

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**СЕРВИС ФОРМИРОВАНИЯ И ПРОВЕРКИ
КОРРЕКТНОСТИ ЭЛЕКТРОННЫХ СООБЩЕНИЙ**

Студент Барсуков Никита Иванович
(подпись, дата) 71.06.24

Руководитель Федюнин Р.Н.
(подпись, дата) 71.06.24

Нормоконтролёр Бычков А.С.
(подпись, дата) 71.06.24

Работа допущена к защите (протокол заседания кафедры от 27.05.24 г. №

11)

Заведующий кафедрой

(подпись)

Митрохин М.А.

Работа защищена с отметкой отлично (протокол заседания ГЭК от

17.06.2024 № 23)

Секретарь ГЭК

(подпись)

Гурин Е.И.

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

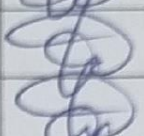
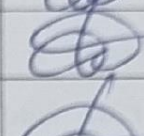
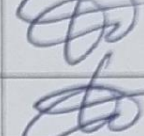
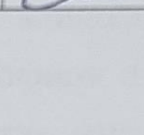
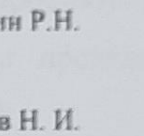
«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 11 » 04 2024г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

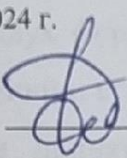
1. Барсуков Никита Иванович гр. 20ВВП1 факультета вычислительной техники.
2. Тема работы «Сервис формирования и проверки корректности электронных сообщений»
Тема утверждена приказом ПГУ №405/о от "19" апреля 2024 г.
3. Руководитель бакалаврской работы к.т.н., доцент Федюнин Р.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать приложение «Сервис формирования и проверки корректности электронных сообщений», выполняющее отправку и прием сообщений от информационной системы предприятия. Приложение должно взаимодействовать с технологией IBM MQ. Сообщений необходимо создавать в формате SOAP и проверять на корректность создаваемые поля структуры XML.
5. Перечень подлежащих разработке вопросов

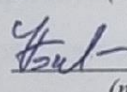
Требуется произвести анализ предметной области, изучить аналоги существующих приложений формирования сообщений, изучить принципы интеграции приложений, изучить способы проектирования и разработки приложений с применением паттерна MVP. Необходимо изучить процесс формирования SOAP сообщений и их отправку в менеджер очередей IBM MQ.

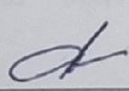
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	15%	26.04.2024-30.04.2024	
2	Анализ требований	4%	01.05.2024-02.05.2024	
3	Выбор технологий, применяемых при проектировании и разработке	15%	03.05.2024-07.05.2024	
4	Описание алгоритма	13%	08.05.2024-11.05.2024	
5	Разработка приложения «Сервис формирования и проверки корректности электронных сообщений»	45%	12.05.2024-29.05.2024	
6	Оформление пояснительной записки	8%	30.05.2024-31.05.2024	

Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы  19.04.24 Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент  19.04.24 Барсуков Н. И.
(подпись, дата)

Работу к защите допустить
Декан факультета  15.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 117 листов, 53 рисунка, 1 таблицу, 5 источников, 3 приложения

СООБЩЕНИЕ, ОЧЕРЕДЬ, МЕНЕДЖЕР ОЧЕРЕДЕЙ, ФОРМИРОВАНИЕ, ПЕРЕДАЧА ДАННЫХ, ИНФОРМАЦИЯ, АСИНХРОННОСТЬ, SOAP, API, XML

Цель работы – разработать сервис по формированию и проверки корректности электронных сообщений в рамках информационной системы предприятия.

Для достижения поставленной задачи рассматривается возможность разработки приложения обладающее следующим функционалом: возможность формирования сообщения в двух форматах, для внутренней отправки и для внешней, учет размера сообщений и его контроль, возможность просматривать содержимое отправленных и принятых сообщений, а также проведение нагрузочных тестов с целью замера скорости отправки сообщений.

Приложение также должно обладать удобным пользовательским интерфейсом.

В результате было разработано приложение выполняющее формирование и отправки электронных сообщений, и имеющее соответствующий функционал для проведения нагрузочных тестов.

					ПГУ1.090301.02.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Барсуков Н.И.		19.04.24	Сервис формирования и проверки корректности электронных сообщений	Лит.	Лист	Листов
Провер.		Федюнин Р.Н.		19.04.24			2	118
Н. Контр		Бычков А.С.		19.04.24		ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		19.04.24	Пояснительная записка	гр. 208ВП1		

Содержание

Перечень сокращений и обозначений	5
Введение.....	6
1 Сервис формирования и проверки корректности электронных сообщений..	7
1.1 Анализ предметной области	7
1.2 Актуальность	11
1.3 Пользователи и сценарии.....	13
1.4 Анализ аналогов.....	15
1.5 Постановка задачи	21
1.6 Разработка сервиса.....	22
1.6.1 Общая архитектура системы	22
1.6.2 Архитектура приложения	24
1.6.3 Проектирования пользовательского интерфейса	27
1.6.4 Проектирование классов представителя	29
1.6.5 Проектирования бизнес логики.....	31
1.7 Выбор инструментов разработки.	33
1.7.1 Выбор языка программирования.	33
1.7.2 Выбор среды для разработки.....	35
1.7.3 Выбор Арі для создания пользовательского интерфейса.....	36
1.7.4 СУБД.....	36
1.8 Разработка приложения.....	37
1.8.1 Разработка представителей.....	37
1.8.2 Разработка моделей	47
1.8.3 Разработка интерфейса.....	55
1.9 Тестирование приложения	65
2 Безопасность жизнедеятельности.....	73
2.1 Эргономика рабочего пространства.....	73
2.1.1 Правила организации рабочего места	73
2.1.2 Освещение	74

2.1.3 Климматические условия.....	74
2.2 Электробезопасность приборов.....	74
Приложение А – UML-диаграммы.....	77
Приложение Б – Листинг программы	82
Приложение В – Презентация.....	110

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ОС – операционная система

ПЭВМ – персональная электронно-вычислительная машина

СУБД – система управления базами данных

ЭС – электронное сообщение

ЭМП – электромагнитное поле

API (Application Programming Interface) - интерфейс программирования приложения

CMS (Cryptographic Message Syntax) – стандарт файлов для защищенных крипто-сообщений

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

JSON (JavaScript Object Notation) – формат обмена данными основанный на JavaScript

MQ (Message Queue) – очередь сообщений

MVP (Model View Presenter) – модель представление представитель

PKCS (Public Key Cryptography Standards) – стандарты криптографии с открытым ключом

SOAP (Simple Object Access Protocol) — протокол доступа к объектам

TCP (Transmission Control Protocol) – протокол управления передачей

XAdES (XML Advanced Electronic Signatures) – усовершенствованная электронная подпись

XML (eXtensible Markup Language) – расширяемый язык разметки

					ПГУ1.090301.02.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Обмен сообщениями – это очень важный процесс коммуникации, использующийся как в простых приложениях, так и в сложных информационных системах. Основания его задача – это передача определенной информации в кратчайшие сроки без потерь. В данном процессе участвует множество сущностей. Одни генерируют информацию, другие её потребляют. Предприятия все чаще интегрируют в свои технологические процессы комплексное программное обеспечение, которое призвано упростить многие рутинные задачи. Основной проблемой данного процесса является стандартизация информации. Разные приложения используют разные форматы данных, что значительно усложняет процесс передачи и хранения.

Для доставки сообщений, в информационной системе предприятия существует адаптер, который берет на себя функции коллектора для сообщений. Его основная задача – это быстрая маршрутизация данных. Он задает критерии для обрабатываемых сообщений, тем самым решая проблему с многообразием форматов данных. Однако, необходим сервис, который будет заниматься формированием сообщений нужного формата.

Для этой цели, в данной выпускной квалификационной работе, был рассмотрен сервис формирования и проверки корректности электронных сообщений.

1 Сервис формирования и проверки корректности электронных сообщений

1.1 Анализ предметной области

Обмен сообщениями – сложный процесс, на который влияет множество факторов. Все они оказывают комплексное воздействие на информационную систему, участвующую в передаче данных. В случае, когда система является локальной рисков значительно меньше, чем, когда она разнесена по различным средам. Разработчик, как правило, должен найти решение для целого ряда проблем, возникающих при разработке сложного интеграционного решения, таких как:

- надежность передачи данных. Распределенная вычислительная среда значительно отличается от локальной. Одним из таких является удаленность компонентов друг от друга, что обязывает передавать данные через сторонние средства. Основным фактором риска в данном случае будут являться задержки и потери информации, что может критически сказаться на работе вычислительной среды;

- скорость передаваемой информации. Обмен сообщениями может происходить как локально в рамках одной системы, так и на больших расстояниях. Разработчик при проектировании системы должен учитывать данный факт;

- различия в форматах данных. Информационные системы участвующие в обмене информацией могут сильно различаться в архитектурном плане, например, использовать разные форматы данных.

Для решения данных проблем в процессе обмена сообщениями участвует ряд сущностей, которые в совокупности минимизируют связанные риски. Для полноценного понимания рассматриваемой темы, необходимо подробно разобрать несколько основных понятий.

					ПГУ1.090301.02.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Обмен сообщениями – это процесс быстрого асинхронного обмена данными между различными системами с минимизацией потенциальных потерь [1]. Программы или целые системы взаимодействуют между собой путем обмена некой структурированной информацией, называемой сообщением. Обмен происходит в так называемом канале или очереди. Это некий логический маршрут, соединяющий компоненты или целые системы для транспортировки данных. Таким образом субъект, отправляющий сообщения, кладет их не в какую-то абстрактную систему управления сообщениями, а в конкретный канал, также и получатель следит за обновлениями конкретных каналов. Таких каналов может быть несколько, для разных типов сообщений. У каждого канала есть свой идентификатор известный как отправителю, так и получателю. Не стоит считать, что каждый отдельный канал будет иметь и отдельную физическую интерпретацию. Зачастую он представляет собой логический адрес. Таким образом несколько каналов могут иметь единую физическую реализацию.

В рамках системы обмена сообщениями каналы не заданы заранее. Их необходимо сконфигурировать заранее, на этапе проектирования. Некоторые системы позволяют создавать новые каналы уже на этапе эксплуатации, однако их распространение является проблематичной задачей. Поэтому их количество и специализацию необходимо назначить заранее. Планируя каналы необходимо знать, что каждый из них требует отпрядённого объема оперативной или дисковой памяти. Невозможно создать бесконечное количество каналов. Поэтому необходимо установить жесткий лимит и следить за его поддержкой.

В обмене так же участвуют и сущности, отвечающие за производство и потребление данных. Называются они достаточно лаконично, прекрасно отражая свою сущность, а именно «производитель» и «потребитель». Существуют также и другие определения, например, «отправитель» и «получатель», а если используется взаимодействие по принципу «публикация-подписка», то обозначаются они как «издатель» и «подписчик». Если конкретизировать, то

первый генерирует информацию и затем помещает её в очередь, а второй забирает сообщений из канала, тем самым его оттуда удаляя.

Остановимся на процессе отправки сообщения и что он из себя представляет. Обычно для передачи информации используют набор байтов. Однако необходимо, чтобы кто-то этот набор байт сформировал и к тому же сделал это правильно. Затем его необходимо передать и преобразовать в читаемый для пользователя вид. Чтобы у производителя и потребителя не возникало конфликтов необходимо также согласовать формат сообщений.

В сообщении, как правило, есть тело и заголовок. Заголовок состоит из данных необходимых для маршрутизации и идентификации сообщения. В нем находится информация об отправителе, получателе, размере данных и т.п. Заголовок важен в первую очередь системе обмена сообщениями, получатель данную информацию зачастую игнорирует. В свою очередь, в теле сообщения может находиться любая информация, которая игнорируется системой, но не получателем [1].

Данная концепция существует уже давно. В классической почтовой службе письма также состоят из нескольких частей. Это логично и просто для понимания. Для системы обмена сообщениями нет никакой разницы в сообщениях: в теле находятся какие-то данные, а в заголовке параметры для доставки. Однако для человека все сообщения имеют разный смысл, в зависимости от их типа. Например, существуют сообщения как для передачи информации, так и для оповещения об определенных событиях.

Сам процесс обмена ложится на специальное программное обеспечение, называемое брокер сообщений или система обмена сообщениями. По своему принципу, он напоминает систему управления базами данных. В обоих случаях система структурирует и систематизирует хранимую и передаваемую информацию. В базах данных эту роль играет схема базы, а в брокере формат сообщения и способ передачи в канале. Так же система обмена сообщениями обеспечивает надежность передачи информации. Достигается это путем

повторной отправки сообщения, в том случае, если в первый раз оно не достигло получателя. Частота и продолжительность отправки зависит от настроек конкретного брокера.

Данный подход усложняет процесс обмена сообщениями, однако, путем такого делегирования можно достичь надежность в передаче данных. Также он обладает целым рядом преимуществ:

– удаленный обмен данными. С помощью обмена сообщениями можно наладить взаимодействие между отдельными приложениями. Для обмена передачи информации по каналу связи данные необходимо сериализовывать в поток байт. С помощью системы управления сообщениями приложение может делегировать неё эту ответственность, тем самым освободив свои ресурсы для других задач;

– асинхронная отправка сообщений. Используя принцип «отправил и забыл», можно легко наладить асинхронное взаимодействие между приложениями. Это достигается тем, что отправитель не дожидается доставки сообщения, вместо этого он кладет сообщение в очередь и приступает к другим задачам;

– возможность использования принципа «передача с промежуточным хранением». Данный принцип заключается в том, что при отправке сообщения, система обмена сообщениями делает резервную копию сообщения на компьютере отправителя, а затем и на компьютере получателя. Таким образом можно достичь надежности в передачи информации и не беспокоится о её потере. Более того, если сообщение по какой-то причине не будет доставлено с первого раза, то отправка будет продолжаться до тех пор, пока оно не будет доставлено окончательно;

– обеспечение работы без сети. Возникают ситуации, когда есть необходимость в написании приложения обеспечивающего синхронизацию данных на сервере в случае непредвиденного отключения от сети. В таком

случае, все данные необходимые для синхронизации помещаются в очередь до тех пор, пока доступ к сети не будет восстановлен.

Помимо достоинств у технологии асинхронного обмена сообщениями есть также и недостатки, некоторые из которых, являются следствием специфики асинхронной модели в целом, однако другие являются субъективными, зависящими от реализации брокера. К некоторым из недостатков относятся:

- относительно сложная модель реализации. Необходимо понимать принципы работы событийно управляемой модели, когда в приложении появляется множество обработчиков событий. Данную систему проблематично отлаживать, а также она плодит множество сущностей;

- сложность в контроле порядка входящих сообщений. Все сообщения доходят за разный промежуток времени, в ситуации, когда порядок не имеет значения это не играет роли, но когда важна четкая последовательность сообщений — это может вызвать серьезные проблемы;

- необходимость поддержки синхронной модели. В некоторых ситуациях важно чтобы сообщений обрабатывалось немедленно, а не через определенный промежуток времени, например, запрос на выполнение финансовой операции.

Асинхронный обмен сообщениями имеет как свои достоинства, так и недостатки, поэтому при проектировании конкретного решения необходимо все тщательно взвесить.

1.2 Актуальность

Тема интеграции программного обеспечения с каждым годом становится более актуальной. Разработчики стремятся разделить функциональные возможности своих приложений на различные независимые компоненты. Это необходимо для более гибкой разработки и поддержки программного обеспечения, а также реализации распределенных информационных систем. Гораздо удобнее поддерживать и отлаживать небольшой компонент, чем разрабатывать огромный монолитный блок.

					<i>ПГУ1.090301.02.001 ПЗ</i>	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

Чтобы обеспечить надежную связь этих компонентов друг с другом и всей системой в целом, используют процесс обмена сообщениями. Специальные сервисы формируют и проверяют сообщения, а отдельные системы занимаются их пересылкой и маршрутизацией. Таким образом с самих компонентов снимается ответственность за отправку и доставку информации. Они могут сосредоточиться на выполнении своих прикладных задач.

Также существует проблема обеспечения обмена сообщениями между разными предприятиями. Необходимо стандартизировать формат данных. Такая проблемы возникает и при обмене в рамках одного предприятия, когда существует много структурных подразделений со своим программным обеспечением.

Для обеспечения бесперебойного обмена сообщениями в рамках сложной информационной системы используют специальные адаптеры, которые берут на себя задачи по транспортировке данных. Они используют единый формат данных тем самым решая проблему с различием приложений.

Информация к адаптерам поступает через сервисы формирования и проверки корректности сообщений, которые подключаясь к определенной очереди сообщений могут обеспечить надежный обмен данными. Более того, именно такая модель обеспечивает асинхронное взаимодействие. Адаптер работает с очередями сообщений с использованием событийно-ориентированной модели. Приложение подписывается на интересующие ему темы и затем читает сообщения по мере поступления. При этом отправитель не обязан дожидаться, когда получатель его прочтет.

Задача интеграции крупной информационной системы тесна связана с процессом обмена сообщениями, ведь именно таким образом и обеспечивается связь различных компонентов как самой системы, так и различных приложений в рамках целого предприятия.

Изучение данной темы актуальна для студентов тем, что потенциальный работодатель будет заинтересован в сотрудниках, имеющих опыт и соответствующий набор навыков и знаний.

1.3 Пользователи и сценарии

На рисунке 1 представлена диаграмма вариантов использования приложения.

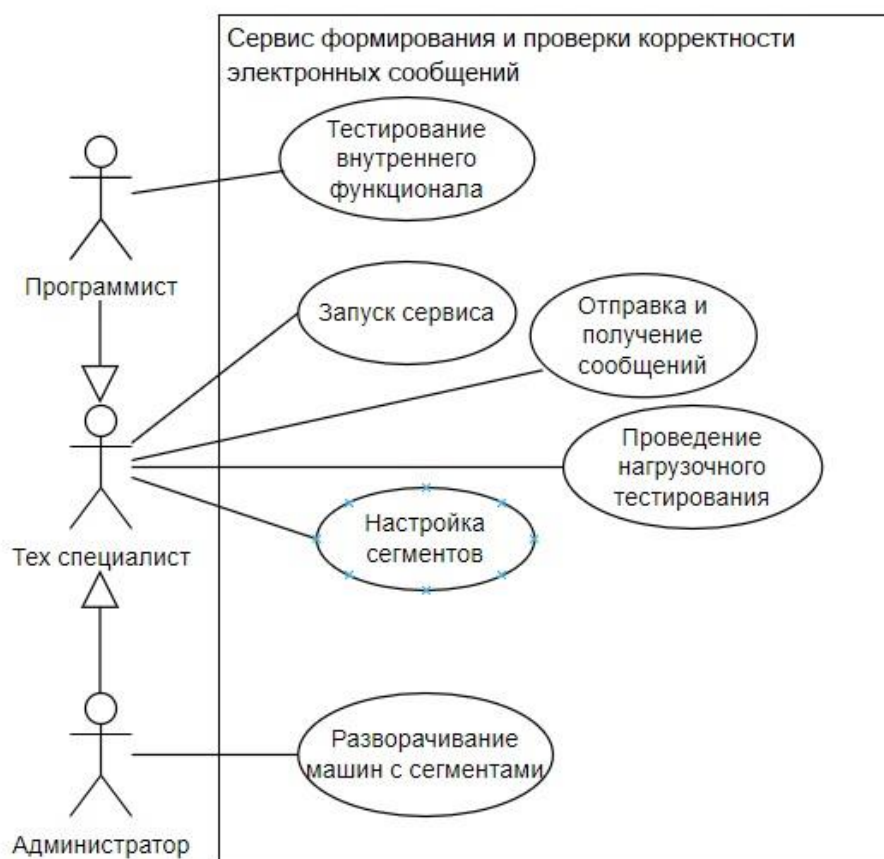


Рисунок 1 - UML диаграмма вариантов использования

Перед тем как приступить к выбору решения, нужно определиться с тем, кто и как будет использовать наш программный продукт. Исходя из анализа предметной области, мы можем выделить следующие группы пользователей:

– технические специалисты, использующие сервис для проверки корректного разворачивания стороннего программного обеспечения;

– администраторы прикладных сервисов, которые будут использовать сервис для отправки сообщений как внутри своей организации, так и за её пределами, а также для проверки более крупного функционала, взаимодействующего с информационной системой предприятия;

– программисты, которые будут использовать сервис для проведения нагрузочных тестов для своих программ связанных с менеджерами очередей, формированием сообщений, криптографией и тп.

Исходя из описания групп пользователей выделим для каждой группы основные варианты взаимодействия с приложением:

– технические специалисты:

- 1) запуск сервиса,
- 2) настройка сегментов отправки,
- 3) получение результатов от нагрузочного тестирования;
- 4) отправка и получение сообщений.

– администраторы прикладных сервисов:

- 1) запуск сервиса,
- 2) настройка сегментов отправки и получения,
- 3) разворачивание машин с сегментами,
- 4) формирование сообщений,
- 5) отправка и получение сообщений.

– программисты:

- 1) запуск сервиса,
- 2) настройка сегментов отправки и получения,
- 3) настройка криптографических библиотек,
- 4) отправка и получение сообщений;
- 5) тестирование внутреннего функционала.

Сценарии могут изменяться в зависимости от требований конкретного заказчика.

В независимости от типа пользователя, можно выделить ряд общих для всех сценариев:

- пользователь выполняет настройку сегмента отправки, указывая в настройках параметры (ip, порт, название очереди сообщений и тп.);
- пользователь выполняет настройку криптографической библиотеки, выбирая между встроенной в ОС и подключением сторонней;
- пользователь выполняет отправку сообщения, предварительно выбрав тип отправки;
- пользователь хочет сформировать электронный документ для отправки, на который для этого ставит подпись и добавляет дополнительные вложения;
- пользователь настраивает периодичность отправки сообщений, количество повторов, лимиты очереди принятия, формат и тип заголовка сообщения;
- пользователь хочет отправить готовое, заранее сформированное сообщение;
- пользователь хочет провести нагрузочное тестирование с последующей выгрузкой отчета о времени выполнения операции, отправленных сообщениях, потерянных сообщениях и скорости обработки сообщений.

1.4 Анализ аналогов

В рамках выпускной квалификационной работы необходимо провести сравнительный анализ различных средств работы с электронными сообщениями, к которым должны предъявляться следующие требования:

- удобный пользовательский интерфейс;
- поддержка работы с технологией MQ;
- формирование сообщения нужного формата с последующей подписью средствами криптографии.

Исходя из требований, предъявляемых к системам формирования сообщений были найдены аналоги, приведенные в таблице 1.

					ПГУ1.090301.02.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 1 Анализ аналогов

Аналог	Поддержка MQ	Удобство использования	Процесс формирования сообщения
<i>JMeter</i>	Да, с использованием сторонних плагинов	Отсутствует интерфейс	Формирует сообщение, может работать со средствами ЭП.
<i>RFHUtil</i>	Да	Очень неинтуитивный интерфейс	Формирование происходит вручную
<i>Thunderbird</i>	Нет	Приемлемый интерфейс	Формирует сообщение, может работать со средствами ЭП.

JMeter – это программный набор инструментов для проведения различных типов нагрузочного тестирования, регрессионного тестирования, функционального и т.п. Так как написан на Java, то не зависит от платформы. Благодаря тому, что он хранит свои планы тестирования в XML, то пользователь имеет возможность написать свой план с помощью любого текстового редактора. Работает по принципу эмуляции группы пользователей, которые отправляют запросы на сервер и затем забирает оттуда статистику.

JMeter – хороший инструмент для проведения различных типов нагрузочных тестов с использованием нужного нам функционала, а именно формирование и отправка сообщений в менеджер очередей. Также способен проводить нагружать различные типы соединений, таких как TCP, HTTP, SOAP и т.п.

У данного инструмента много преимуществ, например, он имеет открытый исходный код, что позволяет писать собственные расширения, а также дорабатывать уже существующий функционал.

Поддержка MQ. JMeter имеет поддержку такого известного менеджера очередей как RabbitMQ.

Удобство использования. Тут возникает огромный камень преткновения – интерфейс отсутствует как таковой. Помимо этого, приходится писать много кода, что крайне нам не подходит.

Процесс формирования сообщения. Позволяет формировать сообщение нужного формата, а также подключать плагины для работы с криптографией.

Преимущества:

- использовать готовые плагины и иметь возможность написать свой;
- пользоваться преимуществами огромного сообщества по всему миру;
- обращаться к API и проверять содержимое ответов;
- преобразовывать JSON, YAML и CSV файлы и использовать данные из них;
- подключаться к базам данных и сервису очередей RabbitMQ.

Недостатки:

- отсутствие какого-либо пользовательского интерфейса;
- требование наличия у пользователя навыков в программировании.

В целом, JMeter – это отличный набор инструментов для работы с сообщениями, он может стать отличным решением для тех, кому не нужен интерфейс как таковой, и кто хорошо разбирается в написании программного кода. Однако, конкретно в нашем случае он не подходит, так как конечный пользователь может не иметь навыков в программировании, но иметь знания в сфере администрировании развертывания вычислительных систем.

RFHUtil – это программная утилита, входящая в состав SupportPac IH03. Основное предназначение данной программы — это чтение очереди сообщений и запись туда информации. Имеет широкий спектр функций по работе с заголовками сообщений. RFHUtil позволяет отображать данные сообщения в различных форматах, а также редактировать их.

На рисунке 2 приведен интерфейс программы RFHUtil.

