|  |
| --- |
| **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-5КМ01 | |  | \_\_\_\_\_\_\_\_\_\_ | |  | Ушаков А.В. |
|  | |  |  | |  |  |
|  | |  |  | \_\_\_ \_\_\_\_\_\_ 2022 г. | | |
|  | |  |  | |  |  |
| Проверил: | доцент, к.т.н. |  | \_\_\_\_\_\_\_\_\_\_ | |  | Калентьев А. А. |
|  |  |  |  | |  |  |
|  |  |  |  | \_\_\_ \_\_\_\_\_\_ 2022 г. | | |

Томск 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc106106203)

[1 Основная часть 4](#_Toc106106204)

[1.1 UML диаграмма вариантов использования 4](#_Toc106106205)

[1.2 UML диаграмма классов 6](#_Toc106106206)

[1.3 Описание классов, образующих связь типа «общее-частное» 8](#_Toc106106207)

[1.4 Дерево ветвлений Git 9](#_Toc106106208)

[1.5 Тестирование программы 10](#_Toc106106209)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc106106210)

[ПРИЛОЖЕНИЕ А 21](#_Toc106106211)

**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 лабораторной работы:

Обучение разработки 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]. Диаграмма 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.

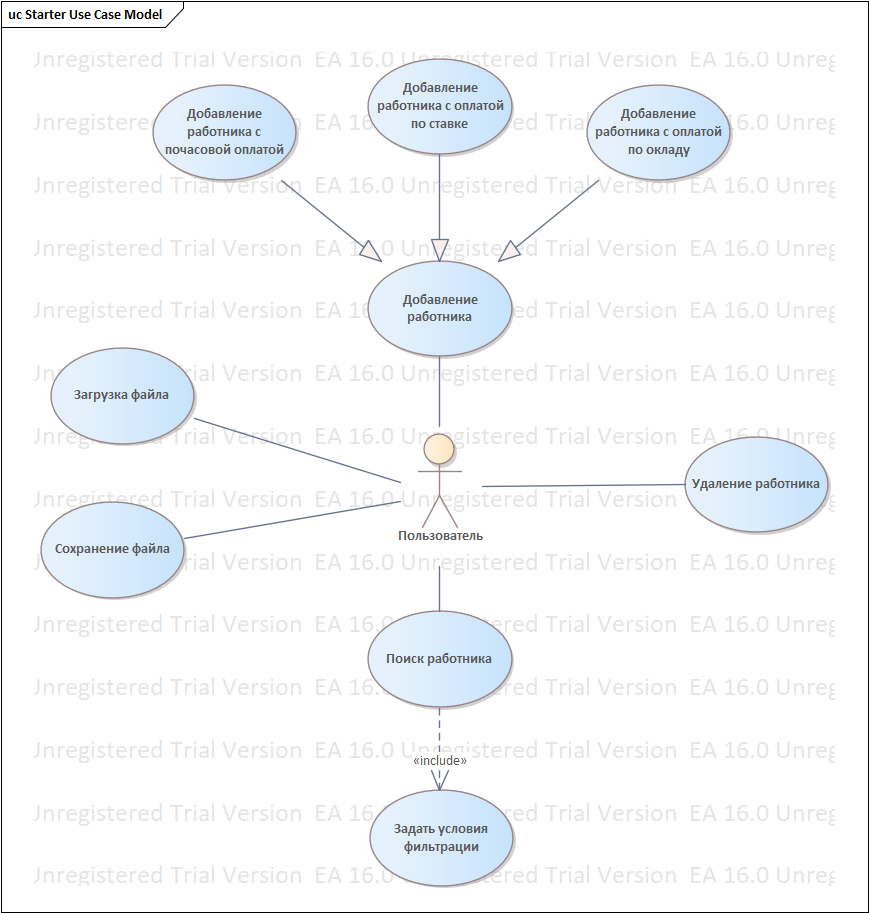


Рисунок 1 – Диаграмма вариантов использования

**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.

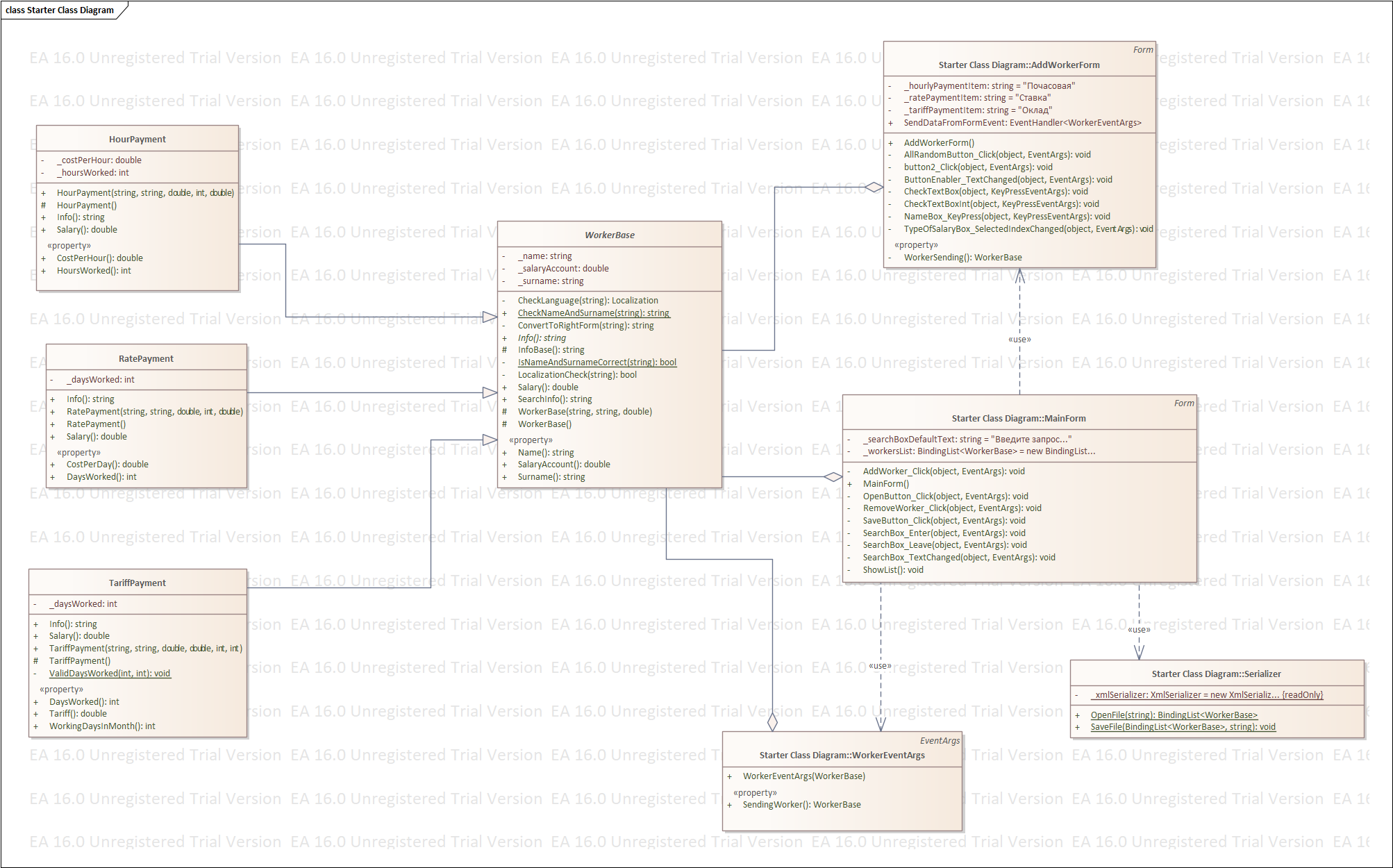


Рисунок 2 – UML диаграмма классов

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

Для классов, образующих связь типа «общее-частное» (наследование, реализация), ниже приведено описание (таблицы 1.1 – 1.4).

