|  |
| --- |
| **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 | |  | \_\_\_\_\_\_\_\_\_\_ | |  | Иманкулов Р.Б. |
|  | |  |  | |  |  |
|  | |  |  |  | | |
|  | |  |  | |  |  |
| Отчёт принял | доцент, к.т.н. |  | \_\_\_\_\_\_\_\_\_\_ | |  | Калентьев А. А. |
|  |  |  |  | |  |  |
|  |  |  |  | \_\_\_ \_\_\_\_\_\_ | | |

Томск 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc74956671)

[1 Основная часть 4](#_Toc74956672)

[1.1 UML диаграмма вариантов использования 4](#_Toc74956673)

[1.2 UML диаграмма классов 5](#_Toc74956674)

[1.3 Описание классов, образующих связь типа «общее-частное» 8](#_Toc74956675)

[1.4 Дерево ветвлений Git 9](#_Toc74956676)

[1.5 Тестирование программы 10](#_Toc74956677)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc74956678)

[ПРИЛОЖЕНИЕ А 23](#_Toc74956679)

**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.

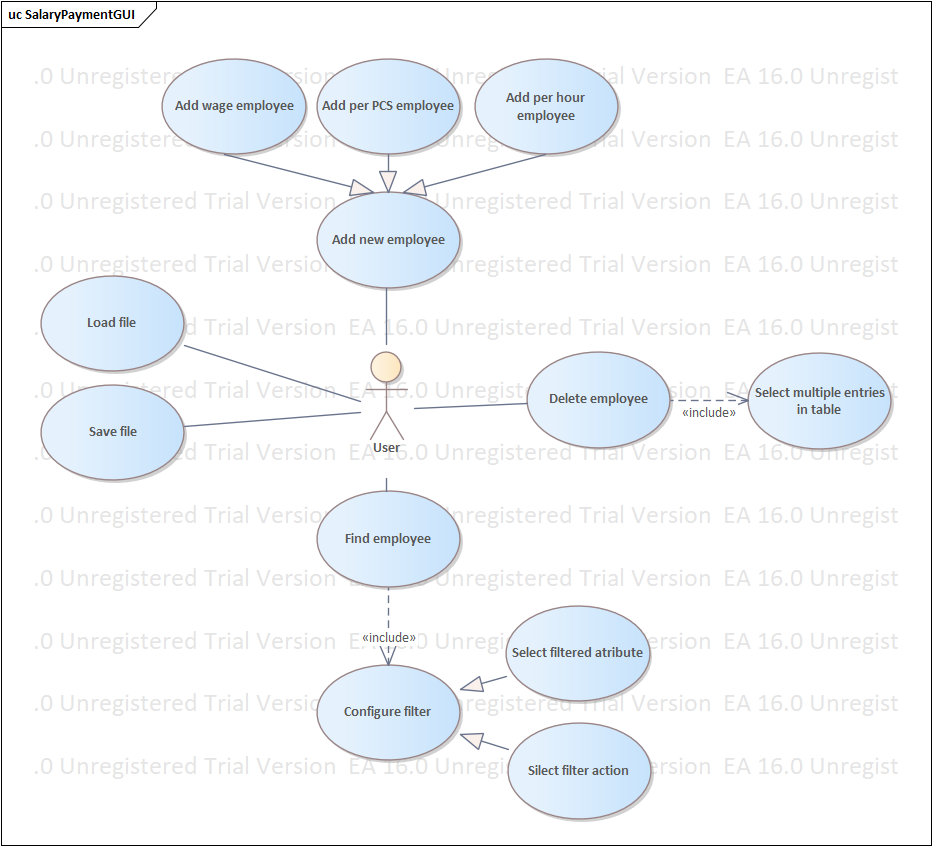


Рисунок 1 – Диаграмма вариантов использования

**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.

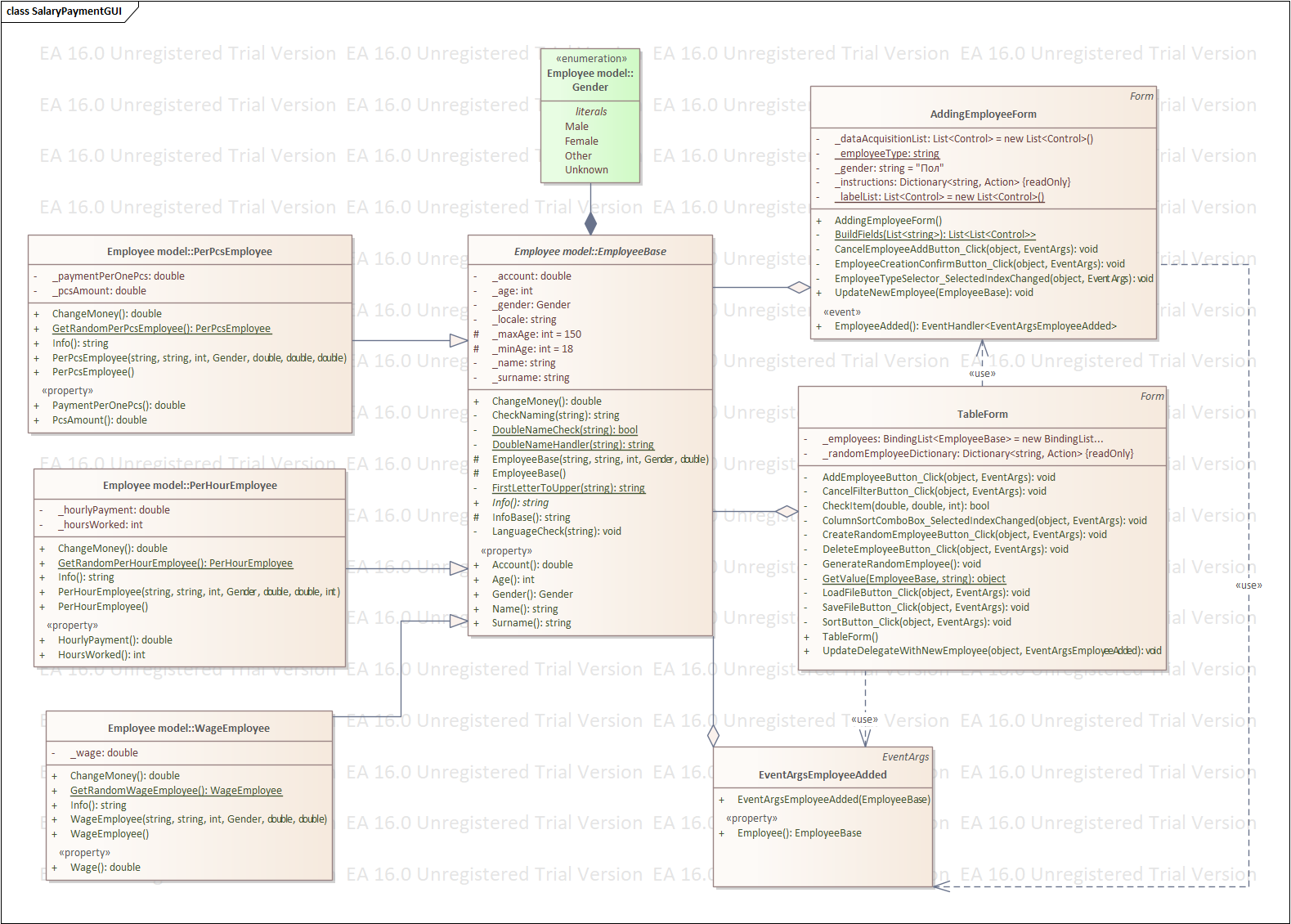


Рисунок 2 – UML диаграмма классов

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

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

Таблица 1 – Описание класса *EmployeeBase*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *EmployeeBase* – абстрактный базовый класс для работников | | |
| Поля | | |
