения «.exe» в любую директорию на своём компьютере. Файл «pick.ico» не является обязательным, это всего лишь иконка приложения.

Создайте свою первую базу данных при помощи кнопки «Создать файл». Создастся файл с одним сотрудником без данных. В текстовых полях интерфейса можно менять информацию о текущем сотруднике. Чтобы выбрать другого сотрудника, используйте выпадающий список с именами сотрудников. Изменять информацию о сотруднике можно как в момент его создания в базе данных, так и в дальнейшем при его выборе посредством выпадающего списка сотрудников. После изменения информации о сотруднике не забудьте сохранить её одноимённой кнопкой интерфейса. Чтобы добавить сотрудника в базу данных, используйте одноимённую кнопку.

Для каждого сотрудника существует собственный календарь. Выберите любой день и укажите количество отработанных часов или произведённых продуктов (услуг). Не забудьте сохранить информацию кнопкой «Записать». Как только все необходимые данные будут внесены, нажмите на кнопку расчёта зарплаты.

В случае возникновения ошибок будет выведено сообщение о месте, в котором возникла ошибка. В ином случае появится окно с выбором пути файла, в который будет записана информация. Итоги расчётов появятся в интерфейсе программы, а также будут записаны в файл, если пользователь выбрал создать его.