|  |
| --- |
| **Министерство образования и науки Российской Федерации**  Федеральное государственное автономное образовательное  учреждение высшего образования  **«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**  **ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»** |

Подразделение: Инженерная школа энергетики

Направление подготовки: 09.04.03 – Прикладная информатика

Отделение: Электроэнергетики и электротехники

**Проектная документация**

**Отчёт по лабораторной работе №5**

по дисциплине: «Основы объектно-ориентированного программирования»

Выполнил студент гр. О-5КМ21 \_\_\_\_\_\_\_\_\_\_ Пушкарев М. А. 17 октября 2024 г.

Отчёт принял доцент, к.т.н. \_\_\_\_\_\_\_\_\_\_ Калентьев А. А. \_\_\_\_\_\_\_ 2024 г.

Томск 2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc179995815)

[1. UML диаграмма вариантов использования 4](#_Toc179995816)

[2. UML диаграмма классов 5](#_Toc179995817)

[3. Описание классов, образующих связь типа «общее-частное» 7](#_Toc179995818)

[4. Дерево ветвлений Git 9](#_Toc179995819)

[5. Тестирование программы 10](#_Toc179995820)

[5.1. Тестовый случай «Добавить элемент» 10](#_Toc179995821)

[5.2. Тестовый случай «Удалить элемент» 13](#_Toc179995822)

[5.3. Тестовый случай «Очистить список» 14](#_Toc179995823)

[5.4 Тестовый случай «Найти элемент» 15](#_Toc179995824)

[5.5. Тестовый случай «Сброс фильтра» 19](#_Toc179995825)

[5.6. Тестовый случай «Сохранить данные» 20](#_Toc179995826)

[5.7. Тестовый случай «Загрузить данные» 22](#_Toc179995827)

[Список использованных источников 25](#_Toc179995828)

[ПРИЛОЖЕНИЕ А 26](#_Toc179995829)

**Введение**

Корректная и полная документация сопровождает разработку программного обеспечения (далее – ПО) от появления идеи до выпуска конечного продукта. Написание документации является обязательным критерием разработки и последующей поддержки проекта [1].

Целью данной лабораторной работы является разработка проектной документации на созданный программный продукт.

Для достижения поставленной цели должны быть выполнены следующие задачи:

* Составление технического задания (далее – ТЗ) на разработанную программу (Приложение А);
* Составление UML диаграммы вариантов использования для разработанной программы;
* Составление UML диаграммы классов;
* Описание классов, образующих связь типа «общее-частное»;
* Привести дерево ветвлений Git;
* Провести тестирование программы.

**1. UML диаграмма вариантов использования**

Вариант использования (use case) — это описание множества последовательных действий (включая вариации), которые выполняются некоторым субъектом с целью получения результата, значимого для некоторого действующего лица [1]. ВИ предполагает взаимодействие действующих лиц и системы или другого объекта. Действующее лицо представляет собой логически связанное множество ролей, которые играют пользователи системы во время взаимодействия с ней.

Диаграмма вариантов использования для разработанного ПО приведена на рисунке 1.

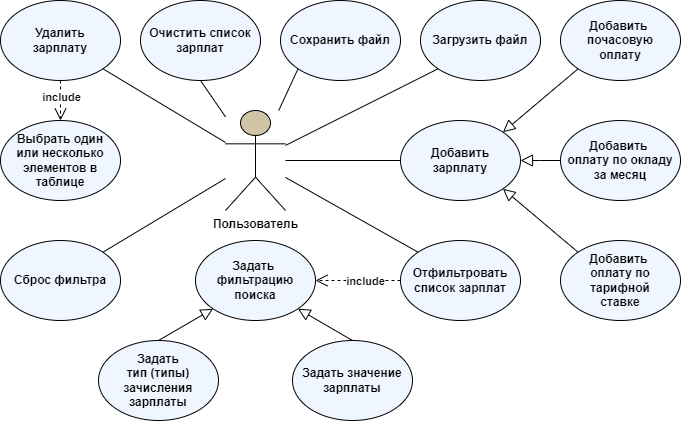


Рисунок 1. Диаграмма вариантов использования

**2. UML диаграмма классов**

Одним из ключевых аспектов проектирования программных систем является детальное описание структуры и взаимодействий между объектами. UML-диаграмма классов является основным инструментом для визуализации и документирования этих структурных аспектов. Она позволяет разработчикам представить классы системы, их атрибуты, методы, а также связи между ними, такие как ассоциации, агрегации, композиции и наследование.

Диаграмма классов предоставляет возможность наглядно продемонстрировать, как различные элементы системы взаимосвязаны и как данные передаются и преобразуются внутри системы. Это особенно важно на этапе проектирования, так как позволяет выявить и устранить потенциальные проблемы в архитектуре системы до начала ее реализации.

Диаграмма классов приведена на рисунке 2.