Таблица 1.1 – Описание интерфейса ISalaryble

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Интерфейс ISalaryble – интерфейс для начисления ЗП | | |
| Свойства | | |
| +Salary | double | Зарплата |

Таблица 1.2 – Описание класса HourPayment

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| HourlyPayment – Класс для почасовой оплаты | | |
| Свойства | | |
| + HoursWorked | int | Отработанные часы |
| + CostPerHour | double | Стоимость часа |
| + SalaryAccount | double | Зарплатный счет |
| Методы | | |
| + HourPayment(double salaryAccount, int hoursWorked, double costPerHour) |  | Конструктор для создания нового экземпляра почасовой оплаты |

Таблица 1.3 – Описание класса RatePayment

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| RatePayment – Класс для оплаты по ставке | | |
| Свойства | | |
| + DaysWorked | int | Отработанные дни |
| + CostPerDay | double | Стоимость дня |
| + SalaryAccount | double | Зарплатный счет |
| Методы | | |
| + RatePayment(double salaryAccount, int daysWorked, double costPerDay) |  | Конструктор для создания нового экземпляра оплаты по ставке |

Таблица 1.4 – Описание класса TariffPayment

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| TariffPayment – Класс для оплаты по окладу | | |
| Свойства | | |
| + DaysWorked | int | Отработанные дни |
| + WorkingDaysInMonth | int | Количество рабочих дней в месяце |
| + Tariff | double | Оклад |
| + SalaryAccount | double | Зарплата |
| Методы | | |
| + TariffPayment(double salaryAccount ,decimal tariff, int workingDaysInMonth,int daysWorked) |  | Конструктор для создания нового экземпляра оплаты по окладу |
| - ValidateDaysWorked(int daysWorked, int workingDaysInMonth) | void | Проверка ввода количества дней |

**1.4 Дерево ветвлений Git**

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

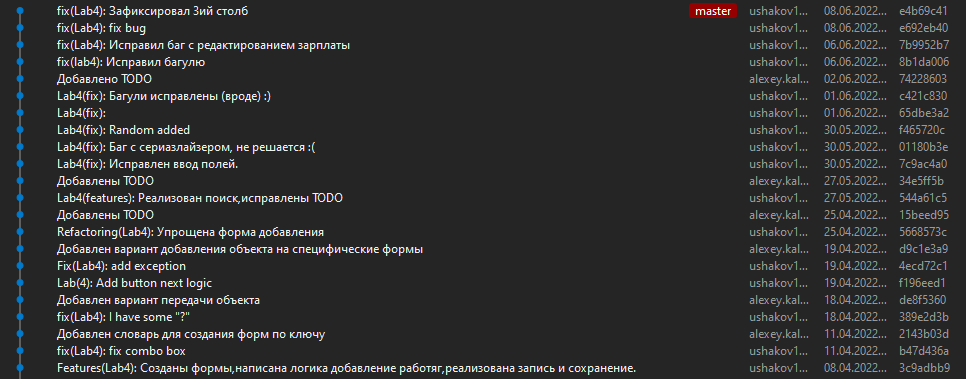


Рисунок 3 – Дерево ветвлений Git

**1.5 Тестирование программы**

Далее приводится процесс функционального тестирования программы.

Графический интерфейс пользователя представлен на рисунке 4.