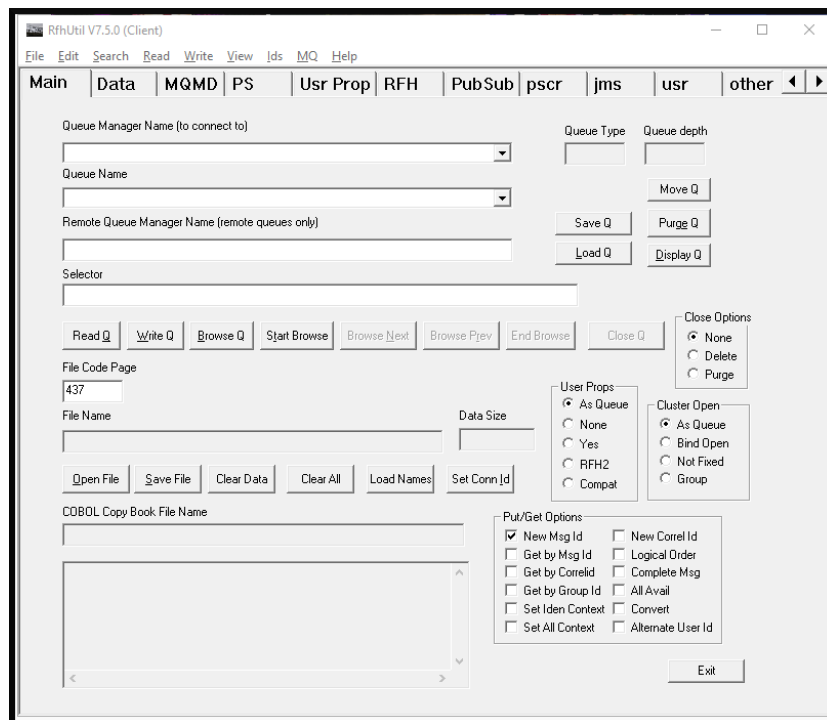


Рисунок 2 – Утилита RFHUtil

Как и предыдущее решение, RFHUtil обладает как преимуществами, так и недостатками. Его сравнение с остальными продуктами выглядит следующим образом:

Поддержка MQ. Поддерживает работу с различными менеджерами очередей, в том числе RabbitMQ и WebSphere MQ.

Удобство использования. Присутствует пользовательский интерфейс, однако он крайне перегружен. Присутствует навигационная панель в верхней части приложения, что упрощает процесс использования, однако однообразие цветовой палитры и нагромождение кнопок вызывает сложность в использовании у новых пользователей.

Процесс формирования сообщения. Все формирование происходит вручную, отсутствуют шаблоны сообщений, где пользователь может задать нужные ему параметры, прикрепить документ и отправить.

Преимущества:

- наличие пользовательского интерфейса;
- работает с различными форматами сообщений;

– позволяет выполнять широкий спектр манипуляций с сообщениями с заголовками и без;

– имеет возможность выгрузки очереди в файл и обратно.

Недостатки:

– неинтуитивный интерфейс;

– формирование сообщения происходит вручную;

– отсутствие криптографии.

Несмотря на свои недостатки, RFHUtil все ещё остается мощным средством для формирования и отправки сообщений. В отличие от предыдущего аналога он обладает пользовательским интерфейсом, хоть и не самым понятным. Есть широкий спектр возможностей для работы с ЭД, однако отсутствие криптографии и не самый приятный внешний вид может оттолкнуть потенциальных соискателей.

Thunderbird – это почтовый клиент с открытым исходным кодом поддерживающий такие почтовые стандарты как: POP, IMAP, LDAP, S/MIME и др.

На рисунке 3 приведен интерфейс почтового клиента Thunderbird.

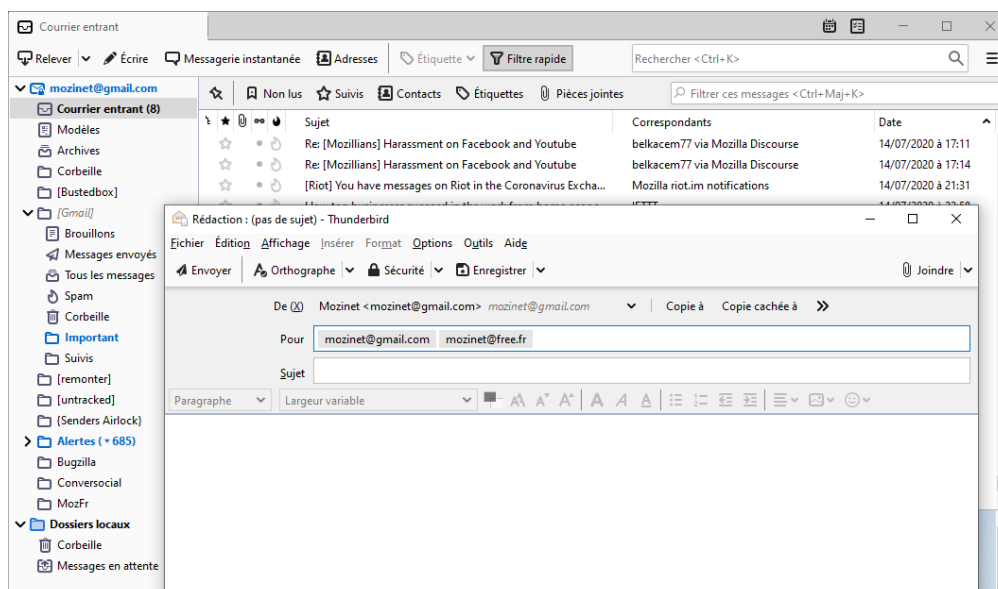


Рисунок 3 – почтовый клиент Thunderbird

Сравнение Thunderbird с остальными аналогами.

Поддержка MQ. Поддержка данной технологии отсутствует.

Удобство использования. Thunderbird обладает удобным и понятным пользовательским интерфейсом. Имеются папки для писем, присутствует возможность сортировки по различным параметрам, начиная от темы письма, заканчивая датой получения или отправки.

Процесс формирования сообщения. Поддерживает необходимые форматы сообщений, позволяет создавать шаблоны, а также их редактировать. Имеет встроенную поддержку криптографии по протоку S/MIME.

Достоинства:

- удобный по сравнению с остальными аналогами пользовательский интерфейс;
- отличный набор возможностей формирования электронных сообщений;
- поддержка современных сетевых и почтовых протоколов;
- возможность отправки защищенного сообщения;
- открытый исходный код.

К недостаткам следует отнести отсутствие работы с менеджерами очередей.

Пожалуй, Thunderbird является одним из самых сильных аналогов, однако, отсутствие менеджеров очередей делает его не самым предпочтительным вариантом в высокопроизводительной системе.

По итогу анализа можно сделать вывод, что все представленные аналоги предоставляют отличный функционал по работе с электронными сообщениями, однако они имеют как свои сильные, так и слабые стороны, которые покажут себя в конкретных задачах.

Например, если во главе угла стоит функциональная гибкость и возможность самому дорабатывать существующее API, то JMeter будет являться отличным выбором. Однако для этого нужны узкоспециализированные знания. Если же нужна хорошая утилита для просмотра содержимого очередей, то для этой цели уже идеально подойдет RFHUtil, но тогда вам придется столкнуться с

не самым интуитивным интерфейсом. В свою очередь, Thunderbird обладает приятным внешним видом и понятной навигацией. Он обладает сильными функциональными возможностями в плане формирования сообщений и использования криптографии. Однако ввиду отсутствия работы с очередями сообщений делает его менее предпочтительным для определенного круга пользователей.

Таким образом, при выборе решения нужно опираться на понятность пользовательского интерфейса, особенности работы с форматами сообщений, безопасность при отправке и получении и в зависимости от конкретных задач делать выбор в сторону одно из доступных решений.

1.5 Постановка задачи

Разработать приложение формирования и проверки корректности электронных сообщений, которое будет отправлять и принимать сообщения от информационной системы предприятия через специальный адаптер. Роль адаптера играет прикладное программное обеспечение реализующее технологию IBM MQ.

К функциональным требованиям, предъявляемым к приложению, относятся:

- формирование электронных сообщений в формате SOAP;
- проверка корректности сформированного сообщения (наличие и правильность заполнения заголовка, проверка размера прикрепляемых файлов);
- общий размер сообщения не должен превышать 100Мб;
- взаимодействие с менеджером очередей IBM MQ (отправка и прием сообщений из очереди);
- удобный пользовательский интерфейс;
- возможность взаимодействия с программой через консоль.

Также приложение должно запускаться на системах со следующими минимальными техническими требованиями:

					ПГУ1.090301.02.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

- процессор Intel Core i3-10105;
- встроенная графика;
- 6 Гб оперативной памяти;
- операционная система не ниже Windows 10.

1.6 Разработка сервиса

1.6.1 Общая архитектура системы

Для того чтобы определиться с конкретным решением, нужно подробно рассмотреть общую архитектуру всей системы, с которой предстоит взаимодействовать приложению. На рисунке Рисунок 4 представлена общая схема данной системы.

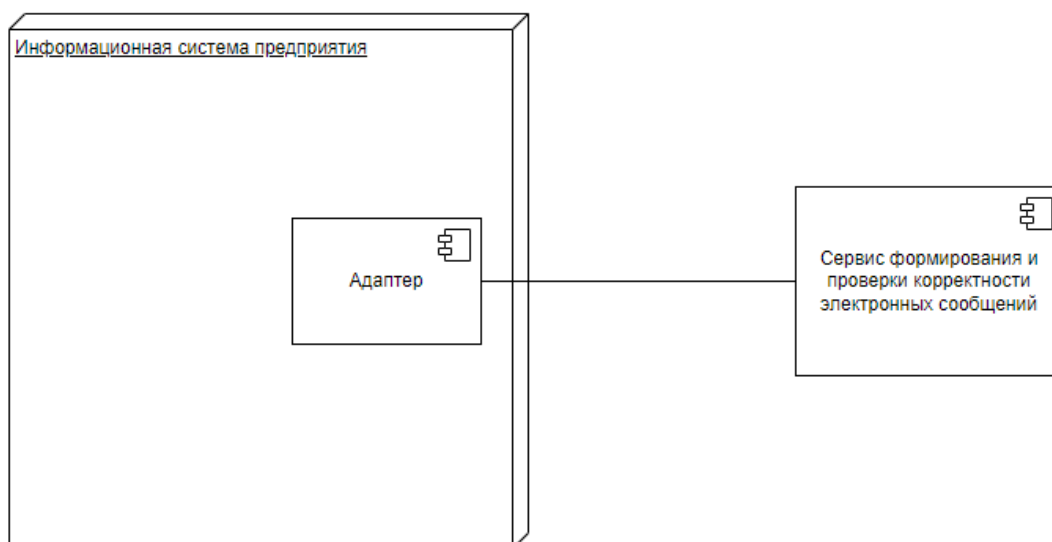


Рисунок 4 - Схема взаимодействия приложения с информационной системой

Существует информационная система предприятия, которая через специальный адаптер принимает и отправляет сообщения. Из рисунка выше можно увидеть, что моё приложение не имеет прямого доступа к данной системе, для этой цели существует адаптер. Это специальный компонент, реализующий менеджер очередей с использованием технологии IBM MQ. Именно с ним и будет работать моё решение.

Чтобы иметь четкое представление о том, как будет организовано взаимодействие приложения с адаптером, необходимо углубиться в технологию IBM MQ.

Основное предназначение очередей– это хранение сообщений до тех пор, пока приложение или получатель данных не будет готов к их прочтению. Таким образом ещё можно достичь асинхронности в обмене. Приложение-отправитель размещает сообщение в очереди и переключается на другие задачи, не дожидаясь ответа или как минимум доставки сообщения. Также использование технологии очередей упрощает разработку. Потому что нет необходимости в написании сложной транспортной логики, всю ответственность за доставку берет на себя менеджер очередей. Конечному приложению нужно лишь подключиться у нужной очереди и соответственно взаимодействовать уже с ней.

Сообщения в IBM MQ это набор байтов, оно состоит из основной части где находятся данные важные для получателя и мета-данных расположенных в заголовке. Размер сообщения ограничен и не может превышать 100 Мб [2]. Это своеобразный максимум выше которого сообщения быть никак не могут. На практике размер может быть ограничен настройками конкретной очереди, но он не должен конфликтовать с лимитом менеджера.

Огромную роль в передаче данных и налаживании взаимодействия между отправителем и получателем играет менеджер очередей. Это специализированное программное средство для управления очередями сообщений и их маршрутизацией. Для передачи данных между приложениями используются каналы. Они в большинстве своем односторонние, соответственно для приема и отправки необходимо выделить отдельные очереди. Как только в очереди приема появляется сообщение, то срабатывает событие, которое уведомляет получателя. Сам он не следит за обновлением очереди, чтобы не нарушался принцип асинхронности. Фактически используется событийно-ориентированная модель взаимодействия, когда приложение подписывается на интересующие ему очереди.

Ещё одна важная часть данной архитектуры – это формат сообщений. Здесь используется протокол SOAP. В данном формате сообщения передаются как XML-файл со специальной структурой [3]. Согласно протоколу, сообщение делится на четыре части:

- Envelope (конверт) – обязательный элемент, обозначающий что данный документ является частью протокола SOAP;

- Header (заголовок) – содержит служебную информацию о сообщении, состав и вид данной информации определяется конкретной системой обмена сообщениями;

- Body (тело) – непосредственно тело сообщения с полезной информацией для получателя;

- Fault (ошибка) – данное поле может быть необязательным, содержит информацию о исключениях, возникших в процессе обмена.

Общий пример того как выглядит сообщение данного протокола приведен ниже.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
    <soap:Fault>
      ...
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

1.6.2 Архитектура приложения

Интерфейс приложения должен в себе совмещать как пользовательский вариант, так и специальный консольный, предназначенный для узкого круга пользователей. Чтобы удовлетворить данное требование и при этом не усложнять процесс разработки путем добавления специальной отдельной версии,

было принято решение использовать шаблон проектирования MVP. Основным преимуществом в использовании данного паттерна является его модульность. При такой организации пользовательский интерфейс можно спокойно отделить от основной логики приложения и заменить на другой. Именно поэтому данный шаблон проектирования и будет использован в разработке приложения.

Использование паттерна MVP, приносит в разработку приложения жесткую структурированность. Как видно из названия, приложение представляет из себя три части:

- модель (Model), она выполняет всю бизнес логику приложения, а именно формирования и отправку сообщений. Модель не имеет никаких связей с пользовательским интерфейсом;

- представление (View) — это непосредственно сам пользовательский интерфейс;

- представитель (Presenter) отслеживает события в представлении, обрабатывает их и вызывает соответствующие операции в модели. Играет роль связующего звена между пользовательским интерфейсом и бизнес-логикой.

MVP призван упростить разработку кода, путем распределения обязанностей между членами команды. С применением данного паттерна приложение становится легче тестировать и поддерживать, так отдельные модули проще изолировать от остальной части программы. Упрощенная схема работы MVP представлена на рисунке 5.

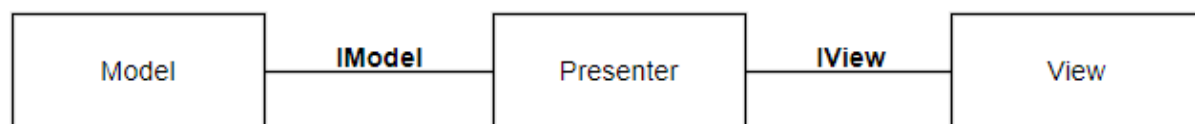


Рисунок 5 - Схема паттерна MVP

Приведенное на рисунке 5 программное обеспечение позволяет формировать сообщения разных форматов, взаимодействовать с очередью IBM

MQ, отправлять сообщения, просматривать отправленные и полученные сообщения.

Исходя из функциональных требований и описания паттерна, составлена общая диаграмма классов (рисунок 6). На её основе будет проведена разработка отдельных компонентов приложения.

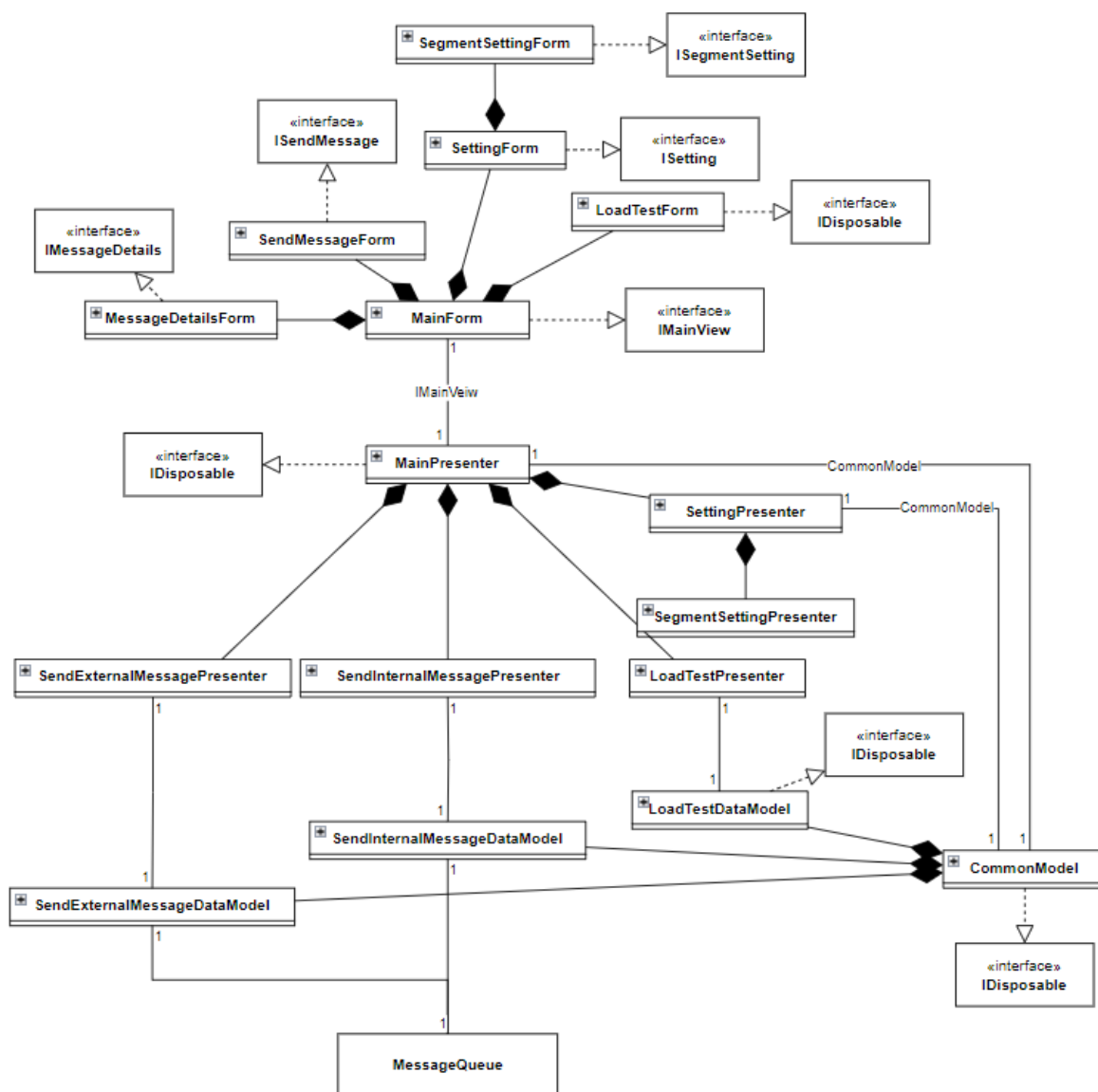


Рисунок 6 - Общая диаграмма классов сервиса

На рисунке 7 приведена диаграмма развертывания приложения.

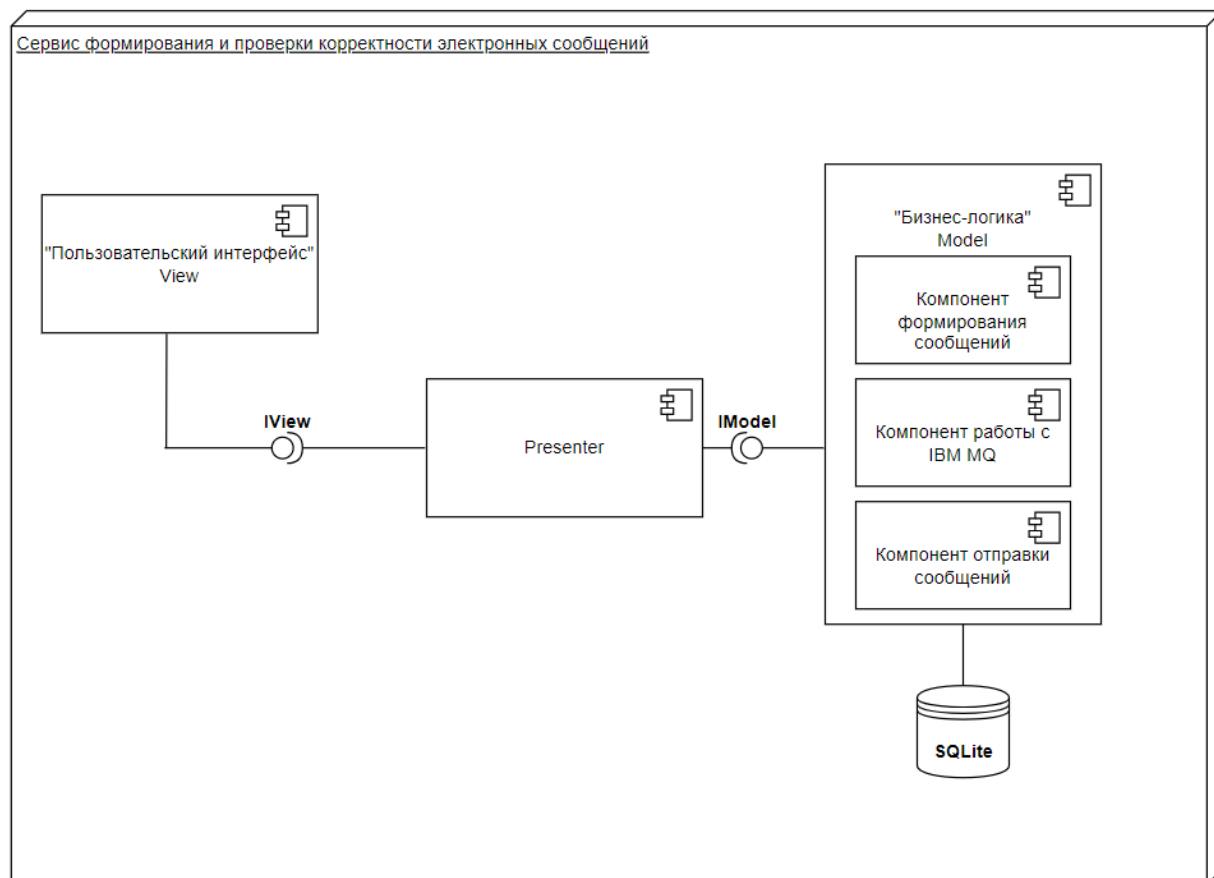


Рисунок 7 – UML диаграмма развертывания

1.6.3 Проектирования пользовательского интерфейса

При проектировании приложения необходимо особое внимание уделить пользовательскому интерфейсу. Каждый элемент интерфейса представляет собой отдельную форму. Схема представления форм View представлена на рисунке Рисунок 8.

Ниже приведено описание форм, их функциональные возможности:

- MainForm – основная форма приложения, из неё пользователь получает доступ ко всему функционалу;
- MessageDetailsForm – форма отвечающая за отображение информации о сообщении, его наименование, идентификатор, а так же список вложенных файлов;

– SendMessageForm – форма формирования и отправки сообщения, в ней пользователь может задать наименование сообщения, прикрепить вложения, установить подпись и выбрать формат сообщения;

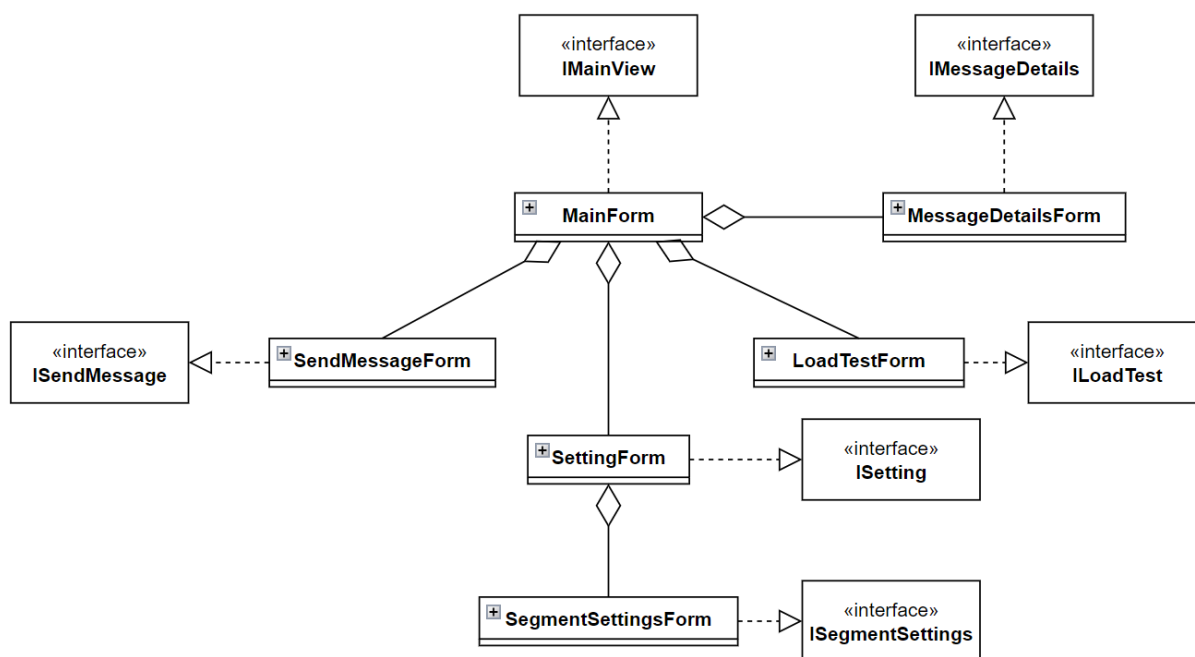


Рисунок 8 - UML диаграмма классов View

– SettingForm – форма отвечающая за отображение настроек приложения, таких как: настройка сегментов отправки и получения, настройка криптографической библиотеки и общие настройки приложения;

– SegmentSettingsForm – форма вызываемая из SettingForm. В ней пользователь может задать параметры для сегмента отправки и получения сообщений.

