Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |  |
| --- | --- |
| Допускаю к защите |  |
| Руководитель |  |
|  | подпись |
|  | А.Ю. Юрин |
|  | И.О. Фамилия |

|  |
| --- |
| Разработка программного обеспечения с использованием |
| методов и средств программной инженерии |

наименование темы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

|  |  |  |
| --- | --- | --- |
| Методы и средства проектирования информационных систем и технологий | | |
|  | 1.011.00.00 - ПЗ |  |

обозначение документа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСМб-19-1 |  |  |  | А.Р. Свистунова |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | А.Ю. Юрин |
|  |  |  |  | подпись |  | И.О. Фамилия |

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2022 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |  |
| --- | --- | --- |
| По курсу | Методы и средства проектирования информационных систем и технологий | |
| Студенту | Свистунова А.Р. | |
|  | (фамилия, инициалы) | |
| Тема проекта | Разработка программного обеспечения с использованием методов и средств программной инженерии | |
| Исходные данные | |  |
| 1. Произвести проектирование компонента для преобразования (конвертации) файла в формате EKB в файл формата draw.io (согласно варианту 11). | | |

|  |
| --- |
| Рекомендуемая литература: |
| 1. Дородных Н.О., Юрин А.Ю. Технология создания продукционных экспертных систем на основе модельных трансформаций. Новосибирск: СО РАН, 2019. 144 стр. 2. Юрин А.Ю., Грищенко М.А. Редактор баз знаний в формате CLIPS // Программные продукты и системы. 2012. № 4. С. 83–87. 3. Юрин А.Ю. CASE-средства: Методические указания по выполнению лабораторных работ. Иркутск: ИРНИТУ, 2018. 87 c. |

Графическая часть на \_\_\_\_\_\_\_\_\_\_\_\_\_\_ листах.

Дата выдачи задания « » 20 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Задание получил |  |  |  |
|  | подпись |  | И.О. Фамилия |

Дата представления проекта руководителю « » 20 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель курсового проектирования |  |  | А.Ю. Юрин |

# 

Содержание

[Содержание 3](#_Toc122351263)

[Введение 4](#_Toc122351264)

[1 Общая часть 5](#_Toc122351265)

[1.1 Описание предметной области 5](#_Toc122351266)

[1.2 Постановка задачи 6](#_Toc122351267)

[2 Специальная часть 7](#_Toc122351268)

[2.1 Техническое задание 7](#_Toc122351269)

[2.1.1 Введение 7](#_Toc122351270)

[2.1.2 Общие сведения 7](#_Toc122351271)

[2.1.3 Назначение и цели создания 8](#_Toc122351272)

[2.1.4 Характеристика объектов автоматизации 8](#_Toc122351273)

[2.1.5 Требования к программе или программному изделию 9](#_Toc122351274)

[2.1.6 Состав и содержание работ по созданию системы 10](#_Toc122351275)

[2.1.7 Порядок контроля и приемки 10](#_Toc122351276)

[2.1.8 Требования к программной документации 11](#_Toc122351277)

[2.1.9 Приложения 11](#_Toc122351278)

[2.2 Описание проекта в Trello 11](#_Toc122351279)

[2.3 Проектирование 13](#_Toc122351280)

[2.3.1 Диаграммы вариантов использования 13](#_Toc122351281)

[2.3.2 Диаграмма последовательностей 18](#_Toc122351282)

[2.3.3 Алгоритмическое обеспечение 19](#_Toc122351283)

[2.3.4 Диаграмма классов 20](#_Toc122351284)

[2.4 Программная реализация 21](#_Toc122351285)

[2.4.1 Автоматически сгенерированный код 21](#_Toc122351286)

[2.4.2 Описание интерфейса 25](#_Toc122351287)

[2.4.3 Тестирование 28](#_Toc122351288)

[Заключение 36](#_Toc122351289)

[Список использованных источников 37](#_Toc122351290)

Приложение А. [Фрагмент сгенерированной документации 38](#_Toc122351292)

Приложение Б. [Листинг программы 39](#_Toc122351294)

Введение

В рамках проекта осуществляется проектирование и разработка программного обеспечения, обеспечивающего автоматизацию процесса трансляции файлов структуры EKB в формат приложения для создания диаграмм и интерактивных досок draw.io.

Актуальность разрабатываемого программного обеспечения определяется временными затратами в рамках ручного переноса проектов между системами и поддержании данных в актуальном виде.

Разрабатываемый программное обеспечение должно автоматизировать деятельность специалистов, в области проектирования информационных систем, сокращая их временные затраты при переносе проектов в рамках представленных систем.

1 Общая часть

1.1 Описание предметной области

Программный компонент осуществляет процесс трансляции файлов EKB в формат приложения для создания диаграмм и интерактивных досок draw.io.

Редактор баз знаний Personal Knowledge Base Designer сохраняет проекты в файлы с расширением ekb и содержит многоуровневую структуру, состоящую из следующих элементов:

1. Templates – список шаблонов;
2. Facts – список фактов;
3. Rules – список правил.



Рисунок 1 – Фрагмент структуры файла ekb

Разметка файлов draw.io состоит из корневого элемента root и множества вложенных объектов структуры mxCell. Каждый элемент данной структуры хранит в себе уникальный id. Связь между элементами осуществляется с помощью указания идентификатора в параметре parent структуры дочернего компонента.

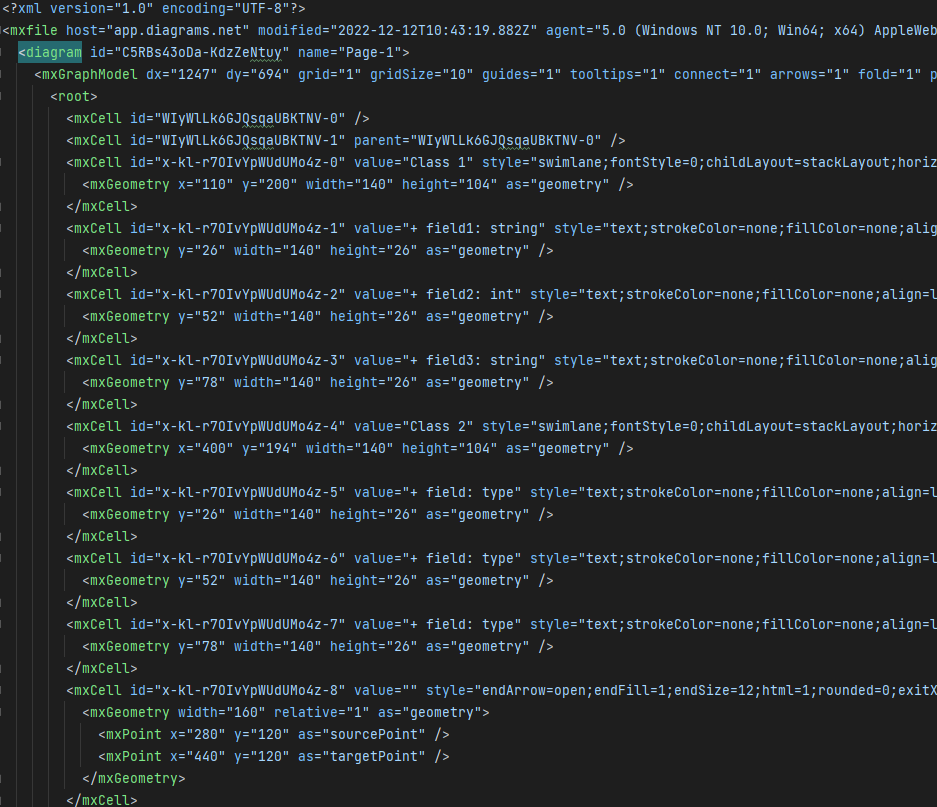


Рисунок 2 – Фрагмент структуры файла drawio.xml

Дополнительно программный компонент предоставляет возможность выбора пользователем места хранения файла для конвертации и места сохранения конвертированного файла xml.

1.2 Постановка задачи

Проектирование и программная реализация компонента для преобразования (конвертации) XML-файлов определенной структуры (согласно варианту задания).

Выполнение работы предполагает решение следующих задач:

* Проектирование, включая:
  + Описание предметной области.
  + Формулировка цели работы.
  + Проектирование компонента.
* Программная реализация компонента, включая автоматизированную генерацию программного кода.
* Тестирование.
* Документирование (описание), включая автоматизированную генерацию документации.

2 Специальная часть

2.1 Техническое задание

2.1.1 Введение

Настоящее техническое задание распространяется на разработку программного компонента для конвертации файлов Personal Knowledge Base Designer в формат приложения для создания диаграмм и интерактивных досок draw.io.

При проектировании баз знаний в определенном программном обеспечении может потребоваться продолжить работу в другом прикладном средстве. Экспортирование наработок в данной ситуации не всегда представляется возможным.

Разрабатываемая программа позволит экспортировать проекты формата EKB в файлы для работы в draw.io в автоматическом режиме.

2.1.2 Общие сведения

**2.1.2.1 Наименование и шифры**

1. Полное название системы

Программный компонент «Транслятор файлов структуры».

1. Шифр системы

Шифр системы – ПК ТФС.

**2.1.2.2 Сведения о заказчиках и исполнителях**

1. Заказчик

Иркутский Национальный Исследовательский Технический университет.

Адрес: 664074 г. Иркутск, ул. Лермонтова, 83.

1. Исполнитель

Студент Свистунова А. Р.

**2.1.2.3 Основание для разработки**

1. Основания для разработки программного компонента

Основанием для исполнения работ по созданию программного компонента, предусмотренных в настоящем ТЗ, является учебный план «09.03.02 Информационные системы и технологии на предприятиях/в машиностроении» и методические указания курсовому проекту по курсу «Методы и средства проектирования информационных систем и технологий».

1. Основания для разработки документа

Основанием для разработки настоящего документа является п.3 к методическим указаниям.

1. Нормативные документы

Настоящее Техническое Задание разработано в соответствии с требованиями ГОСТ 34.602 89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

При разработке и вводе в эксплуатацию автоматизированной системы Исполнитель должен руководствоваться требованиями следующих нормативных документов Госстандарта:

* ГОСТ 34.601.90 – стандарт, устанавливающий стадии и этапы создания АС, а также содержание работ на каждом этапе;
* ГОСТ 34.603.92 – стандарт, устанавливающий виды испытаний АС и общие требования к их проведению.

При создании проектно-эксплуатационной документации Исполнитель должен руководствоваться требованиями ГОСТ РД 50-34.698-90 «Автоматизированные системы. Требования к содержанию документов».

**2.1.2.4 Сроки исполнения работ**

Начало разработки – «01» сентября 2022 г.

Окончание разработки – «25» декабря 2022 г.

**2.1.2.5 Сведения об источниках и порядке финансирования работ**

Источник финансирования – бюджет Иркутского Национального Исследовательского Технического университета.

