|  |
| --- |
| **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 | |  | \_\_\_\_\_\_\_\_\_\_ | |  | Мизев А.В. |
|  | |  |  | |  |  |
|  | |  |  |  | | |
|  | |  |  | |  |  |
| Отчёт принял | доцент, к.т.н. |  | \_\_\_\_\_\_\_\_\_\_ | |  | Калентьев А. А. |
|  |  |  |  | |  |  |
|  |  |  |  |  | | |

Томск 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc120365068)

[1 Основная часть 4](#_Toc120365069)

[1.1 UML диаграмма вариантов использования 4](#_Toc120365070)

[1.2 UML диаграмма классов 6](#_Toc120365071)

[1.3 Описание классов, образующих связь типа «общее-частное» 8](#_Toc120365072)

[1.4 Дерево ветвлений Git 9](#_Toc120365073)

[1.5 Тестирование программы 9](#_Toc120365074)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc120365075)

**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.

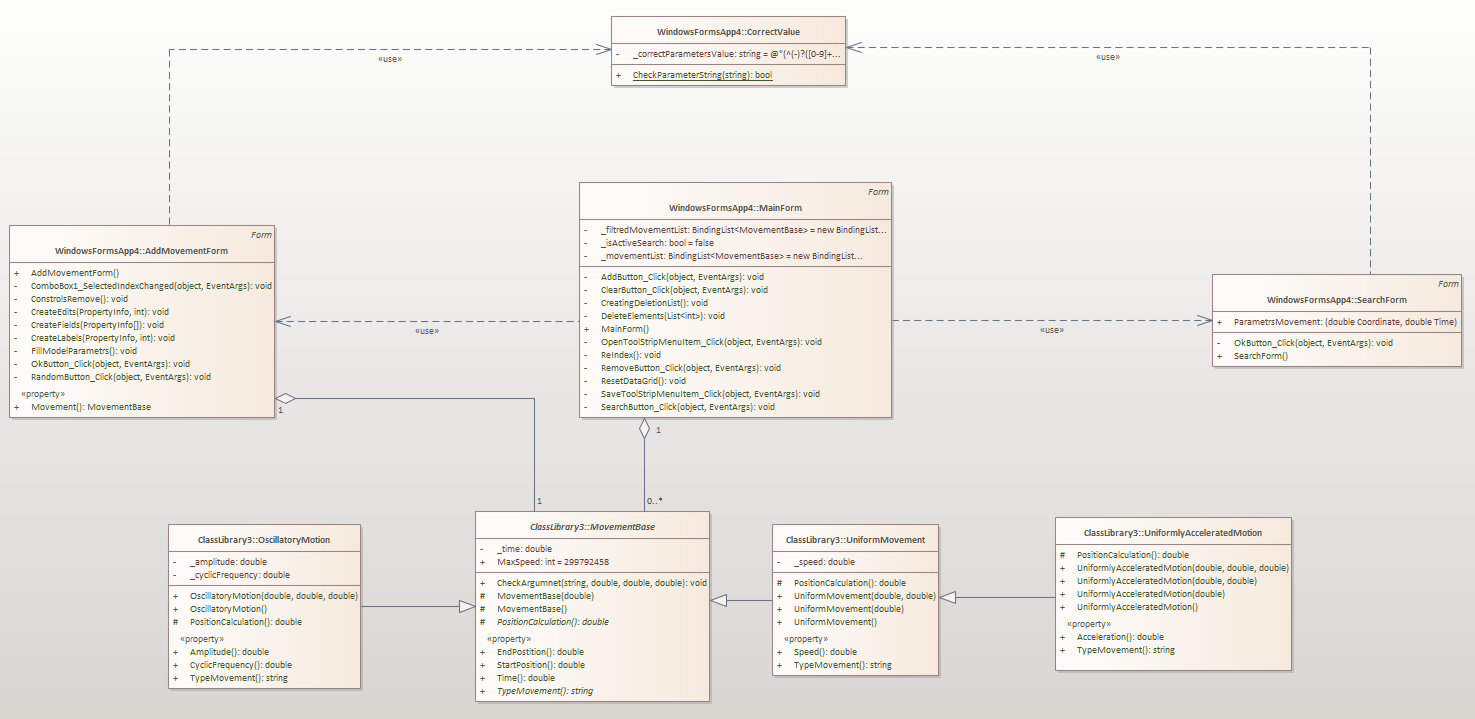


Рисунок 2 – UML диаграмма классов

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

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

Таблица 1 – Описание класса *MovementBase*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *MovementBase* – абстрактный базовый класс для движений | | |
| Поля | | |
| –\_time | double | Время движения |
| +MaxSpeed | int | Максимальная физически возможная скорость |
| Свойства | | |
| + StartPosition | double | Начальная координата |
| + EndPosition | double | Конечная координата |
| + Time | double | Время движения |
| + TypeMovement | string | Тип движения |
| Методы | | |
| – CheckArgument | void | Метод проверки корректности введения параметра |
| # PositionCalculation | double | Метод вычисления конечной координаты |

