|  |
| --- |
| **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-5КМ11 | |  | \_\_\_\_\_\_\_\_\_\_ | Кутявин В.В. | |
|  | |  |  |  | |
|  | |  |  | 28 декабря 2023 г. | |
|  | |  |  |  |  |
| Отчёт принял | доцент, к.т.н. |  | \_\_\_\_\_\_\_\_\_\_ |  | Калентьев А. А. |
|  |  |  |  |  |  |
|  |  |  |  | \_\_\_\_\_\_\_\_\_ 2023 г. | |

Томск 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc154407789)

[1 Основная часть 4](#_Toc154407790)

[1.1 UML диаграмма вариантов использования 4](#_Toc154407791)

[1.2 UML диаграмма классов 5](#_Toc154407792)

[1.3 Описание классов, образующих связь типа «общее-частное» 8](#_Toc154407793)

[1.4 Дерево ветвлений Git 10](#_Toc154407794)

[1.5 Тестирование программы 11](#_Toc154407795)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc154407796)

[ПРИЛОЖЕНИЕ А 23](#_Toc154407797)

**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 Основная часть**

**1.1 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.

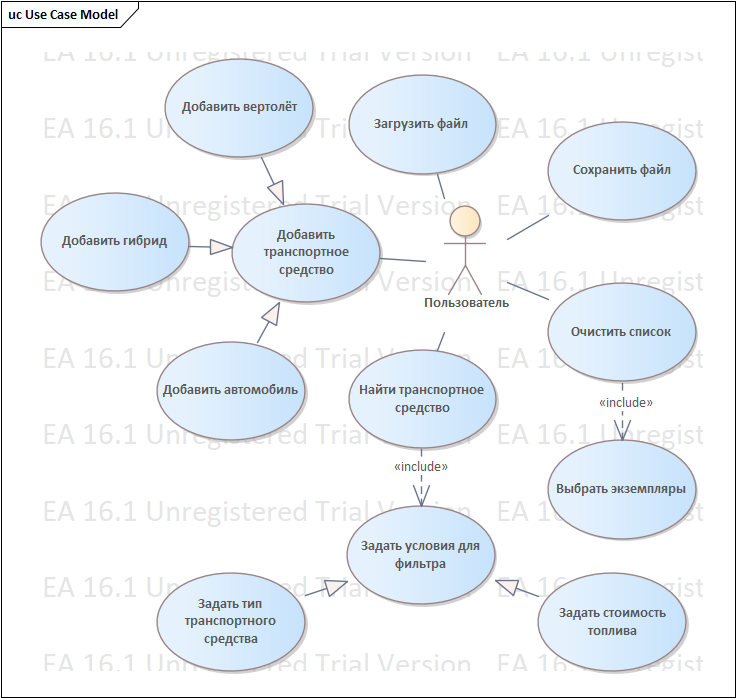


Рисунок 1 – Диаграмма вариантов использования

**1.2 UML диаграмма классов**

Диаграмма классов — это центральная методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов приведена на рисунке 2.