Порядок финансирования определяется условиями методического указания.

**2.1.2.6 Порядок оформления и представления заказчику результатов работ**

Порядок оформления и предъявления работы Заказчику ПК ТФС производится согласно методическому указанию. Приемка системы осуществляется комиссией, созданной Заказчиком.

2.1.3 Назначение и цели создания

**2.1.3.1 Назначение**

Программный компонент ПК ТФС предназначен для автоматической конвертации файлов структуры EKB в формат приложения для создания диаграмм и интерактивных досок draw.io.

**2.1.3.2 Цели создания**

1. Цели создания и внедрения ПК ТФС:

автоматизация конвертации файлов структуры;

1. Цели текущего этапа

Целью работ по данному этапу проекта является создание первой экспериментальной версии программного компонента.

2.1.4 Характеристика объектов автоматизации

**2.1.4.1 Краткие сведения об объекте автоматизации**

Объектом автоматизации является учебный процесс Иркутского Национального Исследовательского Технического университета.

Предметом автоматизации является процесс конвертации файлов структуры.

**2.1.4.2 Сведения об условиях эксплуатации**

Программный компонент будет эксплуатироваться в составе программного комплекса Заказчика.

2.1.5 Требования к программе или программному изделию

**2.1.5.1 Требования к функциональным характеристикам**

1. Программный компонент должен обеспечивать возможность выполнения следующих функций:

указание пути к файлу программного средства Personal Knowledge Base Designer;

указание пути к месту сохранения конвертированного файла;

оповещение пользователя об успешной процедуре конвертации файла;

оповещение пользователя об ошибке при конвертации файла.

1. Исходные данные:

путь к файлу программного средства Personal Knowledge Base Designer;

пусть к месту сохранения конвертированного файла.

**2.1.5.2 Требования к надежности**

Предусмотреть контроль вводимой информации.

Предусмотреть блокировку некорректных действий пользователя при работе с системой.

**2.1.5.3 Требования к составу и параметрам технических средств**

Система должна работать на IBM совместимых персональных компьютерах.

Минимальная конфигурация:

тип процессора: Celeron и выше;

объем оперативного запоминающего устройства: 2048 Мб и более.

**2.1.5.4 Требования к информационной и программной совместимости**

Система должна работать под управлением семейства операционных систем Windows NT.

**2.1.5.5 Требования к организационному обеспечению**

В ходе разработки должно обеспечиваться постоянное взаимодействие между сторонами, для чего ими должны быть сформированы рабочие группы, решающие следующие вопросы:

- административные вопросы;

- инженерно-технические вопросы;

- вопросы методического обеспечения.

2.1.6 Состав и содержание работ по созданию системы

Таблица 1 – Состав и содержание работ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок | Отчетность |
| 1 | Разработка ядра программного компонента. | 01.09.2022-30.10.2022 | Реализация системы на уровне ядра. Описание методов и алгоритмов. |
| 2 | Разработка базового графического интерфейса приложения. | 31.10.2022- 13.11.2022 | Описание базового графического интерфейса. |
| 3 | Разработка набора тестов для проверки программного компонента. | 14.11.2022- 04.12.2022 | Набор тестовых задач. |
| 4 | Тестирование программного продукта и составление программной документации. | 05.12.2022-25.12.2022 | Тесты. Документация. Про­граммный продукт. |

По окончанию каждого этапа заказчику будет предоставлен отчёт о выполненной работе.

2.1.7 Порядок контроля и приемки

**2.1.7.1 Сдача-приемка работ**

Сдача-приёмка работ производится поэтапно, в соответствии с Календарным графиком. Основанием для сдачи-приёмки работ служит Отчёт о завершении работ по этапу.

**2.1.7.2 Предварительные испытания**

При сдаче-приёмке готовых подсистем ПК ТФС, создаваемых в рамках настоящей работы, проводятся предварительные испытания с целью подтверждения работоспособности соответствующей подсистемы и соответствия требованиям ТЗ. Предварительные испытания должны проводиться представителями Заказчика по программе тестирования, составленной Исполнителем и согласованной с Заказчиком.

По итогам испытаний по мере необходимости Исполнителем проводится корректировка проектной и эксплуатационной документации и устранение выявленных недостатков программного обеспечения.

**2.1.7.3 Опытная эксплуатация**

Передача системы в опытную эксплуатацию производится приемо-сдаточной комиссией на основании положительного заключения, сделанного по итогам предварительных испытаний.

Опытная эксплуатация должна проводиться на технических средствах Заказчика.

2.1.8 Требования к программной документации

Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии.

В состав сопровождающей документации должны входить:

Пояснительная записка на 25-30 листах, содержащая описание разработки.

Руководство пользователя.

2.1.9 Приложения

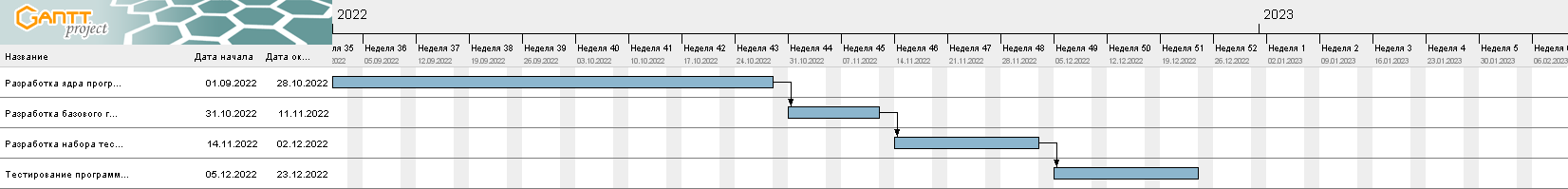


Рисунок 3 – Диаграмма Ганта

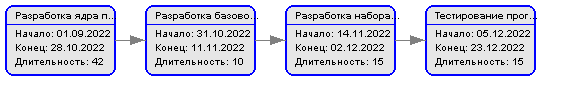


Рисунок 4 – Перт диаграмма

2.2 Описание проекта в Trello

В качестве программы для управления проектом был выбрана Trello - облачную программу для управления проектами небольших групп, разработанную Fog Creek Software.

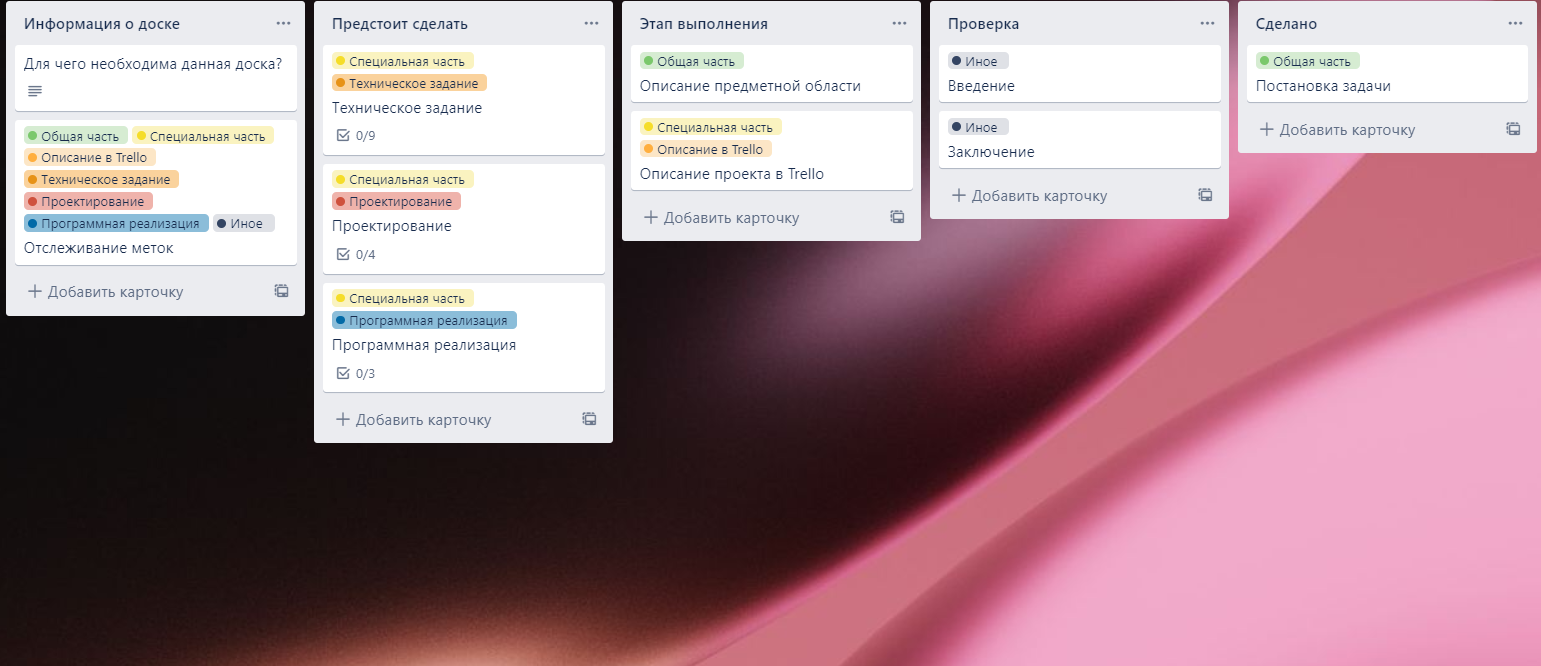


Рисунок 5 – Проект в программе Trello

Созданная доска состоит из пяти колонок. Первая колонка хранит в себе описание проекта и набор меток для карточек.



Рисунок 6 – Описание карточки с информацией о доске

Метки содержат в себе обязательные этапы курсового проекта. Всего было создано семь меток. Каждой карточке на доске прикреплен соответствующий знак.

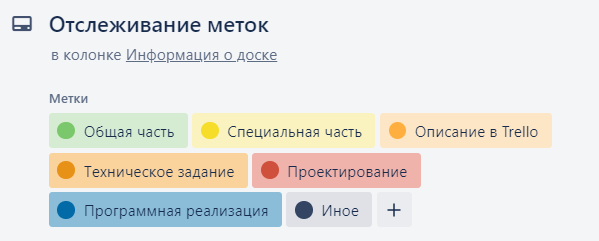


Рисунок 7 – Набор меток

Вторая колонка предназначена для хранения карточек с предстоящими для выполнения задачами.

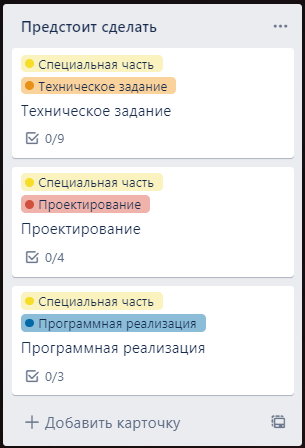


Рисунок 8 – Колонка «Предстоит сделать»

Два последующих блока содержат в себе выполненные карточки и задачи, отправленные на проверку.