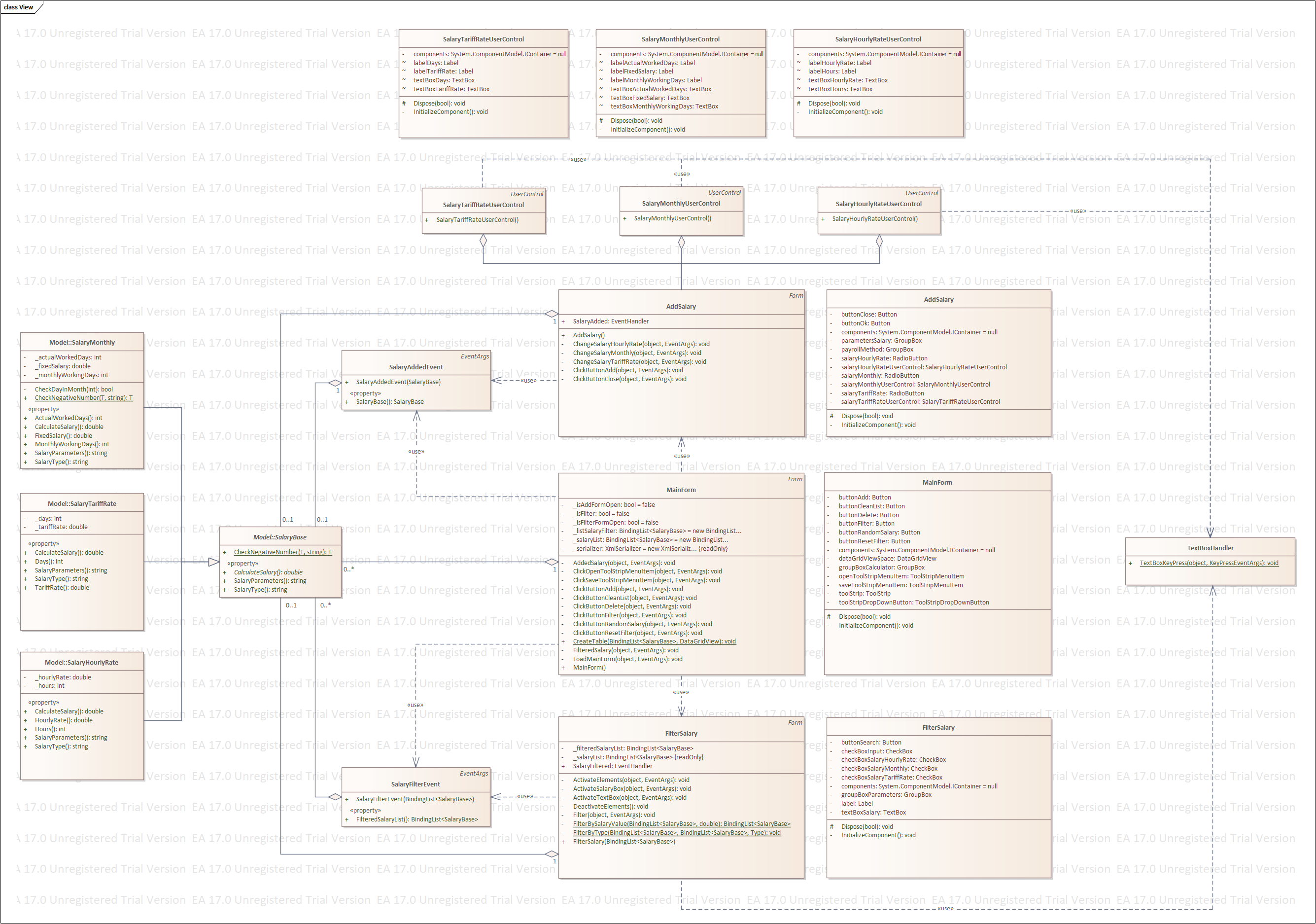


Рисунок 2. UML диаграмма классов

**3. Описание классов, образующих связь типа «общее-частное»**

В таблице 1 приведено описание абстрактного класса *SalaryBase* с его полями, свойствами и методами.

Таблица 1. Описание класса *SalaryBase*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс *SalaryBase* – абстрактный базовый класс для заработных плат | | |
| Свойства | | |
| + SalaryType | string | Тип зарплаты.  Абстрактное свойство, переопределяется в производных классах. |
| + SalaryParameters | string | Параметры зарплаты.  Абстрактное свойство, переопределяется в производных классах. |
| + CalculateSalary | double | Размер заработной платы.  Абстрактное свойство, переопределяется в производных классах. |
| Методы | | |
| + CheckNegativeNumber | double | Проверяет корректность ввода зарплаты. |

В таблицах 2 – 4 приведены описания классов SalaryTariffRate, SalaryMonthly и SalaryHourlyrate, которые наследуются от *SalaryBase*.

Таблица 2. Описание класса SalaryTariffRate

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс SalaryTariffRate – оплата по тарифной ставке | | |
| Поля | | |
| – \_tariffRate | double | Размер тарифной ставки. |
| – \_days | int | Количество фактически отработанных дней. |
| Свойства | | |
| + TariffRate | double | Размер тарифной ставки. |
| + Days | int | Количество рабочих дней. |
| + SalaryType | string | Тип заработной платы. |
| + SalaryParameters | string | Параметры для расчёта заработной платы. |
| + CalculateSalary | double | Вычисление зарплаты по тарифной ставке. |

Таблица 3. Описание класса SalaryMonthly

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс SalaryMonthly – оклад | | |
| Поля | | |
| – \_fixedSalary | double | Размер оклада. |
| – \_monthlyWorkingDays | int | Количество рабочих дней в месяце. |
| – \_actualWorkedDays | int | Количество фактически отработанных дней. |
| Свойства | | |
| + FixedSalary | double | Размер оклада. |
| + MonthlyWorkingDays | int | Количество рабочих дней в месяце. |
| + ActualWorkedDays | int | Количество фактически отработанных дней. |
| + SalaryType | string | Тип заработной платы. |
| + SalaryParameters | string | Параметры для расчёта заработной платы. |
| + CalculateSalary | double | Вычисление зарплаты по окладу. |
| Методы | | |
| + CheckDayInMonth | double | Проверяет корректность ввода количества дней в месяце. |

Таблица 4. Описание класса SalaryHourlyrate

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс SalaryHourlyrate – часовая тарифная ставка | | |
| Поля | | |
| – \_hourlyRate | double | Размер часовой тарифной ставки. |
| – \_hours | int | Количество фактически отработанных часов. |
| Свойства | | |
| + HourlyRate | double | Размер часовой тарифной ставки. |
| + Hours | int | Количество фактически отработанных часов. |
| + SalaryType | string | Тип заработной платы. |
| + SalaryParameters | string | Параметры для расчёта заработной платы. |
| + CalculateSalary | double | Вычисление зарплаты по часовой тарифной ставке. |

**4. Дерево ветвлений Git**

Git [2] — распределённая система управления версиями файлов. Система спроектирована как набор утилит командной строки, специально разработанных с учётом их использования в скриптах. Git обеспечивает управление версиями кода и способствует эффективной совместной работе над проектом.

На рисунке 3 представлено дерево ветвлений Git, полученное по окончании работы с проектом.

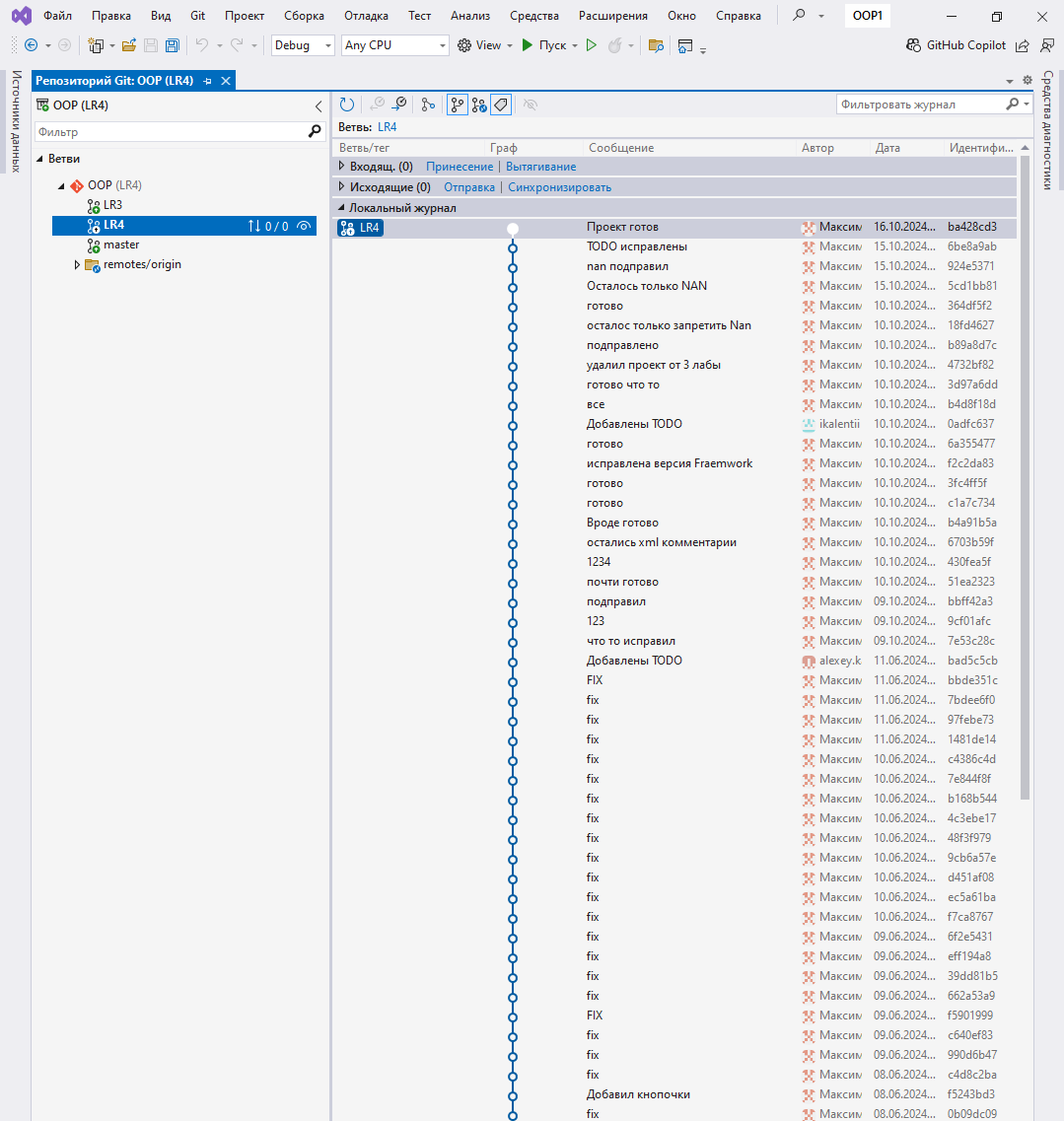


Рисунок 3. Дерево ветвлений Git

**5. Тестирование программы**

Далее приводится процесс функционального тестирования программы.