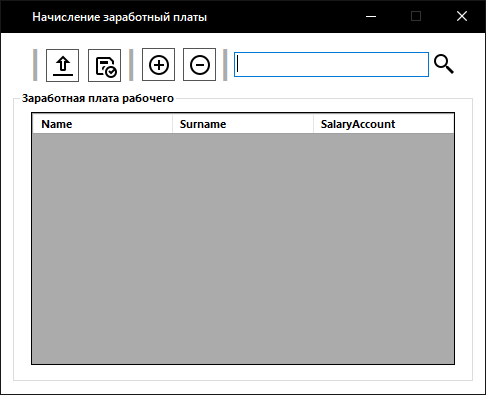


Рисунок 4 – Графический интерфейс пользователя

**1.5.1 Тестовый случай «Добавить элемент»**

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

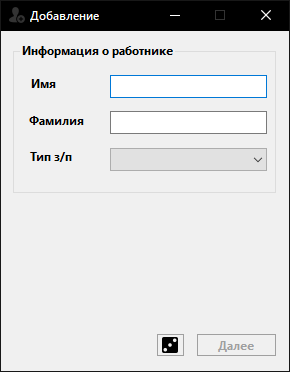


Рисунок 5 – Форма для добавления элемента

Параметры любого из выбранных сотрудников (оклад, сдельный, почасовой) можно ввести, выбрав соответствующий тип сотрудника в выпадающем меню.

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

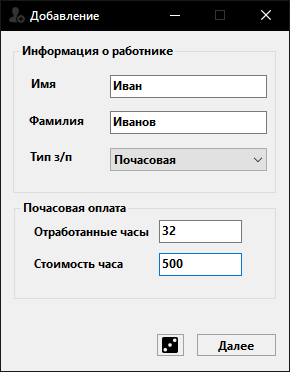


Рисунок 6 – Заполнение полей

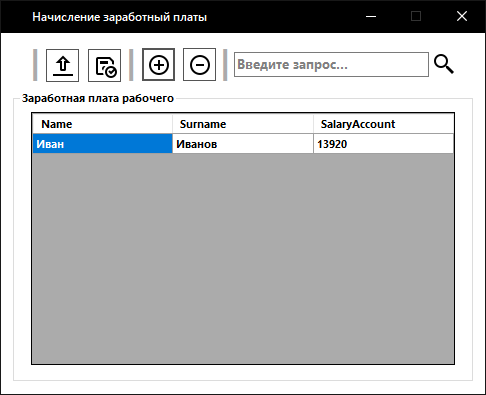


Рисунок 7 – Успешное добавление нового элемента

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

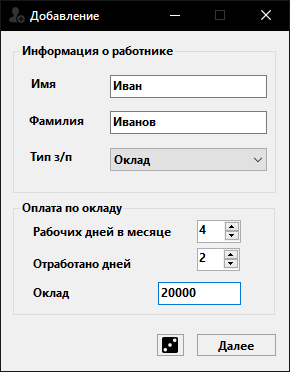


Рисунок 8 – Некорректный ввод (отрицательное число)



Рисунок 9 – Сообщение об ошибке

Подобная обработка предусмотрена для всех параметров сотрудника.

**1.5.2 Тестовый случай «Удалить элемент»**

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

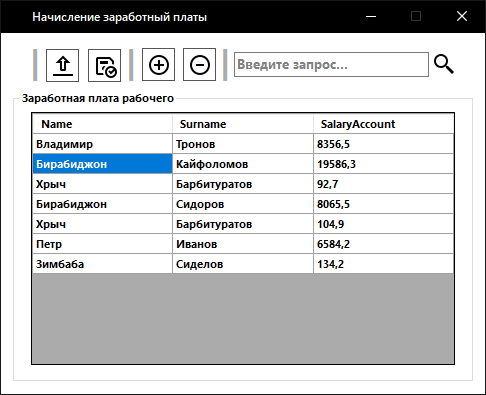


Рисунок 10 – Выбор элемента в таблице

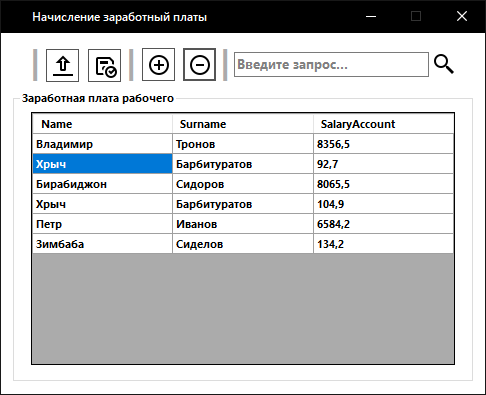


Рисунок 11 – Результат нажатия кнопки «Удалить элемент»