В таблице 2 приведены описания классов *UniformMovement*, который наследуются от *MovementBase*.

Таблица 2 – Описание класса *UniformMovement*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *UniformMovement* – Класс скидки по купону | | |
| Поля | | |
| –\_speed | double | Скорость движения |
| Свойства | | |
| + Speed | double | Скорость движения |
| + TypeMovement | string | Тип движения |
| Методы | | |
| # PositionCalculation | double | Метод вычисления конечной координаты |

В таблице 3 приведены описания классов *UniformlyAcceleratedMotion*, которые наследуются от *UniformMovement*.

Таблица 3 – Описание класса *UniformlyAcceleratedMotion*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *UniformlyAcceleratedMotion* – Класс равноускоренного движения | | |
| Свойства | | |
| + Acceleration | double | Ускорение движения |
| + TypeMovement | string | Тип движения |
| Методы | | |
| # PositionCalculation | double | Метод вычисления конечной координаты |

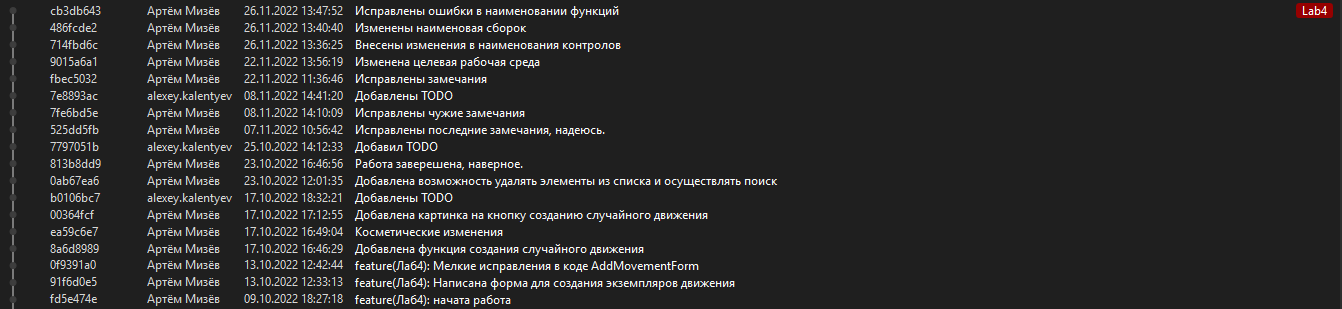
В таблице 4 приведены описания классов *OscillatoryMotion*, которые наследуются от *MovementBase*.

Таблица 4 – Описание класса *OscillatoryMotion*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | | Описание |
| Описание класса | | | |
| Класс *OscillatoryMotion* – Класс колебательного | | | |
| Поля | | | |
| –\_amplitude | double | Амплитуда колебаний | |
| –\_cyclicFrequency | double | Циклическая частота | |
| Свойства | | | |
| + Amplitude | double | | Амплитуда колебаний |
| +\_CyclicFrequency | double | | Циклическая частота |
| + TypeMovement | string | | Тип движения |
| Методы | | | |
| # PositionCalculation | double | | Метод вычисления конечной координаты |