Графический интерфейс пользователя представлен на рисунке 4.

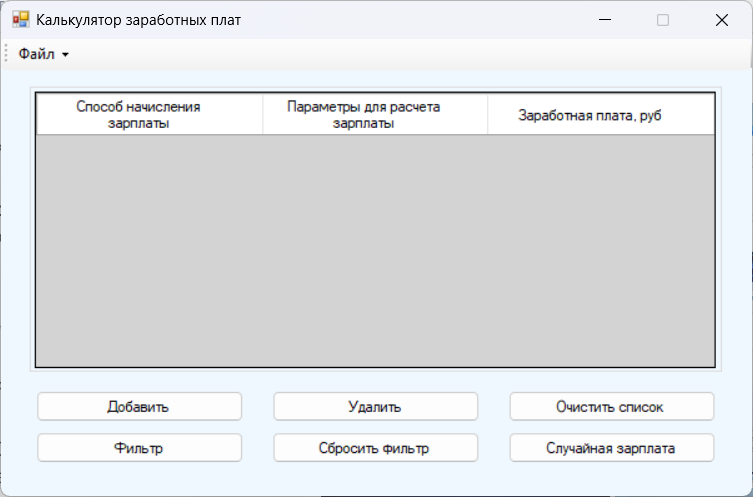


Рисунок 4. Графический интерфейс пользователя

**5.1. Тестовый случай «Добавить элемент»**

Для добавления элемента необходимо вызвать соответствующую форму путём нажатия кнопки «Добавить» (рисунок 5).

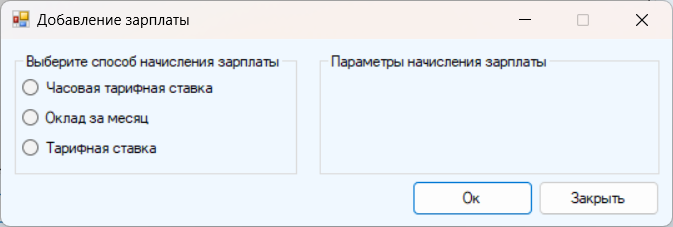


Рисунок 5. Форма для добавления элемента

Параметры любой из выбранных зарплат (часовая тарифная ставка, оклад за месяц, тарифная ставка) можно ввести, нажав на соответствующую кнопку.

После ввода данных необходимо нажать кнопку «ОК», элемент появится в таблице главной формы (рисунки 6 и 7).

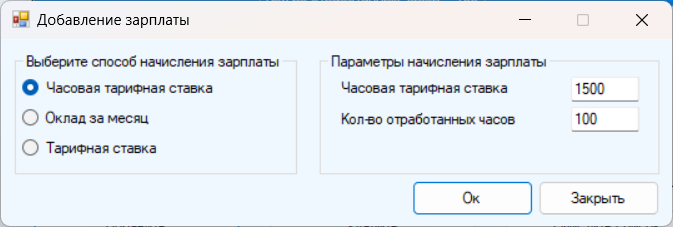


Рисунок 6. Заполнение полей

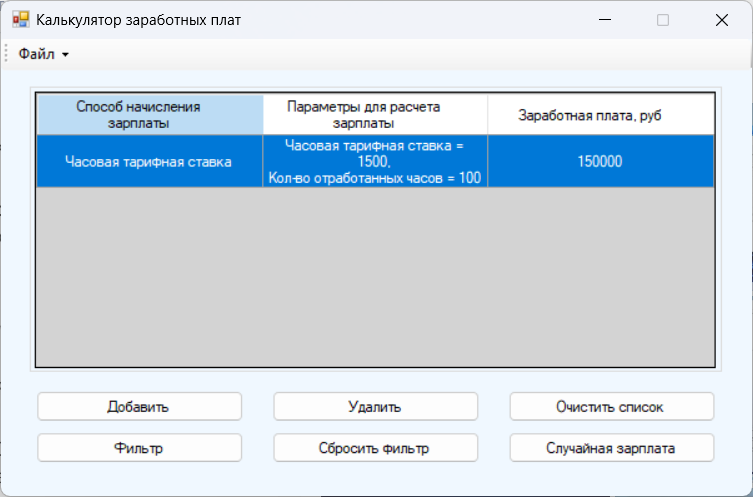


Рисунок 7. Успешное добавление нового элемента

В программе установлено ограничение на ввод символов, отличных от цифр, что препятствует пользователю вводить недопустимые символы. Также программа не даст ввести число с несколькими запятыми.

В случае, если пользователь не заполнил требуемые поля, то программа выдает Предупреждение «Введите данные.». На рисунках 8 и 9 представлен пример, когда незаполненно поле «Кол-во отработанных часов». Обработка незаполненных полей предусмотрена для всех способов начисления заработной платы.

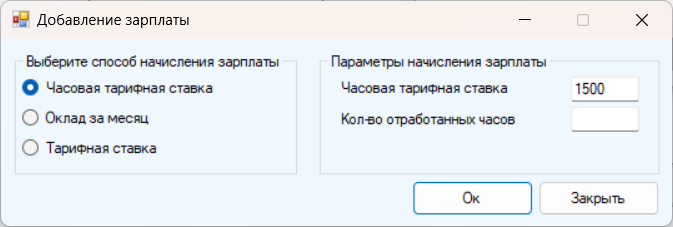


Рисунок 8. Незаполненное поле

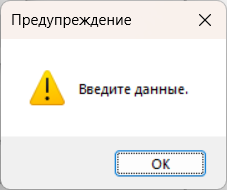


Рисунок 9. Сообщение о предупреждении

Подобная обработка ошибок предусмотрена для всех параметров заработной платы.

В случае ввода в поля «Кол-во рабочих дней в месяце» и «Кол-во отработанных дней» значений больше, чем количество дней в текущем месяце, то программа выдает Предупреждение «Количество дней в месяце не может быть больше 31.» (рисунок 10 и 11).

