Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждения высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по дисциплине

«Основы разработки САПР» (ОРСАПР)

Выполнил:
студент гр. 589-2
_____ Насонов А.А.
«___» _____ 2022 г.
Руководитель:
к.т.н., доцент каф. КСУП
_____ Калентьев А.А.
«___» _____ 2022 г.

Томск, 2022

Оглавление

1 Введение

Автоматизация моделирования имеет огромное значение для развития науки, техники и производства в современном обществе. В настоящее время автоматизация – основной способ повышения производительности и эффективности труда инженерно-технических работников, занимающихся моделированием сложных устройств. Использование автоматизации в проектировании позволяет создавать все более сложные технические объекты и гибко реагировать на появление новых решений и технологий в той или иной области техники. Она позволяет значительно повысить точность расчетов, выбрать наилучшие варианты для реализации на основе строгого математического анализа всех или большинства вариантов проекта с оценкой технических, технологических и экономических характеристик производства и эксплуатации проектируемого объекта, значительно повысить качество конструкторской документации, существенно сократить сроки проектирования и передачи конструкторской документации в производство, эффективнее использовать технологическое оборудование с программным управлением [1].

Таким образом, целью данной работы является разработка плагина, автоматизирующего построение модели «Шкаф-стол» для системы автоматизированного проектирования Компас 3D с помощью интегрированной среды разработки Jetbrains Rider 2022.3. [2]

JetBrains Rider — кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией JetBrains. Поддерживаются языки программирования C#, VB.NET и F#. [2]

2 Постановка и анализ задачи

В рамках лабораторных работ в соответствии с технически заданием требовалось разработать плагин, который на основе входных параметров, интегрируя с системой Компас 3D, строит модель «Шкаф-стол». [3] Необходимо чтобы плагин позволял задавать параметры по умолчанию, а также изменять входные параметры шкафа-стола, такие как:

- ширина шкафа;
- длина шкафа;
- высота ножек;
- высота шкафа-стола;
- закругление ребер шкафа-стола;
- закругление углов шкафа-стола;
- количество полок.

2.1 Описание предмета проектирования

Предметом проектирования является модель шкаф-стола. Данная модель имеет 6 основных параметров:

- W – ширина (мм, диапазон значений: 600-1000 мм);
- L – длина (мм, не меньше чем 75% от ширины, диапазон значений: 450-750 мм);
- H1 – высота ножек (мм, меньше H2 не менее чем в 7.5 раз, но не меньше 100 мм);
- H2 – высота шкафа-стола (мм, диапазон значений: 600-1050 мм);
- R1 – закругление ребер шкафа-стола (мм, диапазон значений: 0-15);
- R2 – закругление углов шкафа-стола (мм, диапазон значений: 0-30).

На рисунке 2.1 представлен чертеж модели.

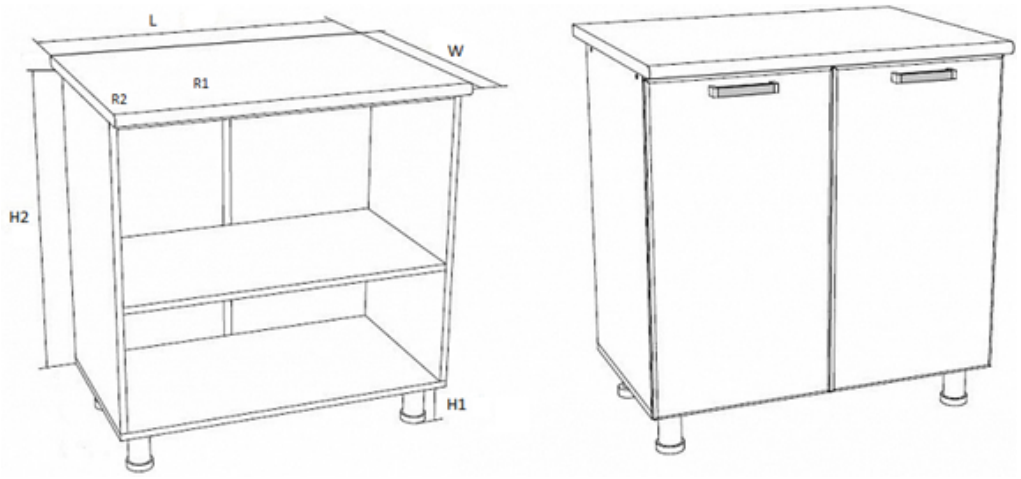

Рисунок 2.1 – Чертеж модели.

2.2 Выбор инструментов и средств реализации

На основе требований к техническому заданию программа выполнена на языке программирования C# в среде Microsoft Visual Studio 2019 с использованием .NET Framework 4.7.2 [2], бибиблиотеки для Kompas 3D [4].

Инструментом тестирования и создания модульных тестов был выбран тестовый фреймворк NUnit [5] версии 3.13.2.

Для реализации пользовательского интерфейса использовалась система для построения настольные приложения WinForms [6].

2.3 Назначение плагина

Назначение разрабатываемого плагина обусловлено быстрым моделированием столов-шкафов разных типов. Благодаря данному расширению, столяры могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

3 Обзор аналогов

Плагин PRO100

PRO100 — программа для проектирования мебели, кухни, ванных комнат, интерьеров офисов и помещений.

PRO100 упрощает работу на каждом этапе производственного процесса. Проектирование мебели, разработка дизайна интерьера, расчёт стоимости, получение списка деталей для производства, оптимизация производства заказов. Кухни, спальни, библиотеки, ванные комнаты, офисы, гардеробные, шкафы-купе, помещения. Одна программа для решения множества задач. [7]

На рисунке 1.1 показан интерфейс программы PRO100.