**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.

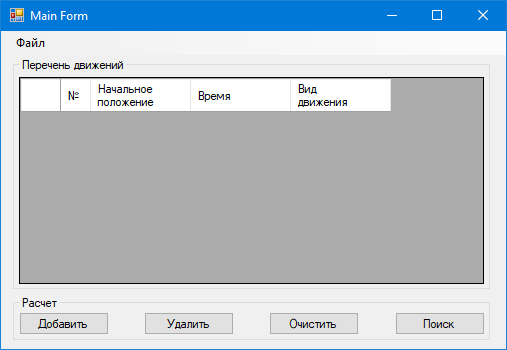


Рисунок 4 – Графический интерфейс пользователя

**1.5.1 Тестовый случай «Добавить движение»**

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

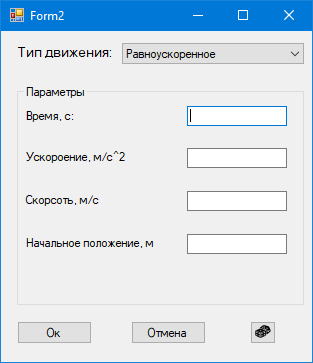


Рисунок 5 – Форма для добавления элемента

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

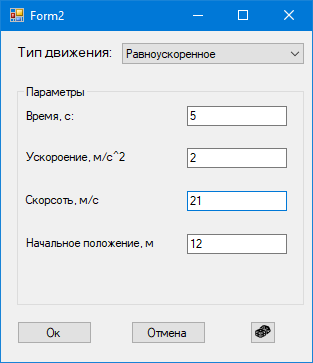


Рисунок 6 – Заполнение полей

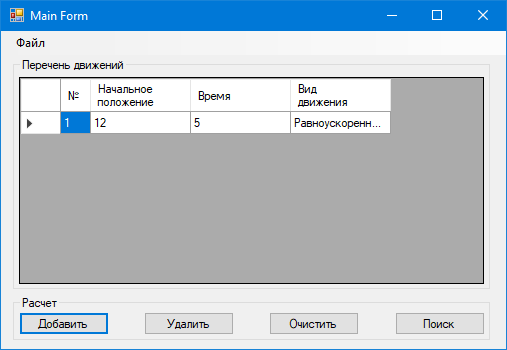


Рисунок 7 – Успешное добавление нового элемента

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

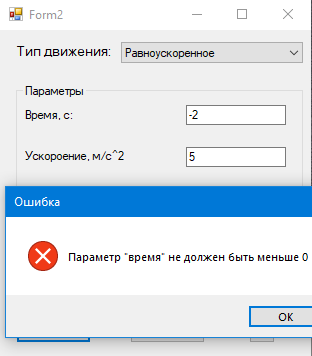


Рисунок 8 – Некорректный ввод (отрицательное число)

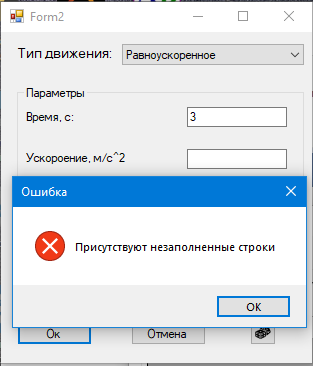


Рисунок 9 – Обработка ошибки

Подобная обработка предусмотрена для всех параметров движения.

**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).

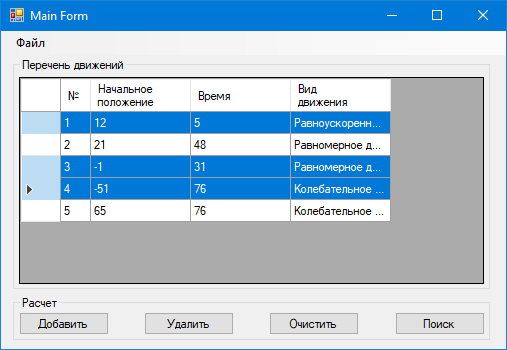


Рисунок 10 – Выбор элемента в таблице

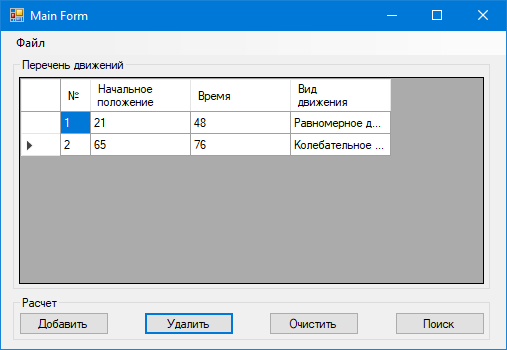


Рисунок 11 – Результат нажатия кнопки «Удалить»

Так же присутствует возможно очистить весь список сразу с помощью кнопки «Очистить».