| -\_account | double | Аккаунт работника |
| -\_age | int | Возраст работника |
| -\_gender | Gender | Пол работника |
| -\_locale | string | Локаль имени работника |
| -\_maxAge | int | Максимальный возраст, принимаемый программой |
| -\_minAge | int | Минимальный возраст, принимаемый программой |
| -\_name | string | Имя работника |
| -\_surname | string | Фамилия работника |
| Свойства | | |
| +Account | double | Аккаунт работника |
| +Age | int | Возраст работника |
| +Gender | Gender | Пол работника |
| +Info | string | Показывает информацию об объекте |
| +Name | string | Имя работника |
| +Surname | string | Фамилия работника |
| Методы | | |
| + ChangeMoney | double | Метод, добавляющий работнику зарплату |
| - CheckNaming (string) | string | Внутренний метод проверяющий локаль имени и фамилии, и прочие проверки имени |
| - DoubleNameCheck (string) | bool | Внутренний метод, проверяющий двойное имя |
| - DoubleNameHandler (string) | string | Внутренний метод, приводящие двойном имени к форме описанной в FirstLetterToUpper |
| - FirstLetterToUpper (string) | string | Внутренний метод, переводящий первые буквы в имени в заглавные, а остальные в строчные |
| - LanguageCheck (string) | void | Внутренний метод проверяющий находятся ли имя и фамилия в одной локали |

В таблицах 2–4 приведены описания классов WageEmployee, PerHourEmployee и PerPcsEmployee, которые наследуются от *EmployeeBase*.