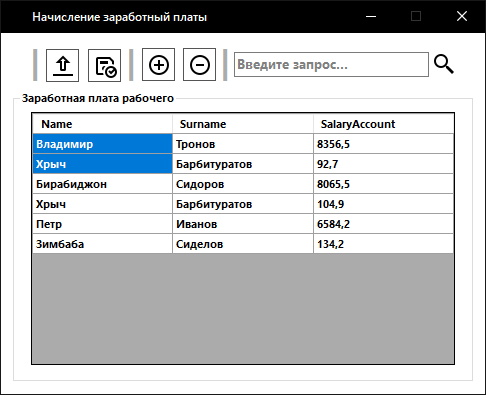


Рисунок 12 – Выбор нескольких элементов для удаления

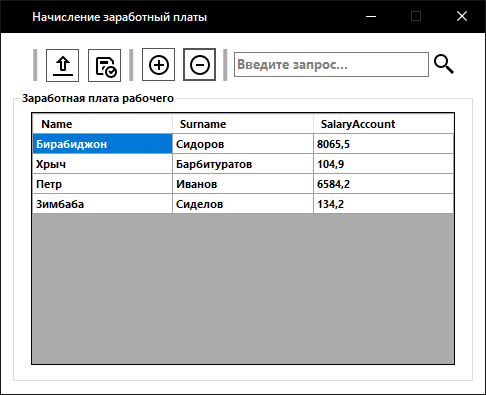


Рисунок 13 – Результат удаления выбранных элементов

**1.5.3 Тестовый случай «Найти элемент»**

Для поиска элементов предусмотрена панель фильтрации сотрудников, имеется строка, подсказывающая куда нужно ввести запрос (рисунок 14)

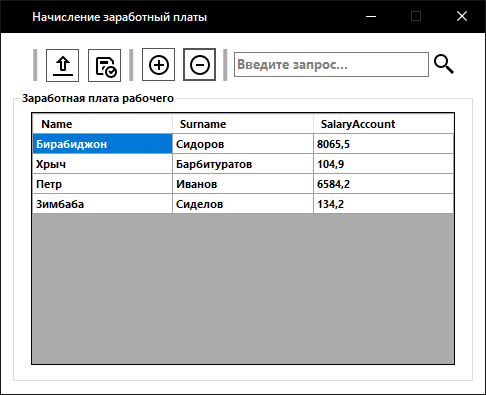


Рисунок 14 – Настройка фильтрации для поиска элементов

Пользователь вводит параметр, по которому бы хотелось сделать запрос. Например – имя (рисунок 15).

Для того, чтобы сбросить фильтр, нужно удалить из поисковой строки свой запрос (рисунок 16).

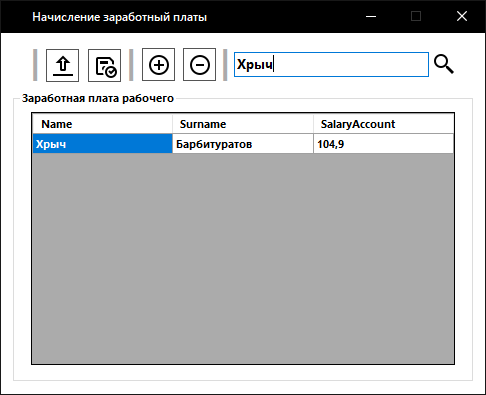


Рисунок 15 – Поиск сотрудника по имени

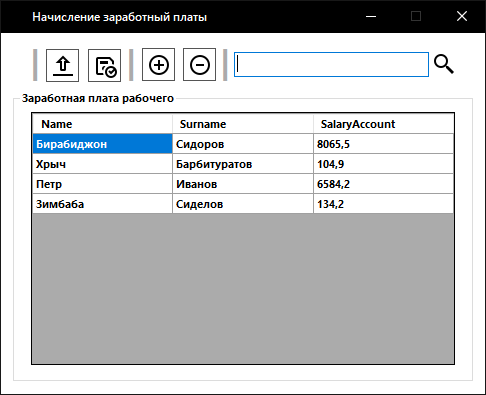


Рисунок 17 – Сброс условий поиска

При вводе зарплаты или фамилии функция работает аналогично.