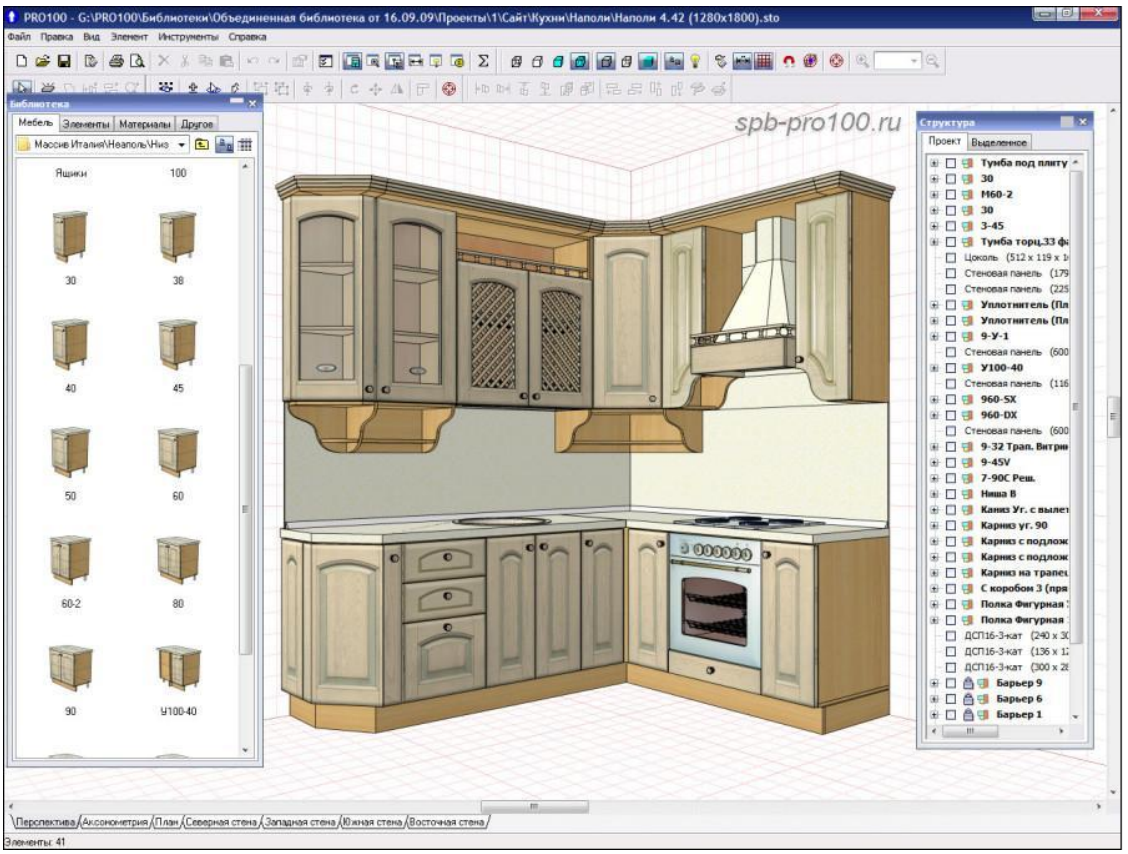

Рисунок 3.1 — Интерфейс PRO100

К3-Мебель

«К3-Мебель» – профессиональный комплекс для проектирования, производства и дизайна корпусной мебели. Мощный и одновременно простой инструмент, позволяющий создать изделие любой степени сложности,

получить полный пакет документации в один клик и представить заказчику реалистичное изображение его будущего проекта. [8]

На рисунке 1.1 показан интерфейс программы К3-Мебель.

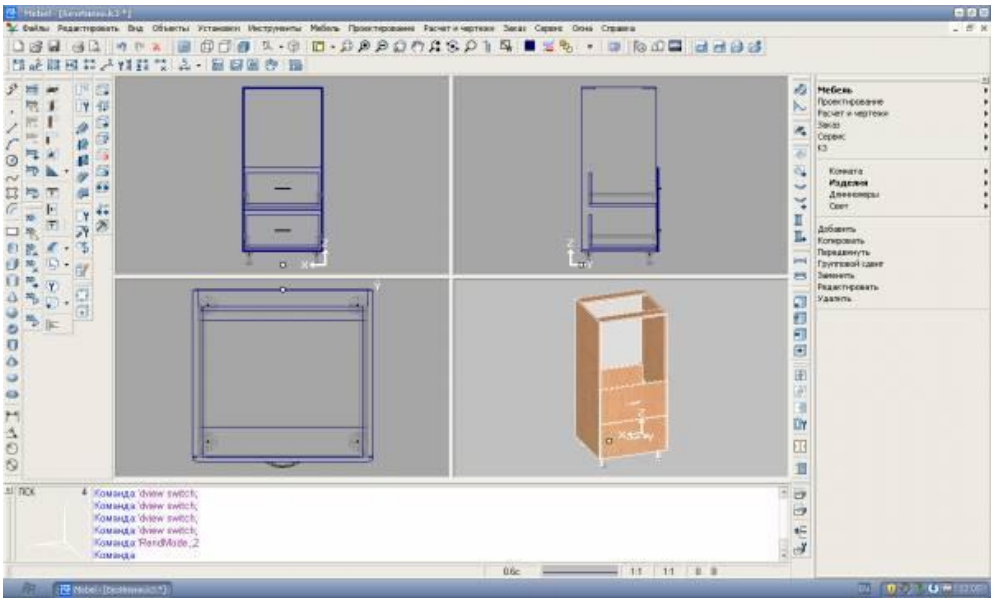

Рисунок 3.2 — Интерфейс К3-Мебель

4 Описание реализации

Для графического описания абстрактной модели проекта, а также пользовательского взаимодействия (сценарии действия) использован стандарт UML.

UML язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML – моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML возможна генерация кода и наоборот.[9]

При использовании UML были простроена диаграмма классов.

4.1 Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперaций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.[9]

На рисунке 4.1 представлена диаграмма классов.