– LoadTestForm – форма содержащая в себе функционал для нагрузочного тестирования.

1.6.4 Проектирование классов представителя

Так как ни у модели, ни у представления нет связи с компонентами друг друга, то представитель является связующим звеном между ними. Все команды от пользовательского интерфейса сначала обрабатываются в специальном наборе классов, который и решает, какие функции будут вызваны для обработки запросов. Внутри представитель должен провести работу с данными, настроить модель на конкретный запрос пользователя и передать управление дальше. Затем бизнес логика выполняет свои действия с данными и отправляет их назад. Представитель отслеживает активность модели и по её завершении готовит ответ для представления.

В данной работе практически у каждого класса в пользовательском интерфейсе есть свой представитель. Это необходимо для масштабирования реализации с целью распределить функционал по конкретным структурным блокам. Существует также и главный представитель, который собирает все данные во едино и руководит работой остальных.

Диаграмма классов представления изображена на рисунке Рисунок 9.

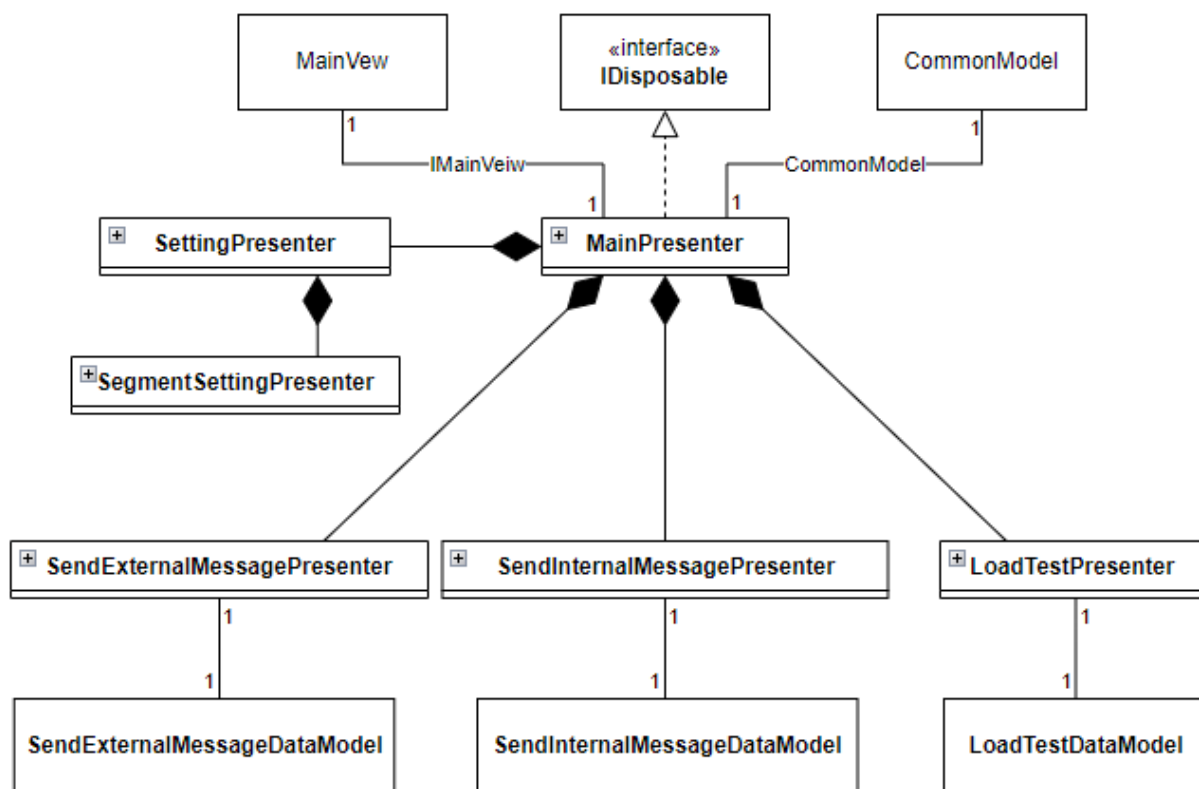


Рисунок 9 - UML диаграмма классов Presenter

Класс MainPresenter – основной представитель главной формы. Он подписывается на все события MainView и отслеживает их. Данный представитель в зависимости от действий пользователя способен передавать управление другим представителям. Имеет связь с общей моделью приложения, и также отслеживает её события.

SettingPresenter – представитель общих настроек приложения. Содержит в себе прочие настройки для которых нужно отдельное окно. По своей сути похож на MainPresenter за тем исключением, что связан со своим представлением, а также имеет ссылки на объекты представителей конкретных настроек.

SegmentSettingsPresenter – представитель для детальных настроек сегментов. С его помощью в модель добавляется информация о новых узлах сети куда необходимо отправлять или откуда нужно получать сообщения. Содержит служебные поля для записи настроек, а также необходимый набор операции,

таких как добавление, удаление, редактирование и заполнение полей в пользовательском интерфейсе.

SendExternalMessagePresenter - представитель для отправки сообщений между разными предприятиями. С помощью этого представителя происходит передача данных из пользовательского интерфейса и подготовка модели для выполнения операции отправки сообщения. Содержит множество прослушивателей событий от представления и бизнес-логики.

SendInternalMessagePresenter - представитель для отправки сообщений в рамках одного предприятия, например для отправки сообщений между разными отделами или структурными подразделениями.

LoadTestPresenter – представитель для управления нагрузочным тестированием. Так же, как и остальные представители данной группы классов, он является промежуточным звеном между моделью и представлением. Имеет в своем наличии методы для регистрации обработчиков событий, а также ряд вспомогательных. К вспомогательным методам относятся функции блокировки и разблокировки пользовательского интерфейса, настройки параметров отправляемых сообщений.

1.6.5 Проектирования бизнес логики

При описании представителей было упомянуто о связанных с ними моделях. Они представляют из себя всю бизнес логику приложения, а именно:

- подготовка сообщений для отправки;
- вызов криптографических функций;
- отправка сообщений в очереди;
- вызов операций нагрузочного тестирования.

Существует как основная модель, так и вспомогательные. Главная предоставляет доступ к общим данным приложения, выполняет инициализацию криптографических библиотек, а также управляет хранилищем сообщений.

Вспомогательные выполняют конкретные функции в зависимости от того к какому представителю они были привязаны, такие как:

- формирование сообщений;
- связь с MQ;
- отправка в MQ;
- запуск нагрузочных тестов.

В данном проекте нагрузочное тестирование доступно только в рамках отдельной сборки по требованию разработчиков, в пользовательском варианте данный функционал недоступен.

Схема классов модели изображена на рисунке Рисунок 10.

CommonModel – данная модель выполняет операции по сохранению настроек приложения, инициализации всех компонентов, подготовки хранилища, создания объекта для работы с очередями сообщений.

SendExternalMessageDataModel – отвечает за подготовку и отставку сообщений между разными предприятиями.

SendInternalMessageDataModel – отвечает за подготовку и отставку сообщений в рамках одного предприятия.

LoadTestDataModel – инициализирует и запускает операции нагрузочного тестирования.

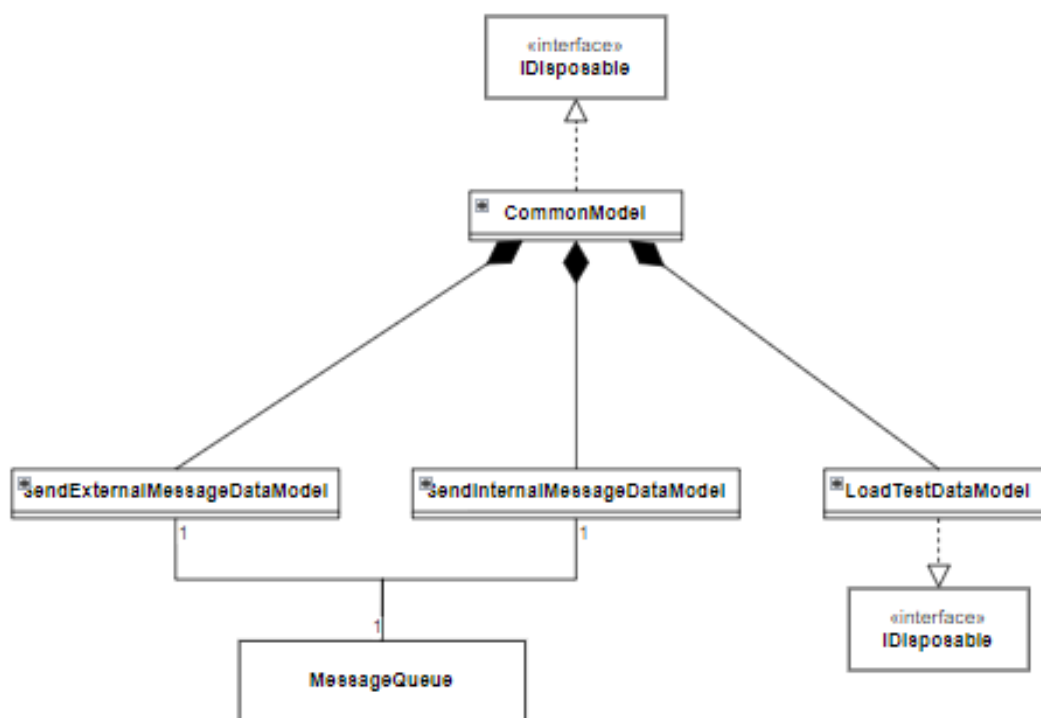


Рисунок 10 - UML диаграмма классов Model

1.7 Выбор инструментов разработки.

1.7.1 Выбор языка программирования.

Существует множество языков программирования, который могут подойти для данной задачи. Рассмотрим самые популярные из них:

– язык Java. Это строго типизированный объектно-ориентированный язык программирования, предназначенный для широкого круга задач. Обладает множеством библиотек для создания приятного пользовательского интерфейса. Однако обладает рядом недостатков: отсутствие поддержки низкоуровневых элементов, таких как указатели; отсутствие делегатов, что, следовательно, усложняет реализацию паттерна MVP.

– Python. Язык программирования с динамической типизацией и автоматическим управлением памятью, поддерживает множество парадигм, например, процедурная, структурная, объектно-ориентированная и тп. Все это позволяет использовать его для широкого спектра задач, начиная от веб-разработки заканчивая сложными научными вычислениями. Несмотря на

широкий ряд преимуществ, у Python есть один существенный недостаток — скорость работы. В отличие от компилируемых языков, Python является интерпретируемым, что сильно сказывается на производительности.

– язык программирования C#, читается как си шарп. Существует две версии о происхождении названия данного языка программирования. С одной стороны, символ «#» взят из музыкальной теории где данный знак обозначает диез, то есть повышение звука на пол тона. Согласно другой версии, C# это улучшенная версия языка программирования C++, где символ решетки — это просто четыре символа плюс, таким образом получается, что рассматриваемый язык является следующим закономерным этапом в развитии языка C. Насколько данное утверждение верно вопрос спорный и требует отдельной темы для размышления. Стоит сказать, что данный язык очень похож на Java, но по своему опыту взаимодействия с обоими языками могу сказать, что C# является его более аккуратной версией. Итак, C# это строго типизированный объектно-ориентированный язык, который является частью платформы .Net, с поддержкой модели асинхронной разработки, а также широким спектром стандартных библиотек. При разработке на данном языке код получается кроссплатформенным, что очень важно в современном мире. По мимо этого впечатляют и его функциональные возможности, а именно возможность использовать лямбда функции, анонимные и асинхронные методы, конструкции LINQ [4][5]. Отдельным достоинством можно выделить наличие делегатов, что позволяет в полной мере использовать событийно-ориентированную модель разработки. C# имеет поддержку технологии Entity Framework, что позволяет создавать объекты баз данных прямо в коде и таким образом взаимодействовать с ними.

В результате анализа существующих языков, было принято решение вести разработку на языке C#, так как он обладает интуитивно понятными средствами для создания удобного пользовательского интерфейса, а также широкими функциональными возможностями.

1.7.2 Выбор среды для разработки

В качестве среды разработки было решено выбрать Microsoft Visual Studio, так как данная среда предоставляет удобный инструментарий для верстки интерфейса Windows Forms, позволяя интуитивно создавать и настраивать пользовательские интерфейсы, размещать элементы управления на форме и задавать им свойства. Это существенно ускоряет процесс разработки и помогает создать эффективный и привлекательный пользовательский интерфейс. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. В данной среде разработки есть удобная возможность подключать не только сторонние библиотеки, но и целые модули. С помощью диспетчера пакетов NuGet разработчик может получить доступ к огромной библиотеке расширений. Все что требуется – это выбрать нужный пакет и нажать на кнопку установки, все остальные действия выполнит студия. Данная возможность поставляется сразу со средой разработки. Также у Visual Studio имеется очень удобный и приятный установщик, который позволяет установить не только актуальные средства разработки, но и предыдущие. К ним относятся не только различные SDK, но и различные средства компиляции, а также поддержка других языков. Помимо установки пакетов или дополнительных средств разработки у Visual Studio есть возможность устанавливать расширения, которые направлены на внедрение вспомогательных функций. Это такие инструменты как автоматическое соблюдение стандартов написания кода, когда на предприятии есть свои правила по оформлению внешнего вида классов, объектов и всего проекта в целом. Дополнительные интерфейсы для более удобного взаимодействия с базами данных, интеграция различных средств для тестирования и автоматической генерации документации для написанной программы. Таким образом можно достаточно гибко настроить среду разработки под конкретные нужды. Все эти факторы делают Microsoft Visual Studio предпочтительным выбором для разработки приложений на платформе Windows.

1.7.3 Выбор Api для создания пользовательского интерфейса

В качестве графической платформы приложения была выбрана платформа Windows Forms, которая предоставляет мощный и гибкий инструмент для создания пользовательского интерфейса. Windows Forms обладает рядом преимуществ, делающих его удобным выбором для разработки приложений. Windows Forms обеспечивает простоту и интуитивность разработки интерфейса благодаря использованию визуального дизайнера форм, который позволяет размещать и настраивать элементы управления на форме в удобном графическом режиме. Это упрощает процесс создания пользовательских интерфейсов и сокращает время разработки. Также Windows Forms предлагает широкий набор предопределенных элементов управления, таких как кнопки, текстовые поля, списки, таблицы и другие, что обеспечивает разработчикам гибкость и возможность создания разнообразных пользовательских интерфейсов. Кроме того, Windows Forms обладает отличной поддержкой событий и обработки пользовательского ввода, что позволяет создавать отзывчивые и интерактивные приложения.

1.7.4 СУБД

В качестве системы управления базами данных было принято решение использовать SQLite, так как он идеально подходит под технические требования. Так как система является встраиваемой, то нет необходимости разворачивать под нее отдельный сервер, то есть по сути, она является частью программы. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором выполняется программа. Так же система имеет поддержку типа данных BLOB (Binary Large Object), что позволяет спокойно хранить текстовые документы размером до 1 гигабайта.

1.8 Разработка приложения

1.8.1 Разработка представителей

На рисунке 11 представлены основные поля и методы класса MainPresenter.

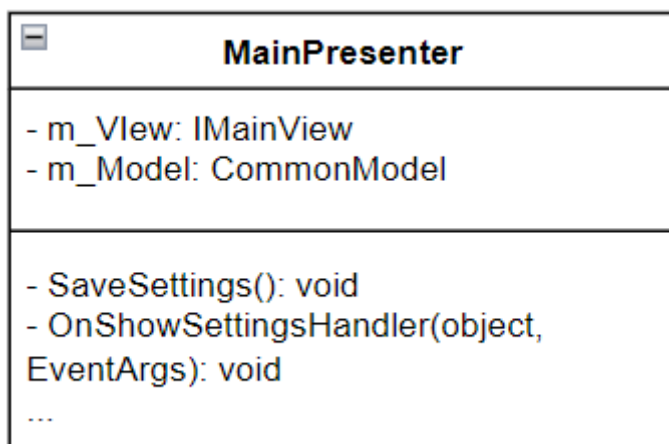


Рисунок 11 - Поля и методы класса MainPresenter

Основные поля класса MainPresenter:

- поле `m_View`. Хранит ссылку на объект реализующее представление для данного представителя;
- поле `m_Model`. Содержит ссылку на объект реализующее модель данных для данного представителя.

Через данные поля у представителя есть связь как с моделью, так и с представлением. При этом ни модель и представление друг о друге не знают. Весь паттерн основан на событийной модели поведения, например, когда срабатывает какое-то событие в модели, то на это реагирует прослушиватель в представителе. Прослушиватель – это специальный метод, который подписан на определённое событие и в момент его срабатывания перехватывает управление программы.

Для регистрации всех обработчиков были созданы методы `SubscribeViewOnEvents` и `SubscribeModelOnEvents`. Пример такого метода можно посмотреть ниже.

```
private void SubscribeViewOnEvents()  
{
```

```

        m_View.SettingsClicked +=
OnShowSettingsHandler;
        m_View.CurrentSegmentSelectedIndexChanged +=
OnCurrentSegmentChangedHandler;
        m_View.NewMessageButtonClicked +=
OnSendMessageHandler;
        m_View.RepeatSendClicked +=
OnRepeatSendMessageHandler;
        m_View.GetMessageButtonClicked +=
OnReceiveMessageHandler;
        m_View.ChangeAndReSendClick +=
OnChangeAndReSendMessageHandler;
        m_View.OutgoingOpenXmlClicked +=
OnShowOutgoingDocumentHandler;
        m_View.IncomingOpenXmlClicked +=
OnShowIncomingDocumentHandler;
        ...
    }

```

Можно пронаблюдать как из объекта представления вызываются события и на них подписываются обработчики. Каждый обработчик выполняет свою определённую задачу по вызову методов как модели, так и представления. Сам по себе представитель не содержит в себе сложной логики, ведь это прерогатива модели. К примеру, рассмотрим обработчик события нажатия на кнопку настроек, код которого представлен ниже.

```

private void OnShowSettingsHandler(object sender, EventArgs e)
{
    try
    {
        using (var SettingsView = new
SettingsPresenter(m_Model))
        {
            SettingsView.ShowDialog();
        }

        // Обновление данных после переопределения настроек.
        SaveSettings();
    }
    catch (Exception ex)
    {
        Log.Trace(ex);
        m_View?.ShowErrorMessage(ex.Message, "При применении
настроек произошла ошибка");
    }
}

```

Сначала создается представитель для окна настроек из которого в свою очередь вызывают метод создания представления. После того как пользователь

обновит настройки и закроет окно, будет вызван метод сохранения новых данных. Также можно наблюдать, что весь код в прослушивателе заключен в блоки обработки исключения, это сделано для того, чтобы программа не закрылась после возникновения исключительной ситуации, вместо этого будет выведено диалоговое окно с описанием ошибки. При этом само приложение продолжит функционировать.

На рисунке 12 представлены основные поля и методы класса SettingsPresenter.

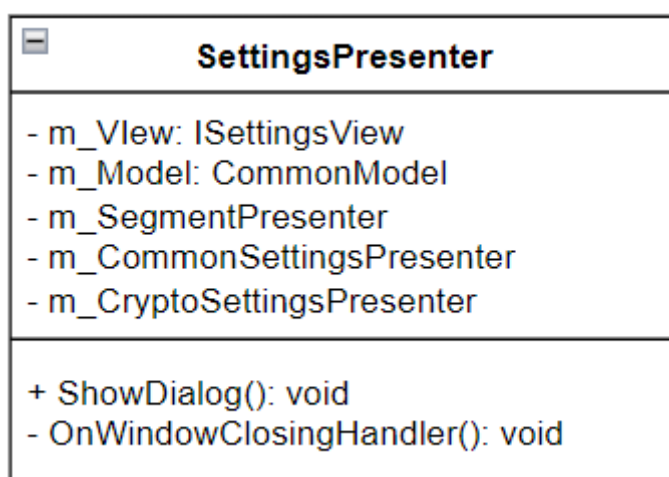


Рисунок 12 - Поля и методы класса SettingsPresenter

Описание полей класса представлено ниже:

- поле `m_View`. Хранит объект представления для данного представителя;
- поле `m_Model`. Ссылка на общую модель приложения;
- поле `m_SegmentsPresenter`. Содержит объект представителя для окна настроек сегментов отправки и получения;
- поле `m_CommonSettingsPresenter`. Объект представителя для окна базовых настроек;
- поле `m_CryptoSettingsPresenter`. Представитель для настроек криптографической подсистемы.

Помимо ссылок на модель и представление, у данного класса есть ссылки и на других представителей. Они являются частью окна настроек и существуют

исключительно в его рамках. Сам представитель служит больше как контейнер для дочерних окон и функционально обделен. Он имеет всего два метода, один из них отвечает за отображение представления, а другой является обработчиком для события закрытия окна.

На рисунке 13 представлены основные поля и методы класса SegmentSettingsPresenter.

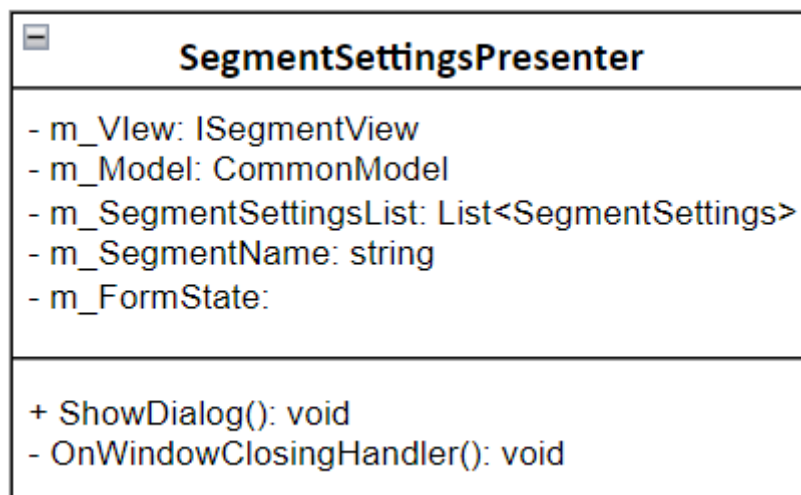


Рисунок 13 - Поля и методы класса SegmentSettingsPresenter

В данном классе определены следующие поля:

- поле `m_View`. Содержит ссылку на объект представления;
- поле `m_Model`. Хранит ссылку на объект модели;
- поле `m_SegmentSettingsList`. Список сегментов доступных для редактирования;
- поле `m_SegmentName`. Хранит имя сегмента, над которым выполняется операция;
- поле `m_FormState`. тип выполняемой операции (изменение, добавление, копирование).

Как и во многих представителях имеет ссылки на соответствующие модель и представление, а также ряд прослушивателей для событий пользовательского интерфейса. Основное взаимодействие происходит через поле `SegmentSettingsList`. Существует целый ряд методов, обеспечивающих разный

функционал работы с данным полем в зависимости от требуемой операции. Рассмотрим, например, метод изменения настроек уже существующего сегмента **ChangeSegment**.

```

public void ChangeSegment()
{
    try
    {
        var segmentSettings = m_SegmentSettingsList.Find(
            settings => settings.SegmentName == m_SegmentName);

        var SegmentCode = m_View.SegmentCode;

        segmentSettings.SegmentName =
m_View.SegmentName;
        segmentSettings.SegmentCode = SegmentCode;
        segmentSettings.ManagerName =
m_View.ManagerName;
        segmentSettings.OutgoingQueueName =
m_View.OutgoingQueueName;
        segmentSettings.IncomingQueueName =
m_View.IncomingQueueName;
        segmentSettings.ChanelName =
m_View.ChanelName;
        segmentSettings.IP = m_View.IP;
        segmentSettings.Port = m_View.Port;
        segmentSettings.AddressToSend =
GetFullAddressFromCode(SegmentCode, m_View.AddressToSend);
        segmentSettings.AddressToAnswer =
GetFullAddressFromCode(SegmentCode, m_View.AddressToAnswer);
        segmentSettings.HashAlgorithmOID =
m_View.HashAlgorithmOID;
        segmentSettings.CheckingAlgorithmOID =
m_View.EdsAlgorithmOID;
        segmentSettings.IncomingQueueDts =
m_View.IncomingQueueDts;

        m_Model.Settings.Save();
    }
    catch (Exception ex)
    {
        Log.Trace(ex);
        m_Model.ViewFactory.ShowWarningMessage(
            ex.ToString(),
            "Операция не выполнена");
    }
}

```

В начале метода мы можем наблюдать, как происходит поиск нужного сегмента в списке, с помощью стандартного метода класса List – Find. После того

как нужный сегмент был найден, происходит обновление полей настроек путем передачи параметров из пользовательского интерфейса. После этого вызывается метод сохранения настроек в файл.

Можно так же наблюдать использование конструкции try-catch для обработки возникших исключений. Все исключительные ситуации отлавливаются, записываются в журнал отладки и затем сообщение о возникшей проблеме выводится на экран.

На рисунке 14 представлены основные поля и методы класса SendExternalMessagePresenter.

