# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 4](#_Toc470019204)

[ВВЕДЕНИЕ 5](#_Toc470019205)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc470019206)

[1.1 Анализ предметной области 6](#_Toc470019207)

[1.2 Анализ существующих решений 6](#_Toc470019208)

[1.3 Постановка задачи 6](#_Toc470019209)

[2 ПРОЕКТИРОВАНИЕ 10](#_Toc470019210)

[2.1 Выбор архитектуры 10](#_Toc470019211)

[2.2 Диаграмма вариантов использования: 12](#_Toc470019212)

[2.2.1 Подробное описание вариантов использования 15](#_Toc470019213)

[2.3 Диаграмма последовательности 17](#_Toc470019214)

[2.4 Проектирование графического интерфейса 18](#_Toc470019215)

[3 РАЗРАБОТКА 20](#_Toc470019216)

[3.1 Реализация выбранной архитектуры в виде диаграммы пакетов 20](#_Toc470019217)

[3.2 Разработка диаграммы классов 20](#_Toc470019218)

[3.3 Общая диаграмма классов с отображением на архитектуру 21](#_Toc470019219)

[3.4 Выбор алгоритма и оптимизация 22](#_Toc470019220)

[3.5 Описание всех методов 28](#_Toc470019221)

[4 ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ 31](#_Toc470019222)

[4.1 Верификация 31](#_Toc470019223)

[4.2 Тестирование 33](#_Toc470019224)

[ВЫВОДЫ 35](#_Toc470019225)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc470019226)

[ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 37](#_Toc470019227)

[Приложение А. Приложение к Техническому заданию 1 41](#_Toc470019228)

[Приложение А. Приложение к Техническому заданию 2 42](#_Toc470019229)

[ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММЫ 43](#_Toc470019230)

[ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ОПЕРАТОРА 50](#_Toc470019231)

[ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ 54](#_Toc470019232)

# ВВЕДЕНИЕ

Данная работа направлена на решение проблемы ведения документации, а именно проблемы затраты большого количества времени на эту рутинную работу. Люди разных профессий периодически обязаны составлять отчеты, вести учёт каких-либо предметов или выполнять другие виды работы с документами. Этот процесс зачастую занимает немало времени. Поэтому актуальность данной работы заключается именно в решении вышеописанной проблемы.

Целью работы является разработка программного продукта предназначенного для применения в учебных заведениях для облегчения ведения документации при подготовке к проведению учебного процесса. Функциональным назначением (задачей) программы является предоставление преподавателю возможности формировать определённую документацию не в ручном, а в полуавтоматическом режиме.

В результате выполнения данного проекта была разработана программа, которая в качестве входных данных принимает документ «Учебное поручение», а результатом работы является документ «Приложение к семестровому учебному плану».

# 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Анализ предметной области

Почтовый клиент — [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), устанавливаемое на компьютере пользователя и предназначенное для получения, написания, отправки и хранения сообщений [электронной почты](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0) одного или нескольких пользователей.

Общение с почтовыми серверами производиться с помощью протоколов IMAP, SMTP, POP3 и некоторых других. На рисунке 1.1 изображена схема работы постового сервера при отправке и получении почты.

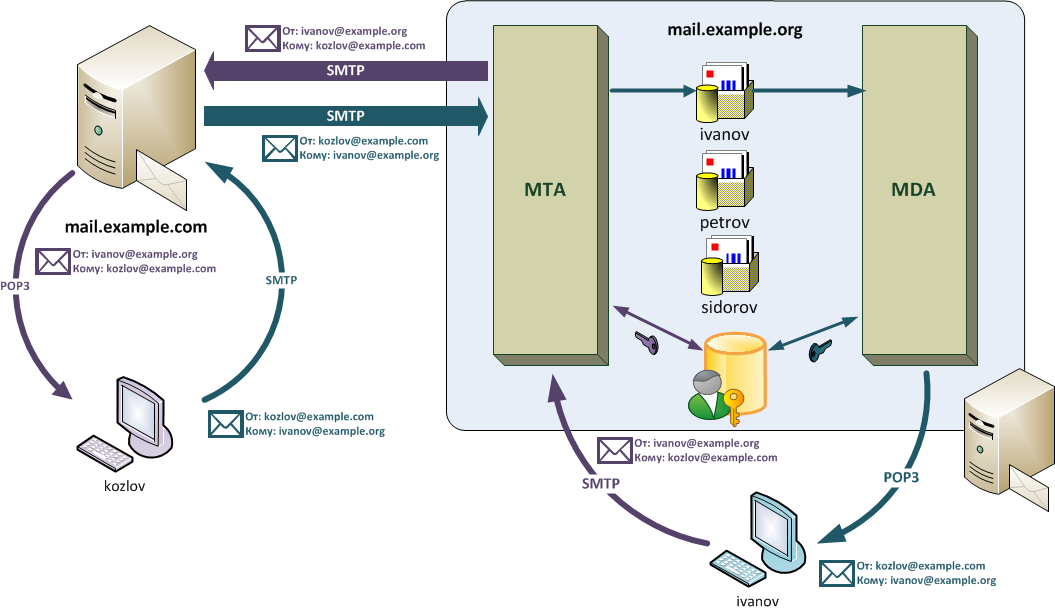


Рисунок 1.1 – Принцип работы почтового сервера

Для решения поставленной задачи было решено выбрать IMAP протокол, так как нам нужно только получение почты. Для этой цели подошёл бы и POP3 протокол, используя его при просмотре почты с использованием этого протокола все электронные письма скачиваются пользователю на компьютер и автоматически удаляются с сервера. А IMAP позволяет управлять электронной почтой прямо на сервере.

## 1.2 Анализ существующих решений

Проблема создания почтового клиента является актуальной и востребованной, поэтому их уже было реализовано большое количество как настольных приложений так и веб-сервисов. На сегодняшний день существуют огромное количество качественных и удобных почтовых клиентов, в том числе и от известных производителей программного обеспечения такого как Microsoft.

Для сравнения возьмем такие почтовые клиенты, как Opera Mail и Mozilla Thunderbird.

Таблица 1.1 – Ср тельная характеристика почтовых клиентов Opera Mail и Mozilla Thunderbird

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Opera Mail** | **Mozilla Thunderbird** |
| Стоимость | бесплатная | бесплатная |
| Русский язык | ✔ | ✔ |
| HTML письма | ✔ | ✔ |
| Анти-спам | ✔ | ✔ |
| Вид диалога | - | ✔ |
| POP3 | ✔ | ✔ |
| IMAP4 | ✔ | ✔ |
| SMTP | ✔ | ✔ |

Оба почтовых клиента имеют удобный интерфейс, просты в работе и хорошо функционируют, однако можно отметить некоторые особенности каждого из них.

Основные особенности программы Opera Mail:

* почтовый клиент поддерживает протоколы smtp, imap, esmtp и pops;
* программа работает с функцией фильтрации спама;
* удобный интерфейс, с удобным размещением окна сообщений;
* комфорт эксплуатации для пользователей операционной системы windows 7 сразу нескольких ящиков почты;
* программа windows оснащена тематической сортировкой электронных писем. Правила распределения сообщений по темам программирует сам пользователь.

Функциональные особенности данного клиента windows:

* mozilla thunderbird поддерживает протоколы smtp, imap и pops, что позволяет совершать эффективную передачу корреспонденции;
* возможность производить удаленный доступ к содержимому почтового сервера;
* простота настройки программного обеспечения;
* одновременное управление большим количеством почтовых ящиков и получение сообщений в одну папку из нескольких аккаунтов;
* программа работает с открытым доступом к передвижению данных из других почтовых программ посредством миграционного помощника;
* удобная система вкладок, структурированная адресная книга;
* мощная система фильтрации корреспонденции и проверки электронной почты;
* возможность работы по проверке орфографии;
* полная поддержка формата HTML в электронных сообщениях.

Как видно из сравнительных характеристик эти почтовые клиенты, как и большинство современных не сильно отличаются по своему функционалу.

Данный курсовой проект не будет представлять из себя конкурентоспособную разработку, так как разрабатываемый почтовый клиент будет работать только на получение писем без возможности их отправки. Несмотря на это уникальными способностями будет возможность разархивирования прикреплённым к письму архивов, а также распределение писем по каталогам в зависимости от отправителя.

## 1.3 Постановка задачи

Разработать ПО почтовый клиент, который будет работать только на получение писем без возможности их отправки. Приложение будет использовать IMAP протокол, с помощью которого можно управлять электронной почтой прямо на почтовом сервере. Поэтому пользователю необходимо активное подключение к интернету.

При запуске программы пользователь должен указать сервер, порт, логин и пароль от своего почтового ящика, а также выбрать использование SSL/TLS.

У данного почтового клиента не будет поддержки HTML сообщений, а только работа с текстовыми сообщениями. Будет возможность разархивировать прикреплённый к сообщению архив.

Главное окно программы будет разделено на 3 части:

– навигация по папкам;

– часть окна где будет выводиться список сообщений в выбранной папке;

– часть окна где будет выводиться тело выбранного сообщения.

Полученные письма будут автоматически распределятся по каталогам в зависимости от отправителя.

# 2 ПРОЕКТИРОВАНИЕ

## 2.1 Выбор архитектуры

Архитектура – это высокоуровневая часть проекта приложения, каркас, состоящий из деталей проекта. Архитектуру также называют «архитектурой системы», «высокоуровневым проектом» и «проектом высокого уровня». Как правило, архитектуру описывают в единственном документе, называемом «спецификацией архитектуры» или «высокоуровневым проектом». Некоторые разработчики проводят различие между архитектурой и высокоуровневым проектом: архитектурой называют характеристики всей системы, тогда как высокоуровневым проектом – характеристики, описывающие

Продуманная архитектура предоставляет структуру, нужную для поддержания концептуальной целостности в масштабе системы. Она предоставляет программистам руководство, уровень детальности которого соответствует их навыкам и выполняемой работе. Она позволяет разделить работу на части, над которыми отдельные разработчики и группы могут трудиться независимо. Хорошая архитектура облегчает конструирование. Плохая архитектура делает его почти невозможным.

В первую очередь архитектура должна включать общее описание системы. Без такого описания вам будет трудно составить согласованную картину из тысячи деталей или хотя бы десятка отдельных классов [1].

Удачная архитектура позволяет:

– сделать возможным/Ускорить/Удешевить достижение бизнес-цели;

– достичь приемлемой производительности/увеличить производительность;

– сделать приложение легко расширяемым на случай, если это понадобится. Надо помнить, что, как правило, расширяемое приложение это простое приложение;

– уменьшить количество ошибок.

Процесс разработки структуры включает в себя создание общей инфраструктуры организации системы и управления ею, выбор протоколов и методов синхронизации и доступа к данным, распределение функций системы между компонентами, физическое распределение, объединение элементов проекта, масштабирование, оптимизацию производительности и выбор оптимальных вариантов среди доступных альтернатив.