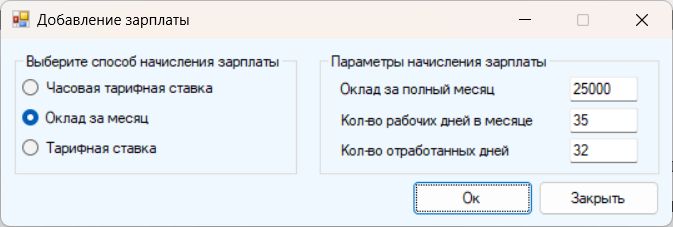


Рисунок 10. Некорректный ввод данных

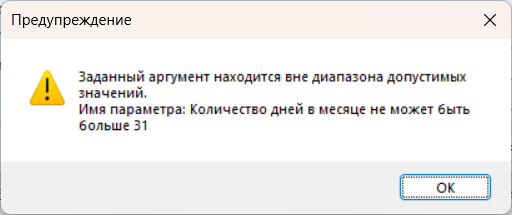


Рисунок 11. Сообщение о предупреждении

**5.2. Тестовый случай «Удалить элемент»**

Для удаления одного или нескольких элементов необходимо выбрать их в таблице и нажать на кнопку «Удалить» (рисунки 12 – 15).

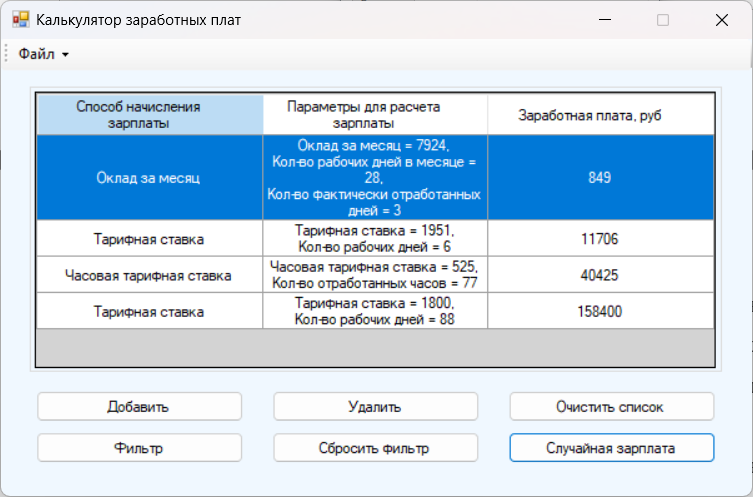


Рисунок 12. Выбор элемента в таблице

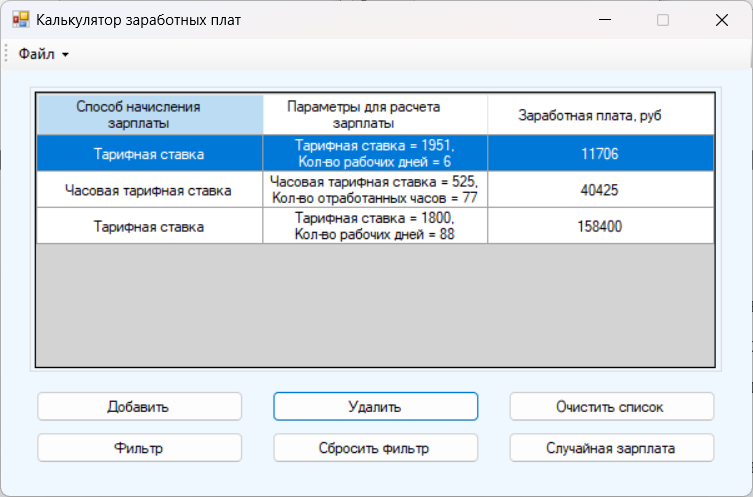


Рисунок 13. Результат нажатия кнопки «Удалить»

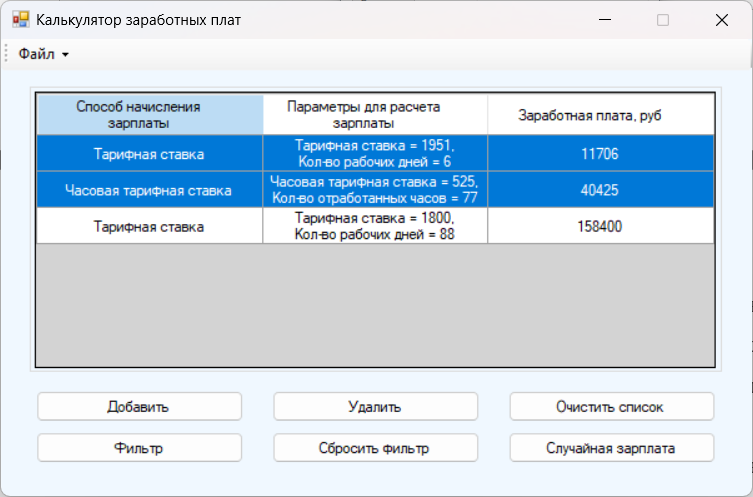


Рисунок 14. Выбор нескольких элементов для удаления

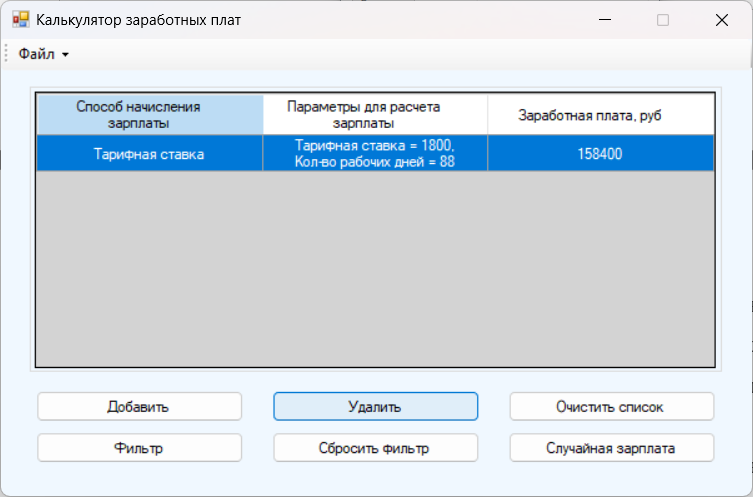


Рисунок 15. Результат удаления выбранных элементов

**5.3. Тестовый случай «Очистить список»**

Для того, чтобы очистить список необходимо нажать на кнопку «Очистить список» (рисунок 16 и 17).

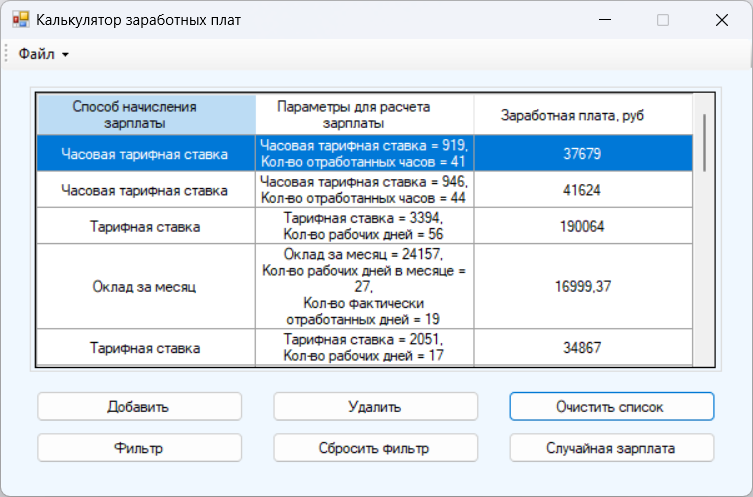


Рисунок 16. Нажатие кнопки «Очистить список»