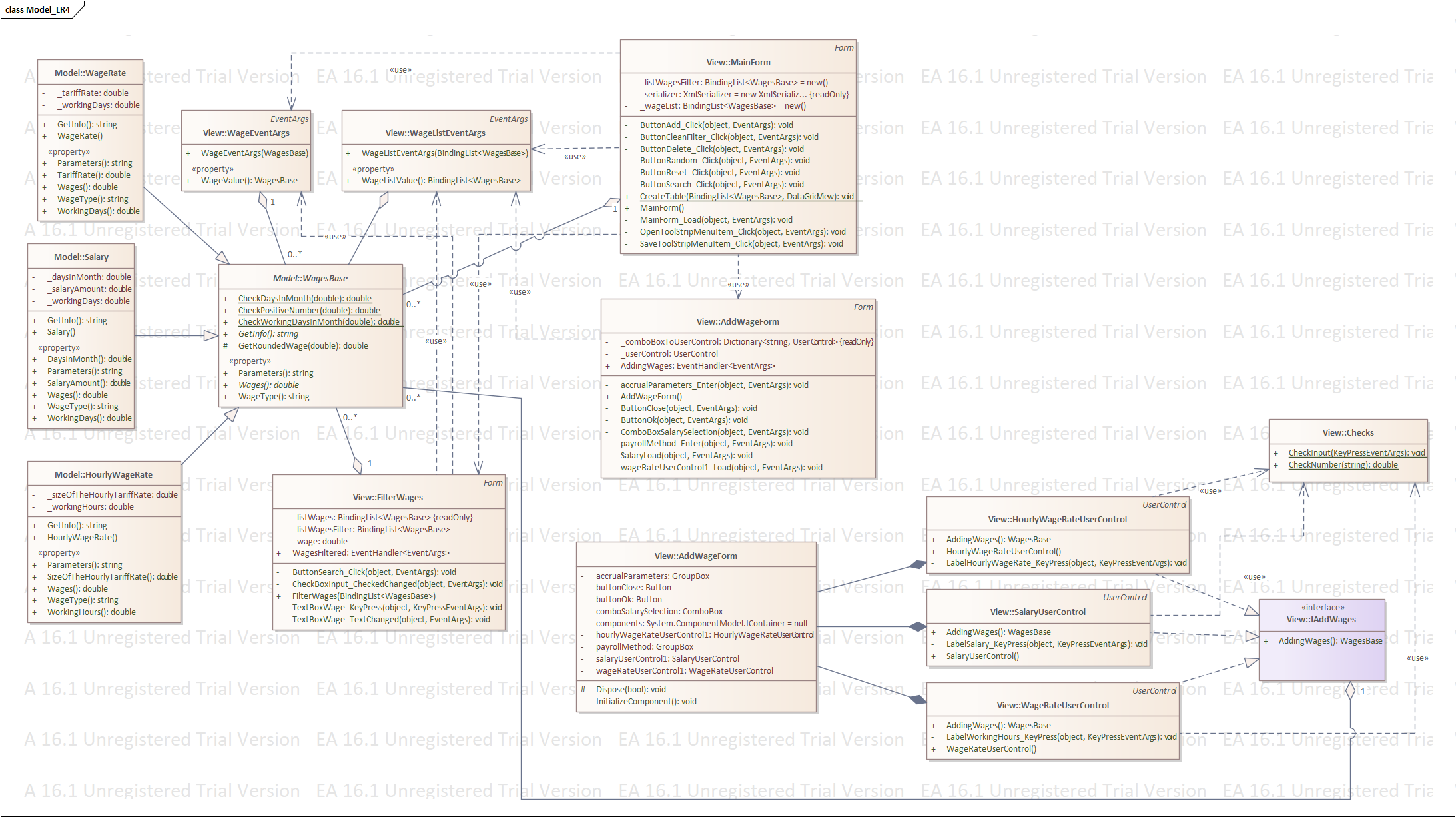


Рисунок 2 – UML диаграмма классо

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

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

Таблица 1 – Описание класса *WagesBase*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *WagesBase* – абстрактный базовый класс для заработных плат | | |
| Свойства | | |
| + WageType | string | Тип зарплаты.  Абстрактное свойство, переопределяется в производных классах. |
| + Wages | double | Размер заработной платы.  Абстрактное свойство, переопределяется в производных классах. |
| + Parameters | string | Параметры зарплаты.  Абстрактное свойство, переопределяется в производных классах. |
| Методы | | |
| + CheckPositiveNumber | double | Проверяет корректность ввода зарплаты |
| + CheckDaysInMonth | double | Проверяет корректность ввода количества дней в месяце |
| + CheckWorkingDaysInMonth | double | Проверяет корректность ввода количества рабочих дней в месяце |
| + GetInfo | string | Информация о зарплате.  Абстрактный метод, переопределяется в производных классах. |
| + GetRoundedWage | double | Округляет величину заработной платы до второго знака |

В таблицах 2–4 приведены описания классов WageRate, Salary и HourlyWageRate, которые наследуются от *WagesBase*.