Существует три типа системной архитектуры:

– одноуровневая;

– двухуровневая;

– многоуровневая.

В одноуровневой системной архитектуре все действия реализации графического интерфейса и решения задачи выполняются на одном уровне системы.

В двухуровневой архитектуре компоненты, отвечающие за графический интерфейс и прикладную логику, размещаются на клиентской машине и обращаются к общему источнику данных по сети.

В многоуровневой архитектуре презентационный сервис (первый уровень), прикладная логика (второй уровень) и сервис данных (третий уровень) отделены друг от друга.

Для данной работы было решено выбрать трёхуровневую архитектуру проекта (рисунок 2.1).

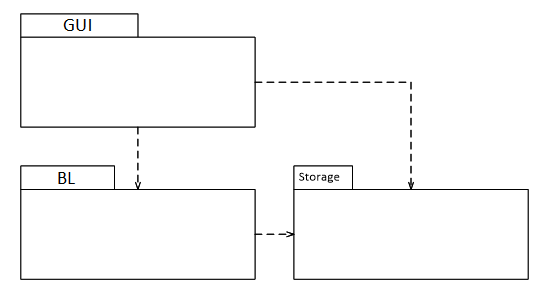


Рисунок 2.1 – Архитектура данного проекта

На рисунке 2.1 представлена архитектура системы, которая включает GUI – графический интерфейс пользователя; BL-бизнес логика; Storage –хранилище на котором хранятся данные.

Это решение обосновано тем, что данная архитектура наиболее подходит для решения поставленной задачи, поскольку мы имеем графический интерфейс, с которым пользователь взаимодействует напрямую; логику проекта, которая будет формировать сам документ «Приложение»; и, конечно же, хранилище данных, в роле которого выступает файловая система ПК на котором будет запущенна программа.

## 2.2 Диаграмма вариантов использования:

**UML –** унифицированный язык моделирования – это 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 [2].

Главные цели в разработке UML:

– предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий им разрабатывать осмысленные модели и обмениваться ими;

– предусмотреть механизмы расширяемости и специализации для расширения базовых концепций;

– обеспечить независимость от конкретных языков программирования и процессов разработки;

– обеспечить формальную основу для понимания этого языка моделирования (язык должен быть одновременно точным и доступным для понимания, без лишнего формализма);

– стимулировать рост рынка объектно-ориентированных инструментальных средств.

Диаграммы вариантов использования (Use Case) показывают взаимодействия между вариантами использования и действующими лицами, отражая функциональные требования к системе с точки зрения пользователя.

Цель построения – документирование функциональных требований в общем виде (требование – простота). Вариант использования - последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой и отражает представление о поведении системы с точки зрения пользователя [3].

В процессе разработки были выделены следующие роли:

1) «User» – Пользователь (GUI);

2) «FileSysem» – Файловая система (Storage);

3) «ExcelComponent» – Excel компонент (BL).

«User» – актор который представляет из себя пользователя программы. «FileSystem» – актор – файловая система ПК. «ExcelComponent» – актор – компонент Excel, который используется в программе.

Таблица 2.1 – Прецедент «Загрузить файл источник»

|  |  |
| --- | --- |
| Название прецедента | «Загрузить файл источник» |
| Описание | Пользователь указывает путь к расположению «Учебного поручения» по которому он хочет создать «Приложение». |

Таблица 2.2 – «Выбрать папку для сохранения»

|  |  |
| --- | --- |
| Название прецедента | «Выбрать папку для сохранения» |
| Описание | Пользователь указывает путь к расположению куда стоит сохранить документ «Приложение». |

Таблица 2.3 – «Указать учебный год»

|  |  |
| --- | --- |
| Название прецедента | «Указать учебный год» |
| Описание | Пользователь указывает год, для которого будет сформирован документ «Приложение». |

Таблица 2.4 – «Сформировать файл»

|  |  |
| --- | --- |
| Название прецедента | «Сформировать файл» |
| Описание | Этот прецедент формирует непосредственно «Приложение».  Предусловие: пользователь выбрал путь к «Учебному поручению», указал директорию для сохранения результата и выбрал учебный год. |

Таблица 2.5 – «Проверить наличие исходного файла»

|  |  |
| --- | --- |
| Название прецедента | «Проверить наличие исходного файла» |
| Описание | Файловая система проверяет правильность выбора файла-источника. |

Таблица 2.6 – «Создать новый файл по шаблону»

|  |  |
| --- | --- |
| Название прецедента | «Создать новый файл по шаблону» |
| Описание | В файловой системе будет создан пустой файл «Приложение», по шаблону, который находиться в ресурсах проекта. |

Таблица 2.7 – «Сформировать содержимое файла-результата »

|  |  |
| --- | --- |
| Название прецедента | «Сформировать содержимое файла-результата» |
| Описание | Программа обращается к Excel компоненту и с помощью его формирует, записывает нужные данные в пустое «Приложение». |

Таблица 2.8 – «Показать »

|  |  |
| --- | --- |
| Название прецедента | «Показать» |
| Описание | Прецедент выводит на экран готовый документ |

Таблица 2.9 – «Сохранить »

|  |  |
| --- | --- |
| Название прецедента | «Сохранить» |
| Описание | Прецедент сохраняет «Приложение» в указанную ранее директорию. |

Таблица 2.10 – Описание ролей

|  |  |
| --- | --- |
| Название роли | Описание |
| «User» | Пользователь (GUI). |
| «FileSysem» | Файловая система (Storage). |
| «ExcelComponent» | Excel компонент (BL). |

Описанные выше роли и варианты использования представлены на рисунке 2.2.



Рисунок 2.2 – Диаграмма вариантов использования

### 2.2.1 Подробное описание вариантов использования

Исходя из анализа построенной диаграммы использования, а также функций, которые должна выполнять программа, были разработаны варианты использования, основные из которых, представленные в таблице 2.11.

Таблица 2.11 – Варианты использования

|  |  |
| --- | --- |
| Номер | ВИ – 01 |
| Название | Указать файл источник |
| Участники | Пользователь |
| Описание | Пользователь указывает путь к расположению «Учебного поручения» по которому он хочет создать «Приложение». |
| Предварительные условия | Программа запущена |
| Выходные условия | Путь к файлу-источнику загружен в программу. |
| Порядок действий | Пользователь вводит данные о расположении исходного файла в файловой системе с помощью диалогового окна, которое становится активным после нажатия на кнопку «Выбрать путь к файлу-источнику». |
| Альтернативный порядок действий | Нет. |
| Приоритет | Высокий. |
|  |  |
| Номер | ВИ – 02 |
| Название | Выбрать папку для сохранения |
| Участники | Пользователь |
| Описание | Пользователь указывает путь к расположению, куда стоит сохранить документ «Приложение». |
| Предварительные условия | Программа запущена |
| Выходные условия | Программа сохранила путь сохранения документа «Приложение» и в ходе её выполнения там будет создан нужный нам документ. |
| Порядок действий | Пользователь выбирает директорию для сохранения с помощью диалогового окна, которое становится активным после нажатия на кнопку «Выбрать путь для сохранения». |
| Альтернативный порядок действий | Нет |
| Приоритет | Высокий. |
|  |  |
| Номер | ВИ – 03 |
| Название | Указать учебный год |
| Участники | Пользователь |
| Описание | Пользователь указывает год, для которого будет сформирован документ «Приложение». |
| Предварительные условия | Программа запущена |
| Выходные условия | Программа сохранила нужный учебный год в памяти. |
| Порядок действий | Пользователь нажимает кнопку «Указать учебный год» |
| Альтернативный порядок действий | Нет |
| Приоритет | Высокий. |
|  |  |
| Номер | ВИ – 04 |
| Название | Сформировать файл |
| Участники | Пользователь |
| Описание | Этот прецедент формирует непосредственно «Приложение». |
| Предварительные условия | Пользователь выбрал путь к «Учебному поручению», указал директорию для сохранения результата и выбрал учебный год. |
| Выходные условия | Программа подготовила все необходимые данные для формирования «Приложения». |
| Порядок действий | Выполнить ВИ – 01, ВИ – 02, ВИ – 03, нажать кнопку «Сформировать файл» |
| Альтернативный порядок действий | Нет |
| Приоритет | Высокий. |
|  |  |
| Номер | ВИ – 05 |
| Название | Сформировать содержимое документа-результата |
| Участники | ExcelComponent |
| Описание | Программа обращается к Excel компоненту и с помощью его формирует, записывает нужные данные в пустое «Приложение». |
| Предварительные условия | Выполнен ВИ – 04, и создан документ по шаблону, который находиться в ресурсах проекта. |
| Выходные условия | Вся необходимая информация теперь записана в «Приложение» |
| Порядок действий | Указать необходимые данные, затем нажать на кнопку «Сформировать документ». |
| Альтернативный порядок действий | Нет. |
| Приоритет | Высокий. |
|  |  |
| Номер | ВИ – 06 |
| Название | Сохранить |
| Участники | Файловая система |
| Описание | Прецедент сохраняет «Приложение» в указанную ранее директорию. |
| Предварительные условия | Содержание результата сформировано. |
| Выходные условия | Документ «Приложение» записан в соответствующую директорию. |
| Порядок действий | Сохранение файлов. |
| Альтернативный порядок действий | Нет. |
| Приоритет | Высокий. |

## 2.3 Диаграмма последовательности

Диаграмма последовательности – это диаграмма, которая служит для представления взаимодействия элементов модели в форме последовательности сообщений и соответствующих событий на линиях жизни объектов в динамике. Только время на данной диаграмме имеет шкалу порядка, а не шкалу отношений и это важно знать [4].