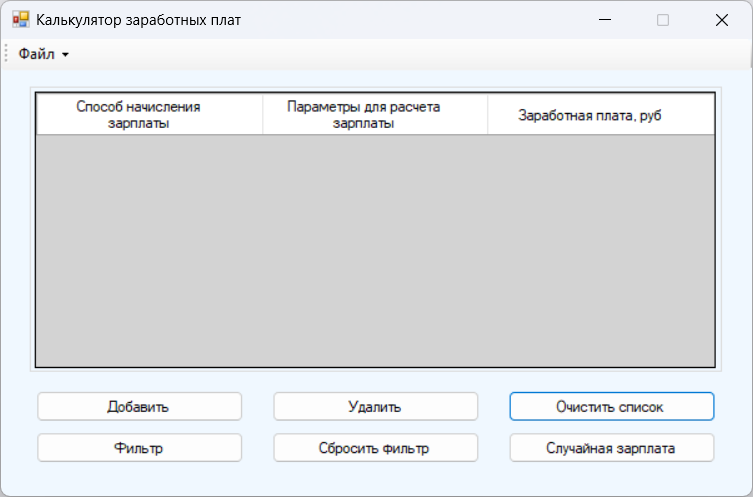


Рисунок 17. Результат нажатия на кнопку «Очистить список»

**5.4 Тестовый случай «Найти элемент»**

Для фильтрации фигур по определенным критериям необходимо нажать кнопку «Фильтр». Откроется соответствующая форма для поиска (рисунок 18).

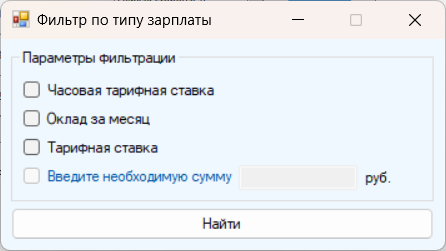


Рисунок 18. Вызов формы для фильтрации списка фигур

Ниже приведены примеры фильтрации списка фигур по разным критериям:

* Тип «Часовая тарифная ставка»

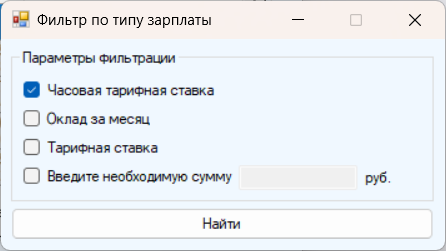


Рисунок 19. Поиск зарплаты по типу «Часова тарифная ставка»

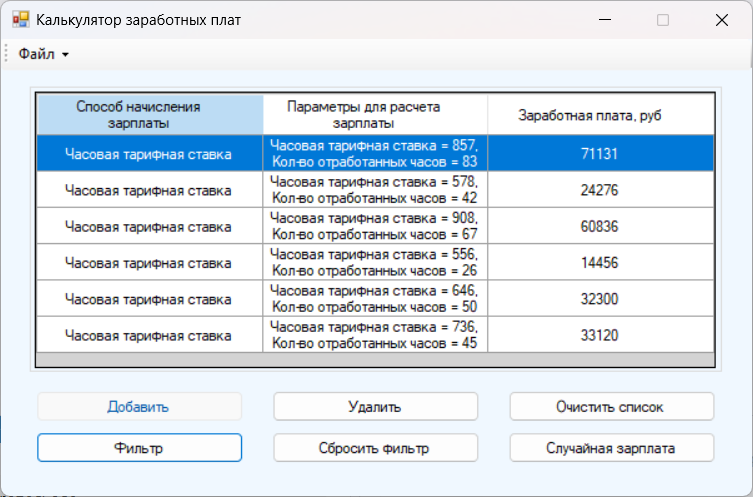


Рисунок 20. Результат поиска зарплаты по типу «Часова тарифная ставка»

* Тип «Оклад за месяц»

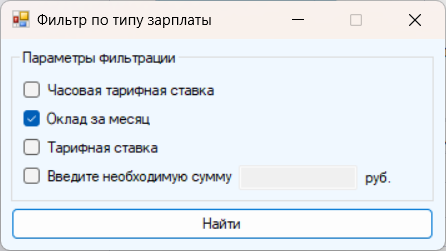


Рисунок 21. Поиск зарплаты по типу «Оклад за месяц»

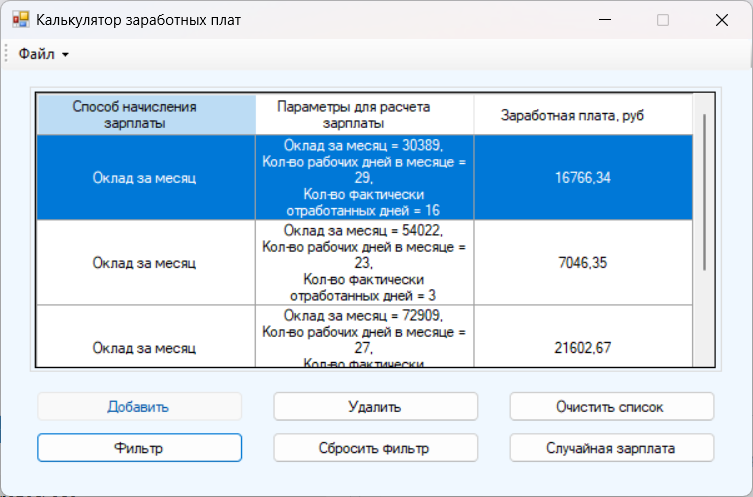


Рисунок 22. Результат поиска зарплаты по типу «Оклад за месяц»

* Тип «Тарифная ставка»

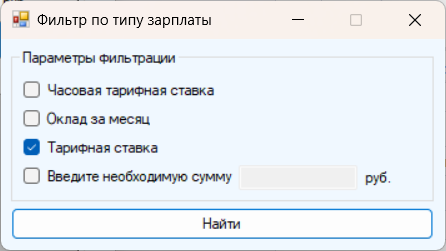


Рисунок 23. Поиск зарплаты по типу «Тарифная ставка»

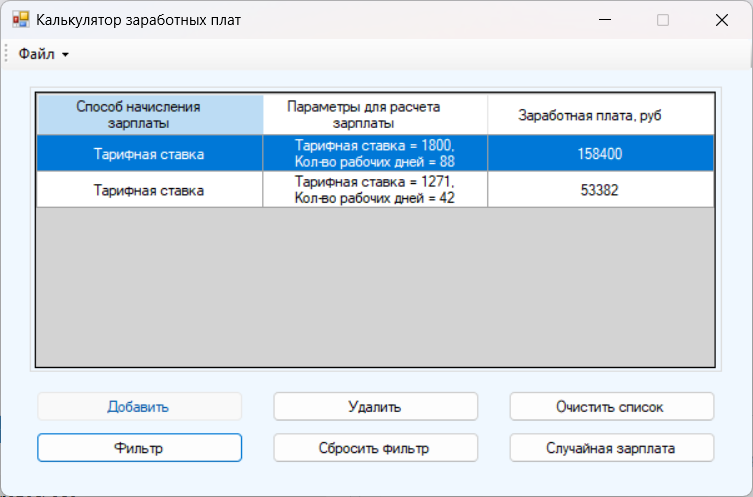


Рисунок 24. Результат поиска зарплаты по типу «Тарифная ставка»

* Значение «Сумма»

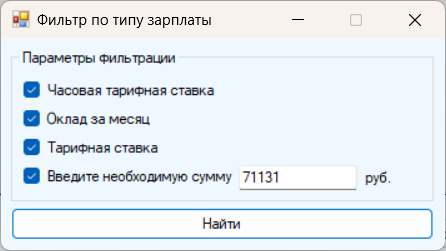


Рисунок 25. Поиск зарплаты по значению «Сумма»