Таблица 2 – Описание класса WageRate

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс WageRate – тарифная ставка | | |
| Поля | | |
| – \_tariffRate | double | Размер тарифной ставки |
| – \_workingDays | double | Количество фактически отработанных дней |
| Свойства | | |
| + TariffRate | double | Размер тарифной ставки |
| + WorkingDays | double | Количество рабочих дней |
| + WageType | string | Тип заработной платы |
| + Parametrs | string | Параметры для расчёта заработной платы |
| + GetInfo | string | Информация о зарплате |
| + Wages | double | Вычисление зарплаты по тарифной ставке |

Таблица 3 – Описание класса Salary

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс Salary – оклад | | |
| Поля | | |
| – \_salaryAmount | double | Размер оклада |
| – \_daysInMonth | double | Количество рабочих дней в месяце |
| –\_workingDays | double | Количество фактически отработанных дней |
| Свойства | | |
| + SalaryAmount | double | Размер оклада |
| + DaysInMonth | double | Количество рабочих дней в месяце |
| + WorkingDays | double | Количество фактически отработанных дней |
| + WageType | string | Тип заработной платы |
| + Parametrs | string | Параметры для расчёта заработной платы |
| + GetInfo | string | Информация о зарплате |
| + Wages | double | Вычисление зарплаты по окладу |

Таблица 4 – Описание класса HourlyWageRate

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс HourlyWageRate – часовая тарифная ставка | | |
| Поля | | |
| – \_sizeOfTheHourlyTariffRate | double | Размер часовой тарифной ставки |
| – \_workingHours | double | Количество фактически отработанных часов |
| Свойства | | |
| + SizeOfTheHourlyTariffRate | double | Размер часовой тарифной ставки |
| + WorkingHours | double | Количество фактически отработанных часов |
| + WageType | string | Тип заработной платы |
| + Parametrs | string | Параметры для расчёта заработной платы |
| + GetInfo | string | Информация о зарплате |
| + Wages | double | Вычисление зарплаты по часовой тарифной ставке |

**1.4 Дерево ветвлений Git**

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

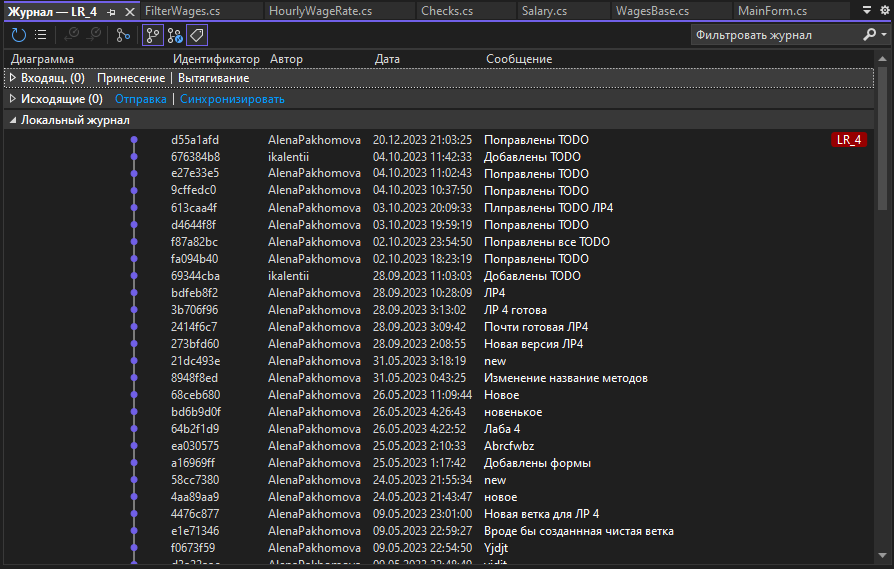


Рисунок 3 – Дерево ветвлений Git

**1.5 Тестирование программы**

Далее приводится процесс функционального тестирования программы.