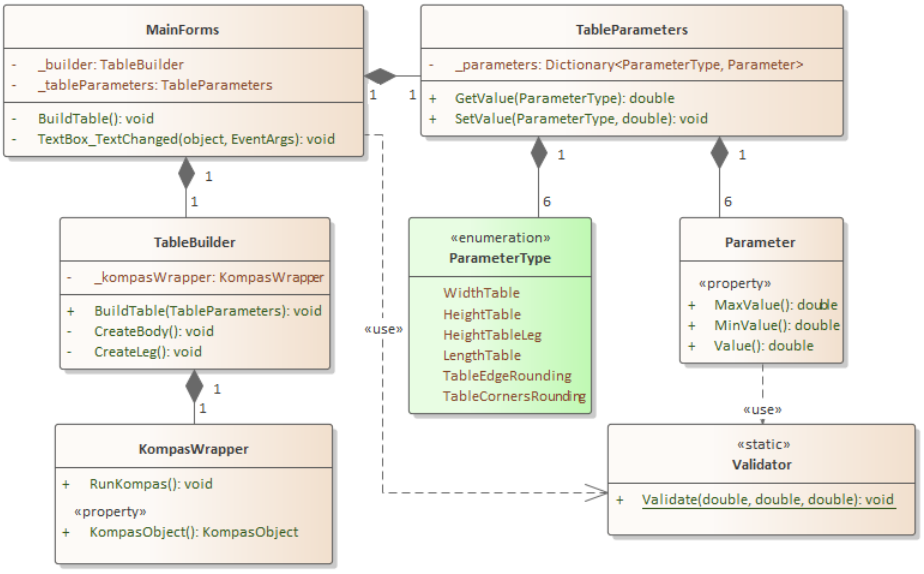

Рисунок 4.1 – Изначальная диаграмма классов

Parameter — класс, содержащий в себе максимальное и минимальное значение числа, а также текущее значение параметра. Проверка изменения значения происходит в классе Validator. Взаимная проверка параметров будет происходить при изменении текста в TextBox. Все элементы TextBox будут подписаны на обработчик события TextBox_TextChanged. Хранить параметры в форме нужно для изменения текущих значений, а также значений минимумов и максимумов параметров.

ParameterType — перечисление, хранящая в себе все названия параметров.

TableParameters — хранит в себе все параметры шкафа-стола, а также через него происходит получение и установка значений параметров.

TableBuilder — класс построения объекта, содержащий в себе класс KompasWrapper (объединяющий API Компаса и программу).

MainForm — главное окно, в котором пользователь сможет изменять значения параметров.

В итоговом проекте созданы следующие классы и методы, которые отображены на итоговой диаграмме классов (рисунок 4.2).

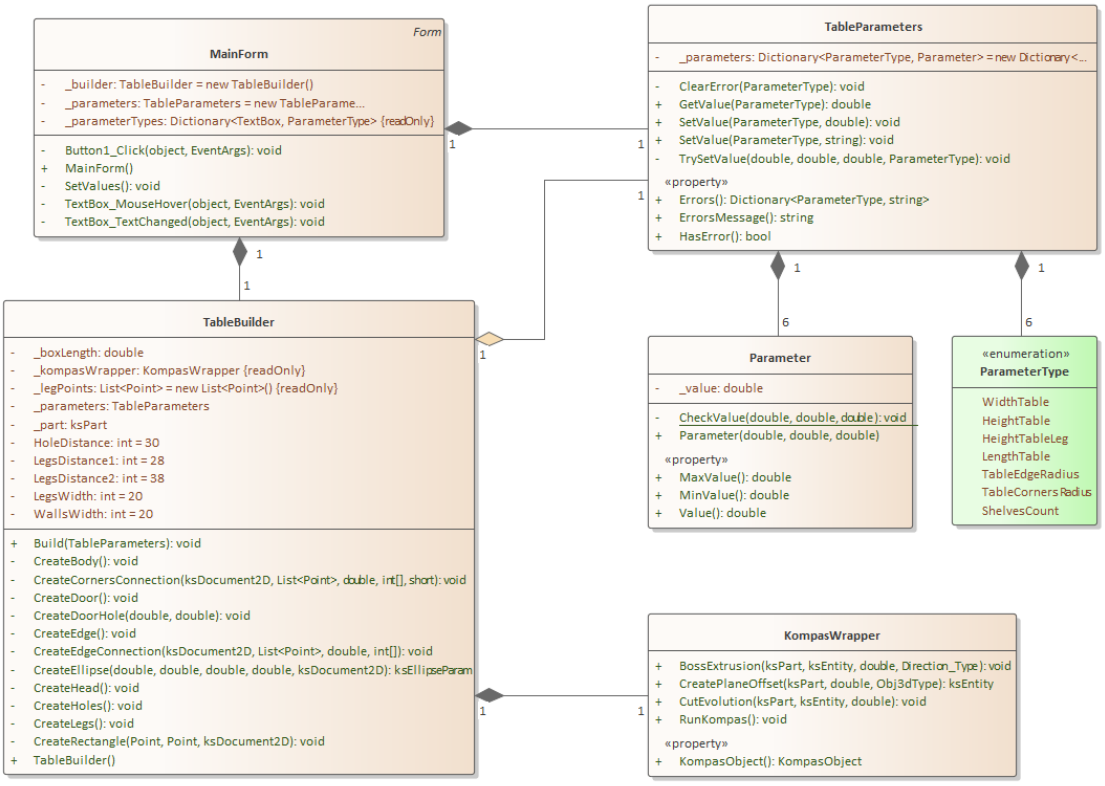

Рисунок 4.2 – Итоговая диаграмма классов

В класс MainForm были добавлены обработчик события TextBox_MouseHover для элементов TextBox. Данный обработчик показывает сообщение об ошибки при наведении курсором на TextBox. Метод SetValues устанавливает значения из TableParameters в TextBox-ы (таблицы 4.1 и 4.2).

В класс TableParameters были добавлены свойства для обработки ошибок (HasError, ErrorsMessage, Errors). Также был добавлен метод установки значения параметру через посылаемую строку (таблицы 4.3 и 4.4).

В класс Parameter был перенесен метод проверки принадлежности значения в диапазон. Класс Validator был убран (таблицы 4.5 и 4.6).

В класс TableBuilder был добавлены константы, а также методы для создания частей шкафа-стола. Было добавлено поле тапа TableParameters для получения значения параметров (таблицы 4.7 и 4.8).

В класс KompasWrapper были добавлены методы для выдавливания и создания эскизов (таблицы 4.9).