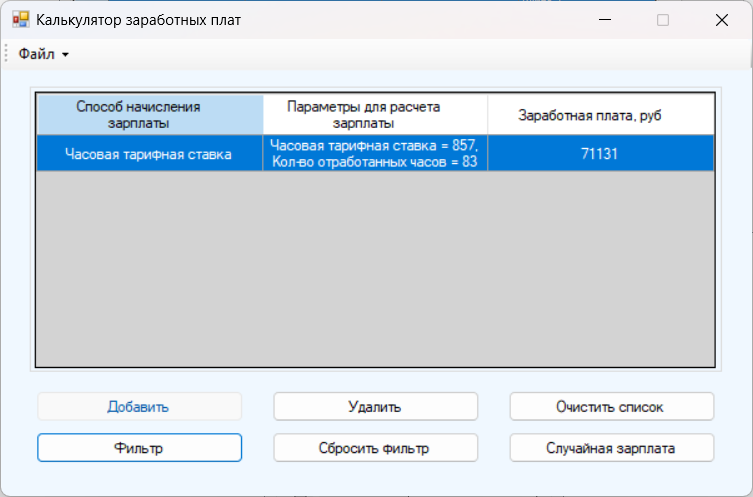


Рисунок 26. Результат поиска зарплаты по значению «Сумма»

На рисунках 27 и 28 представлена реакция системы на некорректное поведение пользователя.

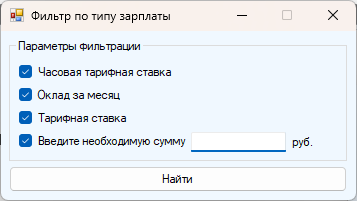


Рисунок 27. Нажатие кнопки «Найти» при незаполненном поле

«Введите необходимую сумму»

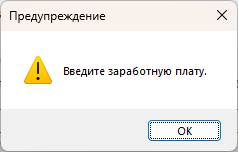


Рисунок 28. Предупреждение пользователя при незаполненном поле

«Введите необходимую сумму»

**5.5. Тестовый случай «Сброс фильтра»**

На рисунках 29 и 30 представлена работа функции сброса фильтра.

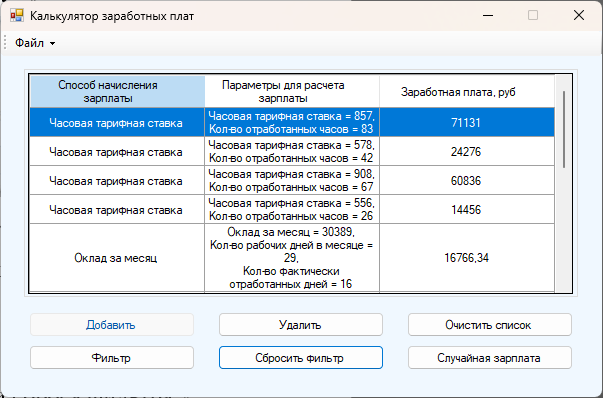


Рисунок 29. Нажатие кнопки «Сбросить фильтр»

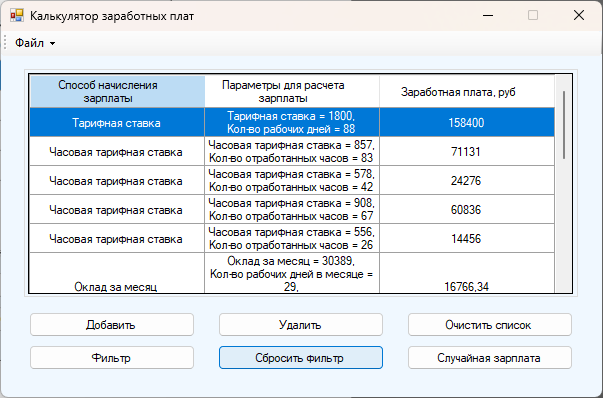


Рисунок 30. Отображение исходного списка после сброса фильтра

**5.6. Тестовый случай «Сохранить данные»**

Для сохранения данных в таблице необходимо нажать на панели «Файл\Сохранить» (рисунок 31). Откроется системный диалог сохранения файла, где пользователь выбирает директорию и указывает имя файла (рисунок 32).

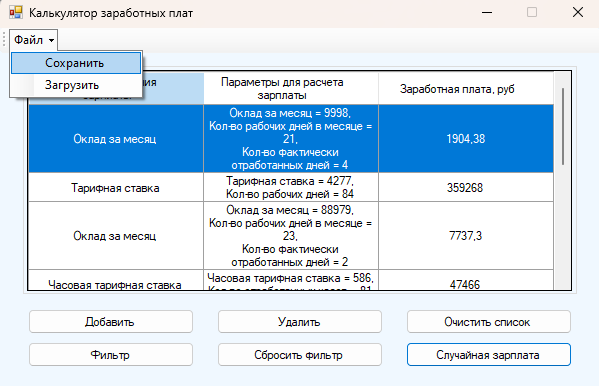


Рисунок 31. Сохранение данных

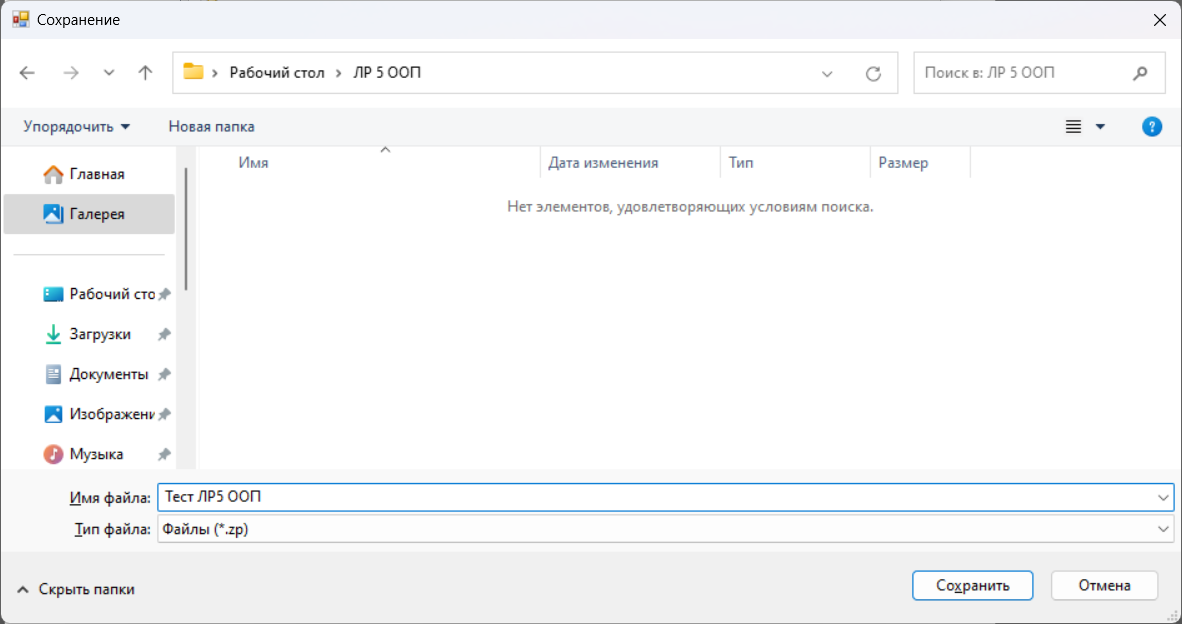


Рисунок 32. Сохранение файла

После сохранения данных в файл появится соответствующее сообщение (рисунки 33 и 34).