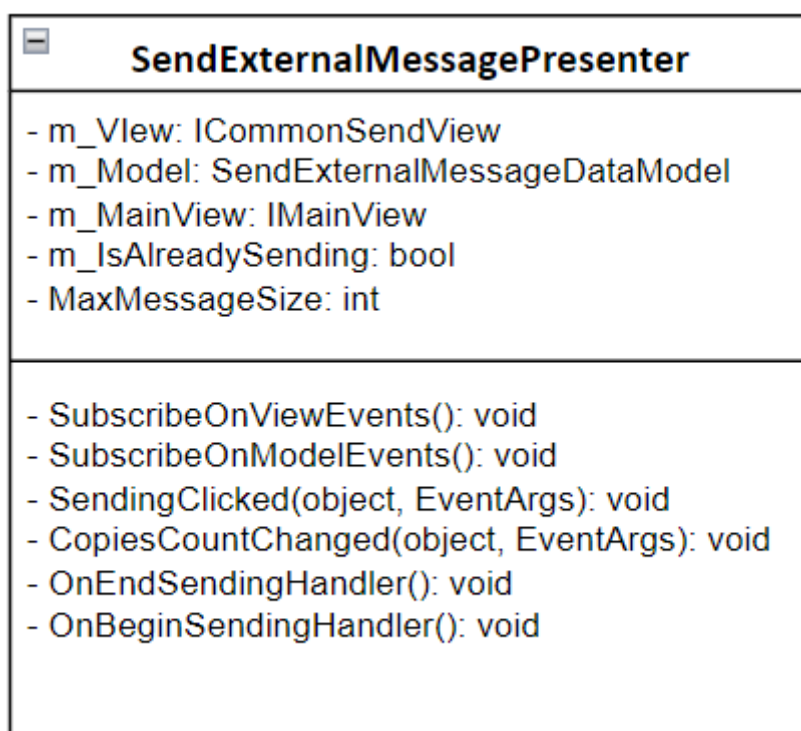


Рисунок 14 - Поля и методы класса SendExternalMessagePresenter

Перечислим основные поля класса:

- поле `m_View`. Хранит объект реализующий представление для данного представителя;
- поле `m_MainView`. Содержит ссылку на родительское окно;
- поле `m_Model`. Хранит объект, содержащий в себе бизнес-логику, связанную с процессом отправки сообщений;

- поле `MaxMessageSize`. Хранит в себе значение максимально допустимого размера сообщения в байтах;
- поле `m_IsAlreadySending`. Обозначает признак того, что в данный момент выполняется отправка сообщения.

Основными методами являются процессы отправки сообщения и изменения параметров отправки. Основу этих функций составляют прослушиватели событий, которые подписаны на определенные события и реагируют на их обновления. Так, например, при изменении количества отправляемых копий срабатывает событие из представления `CopiesCountChanged`. На него подписан одноименный обработчик, в котором в модель передаются соответствующие параметры. Пример кода вы можете наблюдать ниже.

```
private void CopiesCountChanged(object sender, EventArgs e)
{
    // Передача изменений в модель.
    m_Model.PeriodicSend = m_View.IsPeriodSendingChecked();
    m_Model.SendPeriod   = (int)m_View.GetDelaySecondsValue();
    m_Model.ReplayCount  = (int)m_View.GetPeriodCountValue();
}
```

Модель в свою очередь принимает эти параметры и, когда будет нажата кнопка отправки, представитель это увидит и передав нужные данные в модель вызовет процесс отправки из бизнес-логики. Ниже представлена часть обработчика команды отправки сообщения.

```
// Передача всех значений в модель.
m_Model.CertsList           = m_View.CertsList;
m_Model.MsgCount            =
(int)m_View.GetSingleCopiesCount();
m_Model.MsgFormat           = m_View.GetSendType();
m_Model.MsgType             =
m_View.GetMessageStructureType();
m_Model.SendToTtpDirectly   =
m_View.IsSendToTtpQueueDirectly();
m_Model.RandomFormat        = m_View.IsRandomMessages;
m_Model.PeriodicSend        =
m_View.IsPeriodSendingChecked();
m_Model.ReplayCount         =
(int)m_View.GetPeriodicCopiesCount();
m_Model.SendPeriod          =
(int)m_View.GetDelaySecondsValue();
m_Model.UseQueueLimit       = m_View.IsLimited();
m_Model.QueueLimit          = (int)m_View.GetQueueLimit();
```

```

        m_Model.CleanSendingTtpQueue      =
m_View.IsCleanTTPOutChecked();
        m_Model.SendingTtpQueueName       = m_View.GetTTPOutQName();
        m_Model.AttachedFiles              = m_View.FilesHelper;
        m_Model.MaximumMessageSize         = MaxMessageSize;

// Блокировка представления.
LockUi();

// Подготовка операции отправки сообщений и запуск в фоне.
m_Model.RunSending();

```

Здесь отчётливо видна реализация паттерна MVP, когда через методы представителя данные переходят от пользовательского интерфейса в модель. Процесс отправки может занимать определённое количество времени. В модели под операцию отправки выделяется отдельный поток. Чтобы не произошло ситуации, когда пользователь несколько раз нажмет на кнопку отправки и создаст огромное количество потоков, вызвав тем самым непредвиденное падение программы, была написана блокировка пользовательского интерфейса на время отправки. Более того, представление переходит в режим отправки:

- появляется текст, сигнализирующий о выполнении операции;
- текст кнопки меняется с «Отправить» на «Прервать отставку».

Таким образом пользователь может в любой момент прервать процедуру отправки сообщений.

На рисунке 14 представлены основные поля и методы класса SendExternalMessagePresenter.

Данный представитель очень похож на предыдущий. У них очень схожая структура, однако есть ряд отличий. Заключаются они в том, что используют разные представления и модели. Именно в пользовательском интерфейсе и заключается основное различие, а именно в генерируемых событиях. Но несмотря на это, структурно классы практически одинаковы.

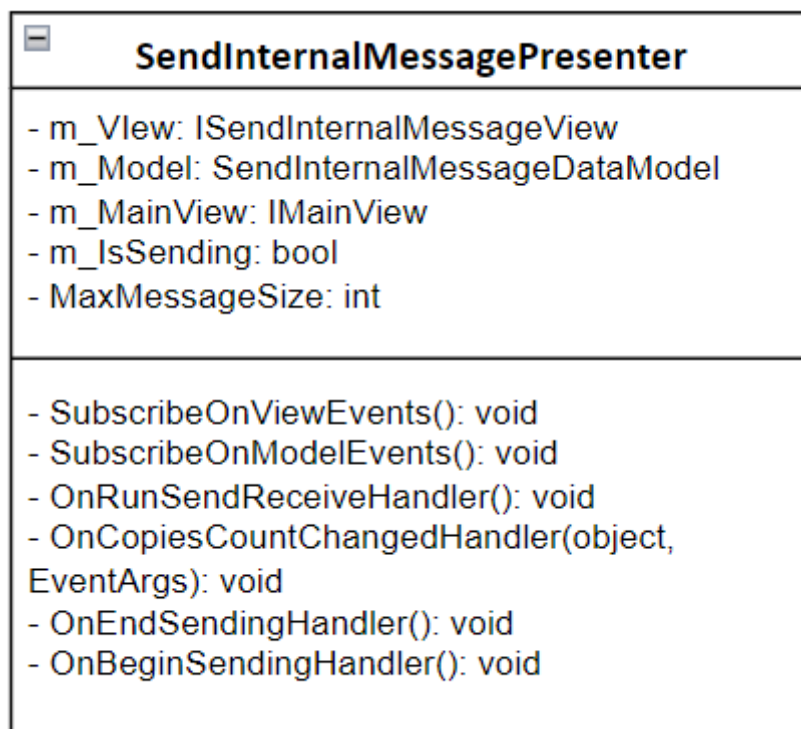


Рисунок 15 - Поля и методы класса SendInternalMessagePresenter

На рисунке 16 представлены основные поля и методы класса LoadTestPresenter.

К полям класса относятся:

- поле m_View. Хранит ссылку на объект пользовательского интерфейса для данного представителя;
- поле m_Model. Содержит ссылку на модель с реализацией бизнес-логики;
- поле m_IsTestWorking. Флаг обозначающий запуск теста;
- поле m_Mode. Отображает текущий режим работы теста.

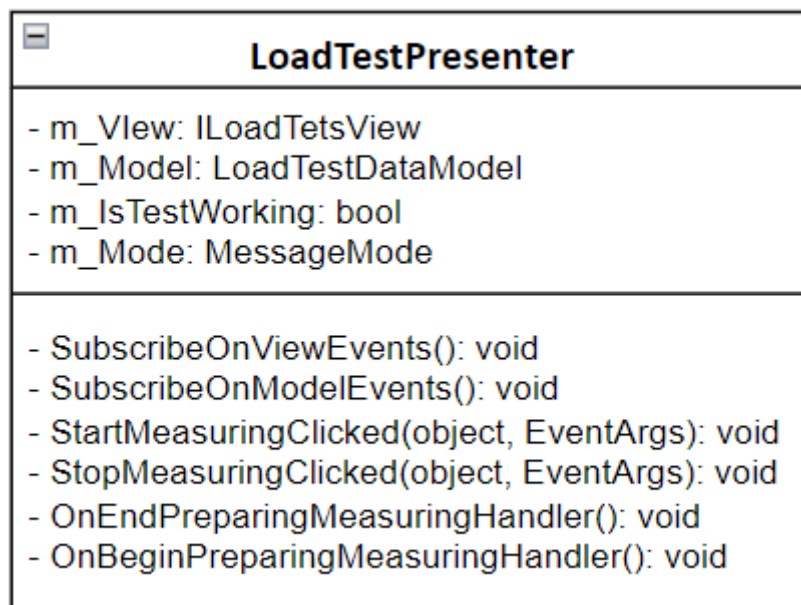


Рисунок 16 - Поля и методы класса LoadTestPresenter

Замеры проводимы в модели связаны в основном со скоростью отправки сообщений. На их основе разработчик может определить слабые участки в сети или неисправность в работе менеджера очередей. Информацию о проведении нагрузочных тестов представитель получает от модели и затем передает её в пользовательский интерфейс. Когда из пользовательского интерфейс приходит команда о выгрузке отчета о тестировании, представитель посылает запрос к модели. Метод отвечающий за этот функционал приведен ниже.

```
private void OnUnloadReportClicked(object sender, EventArgs e)
{
    try
    {
        var UserChoice =
m_Model.ApplicationModel.ViewFactory.ShowSaveFileDialog(
            out string path,
            "txt",

$"LoadReport_{DateTime.Now.ToString("g").Replace(':', '_').Replace(' ', '_')}");

        if (UserChoice)
        {
            m_Model.RememberMeasuring();
            File.WriteAllText(path, m_Model.UnloadData ?? "");
        }
    }
}
```

```

    }
    catch (Exception ex)
    {
        Log.Trace(ex);

        m_Model.ApplicationModel.ViewFactory.ShowWarningMessage(
            $"Во время выгрузки отчета о нагрузочном
            тестировании произошла ошибка. {ex.Message}",
            "Ошибка");
    }
}

```

Файл отчета будет сохранен в то место, куда укажет пользователь в диалоговом окне вызванное метом ShowSaveFileDialog.

1.8.2 Разработка моделей

На рисунке 17 представлены основные поля и методы класса LoadTestPresenter.

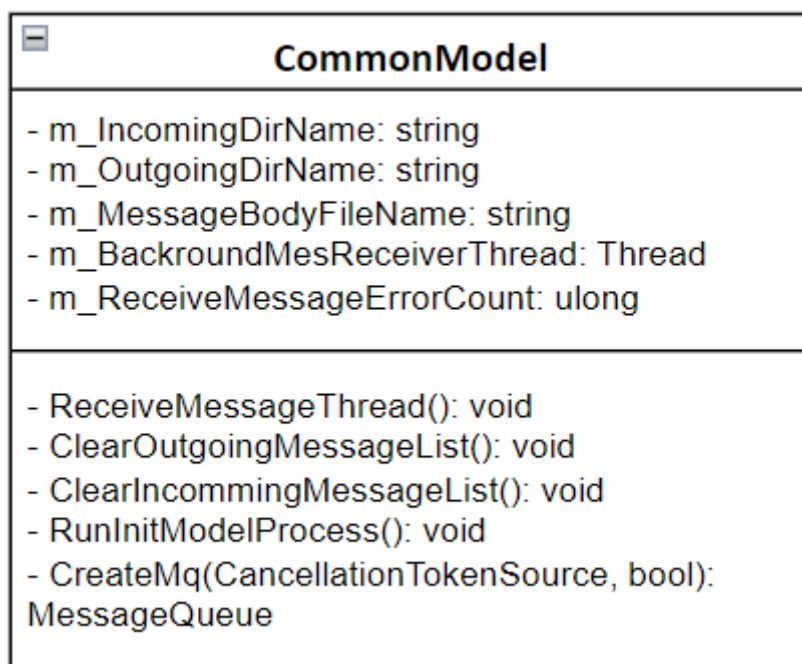


Рисунок 17 - Поля и методы класса CommonModel

К полям данного класса относятся:

- поле `m_IncomingDirName`. Хранит путь к директории для сохранения файлов входящих сообщений;
- поле `m_OutgoingDirName`. Содержит путь к директории для сохранения файлов исходящих сообщений;

- поле `m_MessageBodyFileName`. Хранит имя файла с телом сообщения;
- поле `m_BackgroundMesReceiverThread`. Содержит объект потока для автоматического считывания сообщений их входящей очереди;
- поле `m_ReceiveMessageErrorCount`. Счетчик ошибок при получении сообщений.

Так как данная модель выполняет общие действия по обеспечению функционирования приложения, то её методы в основном связаны с настройками, инициализацией компонентов, а также считывания сообщений из очереди. Для отправки и формирования сообщений используются другие модели. Стоит отметить, что в приложении имеется возможность сохранения сообщения в файл для просмотра его в будущем. Для этого необходимо инициализировать соответствующие директории, а точнее проверить их существования. В случае если приложение было запущено впервые, то каталоги будут созданы автоматически.

Действия по инициализации директорий происходят в конструкторе класса и приведены ниже.

```
// Проверка и создание каталогов для хранения сообщений.
if (!Directory.Exists(IncomingDirName))
    Directory.CreateDirectory(IncomingDirName);

if (!Directory.Exists(OutgoingDirName))
    Directory.CreateDirectory(OutgoingDirName);
```

А также создается и запускается поток для извлечения сообщений из входящей очереди в фоне.

```
m_BackgroundMessageReceiverThread = new
Thread(ReceiveMessageThread);
m_BackgroundMessageReceiverThread.Start();
```

Данная функция носит опциональный характер и может быть отключена в настройках приложения. Модель это отслеживает и реагирует в зависимости от выбора пользователя, а именно пропускает итерацию в цикле проверки очереди.

```
// Ничего не делаем, если автоматическое считывание сообщений
отключено.
if (!Settings.AutoMessageGetting)
    continue;
```

Для того чтобы получать сообщения необходимо иметь ссылку на очередь сообщений. Сделать это можно вызвав метод создания клиента очереди `MessageQueue`.

```
private MessageQueue CreateMq(CancellationTokenSource source,
bool isNeedEventHanding = true)
{
    var Result = new MessageQueue(source, TtpSettings);

    if(isNeedEventHanding)
        Result.NotificationGenerated +=
OnMessageQueueNotificationGeneratedHandler;

    return Result;
}
```

Для получения сообщения из очереди необходимо вызвать из объекта `MessageQueue` метод `GetMessage` передав в качестве параметра настройки подключения к очереди.

На рисунке 18 представлены основные поля и методы класса `SendExternalMessageDataModel`.

SendExternalMessageDataModel
<ul style="list-style-type: none"> - <code>m_SendingThread</code>: <code>Thread</code> - <code>m_SendingThreadStoped</code>: <code>ManualResetEvent</code> - <code>m_SegmentSettingList</code>: <code>List<SegmentSetting></code> - <code>m_SendingQueue</code>: <code>MessageQueue</code>
<ul style="list-style-type: none"> - <code>RunSending()</code>: <code>void</code> - <code>SendingThread()</code>: <code>void</code> - <code>CreateSegmentSettings()</code>: <code>SegmentSettings</code> - <code>PrepareSendingMessage()</code>: <code>void</code> - <code>SetupMq()</code>: <code>MessageQueue</code> - <code>CreateMq(CancellationTokenSource, bool)</code>: <code>MessageQueue</code>

Рисунок 18 - Поля и методы класса `SendExternalMessageDataModel`

К данному классу относятся следующие поля:

– поле `m_SendingThread`. Объект хранящий ссылку на поток для отправки сообщений в фоне;

- поле `m_SendingThreadStoped`. Событие об остановке потока отправки сообщений;
- поле `m_SegmentSettingList`. Содержит список с настройками сегментов для отправки сообщений;
- поле `m_SendingQueue`. Хранит объект для взаимодействия с очередью отправки сообщений.

В этой модели выполняются операции, связанные с подготовкой и отправкой сообщений в соответствующие им очереди. Процесс подготовки заключается в построении тела сообщения и заполнения всех параметров, связанных с его отправкой. Отправки происходит через объект `MessageQueue`, который предварительно нужно настроить. Процесс настройки представлен ниже.

```
private MessageQueue SetupMq ()
{
    var CountOfCopies = Decimal.ToInt32(MsgCount);

    if (null != OnSendingProgress)
        OnSendingProgress.Invoke(MsgCount, 0);

    var SendingSettings = new SendingSettings()
    {
        Count                = CountOfCopies,
        SendType              = MsgFormat,
        MessageStructureType  = MsgType
    };

    var Result = CreateMq(SendingSettings);

    return Result;
}
```

После заполнения структуры с настройками необходимо создать и сам объект для взаимодействия с MQ.

```
private MessageQueue CreateMq(SendingSettings sendingSettings)
{
    var Result = new MessageQueue(
        m_RandomMessages, m_SegmentSettings, sendingSettings,
        ApplicationModel.TtpSettings, CancellationToken);

    Result.OperationStatusChanged +=
        OperationStatusChangedHandler;
```

```

        Result.OperationEnded                +=
OperationEndedHandler;
        Result.OperationInfoChanged          +=
OnOperationInfoChangedHandler;
        Result.NotificationGenerated        +=
OnNotificationGeneratedHandler;

        return Result;
    }

```

Для информирования программы об окончании отправки, а также о текущем статусе задачи, были созданы прослушиватели событий. Все они просто перебрасывают информацию в представитель, а он уже решает, что с ним делать дальше.

На рисунке 19 представлены основные поля и методы класса SendExternalMessageDataModel.

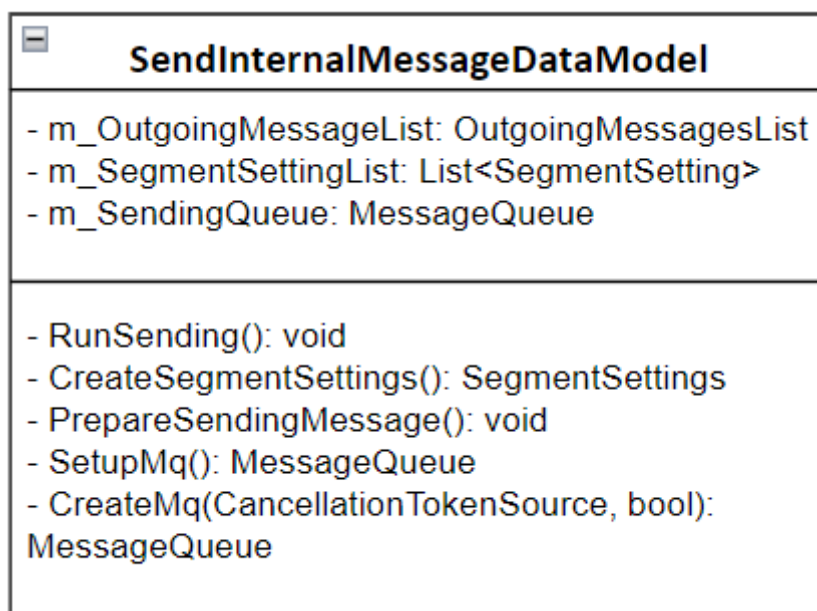


Рисунок 19 - Поля и методы класса SendInternalMessageDataModel

Данный класс очень похож на предыдущий, основные отличия заключаются в процессе формирования сообщения, так как они предназначены для обмена в рамках единого предприятия, то некоторые поля в XML –файле будут отличаться. Соответственно используются другие шаблоны и методы построения сообщений. Основное отличие данного режима отправки заключается в необходимости заранее подготавливать документ для отправки.

Для этого используется специальный компоновщик, который принимает отправляемый файл, подписывает его средствами криптографии и сохраняет в формате CMS.

На рисунке 20 представлены основные поля и методы класса SendExternalMessageDataModel.

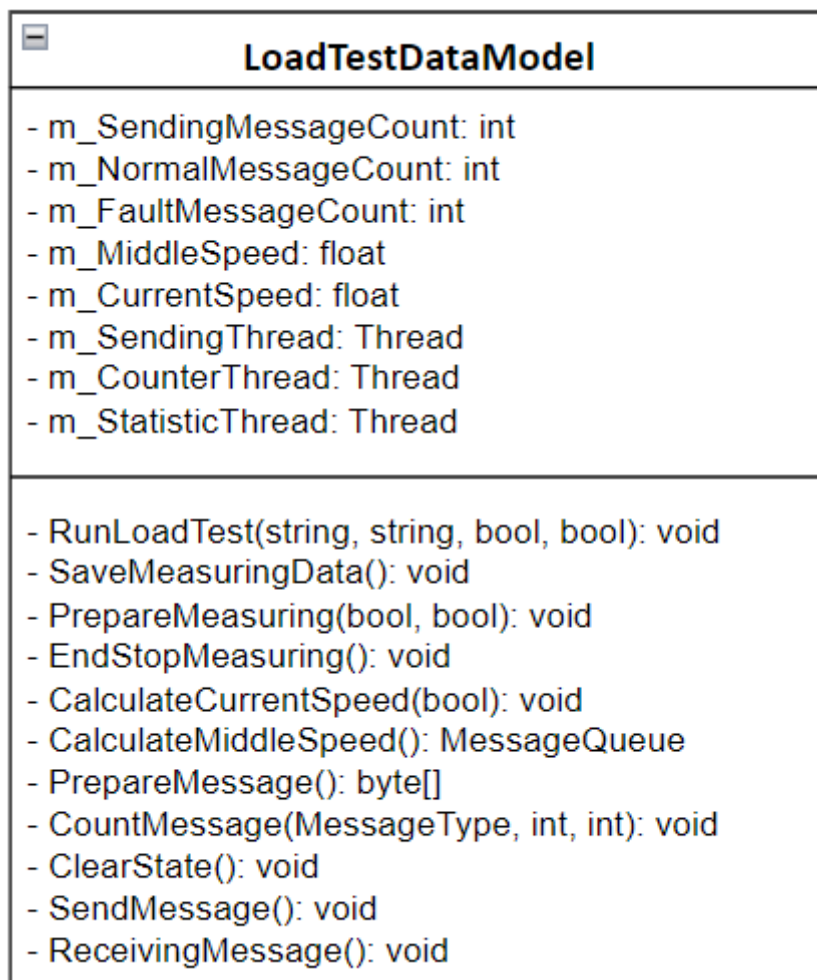


Рисунок 20 - Поля и методы класса LoadTestDataModel

Основные поля класса описаны в следующем порядке:

- поле `m_SendingMessageCount`. Общее количество отправленных сообщений;
- поле `m_NormalMessageCount`. Содержит общее количество успешно полученных сообщений;

- поле `m_FaultMessageCount`. Хранит количество полученных сообщений с ошибкой;
- поле `m_MiddleSpeed`. Содержит рассчитанную среднюю скорость отправки или получения;
- поле `m_CurrentSpeed`. Текущая скорость отправки или получения;
- поле `m_SendingThread`. Хранит объект потока для отправки сообщений;
- поле `m_CounterThread`. Содержит ссылку на поток для подсчета количества сообщений в очереди;
- поле `m_StatisticThread`. Поток для подсчета статистики.

Как можно видеть из описания полей, данный класс выполняет функции запуска нагрузочных тестов и сбора статистики о них. Необходимо это для определения слабых мест в адаптере, а также изучения поведения системы при возможных DDoS атаках. Методы данного класса в основном направлены на запуск различных фоновых операций по мониторингу отправляемых сообщений с последующей отправкой собранной информацией в представитель. При запуске операции нагрузочного тестирования после старта процесса отправки сообщений, запускаются все мониторинговые функции. К ним в основном относятся замеры текущей и средней скорости, подсчёт общего количества отправляемых сообщений, успешно отправленных и количества неудач.

Перед запуском операций, как и в моделях связанных с отправкой сообщений, происходит инициализация очереди MQ. Расчет скоростей отправки сообщений происходит в следующих методах.

```
private void CalculateCurrentSpeed(bool isOver)
{
    if (isOver)
    {
        int normalAmount    = 0;
        int faultAmount      = 0;
        int time             = 0;

        foreach (var value in CurrentSpeedQueue)
        {
            normalAmount    += value.AmountNormalMessages;
            faultAmount      += value.AmountFaultMessages;
            time             += value.Time;
        }
    }
}
```

```

        }

        if (time > 0)
            CurrentSpeed = (float) (normalAmount + faultAmount)
/ time;
        else
            CurrentSpeed = 0;
    }
    else
    {
        if (CurrentTime > 0)
            CurrentSpeed = (float) (NormalMessageCount +
FaultMessageCount) / (float)CurrentTime;
        else
            CurrentSpeed = 0;
    }
}

private void CalculateMiddleSpeed()
{
    if (CurrentTime > 0)
        MiddleSpeed = (float) (NormalMessageCount +
FaultMessageCount) / (float)CurrentTime;
    else
        MiddleSpeed = 0;
}

```

Учет количества сообщений происходит в отдельном методе, который вызывается в цикле перебора полученного списка сообщений.

```

private void CountMessage(MessageType type, ref int
normalCounter, ref int faultCounter)
{
    switch (type)
    {
        case MessageType.Normal:
        {
            NormalMessageCount++;
            normalCounter++;

            break;
        }

        case MessageType.Fault:
        {
            FaultMessageCount++;
            faultCounter++;

            break;
        }

        case MessageType.None:

```

```

        {
            FaultMessageCount++;
            faultCounter++;

            break;
        }

        default:
            break;
    }

}

```

После завершения сбора статистики происходит пересылка всех необходимых данных в представление. Делается это через вызов событий с передачей через них полей с подсчитанными данными.

```

OnFaultMessageCountChanged?.Invoke (FaultMessageCount);
OnSendingMessageCountChanged?.Invoke (SendingMessageCount);
OnMiddleSpeedChanged?.Invoke (MiddleSpeed);
OnCurrentSpeedChanged?.Invoke (CurrentSpeed);
OnLast10MinutsSpeedChanged?.Invoke (Last10Speed);
OnRuntimeValueChanged?.Invoke (CommonStopwatch.Elapsed);

```

1.8.3 Разработка интерфейса

В ходе изучения технологий и планирования архитектуры приложения, был разработан удобный и понятный интерфейс. В данном разделе, приведено его описание. На рисунке 21 представлено главное окно приложения.