Таблица 2 – Описание класса WageEmployee

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс WageEmployee – работник, работающий за оклад | | |
| Поля | | |
| –\_wage | double | Оклад работника |
| Свойства | | |
| +Info | string | Показывает информацию об объекте |
| +Wage | double | Оклад работника |
| Методы | | |
| +ChangeMoney | double | Метод, добавляющий работнику зарплату |
| +GetRandomWageEmployee | WageEmployee | Метод возвращает работника со случайными значениями в полях |

Таблица 3 – Описание класса PerHourEmployee

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс PerHourEmployee – работник, работающий по часам | | |
| Поля | | |
| -\_hourlyPayment | double | Почасовой тариф работника |
| -\_hoursWorked | int | Количество часов, отработанных работником |
| Свойства | | |
| +HourlyPayment | double | Почасовой тариф работника |
| +HoursWorked | int | Количество часов, отработанных работником |
| +Info | string | Показывает информацию об объекте |
| Методы | | |
| +ChangeMoney | double | Метод, добавляющий работнику зарплату |
| +GetRandomPerHourEmployee | PerHourEmployee | Метод возвращает работника со случайными значениями в полях |

Таблица 4 – Описание класса PerPcsEmployee

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс PerPcsEmployee – работник, работающий сдельно | | |
| Поля | | |
| -\_paymentPerOnePcs | double | Поштучный тариф работника |
| -\_pcsAmount | double | Сколько штук произвел работник |
| Свойства | | |
| +Info | string | Показывает информацию об объекте |
| +PaymentPerOnePcs | double | Поштучный тариф работника |
| +PcsAmount | double | Сколько штук произвел работник |
| Методы | | |
| +ChangeMoney | double | Метод, добавляющий работнику зарплату |
| +GetRandomPerPcsEmployee | PerPcsEmployee | Метод возвращает работника со случайными значениями в полях |

**1.4 Дерево ветвлений Git**

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

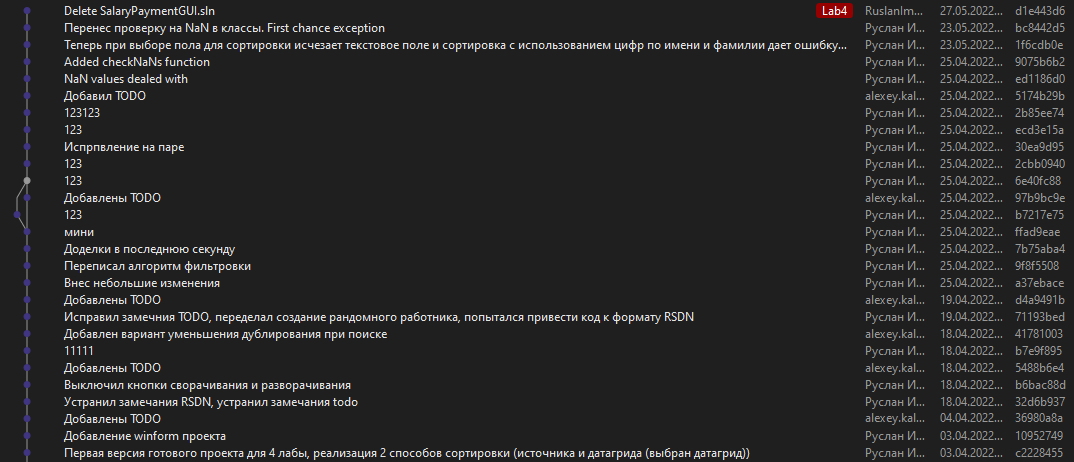


Рисунок 3 – Дерево ветвлений Git

**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).

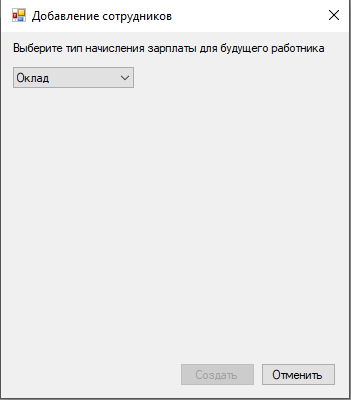


Рисунок 5 – Форма для добавления элемента

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).

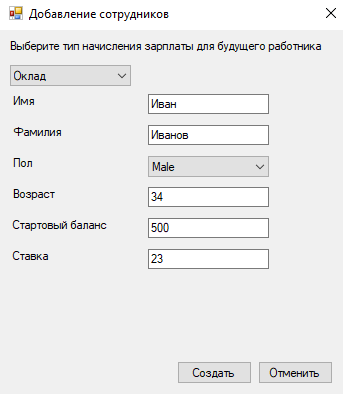


Рисунок 6 – Заполнение полей

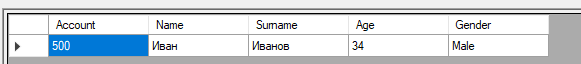


Рисунок 7 – Успешное добавление нового элемента

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.