Графический интерфейс пользователя представлен на рисунке 4.

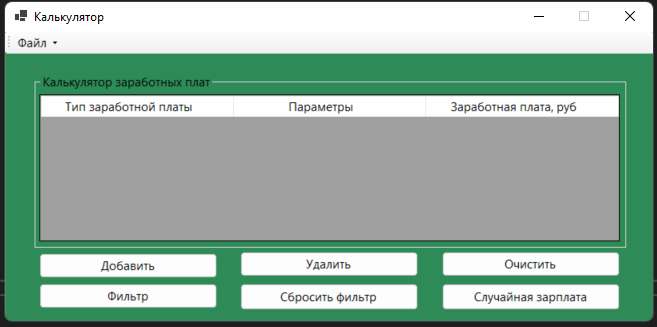


Рисунок 4 – Графический интерфейс пользователя

**1.5.1 Тестовый случай «Добавить элемент»**

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

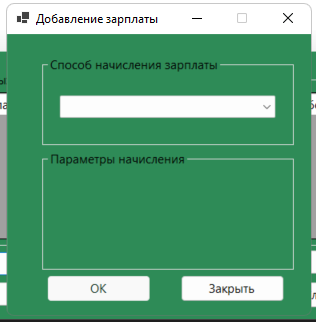


Рисунок 5 – Форма для добавления элемента

Параметры любой из выбранных зарплат (оклад, тарифная ставка, часовая тарифная ставка) можно ввести, выбрав соответствующую зарплату в выпадающем меню.

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

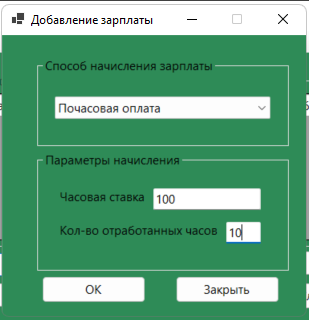


Рисунок 6 – Заполнение полей

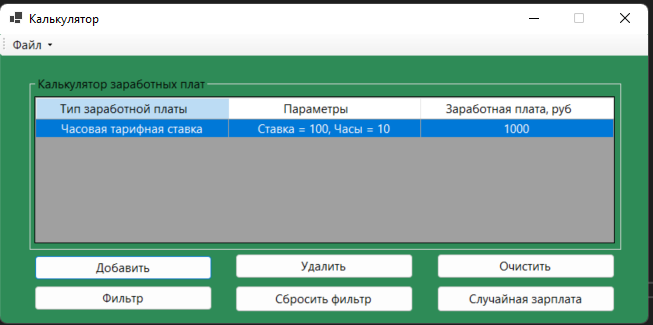


Рисунок 7 – Успешное добавление нового элемента

В программе предусмотрена система обработки некорректного ввода данных пользователем. Например, при попытке ввести число с несколькими запятыми появится соответствующее сообщение об ошибке (рисунки 8-9). Аналогичным образом обрабатывается попытка ввода NaN.

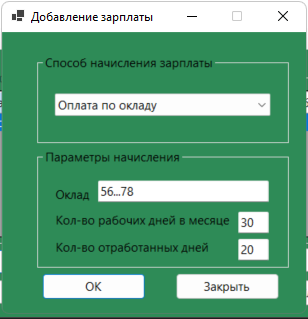


Рисунок 8 – Некорректный ввод

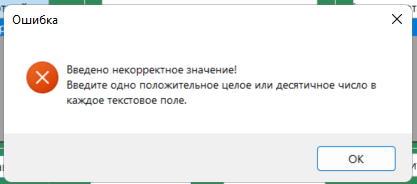


Рисунок 9 – Сообщение об ошибке

Подобная обработка ошибок предусмотрена для всех параметров заработной платы.