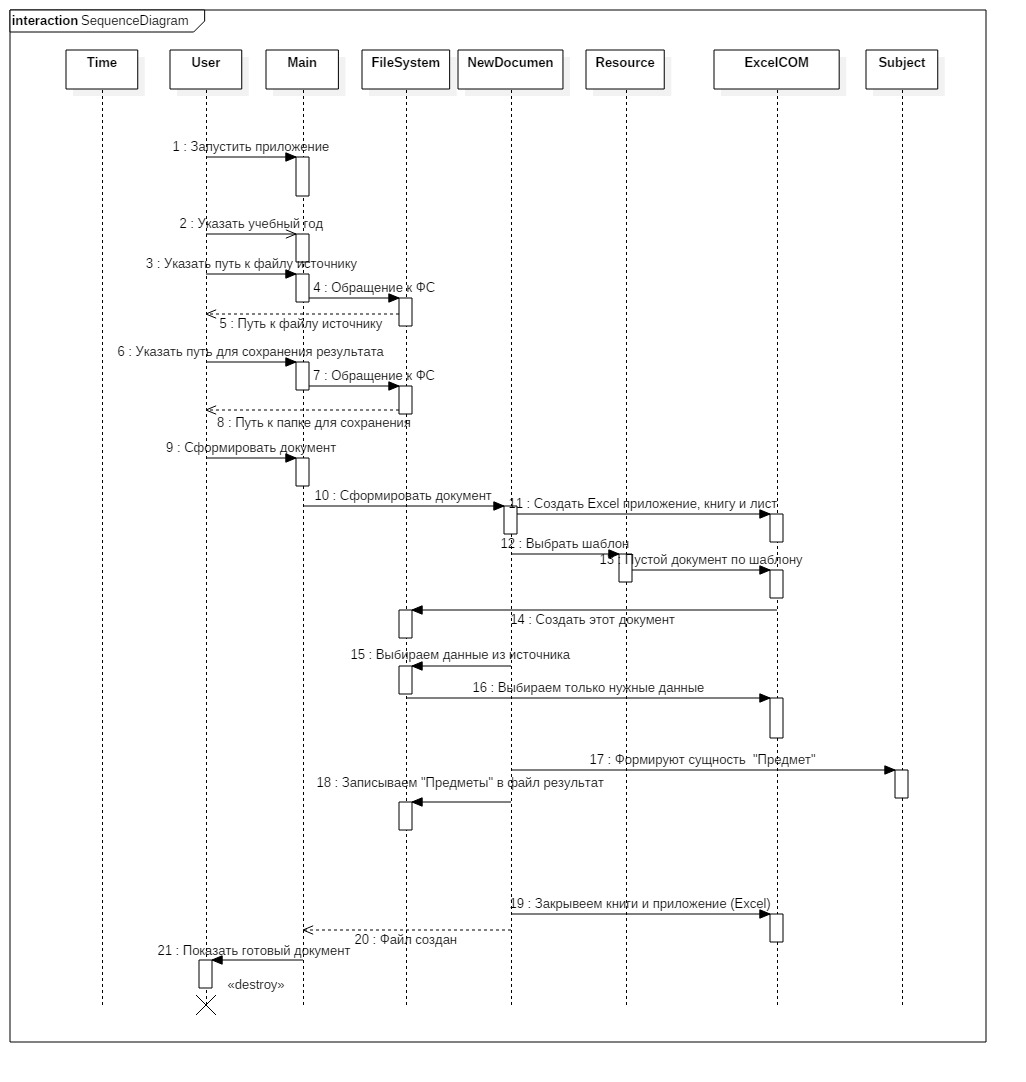


Рисунок 2.3 – Диаграмма последовательности полного бизнес-процесса создания документа «Приложение»

Данная диаграмма показывает порядок взаимодействия элементов модели и соответствующих событий в конкретно в данном проекте. Главная линия жизни является «User».

## 2.4 Проектирование графического интерфейса

На основании поставленных задач было решено выбрать графический интерфейс, который изображен на рисунке 2.4.

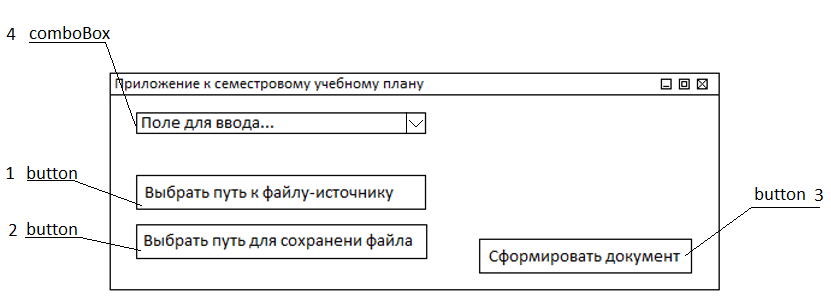


Рисунок 2.4 – Предположительный графический интерфейс программы

Таблица 2.12 – Описание элементов управления формы

|  |  |
| --- | --- |
| Элемент управления | Описание |
| 1 | Кнопка, отвечающая за выбор пользователем пути к файлу-источнику. |
| 2 | Кнопка, отвечающая за выбор пользователем директории для сохранения «Приложения». |
| 3 | Кнопка, отвечающая непосредственно за выбор вызов функции формирования документа. |
| 4 | Поле для выбора/ввода учебного года, для которого будет сформировано  «Приложение». Было решено выбрать в качестве этого элемента управления comboBox, т.к. это удобно и занимает меньше времени пользователя. |

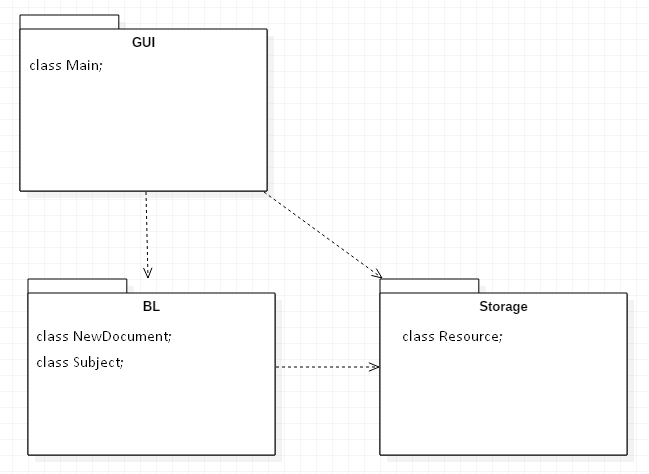
При попытке закрытия формы во время формирования документа пользователю будет выдано окно подтверждения закрытия формы. В роле этого диалоговое окна будет выступать стандартныйMessageBox с двумя кнопками «Да» и «Нет» и в зависимости от выбора пользователядальше развиваться последующие события.

В дальнейшем программа может улучшаться, и реализации новых функций, возможно, понадобятся новые элементы управления, но на данный момент этого достаточно, для выполнения базового функционала.

# 3 РАЗРАБОТКА

## 3.1 Реализация выбранной архитектуры в виде диаграммы пакетов

На основании выбранной архитектуры можно разработать диаграмму пакетов, которая представлена ниже на рисунке 2.5.

Рисунок 2.5 – Диаграмма пакетов выбранной проекта

GUI обращается к Storage посредством методов классов OpenFileDialog и FolderBrowseDialog, для открытия и сохранения файлов соответственно. GUI взаимодействует с BL посредством метода formateDocument() класса NewDocument, для формирования «Приложения». BL взаимодействует с Storage с помощью метода getUri() класса Resource, для загрузки шаблона для «Приложения» из ресурсов проекта.

## 3.2 Разработка диаграммы классов

Исходя из задания проекта, сложно выделить определённую сущность, т.к. в процессе формирования «Приложения» только определённые данные из «Учебного поручения», которые в свою очередь также по-разному представлены в источнике. Но всё-таки была выделена одна сущность Subject, которая отвечает за формирование информации о предмете, после того как в результат уже была сделана запись с именем этого предмета (подробнее об этом в схеме алгоритма). После формирования всех Subject-ов они в цикле записываются в «Приложение» и таким образом формируется основная часть документа.

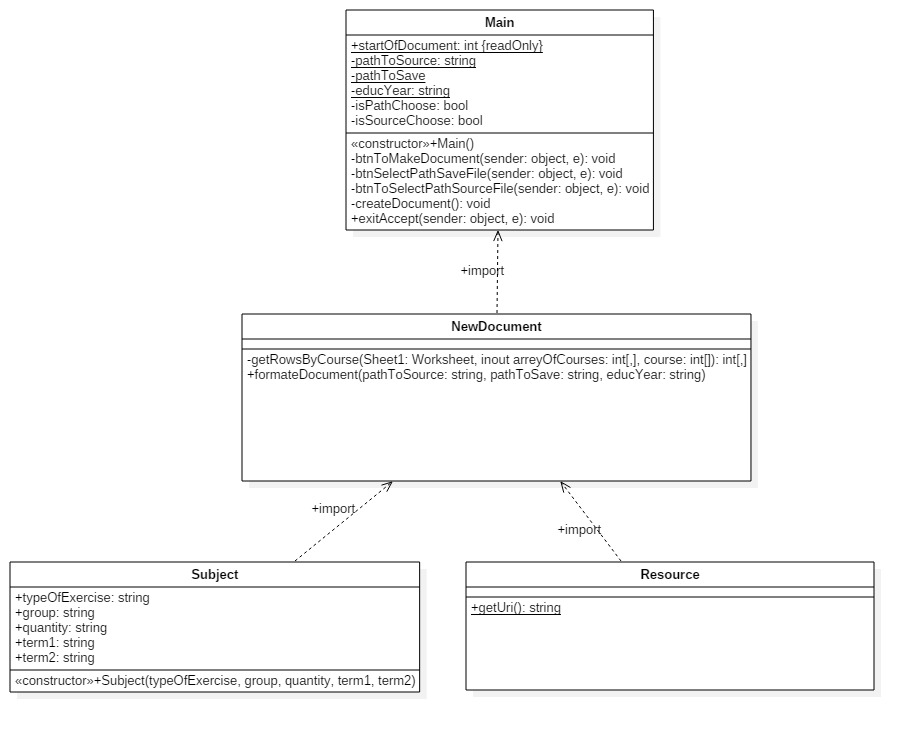


Рисунок 3.1 – Диаграмма классов проекта

На рисунке 3.1 приведена общая диаграмма классов, без проекции на архитектуру проекта, в следующем разделе будет приведена она же, разбита на пакеты, которая даст полное представление о структуре проекта.

## 3.3 Общая диаграмма классов с отображением на архитектуру

На приведенной ниже гибридной диаграмме пакетов и классов показаны основные связи. Связи представлены на различных уровнях, как между пакетами, так и между классами. Связи между пакетами показаны с помощью методов, которые ссылаются на классы других пакетов, тем самым осуществляя взаимосвязь между отдельными частями архитектуры.