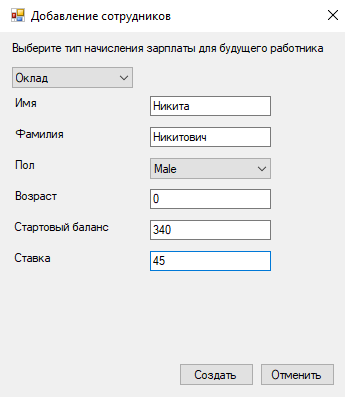


Рисунок 8 – Некорректный ввод (отрицательное число)

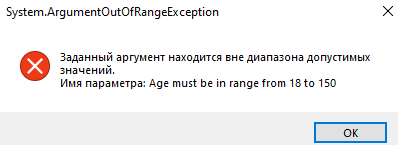


Рисунок 9 – Сообщение об ошибке

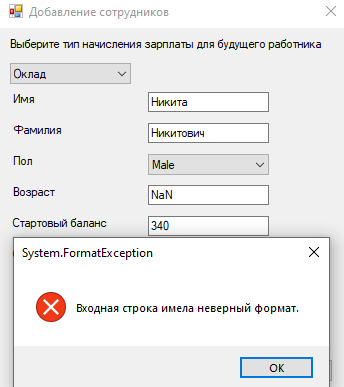


Рисунок 10 – Обработка ошибки NaN

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 11 и 12).

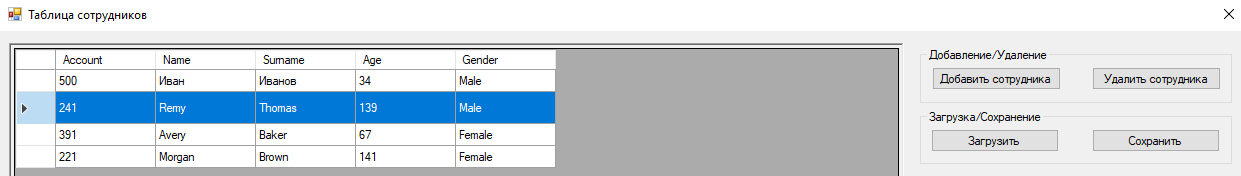


Рисунок 11 – Выбор элемента в таблице

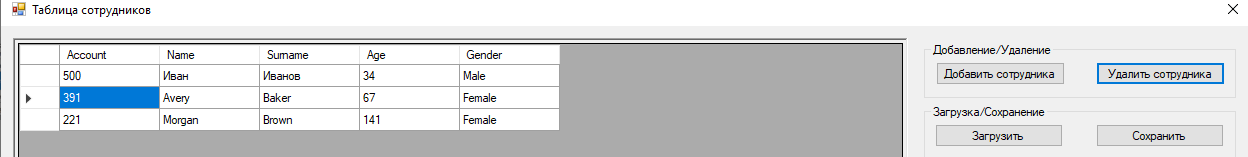


Рисунок 12 – Результат нажатия кнопки «Удалить элемент»

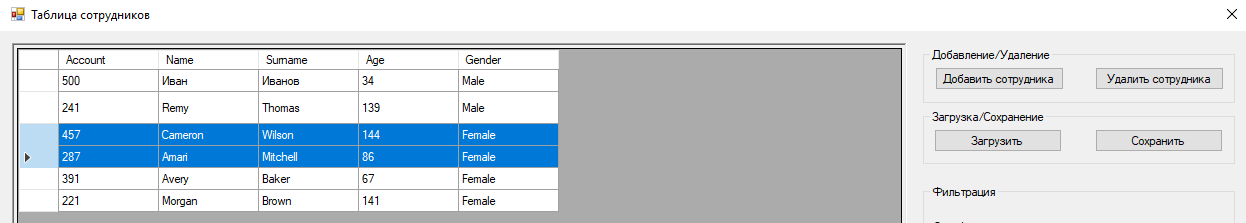


Рисунок 13 – Выбор нескольких элементов для удаления

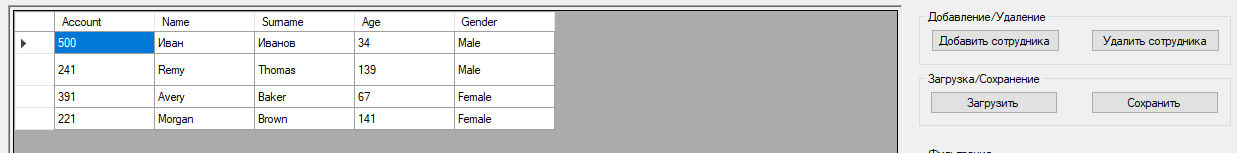


Рисунок 14 – Результат удаления выбранных элементов

**1.5.3 Тестовый случай «Найти элемент»**

Для поиска элементов предусмотрена панель фильтрации (рисунок 15)