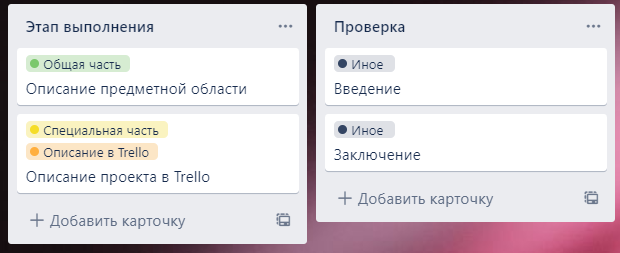


Рисунок 9 – Колонки «Этап выполнения» и «Проверка»

Колонка «Сделано» хранит карточки с выполненными и проверенными заданиями.

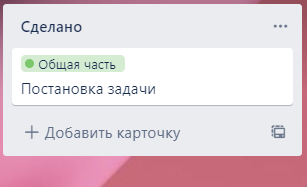


Рисунок 10 – Колонка «Сделано»

2.3 Проектирование

В качестве инструмента для проектирования было использовано CASE-средство Enterprise Architect. Enterprise Architect представляет собой уникальный и многофункциональный инструмент, предназначенный для осуществления визуального моделирования и дизайна. Главная особенность данного программного обеспечения заключается в том, что его система основана на OMG UML – одно из самых эффективных и мощных платформ в современном мире. Платформа поддерживает: проектирование и построение программных комплексов, моделирование бизнес-процессов и моделирование отраслевых доменов.

2.3.1 Диаграммы вариантов использования

На основе составленного технического задания была составлена диаграмма вариантов использования программного компонента.

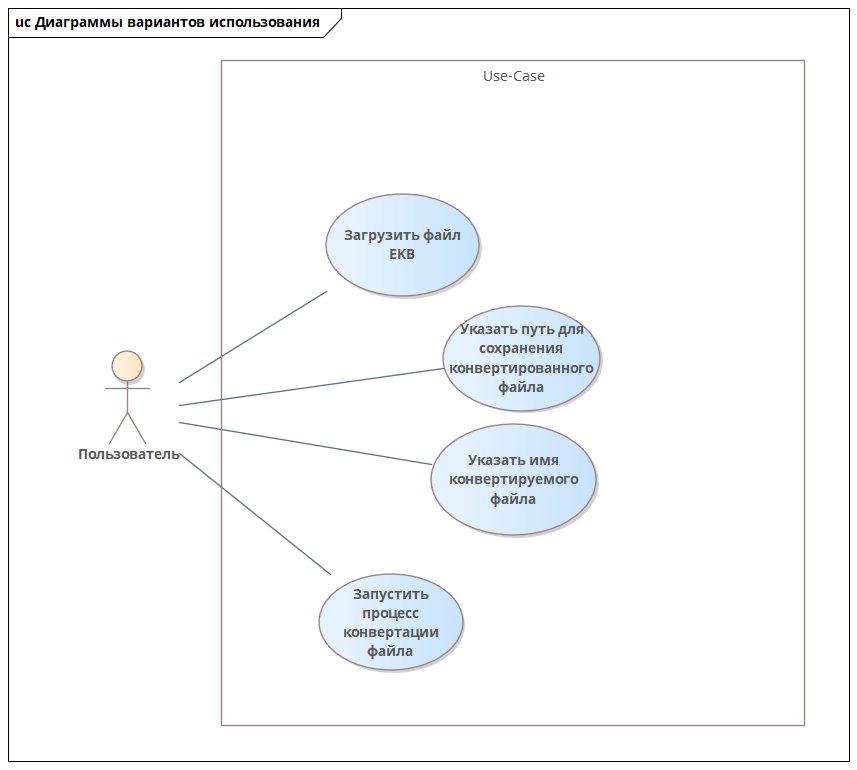


Рисунок 11 – Диаграмма вариантов использования программного компонента

Основываясь на разработанной диаграмме, были описаны основные сценарии вариантов использования приложения.

**Спецификация варианта использования «Загрузить файл EKB»**

**Цель**: загрузка файла EKB для дальнейшей конвертации

**Активные субъекты**: пользователь

**Краткое описание**: пользователь загружает файл EKB посредством меню загрузки файла.

**Основной поток событий**:

* 1. Пользователь запускает механизм загрузки файла EKB.
  2. Приложение открывает окно интерфейса-меню выбора файла.
  3. Пользователь указывает путь до необходимого файла с помощью меню загрузки файла.
  4. Система записывает полный пуль до файла в заранее определенную переменную.
  5. Пользователь получает уведомление об успешной загрузке файла.

**Альтернативные потоки событий:**

1. Активация сценария «Указать путь места для сохранения файла».

**Специальные требования**: нет.

**Предусловия**: нет.

**Постусловия**: после активации варианта должен быть активирован один или несколько сценариев вариантов использования: «Указать путь места для сохранения файла», «Указать имя конвертируемого файла» или «Запустить процесс конвертации файла».

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл EKB» и «Указать путь места для сохранения файла».

**Спецификация варианта использования «Указать путь места для сохранения файла»**

**Цель**: указание места сохранения файла.

**Активные субъекты**: пользователь.

**Краткое описание**: пользователь указывает директорию для сохранения файла посредством меню сохранения файла.

**Основной поток событий**:

* 1. Пользователь запускает механизм выбора места для сохранения.
  2. Приложения открывает окно интерфейса-меню выбора папки.
  3. Пользователь указывает путь до необходимой папки с помощью меню.
  4. Система записывает полный пуль в заранее определенную переменную.
  5. Пользователь получает уведомление об успешном выборе места сохранения файла.

**Альтернативные потоки событий:**

1. Активация сценария «Загрузить файл EKB».

**Специальные требования**: нет.

**Предусловия**: нет.

**Постусловия**: после активации варианта должен быть активирован один или несколько сценариев вариантов использования: «Загрузить файл EKB» или «Запустить процесс конвертации файла».

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл EKB», «Указать имя конвертируемого файла» и «Указать путь места для сохранения файла».

**Спецификация варианта использования «Указать имя конвертируемого файла»**

**Цель**: указание имени конвертированного файла.

**Активные субъекты**: пользователь.

**Краткое описание**: пользователь указывает имя для файла.

**Основной поток событий**:

1. Пользователь вводит имя файла в специальное поле в окне приложения.
2. При запуске механизма конвертации система обращается к текстовому полю для получения имени файла.
3. Система присваивает введенное имя конечному файлу.

**Альтернативные потоки событий:** нет.

**Специальные требования**: нет.

**Предусловия**: нет.

**Постусловия**: нет.

**Дополнительные замечания**: в случае игнорирования пользователем поля имени файла система присвоит конвертированному файлу имя по умолчанию.

**Спецификация варианта использования «Запустить процесс конвертации файла»**

**Цель**: конвертация файла.

**Активные субъекты**: пользователь.

**Краткое описание**: пользователь запускает механизм конвертации файла.

**Основной поток событий**:

* 1. Пользователь запускает механизм загрузки файла EKB.
  2. Приложения открывает окно интерфейса-меню выбора файла.
  3. Пользователь указывает путь до необходимого файла с помощью меню загрузки файла.
  4. Система записывает полный пуль до файла в заранее определенную переменную.
  5. Пользователь получает уведомление об успешной загрузке файла.
  6. Пользователь запускает механизм выбора папки для сохранения.
  7. Приложения открывает окно интерфейса-меню выбора папки.
  8. Пользователь указывает путь до необходимой директории с помощью меню.
  9. Система записывает полный пуль до конечной папки в заранее определенную переменную.
  10. Пользователь получает уведомление об успешном выборе места сохранения файла.
  11. Пользователь запускает конвертацию файла.
  12. Файл загружается в приложение из указанного пути.
  13. Создается объект класса EKB с полями файла.
  14. Данные из файла заносятся в соответствующие переменные объекта класса.
  15. Создается объект класса Draw.io.
  16. Поля класса EKB и Draw.io соотносятся и перезаписываются
  17. Конечный объект Draw.io дополняется необходимыми конструкциями для работоспособности файла.
  18. Объект Draw.io с дополненными конструкциями сохраняется в заданной пользователем папке с указанным именем.
  19. Пользователь получает уведомление об успешной конвертации файла.

**Альтернативные потоки событий:** нет.

**Специальные требования**: нет.

**Предусловия**:

1. Выполнен сценарий «Загрузить файл EKB».
2. Выполнен сценарий «Указать путь места для сохранения файла».

**Постусловия**: нет.

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл EKB» и «Указать путь места для сохранения файла».

2.3.2 Диаграмма последовательностей

На основе диаграммы вариантов использования была составлена диаграмма последовательностей.