**1.5.4 Тестовый случай «Сохранить данные»**

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

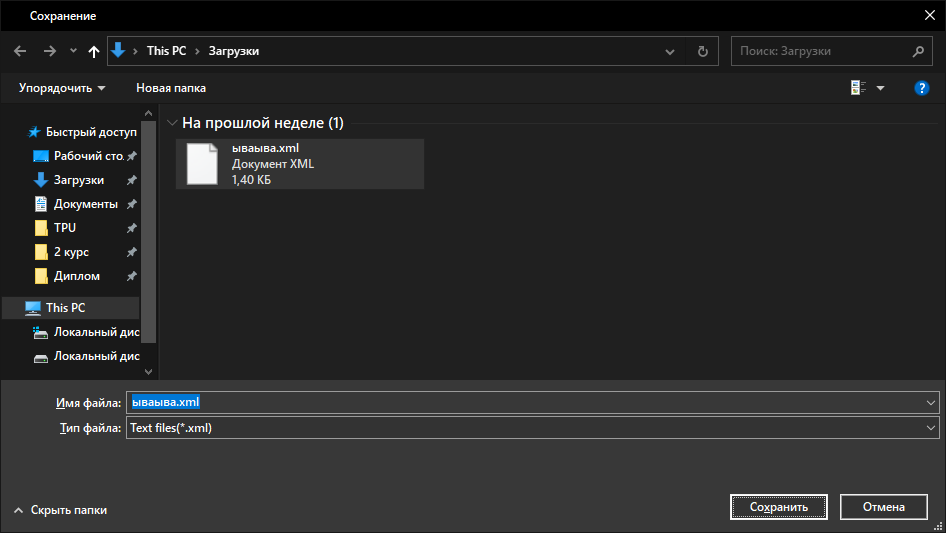


Рисунок 18 – Сохранение файла



Рисунок 19 – Результат сохранения файла

**1.5.5 Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо нажать на соответствующую кнопку  .

Далее откроется системный диалог загрузки файла (рисунок 20).