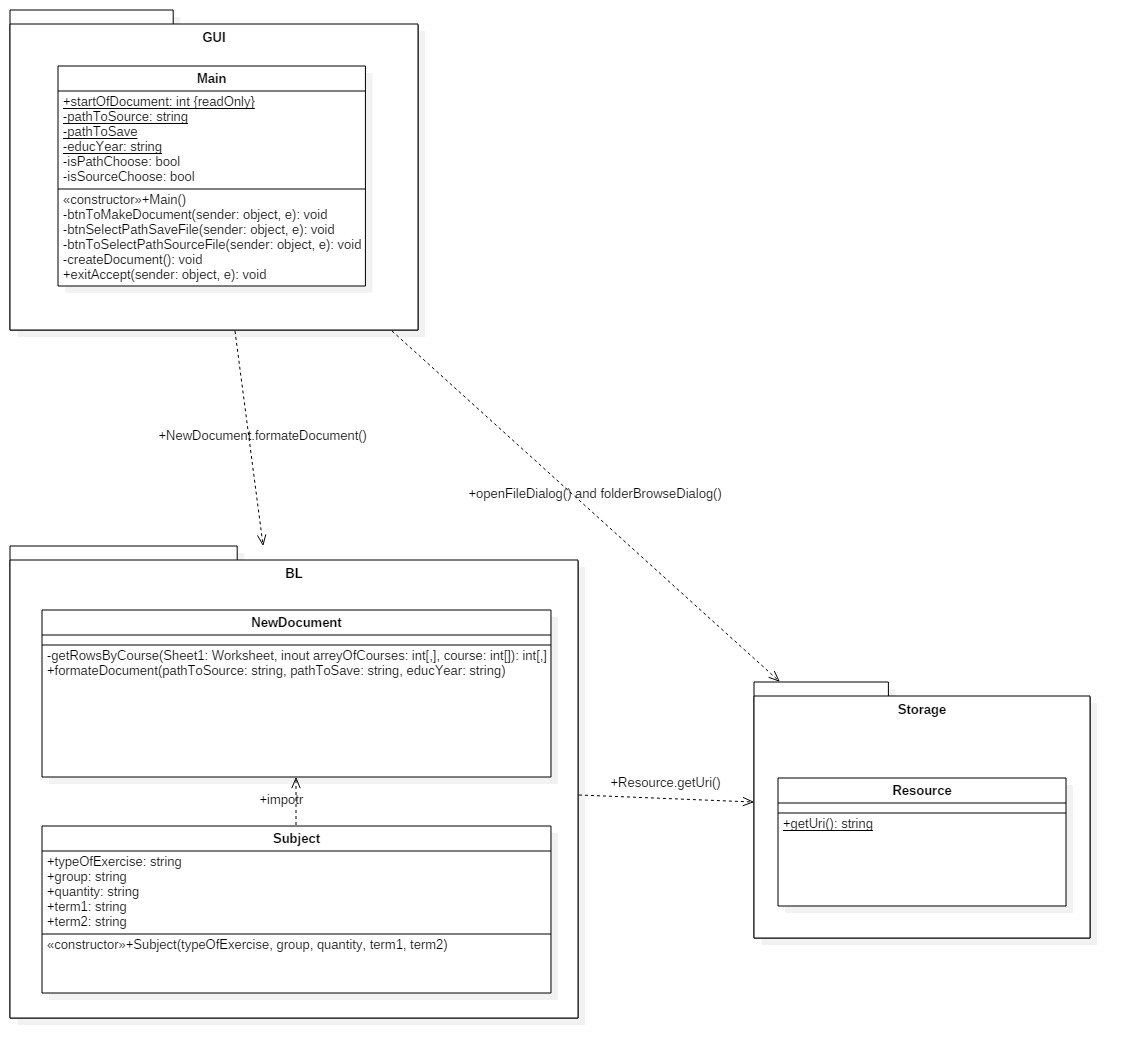


Рисунок 3.2 – Диаграмма классов проекта с проекцией на архитектуру

## 3.4 Выбор алгоритма и оптимизация

В ходе разработки программы было использовано несколько алгоритмов, которые с каждым разом улучшались. Улучшалось как время выполнения программы, так и сама логика алгоритма. Несмотря на то, что в техническом задании не предъявлены требования к временным характеристикам, программа частично строилась исходя из соображений оптимального времени выполнения.

Первый разработанный алгоритм имел время выполнения около 12 минут. Было решено ускорить формирование документа путем изменения алгоритма обхода строк в документе-источнике. Все предметы однократно разбивались на курсы, на котором они присутствуют, и заносились в двухмерный массив, в котором строками выступают номер курса, а столбцы – номер предмета по порядку на данном курсе. Это позволило ускорить алгоритм до 5-ти минут. Последним решением, которое было предпринято в ходе улучшения алгоритма было занесение предметов в формирующемся документе в массив. Это позволило ограничить количество обращений непосредственно к документу, и тем самым ещё улучшить временные характеристики программы. В итоге скорость формирования документа была сокращена с 11 – 12 минут до 40с – 1мин.

На рисунке 3.3 представлена конечная схема алгоритма проекта.



Рисунок 3.3 – Обобщенный алгоритм работы программы



Рисунок 3.4 –Алгоритм работы функции getRowsByCourse()



B

A

Рисунок 3.5 –Алгоритм работы функции newDocument() начало



С

С

B

A

Рисунок 3.6 –Алгоритм работы функции newDocument() конец

## 3.5 Описание всех методов

Для реализации задуманной архитектуры проекта были использованы методы и функции, которые описаны в таблице 3.1.

Таблица 3.1 – Описание методов

|  |  |
| --- | --- |
| Метод | Описание |
| formateDocument(string pathToSource,string pathToSave,string educYear) | Метод, в которым непосредственно осуществляется формирование документа.  − класс – NewDocument;  − уровень доступа – public;  − возвращаемое значение – void;  −pathToSource – путь к источнику;  – pathToSave – путь к папке сохранения;  – educYear – учебный год. |
| getRowsByCourse(Excel.Worksheet Sheet1, int[,] arreyOfCourses, int[] course) | Метод класса, который формирует двумерный массив куда однократно заносятся номера предметов текущего курса, для ускорения обращения к ним в дальнейшем.  − класс – NewDocument;  − уровень доступа – private;  − возвращаемое значение – int[,];  – Sheet1 – лист Excel «Учебное поручение»;  – arreyOfCourses – массив с номерами строк уникальных предметов;  – course – номер курса. |
| getUri() | Загружает из ресурсов шаблон пустого документа «Приложение» и делает его удобным для использования.  − класс – Resource;  − уровень доступа – public;  − возвращаемое значение – string – шаблон; |
| btnSelectPathSaveFile(object sender, EventArgs e) | Обработчик события нажатия на кнопку, который позволяет указать директорию в которую будет сохранен результат.  − класс – Main;  − уровень доступа – private;  − возвращаемое значение – void; |
| btnToSelectPathSourceFile(object sender, EventArgs e) | Обработчик события нажатия на кнопку, который позволяет указать путь к источнику («Учебное поручение»).  − класс – Main;  − уровень доступа – private;  − возвращаемое значение – void; |
| aboutShow(object sender, EventArgs e) | Обработчик события нажатия на кнопку, который показывает окно «О программе».  − класс – Main;  − уровень доступа – private;  − возвращаемое значение – void; |
| exitAccept(object sender, EventArgs e) | Обработчик события закрытия формы, в котором пользователь должен подтвердить свои намерения закрыть программу.  − класс – Main;  − уровень доступа – private;  − возвращаемое значение – void; |
| btnToMakeDocument(object sender, EventArgs e) | Обработчик события нажатия на кнопку, после которого выполняется проверка введённых данных, и выводится соответствующее сообщение («Документ формируется» или «Введите корректные данные»)  − класс – Main;  − уровень доступа – private;  − возвращаемое значение – void; |
| createDocument() | Метод-поток, который создан для организации многопоточности, который вызывает метод formateDocument(), который формирует «Приложение».  − класс – Main;  − уровень доступа – private;  − возвращаемое значение – void; |

Таблица 3.2 – Описание переменных

|  |  |
| --- | --- |
| Переменная | Описание |
| public const int startOfDocument | Реальное начало таблицы в шаблоне  – класс – Main; |
| private static string pathToSource | Путь к файлу-источнику  – класс – Main; |
| private static stringpathToSave | Путь к директории для сохранения файла результата  – класс – Main; |
| private static stringeducYear | Учебный год, для которой будет создаваться документ  – класс – Main; |
| private bool isPathChoose | Указывает на то выбран ли путь к файлу источнику  – класс – Main; |
| private bool isSourceChoose | Указывает на то выбран ли путь к директории для сохранения  – класс – Main; |
| private bool isCreating | Указывает формируеться ли сейчас документ (для опроса закрытия приложения)  – класс – Main; |
| public string typeOfExercise | Вид занятия лаб/практ  – класс – Subject; |
| public string group | Номер группы  – класс – Subject; |
| public string quantity | Количество человек в группе  – класс – Subject; |
| public string term1 | Количество часов в 1й чверти  – класс – Subject; |
| public string term2 | Количество часов во 2й чверти  – класс – Subject; |

# 4 ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ

## 4.1 Верификация

В техническом задании (ПРИЛОЖЕНИЕ А), были определены следующие задачи проекта:

1а) создание нового (пустого) файла «Приложение к семестровому учебному плану» по шаблону из ресурсов проекта;

2а) открытие (загрузка) существующего файла «Учебное поручение» для выборки данных;

3а) автоматическое редактирование открытого (далее – текущего) файла;

4а) сохранение файла с предопределённым именем;

5а) отображение названия программы, версии программы и комментариев разработчика.

В результате разработки программы все вышеперечисленные задачи были решены и написаны функции для их реализации.

В таблице 4.1 приведены методы, которые подтверждают выполнение вышеперечисленных функций.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование в ТЗ  (Задача или функция) | Номер функции | Проектирование  (Прецедент) | Подтверждение выполнения. Разработка  (Класс.Метод()) | Вывод |
| 1. Создание нового (пустого) файла по шаблону | 1 |  |  | Pass |
| a) создание файла по шаблону | 1а) | Создать новый файл по шаблону | ObjExcel.Workbooks.add(); | Pass |
| 2. Открытие существующего файла для выборки данных | 2 |  |  | Pass |
| a) открытие файла | 2a) | Указать файл источник | Main.openFileDialog() | Pass |
| 3. Автоматическое редактирование открытого | 3 |  |  | Pass |
| a) автоматическое редактирование открытого | 3а) | Сформировать содержимое документа-результата | newDocument. formateDocument(); | Pass |
| 4. Сохранение файла с предопределённым именем | 4 |  |  | Pass |
| a) сохранение файла с предопределённым именем | 4а) | Сохранить | newDocument.Obj WorkBook.SaveAs(); | Pass |
| 5. Отображение справки | 5 |  |  | Pass |
| a) отображение справки | 5а) | – | Main.aboutShow(); | Pass |

Таблица 4.1 – Верификация функций

В проекте помимо остальных решений используется многопоточность. Это позволяет корректно управлять формой в процессе формирования документа. Иначе при вызове метода formateDocument(), запускается приложение Excel, а доступ к форме блокируется, это выглядит, как зависание программы и может быть воспринято как ошибка, поэтому и была использована многопоточность.

Было решено формировать название готового документа с включением в него выбранного учебного года. Это позволит не путаться в документах и облегчит поиск нужного преподавателю «Приложения».

Также добавлена функция подтверждения выхода из программы в процессе формирования документа. Иначе если пользователь закроет программу, которая не до конца сформировала документ, он в итоге будет сформирован до конца, но в приложение Excel не будет закрыто автоматически и это придется сделать пользователю самостоятельно.

В результате верификации проекта было установлено, что все задачи решаются теми или иными методами, подтверждение приведено в разделе 4.2 тестирование.