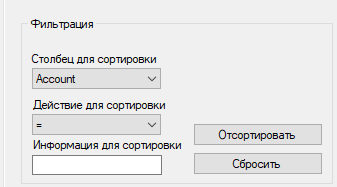


Рисунок 15 – Настройка фильтрации для поиска элементов

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

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

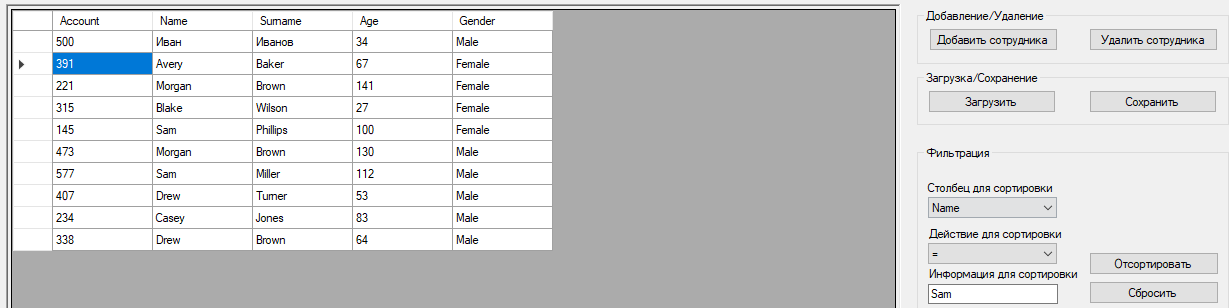


Рисунок 16 – Поиск сотрудника по имени

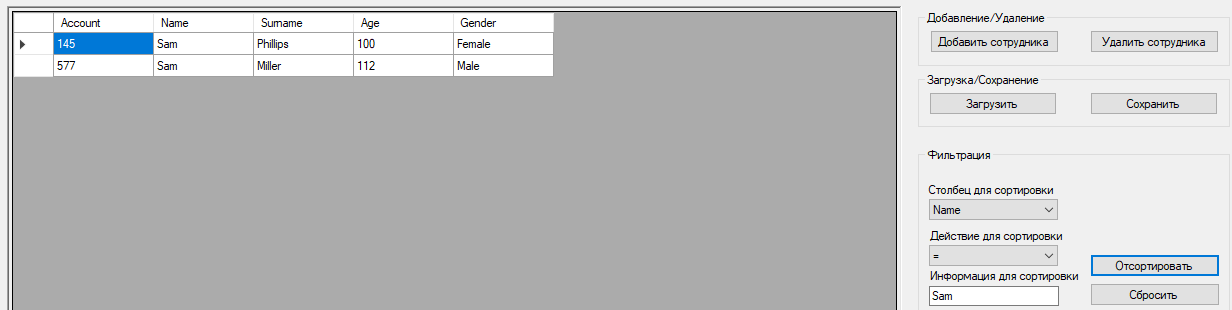


Рисунок 17 – Результат поиска сотрудника

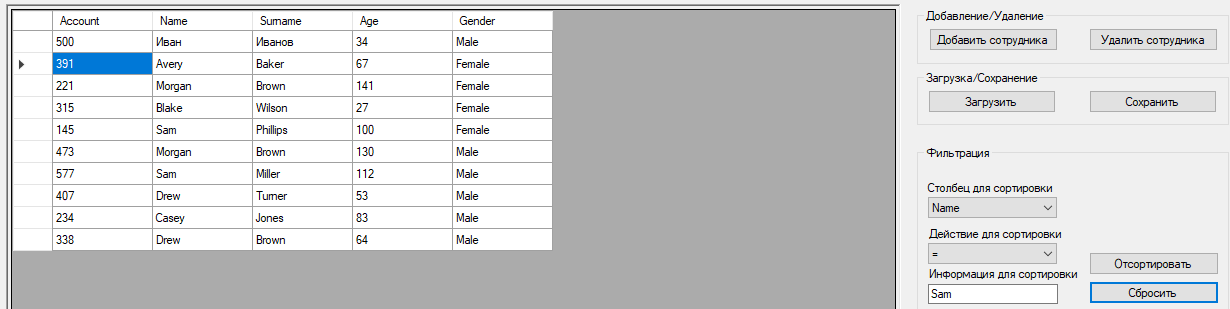


Рисунок 18 – Сброс условий поиска

Фильтрация осуществляется последовательно, это значит, что можно менять параметры фильтрации и применять их для уже отфильтрованного списка (рисунок 19-22)