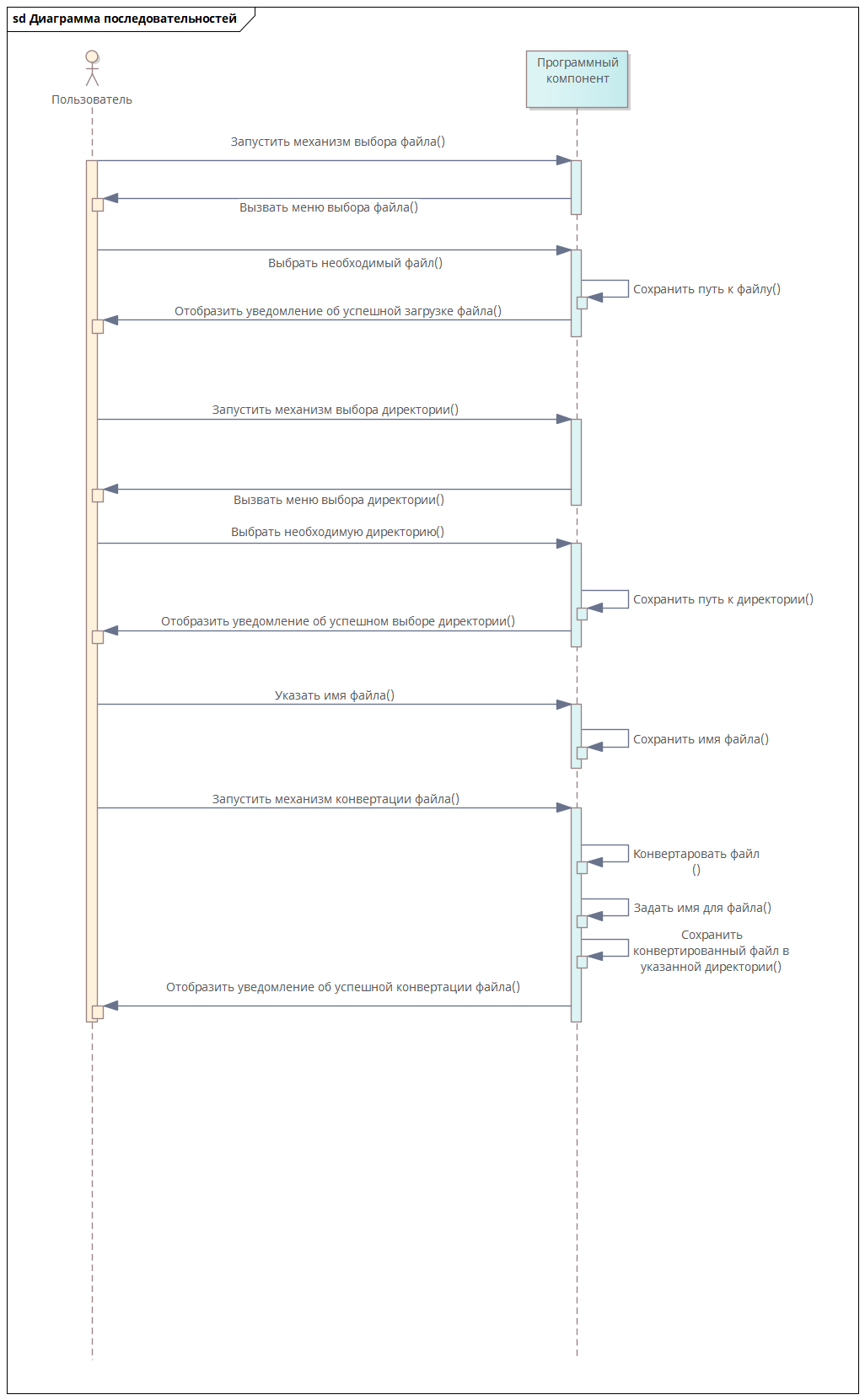


Рисунок 12 – Диаграмма последовательностей

Пользователь с помощью меню выбора файла операционной системы задает приложению директорию для загрузки файла и папку для сохранения результата конвертации. Далее пользователь, по личному усмотрению, задает конечному файлу имя в объект формы (текстовое поле ввода). Следующим шагом участник системы запускает механизм конвертации файла посредством нажатия на соответствующий объект графического интерфейса (кнопку). В результате в директории создается файл с соответствующим расширением.

2.3.3 Алгоритмическое обеспечение

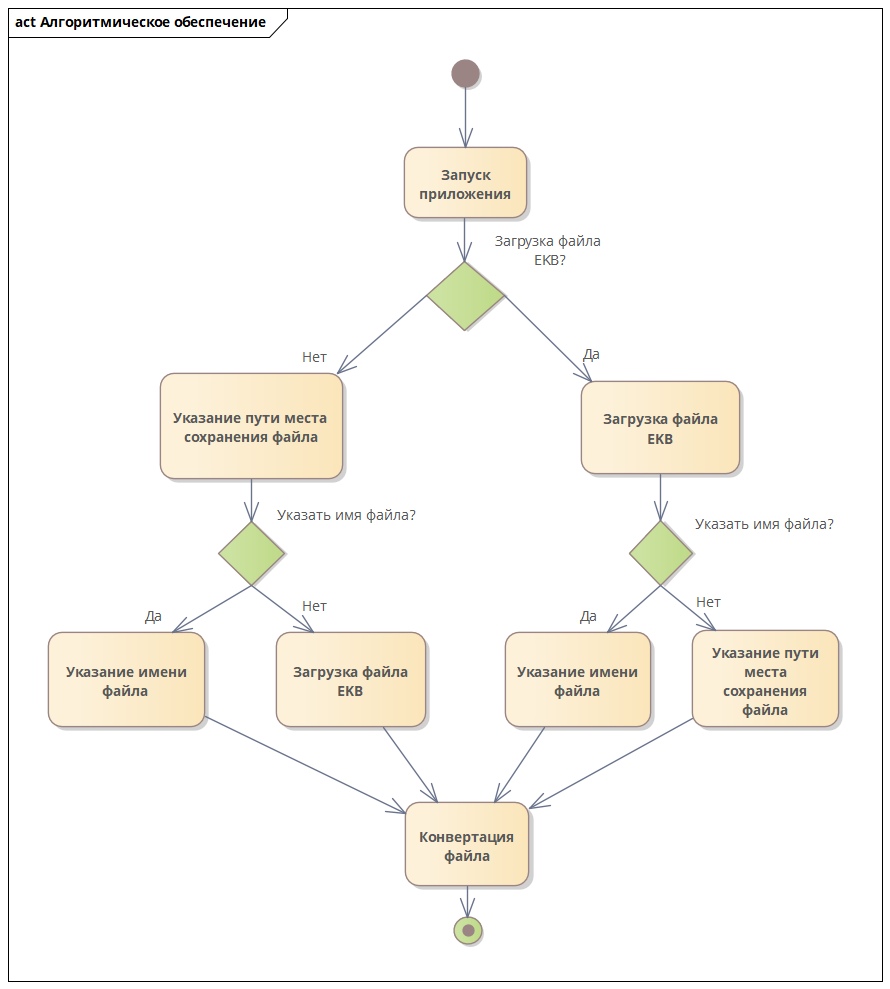


Рисунок 13 – Диаграмма деятельности

Первым этапом после инициализации деятельности выступает блок «Запуск приложения». На данном этапе пользователь запускает программный компонент. После участнику системы представляется выбор очередности действия. В случае выполнения первого вариант пользователь укажет путь к директории для сохранения конвертированного файла, далее загрузит файл для конвертации. В противном случае, пользователь сначала загрузит файл для работы, а после укажет путь к месту сохранения. В каждом ветвлении присутствует дополнительное расхождение. Это обусловлено вариативностью, возникшей при появлении текстового поля для ввода имени файла. Ветвления сходятся в блоке «Конвертация файла», в котором осуществляется указанный в названии механизм. Данный блок является финальным.

2.3.4 Диаграмма классов

На основе разработанной проектной документации была создана диаграмма классов программного компонента.

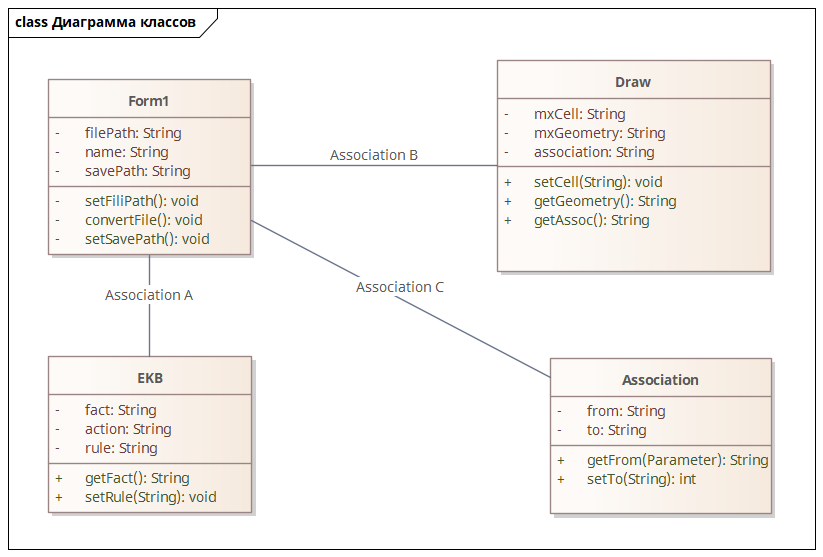


Рисунок 14 – Диаграмма классов

Диаграмма представляет из себя четыре класса, необходимых для полноценного функционирования приложения с целью выполнения поставленных задач. Класс Form1 представляет из себя контроллер графического приложения. Основными переменными являются: путь до папки с файлом и директория для сохранения. Соответствующие методы предназначены для получения необходимых местоположений. Дополнительно спроектирован метод, реализующий механизм конвертации файла.

Классы EKB и Draw предназначены для хранения данных из файлов. При запуске процесса конвертации файл EKB будет декомпозирован в отдельные переменные объекта класса. Далее полученные значения будут пересобраны в экземпляре Draw и дополнены необходимыми для структурной целостности файла данными.

Дополнительно спроектирован класс Association. Данный класс необходим для хранения информации о связях между классами.

Спроектированные блоки условных data-классов представлены в виде демонстрационных моделей: отражают неполные наборы переменных и методов getter – setter, так как представленного наполнения достаточно для генерации первичного скелетного кода для последующего расширения.

2.4 Программная реализация

2.4.1 Автоматически сгенерированный код

На основе разработанной диаграммы классов был сгенерирован код. В качестве выбранного языка программирования был установлен C#.

**Код класса Form1**

///////////////////////////////////////////////////////////

// Form1.cs

// Implementation of the Class Form1

// Generated by Enterprise Architect

// Created on: 19-дек-2022 13:02:57

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class Form1 {

private String filePath;

private String name;

private String savePath;

public Association m\_Association;

public EKB m\_EKB;

public Draw m\_Draw;

public Form1(){

}

~Form1(){

}

///

/// <param name="Parameter A"></param>

private void setFiliPath(Parameter A){

}

private void convertFile(){

}

///

/// <param name="Parameter"></param>

private void setSavePath(Parameter){

}

///

/// <param name="Parameter A"></param>

private void setFiliPath(Parameter A){

}

///

/// <param name="Parameter"></param>

private void setSavePath(Parameter){

}

}//end Form1

**Код класса EKB**

///////////////////////////////////////////////////////////

// EKB.cs

// Implementation of the Class EKB

// Generated by Enterprise Architect

// Created on: 19-дек-2022 13:04:02

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class EKB {

private String fact;

private String action;

private String rule;

public EKB(){

}

~EKB(){

}

///

/// <param name="Parameter A"></param>

public String getFact(Parameter A){

return "";

}

///

/// <param name="Parameter"></param>

public void setRule(String Parameter){

}

///

/// <param name="Parameter A"></param>

public String getFact(Parameter A){

return "";

}

}//end EKB

**Код класса Draw**

///////////////////////////////////////////////////////////

// Draw.cs

// Implementation of the Class Draw

// Generated by Enterprise Architect

// Created on: 19-дек-2022 13:04:50

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class Draw {

private String mxCell;

private String mxGeometry;

private String association;

public Draw(){

}

~Draw(){

}

///

/// <param name="Parameter A"></param>

public void setCell(String Parameter A){

}

///

/// <param name="Parameter"></param>

public String getGeometry(Parameter){

return "";

}

public String getAssoc(){

return "";

}

}//end Draw

**Код класса Association**

///////////////////////////////////////////////////////////

// Association.cs

// Implementation of the Class Association

// Generated by Enterprise Architect

// Created on: 19-дек-2022 13:04:06

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class Association {

private String from;

private String to;

public Association(){

}

~Association(){

}

///

/// <param name="A"></param>

public String getFrom(Parameter A){

return "";

}

///

/// <param name="parameter"></param>

public int setTo(String parameter){

return 0;

}

}//end Association

2.4.2 Описание интерфейса

В качестве среды для создания макетов графического интерфейса был использован онлайн-редактор для создания интерфейсов и прототипов Figma. Основываясь на диаграмме вариантов использования, было спроектировано пять прототипов возможных состояний программного компонента.