## 4.2 Тестирование

При разработке данного проекта были реализованы такие функции:

1) создание нового (пустого) файла «Приложение к семестровому учебному плану» по шаблону из ресурсов проекта;

2) открытие (загрузка) существующего файла «Учебное поручение» для выборки данных;

3) автоматическое редактирование открытого (далее – текущего) файла;

4) сохранение файла с предопределённым именем в соответствии с выбранным пользователем учебным годом;

5) отображение названия программы, версии программы и комментариев разработчика.

Таблица 4.2 – Тестирование программы и её результаты

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемая функция | Номер теста | Цель теста | Результат теста |
| 1) Проверка указанных данных | 1 | Проверка на некорректный ввод пути к файлу-источнику | Pass |
| 2 | Проверка на некорректный ввод пути к папке сохранения | Pass |
| 2) Корректный ответ на закрытие программы во время формирования документа | 3 | Закрытие программы до окончания формирования документа | Pass |
| 3) Сохранение документа с правильным именем | 4 | Проверка на корректное сохранение документа с правильном именем | Pass |
| 4) Формирование документа | 5 | Проверить созданный документ на соответствие ожидаемому результату | Pass |
| 5) Вывод справки | 6 | Пользователь нажимает на «Help–>About…» | Pass |

В результате тестирования программы была обнаружена ошибка, в столбце «Кількість студентів у групі» было значение, отвечающее за две группы целиком. Эта ошибка была вызвана несоответствием требованию входной информации, пользователь мог увидеть что-то подобное (рисунок 4.1). Она была исправлена при последующем тестировании.

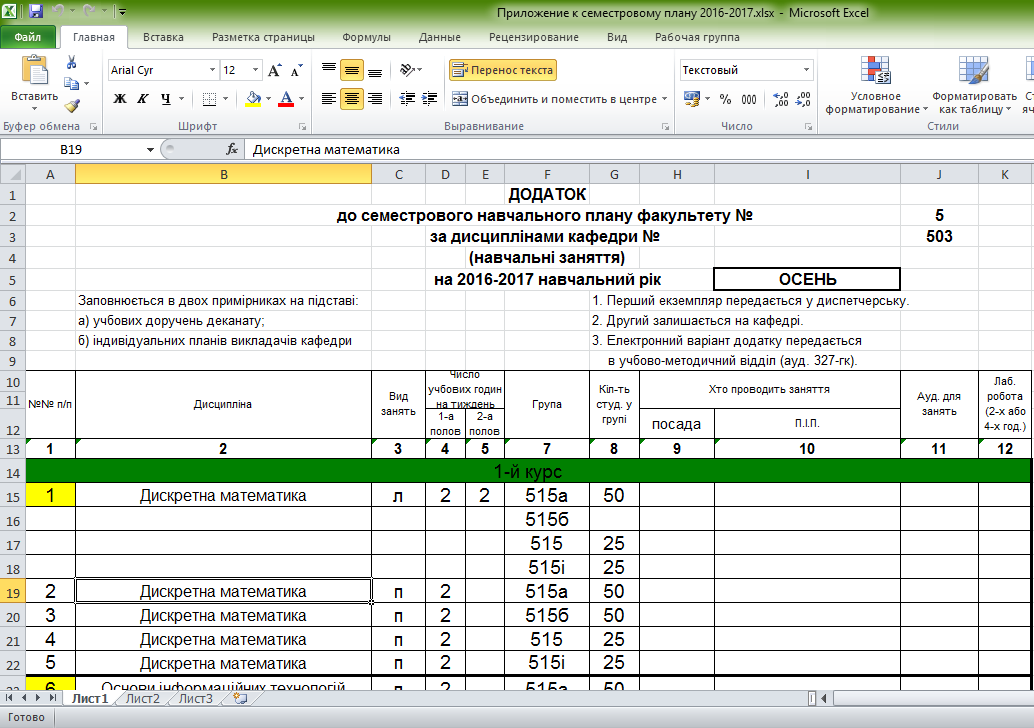


Рисунок 4.1 – Ошибка, которая возникала при обработке информации

Как видим из таблицы 4.2 тестирование и вышеприведенных скриншотов тестирование программы выполнено успешно.

# ВЫВОДЫ

В результате выполнения данной работы был реализован проект, позволяющий в полуавтоматическом режиме формировать документ «Приложение к семестровому учебному плану» на основе документа «Учебное поручение».

Результат выполнения данной работы:

1) проведен анализ предметной области и существующих решений, в результате чего было сформировано постановку задачи;

2) на основе постановки задачи сформировано техническое задание;

3) проведено проектирование программы:

– выбрана архитектура проекта;

– построено диаграмма вариантов использования;

– построено диаграмма последовательности;

­– разработан алгоритм работы программы.

4) в ходе разработки программы были выполнены такие шаги:

– реализована архитектура проекта;

– разработана диаграмма пакетов;

– разработана диаграмма классов;

­– был оптимизирован изначальный алгоритм работы программы.

5) проведена успешная верификация программы;

6) проведено успешное тестирование программы;

7) написано руководство оператора;

8) сделана презентация.

Разработанная программа верифицирована и протестирована.

Путей развития данного проекта существует большое количество, вот некоторые из них:

1) добавить возможность, использовать в качестве источника не только Excel документы;

2) реализовать данный проект в виде веб-сервиса;

3) использовать более быстрый алгоритм формирования документа;

4) реализовать возможность выбора кафедры, для которой формировать «Приложение»;

5) решить проблему зависания процесса Excel в системе, при закрытии программы во время формирования документа.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Совершенный код. Мастер-класс / Пер. с англ. — М. : Издательство «Русская редакция», 2010. — 41с.

2 Грейди Буч. Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. — СПб.: Питер, 2004. — 432 с.

3 Язык UML. Диаграммы вариантов использования [Электронный ресурс]// URL: [http://www.its.kpi.ua/itm/tkot/Students/Lec2\_1\_UML[p].pdf](http://www.its.kpi.ua/itm/tkot/Students/Lec2_1_UML%5bp%5d.pdf). (Дата обращения: 01.12.2016).

4 Язык UML в анализе и проектировании систем и бизнес-процессов [Электронный ресурс] // URL: http://www.slideshare.net/DEVTYPE/3-48780912.(Дата обращения: 01.12.2016).

5 Microsoft.Office.Interop.Excel namespace [Электронный ресурс] // URL: <https://msdn.microsoft.com/en-us/library/office/microsoft.office.interop.excel.aspx>(Дата обращения: 18.11.2016).

# ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1 Введение

1.1 Наименование программы

Наименование ­– "Программа для генерации приложения к семестровому учебному плану" (далее по тексту - Программа).

1.2 Краткая характеристика области применения

Программа предназначена для применения в учебных заведениях для облегчения ведения документации при подготовке к проведению учебного процесса.

2 Основания для разработки

2.1 Основание для проведения разработки

Основание для разработки - задание на курсовой проект по дисциплине "Объектно-ориентированное программирование", в целях закрепления, углубления и обобщения знаний, полученных за время обучения, и их применения к комплексному решению конкретной.

2.2 Наименование и условное обозначение темы разработки

Наименование темы разработки – "Разработка программного обеспечения для генерации приложения к семестровому учебному плану".

Условное обозначение темы разработки (шифр темы) – "РПОГ-01".

3 Назначение разработки

3.1 Функциональное назначение

Функциональным назначением программы является предоставление преподавателю возможности формировать определённую документацию не в ручном, а в полуавтоматическом режиме.

3.2 Эксплуатационное назначение

Программа должна эксплуатироваться в учебных заведениях.

Конечные пользователи программы – сотрудники института, которые занимаются подготовкой, организацией и проведением учебного процесса.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам

4.1.1 Требования к составу выполняемых функций

Программа должна обеспечивать возможность решения задач:

1) создание нового (пустого) файла «Приложение к семестровому учебному плану» по шаблону из ресурсов проекта;

2) открытие (загрузка) существующего файла «Учебное поручение» для выборки данных;

3) автоматическое редактирование открытого (далее - текущего) файла;

4) сохранение файла с предопределённым именем;

5) отображение названия программы, версии программы и комментариев разработчика.

4.1.2 Требования к организации входных данных

Входные данные программы должны быть организованы в виде отдельного файла определенного формата.

Файлы указанного формата должны размещаться (храниться) на локальных или съемных носителях, отформатированных согласно требованиям операционной системы.

4.1.3 Требования к организации выходных данных

См. Требования к организации входных данных.

4.1.4 Требования к временным характеристикам

Требования к временным характеристикам программы не предъявляются.

4.2 Требования к составу и параметрам технических средств

Для исполнения программы необходим универсальный компьютер любого типа, для которого получен двоичный код, представленный в виде исполнимого файла (.exe-файла), способного выполняться установленным в ПК процессором.

Для запуска программы на компьютерах соответствующего типа должен быть получен исполнимый файл, способный выполняться на компьютерах именно этого типа. Для этого необходимо откомпилировать исходные файлы на компьютерах того типа, на которых предполагается исполнять результирующую программу.

В состав технических средств должен входить х64 совместимый компьютер, включающий в себя:

1) процессор с тактовой частотой 2.7 Ghz;

2) оперативную память объемом 4 GB, не менее;

3) свободного дискового пространства 100 MB

4) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

5) клавиатуру и мышь;

6) жесткий диск.

4.3 Требования к информационной и программной  
совместимости

Поддержка .xlsx файлов (версия MS Excel2010).

4.3.1 Требования к информационным структурам и методам решения

Входные данные должны быть представлены в виде .xls/.xlsx файлов. Информационная структура файла должна включать в себя таблицу, содержащую разметку, согласно Приложению 1.

4.3.2 Требования к исходным кодам и языкам  
программирования

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда VisualStudio .NET.

4.3.3 Требования к программным средствам, используемым программой

На ПК должны быть установлены:

1) Операционная система Windows 10 x64;

2) .Net Framework 4.5;

3) Windows Forms 2.0;

4) Microsoft Excel 2010.

4.3.4 Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

5 Требования к программной документации

5.1 Предварительный состав программной документации

В результате разработки программы должна быть представлена следующая программная документация:

а) техническое задание;

б) схемы алгоритмов;

в) тексты программ;

г) план тестирования и верификации;

д) руководство оператора;

е) пояснительная записка;

6 Стадии и этапы разработки

6.1 Стадии разработки

Разработка должна быть проведена в две стадии:

1) разработка технического задания;

2) рабочее проектирование.

6.2 Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены следующие этапы работ:

1) разработка программы;

2) разработка программной документации;

3) испытания программы.

6.3 Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены такие работы:

1) постановка задачи;

2) определение и уточнение требований к техническим средствам;

3) определение требований к программе;

4) определение стадий, этапов и сроков разработки программы и документации на неё;

5) выбор языков программирования;

6) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе испытаний программы должны быть выполнены следующие виды работ:

1) разработка, согласование и утверждение программы и методики испытаний;

2) проведение приемосдаточных испытаний;

3) корректировка программы и программной документации по результатам испытаний.