**1.5.2 Тестовый случай «Удалить элемент»**

Для удаления одного или нескольких элементов необходимо выбрать их в таблице и нажать на кнопку «Удалить» (рисунки 10 и 11).

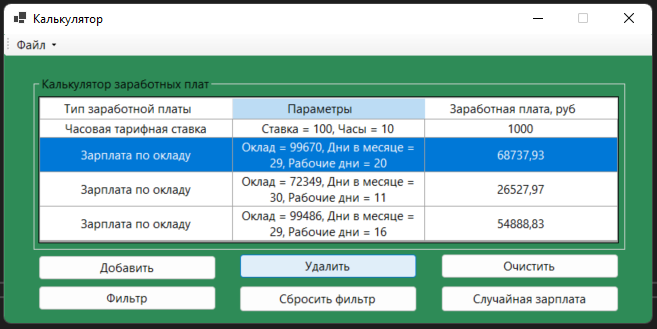


Рисунок 10 – Выбор элемента в таблице

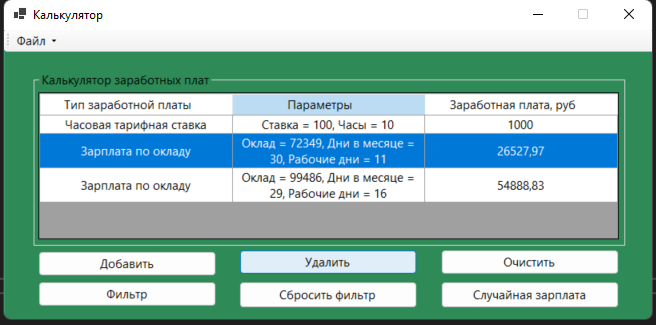


Рисунок 11 – Результат нажатия кнопки «Удалить»

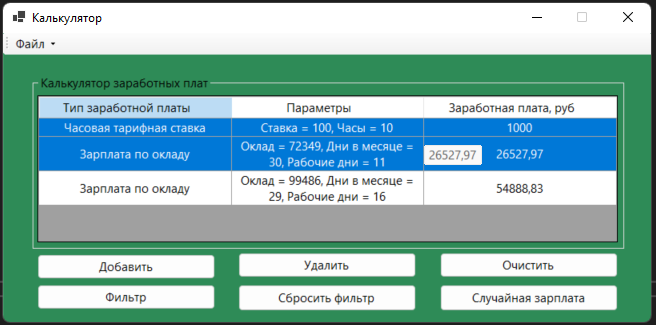


Рисунок 12 – Выбор нескольких элементов для удаления

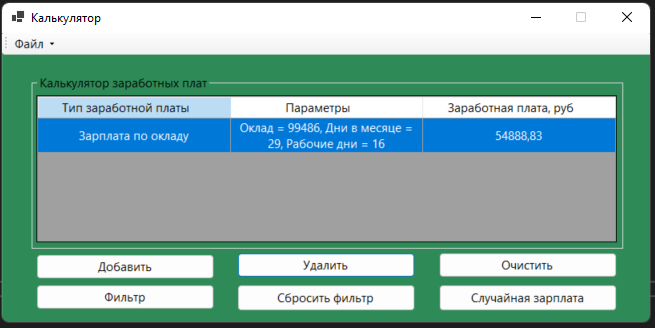


Рисунок 13 – Результат удаления выбранных элементов

**1.5.3 Тестовый случай «Найти элемент»**

Для поиска элементов в списке необходимо нажать кнопку «Фильтр» (рисунок 14). Откроется соответствующая форма для поиска (рисунок 15).