Таблица 4.1 – Поля класса MainForm

Название	Тип	Описание
_parameters	TableParameters	Параметры стола
_builder	TableBuilder	Строитель стола
_parameterTypes	Dictionary<TextBox, ParameterType>	Словарь текстбоксов

Таблица 4.2 – Методы класса MainForm

Название	Параметры	Возвращаемый тип	Описание
MainForm	–	–	Конструктор
SetValues	–	void	Установить значения из экземпляра класса
Button1_Click	object, EventArgs	void	Обработчик события нажатия на кнопку
TextBox_TextChanged	object, EventArgs	void	Обработчик события для текстбоксов
TextBox_MouseHover	object, EventArgs	void	Обработчик события наведения мыши на текстбокс

Таблица 4.3 – Поля класса TableParameters

Название	Тип	Описание
_parameters	Dictionary<ParameterType, Parameter>	Словарь параметров стола

Таблица 4.4 – Методы и свойства класса TableParameters

Название	Параметры	Возвращаемый тип	Описание
Errors	–	Dictionary<Parameter Type, string>	Свойство, возвращающее словарь ошибок
HasError	–	bool	Свойство возвращает true, если есть ошибка
ErrorsMessage	–	string	Свойство возвращает тексты ошибок
GetValue	ParameterType	double	Метод возвращает значение параметра
SetValue	ParameterType, double	void	Метод устанавливает значение параметру
SetValue	ParameterType, string	void	Метод устанавливает значение параметру
ClearError	ParameterType	void	Метод очищает ошибку по типу
TrySetValue	double, double, double, ParameterType		Метод, который пытается установить новое значение параметра

Таблица 4.5 – Поля класса Parameter

Название	Тип	Описание
_value	double	Текущее значение параметра

Таблица 4.6 – Методы и свойства класса Parameter

Название	Параметры	Возвращаемый тип	Описание
Value	–	double	Свойство возвращает и устанавливает значение параметра
MaxValue	–	double	Свойство возвращает и устанавливает максимальное значение
MinValue	–	double	Свойство возвращает и устанавливает минимальное значение
Parameter	double, double, double	–	Конструктор
CheckValue	double, double, double	void	Метод проверяет значение на принадлежность промежутку

Таблица 4.7 – Поля класса TableBuilder

Название	Тип	Описание
WallsWidth	int	Толщина стенок (значение 20)
LegsWidth	int	Ширина ножек (значение 20)
LegsDistance1	int	Первое расстояние от угла (значение 28)
LegsDistance2	int	Второе расстояние от угла (значение 38)
HoleDistance	int	Расстояние от отверстий (значение 30)
_legPoints	List<Point>	Координаты ножек комода
_boxLength	double	Длина ящиков
_kompasWrapper	KompasWrapper	Экземпляр класса работы с Компас 3D
_part	ksPart	Часть модели
_parameters	TableParameters	Параметры модели

Таблица 4.8 – Методы класса TableBuilder

Название	Параметры	Возвращаемый тип	Описание
TableBuilder	–	–	Конструктор
Build	TableParameters	void	Метод строит модель
CreateDoor	–	void	Метод создает двери
CreateDoorHole	double, double	void	Метод делает отверстия в двери
CreateHead	–	void	Метод создает верхнюю часть стола
CreateCornersConnection	ksDocument2D, List<Point>, double, int[],short	void	Метод создает соединения углов
CreateEdge	–	void	Метод создает округленные ребра
CreateEdgeConnection	ksDocument2D, List<Point>, double, int[]	void	Метод создает соединения ребер
CreateLegs	–	void	Метод создает ножки
CreateBody	–	void	Метод создает тело стола
CreateHoles	–	void	Метод создает отверстия для ящиков
CreateRectangle	Point, Point, ksDocument2D	void	Метод создает прямоугольник по двум точкам
CreateEllipse	double, double, double, double, ksDocument2D	void	Метод создает эллипс

Таблица 4.9 – Методы и свойства класса KompasWrapper

Название	Параметры	Возвращаемый тип	Описание
KompasObject	–	KompasObject	Свойство, возвращающее экземпляр Компас 3D
RunKompas	–	void	Метод запускает Компас 3D
BossExtrusion	ksPart, ksEntity, double, Direction_Type	void	Метод выдавливание объекта

CutEvolution	ksPart, ksEntity, double	void	Выдавливание с вырезом
CreatePlaneOffset	ksPart, double, Obj3dType	ksEntity	Создание плоскости относительно плоскости type на расстоянии на определенном расстоянии

5 Описание программы для пользователя

Макет пользовательского интерфейса представляет собой форму для ввода параметров шкафа-стола. Построение модели осуществляется путем нажатия на кнопку «Построить». При попытке ввода недопустимых символов, поле с неверными значениями будет подсвечиваться красным и при наведении на нее будет показываться сообщение ошибки.

Плагин состоит из диалогового окна, которое имеет 7 полей ввода параметров, 1 кнопку.

На рисунке 5.1 представлен пользовательский интерфейс.