Первый фрагмент отображает первоначальное состояние приложения после запуска. Пользователю предоставляется возможность загрузить файл для трансляции, выбрать место для сохранения результата, указать имя результирующего файла.

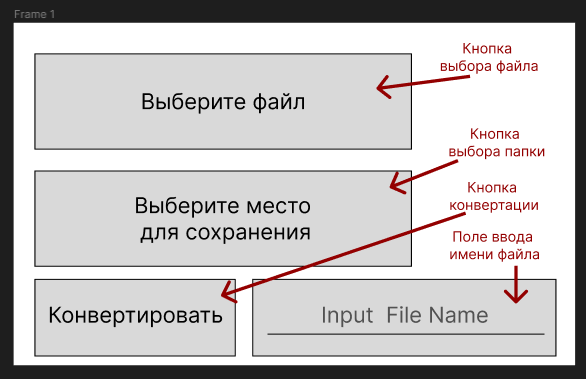


Рисунок 15 – Макет стартового окна

При нажатии на кнопки загрузки или выбора директории открывается окно соответствующего выбора.

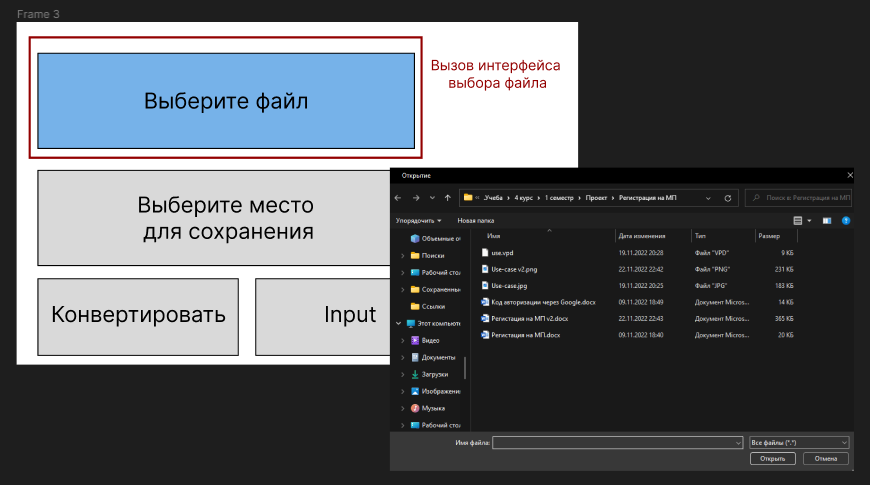


Рисунок 16 – Взаимодействие с кнопками выбора

Приложение визуально отображает этапы работы в процессе подготовки. В случае успеха загрузки файла или выбора директории текст на соответствующем элементе управления изменится. Дополнительно отобразится графический элемент статуса. При альтернативном исходе внешний вид также изменяется.

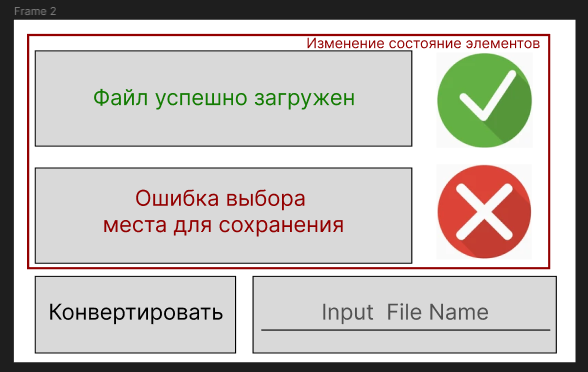


Рисунок 17 – Отображение состояния приложения

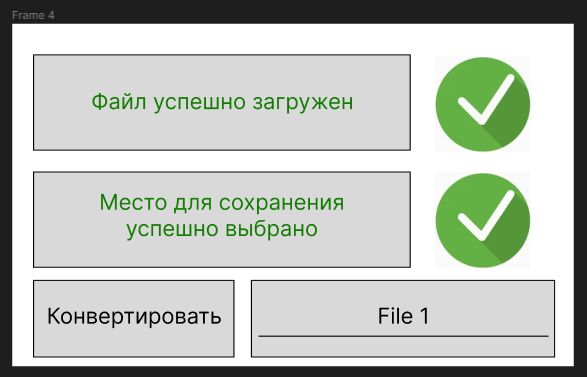


Рисунок 18 – Успешный подготовительный этап

После завершения данного процесса отображается окно с сообщением об успешной конвертацией или ошибкой.

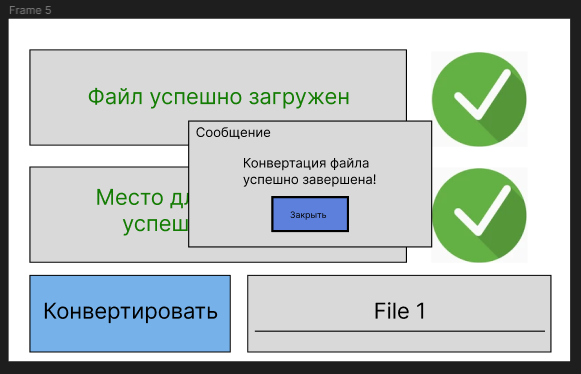


Рисунок 19 – Визуализация завершения процесса

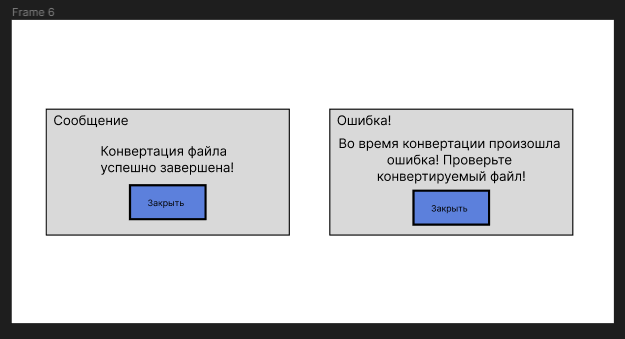


Рисунок 20 – Вариации окна с сообщением

Конечный вариант графического интерфейса создан с помощью элементов окна Windows Forms.

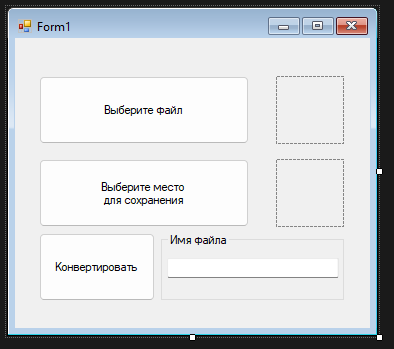


Рисунок 21 – Графический интерфейс в среде разработки

2.4.3 Тестирование

Для тестирования разработанного программного компонента была использована компромиссный стратегия. Компромиссная стратегия – проектирование тестов, исходя из принципов:

1. проверка функции или возможности;
2. проверка каждой области и границы изменения значений какой-либо входной величины;
3. проверка каждого особого случая или исключительной ситуации;
4. каждая команда программы должна проработать хотя бы на одном тесте.

Таблица 2 – Тесты программного компонента

|  |  |  |  |
| --- | --- | --- | --- |
| N | Описание проверяемой ситуации | Тест (входные данные) | Результат |
|  | Проверка работы компонента со стандартным файлом. | Полученный файл с данными. | Конвертированный файл с расширением xml. |
|  | Проверка работы компонента с файлом нестандартной конфигурации. | Файл нестандартной конфигурации. | Окно с уведомлением об ошибке конвертации. |
|  | Попытка конвертировать файл без указания директории для сохранения. | Указан путь к файлу с данными. | Кнопка конвертации сохранила неактивное состояние. |
|  | Попытка конвертировать файл без указания файла для конвертации. | Указана директория для сохранения файла. | Кнопка конвертации сохранила неактивное состояние. |
|  | Проверка работы компонента с файлом нестандартного для приложения компонента. | Файл с расширением xml. | Окно с уведомлением об ошибке конвертации. |
|  | Проверка работы компонента с пустым файлом. | Пустой файл с данными. | Окно с уведомлением об ошибке конвертации. |
|  | Повторная конвертация загруженного файла без изменения директории. | Полученный файл с данными. | Перезаписанный файл с расширением xml. |
|  | Проверка работы компонента с расширенным файлом. | Расширенный файл с данными. | Конвертированный файл с расширением xml. |

Результаты тестирования представлены на рисунках ниже.

**Тест №1 – Проверка работы компонента со стандартным файлом.**

Для выполнения первого теста созданная в Enterprise Architect схема была воссоздана в Personal knowledge Base Designer.