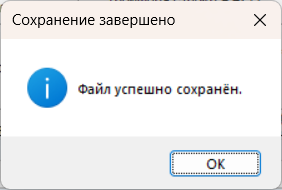


Рисунок 33. Сообщение о сохранении файла

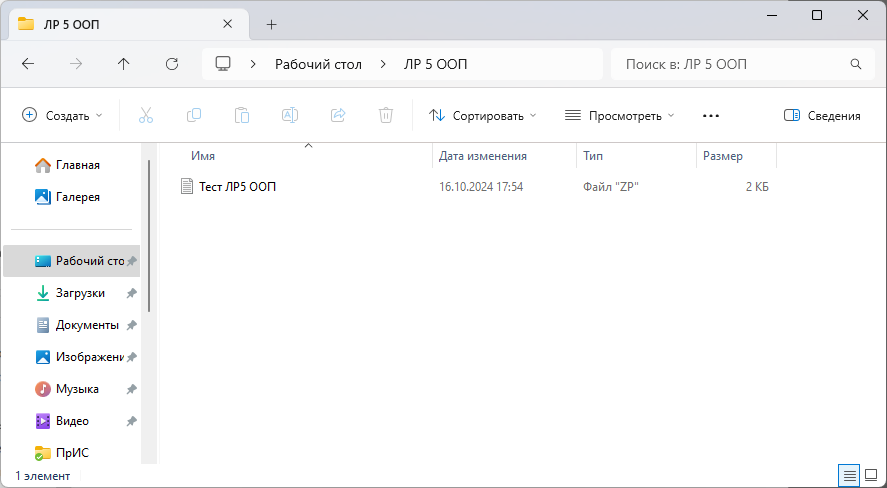


Рисунок 34. Результат сохранения файла

В случае, если таблица пуста, сохранение не производится (рисунок 35).

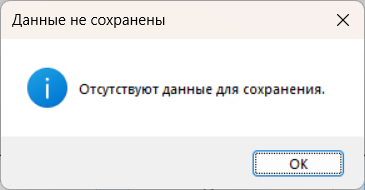


Рисунок 35. Результат нажатия на кнопку «Сохранить» при пустой таблице

**5.7. Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо нажать на соответствующую кнопку (рисунок 36).

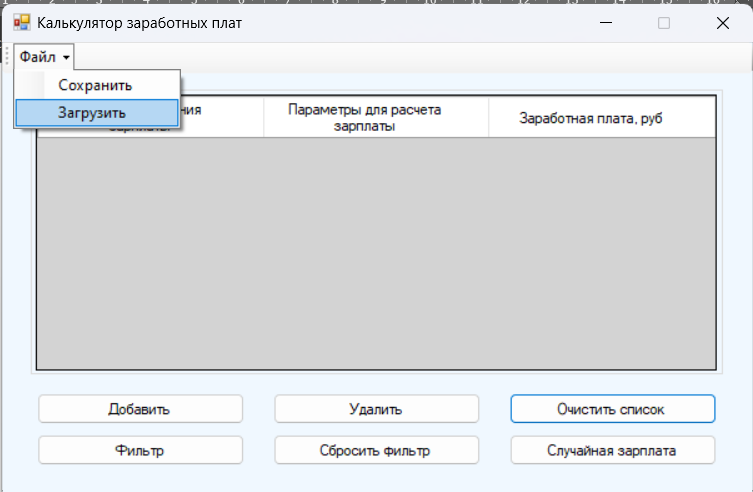


Рисунок 36. Загрузить данные в таблицу

Далее откроется системный диалог загрузки файла (рисунок 37). После успешной загрузки появится сообщение (рисунки 37 – 39).

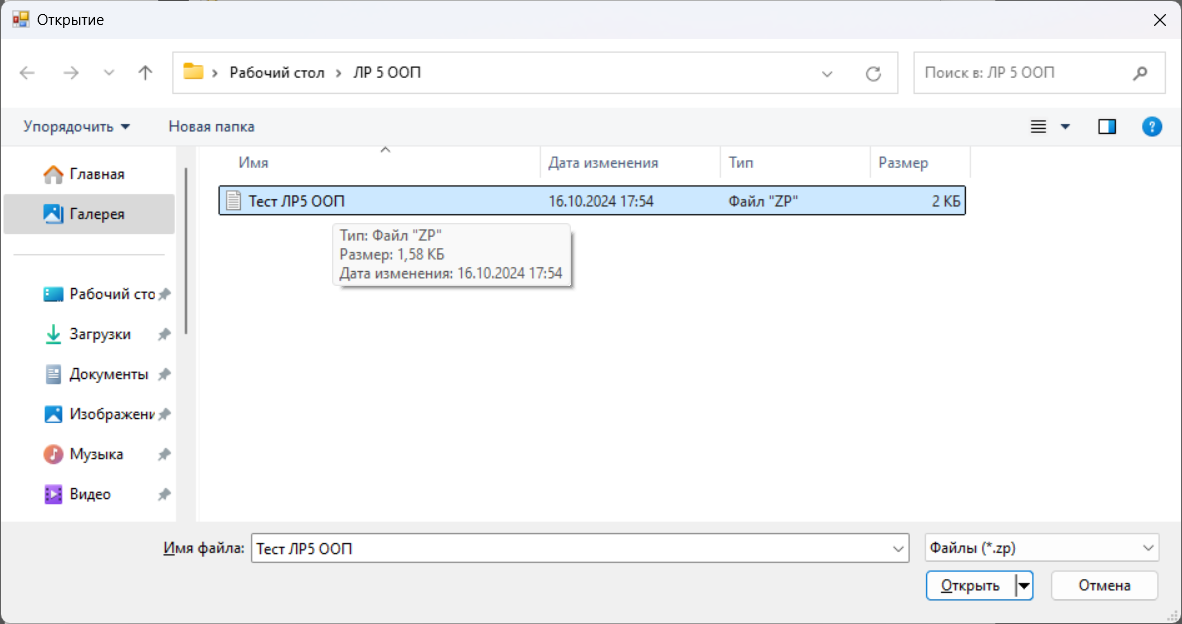


Рисунок 37. Выбор файла для загрузки

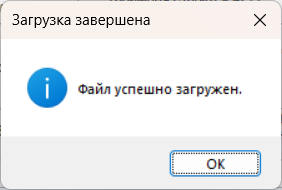


Рисунок 38. Сообщение, информирующее об успешной загрузке файла

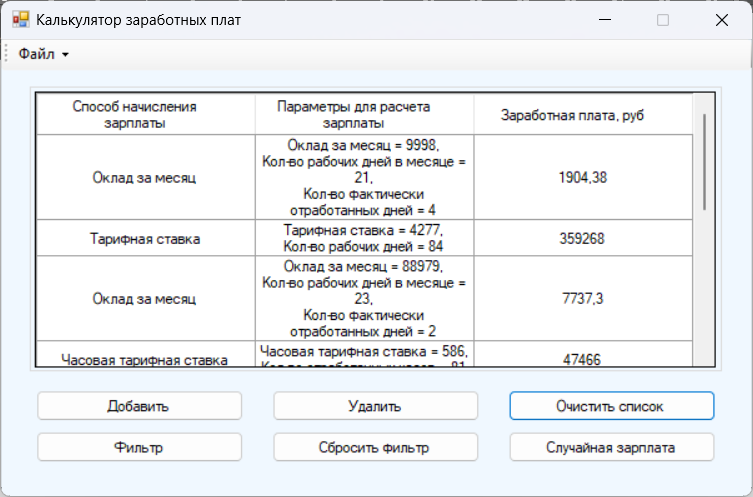


Рисунок 39 Результат загрузки данных

В случае, если структура загружаемого файла не распознана, или в файле присутствуют некорректные значения параметров, или значения отсутствуют, появится соответствующее сообщение, представленное на рисунке 40.