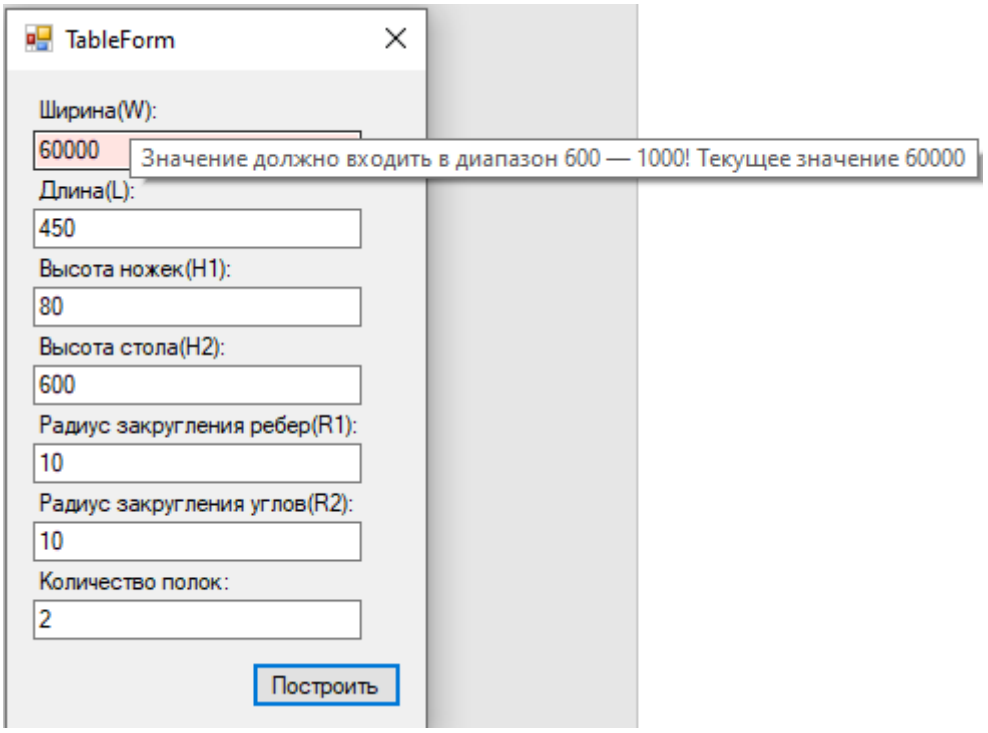

Рисунок 5.1 – Пользовательский интерфейс

Если ввести неверные параметры, после нажатия кнопки «Построить», высветится окно с просьбой ввести правильные параметры в поля ввода (рисунок 5.2).

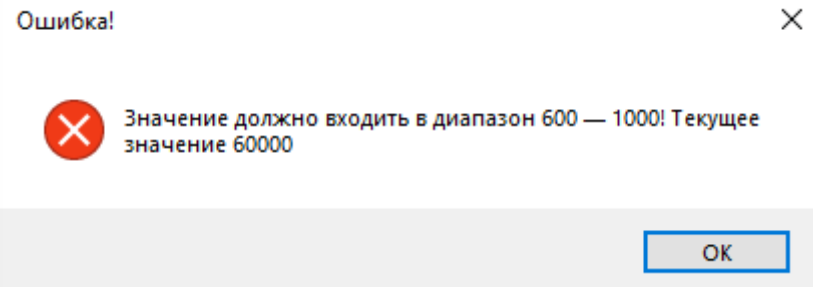

Рисунок 5.2 — Окно ошибки

После ввода необходимых параметров, построить деталь в САПР Компас 3D можно с помощью кнопки «Построить». Шкаф-стол, построенный по заданным параметрам по умолчанию в САПР Компас 3D, представлен на рисунке 5.3.

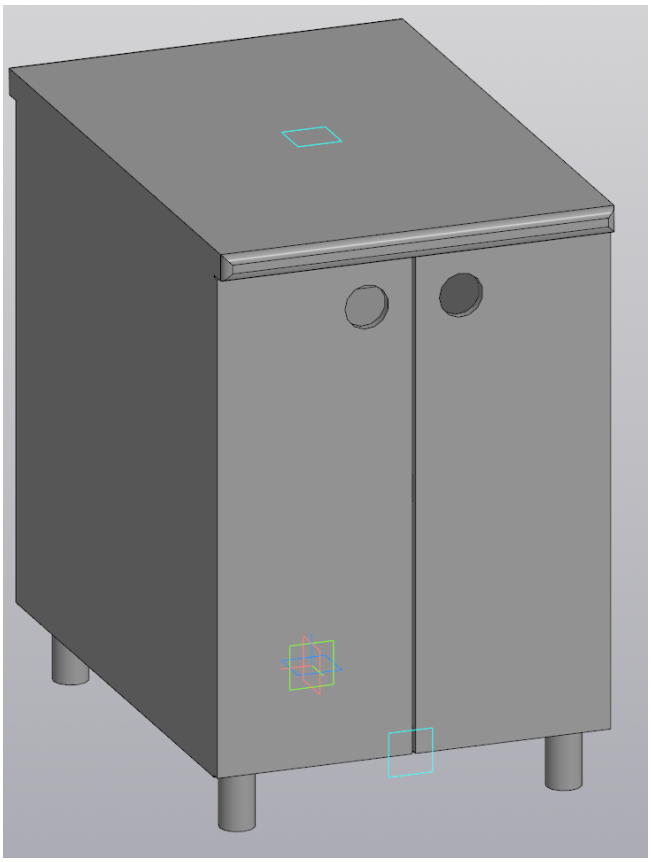

Рисунок 5.3 — Построенный шкаф-стол

6 Тестирование программы

Тестирование позволяет убедиться в работоспособности программы, выявлять ошибки при изменении какого-либо функционала.

6.1 Функциональное тестирование

При функциональном тестировании проверялось корректность работы плагина «Шкаф-стол», а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами. [10]

Проведено тестирование максимальных и минимальных параметров модели.

На рисунках 6.1 представлен проверка размеров модели с минимальным введенными параметрами в САПР Kompas 3D (ширина 600 миллиметров, длина 450 миллиметров, высота ножек 80 миллиметров, высота шкафа-стола 600 миллиметров, радиус закругления ребер 10 мм, радиус закругления углов 10 мм, количество полок 2).