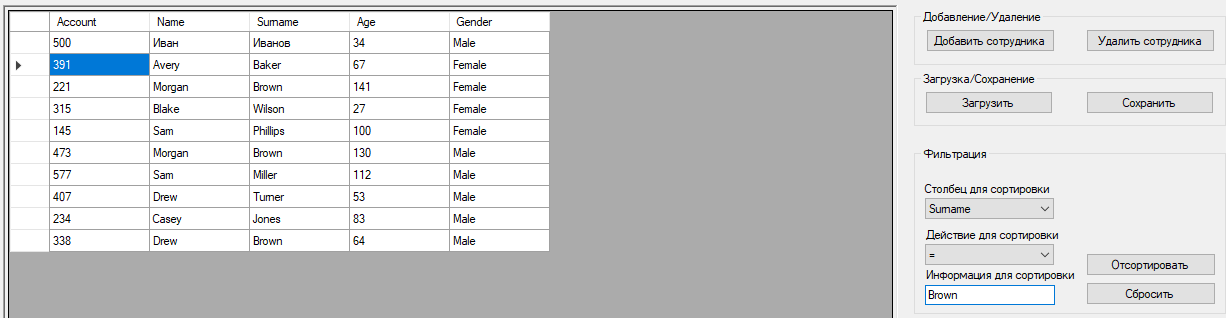


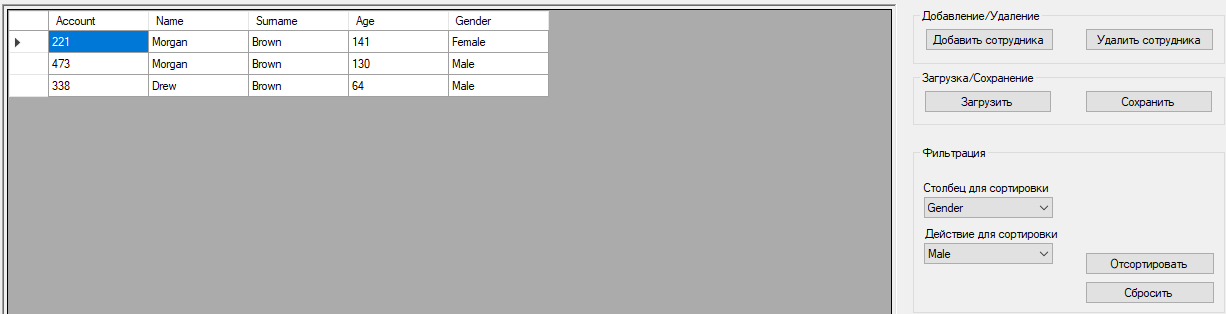
Рисунок 19 – Фильтрация сотрудников по фамилии Brown

Рисунок 20 – Фильтрация сотрудников по полу

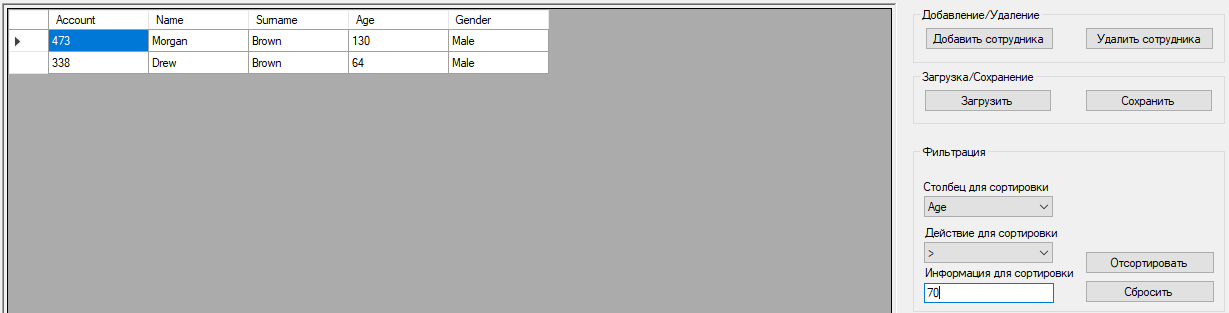


Рисунок 21 – Фильтрация сотрудников по возрасту (более 70)



Рисунок 22 – Конечный результат фильтрации

**1.5.4 Тестовый случай «Сохранить данные»**

Для сохранения данных в таблице необходимо нажать на кнопку «Сохранить» (рисунок 23). Откроется системный диалог сохранения файла, где пользователь выбирает директорию и указывает имя файла (рисунок 24).