## Приложение А. Приложение к Техническому заданию 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Назвадисципліни | Факультет | Кафедра | Спеціальність | Курс | Семестр | Групи | Кількістьгруп | Кількістьстуд | Кількість годин на 1 половину семестру | | | | | | Кількість годин на 2 половину семестру | | | | | |
|
| Лекції | | Лаб. роботи | | Практ. заняття | | Лекції | | Лаб. роботи | | Практ. заняття | |
|
| 1 | Комп'ютернасхемотехніка | 3 | 305 | 6.050202 | 2 | 4 | 329 | 1 | 25 | 2 | 20 | 1 | 10 |  |  | 2 | 18 | 1 | 9 |  |  |
| 2 | Мікропроцесорнатехніка | 3 | 305 | 6.050202 | 3 | 5 | 339 | 1 | 19 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | 7 | 1 | 7 | 1 | 7 |
| 3 | Надійністьапаратно-програмнихкомплексів | 3 | 305 | 6.050202 | 4 | 7 | 349 | 1 | 19 | 2 | 16 | 2 | 16 |  |  | 2 | 14 | 2 | 14 |  |  |
| 4 | Програмнезабезпеченнякомп'ютерних мереж | 4 | 405 | 7.04030301 | 5 | 10 | 450м | 1 | 15 | 1 | 10 | 1 | 10 |  |  | 1 | 9 | 2 | 18 |  |  |
| 5 | Дискретна математика | 5 | 501 | 6.050901 | 1 | 1 | 516 | 1 | 25 | 2 | 24 |  |  | 2 | 24 |  |  |  |  |  |  |
| 6 | Обчислювальнатехніка та програмування | 5 | 501 | 6.050901 | 1 | 1 | 516 | 1 | 25 | 3 | 36 | 2 | 24 |  |  |  |  |  |  |  |  |
| 7 | Комп'ютернасхемотехніка | 5 | 501 | 6.050901 | 2 | 3 | 526 | 1 | 27 | 2 | 16 | 1 | 8 |  |  | 1 | 7 | 1 | 7 |  |  |
| 8 | Основироботи в Internet | 5 | 501 | 6.050901 | 3 | 6 | 536 | 1 | 15 | 1 | 10 | 1 | 10 |  |  | 1 | 9 | 1 | 9 |  |  |
| 9 | Комп'ютернасхемотехніка | 5 | 501 | 6.050901 (3 роки) | 2 | 3 | 516ст | 1 | 15 | 2 | 16 | 1 | 8 |  |  | 1 | 7 | 1 | 7 |  |  |
| 10 | Основироботи в Internet | 5 | 501 | 6.050901 (3 роки) | 3 | 6 | 526ст | 1 | 4 | 1 | 10 | 1 | 10 |  |  | 1 | 9 | 1 | 9 |  |  |

## Приложение А. Приложение к Техническому заданию 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №№ п/п | Дисципліна | Вид занять | Число учбових годин на тиждень | | Група | Кіл-ть студ. у групі | Хто проводить заняття | | Ауд. для занять | Лаб. робота (2-х або 4-х год.) |
|
| 1-а полов | 2-а полов | посада | П.І.П. |
| **1** | **2** | **3** | **4** | **5** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1-й курс | | | | | | | | | | |
| 1 | Дискретна математика | л | 2 | 2 | 515а | 25 |  |  |  |  |
|  |  |  |  |  | 515б | 25 |  |  |  |  |
|  |  |  |  |  | 515 | 25 |  |  |  |  |
|  |  |  |  |  | 515і | 25 |  |  |  |  |
| 2 | Дискретна математика | п | 2 |  | 515а | 25 |  |  |  |  |
| 3 | Дискретна математика | п | 2 |  | 515б | 25 |  |  |  |  |
| 4 | Дискретна математика | п | 2 |  | 515 | 25 |  |  |  |  |
| 5 | Дискретна математика | п | 2 |  | 515і | 25 |  |  |  |  |
| 6 | Основиінформаційнихтехнологій | л | 2 |  | 515а | 25 |  |  |  |  |
|  |  |  |  |  | 515б | 25 |  |  |  |  |
|  |  |  |  |  | 515 | 25 |  |  |  |  |
|  |  |  |  |  | 515і | 25 |  |  |  |  |
| 7 | Основиінформаційнихтехнологій | лаб | 2 |  | 515а | 25 |  |  |  |  |
| 8 | Основиінформаційнихтехнологій | лаб | 2 |  | 515б | 25 |  |  |  |  |
| 9 | Основиінформаційнихтехнологій | лаб | 2 |  | 515 | 25 |  |  |  |  |
| 10 | Основиінформаційнихтехнологій | лаб | 2 |  | 515і | 25 |  |  |  |  |

# ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММЫ

Класс Main.cs

using System;

usingSystem.Windows.Forms;

usingSystem.Threading;

namespace WindowsFormsApplication7

{

publicpartialclassMain : Form

{

publicMain()

{

InitializeComponent();

comboBox1.Text = comboBox1.Items[13].ToString();

}

publicconstintstartOfDocument = 14; //Реальноеначалотаблицывшаблоне

privatestaticstringpathToSource, //Путькфайлу-источнику

pathToSave, //Путь к директории для сохранения файла результата

educYear; //Учебный год, для которой будет создаваться документ

privateboolisPathChoose = false, //Указывает на то выбран ли путь к файлу источнику

isSourceChoose = false, //Указывает на то выбран ли путь к директории для сохранения

isCreating=false; //Указывает формируеться ли сейчас документ (для опроса закрытия приложения)

///<summary>

/// Ф-ция, которая вызывает метод формирования документа (для многопоточности)

///</summary>

privatevoidcreateDocument()

{

NewDocumentresultDoc = newNewDocument();

isCreating = true;

resultDoc.formateDocument(@pathToSource, pathToSave, educYear);

isCreating = false;

Thread.Sleep(0);

}

///<summary>

/// Обработчик события нажатия на кнопку, который отвечает непосредственно за создание нужного документа

///</summary>

///<param name="sender"></param>

///<param name="e"></param>

privatevoidbtnToMakeDocument(object sender, EventArgs e)

{

if (isSourceChoose&&isPathChoose)

{

educYear = comboBox1.Text;

ThreadmyThread = newThread(createDocument); // Слздаёмновыйпотокдля

myThread.Start();

MessageBox.Show("Документ формируется, пожалуйста не закрывайте программу.\nСформированный документ будет выведен на экран.", "Внимание!");

Thread.Sleep(0);

Console.Read();

}

else

if (isSourceChoose)

MessageBox.Show("Выбирите директорию для сохранения файла", "Ошибка");

else

MessageBox.Show("Выбирите файл - источник", "Ошибка");

}

///<summary>

/// Обработчик события нажатия на кнопку, который отвечает за выбор директории для сохранения результата

///</summary>

///<param name="sender"></param>

///<param name="e"></param>

privatevoidbtnSelectPathSaveFile(object sender, EventArgs e)

{

var save = newFolderBrowserDialog();

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

{

pathToSave = save.SelectedPath;

isPathChoose = true;

}

}

///<summary>

/// Обработчик события нажатия на кнопку, который отвечает за выбор пути к файлу-источнику

///</summary>

///<param name="sender"></param>

///<param name="e"></param>

privatevoidbtnToSelectPathSourceFile(object sender, EventArgs e)

{

OpenFileDialog open = newOpenFileDialog();

open.Filter = "Excel Files|\*.xls;\*.xlsx;\*.xlsm";

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

{

pathToSource = open.FileName;

isSourceChoose = true;

}

}

///<summary>

///Выводитокно About

///</summary>

///<param name="sender"></param>

///<param name="e"></param>

privatevoidaboutShow(object sender, EventArgs e)

{

newAbout().ShowDialog();

}

///<summary>

///Подтверждениезакрытияформы

///</summary>

///<param name="sender"></param>

///<param name="e"></param>

privatevoidexitAccept(object sender, FormClosingEventArgs e)

{

if (isCreating)

{

var result = MessageBox.Show("Документещёнесформировался, выточнохотитевыйти?", "Внимание",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result != DialogResult.Yes)

e.Cancel = true;

}

}

}

}

КлассNewDocument.cs

usingMicrosoft.Office.Interop.Excel;

using System;

usingSystem.Drawing;

usingSystem.Collections.ObjectModel;

namespaceCourseProject

{

classNewDocument

{

///<summary>

/// Метод, который формирует двумерный массив куда однократно заносятся номера предметов текущего курса, для ускорения обращения к ним в дальнейшем.

///</summary>

///<param name="Sheet1">Лист Excel «Учебноепоручение»</param>

///<param name="arreyOfCourses">Массив с номерами строк уникальных предметов</param>

///<param name="course">Номеркурса</param>

///<returns>Двумерный массив номеров предметов текущего курса</returns>

privateint[,] getRowsByCourse(Microsoft.Office.Interop.Excel.Worksheet Sheet1, int[,] arreyOfCourses, int[] course)

{

for (inti = 1; i< 296; i++)

{

string temp = Sheet1.Cells[i, 6].Text;

if ("" + Sheet1.Cells[i, 4].Text == "503")

switch (temp)

{

case"1":

arreyOfCourses[0, course[0]] = i;

course[0]++;

break;

case"2":

arreyOfCourses[1, course[1]] = i;

course[1]++;

break;

case"3":

arreyOfCourses[2, course[2]] = i;

course[2]++;

break;

case"4":

arreyOfCourses[3, course[3]] = i;

course[3]++;

break;

case"5":

arreyOfCourses[4, course[4]] = i;

course[4]++;

break;

}

}

returnarreyOfCourses;

}

///<summary>

/// Метод, в которым непосредственно осуществляется формирование документа

///</summary>

///<param name="pathToSource">Путькисточнику</param>

///<param name="pathToSave">Путькпапкесохранения</param>

///<param name="educYear">Учебныйгод</param>

publicvoidformateDocument(stringpathToSource,stringpathToSave,stringeducYear)

{

ObservableCollection<Subject> subjects = newObservableCollection<Subject>();

int count = Main.startOfDocument,

id = 1;

int[] course = newint[5] { 0, 0, 0, 0, 0 };

int[,] arreyOfCourses = newint[5, 100];

int[] namesCount = newint[5] { 0, 0, 0, 0, 0 };

int[,] names = newint[5, 100];

string[] groups = newstring[0];

boolisExist = false;

stringcurrentName,

currentSpec;

Microsoft.Office.Interop.Excel.Workbook ObjWorkBook1,

ObjWorkBook2;

Microsoft.Office.Interop.Excel.Worksheet ObjWorkSheet1,

ObjWorkSheet2;

Microsoft.Office.Interop.Excel.ApplicationObjExcel = newMicrosoft.Office.Interop.Excel.Application();

ObjExcel.Visible = true;

ObjWorkBook1 = ObjExcel.Workbooks.Open(pathToSource);