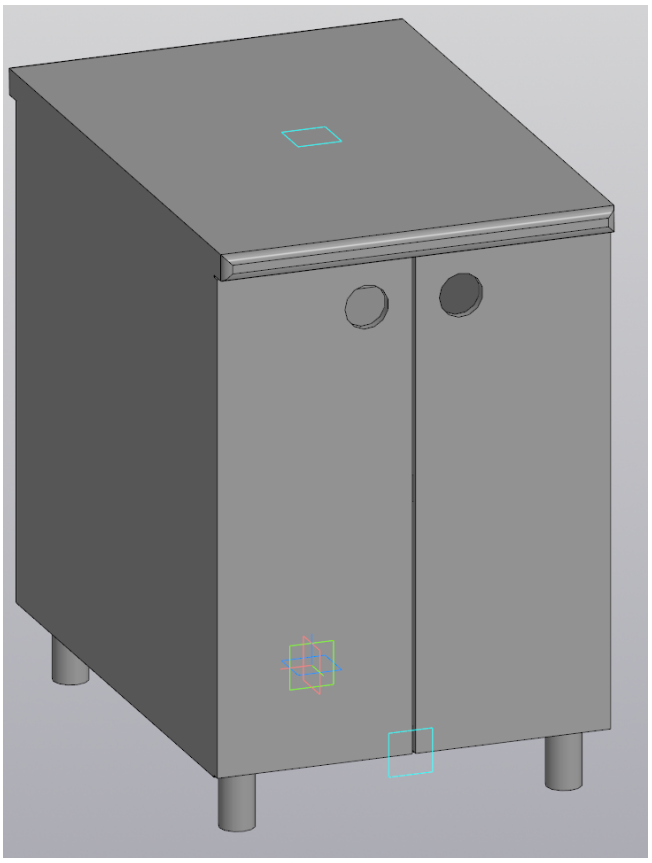

Рисунок 6.1 – Модель с минимальными введенными параметрами в Kompas 3D

Ниже на рисунках 6.2 представлена проверка размеров модели с максимальными введенными параметрами в САПР Kompas 3D (ширина 1000 миллиметров, длина 1050 миллиметров, высота ножек 80 миллиметров, высота шкафа-стола 1050 миллиметров, радиус закругления ребер 15 мм, радиус закругления углов 30 мм, количество полок 4).

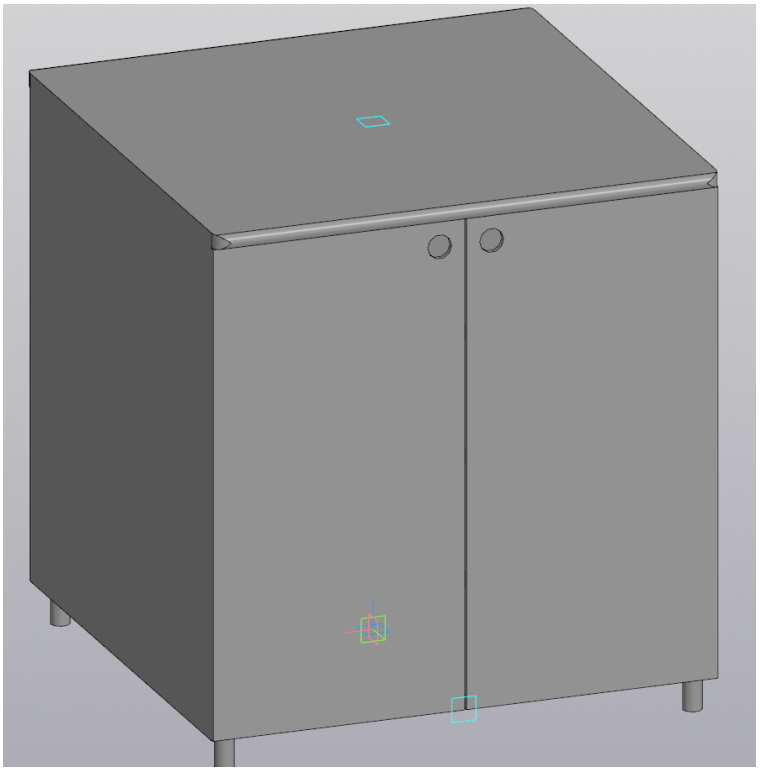

Рисунок 6.2 – Модель с максимально веденными параметрами в Kompas 3D

6.2 Модульное тестирование

В целях проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версии 3.13 проведено модульное тестирование [11], проверялись открытые поля и методы. На рисунке 6.3 представлено тестирование классов проекта Core. Степень покрытия проектов — сто процентов. Было написано 42 теста.

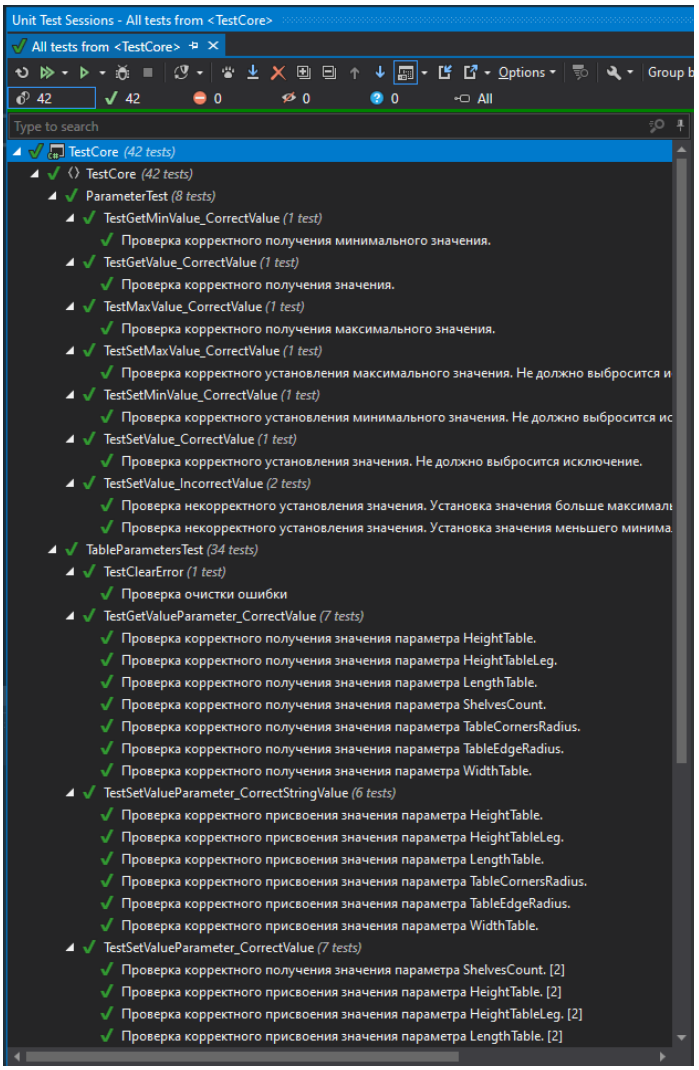

Рисунок 6.3 – Тестирование классов

В таблицах 6.1 и 6.2 перечислены все тестовые методы, а в таблицах 6.3 и 6.4 описания тест-кейсов.