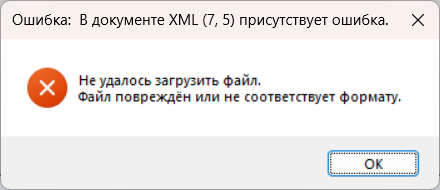


Рисунок 40. Загрузка некорректного файла

**Список использованных источников**

1. Калентьев, А. А. Новые технологии в программировании: учебное пособие / А. А. Калентьев, Д. В. Гарайс, А. Е. Гориянов. – Томск : Эль Контент, 2014. – 176 с. – ISBN 978-5-4332-0185-9.

2. Git [Электронный ресурс]. — URL : http://git-scm.com/ (Дата обращения: 16.10.2024)

**ПРИЛОЖЕНИЕ А**

**Техническое задание на создание автоматизированной системы**

Программа для расчёта заработной платы

Разработчик: студент гр. О-5КМ21 НИ ТПУ Пушкарев М. А.

Заказчик: канд. техн. наук, доцент каф. КСУП ТУСУР Калентьев А. А.

Томск 2024

**1. Общие сведения**

**1.1. Полное наименование системы и её условное обозначение**

Полное наименование: «Программа для расчёта заработной платы».

Условное обозначение: «Калькулятор ЗП»

**1.2. Наименование предприятий разработчика и заказчика системы**

Заказчик: Канд. техн. наук, доцент каф. КСУП ТУСУР Калентьев А. А.

Разработчик: Студент гр. О-5КМ21 НИ ТПУ Пушкарев М. А.

**1.3. Плановые сроки начала и окончания работы по созданию системы**

Начало работ: 21 мая 2024 г.

Окончание работ: 17 октября 2024 г.

**2. Назначение и цели создания системы**

**2.1. Назначение системы**

Система предназначена для расчёта разных видов заработной платы: часовой тарифной ставке, оклада за месяц, тарифной ставке.

**2.2. Цели создания системы**

Система создаётся в целях сокращения трудозатрат бухгалтеров при расчётах заработной платы работников.

**3. Характеристика объектов автоматизации**

Вычисление заработной платы может пригодиться на любом предприятии.

Поскольку такие расчёты выполняются бухгалтерами при формировании бюджета компании ежемесячно, представляется целесообразным автоматизировать этот процесс.

**4 Требования к системе**

Таблица 4.1. Префиксы мнемонических идентификаторов требований и их расшифровка

|  |  |
| --- | --- |
| **Префикс** | **Тип требования** |
| A | Архитектурное требование |
| С | Требование к программной или аппаратной совместимости |
| D | Требование к структуре данных |
| F | Функциональное требование |
| U | Требование к пользовательскому интерфейсу |

**4.1 Требования к архитектуре**

**А01.** Система должна быть реализована в виде настольного приложения.

**4.2 Требования к структуре данных**

**D01.** Данные о параметрах заработной платы должны храниться в формате \*.zp.

Структура XML-документа представляет собой иерархическую организацию элементов, начиная от корневого элемента и включая вложенные элементы. В случае данного XML-документа структура будет следующей:

**<ArrayOfSalaryBase>** <!-- Корневой объект -->

<**SalaryBase**> <!-- Элемент, представляющий способ начисления ЗП -->

<!-- Дополнительные элементы, зависящие от способа начисления ЗП -->

**</SalaryBase>**

<!-- Возможно, другие элементы SalaryBase, представляющие другие способы начисления ЗП -->

**</ArrayOfSalaryBase>**

Для каждого способа начисления заработной платы внутри элемента *SalaryBase* будут различные дочерние элементы, содержащие информацию о характеристиках каждого способа.

* Для типа*SalaryMonthly*:

<**SalaryBase** xsi:type="**SalaryMonthly**">

<**FixedSalary**>!-- Оклад за полный месяц --</**FixedSalary**>

<**MonthlyWorkingDays**>!-- Кол-во рабочих дней в месяце --</**MonthlyWorkingDays**>

<**ActualWorkedDays**>!-- Кол-во фактически отработанных дней в месяце --</**ActualWorkedDays**>

</**SalaryBase**>

* Для типа*SalaryTariffRate*:

</**SalaryBase**>

<**SalaryBase** xsi:type="**SalaryTariffRate**">

<**TariffRate**>!-- Размер тарифной ставки --</</**TariffRate**>

<**Days**>!-- Кол-во отработанных дней --</</**Days**>

</**SalaryBase**>

* Для типа*SalaryHourlyRate*:

</**SalaryBase**>

<**SalaryBase** xsi:type="**SalaryHourlyRate**">

<**HourlyRate**>!-- Размер часовой тарифной ставки --</</</**HourlyRate**>

<**Hours**>!-- Кол-во отработанных часов --</</</**Hours**>

</**SalaryBase**>

**4.3 Функциональные требования**

**F01.** Система должна рассчитывать заработную плату следующих видов:

* часовая тарифная ставка;
* оклад за месяц;
* тарифная ставка.

**F01.01.** Размер часовой тарифной ставки должен определяться по выражению

где *a* – размер часовой тарифной ставки, руб;

*b* – количество отработанных часов, шт.

**F01.02.** Размер оклада за месяц должен определяться по выражению

где *a* – оклад, руб;

*b* – количество рабочих дней в месяце, шт;

– количество фактически отработанных дней, шт.

**F01.03.** Размер тарифной ставки должен определяться по выражению

где *a* – размер тарифной ставки, руб;

*b* – количество фактически отработанных дней, шт.

**F02.** В системе должен быть реализован список элементов заработной платы.

**F02.01.** Каждый элемент должен иметь следующие параметры:

* тип заработной платы;
* значение заработной платы.

**F03.** В системе должна присутствовать функция добавления элементов в список.

**F04.** В системе должна присутствовать функция удаления элементов из списка.

**F05.** В системе должна присутствовать функция очистки списка.

**F06.** В системе должна присутствовать функция поиска элементов по параметрам, указанным в **F02.01**.

**F07.** В системе должна присутствовать функция сохранения списка элементов в файл (**D01**).

**F08.** В системе должна присутствовать функция загрузки списка элементов из файла (**D01**).

**4.4 Требования к пользовательскому интерфейсу**

**U01.** Система должна иметь графический интерфейс пользователя.

**U02.** Данные должны быть представлены в табличном виде.

**U03.** В системе должна быть реализована система обработки ошибок.

**4.5 Требования к программному обеспечению**

**C01.** Система должна работать на операционной системе 64-бит Windows 11 22H2. Работоспособность на других выпусках и версиях не гарантируется.

**C02.** На рабочей станции должен быть установлен .NET Framework версии 4.8.

**C03.** Процессор: процессор с частотой 1 гигагерц (ГГц) или выше

**C04.** ОЗУ: 1 ГБ для 32-разрядных систем или 2 ГБ для 64-разрядных систем.

**C05.** Место на жестком диске: 50 МБ.