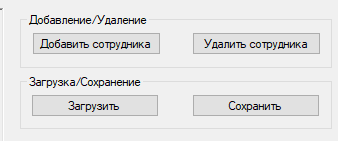


Рисунок 23 – Сохранение данных

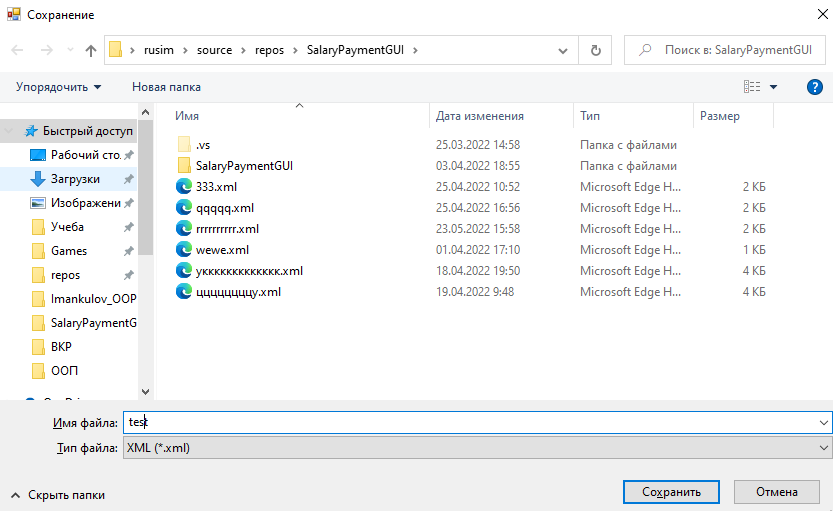


Рисунок 24 – Сохранение файла

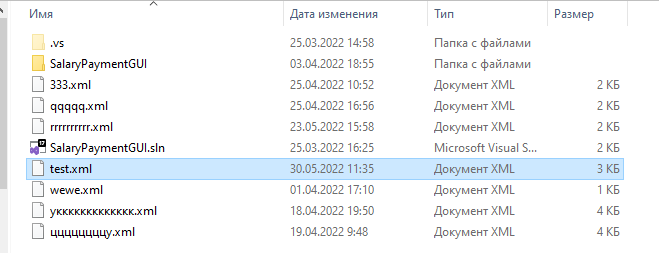


Рисунок 25 – Результат сохранения файла

**1.5.5 Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо нажать на соответствующую кнопку (рисунок 26).

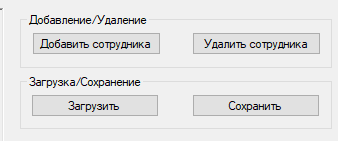


Рисунок 26 – Загрузить данные в таблицу

Далее откроется системный диалог загрузки файла (рисунок 27).

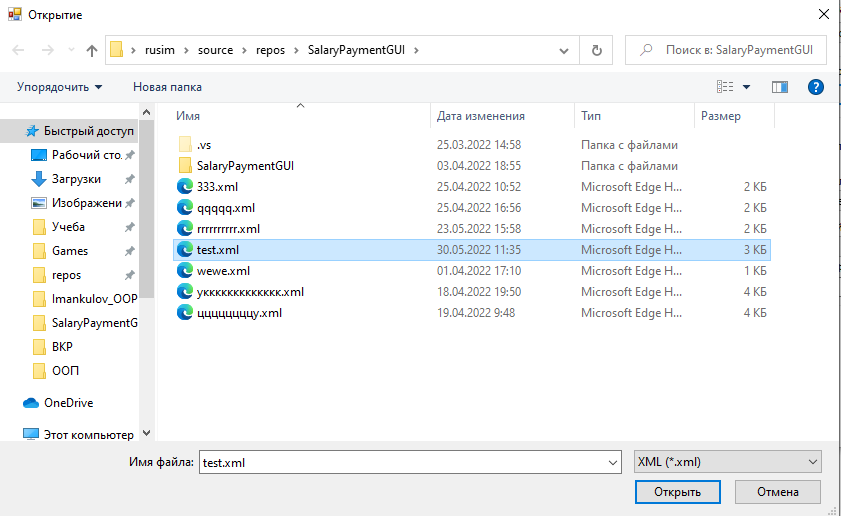


Рисунок 27 – Выбор файла для загрузки

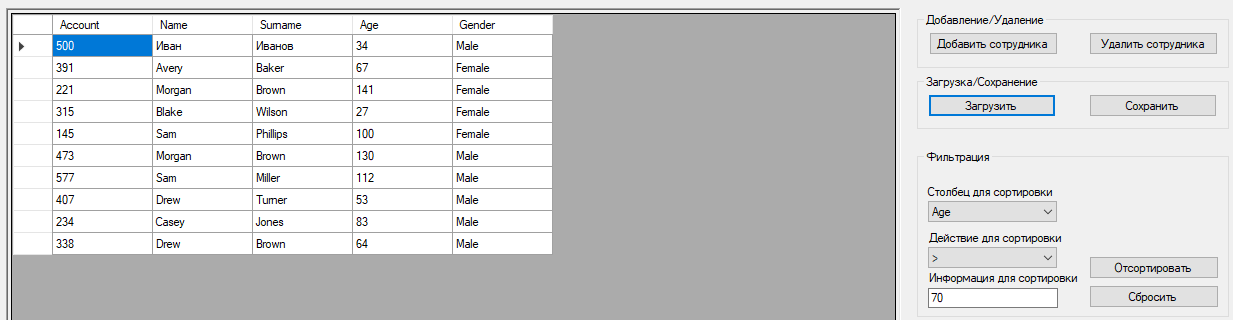


Рисунок 28 – Результат загрузки данных

В случае, если XSD схема файла не соответствует установленному формату, появится соответствующее сообщение (рисунок 28).