Таблица 6.1 – Тестовые методы ParameterTest

Название теста	Параметры	Описание
TestSetValue_IncorrectValue	double	Проверка некорректного установления значения. Установка значения меньшего минимального и больше максимального.
TestSetValue_CorrectValue	–	Проверка корректного установления значения без выброса исключения.
TestGetValue_CorrectValue	–	Проверка корректного получения значения.
TestSetMaxValue_CorrectValue	–	Проверка корректного установления максимального значения.

TestMaxValue_CorrectValue	–	Проверка корректного получения максимального значения
TestSetMinValue_CorrectValue	–	Проверка корректного установления минимального значения.
TestGetMinValue_CorrectValue	–	Проверка корректного получения минимального значения.

Таблица 6.2 – Тестовые методы TableParametersTest

Название теста	Параметры	Описание
TestClearError	–	Проверка очистки ошибки.
TestGetValueParameter_ CorrectValue	ParameterType, int	Проверка корректного получения значения параметров.
TestSetValueParameter_ CorrectStringValue	ParameterType, string	Проверка корректного присвоения строкового значения параметров.
TestSetValueParameter_ CorrectValue	ParameterType, int	Проверка корректного присвоения значения параметров.
TestSetValueParameter_ IncorrectStringValue	ParameterType, string	Проверка некорректного присвоения строкового значения параметров.
TestSetValueParameter_ IncorrectValue	ParameterType, int	Проверка некорректного присвоения значения параметров.

Таблица 6.3 – Описание тест-кейсов ParameterTest

Название теста	Тест-кейсы
TestSetValue_IncorrectValue	Проверка некорректного установления значения. Установка значения меньшего минимального
	Проверка некорректного установления значения. Установка значения больше максимального.
TestSetValue_CorrectValue	Проверка корректного установления значения.
TestGetValue_CorrectValue	Проверка корректного получения значения.
TestSetMaxValue_CorrectValue	Проверка корректного установления максимального значения.
TestMaxValue_CorrectValue	Проверка корректного получения максимального значения.
TestSetMinValue_CorrectValue	Проверка корректного установления минимального значения.
TestGetMinValue_CorrectValue	Проверка корректного получения минимального значения.

Таблица 6.4 – Описание тест-кейсов TableParametersTest

Название теста	Тест-кейсы
TestGetValueParameter_ CorrectValue	Проверка корректного получения значения параметра HeightTable.

	Проверка корректного получения значения параметра HeightTableLeg.
	Проверка корректного получения значения параметра WidthTable.
	Проверка корректного получения значения параметра LengthTable.
	Проверка корректного получения значения параметра TableCornersRadius.
	Проверка корректного получения значения параметра TableCornersRadius.
	Проверка корректного получения значения параметра TableEdgeRadius.
	Проверка корректного получения значения параметра ShelvesCount.
TestSetValueParameter_ CorrectStringValue	Проверка корректного получения значения параметра HeightTable.
	Проверка корректного получения значения параметра HeightTableLeg.
	Проверка корректного получения значения параметра WidthTable.
	Проверка корректного получения значения параметра LengthTable.
	Проверка корректного получения значения параметра TableCornersRadius.
	Проверка корректного получения значения параметра TableCornersRadius.
	Проверка корректного получения значения параметра TableEdgeRadius.
	Проверка корректного получения значения параметра ShelvesCount.
TestSetValueParameter_ IncorrectStringValue	Проверка некорректного получения значения параметра HeightTable.
	Проверка некорректного получения значения параметра HeightTableLeg.
	Проверка некорректного получения значения параметра WidthTable.
	Проверка некорректного получения значения параметра LengthTable.
	Проверка некорректного получения значения параметра TableCornersRadius.
	Проверка некорректного получения значения параметра TableCornersRadius.
	Проверка некорректного получения значения параметра TableEdgeRadius.
	Проверка некорректного получения значения параметра ShelvesCount.
TestSetValueParameter_ CorrectValue	Проверка корректного получения значения параметра HeightTable.
	Проверка корректного получения значения параметра HeightTableLeg.

	Проверка корректного получения значения параметра WidthTable.
	Проверка корректного получения значения параметра LengthTable.
	Проверка корректного получения значения параметра TableCornersRadius.
	Проверка корректного получения значения параметра TableCornersRadius.
	Проверка корректного получения значения параметра TableEdgeRadius.
	Проверка корректного получения значения параметра ShelvesCount.
TestSetValueParameter_ IncorrectValue	Проверка некорректного получения значения параметра HeightTable.
	Проверка некорректного получения значения параметра HeightTableLeg.
	Проверка некорректного получения значения параметра WidthTable.
	Проверка некорректного получения значения параметра LengthTable.
	Проверка некорректного получения значения параметра TableCornersRadius.
	Проверка некорректного получения значения параметра TableCornersRadius.
	Проверка некорректного получения значения параметра TableEdgeRadius.
	Проверка некорректного получения значения параметра ShelvesCount.

6.3 Нагрузочное тестирование

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование [12]. Тестирование производилось на ПК со следующей конфигурацией:

- ЦП Intel Core i5 10500H 2.5ГГц;
- 8 ГБ ОЗУ;
- графический процессор объемом памяти 6 ГБ.

На рисунке 6.4 для проведения нагрузочного тестирования был добавлен секундомер («Stopwatch»), который засекал время от начала построения, с каждым успешным построением фигуры производилась запись результатов в текстовый файл «log.txt».

```
const int bitsInGigabyte = 1073741824;
var builder = new TableBuilder();
var stopWatch = new Stopwatch();
stopWatch.Start();
var tableParameters = new TableParameters();
var streamWriter = new StreamWriter( path: "log.txt",  append: true);
var count = 0;
while (true)
{
    builder.Build(tableParameters);
    var computerInfo = new ComputerInfo();
    var usedMemory :ulong = (computerInfo.TotalPhysicalMemory - computerInfo.AvailablePhysicalMemory)
                    / bitsInGigabyte;
    streamWriter.WriteLine(
        $"{++count}\t{stopWatch.Elapsed:hh\\:mm\\:ss}\t{usedMemory}");
    streamWriter.Flush();
}
```

Рисунок 6.4 – Зацикливание перестроения фигуры

На графике, изображенном на рисунке 6.5 ось «X» - количество построенных деталей, ось «Y» - количество потребляемой оперативной памяти. На графике, изображенном на рисунке 6.6 в текущей главе, ось «X» – время в секундах, ось «Y» – количество построенных деталей. На протяжении всех тестов (продолжительностью до сбоя Kompas 3D и Inventor) общая загруженность процессора была в пределах 30 процентов, потребление ОЗУ плагином прямолинейное от 13МБ до 17МБ для тестирования на САПР Kompas 3D.