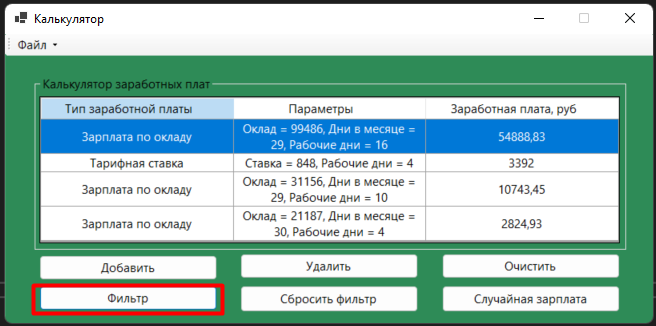


Рисунок 14 – Вызов формы для поиска элементов

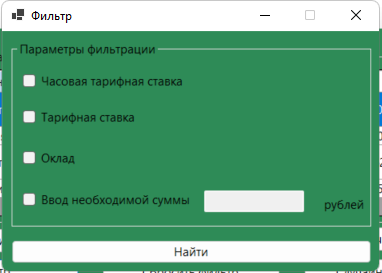


Рисунок 15 – Форма для поиска

Далее пользователь выбирает параметры, по которым требуется найти элемент, вводит значения этих параметров и нажимает кнопку «Найти» (рисунки 16 и 17).

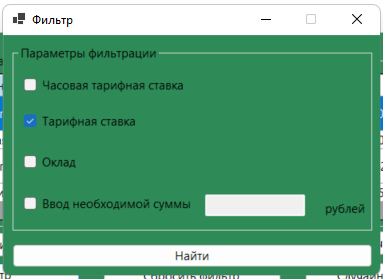


Рисунок 16 – Поиск элемента по названию

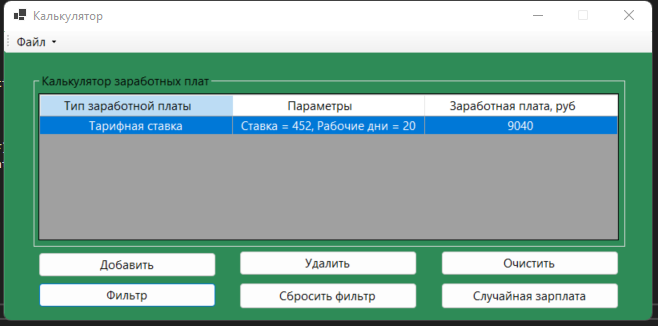


Рисунок 17 – Результат поиска элемента

Для того, чтобы сбросить фильтр, предусмотрена соответствующая кнопка «Сбросить фильтр» (рисунок 18).

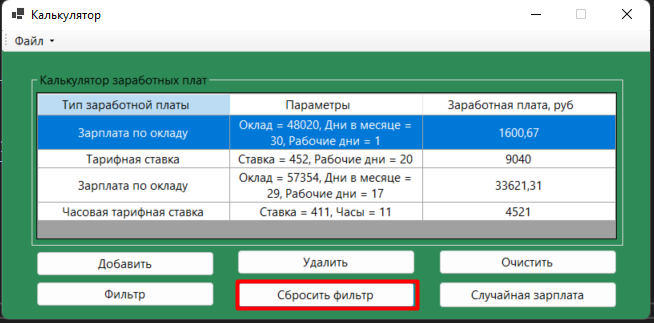


Рисунок 18 – Сброс условий поиска

**1.5.4 Тестовый случай «Сохранить данные»**

Для сохранения данных в таблице необходимо нажать на панели «Файл\Сохранить» (рисунок 19). Откроется системный диалог сохранения файла, где пользователь выбирает директорию и указывает имя файла (рисунок 20).