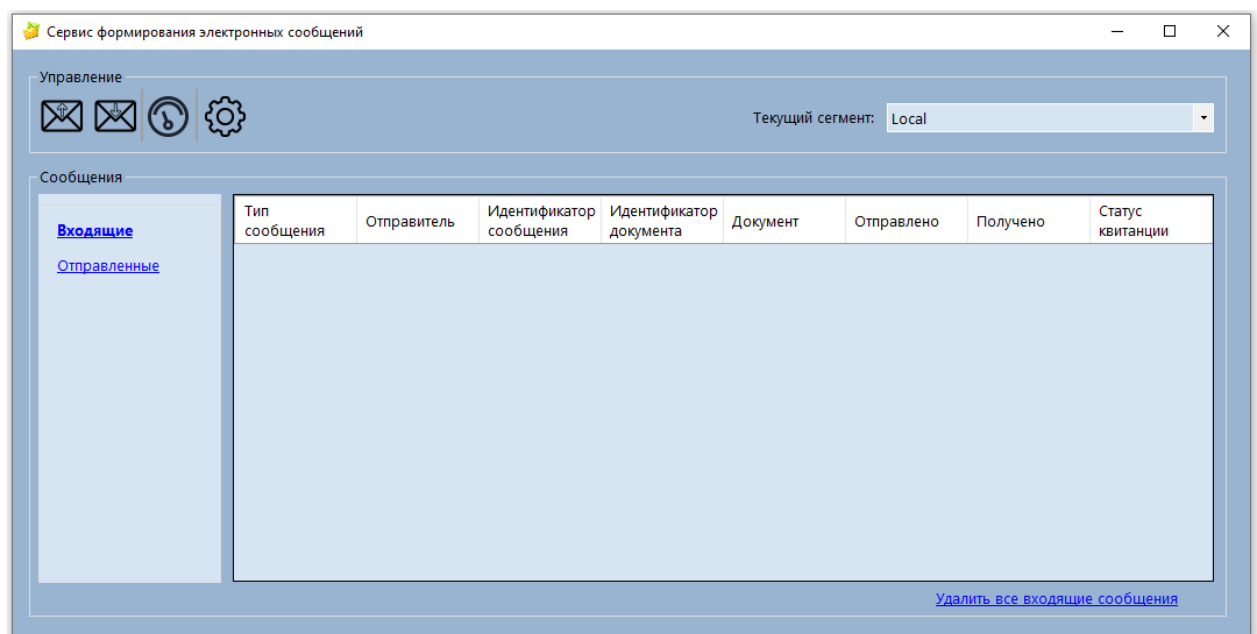


Рисунок 21 - Главное окно приложения

В разделе «Управление» находятся основные навигационные кнопки. С их помощью пользователь может выполнить отправку сообщения, принять сообщение из очереди, выполнить операцию нагрузочного тестирования, а также открыть окно с настройками приложения. В крайней правой части раздела находится окно выбора текущего сегмента. Сегмент – это логический узел, который определяет параметры для отправки сообщений в очередь. Его параметры задаются администратором информационной системы, в которой развернуто приложение.

В разделе «Сообщения» можно увидеть окно с отправленными и полученными сообщениями. Для переключения отображения типа сообщений с входящих на отправленные, необходимо нажать на соответствующий гипертекст.

При нажатии на кнопку имеющую вид шестерни, открывается окно с настройками приложения (рисунок 22).

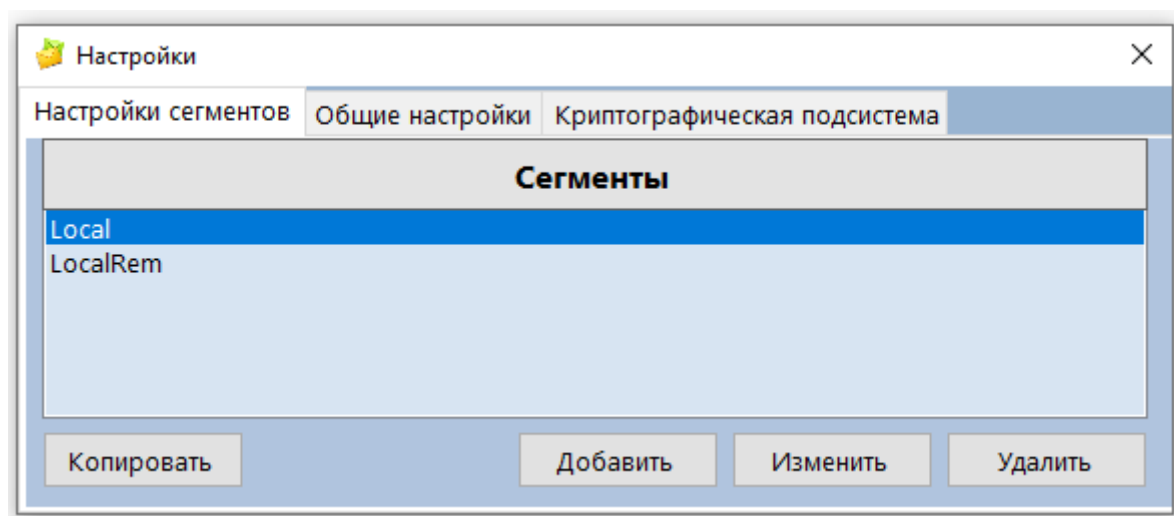


Рисунок 22 - Окно настроек приложение. Настройки сегментов

По умолчанию первыми открываются настройки сегментов, с помощью вкладок в верхней части окна, можно переключаться между разделами настроек.

В разделе с настройками сегментов можно добавить, изменить или удалить сегмент. Также имеется возможность копирования сегмента с последующим его редактированием. По центру окна находится список с созданными сегментами.

При нажатии на кнопку «Изменить» откроется окно редактирования настроек (рисунок 23).

Рисунок 23 - Окно настроек сегмента

Здесь можно изменять или просматривать настройки сегмента. В основном они в себя включают параметры для подключения к менеджеру очередей, а также выбор алгоритма криптографии для подписания сообщения.

При добавлении нового сегмента открывается по сути тоже самое окно, но с пустыми полями для ввода. При этом пустые поля помечаются восклицательным знаком сигнализирующим об обязательности данного параметра к заполнению. До тех пор, пока все обязательные параметры не будут заполнены, кнопка сохранения будет оставаться неактивной (рисунок 24).

Рисунок 24 - Пустое окно настроек сегмента

При удалении сегмента, выведется стандартное диалоговое окно с подтверждением данной операции (рисунок 25).

Рисунок 25 - Диалоговое окно удаления сегмента

Во вкладке «Общие настройки» можно задать автоматическое получение сообщений из очереди. Данный функционал представлен на рисунке Рисунок 26

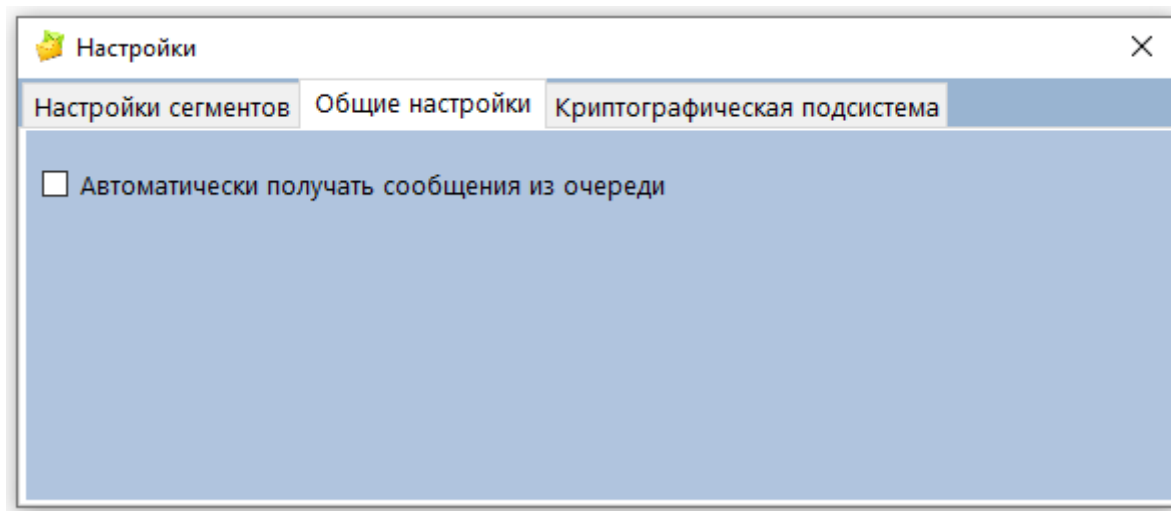


Рисунок 26 - Общие настройки приложения

Во вкладке «Криптографическая подсистема» представлена возможность выбора используемой криптографической библиотеки (рисунок 27).

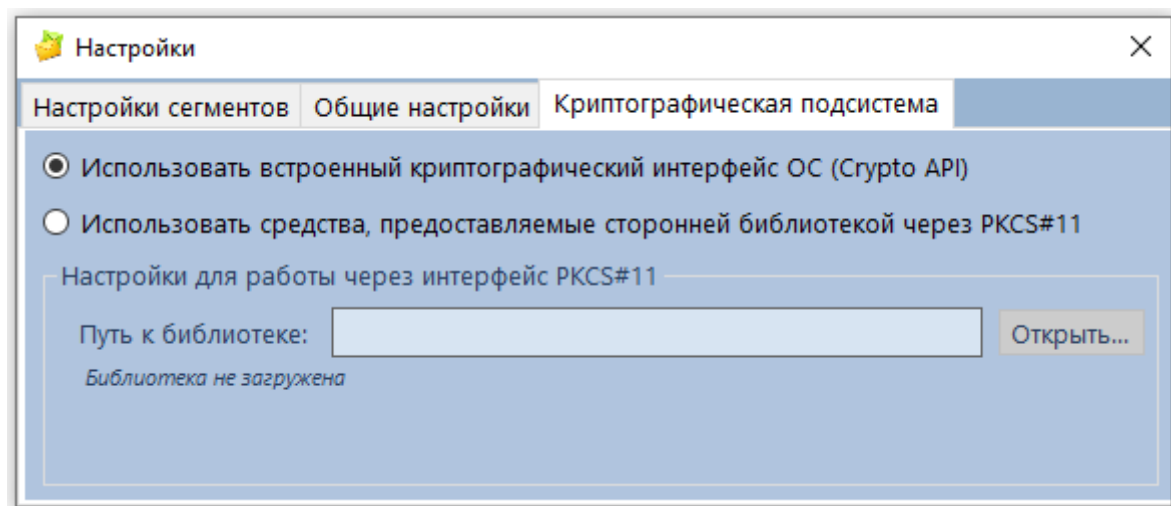


Рисунок 27 - Настройка криптографии

По умолчанию используется стандартная библиотека CryptoApi, которая использует стандартные функции операционной системы Windows. Также пользователь может подключить стороннюю библиотеку написанную с использованием стандарта PKCS#11.

Приложение поддерживает два режима отправки:

- Внешний, для отправки сообщений в сторонние предприятия;
- Внутренний, для отправки сообщений в рамках одного предприятия.

На рисунке 28 представлен выбор режима отправки.

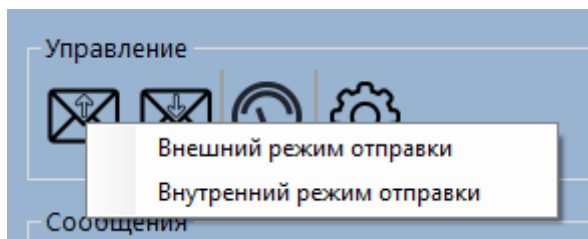


Рисунок 28 - Режимы отправки сообщений

При нажатии на один из пунктов всплывающего меню, отображается соответствующий интерфейс для отправки.

Рассмотрим внешний режим (рисунок 29).

Рисунок 29 - Окно отправки внешнего сообщения

В верхней части окна имеется возможность загрузки сообщения из заранее сконфигурированного шаблона. В разделе «Вложения» пользователь может прикреплять к сообщению файлы и документы. Общий размер вложений не должен превышать максимально допустимый. В разделе «Подпись документа» происходит добавление сертификата для генерации подписи. В случае если активировано использование формата XAdES, то предоставляется возможность привязать атрибутные сертификаты. Они содержат информацию об определенных атрибутах пользователя и несут дополнительную уточняющую информацию. В разделе «Отправка» можно настроить параметры отправки сообщения. Например, задать период отправки, количество сообщений, отправленных за раз, формат сообщения, тип сообщения (корректное и с отсутствием некоторых полей), а также установить лимит сообщений в очереди после превышения которого отправки прервется до тех пор, пока очередь не будет разгружена.

Перейдем к внутреннему режиму отправки (рисунок 30).

Рисунок 30 - Окно отправки внутренних сообщений

В блоке «CMS-сообщение» предоставляется возможность создания или прикрепления файла формата CMS для последующей отправки. При создании файла приложение учитывает его размер и в случае превышения максимально допустимого, блокирует отправку. В блоке «Отправка», как и в предыдущем режиме, происходит редактирование параметров отправки. Кнопка отправки заблокирована до тех пор, пока к сообщению не будет прикреплен файл для отправки.

Если в основном окне нажать на иконку спидометра, то откроется интерфейс запуска нагрузочного тестирования (рисунок 31).

Рисунок 31 - Окно проведения нагрузочного тестирования

Данное окно по своей структуре напоминает интерфейс представленный на рисунке Рисунок 29. Основное отличие заключается в блоках «Отправка» и «Результат». В данном разделе выводятся данные собранные во время проведения нагрузочного тестирования.

Также у данного приложения имеется консольный интерфейс взаимодействия с пользователем. С помощью специального набора команд можно вызывать соответствующие функции. О них можно подробно узнать, вызвав команду /help (рисунок 32).

```
Please enter the command, or use /help for more information:

/help
Отправка сообщения:
-S {путь до файла вложения}
-t {ex - внешнее; in - внутреннее}
-n {количество сообщений в пачке}
-d {период отправки}

Прием сообщения: -R

Просмотр списка сообщений: -V -mt {s - отправленные; r - принятые}

Please enter the command, or use /help for more information:

_
```

Рисунок 32 - Пример вызова команды /help

Текст в фигурных скобках носит справочный характер, вместо них пользователю необходимо ввести конкретное значение.

Команда отправки сообщений состоит из нескольких параметров, пример их использования можно увидеть на рисунке 33.

```
Please enter the command, or use /help for more information:

-S "D:/Messages/data_28.05.2024.xml" -t ex -n 1
Отправка выполнена успешно!
```

Рисунок 33 - Отправка сообщения

После успешной отправки на экране появится информационное сообщений о завершении операции.

Просмотр сообщений выполнен в стиле двух таблиц, отдельно для отправленных и полученных сообщений. Отличаются они набором колонок. На рисунке 34 представлена таблица для отправленных сообщений.

```
Please enter the command, or use /help for more information:
```

```
-V -mt s
```

Id	Отправитель	Получатель	Вложение	Дата отправки
60asf8a792h	Local	LocalR	doc.txt	28.05.2024

Рисунок 34 - Вывод таблицы отправленных сообщений

Чтобы проверить входящую очередь и получить оттуда сообщения, необходимо ввести команду, представленную на рисунке 35.

```
Please enter the command, or use /help for more information:
```

```
-R
```

Получено сообщений: 1

Рисунок 35 - Процесс получения сообщения

Информация о полученных сообщениях будет выведена на экран.

Чтобы посмотреть список полученных сообщений, необходимо ввести команду просмотра таблицы, но с параметром r (received). Пример использования данной команды представлен на рисунке 36.

```
Please enter the command, or use /help for more information:
```

```
-V -mt r
```

Id	Отправитель	Вложение	Дата отправки	Дата получения
g12hhkf1212	LocalR	doc.txt	28.05.2024	28.05.2024

Рисунок 36 - Вывод таблицы полученных сообщений

1.9 Тестирование приложения

Для тестирования приложения мы рассмотрим процессы отправки сообщений для разных режимов.

Перед отправкой сообщения, проверим что список сообщений пуст. Для этого в главном окне переключим представление таблицы с «Входящие» на «Отправленные» (рисунок 37).

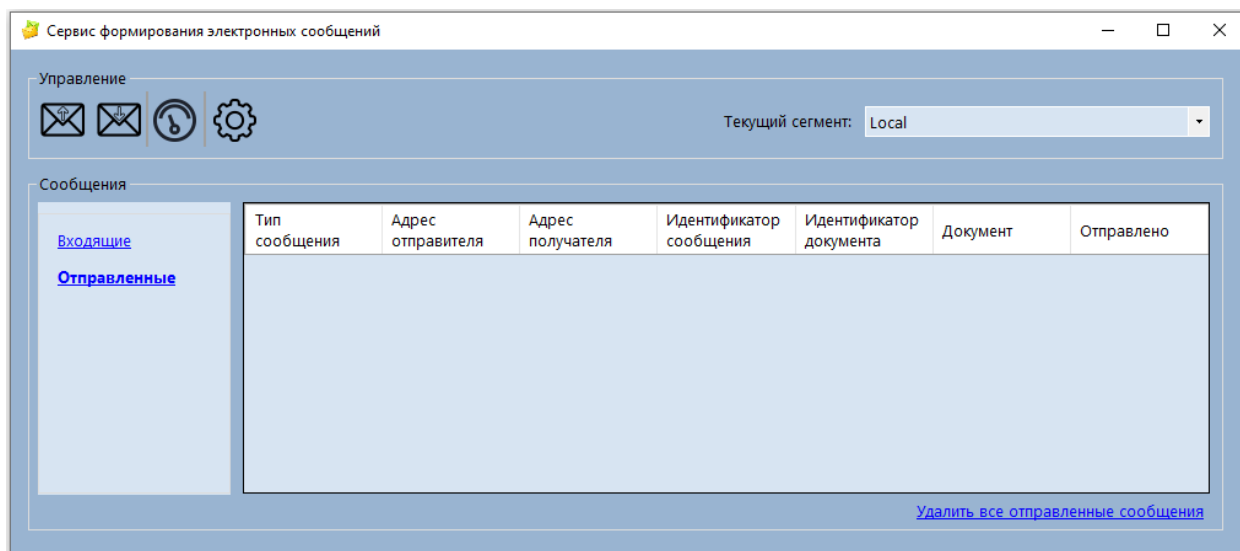


Рисунок 37 - Список отправленных сообщений

Теперь в окне отправки внешнего сообщений добавим к файлу вложение, при этом обратите внимание на показатель доступного места. На рисунке 38 отлично видно, что на данный момент прикрепленных файлов нет и доступно 74,63 мегабайта.

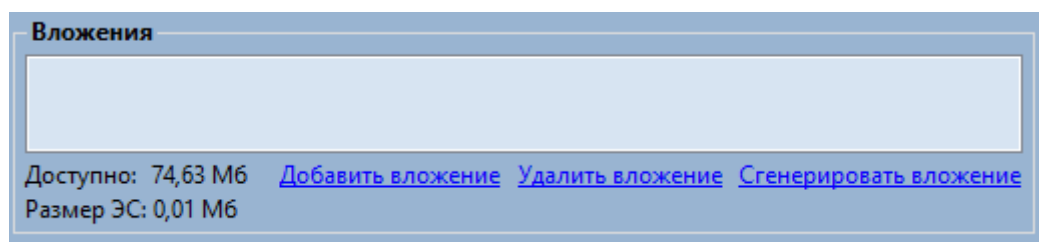


Рисунок 38 - Список вложений

При нажатии на кнопку «Добавить вложение» откроется стандартный проводник Windows с возможностью выбора файла. После прикрепления вложения доступный размер уменьшится (рисунок 37).

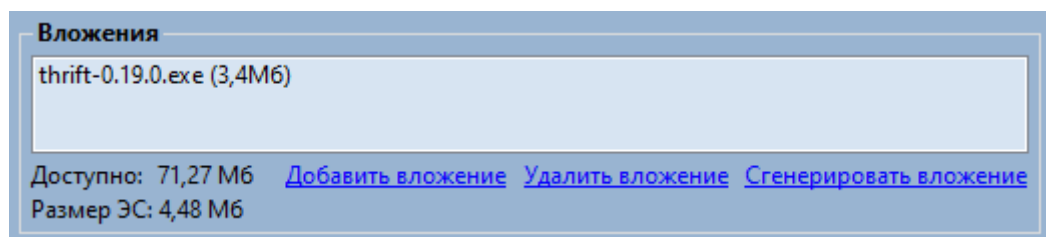


Рисунок 39 - Добавленные к сообщению вложения

Как нетрудно заметить, добавить можно файл абсолютно любого разрешения. Также в приложении имеется возможность сгенерировать файл, заполнив его мусорными данными. Процесс генерации файла представлена на рисунке 40.

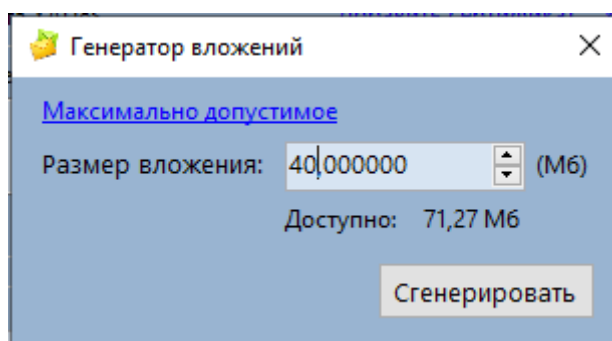


Рисунок 40 - Генерация файла

При выборе варианта «Максимально допустимое» в счетчике будет выставлено значение равное оставшемуся свободному месту.

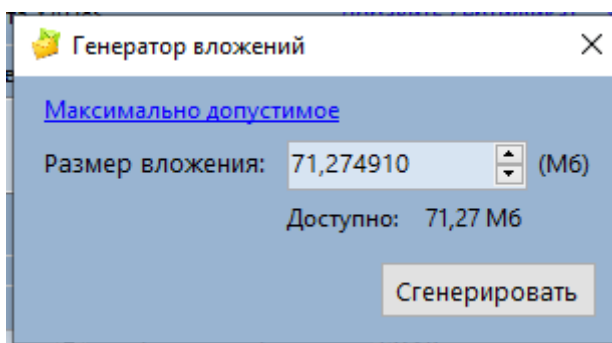


Рисунок 41 - Генерация файла максимально допустимого размера

Сгенерированный файл автоматически прикрепляется к сообщению и ему присевается уникальный идентификатор (рисунок 42).

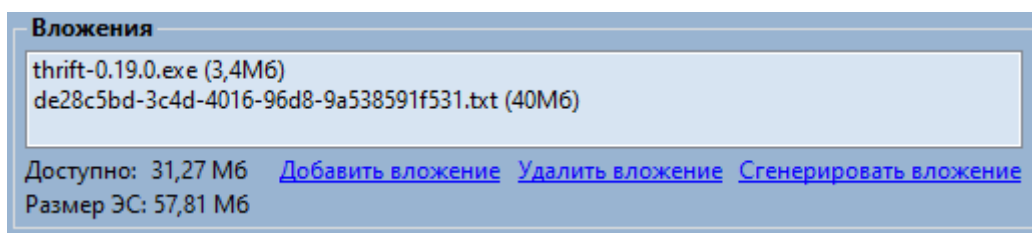


Рисунок 42 - Добавление сгенерированного файла

После нажатия на кнопку отправки происходит формирования сообщения и затем, в случае успеха, выводится окно с временем отправки (рисунок 43).

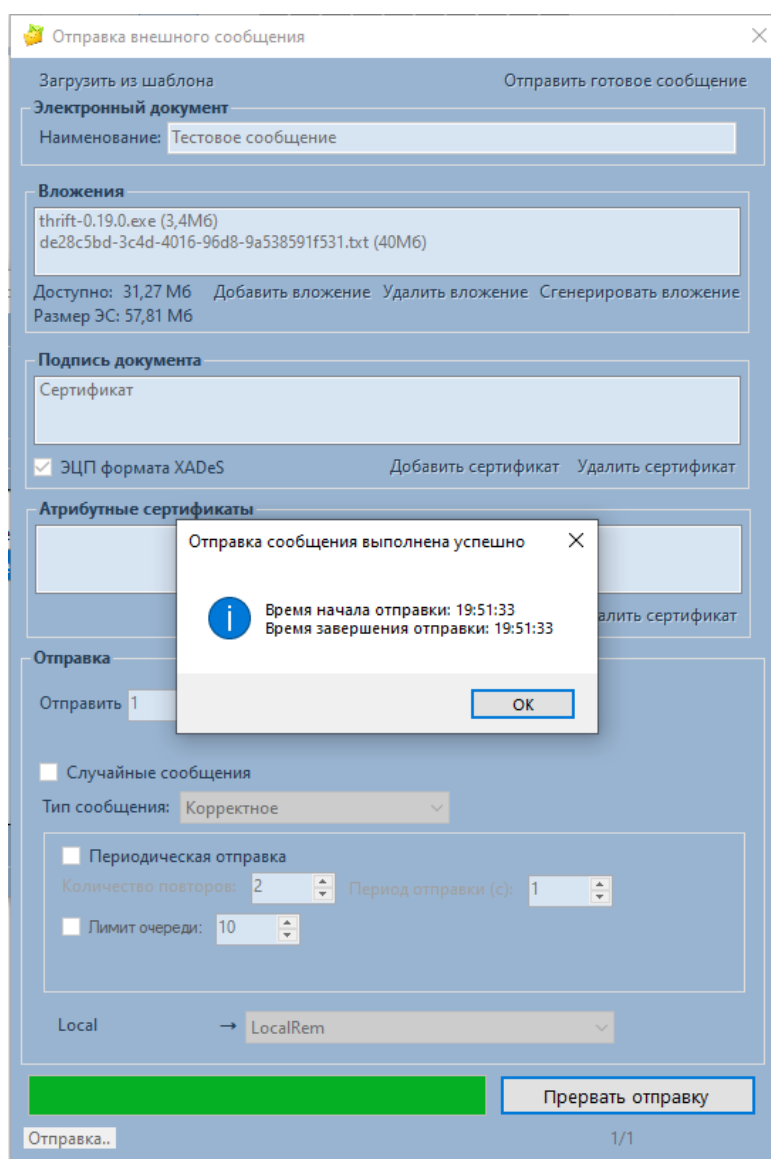


Рисунок 43 - Успешная отправка сообщения

После нажатия на кнопку «Ок» окно пропадет, а в нижней части окна появится идентификатор сообщения (рисунок 44). При этом если нажать на данное поле или воспользоваться комбинацией клавиш **ctrl+c**, то уникальный номер будет скопирован в буфер обмена.