На рисунке 6.5 представлено тестирование зацикленного перестроения фигуры со следующими параметрами:

- ширина 600 миллиметров;
- длина 450 миллиметров;
- высота ножек 80 миллиметров;
- высота шкафа-стола 600 миллиметров;
- радиус закругления ребер 10 градусов;
- радиус закругления углов 10 градусов;
- количество полок 2.

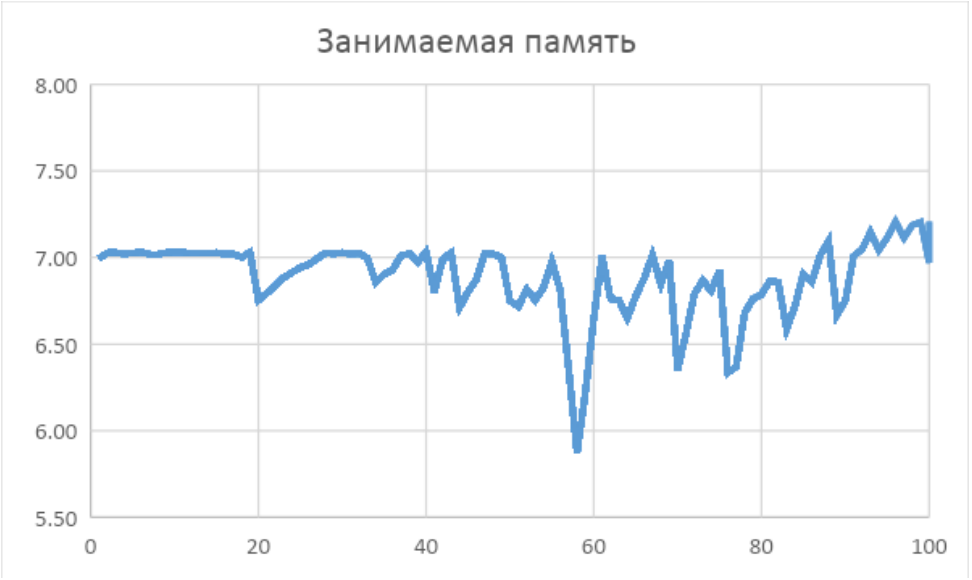

Рисунок 6.5 – График зависимости загруженности памяти от количества деталей

Рисунок 6.6 – График зависимости времени от количества построенных деталей с параметрами по умолчанию

Исходя из вышеуказанных графиков на рисунках 6.5 и 6.6, время построения было линейно возрастало до 57 построения, потом время следующего построения увеличилось и затем вернулось в линейное возрастание. Возможно, это связано с тем, что фоном операционная система

запустила задачу, что сказалось на времени построения. На рисунке 6.8 видно, что до запуска плагина, было занято около 7000 МБ оперативной памяти системой и сторонними процессами, которые к самому плагину отношения не имеют. В основном уровень занимаемой памяти был в пределах 7 ГБ, но на 58 построении был резкий скачек до 5.87 ГБ. Возможно, это связано с тем, что Компас 3D включил методы оптимизации, или операционная система временно прекратила какой-то процесс. На последнем построении детали (102 по счету) программа Компас 3D перестала отвечать и дальнейшее построение было невозможно. Возможно, это связанно с недостатком оперативной памяти ПК.

Заключение

В ходе выполнения лабораторных работ были изучены предметная область проектирования, предмет проектирования, аналоги предмета проектирования, API, на основании полученных данных были спроектированы UML диаграммы классов. Был разработан плагин для создания 3D моделей «Шкаф-стол» в САПР Компас 3D. Было проведено функциональное и нагрузочное тестирование плагина.

Список использованных источников

1. Автоматизация вычислительных процедур в прикладных задачах инженерного проектирования [Электронный ресурс]. – URL: https://scienceforum.ru/2014/article/2014000201 (дата обращения: 26.12.2022).
2. Jetbrains Rider [Электронный ресурс]. – URL: https://www.jetbrains.com/ru-ru/rider/ (дата обращения: 26.12.2022).
3. КОМПАС-3D. Официальный сайт САПР КОМПАС [Электронный ресурс]. — Режим доступа: https://kompas.ru/ (дата обращения: 26.12.2022).
4. КОМПАС-3D для разработчиков [Электронный ресурс]. – URL: https://kompas.ru/solutions/developers/ (дата обращения: 26.12.2022).
5. NUnit [Электронный ресурс]. – URL: https://nunit.org/ (дата обращения: 26.12.2022).
6. Руководство по классическим приложениям (Windows Forms .NET) [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0 (дата обращения: 26.12.2022).
7. Сайт PRO100 [Электронный ресурс]. – https://www.ecru.pl/ru (дата обращения 10.12.2022).
8. Сайт К3-Мебель [Электронный ресурс]. – https://k3-mebel.ru/ (дата обращения 10.12.2022).
9. UML. [Электронный ресурс]. – Режим доступа: http://www.uml.org/ (дата обращения: 26.12.2022).
10. Функциональное тестирование [Электронный ресурс]. – URL: https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/ (дата обращения: 26.12.2022).
11. Юнит-тестирование для чайников [Электронный ресурс]. – URL: https://habr.com/ru/post/169381/ (дата обращения: 26.12.2022).

12. Нагрузочное тестирование: с чего начать и куда смотреть [Электронный ресурс]. – URL: https://habr.com/ru/company/jugru/blog/329174/ (дата обращения: 26.12.2022).