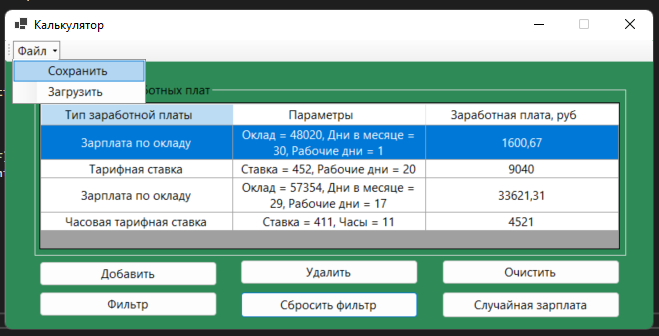


Рисунок 19 – Сохранение данных

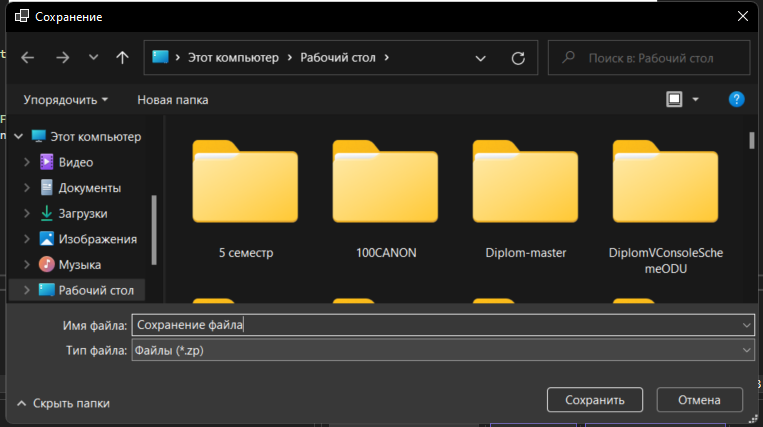


Рисунок 20 – Сохранение файла

После сохранения данных в файл появится соответствующее сообщение (рисунки 21 и 22).

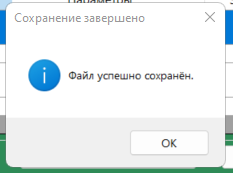


Рисунок 21 – Сообщение о сохранении файла

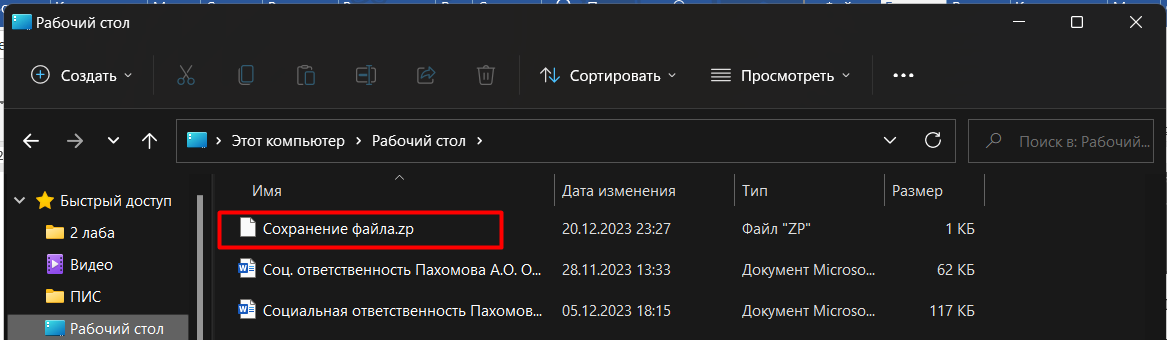


Рисунок 22 – Результат сохранения файла

В случае, если таблица пуста, сохранение не производится (рисунок 23).

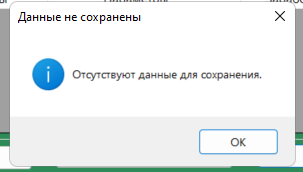


Рисунок 23 – Результат нажатия на кнопку «Сохранить» при пустой таблице

**1.5.5 Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо нажать на соответствующую кнопку (рисунок 24).

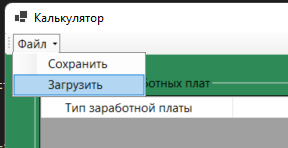


Рисунок 24 – Загрузить данные в таблицу

Далее откроется системный диалог загрузки файла (рисунок 25). После успешной загрузки появится сообщение (рисунки 26 и 27).