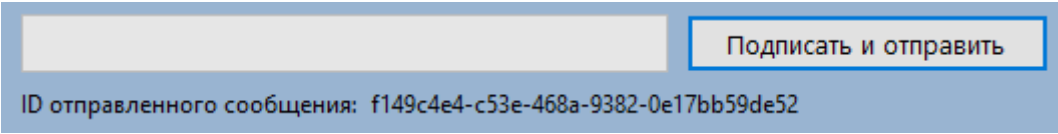


Рисунок 44 - Идентификатор отправленного сообщения

Теперь в главном окне приложения можно видеть отправленное сообщение (рисунок 45).

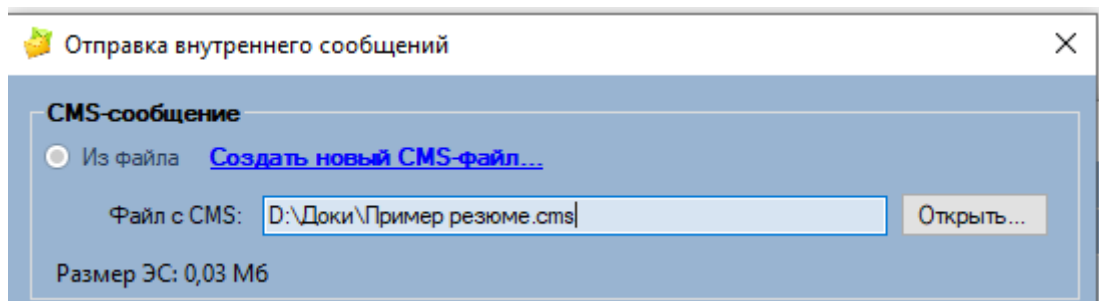
Тип сообщения	Адрес отправителя	Адрес получателя	Идентификатор сообщения	Идентификатор документа	Документ	Отправлено
Внешнее	MQ://LC/Local2	MQ://LC/Local1	f149c4e4-c53e-...	e552e79a-1869-...	Тестовое сооб...	05.06.2024 19:5...

Рисунок 45 - Отправленное сообщение

Таблица, представленная на рисунке 45 имеет ряд особенностей для взаимодействия с сообщениями. Помимо возможности сортировки столбцов, имеется также функционал просмотра содержимого. Для этого необходимо нажать правой кнопкой мыши по сообщению и в открывшемся меню выбрать пункт «Просмотр XML». После этого откроется окно в браузере Edge, и вы можете просмотреть тело сообщения в формате SOAP (рисунок 46).

[illegible]

Для отправки внутреннего сообщения необходимо иметь заранее созданный CMS-файл. В случае, когда у пользователя такого файла не окажется, он может его сгенерировать в приложении. Для этого необходимо нажать на гипер-текст «Создать новый CMS-файл...» в окне отправки внутреннего сообщения (рисунок 47).



Как видно из рисунка 48, кнопка создания заблокирована. Чтобы создать и подписать файл необходимо выбрать содержимое.

Рисунок 48 - Создание файла CMS

После выбора файла, необходимо определиться с директорией куда он будет сохранен и нажать на кнопку «Создать и подписать» (рисунок 49).

После создания файла, окно автоматически закроется и созданное вложение прикрепиться к сообщению, при этом обновится счетчик размера ЭС.

Рисунок 49 - Добавление созданного файла к ЭС

Точно также, как и для отправки внутреннего сообщения, после завершения отправки появится информационное сообщение с временем отправки (рисунок 50).

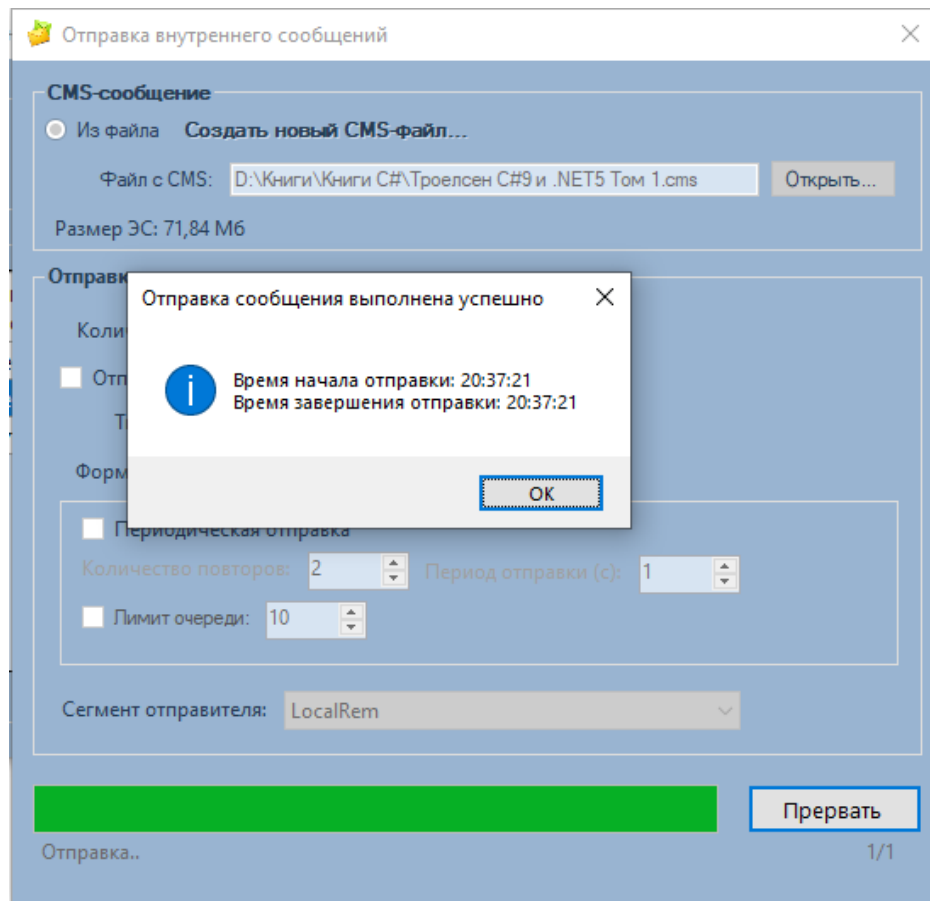


Рисунок 50 - Результат отправки внутреннего сообщения

В нижней части окна появляется идентификатор сообщения, а также выводится информация о времени отправки (рисунок 51).

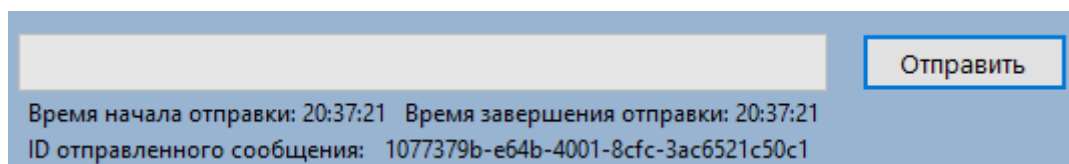


Рисунок 51 - Идентификатор отправленного внутреннего сообщения

После отправки его можно найти в таблице отправленных сообщений и просмотреть все необходимые параметры (рисунок 52).

2 Безопасность жизнедеятельности

При работе с персональными электронно вычислительными машинами необходимо учитывать ряд факторов влияющих на здоровье сотрудников. Основными источниками вредного воздействия на человеческий организм можно считать малую физическую активность при работе с ПЭВМ, а также электромагнитное излучение. Все это может негативно сказываться на продуктивности сотрудника и его физическом здоровье. Чтобы минимизировать вредные факторы необходимо соблюдать ряд правил при работе и организации рабочего пространства.

К таким правилам относятся:

- эргономика рабочего пространства;
- электробезопасность приборов.

2.1 Эргономика рабочего пространства

Понятие эргономика включает в себя целый ряд аспектов, начиная от удобства рабочего места, заканчивая относительной влажностью воздуха в помещении. Чтобы добиться оптимальных условий для работы необходимо следовать общепринятым требованиям к организации рабочего пространства.

2.1.1 Правила организации рабочего места

Так как большую часть времени оператор программист проводит сидя за ПЭВМ, то для сохранения здоровья сотрудников очень важно уделить внимание эргономике рабочего места.

К основным правилам организации рабочего места относятся:

- удобное кресло, позволяющее регулировать углы наклона спинки и подлокотников;
- расстояние от глаз до монитора должно быть не менее 600 мм;
- стол должен иметь пространство для ног, колени не должны упираться;

– вся проводка должна быть аккуратно уложена и не мешать работе сотрудника.

2.1.2 Освещение

Важную роль в здоровье зрительных органов играет освещенность рабочего места. Согласно требованиям СанПиН, допустимый предел освещенности составляет 300-500 лк. При этом освещенность экрана монитора не должна превышать 300 лк. Не допускается сильное мерцание искусственного света в помещении.

2.1.3 Климатические условия

Ещё одним важным аспектом в организации рабочего пространства играет микроклимат на рабочем месте. Например, температура воздуха в зимний период должна держаться в диапазоне от 22 до 24 градусов по Цельсию, в летний – от 23 до 25. При этом относительная влажность должна быть в пределах нормы и составлять от 40 до 60 процентов. Помещение должно хорошо проветриваться.

2.2 Электробезопасность приборов

Персональный компьютер и его периферийные устройства — это сложные технические приборы, при обращении с ними важно соблюдать ряд правил и требований. Например, все приборы должны быть исправны, а проводка изолирована. Сотрудник не должен самостоятельно чинить устройства. Приборы не должны создавать ЭМП превышающие допустимые пределы, установленные нормами СанПиН.

Заключение

В процессе выполнения выпускной квалификационной работы был разработан сервис формирования и проверки корректности электронных сообщений, для чего использованы современные технологии в области отправки сообщений, включающие средства для создания пользовательского интерфейса и работы с базами данных.

Данное приложение работает в рамках информационной системы предприятия и взаимодействует с ним через адаптер. Сервис позволяет пользователю выполнять функции по формированию сообщения нужного формата и отправлять сообщение в двух режимах. Приложение обладает удобным и понятным интерфейсом и соответствует всем поставленным задачам.

Задание на дипломное проектирование выполнено в полном объеме. В зависимости от требований заказчика приложение будет дорабатываться и улучшаться, например путем добавление возможностей для взаимодействия с брокером сообщений Apache Kafka.

Список использованных источников

1. Грегор Хоп, Бобби Вульф, Шаблоны интеграции корпоративных приложений. : Пер. с англ. М. : ООО “И.Д. Вильямс”, 2007. - 672 с.

2. IBM Documentation [Электронный ресурс]. URL: <https://www.ibm.com/docs/en/ibm-mq/9.3?topic=overview-mq-objects> (дата обращения: 24.04.2024)

3. Введение в SOAP и WSDL [Электронный ресурс]. URL: <https://vbeg.ru/tezam/soap-i-wsdl-teoriya/?ysclid=lwrsewbzhhb145714113> (дата обращения: 28.04.2024)

4. Троелсен, Э., Джепикс, Ф. Язык программирования C# 9 и платформа .NET 5: основные принципы и практики программирования, том 1, 10-е изд./Эндрю Троелсен, Филипп Джепикс; пер. с англ. Ю.Н. Артеменко. — Киев. : “Диалектика”, 2022.— 770 с.: ил. — Парал.тит. Англ. - ISBN 978-617-7987-80-1

5. Алекс Дэвис, Асинхронное программирование в C# 5.0. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2018. – 120 с.:

Приложение А

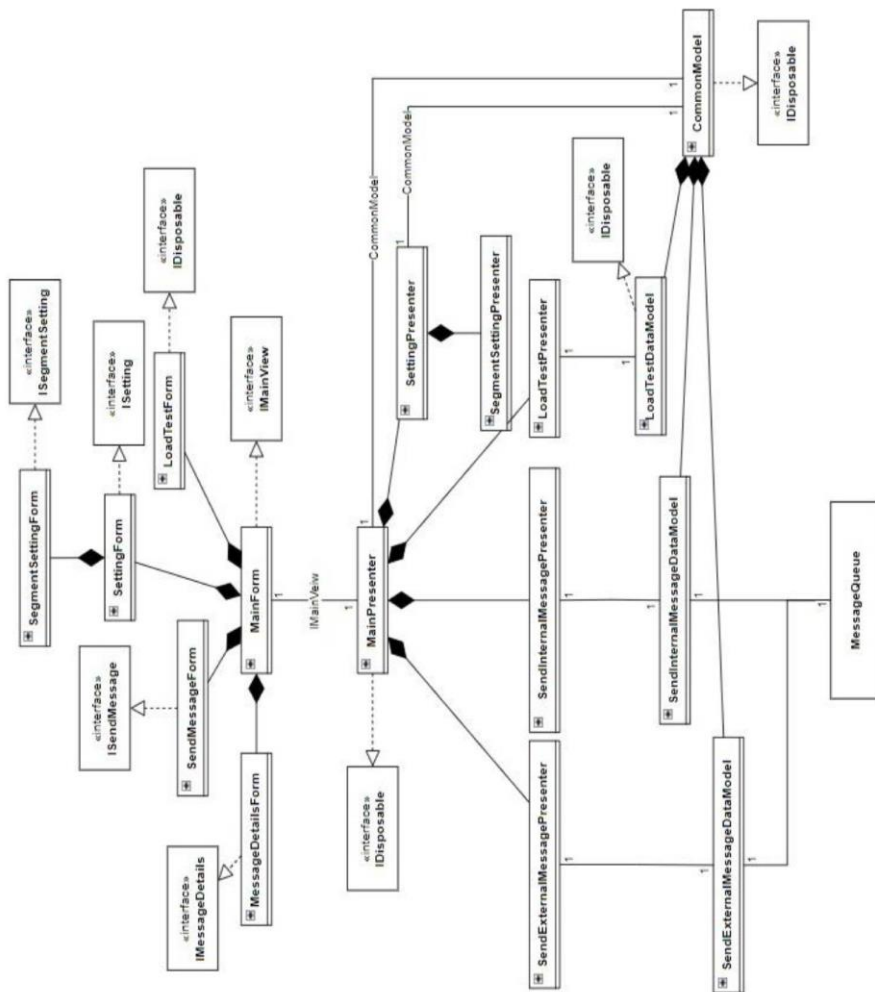
(Обязательное)

Сервис формирования и проверки корректности электронных сообщений

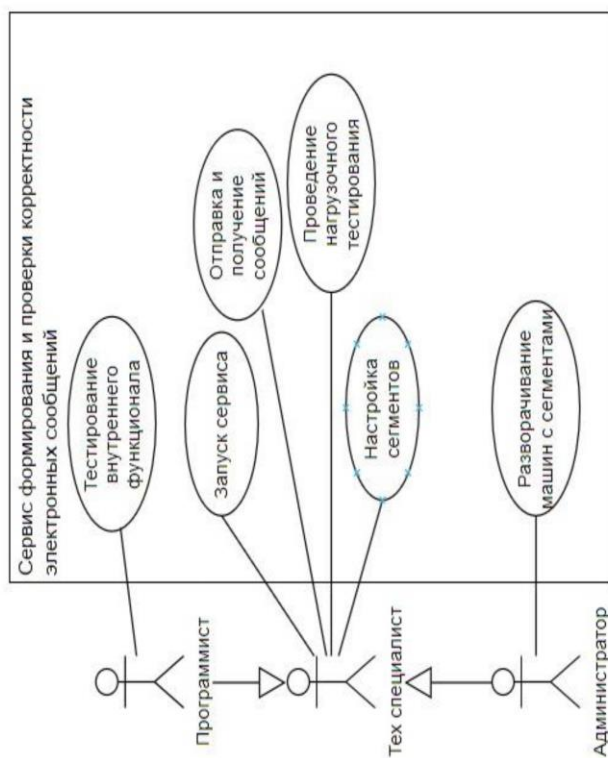
UML-диаграммы

					ПГУ1.090301.02.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

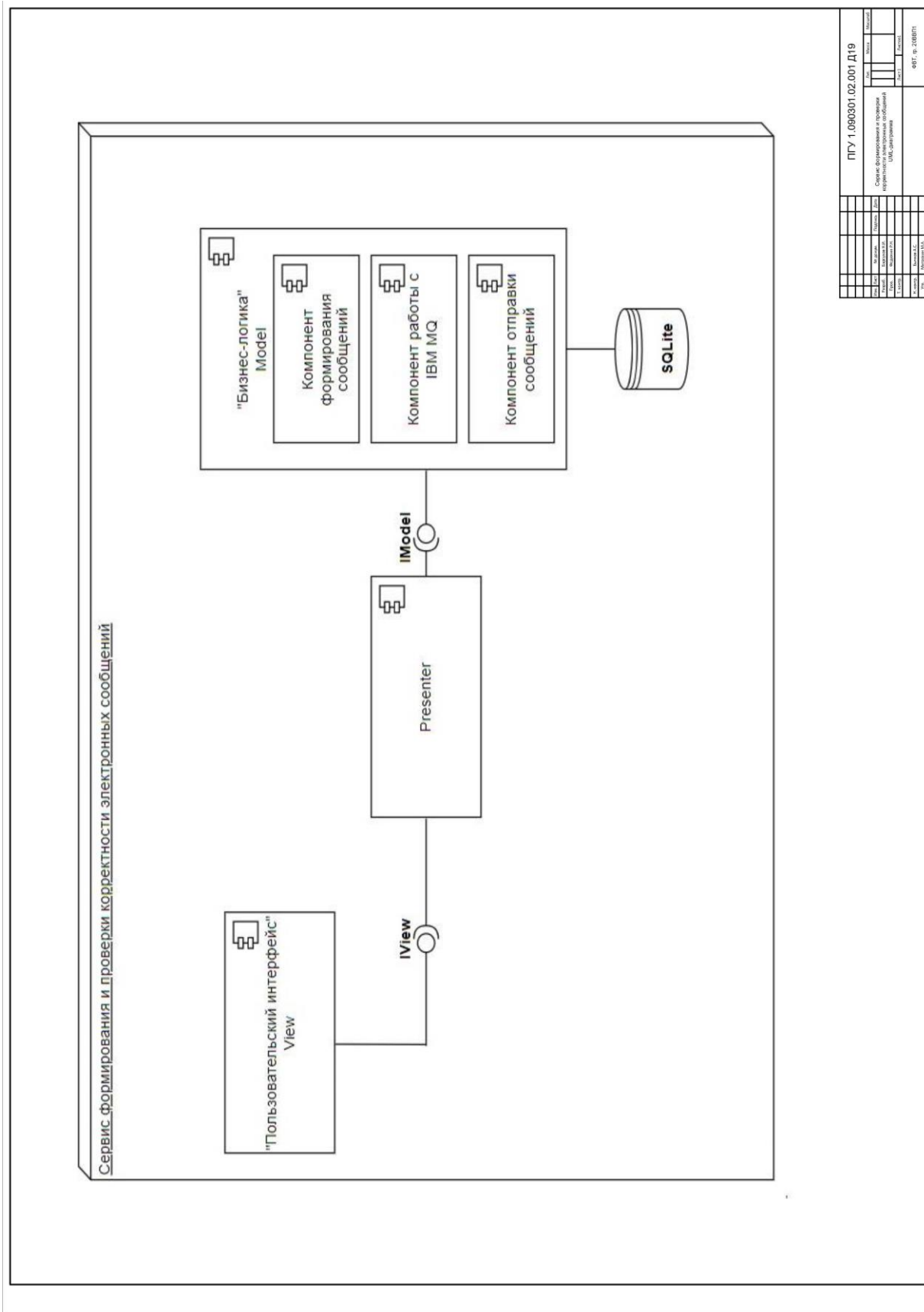
Приложение А.1 – UML-диаграмма классов

[illegible]

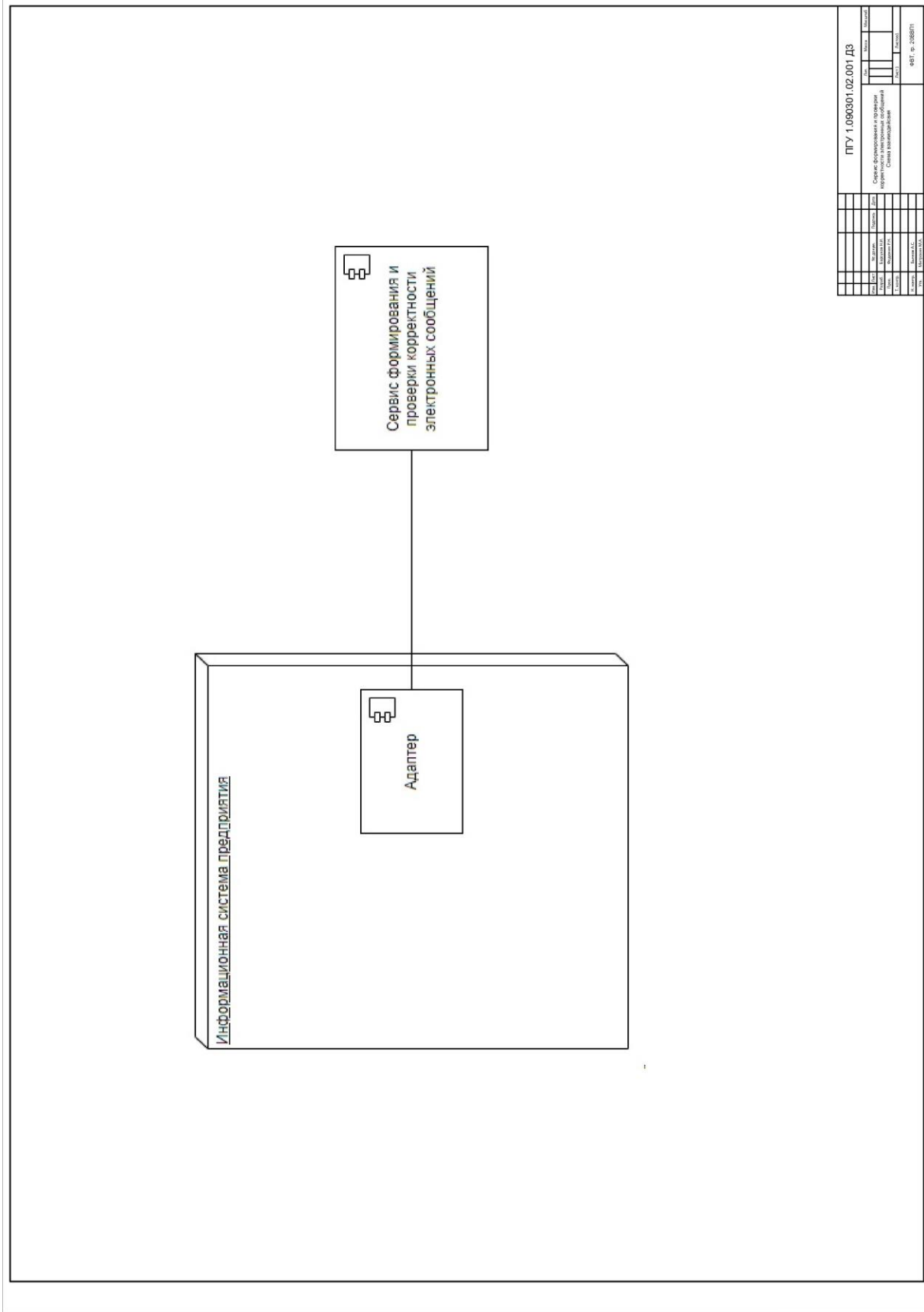
Приложение А.2 – UML-диаграмма прецедентов пользователя

[illegible]

Приложение А.3 – UML-диаграмма развертывания



Приложение А.4 – Схема взаимодействия компонентов системы



Изм.	Лист	№ докум.	Подпись	Дата

Приложение Б

(Обязательное)

Сервис формирования и проверки корректности электронных сообщений

Листинг программы

					ПГУ1.090301.02.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «MainView.Designer.cs»

```
namespace Views.Main
{
    partial class MainForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components =
null;

        #region Windows Form Designer generated code

        #endregion

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources
should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new
System.ComponentModel.Container();
            this.IncomingMessagesLinkLabel = new
System.Windows.Forms.LinkLabel();
            this.OutgoingMessagesLinkLabel = new
System.Windows.Forms.LinkLabel();
            this.ControlGroupBox = new
System.Windows.Forms.GroupBox();
            this.panel3 = new System.Windows.Forms.Panel();
            this.panel4 = new System.Windows.Forms.Panel();
            this.panel1 = new System.Windows.Forms.Panel();
            this.panel2 = new System.Windows.Forms.Panel();
        }
    }
}
```

```

        this.GetMessageButton = new
System.Windows.Forms.Button();
        this.ErrorCounterLabel = new
System.Windows.Forms.Label();
        this.NewMessageButton = new
System.Windows.Forms.Button();
        this.LoadTestButton = new
System.Windows.Forms.Button();
        this.SettingsButton = new
System.Windows.Forms.Button();
        this.CurrentSegmentLabel = new
System.Windows.Forms.Label();
        this.CurrentSegmentComboBox = new
System.Windows.Forms.ComboBox();
        this.MessageGroupBox = new
System.Windows.Forms.GroupBox();
        this.DeleteLinkLabel = new
System.Windows.Forms.LinkLabel();
        this.GroupBox3 = new
System.Windows.Forms.GroupBox();
        this.IncomingMessagesDataGridView = new
System.Windows.Forms.DataGridView();
        this.OutgoingMessagesDataGridview = new
System.Windows.Forms.DataGridView();
        this.OutgoingContextMenuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
        this.ChangeAndRetrySendToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.RetrySendToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.OpenAsXMLToolStripMenuItem1 = new
System.Windows.Forms.ToolStripItem();
        this.CheckoutSignatureToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.CopyOutDocIdMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.CopyOutMessageIdMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.IncomingContextMenuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
        this.DetailMessageToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.OpenAsXMLToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.CheckSignToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.CopyInDocIdMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.CopyInMessageIdMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.ToolTip = new
System.Windows.Forms.ToolTip(this.components);