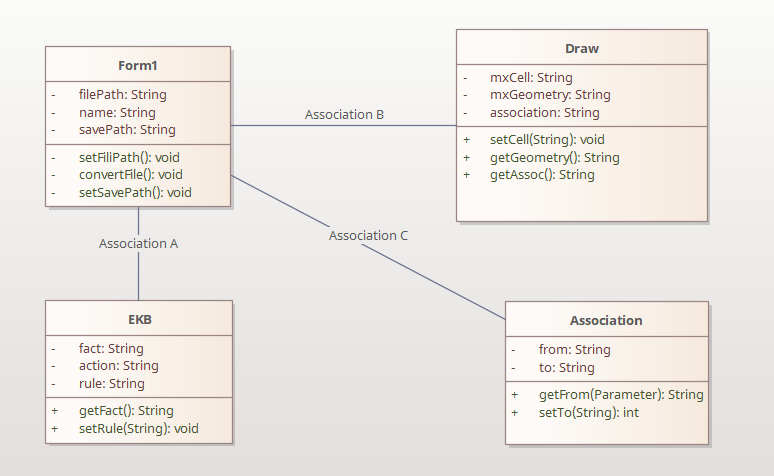


Рисунок 22 – Оригинальная схема

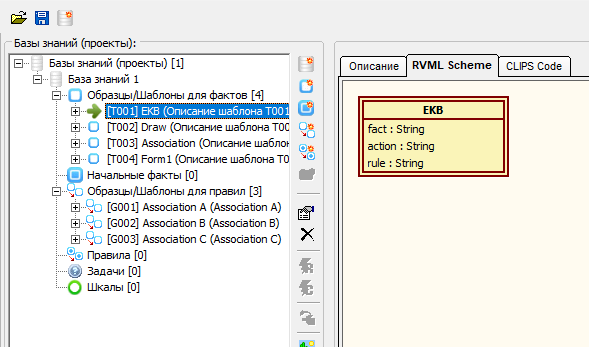


Рисунок 23 – Воссозданная схема

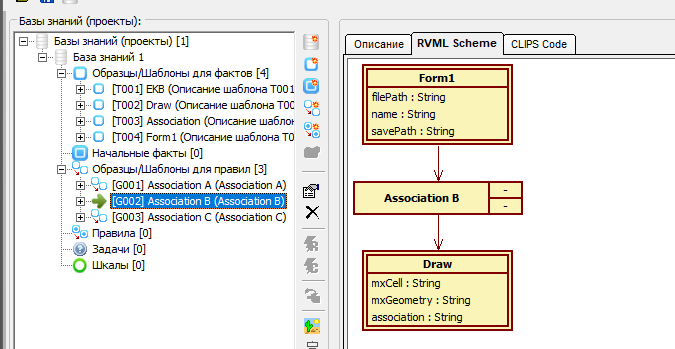


Рисунок 24 – Воссозданная схема

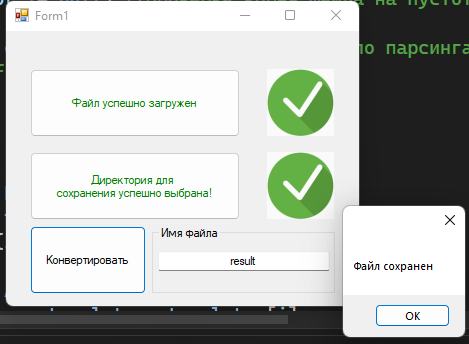


Рисунок 25 – Работа компонента



Рисунок 26 – Файл с результатом

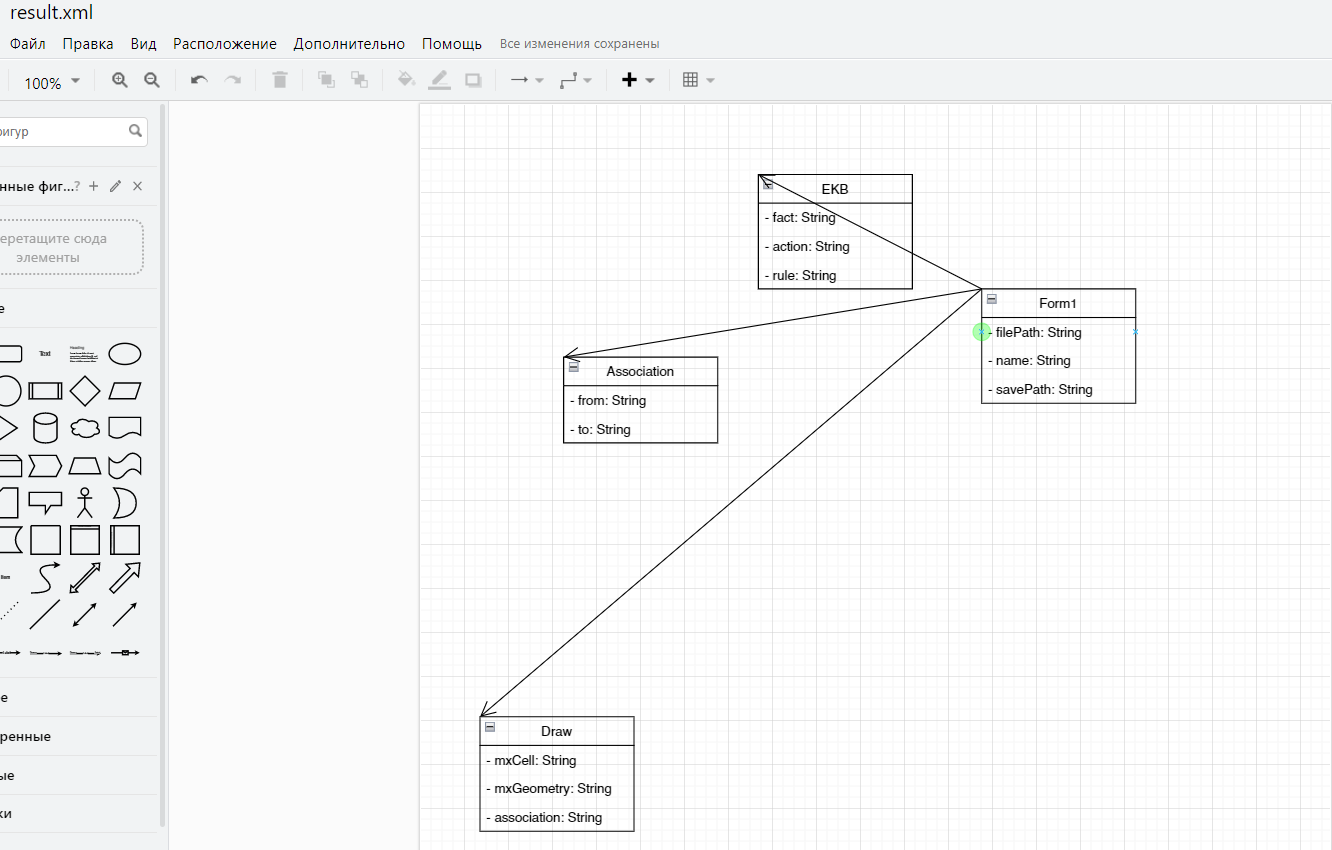


Рисунок 27 – Результат работы компонента

**Тест №2 - Проверка работы компонента с файлом нестандартной конфигурации.**

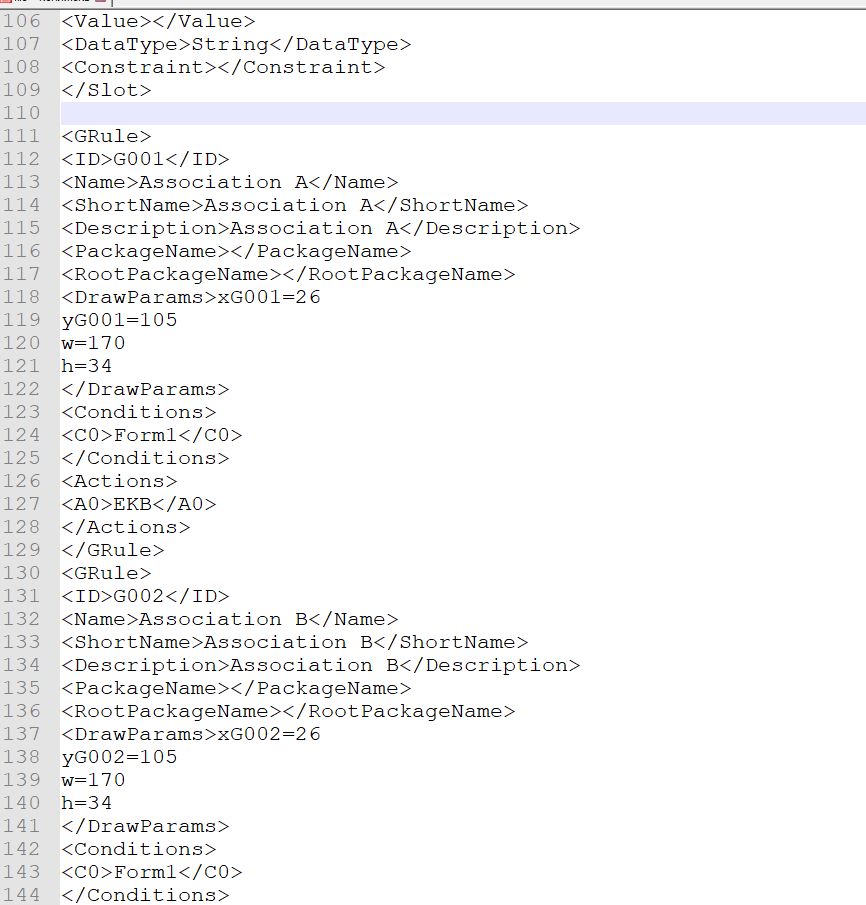


Рисунок 29 – Содержимое файла нестандартной конфигурации

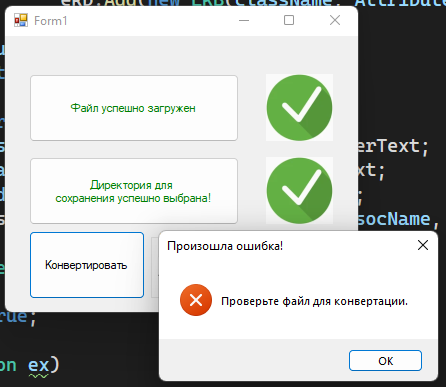


Рисунок 30 – Результат выполнения теста

**Тест №3 – Попытка конвертировать файл без указания директории для сохранения.**

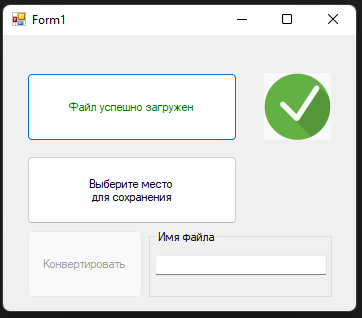


Рисунок 31 – Результат выполнения теста

**Тест 4 – Попытка конвертировать файл без указания файла для конвертации.**

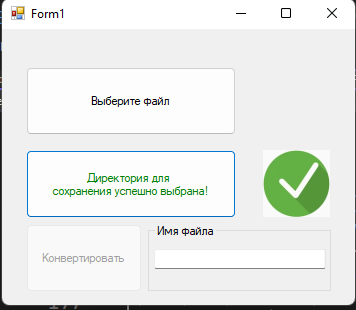


Рисунок 32 – Результат выполнения теста

**Тест №5 – Проверка работы компонента с файлом нестандартного для приложения компонента.**

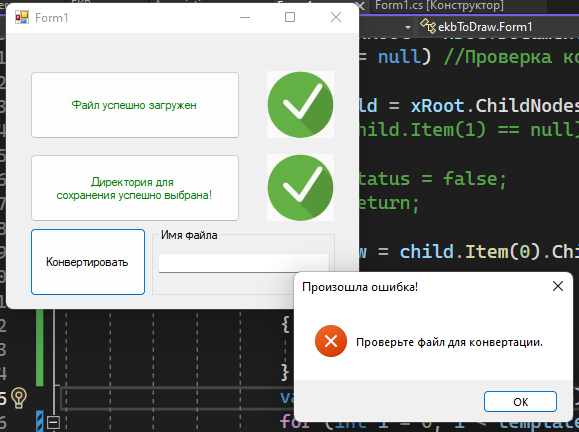


Рисунок 33 – Результат выполнения теста

**Тест №6 – Проверка работы компонента с пустым файлом.**

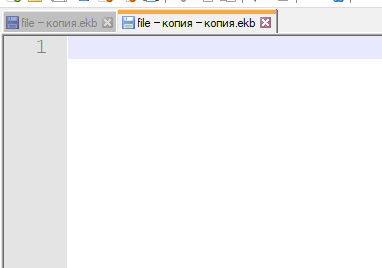


Рисунок 34 – Пустой файл для конвертации

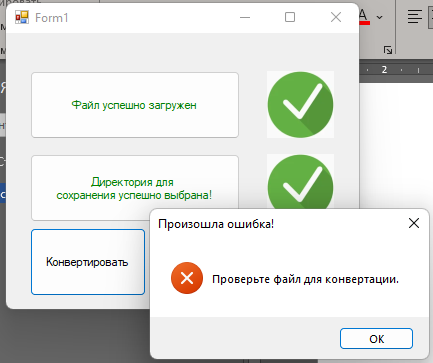


Рисунок 35 – Результат выполнения теста

**Тест №7 – Повторная конвертация загруженного файла без изменения директории.**

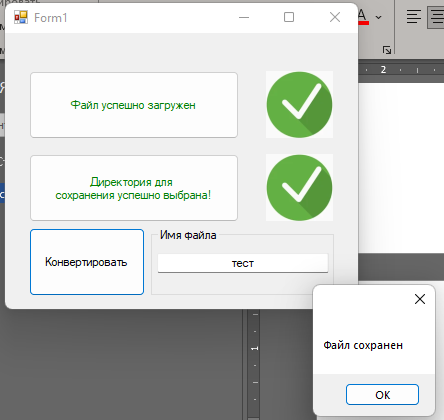


Рисунок 36 – Результат выполнения теста

**Тест №8 – Проверка работы компонента с расширенным файлом.**

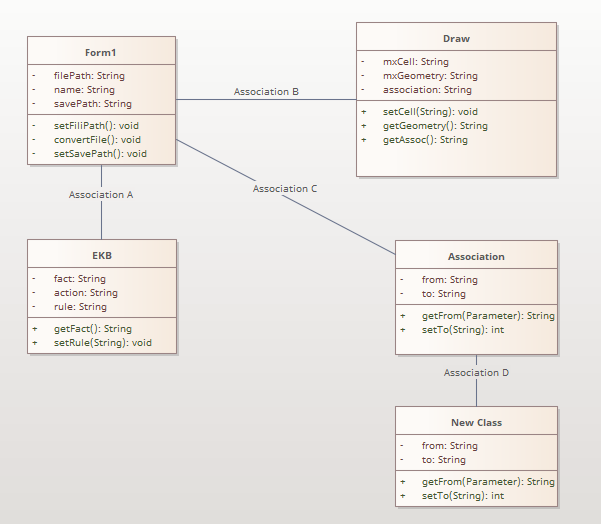


Рисунок 37 – Дополненная структура

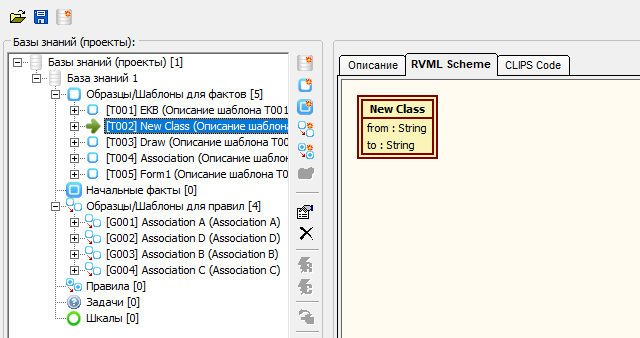


Рисунок 38 – Новый класс

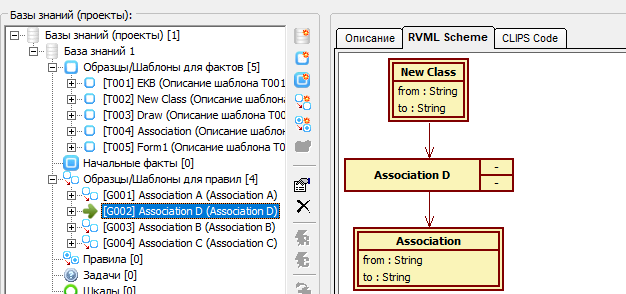


Рисунок 39 – Новая связь

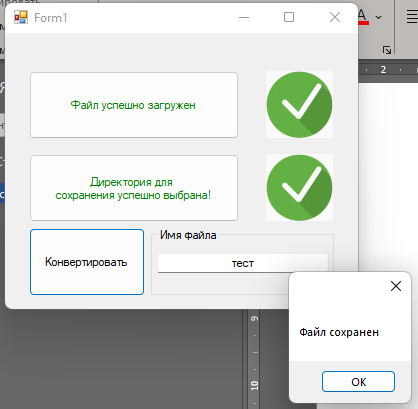


Рисунок 40 – Работа компонента

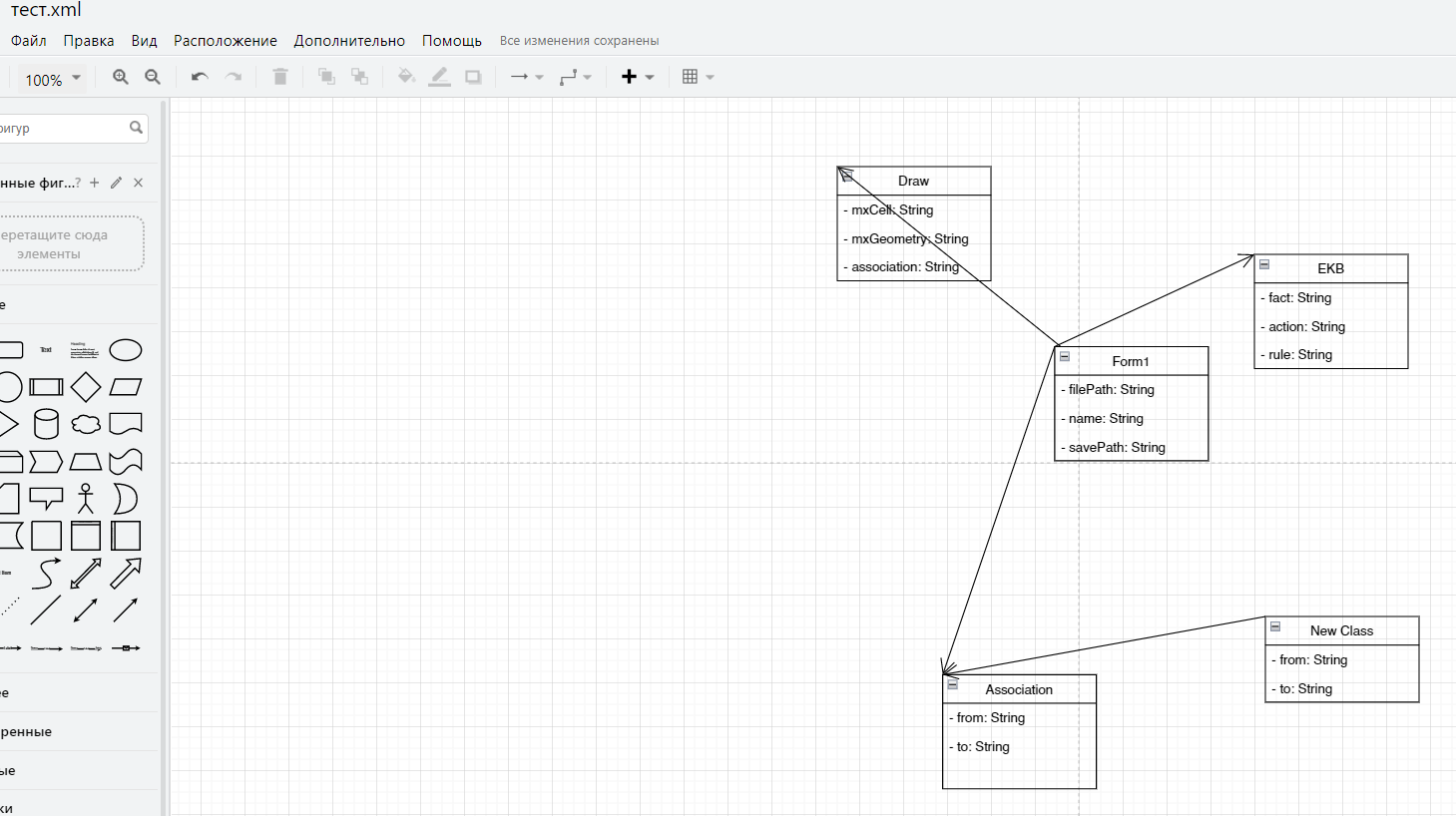


Рисунок 41 – Результат работы компонента

Заключение