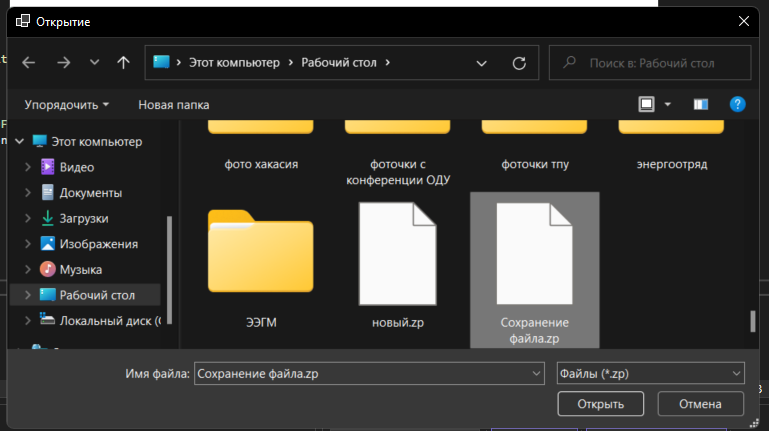


Рисунок 25 – Выбор файла для загрузки

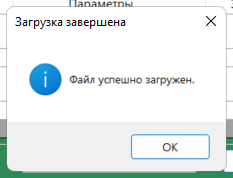


Рисунок 26 – Сообщение, информирующее об успешной загрузке файла

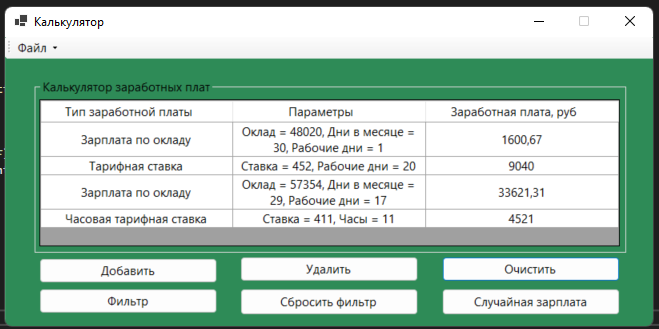


Рисунок 27 – Результат загрузки данных

**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.

**ПРИЛОЖЕНИЕ А**

**Техническое задание на создание автоматизированной системы**

Программа для расчёта заработной платы

Разработчик: студент гр. О-5КМ11 НИ ТПУ Пахомова А.О.

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

Томск 2023

**1 Общие сведения**

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

Полное наименование: «Программа для расчёта заработной платы».

Условное обозначение: Система.

**1.2 Наименование предприятий разработчика и заказчика системы**

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

Разработчик: Студент гр. О-5КМ11 НИ ТПУ Пахомова А.О.

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

Начало работ: 1 мая 2023 г.

Окончание работ: 20 декабря 2023 г.

**2 Назначение и цели создания системы**

**2.1 Назначение системы**

Система предназначена для расчёта разных видов заработной платы: оклада, тарифной ставки и часовой тарифной ставки.

**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 XML-файле с расширением \*.zp.

**4.3 Функциональные требования**

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

* оклад;
* тарифная ставка;
* часовая тарифная ставка.

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

где *a* – размер тарифной ставки, руб;

*b* – количество фактически отработанных дней, шт.

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

где *a* – оклад, руб;

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

– количество фактически отработанных дней, шт.

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

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

*b* – количество фактически отработанных часов, 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 поиска элементов по параметрам, указанным в **F02.01**.

**F06.** В системе должна присутствовать функция сохранения списка элементов в файл (**D01**).

**F07.** 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 Windows 10. Работоспособность на других выпусках и версиях не гарантируется.

**C02.** На рабочей станции должен быть установлен .NET Framework версии 6.0.

**4.6 Требования к аппаратному обеспечению**

**C03.** Процессор – не менее 2500 МГц.

**C04.** ОЗУ – не менее 2 ГБ.