```

```

        this.DvcsOutgoingContextMenuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
        this.ShowDvcsBodyMenuToolItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.ShowDvcsRequestCmsToolItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.OpenSendedDvcsMessageDirMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.CopyDvcsOutMessageIdMenuStrip = new
System.Windows.Forms.ToolStripMenuItem();
        this.DvcsIncomingContextMenuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
        this.ShowIncominDvcsMsgBodyMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.ShowIncominDvcsMsgDvcsMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.ShowIncominDvcsMsgDirMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.CopyDvcsInMessageIdMenuStrip = new
System.Windows.Forms.ToolStripMenuItem();
        this.SendMessageContextMenuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
        this.toolStripMenuItem1 = new
System.Windows.Forms.ToolStripMenuItem();
        this.toolStripMenuItem2 = new
System.Windows.Forms.ToolStripMenuItem();
        this.LoadTestContextMenuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
        this.RunLoadTestG2GModeButton = new
System.Windows.Forms.ToolStripMenuItem();
        this.RunLoadTestB2GModeButton = new
System.Windows.Forms.ToolStripMenuItem();
        this.ControlGroupBox.SuspendLayout();
        this.panel3.SuspendLayout();
        this.panell1.SuspendLayout();
        this.MessageGroupBox.SuspendLayout();
        this.GroupBox3.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.IncomingMessage
sDataGridView)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.OutgoingMessage
sDataGridView)).BeginInit();
        this.OutgoingContextMenuStrip.SuspendLayout();
        this.IncomingContextMenuStrip.SuspendLayout();
        this.DvcsOutgoingContextMenuStrip.SuspendLayout();
        this.DvcsIncomingContextMenuStrip.SuspendLayout();
        this.SendMessageContextMenuStrip.SuspendLayout();
        this.LoadTestContextMenuStrip.SuspendLayout();
        this.SuspendLayout();
        //
        // IncomingMessagesLinkLabel

```

```

        //
        this.IncomingMessagesLinkLabel.AutoSize = true;
        this.IncomingMessagesLinkLabel.Font = new
System.Drawing.Font("Segoe UI", 9.75F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
        this.IncomingMessagesLinkLabel.Location = new
System.Drawing.Point(13, 22);
        this.IncomingMessagesLinkLabel.Name =
"IncomingMessagesLinkLabel";
        this.IncomingMessagesLinkLabel.Size = new
System.Drawing.Size(73, 17);
        this.IncomingMessagesLinkLabel.TabIndex = 1;
        this.IncomingMessagesLinkLabel.TabStop = true;
        this.IncomingMessagesLinkLabel.Text = "Входящие";
        this.IncomingMessagesLinkLabel.LinkClicked += new
System.Windows.Forms.LinkLabelLinkClickedEventHandler(this.Incom
ingLinkClick);
        //
        // OutgoingMessagesLinkLabel
        //
        this.OutgoingMessagesLinkLabel.AutoSize = true;
        this.OutgoingMessagesLinkLabel.Location = new
System.Drawing.Point(13, 52);
        this.OutgoingMessagesLinkLabel.Name =
"OutgoingMessagesLinkLabel";
        this.OutgoingMessagesLinkLabel.Size = new
System.Drawing.Size(96, 17);
        this.OutgoingMessagesLinkLabel.TabIndex = 2;
        this.OutgoingMessagesLinkLabel.TabStop = true;
        this.OutgoingMessagesLinkLabel.Text =
"Отправленные";
        this.OutgoingMessagesLinkLabel.LinkClicked += new
System.Windows.Forms.LinkLabelLinkClickedEventHandler(this.Outgo
ingLinkClick);
        //
        // ControlGroupBox
        //
        this.ControlGroupBox.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anc
h
orStyles.Top | System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right));
        this.ControlGroupBox.BackColor =
System.Drawing.SystemColors.ActiveCaption;
        this.ControlGroupBox.Controls.Add(this.panel3);
        this.ControlGroupBox.Controls.Add(this.panel1);

this.ControlGroupBox.Controls.Add(this.GetMessageButton);

this.ControlGroupBox.Controls.Add(this.ErrorCounterLable);

this.ControlGroupBox.Controls.Add(this.NewMessageButton);

```

```

this.ControlGroupBox.Controls.Add(this.LoadTestButton);

this.ControlGroupBox.Controls.Add(this.SettingsButton);

this.ControlGroupBox.Controls.Add(this.CurrentSegmentLabel);

this.ControlGroupBox.Controls.Add(this.CurrentSegmentComboBox);
    this.ControlGroupBox.Location = new
System.Drawing.Point(14, 14);
    this.ControlGroupBox.Margin = new
System.Windows.Forms.Padding(3, 5, 3, 5);
    this.ControlGroupBox.Name = "ControlGroupBox";
    this.ControlGroupBox.Padding = new
System.Windows.Forms.Padding(3, 5, 3, 5);
    this.ControlGroupBox.Size = new
System.Drawing.Size(974, 76);
    this.ControlGroupBox.TabIndex = 4;
    this.ControlGroupBox.TabStop = false;
    this.ControlGroupBox.Text = "Управление";
    //
    // panel3
    //
    this.panel3.BackColor =
System.Drawing.SystemColors.ControlDark;
    this.panel3.Controls.Add(this.panel4);
    this.panel3.Location = new System.Drawing.Point(143,
20);

    this.panel3.Name = "panel3";
    this.panel3.Size = new System.Drawing.Size(2, 45);
    this.panel3.TabIndex = 4;
    //
    // panel4
    //
    this.panel4.BackColor =
System.Drawing.SystemColors.ControlDark;
    this.panel4.Location = new System.Drawing.Point(32,
0);

    this.panel4.Name = "panel4";
    this.panel4.Size = new System.Drawing.Size(2, 52);
    this.panel4.TabIndex = 4;
    //
    // panel1
    //
    this.panel1.BackColor =
System.Drawing.SystemColors.ControlDark;
    this.panel1.Controls.Add(this.panel2);
    this.panel1.Location = new System.Drawing.Point(96,
20);

    this.panel1.Name = "panel1";
    this.panel1.Size = new System.Drawing.Size(2, 45);
    this.panel1.TabIndex = 3;

```

```

        //
        // panel2
        //
        this.panel2.BackColor =
System.Drawing.SystemColors.ControlDark;
        this.panel2.Location = new System.Drawing.Point(32,
0);

        this.panel2.Name = "panel2";
        this.panel2.Size = new System.Drawing.Size(2, 52);
        this.panel2.TabIndex = 4;
        //
        // GetMessageButton
        //
        this.GetMessageButton.BackColor =
System.Drawing.Color.Transparent;
        this.GetMessageButton.BackgroundImage =
global::TTPEmulV2.Properties.Resources.GetMessage;
        this.GetMessageButton.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Zoom;
        this.GetMessageButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.GetMessageButton.ForeColor =
System.Drawing.SystemColors.Control;
        this.GetMessageButton.Location = new
System.Drawing.Point(54, 26);
        this.GetMessageButton.Name = "GetMessageButton";
        this.GetMessageButton.Size = new
System.Drawing.Size(40, 35);
        this.GetMessageButton.TabIndex = 12;
        this.ToolTip.SetToolTip(this.GetMessageButton,
"Получить сообщение");
        this.GetMessageButton.UseVisualStyleBackColor =
false;

        this.GetMessageButton.Click += new
System.EventHandler(this.GetMessageButtonClick);
        //
        // ErrorCounterLable
        //
        this.ErrorCounterLable.AutoSize = true;
        this.ErrorCounterLable.Location = new
System.Drawing.Point(302, 35);
        this.ErrorCounterLable.Name = "ErrorCounterLable";
        this.ErrorCounterLable.Size = new
System.Drawing.Size(212, 17);
        this.ErrorCounterLable.TabIndex = 9;
        this.ErrorCounterLable.Text = "Ошибок получения
сообщений: 0";
        this.ErrorCounterLable.TextAlign =
System.Drawing.ContentAlignment.MiddleLeft;
        this.ErrorCounterLable.Visible = false;
        //
        // NewMessageButton

```

```

        //
        this.NewMessageButton.BackColor =
System.Drawing.Color.Transparent;
        this.NewMessageButton.BackgroundImage =
global::TTPEmulV2.Properties.Resources.SendMessage;
        this.NewMessageButton.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Zoom;
        this.NewMessageButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.NewMessageButton.ForeColor =
System.Drawing.SystemColors.Control;
        this.NewMessageButton.Location = new
System.Drawing.Point(8, 26);
        this.NewMessageButton.Name = "NewMessageButton";
        this.NewMessageButton.Size = new
System.Drawing.Size(40, 35);
        this.NewMessageButton.TabIndex = 11;
        this.ToolTip.SetToolTip(this.NewMessageButton,
"Отправить сообщение");
        this.NewMessageButton.UseVisualStyleBackColor =
false;
        this.NewMessageButton.Click += new
System.EventHandler(this.NewMessageButtonClick);
        //
        // LoadTestButton
        //
        this.LoadTestButton.BackColor =
System.Drawing.Color.Transparent;
        this.LoadTestButton.BackgroundImage =
global::TTPEmulV2.Properties.Resources.loadTest;
        this.LoadTestButton.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Stretch;
        this.LoadTestButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.LoadTestButton.ForeColor =
System.Drawing.SystemColors.Control;
        this.LoadTestButton.Location = new
System.Drawing.Point(102, 26);
        this.LoadTestButton.Name = "LoadTestButton";
        this.LoadTestButton.Size = new
System.Drawing.Size(35, 35);
        this.LoadTestButton.TabIndex = 10;
        this.ToolTip.SetToolTip(this.LoadTestButton,
"Нагрузочное тестирование");
        this.LoadTestButton.UseVisualStyleBackColor = false;
        this.LoadTestButton.Click += new
System.EventHandler(this.LoadTestButtonClicked);
        //
        // SettingsButton
        //
        this.SettingsButton.BackColor =
System.Drawing.Color.Transparent;

```

```

        this.SettingsButton.BackgroundImage =
global::TTPEmulV2.Properties.Resources.settings_40;
        this.SettingsButton.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Stretch;
        this.SettingsButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.SettingsButton.ForeColor =
System.Drawing.SystemColors.Control;
        this.SettingsButton.Location = new
System.Drawing.Point(151, 26);
        this.SettingsButton.Name = "SettingsButton";
        this.SettingsButton.Size = new
System.Drawing.Size(35, 35);
        this.SettingsButton.TabIndex = 6;
        this.ToolTip.SetToolTip(this.SettingsButton,
"Настройки");
        this.SettingsButton.UseVisualStyleBackColor = false;
        this.SettingsButton.Click += new
System.EventHandler(this.SettingsButtonClick);
        //
        // CurrentSegmentLabel
        //
        this.CurrentSegmentLabel.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Ancho
rStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
        this.CurrentSegmentLabel.Location = new
System.Drawing.Point(564, 31);
        this.CurrentSegmentLabel.Name =
"CurrentSegmentLabel";
        this.CurrentSegmentLabel.Size = new
System.Drawing.Size(112, 25);
        this.CurrentSegmentLabel.TabIndex = 5;
        this.CurrentSegmentLabel.Text = "Текущий сегмент:";
        this.CurrentSegmentLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
        //
        // CurrentSegmentComboBox
        //
        this.CurrentSegmentComboBox.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Ancho
rStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
        this.CurrentSegmentComboBox.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
        this.CurrentSegmentComboBox.FormattingEnabled =
true;
        this.CurrentSegmentComboBox.Location = new
System.Drawing.Point(682, 32);
        this.CurrentSegmentComboBox.Margin = new
System.Windows.Forms.Padding(3, 5, 3, 5);
        this.CurrentSegmentComboBox.Name =
"CurrentSegmentComboBox";

```

```

        this.CurrentSegmentComboBox.Size = new
System.Drawing.Size(281, 25);
        this.CurrentSegmentComboBox.TabIndex = 6;
        this.CurrentSegmentComboBox.SelectedIndexChanged +=
new
System.EventHandler(this.CbCurrentSegmentSelectedIndexChange);
        //
        // MessageGroupBox
        //
        this.MessageGroupBox.Anchor =
((System.Windows.Forms.AnchorStyles) (((System.Windows.Forms.Anc
horStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right)));

this.MessageGroupBox.Controls.Add(this.DeleteLinkLabel);
        this.MessageGroupBox.Controls.Add(this.GroupBox3);

this.MessageGroupBox.Controls.Add(this.IncomingMessagesDataGridV
iew);

this.MessageGroupBox.Controls.Add(this.OutgoingMessagesDataGridV
iew);

        this.MessageGroupBox.Location = new
System.Drawing.Point(14, 100);
        this.MessageGroupBox.Margin = new
System.Windows.Forms.Padding(3, 5, 3, 5);
        this.MessageGroupBox.Name = "MessageGroupBox";
        this.MessageGroupBox.Padding = new
System.Windows.Forms.Padding(3, 5, 3, 5);
        this.MessageGroupBox.Size = new
System.Drawing.Size(974, 292);
        this.MessageGroupBox.TabIndex = 5;
        this.MessageGroupBox.TabStop = false;
        this.MessageGroupBox.Text = "Сообщения";
        //
        // DeleteLinkLabel
        //
        this.DeleteLinkLabel.Anchor =
((System.Windows.Forms.AnchorStyles) ((System.Windows.Forms.Ancho
rStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
        this.DeleteLinkLabel.AutoSize = true;
        this.DeleteLinkLabel.Font = new
System.Drawing.Font("Segoe UI", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        this.DeleteLinkLabel.Location = new
System.Drawing.Point(721, 266);
        this.DeleteLinkLabel.Name = "DeleteLinkLabel";
        this.DeleteLinkLabel.Size = new
System.Drawing.Size(216, 17);
        this.DeleteLinkLabel.TabIndex = 7;

```

```

        this.DeleteLinkLabel.TabStop = true;
        this.DeleteLinkLabel.Text = "Удалить все входящие
сообщения";
        this.DeleteLinkLabel.LinkClicked += new
System.Windows.Forms.LinkLabelLinkClickedEventHandler(this.Clear
LinkLabelClicked);
        //
        // GroupBox3
        //
        this.GroupBox3.Anchor =
((System.Windows.Forms.AnchorStyles) (((System.Windows.Forms.Anc
horStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)));
        this.GroupBox3.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;

this.GroupBox3.Controls.Add(this.IncomingMessagesLinkLabel);

this.GroupBox3.Controls.Add(this.OutgoingMessagesLinkLabel);
        this.GroupBox3.Location = new
System.Drawing.Point(8, 24);
        this.GroupBox3.Margin = new
System.Windows.Forms.Padding(3, 5, 3, 5);
        this.GroupBox3.Name = "GroupBox3";
        this.GroupBox3.Padding = new
System.Windows.Forms.Padding(3, 5, 3, 5);
        this.GroupBox3.Size = new System.Drawing.Size(157,
238);
        this.GroupBox3.TabIndex = 5;
        this.GroupBox3.TabStop = false;
        //
        // IncomingMessagesDataGridView
        //
        this.IncomingMessagesDataGridView.AllowUserToAddRows
= false;

this.IncomingMessagesDataGridView.AllowUserToDeleteRows = false;

this.IncomingMessagesDataGridView.AllowUserToResizeRows = false;
        this.IncomingMessagesDataGridView.Anchor =
((System.Windows.Forms.AnchorStyles) (((System.Windows.Forms.Anc
horStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right)));

this.IncomingMessagesDataGridView.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill;
        this.IncomingMessagesDataGridView.BackgroundColor =
System.Drawing.SystemColors.GradientInactiveCaption;

this.IncomingMessagesDataGridView.ColumnHeadersHeightSizeMode =

```

```

System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
        this.IncomingMessagesDataGridView.Location = new
System.Drawing.Point(175, 24);
        this.IncomingMessagesDataGridView.Margin = new
System.Windows.Forms.Padding(3, 5, 3, 5);
        this.IncomingMessagesDataGridView.MultiSelect =
false;
        this.IncomingMessagesDataGridView.Name =
"IncomingMessagesDataGridView";
        this.IncomingMessagesDataGridView.ReadOnly = true;
        this.IncomingMessagesDataGridView.RowHeadersVisible
= false;
        this.IncomingMessagesDataGridView.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect;
        this.IncomingMessagesDataGridView.Size = new
System.Drawing.Size(788, 237);
        this.IncomingMessagesDataGridView.TabIndex = 3;

this.IncomingMessagesDataGridView.CellMouseDoubleClick += new
System.Windows.Forms.DataGridViewCellMouseEventHandler(this.Inco
mingMessagesDataGridViewCellMouseDoubleClick);
        this.IncomingMessagesDataGridView.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.IncomingMouseDown);
        //
        // OutgoingMessagesDataGridView
        //
        this.OutgoingMessagesDataGridView.AllowUserToAddRows
= false;

this.OutgoingMessagesDataGridView.AllowUserToDeleteRows = false;

this.OutgoingMessagesDataGridView.AllowUserToResizeRows = false;
        this.OutgoingMessagesDataGridView.Anchor =
((System.Windows.Forms.AnchorStyles) (((System.Windows.Forms.Anc
horStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right)));

this.OutgoingMessagesDataGridView.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill;
        this.OutgoingMessagesDataGridView.BackgroundColor =
System.Drawing.SystemColors.GradientInactiveCaption;

this.OutgoingMessagesDataGridView.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.Auto
Size;
        this.OutgoingMessagesDataGridView.Location = new
System.Drawing.Point(175, 24);
        this.OutgoingMessagesDataGridView.Margin = new
System.Windows.Forms.Padding(3, 5, 3, 5);

```

```

        this.OutgoingMessagesDataGridView.MultiSelect =
false;
        this.OutgoingMessagesDataGridView.Name =
"OutgoingMessagesDataGridView";
        this.OutgoingMessagesDataGridView.ReadOnly = true;
        this.OutgoingMessagesDataGridView.RowHeadersVisible
= false;
        this.OutgoingMessagesDataGridView.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect;
        this.OutgoingMessagesDataGridView.Size = new
System.Drawing.Size(788, 237);
        this.OutgoingMessagesDataGridView.TabIndex = 4;

this.OutgoingMessagesDataGridView.CellMouseDoubleClick += new
System.Windows.Forms.DataGridViewCellEventHandler(this.Outg
oingMessagesDataGridViewCellMouseDoubleClick);
        this.OutgoingMessagesDataGridView.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.OutgoingMouseDown);
        //
        // OutgoingContextMenuStrip
        //
        this.OutgoingContextMenuStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
        this.ChangeAndRetrySendToolStripMenuItem,
        this.RetrySendToolStripMenuItem,
        this.OpenAsXMLToolStripMenuItem1,
        this.CheckoutSignatureToolStripMenuItem,
        this.CopyOutDocIdMenuItem,
        this.CopyOutMessageIdMenuItem});
        this.OutgoingContextMenuStrip.Name =
"contextMenuStrip1";
        this.OutgoingContextMenuStrip.Size = new
System.Drawing.Size(302, 136);
        //
        // ChangeAndRetrySendToolStripMenuItem
        //
        this.ChangeAndRetrySendToolStripMenuItem.Name =
"ChangeAndRetrySendToolStripMenuItem";
        this.ChangeAndRetrySendToolStripMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.ChangeAndRetrySendToolStripMenuItem.Text =
"Изменить и повторить отправку";
        this.ChangeAndRetrySendToolStripMenuItem.Click +=
new System.EventHandler(this.ChangeAndReSendClickHandler);
        //
        // RetrySendToolStripMenuItem
        //
        this.RetrySendToolStripMenuItem.Name =
"RetrySendToolStripMenuItem";
        this.RetrySendToolStripMenuItem.Size = new
System.Drawing.Size(301, 22);

```

```

        this.RetrySendToolStripMenuItem.Text = "Повторить
отправку";
        this.RetrySendToolStripMenuItem.Click += new
System.EventHandler(this.RepeatSendClick);
        //
        // OpenAsXMLToolStripMenuItem1
        //
        this.OpenAsXMLToolStripMenuItem1.Name =
"OpenAsXMLToolStripMenuItem1";
        this.OpenAsXMLToolStripMenuItem1.Size = new
System.Drawing.Size(301, 22);
        this.OpenAsXMLToolStripMenuItem1.Text = "Открыть как
XML";
        this.OpenAsXMLToolStripMenuItem1.Click += new
System.EventHandler(this.OutgoingOpenXmlClick);
        //
        // CheckoutSignatureToolStripMenuItem
        //
        this.CheckoutSignatureToolStripMenuItem.Name =
"CheckoutSignatureToolStripMenuItem";
        this.CheckoutSignatureToolStripMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.CheckoutSignatureToolStripMenuItem.Text =
"Проверить подпись";
        this.CheckoutSignatureToolStripMenuItem.Click += new
System.EventHandler(this.CheckOutgoingMessageSignClick);
        //
        // CopyOutDocIdMenuItem
        //
        this.CopyOutDocIdMenuItem.Name =
"CopyOutDocIdMenuItem";
        this.CopyOutDocIdMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.CopyOutDocIdMenuItem.Text = "Скопировать
идентификатор документа";
        this.CopyOutDocIdMenuItem.Click += new
System.EventHandler(this.CopyOutDocIdMenuItemClick);
        //
        // CopyOutMessageIdMenuItem
        //
        this.CopyOutMessageIdMenuItem.Name =
"CopyOutMessageIdMenuItem";
        this.CopyOutMessageIdMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.CopyOutMessageIdMenuItem.Text = "Скопировать
идентификатор сообщения";
        this.CopyOutMessageIdMenuItem.Click += new
System.EventHandler(this.CopyOutMessageIdMenuItemClick);
        //
        // IncomingContextMenuStrip
        //

```

```

        this.IncomingContextMenuStrip.Items.AddRange (new
System.Windows.Forms.ToolStripItem[] {
        this.DetailMessageToolStripMenuItem,
        this.OpenAsXMLToolStripMenuItem,
        this.CheckSignToolStripMenuItem,
        this.CopyInDocIdMenuItem,
        this.CopyInMessageIdMenuItem});
        this.IncomingContextMenuStrip.Name =
"CmsIncomingContextMenuStrip";
        this.IncomingContextMenuStrip.Size = new
System.Drawing.Size(302, 114);
        //
        // DetailMessageToolStripMenuItem
        //
        this.DetailMessageToolStripMenuItem.Name =
"DetailMessageToolStripMenuItem";
        this.DetailMessageToolStripMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.DetailMessageToolStripMenuItem.Text = "Детали
сообщения";
        this.DetailMessageToolStripMenuItem.Click += new
System.EventHandler(this.DetailMessageToolStripMenuItem_Click);
        //
        // OpenAsXMLToolStripMenuItem
        //
        this.OpenAsXMLToolStripMenuItem.Name =
"OpenAsXMLToolStripMenuItem";
        this.OpenAsXMLToolStripMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.OpenAsXMLToolStripMenuItem.Text = "Открыть как
XML";
        this.OpenAsXMLToolStripMenuItem.Click += new
System.EventHandler(this.IncomingOpenXmlClick);
        //
        // CheckSignToolStripMenuItem
        //
        this.CheckSignToolStripMenuItem.Name =
"CheckSignToolStripMenuItem";
        this.CheckSignToolStripMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.CheckSignToolStripMenuItem.Text = "Проверить
подпись";
        this.CheckSignToolStripMenuItem.Click += new
System.EventHandler(this.CheckIncomingMessageSignClick);
        //
        // CopyInDocIdMenuItem
        //
        this.CopyInDocIdMenuItem.Name =
"CopyInDocIdMenuItem";
        this.CopyInDocIdMenuItem.Size = new
System.Drawing.Size(301, 22);

```

```

        this.CopyInDocIdMenuItem.Text = "Скопировать
идентификатор документа";
        this.CopyInDocIdMenuItem.Click += new
System.EventHandler(this.CopyInDocIdMenuItemClick);
        //
        // CopyInMessageIdMenuItem
        //
        this.CopyInMessageIdMenuItem.Name =
"CopyInMessageIdMenuItem";
        this.CopyInMessageIdMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.CopyInMessageIdMenuItem.Text = "Скопировать
идентификатор сообщения";
        this.CopyInMessageIdMenuItem.Click += new
System.EventHandler(this.CopyInMessageIdMenuItemClick);
        //
        // DvcsOutgoingContextMenuStrip
        //
        this.DvcsOutgoingContextMenuStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
        this.ShowDvcsBodyMenuToolItem,
        this.ShowDvcsRequestCmsToolItem,
        this.OpenSendedDvcsMessageDirMenuItem,
        this.CopyDvcsOutMessageIdMenuStrip});
        this.DvcsOutgoingContextMenuStrip.Name =
"contextMenuStrip1";
        this.DvcsOutgoingContextMenuStrip.Size = new
System.Drawing.Size(302, 92);
        //
        // ShowDvcsBodyMenuToolItem
        //
        this.ShowDvcsBodyMenuToolItem.Name =
"ShowDvcsBodyMenuToolItem";
        this.ShowDvcsBodyMenuToolItem.Size = new
System.Drawing.Size(301, 22);
        this.ShowDvcsBodyMenuToolItem.Text = "Показать тело
сообщения";
        this.ShowDvcsBodyMenuToolItem.Click += new
System.EventHandler(this.OnShowDvcsBodyMenuToolItemClickHandler)
;
        //
        // ShowDvcsRequestCmsToolItem
        //
        this.ShowDvcsRequestCmsToolItem.Name =
"ShowDvcsRequestCmsToolItem";
        this.ShowDvcsRequestCmsToolItem.Size = new
System.Drawing.Size(301, 22);
        this.ShowDvcsRequestCmsToolItem.Text = "Показать
DVCS-запрос";
        this.ShowDvcsRequestCmsToolItem.Click += new
System.EventHandler(this.OnShowDvcsRequestCmsToolItemClickHandle
r);