**1.5.3 Тестовый случай «Поиск»**

Для поиска элементов предусмотрена панель параметров поиска (рисунок 12)

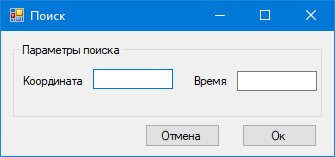


Рисунок 12 – Настройка параметров для поиска элементов

Пользователь выводит значения параметров движения и нажимает кнопку «Ок» (рисунки 13 и 14). Для того, чтобы сбросить фильтр достаточна нажать на туже кнопку, что и для поиска, для удобства пользователя на ней меняется надпись на “Сброс” (рисунок 14).

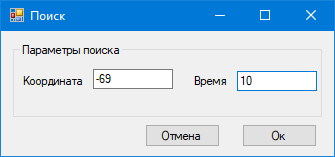


Рисунок 13 – Поиск движения по параметрам

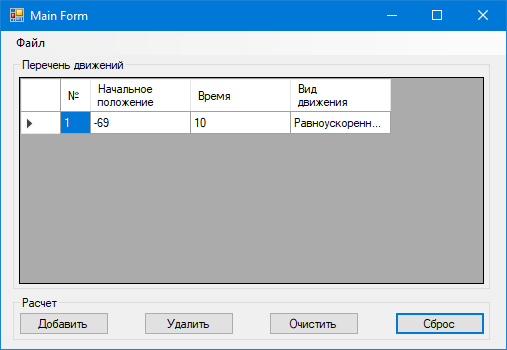


Рисунок 14 – Результат поиска

**1.5.4 Тестовый случай «Сохранить данные»**

Для сохранения данных в таблице необходимо в меню выбрать опцию «Сохранить». Откроется системный диалог сохранения файла, где пользователь выбирает директорию и указывает имя файла (рисунок 15).

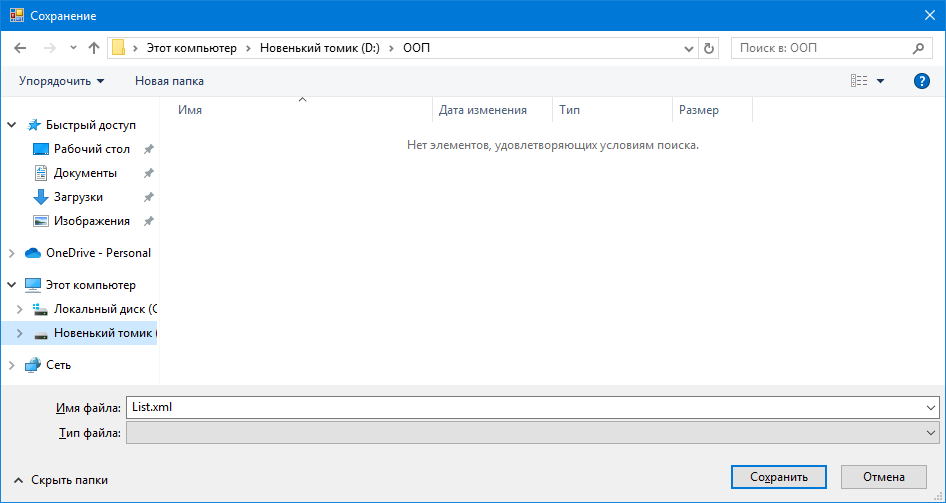


Рисунок 15 – Сохранение файла

**1.5.5 Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо в меню выбрать опцию «Открыть». Далее откроется системный диалог загрузки файла (рисунок 16).