В ходе выполнения проекта было разработано программное обеспечение, обеспечивающее автоматизированное преобразование файлов EKB в формат приложения для создания диаграмм и интерактивных досок draw.io.

Составлено техническое задание для выполнения задания в рамках курсового проекта, отвечающее стандартам и требованиям ГОСТ, описан состав и содержание работы, составлены Перт диаграмма и диаграмма Ганта.

Спроектирована доска в специализированном программном обеспечении Trello для отслеживания процесса выполнения задания в рамках курсового проекта.

С помощью CASE-средства Enterprise Architect спроектированы:

1. Диаграммы вариантов использования в нотации UML.
2. Описаны основные варианты использования разработанного программного компонента.
3. Разработана системная диаграмма последовательности действий.
4. Разработана диаграмма классов уровня проектирования в нотации UML.
5. Произведена генерация скелетного кода на основе диаграмм классов для целевого языка программирования.

Описан графический интерфейс и разработана программная реализация приложения. Разработанное приложение успешно прошло составленные тесты.

Список использованных источников

1. Гутгарц Р.Д. Проектирование автоматизированных систем обработки информации и управления. Методические указания по выполнению курсового проекта. Иркутск, ИрГТУ, 2010. 57 с.
2. Коваленко В.В. Проектирование информационных систем: учеб. пособие. Иркутск: Форум, НИЦ ИНФРА. М, 2014.  320 с.
3. Исаев Г.Н. Проектирование информационных систем: Учебное пособие/ Москва: Изд-во Омега Л, 2015. 424 с.
4. Кватрани Т. RationalRose 2000 и UML. Визуальное моделирование. М.: ДМК Пресс, 2001. 176 с.
5. Дородных Н.О., Юрин А.Ю. Технология создания продукционных экспертных систем на основе модельных трансформаций. Новосибирск: СО РАН, 2019. 144 стр.
6. Юрин А.Ю., Грищенко М.А. Редактор баз знаний в формате CLIPS // Программные продукты и системы. 2012. № 4. С. 83–87.
7. Юрин А.Ю. CASE-средства: Методические указания по выполнению лабораторных работ. Иркутск: ИРНИТУ, 2018. 87 c.