ObjWorkBook2 = ObjExcel.Workbooks.Add(Template: Resourсe.getUri());

ObjWorkSheet1 = (Microsoft.Office.Interop.Excel.Worksheet)ObjWorkBook1.Sheets[1];

ObjWorkSheet2 = (Microsoft.Office.Interop.Excel.Worksheet)ObjWorkBook2.Sheets[1];

ObjWorkSheet2.Cells[5, 6].Value = "на " + educYear + " навчальнийрік";

Microsoft.Office.Interop.Excel.Range excelcells3 = ObjWorkSheet2.get\_Range("A" + 14, "K" + 700);

excelcells3.Borders[XlBordersIndex.xlInsideHorizontal].Weight = Microsoft.Office.Interop.Excel.XlBorderWeight.xlThin;

excelcells3.Borders[XlBordersIndex.xlInsideVertical].Weight = Microsoft.Office.Interop.Excel.XlBorderWeight.xlThin;

arreyOfCourses = getRowsByCourse(ObjWorkSheet1, arreyOfCourses, course);

for (int j = 0; j <= 4; j++)//Циклпокурсам

{

Microsoft.Office.Interop.Excel.Rangeexcelcells = ObjWorkSheet2.get\_Range("A" + count, "K" + count);

excelcells.Interior.Color = Color.Green;

excelcells.Merge(Type.Missing);

ObjWorkSheet2.Cells[count, 1].Value = (j + 1) + "-йкурс";

count++;//текущее положение в НОВОМ доке++

for (inti = 0; i< course[j]; i++)

{

for (int p = 0; p <namesCount[j]; p++)//Проверяем наличие такой записи в новом документе

{

if (ObjWorkSheet1.Cells[arreyOfCourses[j, i], 2].Text == ObjWorkSheet2.Cells[names[j, p], 2].Value)

{

isExist = true;

break;

}

}

if (!isExist) //Если такого предмета в этом курсе ещё не было

{

ObjWorkSheet2.Cells[count, 1].Interior.Color = Color.Yellow;

names[j, namesCount[j]] = count;

namesCount[j]++;

ObjWorkSheet2.Cells[count, 2].Value = ObjWorkSheet1.Cells[arreyOfCourses[j, i], 2].Text;

currentName = ObjWorkSheet1.Cells[arreyOfCourses[j, i], 2].Text; //Сохраняемназваниеэтогопредмета

ObjWorkSheet2.Cells[count, 3].Value = "л";

currentSpec = "";

for (int k = i; k < course[j]; k++) //Пробегаемсяпоисточнику

{

if (ObjWorkSheet1.Cells[arreyOfCourses[j, k], 2].Text == currentName)//Еслитакойпредметнаэтомкурсеесть

{

if (ObjWorkSheet1.Cells[arreyOfCourses[j, k], 5].Text != currentSpec) //еслинетажеспециальностьиимя

{

currentSpec = ObjWorkSheet1.Cells[arreyOfCourses[j, k], 5].Text; //записавыемспециальность

Array.Resize<string>(ref groups, ObjWorkSheet1.Cells[arreyOfCourses[j, k], 8].Text.Split('\n').Length);

groups = ObjWorkSheet1.Cells[arreyOfCourses[j, k], 8].Text.Split('\n');

for (int w = 0; w <groups.Length; w++)

{

ObjWorkSheet2.Cells[count+w, 7].Value = "" + (int)ObjWorkSheet1.Cells[arreyOfCourses[j, k], 10].Value/groups.Length;

}

if (names[j, namesCount[j]-1] == count)

{

ObjWorkSheet2.Cells[count, 4].Value = ObjWorkSheet1.Cells[arreyOfCourses[j, k], 11].Value;

ObjWorkSheet2.Cells[count, 5].Value = ObjWorkSheet1.Cells[arreyOfCourses[j, k], 17].Value;

}

for (int l = 0; l <groups.Length; l++)

{

ObjWorkSheet2.Cells[count, 6].Value = groups[l];

count++;

if (ObjWorkSheet1.Cells[arreyOfCourses[j, k], 13].Text != "")

{

subjects.Add(newSubject("лаб", ""+groups[l], ""+int.Parse(ObjWorkSheet1.Cells[arreyOfCourses[j, k], 10].Text) / groups.Length, ObjWorkSheet1.Cells[arreyOfCourses[j, k], 13].Text, ObjWorkSheet1.Cells[arreyOfCourses[j, k], 19].Text));

}

else

{

subjects.Add(newSubject("п", "" + groups[l], ""+int.Parse(ObjWorkSheet1.Cells[arreyOfCourses[j, k], 10].Text) / groups.Length, ObjWorkSheet1.Cells[arreyOfCourses[j, k], 15].Text, ObjWorkSheet1.Cells[arreyOfCourses[j, k], 21].Text));

}

}

}

else

{

ObjWorkSheet2.Cells[names[j, namesCount[j]-1], 5].Value = ObjWorkSheet1.Cells[arreyOfCourses[j, k], 17].Value;

}

}

}

for (int q = 0; q <subjects.Count; q++)

{

ObjWorkSheet2.Cells[count, 3].Value = subjects[q].typeOfExercise;

ObjWorkSheet2.Cells[count, 4].Value = subjects[q].term1;

ObjWorkSheet2.Cells[count, 5].Value = subjects[q].term2;

ObjWorkSheet2.Cells[count, 6].Value = subjects[q].group;

ObjWorkSheet2.Cells[count, 7].Value = subjects[q].quantity;

ObjWorkSheet2.Cells[count, 2].Value = currentName;

count++;

}

subjects.Clear();

Microsoft.Office.Interop.Excel.Range excelcells1 = ObjWorkSheet2.get\_Range("A" + (count - 1), "K" + (count - 1));

excelcells1.Borders[XlBordersIndex.xlEdgeBottom].Weight = Microsoft.Office.Interop.Excel.XlBorderWeight.xlThick;

}

isExist = false;

}

}

for (int z = 14; z <= count + 1; z++)

{

if (ObjWorkSheet2.Cells[z, 2].Text != "")

{

ObjWorkSheet2.Cells[z, 1].Value = id;

id++;

}

}

Microsoft.Office.Interop.Excel.Range excelcells2 = ObjWorkSheet2.get\_Range("K" + 14, "K" + (count - 1));

excelcells2.Borders[XlBordersIndex.xlEdgeRight].Weight = Microsoft.Office.Interop.Excel.XlBorderWeight.xlThick;

Microsoft.Office.Interop.Excel.Range excelcells4 = ObjWorkSheet2.get\_Range("A" + count, "K" + 700);

excelcells4.Borders[XlBordersIndex.xlInsideHorizontal].LineStyle = Microsoft.Office.Interop.Excel.XlLineStyle.xlLineStyleNone;

excelcells4.Borders[XlBordersIndex.xlInsideVertical].LineStyle = Microsoft.Office.Interop.Excel.XlLineStyle.xlLineStyleNone;

ObjWorkBook1.Close();

ObjWorkBook2.SaveAs(pathToSave + "\\Приложениексеместровомуплану " + educYear);

ObjExcel.Visible = true;

}

}

}

КлассResource.cs

using System.IO;

namespace WindowsFormsApplication7

{

classResourсe

{

///<summary>

/// Метод, который загружает из ресурсов шаблон пустого до-кумента «Приложение» и делает его удоб-ным для использования.

///</summary>

///<returns></returns>

publicstaticstringgetUri()

{

var resource = new { Name = "Приложение.xltx", Buff = WindowsFormsApplication7.Properties.Resources.Приложение};

vartempDirectory = Path.GetDirectoryName(Path.GetTempFileName());

var path = string.Format("{0}\\{1}", tempDirectory, resource.Name);

if (!File.Exists(path) || File.ReadAllBytes(path).Length.Equals(0))

{

var stream = newMemoryStream(resource.Buff);

using (var file = newFileStream(path, FileMode.Create))

{

var buffer = newbyte[4096];

intbytesRead;

while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)

{

file.Write(buffer, 0, bytesRead);

}

}

}

return path;

}

}

}

КлассSubject.cs

namespace WindowsFormsApplication7

{

classSubject

{

publicstringtypeOfExercise; //Видзанятиялаб/практ

publicstring group; // Номергруппы

publicstring quantity; // Количествочеловеквгруппе

publicstring term1; // Количество часов в 1й чверти

publicstring term2; // Количество часов во 2й чверти

///<summary>

///Конструкторкласса Subject

///</summary>

public Subject(stringtypeOfExercise, string group, string quantity, string term1, string term2)

{

this.typeOfExercise = typeOfExercise;

this.group = group;

this.quantity = quantity;

this.term1 = term1;

this.term2 = term2;

}

}

}

# ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ОПЕРАТОРА

**П.В.1 Назначение программы**

Данная работа направлена на решение проблемы ведения документации. Она предназначена для применения в учебных заведениях для облегчения ведения документации при подготовке к проведению учебного процесса. Функциональным назначением (задачей) программы является предоставление преподавателю возможности формировать определённую документацию не в ручном, а в полуавтоматическом режиме.

В конечном итоге программа, в качестве входных данных принимает документ «Учебное поручение», а результатом работы является документ «Приложение к семестровому учебному плану».

**П.В.2 Условие выполнения программы**

В состав технических средств должен входить х64 совместимый компьютер, включающий в себя:

1) процессор с тактовой частотой 2.7Ghz;

2) оперативную память объемом 4GB, не менее;

3) свободного дискового пространства 100MB

4) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

5) клавиатуру и мышь;

6) жесткий диск.

На ПК должны быть установлены:

1) ОСWindows 10x64;

2) .Net 4.5;

3) WF 2.0;

4) MSExcel 2010.