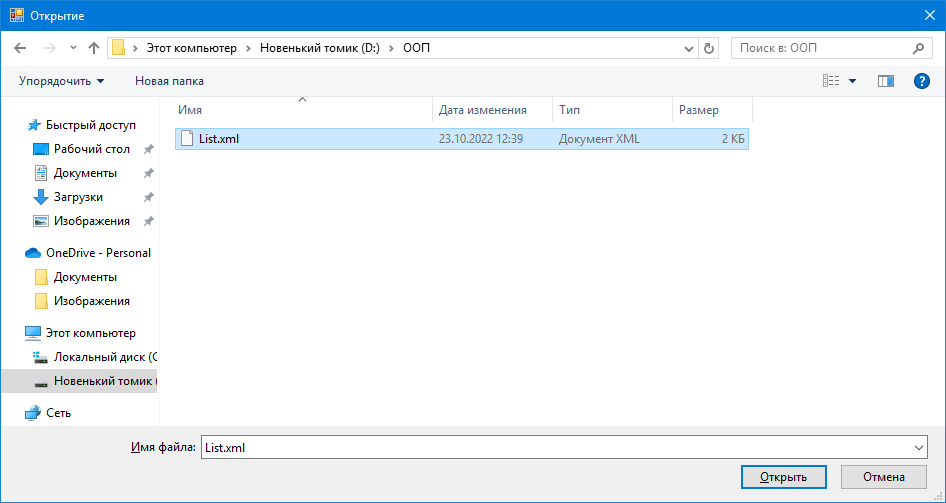


Рисунок 16 – Выбор файла для загрузки

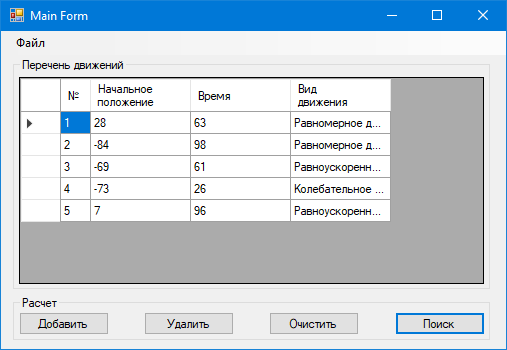


Рисунок 17 – Результат загрузки данных

В случае, если схема файла не соответствует установленному формату, появится соответствующее сообщение (рисунок 18).

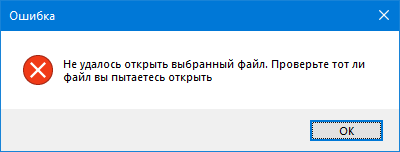


Рисунок 18 – Загрузка повреждённого файла

**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 сохранения видов движения

Разработчик: студент гр. О-5КМ01 Мизев 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: «Программа для просмотра созданных экземпляров различных видов движения».

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 Мизев А.В.

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

Начало работ: 01 октября 2022 г.

Окончание работ: 30 ноября 2022 г.

**2 Назначение и цели создания системы**

**2.1 Назначение системы**

Система предназначена для просмотра созданных видов движений с заданными параметрами.

**2.2 Цели создания системы**

Система создается в целях автоматизации расчета и контроля процесса создания экземпляров движения.

**3 Характеристика объектов автоматизации**

Система пригодится в образовательном процессе, когда необходимо создавать задачи для проведения занятий. Преподаватель имеет возможность создавать архив похожих задач для проверки знаний учащихся через определение конечной координаты движения по заданным параметрам.

**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 движения хранятся в XML-файле с расширением \*.xml.

**D01.01.** XML-файла должен иметь следующую структуру:

**4.3 Функциональные требования**

**F01.** В системе должен быть реализован список движений.

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

* Начальная координата;
* Тип движения;
* Время движения.

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

Равномерное движение:

* скорость.

Равноускоренное движение:

* скорость;
* ускорение.

Колебательное движение:

* амплитуда;
* циклическая частота.

**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 4.8

**4.6 Требования к аппаратному обеспечению**

**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: 16 ГБ для 32-разрядных систем или 20 ГБ для 64-разрядных систем