Приложение А

Фрагмент сгенерированной документации

**ActivityFinal**

ActivityFinal in package 'Алгоритмическое обеспечение'

| INCOMING BEHAVIORAL RELATIONSHIPS |
| --- |
| ControlFlow from Конвертация файла to ActivityFinal |

**ActivityInitial**

ActivityInitial in package 'Алгоритмическое обеспечение'

| OUTGOING BEHAVIORAL RELATIONSHIPS |
| --- |
| ControlFlow from ActivityInitial to Запуск приложения |

**ActivityInitial**

ActivityInitial in package 'Алгоритмическое обеспечение'

| OUTGOING BEHAVIORAL RELATIONSHIPS |
| --- |
| ControlFlow from ActivityInitial to Запуск приложения |

**ActivityFinal**

ActivityFinal in package 'Алгоритмическое обеспечение'

| INCOMING BEHAVIORAL RELATIONSHIPS |
| --- |
| ControlFlow from Завершение работы приложения to ActivityFinal |

**Action1**

Action in package 'Алгоритмическое обеспечение'

Action1

Version 1.0 Phase 1.0 Proposed

worker created on 20.11.2022. Last modified 20.11.2022

| INCOMING BEHAVIORAL RELATIONSHIPS |
| --- |
| ControlFlow from Запуск приложения to Action1 |

Приложение Б

Листинг программы

**Класс Form1**

public partial class Form1 : Form

{

string filename; //Путь к файлу

string pathToDid; //Путь к директории

bool status = false;

string drawText = "";

//Листы объектов классов

List<EKB> ekb;

List<Association> associations;

string deepId = "";

List<(string, string, int, int)> dataList;

int sourseX = 0;

int sourseY = 0;

int targetX = 0;

int targetY = 0;

public Form1()

{

InitializeComponent();

openFileDialog1.Filter = "Файл ekb (\*.ekb)|\*.ekb|All files(\*.\*)|\*.\*"; //Фильтр файлов

convertBtn.Enabled = false;

ekb = new List<EKB>();

associations = new List<Association>();

}

private void fileBtn\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

{

fileBtn.Text = "Ошибка загрузки файла!";

fileBtn.ForeColor = Color.Red;

fileImg.Image = ekbToDraw.Properties.Resources.error;

filename = null;

}

else

{

// получаем выбранный файл

filename = openFileDialog1.FileName;

fileBtn.Text = "Файл успешно загружен";

fileImg.Image = ekbToDraw.Properties.Resources.ok;

fileBtn.ForeColor = Color.Green;

}

checkToReady();

}

private void pathBtn\_Click(object sender, EventArgs e)

{

FolderBrowserDialog FBD = new FolderBrowserDialog();

FBD.ShowNewFolderButton = false;

if (FBD.ShowDialog() == DialogResult.OK)

{

pathBtn.Text = "Директория для \r\nсохранения успешно выбрана!";

pathImg.Image = ekbToDraw.Properties.Resources.ok;

pathBtn.ForeColor = Color.Green;

pathToDid = FBD.SelectedPath;

}

else

{

pathBtn.Text = "Ошибка выбора \r\nдиректории для сохранения";

pathImg.Image = ekbToDraw.Properties.Resources.error;

pathToDid = null;

pathBtn.ForeColor = Color.Red;

}

checkToReady();

}

private void checkToReady() //Метод проверки выбора директории и файла

{

if (filename != null && pathToDid != null)

{

convertBtn.Enabled = true;

}

else

{

convertBtn.Enabled = false;

}

}

private void convertBtn\_Click(object sender, EventArgs e)

{

drawText = "";

parseTheFile();

convertToDraw(); //Метод преобразования файла в формат EKB

if (status)

{

saveToDraw();

}

else

{

MessageBox.Show("Проверьте файл для конвертации.", "Произошла ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

ekb.Clear();

associations.Clear();

dataList.Clear();

}

private void parseTheFile() //Метод парсинга файла

{

try

{

string attr = "";

string className = "";

string start = "";

string end = "";

string typeText = "";

string assocName = "";

List<(string, string)> Attribute;

filename = openFileDialog1.FileName;

XmlDocument xDoc = new XmlDocument();

xDoc.Load(filename);

XmlElement xRoot = xDoc.DocumentElement;

if (xRoot != null) //Проверка корня файла на пустоту

{

var child = xRoot.ChildNodes; //Начало парсинга

var know = child.Item(0).ChildNodes;

if(know.Item(6) == null)

{

return;

}

var templates = know.Item(6).ChildNodes;

for (int i = 0; i < templates.Count; i++)

{

Attribute = new List<(string, string)>();

var template = templates[i];

className = template.ChildNodes[1].InnerText;

var templateInside = template.ChildNodes;

var check = templateInside.Item(7);

if(check == null)

{

return;

}

var slots = templateInside.Item(7).ChildNodes;

for (int j = 0; j < slots.Count; j++)

{

var slot = slots[j];

attr = slot.ChildNodes[0].InnerText;

typeText = slot.ChildNodes[4].InnerText;

Attribute.Add((attr, typeText));

}

ekb.Add(new EKB(className, Attribute));

}

var grules = know.Item(8).ChildNodes;

for(int i = 0; i < grules.Count; i++)

{

var grule = grules[i];

assocName = grule.ChildNodes[1].InnerText;

start = grule.ChildNodes[7].InnerText;

end = grule.ChildNodes[8].InnerText;

associations.Add(new Association(assocName, start, end));

}

Console.WriteLine();

}

status = true;

}

catch (Exception ex)

{

status = false;

return;

}

}

private void convertToDraw() //Метод преобразования файла в формат EKB

{

dataList = new List<(string, string, int, int)>();

var drawHeader = $"<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +

$"\r\n<mxfile host=\"app.diagrams.net\" modified=\"2022-12-12T10:43:19.882Z\" agent=\"5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 YaBrowser/22.11.2.807 Yowser/2.5 Safari/537.36\" etag=\"o0eCUbzZ9Tfc7CzTUTEk\" version=\"20.6.2\" type=\"device\">" +

$"\r\n <diagram id=\"C5RBs43oDa-KdzZeNtuy\" name=\"Page-1\">" +

$"\r\n <mxGraphModel dx=\"1247\" dy=\"694\" grid=\"1\" gridSize=\"10\" guides=\"1\" tooltips=\"1\" connect=\"1\" arrows=\"1\" fold=\"1\" page=\"1\" pageScale=\"1\" pageWidth=\"827\" pageHeight=\"1169\" math=\"0\" shadow=\"0\">" +

$"\r\n <root>";

drawText += drawHeader;

var drawBody = $" \r\n<mxCell id=\"WIyWlLk6GJQsqaUBKTNV-0\" />" +

$"\r\n <mxCell id=\"WIyWlLk6GJQsqaUBKTNV-1\" parent=\"WIyWlLk6GJQsqaUBKTNV-0\" />";

drawText += drawBody;

int count = 0;

var x = 0;

var y = 0;

Random rnd = new Random();

for (int i = 0; i < ekb.Count; i++)

{

x = rnd.Next(500) + rnd.Next(-300, 100);

y = rnd.Next(400) + rnd.Next(-400, 323);

string id = $"x-kl-r7OIvYpWUdUMo4z-{count}";

dataList.Add((id, ekb[i].ClassName, x, y));

int paramCount = 26;

var classText = $"\r\n<mxCell id=\"{id}\" value=\"{ekb[i].ClassName}\" style=\"swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;startSize=26;fillColor=none;horizontalStack=0;resizeParent=1;resizeParentMax=0;resizeLast=0;collapsible=1;marginBottom=0;\" vertex=\"1\" parent=\"WIyWlLk6GJQsqaUBKTNV-1\">" +

$"\r\n <mxGeometry x=\"{x}\" y=\"{y}\" width=\"140\" height=\"104\" as=\"geometry\" />" +

$"\r\n </mxCell>";

drawText += classText;

count++;

getClassName(ekb[i].ClassName);

for (int j = 0; j < ekb[i].Attribute.Count; j++)

{

var paramText = $"\r\n<mxCell id=\"x-kl-r7OIvYpWUdUMo4z-{count}\" value=\"- {ekb[i].Attribute[j].Item1}: {ekb[i].Attribute[j].Item2}\" style=\"text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;\" vertex=\"1\" parent=\"{deepId}\">" +

$"\r\n <mxGeometry y=\"{paramCount}\" width=\"140\" height=\"26\" as=\"geometry\" />" +

$"\r\n</mxCell>";

count++;

paramCount += 26;

drawText += paramText;

}

}

for (int j = 0; j < associations.Count; j++)

{

getClassCoord(associations[j].SourceName, associations[j].TargetName);

var assocText = $" \r\n<mxCell id=\"x-kl-r7OIvYpWUdUMo4z-{count}\" value=\"\" style=\"endArrow=open;endFill=1;endSize=12;html=1;rounded=0;exitX=1.019;exitY=0.108;exitDx=0;exitDy=0;exitPerimeter=0;entryX=-0.02;entryY=0.13;entryDx=0;entryDy=0;entryPerimeter=0;\" edge=\"1\" parent=\"WIyWlLk6GJQsqaUBKTNV-1\" source=\"{associations[j].SourceName}\" target=\"{associations[j].TargetName}\">" +

$"\r\n <mxGeometry width=\"160\" relative=\"1\" as=\"geometry\">" +

$"\r\n <mxPoint x=\"{sourseX}\" y=\"{sourseY}\" as=\"sourcePoint\" />" +

$"\r\n <mxPoint x=\"{targetX}\" y=\"{targetY}\" as=\"targetPoint\" />" +

$"\r\n </mxGeometry>" +

$"\r\n </mxCell>";

count++;

drawText += assocText;

}

var drawFooter = $" \r\n</root>" +

$"\r\n </mxGraphModel>" +

$"\r\n </diagram>" +

$"\r\n</mxfile>";

drawText+= drawFooter;

}

private void getClassName(string className)

{

foreach(var data in dataList)

{

if(data.Item2 == className)

{

deepId = data.Item1;

}

}

}

private void getClassCoord(string className, string classNameTarget)

{

foreach (var data in dataList)

{

if (data.Item2 == className)

{

sourseX = data.Item3;

sourseY = data.Item4;

}

if (data.Item2 == classNameTarget)

{

targetX = data.Item3;

targetY = data.Item4;

}

}

}

private void saveToDraw() //Метод сохранения файла

{

if(fileNameText.Text.Length == 0)

{

string path = $"{pathToDid}/file.xml";

System.IO.File.WriteAllText(path, drawText);

}

else

{

string path = $"{pathToDid}/{fileNameText.Text}.xml";

System.IO.File.WriteAllText(path, drawText);

}

MessageBox.Show("Файл сохранен");

}

}

**Класс EKB**

internal class EKB

{

public string ClassName { get; set; }

public List<(string, string)> Attribute { get; set; }

public EKB(string className, List<(string, string)> attribute)

{

ClassName = className;

Attribute = attribute;

}

}

**Класс Association**

internal class Association

{

public string AssotionName { get; set; }

public string SourceName { get; set; }

public string TargetName { get; set; }

public Association(string assotionName, string sourceName, string targetName)

{

AssotionName = assotionName;

SourceName = sourceName;

TargetName = targetName;

}

}