```

```

//
// OpenSendedDvcsMessageDirMenuItem
//
this.OpenSendedDvcsMessageDirMenuItem.Name =
"OpenSendedDvcsMessageDirMenuItem";
this.OpenSendedDvcsMessageDirMenuItem.Size = new
System.Drawing.Size(301, 22);
this.OpenSendedDvcsMessageDirMenuItem.Text =
"Открыть каталог";
this.OpenSendedDvcsMessageDirMenuItem.Click += new
System.EventHandler(this.OnOpenSendedDvcsMessageDirMenuItemClick
Handler);
//
// CopyDvcsOutMessageIdMenuStrip
//
this.CopyDvcsOutMessageIdMenuStrip.Name =
"CopyDvcsOutMessageIdMenuStrip";
this.CopyDvcsOutMessageIdMenuStrip.Size = new
System.Drawing.Size(301, 22);
this.CopyDvcsOutMessageIdMenuStrip.Text =
"Скопировать идентификатор сообщения";
this.CopyDvcsOutMessageIdMenuStrip.Click += new
System.EventHandler(this.OnCopyDvcsOutMessageIdMenuStripClickHan
dler);
//
// DvcsIncomingContextMenuStrip
//
this.DvcsIncomingContextMenuStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
this.ShowIncominDvcsMsgBodyMenuItem,
this.ShowIncominDvcsMsgDvcsMenuItem,
this.ShowIncominDvcsMsgDirMenuItem,
this.CopyDvcsInMessageIdMenuStrip});
this.DvcsIncomingContextMenuStrip.Name =
"CmsIncomingContextMenuStrip";
this.DvcsIncomingContextMenuStrip.Size = new
System.Drawing.Size(302, 92);
//
// ShowIncominDvcsMsgBodyMenuItem
//
this.ShowIncominDvcsMsgBodyMenuItem.Name =
"ShowIncominDvcsMsgBodyMenuItem";
this.ShowIncominDvcsMsgBodyMenuItem.Size = new
System.Drawing.Size(301, 22);
this.ShowIncominDvcsMsgBodyMenuItem.Text = "Показать
тело сообщения";
this.ShowIncominDvcsMsgBodyMenuItem.Click += new
System.EventHandler(this.OnShowIncominDvcsMsgBodyMenuItemClickHa
ndler);
//
// ShowIncominDvcsMsgDvcsMenuItem
//

```

```

        this.ShowIncominDvcsMsgDvcsMenuItem.Name =
"ShowIncominDvcsMsgDvcsMenuItem";
        this.ShowIncominDvcsMsgDvcsMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.ShowIncominDvcsMsgDvcsMenuItem.Text = "Показать
DVCS-ответ";
        this.ShowIncominDvcsMsgDvcsMenuItem.Click += new
System.EventHandler(this.OnShowIncominDvcsMsgDvcsMenuItemClickHa
ndler);
        //
        // ShowIncominDvcsMsgDirMenuItem
        //
        this.ShowIncominDvcsMsgDirMenuItem.Name =
"ShowIncominDvcsMsgDirMenuItem";
        this.ShowIncominDvcsMsgDirMenuItem.Size = new
System.Drawing.Size(301, 22);
        this.ShowIncominDvcsMsgDirMenuItem.Text = "Открыть
каталог";
        this.ShowIncominDvcsMsgDirMenuItem.Click += new
System.EventHandler(this.OnShowIncominDvcsMsgDirMenuItemClickHan
dler);
        //
        // CopyDvcsInMessageIdMenuStrip
        //
        this.CopyDvcsInMessageIdMenuStrip.Name =
"CopyDvcsInMessageIdMenuStrip";
        this.CopyDvcsInMessageIdMenuStrip.Size = new
System.Drawing.Size(301, 22);
        this.CopyDvcsInMessageIdMenuStrip.Text =
"Скопировать идентификатор сообщения";
        this.CopyDvcsInMessageIdMenuStrip.Click += new
System.EventHandler(this.OnCopyDvcsInMessageIdMenuStripClickHand
ler);
        //
        // SendMessageContextMenuStrip
        //
        this.SendMessageContextMenuStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
        this.toolStripMenuItem1,
        this.toolStripMenuItem2});
        this.SendMessageContextMenuStrip.Name =
"CmsIncomingContextMenuStrip";
        this.SendMessageContextMenuStrip.Size = new
System.Drawing.Size(236, 48);
        //
        // toolStripMenuItem1
        //
        this.toolStripMenuItem1.Name = "toolStripMenuItem1";
        this.toolStripMenuItem1.Size = new
System.Drawing.Size(235, 22);
        this.toolStripMenuItem1.Text = "Внешний режим
отправки";

```

```

        this.toolStripMenuItem1.Click += new
System.EventHandler(this.OnSendG2GMessageButtonClickHandler);
        //
        // toolStripMenuItem2
        //
        this.toolStripMenuItem2.Name = "toolStripMenuItem2";
        this.toolStripMenuItem2.Size = new
System.Drawing.Size(235, 22);
        this.toolStripMenuItem2.Text = "Внутренний режим
отправки";
        this.toolStripMenuItem2.Click += new
System.EventHandler(this.OnSendDvcsMessageButtonClickHandler);
        //
        // LoadTestContextMenuStrip
        //
        this.LoadTestContextMenuStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
        this.RunLoadTestG2GModeButton,
        this.RunLoadTestB2GModeButton});
        this.LoadTestContextMenuStrip.Name =
"CmsIncomingContextMenuStrip";
        this.LoadTestContextMenuStrip.Size = new
System.Drawing.Size(270, 48);
        //
        // RunLoadTestG2GModeButton
        //
        this.RunLoadTestG2GModeButton.Name =
"RunLoadTestG2GModeButton";
        this.RunLoadTestG2GModeButton.Size = new
System.Drawing.Size(269, 22);
        this.RunLoadTestG2GModeButton.Text = "Тестирование
внешнего режима";
        this.RunLoadTestG2GModeButton.Click += new
System.EventHandler(this.SpeedTestToolStripMenuItem_Click);
        //
        // RunLoadTestB2GModeButton
        //
        this.RunLoadTestB2GModeButton.Name =
"RunLoadTestB2GModeButton";
        this.RunLoadTestB2GModeButton.Size = new
System.Drawing.Size(269, 22);
        this.RunLoadTestB2GModeButton.Text = "Тестирование
внутреннего режима";
        this.RunLoadTestB2GModeButton.Click += new
System.EventHandler(this.OnRunLoadTestB2GClickHandler);
        //
        // MainForm
        //
        this.AutoScaleDimensions = new
System.Drawing.SizeF(7F, 17F);
        this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;

```

```

        this.BackColor =
System.Drawing.SystemColors.ActiveCaption;
        this.ClientSize = new System.Drawing.Size(1008,
411);

        this.Controls.Add(this.MessageGroupBox);
        this.Controls.Add(this.ControlGroupBox);
        this.Font = new System.Drawing.Font("Segoe UI",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        this.Margin = new System.Windows.Forms.Padding(3, 5,
3, 5);

        this.MinimumSize = new System.Drawing.Size(1024,
450);

        this.Name = "MainForm";
        this.Text = "Сервис формирования электронных
сообщений";

        this.ControlGroupBox.ResumeLayout(false);
        this.ControlGroupBox.PerformLayout();
        this.panel3.ResumeLayout(false);
        this.panel1.ResumeLayout(false);
        this.MessageGroupBox.ResumeLayout(false);
        this.MessageGroupBox.PerformLayout();
        this.GroupBox3.ResumeLayout(false);
        this.GroupBox3.PerformLayout();

((System.ComponentModel.ISupportInitialize) (this.IncomingMessage
sDataGridView)).EndInit();

((System.ComponentModel.ISupportInitialize) (this.OutgoingMessage
sDataGridView)).EndInit();
        this.OutgoingContextMenuStrip.ResumeLayout(false);
        this.IncomingContextMenuStrip.ResumeLayout(false);

this.DvcsOutgoingContextMenuStrip.ResumeLayout(false);

this.DvcsIncomingContextMenuStrip.ResumeLayout(false);

this.SendMessageContextMenuStrip.ResumeLayout(false);
        this.LoadTestContextMenuStrip.ResumeLayout(false);
        this.ResumeLayout(false);

    }

    #endregion
    private System.Windows.Forms.LinkLabel
IncomingMessagesLinkLabel;
    private System.Windows.Forms.LinkLabel
OutgoingMessagesLinkLabel;
    private System.Windows.Forms.GroupBox ControlGroupBox;
    private System.Windows.Forms.GroupBox MessageGroupBox;
    private System.Windows.Forms.DataGridView
IncomingMessagesDataGridView;

```

```

        private System.Windows.Forms.DataGridView
OutgoingMessagesDataGridView;
        private System.Windows.Forms.Label CurrentSegmentLabel;
        private System.Windows.Forms.ComboBox
CurrentSegmentComboBox;
        private System.Windows.Forms.GroupBox GroupBox3;
        private System.Windows.Forms.ContextMenuStrip
OutgoingContextMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
RetrySendToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
ChangeAndRetrySendToolStripMenuItem;
        private System.Windows.Forms.ContextMenuStrip
IncomingContextMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
OpenAsXMLToolStripMenuItem1;
        private System.Windows.Forms.ToolStripMenuItem
CheckSignToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
OpenAsXMLToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
CheckoutSignatureToolStripMenuItem;
        private System.Windows.Forms.Button SettingsButton;
        private System.Windows.Forms.Label ErrorCounterLable;
        private System.Windows.Forms.ToolStripMenuItem
DetailMessageToolStripMenuItem;
        private System.Windows.Forms.LinkLabel DeleteLinkLabel;
        private System.Windows.Forms.Button LoadTestButton;
        private System.Windows.Forms.Button GetMessageButton;
        private System.Windows.Forms.Button NewMessageButton;
        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.Panel panel3;
        private System.Windows.Forms.Panel panel4;
        private System.Windows.Forms.Panel panel2;
        private System.Windows.Forms.ToolTip ToolTip;
        private System.Windows.Forms.ToolStripMenuItem
CopyInDocIdMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
CopyInMessageIdMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
CopyOutDocIdMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
CopyOutMessageIdMenuItem;
        private System.Windows.Forms.ContextMenuStrip
DvcsOutgoingContextMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
ShowDvcsBodyMenuToolItem;
        private System.Windows.Forms.ToolStripMenuItem
ShowDvcsRequestCmsToolItem;
        private System.Windows.Forms.ToolStripMenuItem
OpenSendedDvcsMessageDirMenuItem;

```

```

        private System.Windows.Forms.ContextMenuStrip
DvcsIncomingContextMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
ShowIncominDvcsMsgBodyMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
ShowIncominDvcsMsgDvcsMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
ShowIncominDvcsMsgDirMenuItem;
        private System.Windows.Forms.ContextMenuStrip
SendMessageContextMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
toolStripMenuItem1;
        private System.Windows.Forms.ToolStripMenuItem
toolStripMenuItem2;
        private System.Windows.Forms.ContextMenuStrip
LoadTestContextMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
RunLoadTestG2GModeButton;
        private System.Windows.Forms.ToolStripMenuItem
RunLoadTestB2GModeButton;
        private System.Windows.Forms.ToolStripMenuItem
CopyDvcsOutMessageIdMenuStrip;
        private System.Windows.Forms.ToolStripMenuItem
CopyDvcsInMessageIdMenuStrip;
    }
}

```

Файл «MessageForm.Designer.cs»

```

namespace Views
{
    partial class MessageForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components =
null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources
should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}

```

```

#region Windows Form Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    System.ComponentModel.ComponentResourceManager
resources = new
System.ComponentModel.ComponentResourceManager (typeof (MessageFor
m));

    this.ElectronicDocumentGroupBox = new
System.Windows.Forms.GroupBox();
    this.DocRegNumberLabel = new
System.Windows.Forms.Label();
    this.OrgNameTextBox = new
System.Windows.Forms.TextBox();
    this.DocNameLabel = new
System.Windows.Forms.Label();
    this.PersonNameTextBox = new
System.Windows.Forms.TextBox();
    this.OrgNameLabel = new
System.Windows.Forms.Label();
    this.DocIDTextBox = new
System.Windows.Forms.TextBox();
    this.FaceLabel = new System.Windows.Forms.Label();
    this.DocNameTextBox = new
System.Windows.Forms.TextBox();
    this.FilesGrouBox = new
System.Windows.Forms.GroupBox();
    this.SaveFileLinkLabel = new
System.Windows.Forms.LinkLabel();
    this.FilesListBox = new
System.Windows.Forms.ListBox();
    this.ElectronicDocumentGroupBox.SuspendLayout();
    this.FilesGrouBox.SuspendLayout();
    this.SuspendLayout();
    //
    // ElectronicDocumentGroupBox
    //
    this.ElectronicDocumentGroupBox.Anchor =
System.Windows.Forms.AnchorStyles.None;

    this.ElectronicDocumentGroupBox.Controls.Add (this.DocRegNumberLa
bel);

    this.ElectronicDocumentGroupBox.Controls.Add (this.OrgNameTextBox
);

    this.ElectronicDocumentGroupBox.Controls.Add (this.DocNameLabel);

```

```

this.ElectronicDocumentGroupBox.Controls.Add(this.PersonNameText
Box);

this.ElectronicDocumentGroupBox.Controls.Add(this.OrgNameLabel);

this.ElectronicDocumentGroupBox.Controls.Add(this.DocIDTextBox);

this.ElectronicDocumentGroupBox.Controls.Add(this.FaceLabel);

this.ElectronicDocumentGroupBox.Controls.Add(this.DocNameText
Box);

        this.ElectronicDocumentGroupBox.Font = new
System.Drawing.Font("Segoe UI", 9F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
        this.ElectronicDocumentGroupBox.Location = new
System.Drawing.Point(12, 31);
        this.ElectronicDocumentGroupBox.Name =
"ElectronicDocumentGroupBox";
        this.ElectronicDocumentGroupBox.Size = new
System.Drawing.Size(510, 125);
        this.ElectronicDocumentGroupBox.TabIndex = 10;
        this.ElectronicDocumentGroupBox.TabStop = false;
        this.ElectronicDocumentGroupBox.Text = "Электронный
документ";
        //
        // DocRegNumberLabel
        //
        this.DocRegNumberLabel.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.DocRegNumberLabel.Location = new
System.Drawing.Point(10, 44);
        this.DocRegNumberLabel.Name = "DocRegNumberLabel";
        this.DocRegNumberLabel.Size = new
System.Drawing.Size(234, 23);
        this.DocRegNumberLabel.TabIndex = 2;
        this.DocRegNumberLabel.Text = "Регистрационный номер
документа: ";
        this.DocRegNumberLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
        //
        // OrgNameTextBox
        //
        this.OrgNameTextBox.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
        this.OrgNameTextBox.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.OrgNameTextBox.Location = new
System.Drawing.Point(246, 70);
        this.OrgNameTextBox.Name = "OrgNameTextBox";
        this.OrgNameTextBox.ReadOnly = true;

```

```

        this.OrgNameTextBox.Size = new
System.Drawing.Size(224, 23);
        this.OrgNameTextBox.TabIndex = 7;
        //
        // DocNameLabel
        //
        this.DocNameLabel.Font = new
System.Drawing.Font("Segoe UI", 9F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        this.DocNameLabel.Location = new
System.Drawing.Point(10, 18);
        this.DocNameLabel.Name = "DocNameLabel";
        this.DocNameLabel.Size = new
System.Drawing.Size(234, 23);
        this.DocNameLabel.TabIndex = 1;
        this.DocNameLabel.Text = "Наименование электронного
документа:";
        this.DocNameLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
        //
        // PersonNameTextBox
        //
        this.PersonNameTextBox.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
        this.PersonNameTextBox.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.PersonNameTextBox.Location = new
System.Drawing.Point(246, 96);
        this.PersonNameTextBox.Name = "PersonNameTextBox";
        this.PersonNameTextBox.ReadOnly = true;
        this.PersonNameTextBox.Size = new
System.Drawing.Size(224, 23);
        this.PersonNameTextBox.TabIndex = 8;
        //
        // OrgNameLabel
        //
        this.OrgNameLabel.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.OrgNameLabel.Location = new
System.Drawing.Point(10, 70);
        this.OrgNameLabel.Name = "OrgNameLabel";
        this.OrgNameLabel.Size = new
System.Drawing.Size(234, 23);
        this.OrgNameLabel.TabIndex = 3;
        this.OrgNameLabel.Text = "Наименование организации:
";
        this.OrgNameLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
        //
        // DocIDTextBox
        //

```

```

        this.DocIDTextBox.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
        this.DocIDTextBox.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.DocIDTextBox.Location = new
System.Drawing.Point(246, 44);
        this.DocIDTextBox.Name = "DocIDTextBox";
        this.DocIDTextBox.ReadOnly = true;
        this.DocIDTextBox.Size = new
System.Drawing.Size(224, 23);
        this.DocIDTextBox.TabIndex = 6;
        //
        // FaceLabel
        //
        this.FaceLabel.Font = new System.Drawing.Font("Segoe
UI", 9F);
        this.FaceLabel.Location = new
System.Drawing.Point(10, 95);
        this.FaceLabel.Name = "FaceLabel";
        this.FaceLabel.Size = new System.Drawing.Size(234,
23);
        this.FaceLabel.TabIndex = 4;
        this.FaceLabel.Text = "Должностное лицо: ";
        this.FaceLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
        //
        // DocNameTextBox
        //
        this.DocNameTextBox.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
        this.DocNameTextBox.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.DocNameTextBox.Location = new
System.Drawing.Point(246, 18);
        this.DocNameTextBox.Name = "DocNameTextBox";
        this.DocNameTextBox.ReadOnly = true;
        this.DocNameTextBox.Size = new
System.Drawing.Size(224, 23);
        this.DocNameTextBox.TabIndex = 5;
        //
        // FilesGrouBox
        //
        this.FilesGrouBox.Anchor =
System.Windows.Forms.AnchorStyles.None;

        this.FilesGrouBox.Controls.Add(this.SaveFileLinkLabel);
        this.FilesGrouBox.Controls.Add(this.FilesListBox);
        this.FilesGrouBox.Font = new
System.Drawing.Font("Segoe UI", 9F,
System.Drawing.FontStyle.Bold);
        this.FilesGrouBox.Location = new
System.Drawing.Point(12, 162);

```

```

        this.FilesGrouBox.Name = "FilesGrouBox";
        this.FilesGrouBox.Size = new
System.Drawing.Size(510, 103);
        this.FilesGrouBox.TabIndex = 11;
        this.FilesGrouBox.TabStop = false;
        this.FilesGrouBox.Text = "Вложения";
        //
        // SaveFileLinkLabel
        //
        this.SaveFileLinkLabel.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Ancho
rStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));
        this.SaveFileLinkLabel.AutoSize = true;
        this.SaveFileLinkLabel.Enabled = false;
        this.SaveFileLinkLabel.Font = new
System.Drawing.Font("Segoe UI", 9F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
        this.SaveFileLinkLabel.Location = new
System.Drawing.Point(399, 71);
        this.SaveFileLinkLabel.Name = "SaveFileLinkLabel";
        this.SaveFileLinkLabel.Size = new
System.Drawing.Size(105, 15);
        this.SaveFileLinkLabel.TabIndex = 2;
        this.SaveFileLinkLabel.TabStop = true;
        this.SaveFileLinkLabel.Text = "Выгрузить в файл";
        this.SaveFileLinkLabel.Click += new
System.EventHandler(this.SaveFileButtonClick);
        //
        // FilesListBox
        //
        this.FilesListBox.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
orStyles.Top | System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right)));
        this.FilesListBox.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
        this.FilesListBox.Font = new
System.Drawing.Font("Segoe UI", 9F);
        this.FilesListBox.FormattingEnabled = true;
        this.FilesListBox.ItemHeight = 15;
        this.FilesListBox.Location = new
System.Drawing.Point(6, 19);
        this.FilesListBox.Name = "FilesListBox";
        this.FilesListBox.Size = new
System.Drawing.Size(498, 49);
        this.FilesListBox.TabIndex = 0;
        this.FilesListBox.SelectedValueChanged += new
System.EventHandler(this.FilesListBoxSelectedValueChanged);
        //
        // MessageForm
        //

```

```

        this.AutoScaleDimensions = new
System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode =
System.Windows.Forms.AutoScaleModeMode.Font;
        this.BackColor =
System.Drawing.SystemColors.ActiveCaption;
        this.ClientSize = new System.Drawing.Size(537, 281);
        this.Controls.Add(this.FilesGrouBox);
        this.Controls.Add(this.ElectronicDocumentGroupBox);
        this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
        this.Icon =
((System.Drawing.Icon) (resources.GetObject("$this.Icon")));
        this.MaximizeBox = false;
        this.MinimizeBox = false;
        this.Name = "MessageForm";
        this.Text = "Детали сообщения";
        this.ElectronicDocumentGroupBox.ResumeLayout(false);
        this.ElectronicDocumentGroupBox.PerformLayout();
        this.FilesGrouBox.ResumeLayout(false);
        this.FilesGrouBox.PerformLayout();
        this.ResumeLayout(false);
    }

    #endregion

    private System.Windows.Forms.GroupBox
ElectronicDocumentGroupBox;
    private System.Windows.Forms.Label DocRegNumberLabel;
    private System.Windows.Forms.TextBox OrgNameTextBox;
    private System.Windows.Forms.Label DocNameLabel;
    private System.Windows.Forms.TextBox PersonNameTextBox;
    private System.Windows.Forms.Label OrgNameLabel;
    private System.Windows.Forms.TextBox DocIDTextBox;
    private System.Windows.Forms.Label FaceLabel;
    private System.Windows.Forms.TextBox DocNameTextBox;
    private System.Windows.Forms.GroupBox FilesGrouBox;
    private System.Windows.Forms.ListBox FilesListBox;
    private System.Windows.Forms.LinkLabel
SaveFileLinkLabel;
    }
}

```

Приложение В

(Обязательное)

Сервис формирования и проверки корректности электронных сообщений

Презентация

					<i>ПГУ1.090301.02.001 ПЗ</i>	Лист
						110
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы: «Сервис формирования и проверки корректности электронных сообщений»

Автор работы: студент группы 20ВВП1 Барсуков Н.И.
Научный руководитель: к.т.н. доцент кафедры ВТ ПГУ Федюнин Р.Н.

Пенза 2024

Цель работы

Цель работы – разработать сервис по формированию и проверки корректности электронных сообщений в рамках информационной системы предприятия.

Объектом проекта является приложение, обладающее следующим функционалом:

- формирование сообщения в двух форматах, для внутренней отправки и внешней;
- учет размера сообщения и контроль за его содержимым;
- хранение отправленных и полученных сообщений с возможностью просмотра их содержимого.

					ПГУ1.090301.02.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

Актуальность

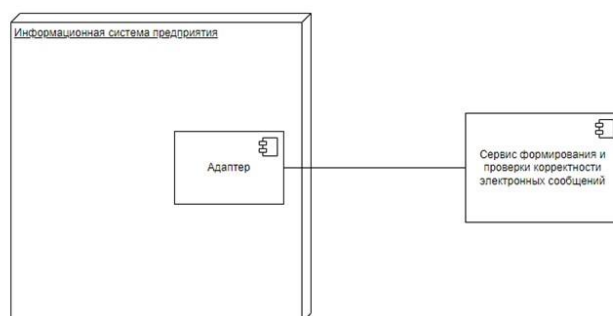
Обмен сообщениями необходим для:

- обеспечения нужд интеграции;
- стандартизации форматов данных;
- возможность асинхронного обмена данными.

Аналоги

Аналог	Поддержка MQ	Удобство использования	Процесс формирования сообщения
<i>JMeter</i>	Да, с использованием сторонних плагинов	Отсутствует интерфейс	Формирует сообщение, может работать со средствами ЭП.
<i>RFHUtil</i>	Да	Очень неинтуитивный интерфейс	Формирование происходит вручную
<i>Thunderbird</i>	Нет	Приемлемый интерфейс	Формирует сообщение, может работать со средствами ЭП.

Общее описание системы



Описание протокола SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
    <soap:Fault>
      ...
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Стек технологий

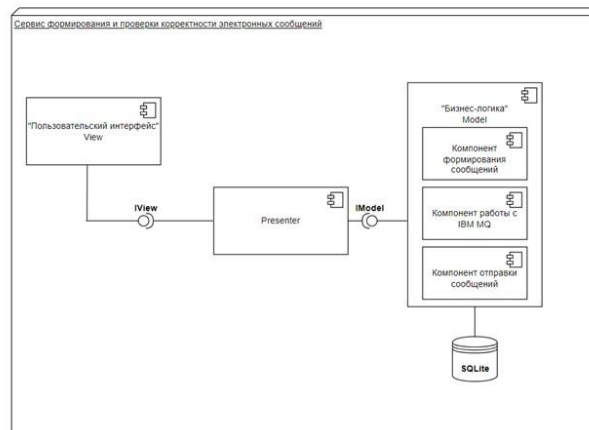


Архитектура приложения

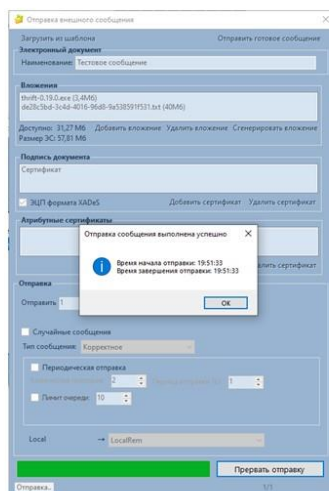
Паттерн MVP



Архитектура приложения



Тестирование приложения



This XML file does not appear to have any style information associated with it. The document tree is shown below.

[illegible]

Результат работы

- Разработан и отлажен пользовательский интерфейс
- Разработан и отлажен процесс формирования электронных сообщений
- Налажено взаимодействие между бизнес-логикой и пользовательским интерфейсом приложения
- Реализовано подключение к удаленной очереди и отправка в неё сообщений

					ПГУ1.090301.02.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		