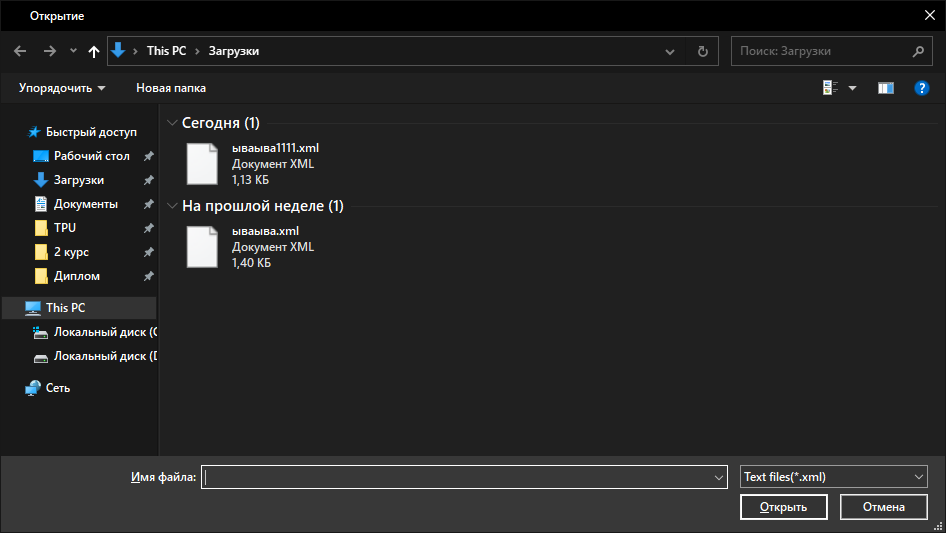


Рисунок 20 – Выбор файла для загрузки

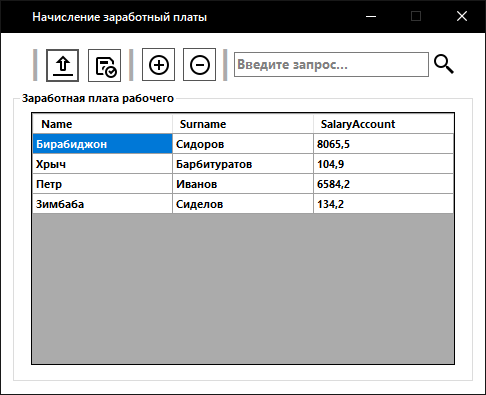


Рисунок 21 – Результат загрузки данных

В случае, если XSD схема файла не соответствует установленному формату, появится соответствующее сообщение (рисунок 22).

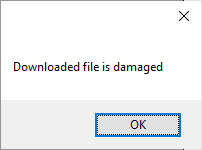


Рисунок 22 – Загрузка повреждённого файла

**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.

**ПРИЛОЖЕНИЕ А**

**Техническое задание на создание автоматизированной системы**

Программа для начисления заработной платы

Разработчик: студент гр. О-5КМ01 НИ ТПУ Ушаков А.В.

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

Томск 2022

**1 Общие сведения**

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

Полное наименование: «Программное обеспечение для начисления заработной платы сотрудникам различными способами: почасовая оплата, оплата по окладу и оплата по ставке».

Краткое наименование: «Начисление 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-5КМ01 НИ ТПУ Ушаков А.В.

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

Начало работ: 22 марта 2022 г.

Окончание работ: 9 июня 2022 г.

**2 Назначение и цели создания системы**

**2.1 Назначение системы**

Разрабатываемая система предназначена для начисления ЗП сотрудникам различными способами: почасовая оплата, оплата по окладу и оплата по ставке.

**2.2 Цели создания системы**

Разрабатываемая система создается с целью уменьшения трудозатрат заказчика, а также для автоматизации процесса начисления ЗП.

**3 Характеристика объектов автоматизации**

Процесс расчета ЗП выполняется в соответствии с требованиями законодательства и не является индивидуальным для каждого работника, а меняется только в зависимости от способа начисления. Исходя из этого процесс начисления ЗП может быть автоматизирован.

**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 заработной плате сотрудников должны храниться в XML-файле с расширением \*.xml.

**D01.01.** Формат XML-файла должен соответствовать следующей XSD-схеме:



**4.3 Функциональные требования**

**F01.** Должна быть реализована возможность начисления ЗП следующими способами: почасовая оплата, оплата по окладу и оплата по ставке

**F01.1.** Для начисления ЗП по почасовой оплате должны задаваться количество отработанных часов и стоимость одного часа.

**F01.2.** Для начисления ЗП по окладу должны задаваться количество рабочих дней в месяце, величина оклада и количество отработанных дней.

**F01.3** Для начисления ЗП по ставке должны задаваться количество дней, размер ставки.

**F02.** Расчет ЗП должен производиться в соответствии с нормативными документами.

**D01.** Должна быть возможность сохранения данных в формате XML.

**D02.** Должна быть возможность загрузки данных в формате XML.

**S01.** Установка и функционирование Системы не должны приводить к необходимости изменения настроек антивирусного программного обеспечения.

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

**U02.** Должна быть реализована система обработки ошибок.

**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 Pro 20H1, 20H2, 21H1, 21H2. Работоспособность на других выпусках и версиях не гарантируется.

**C02.** На рабочей станции должен быть установлен .NET Runtime 5.0.17.

**4.6 Требования к аппаратному обеспечению**

**C03.** Для работы системы необходим ПК/ноутбук.

**C04.** Не менее 1 гигагерц (GHz).

**C05.** ОЗУ – не менее 1 ГБ для 32-разрядной системы и 2 ГБ для 64-разрядной.

**C06.** Накопитель – 20 ГБ свободного пространства для 64-разрядных и 16 ГБ свободного пространства для 32-разрядных систем.

**C07.** Разрешение экрана – минимум 800 х 600 пикселей.

**C08.** Видеоадаптер DirectX 9 или более поздняя версия с драйвером WDDM 1.0