**П.В.3 Выполнение программы**

1) Запустите приложение «CourseProject.exe»;

2) Укажите учебный год, для которого будете формировать «Приложение»;

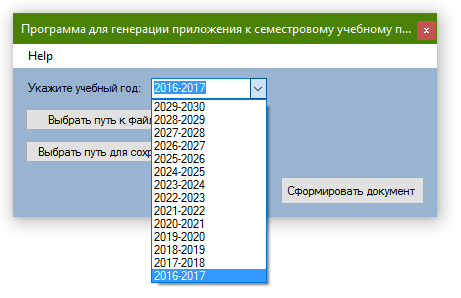


Рисунок П.В.1 – Выбор учебного года

3) Нажмите на кнопку «Выбрать путь к файлу-источнику». Выберите документ указанного формата, где у вас находиться информация согласно синтаксису, который указан в техническом задании (ПРИЛОЖЕНИЕ А – Приложение к ТЗ 1);

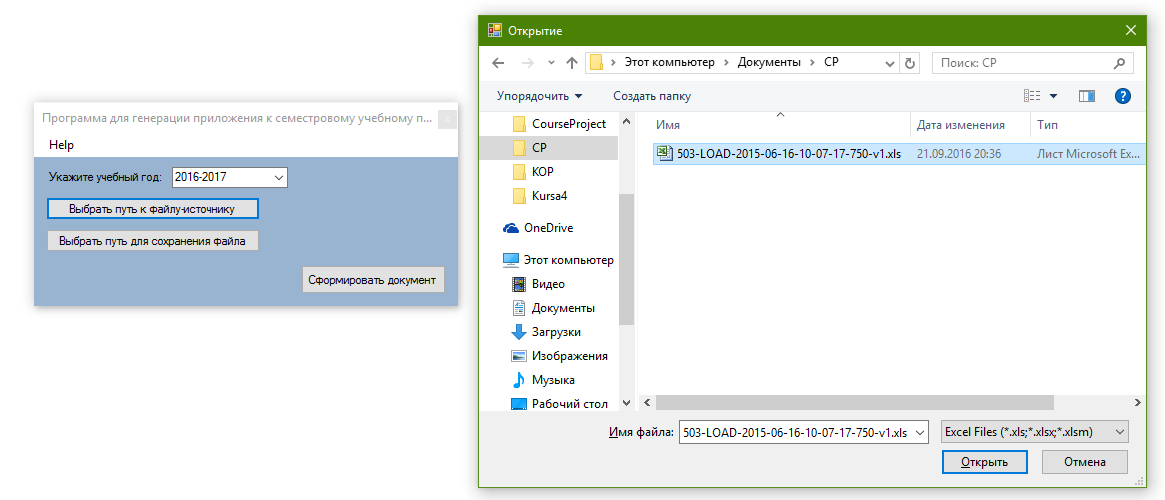


Рисунок П.В.2 – Выбор файла-источника

4) Нажмите на кнопку «Выбрать путь для сохранения файла». Выберите путь, куда следует сохранить готовый документ «Приложение»;

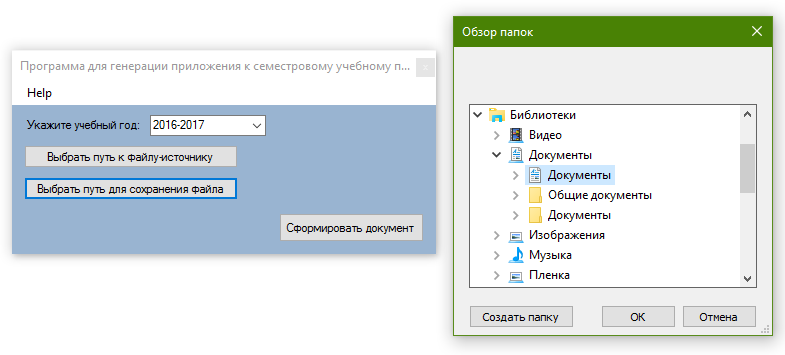


Рисунок П.В.3 – Выбор пути сохранения результата

5) Нажмите на кнопку «Сформировать документ»;

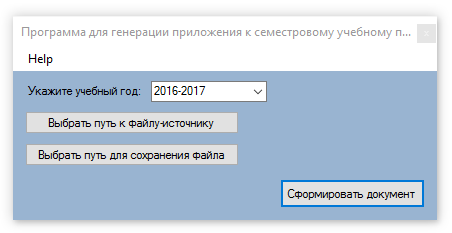


Рисунок П.В.4 – Сформировать документ

6) После того, как документ будет сформирован, он откроется, вы сможете его просмотреть и отредоктировать (если понадобиться);

7) Закрыть приложение.

**П.В.4 Сообщениея оператору**

В ходе выполнения программы оператору могут быть выданы следующие сообщения:

1) При нажатии на кнопку «Сформировать документ», до этого не выбрав путь к файлу источнику рисунок П.В.5;

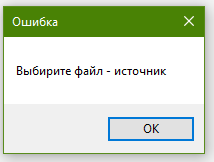


Рисунок П.В.5 – Ошибка 1

2) При нажатии на кнопку «Сформировать документ», до этого не выбрав путь для сохранения результата рисунок П.В.6;

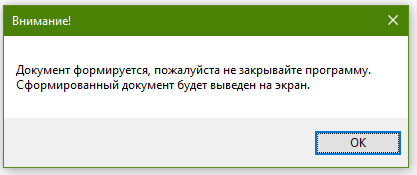


Рисунок П.В.6 – Ошибка 2

3) Если в процессе формирование документа (до того как будет он будет выведен на экран) попытаться закрыть программу то пользователь увидит сообщение подтверждение закрытия программы рисунок П.В.7.

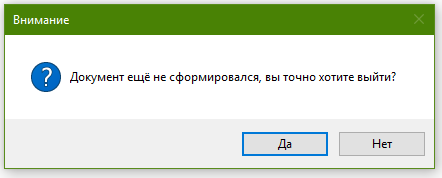


Рисунок П.В.7 – Подтверждение закрытия программы

# ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ

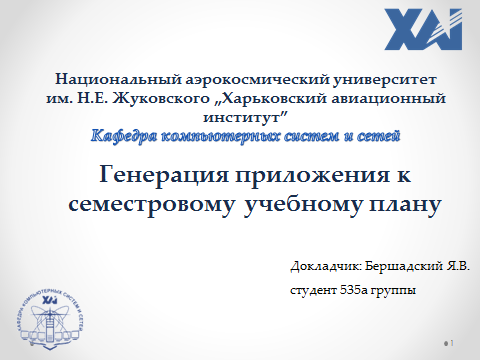


Рисунок П.Г.1 – Слайд 1



Рисунок П.Г.2 – Слайд 2

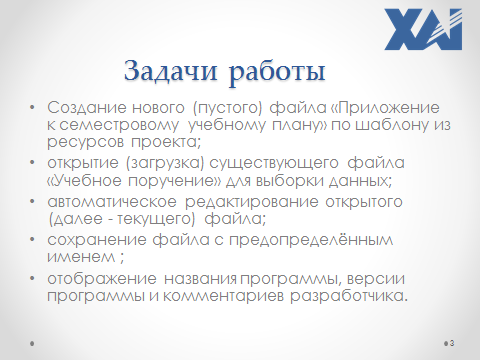


Рисунок П.Г.3 – Слайд 3



Рисунок П.Г.4 – Слайд 4



Рисунок П.Г.5 – Слайд 5



Рисунок П.Г.6 – Слайд 6



Рисунок П.Г.7 – Слайд 7



Рисунок П.Г.8 – Слайд 8



Рисунок П.Г.9 – Слайд 9

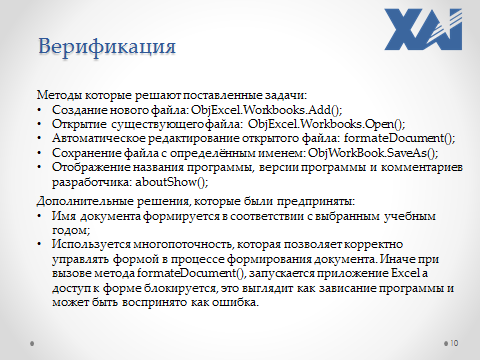


Рисунок П.Г.10 – Слайд 10

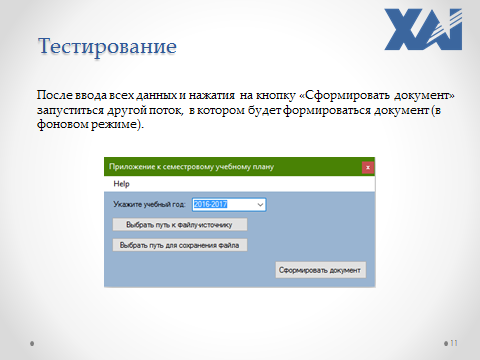


Рисунок П.Г.11 – Слайд 11

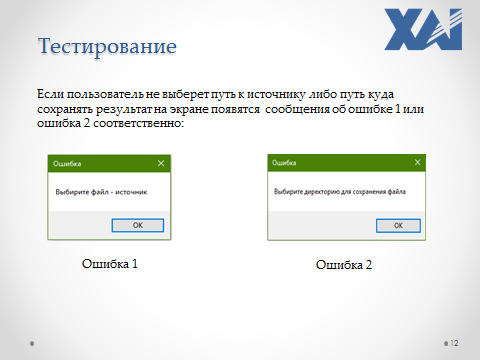


Рисунок П.Г.12 – Слайд 12

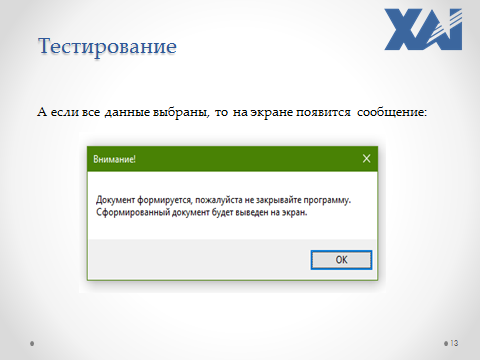


Рисунок П.Г.13 – Слайд 13

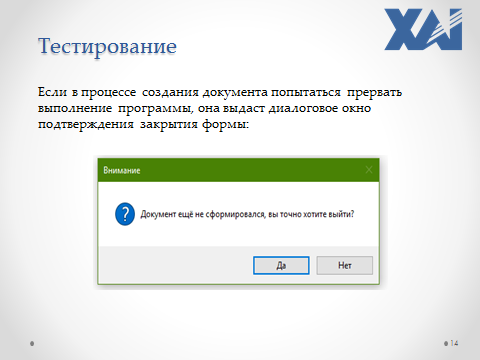


Рисунок П.Г.14 – Слайд 14

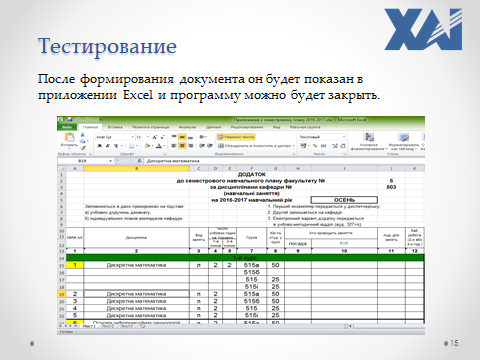


Рисунок П.Г.15 – Слайд 15

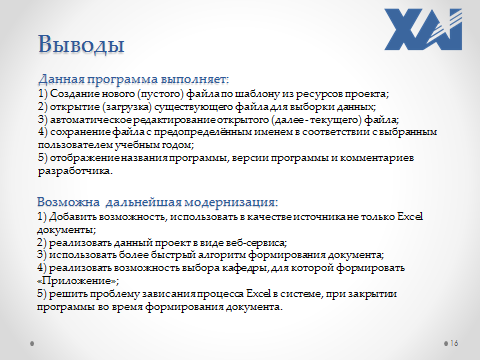


Рисунок П.Г.16 – Слайд 16



Рисунок П.Г.17 – Слайд 17