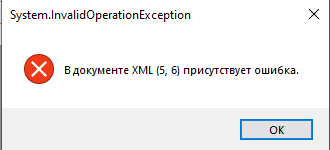


Рисунок 28 – Загрузка повреждённого файла

**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 просмотра списка работников

Разработчик: студент гр. О-5КМ01 НИ Иманкулов Р.Б.

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: «Программа для просмотра списка работников».

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 Иманкулов Р.Б.

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

Начало работ: 3 апреля 2022 г.

Окончание работ: 2 июня 2022 г.

**2 Назначение и цели создания системы**

**2.1 Назначение системы**

Система предназначена для просмотра списка сотрудников.

**2.2 Цели создания системы**

Система создаётся в целях сокращения трудозатрат сотрудников HR отдела по поиску и мониторингу сотрудников.

**3 Характеристика объектов автоматизации**

Поиск и мониторинг сотрудников в крупной компании утомительное и трудозатратное дело. Разработанное решение поможет снизить трудозатраты работников HR отделов.

**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 элементов электрических схем должны храниться в XML-файле с расширением \*.xml.

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

<ArrayOfEmployeeBase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<EmployeeBase xsi:type="[EMPLOYEE CLASS]">

<Account>[EMPLOYEE ACCOUNT]</Account>

<Name>>[EMPLOYEE NAME]</Name>

<Surname>[EMPLOYEE SURNAME]</Surname>

<Age>[EMPLOYEE AGE]</Age>

<Gender>[EMPLOYEE GENDER]</Gender>

/[DATA DEPENDANT ON EMPLOYEE CLASS]

<HourlyPayment>[EMPLOYEE PARAMETER]</HourlyPayment>

<HoursWorked>[EMPLOYEE PARAMETER]</HoursWorked>

[DATA DEPENDANT ON EMPLOYEE CLASS]/

</EmployeeBase>

<EmployeeBase xsi:type="PerHourEmployee">

<Account>529</Account>

<Name>Emerson</Name>

<Surname>Moore</Surname>

<Age>115</Age>

<Gender>Male</Gender>

<HourlyPayment>39</HourlyPayment>

<HoursWorked>3</HoursWorked>

</EmployeeBase>

</ArrayOfEmployeeBase>

**4.3 Функциональные требования**

**F01.** В системе должен быть реализован список элементов электрических схем.

**F01.01.** Каждый элемент должен иметь следующие обязательные параметры:

* имя;
* фамилия;
* аккаунт;
* возраст;
* пол.

**F01.02.** В зависимости от типа элемента необходимо заполнить следующие параметры:

Сотрудник с окладной системой оплаты труда:

* оклад.

Сотрудник с почасовой системой оплаты труда:

* количество отработанных часов;
* почасовая ставка.

Сотрудник со сдельной системой оплаты труда:

* количество выполненных работ;
* оплата за единицу выполненной работы.

**F02.** В системе должна присутствовать функция добавления элементов в список.

**F03.** В системе должна присутствовать функция удаления элементов из списка.

**F04.** В системе должна присутствовать функция поиска элементов по параметрам, указанным в **F01.01**.

**F05.** В системе должна присутствовать функция сохранения списка элементов в файл (**D01**).

**F06.** 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 Pro 20H2. Работоспособность на других выпусках и версиях не гарантируется.

**C02.** На рабочей станции должен быть установлен .NET Framework версии 4.7.2.

**4.6 Требования к аппаратному обеспечению**

**C03.** Процессор – Intel CPU: все процессоры Intel Core i3/i5/i7/i9-Nxxx от 6 поколения, Intel W-12xx/W-108xx, Intel Xeon (SP 32xx, 42xx, 52xx, 62xx и 82xx), Atom (J4xxx/J5xxx and N4xxx/N5xxx), Celeron и Pentium.

**C04.** ОЗУ – не менее 1 ГБ для 32-разрядной системы и 2 ГБ для 64-разрядной.

**C05.** Накопитель – 20 ГБ свободного пространства для 64-разрядных и 16 ГБ свободного пространства для 32-разрядных систем.

**C06.** Разрешение экрана – минимум 800 х 600 пикселей