МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО

«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Информатика и программное обеспечение»

КУРСОВОЙ ПРОЕКТ

по курсу

«Спецификация, архитектура и проектирование программных систем»

Тема: «Проектирование программного комплекса управления надежностью отправки электронных писем»

Документы текстовые

Всего ____ листов в папке

Руководитель
_____ к.т.н., доц. Лагерев Д.Г.
«___»_____2021 г.

Студент
_____ Сухарев Е.А.
«___»_____2021 г.

БРЯНСК 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5

1. АНАЛИЗ ТРЕБОВАНИЙ 6

1.1. Обзор процесса отправки сообщения 6

1.1.1. Общие сведения 6

1.1.2. Основные понятия 7

1.1.3. Процесс отправки сообщения 8

1.1.4. Основные трудности при отправке сообщения 10

1.2. Обзор программ-аналогов 10

1.2.1. Amazon SES 11

1.2.2. Sendgrid 11

1.2.3. Tin-cat email queue 12

1.2.4. Сравнение 13

1.3. Функциональная модель 14

1.3.1. Оператор 14

1.3.2. Администратор 14

1.4. Выводы 15

2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 17

2.1. Введение 17

2.2. Основание для разработки 17

2.3. Назначение и область применения 17

2.4. Требование к программному комплексу 17

2.4.1. Требования к функциональным характеристикам 17

2.4.2. Требования к надежности 20

2.5. Условия эксплуатации 22

2.5.1. Климатические условия эксплуатации 22
2.5.2. Требования к квалификации и численности персонала 22
2.5.3. Требования к составу и параметрам технических средств 22
2.5.4. Требования к информационной и программной совместимости 23
2.6. Программная документация 23
2.6.1. Предварительный состав программной документации 23
2.7. Стадии и этапы разработки 23
2.7.1. Стадии разработки 23
2.7.2. Содержание работ по этапам 24
2.8. Порядок контроля и приемки 24
2.8.1. Виды испытаний 24
2.8.2. Общие требования к приемке работы 25
3. ПРОЕКТИРОВАНИЕ 26
3.1. Проектирование архитектуры 26
3.1.1. Клиентская часть 26
3.1.2. Серверная часть 28
3.1.3. Сервер СУБД 29
3.2. Низкоуровневое проектирование 29
3.2.1. ER-диаграмма 29
3.2.2. IDEF0-Диаграмма 34
3.2.3. BPMN 36
3.3. Проектирование интерфейса 39
4. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ 41
4.1. Тестирование web API методом черного ящика 41

4.2. Модульное тестирование 43

4.3. Кросс-браузерное тестирование пользовательской части 45

ЗАКЛЮЧЕНИЕ 48

СПИСОК ЛИТЕРАТУРЫ 49

ВВЕДЕНИЕ

Информационные технологии получают всё большее развитие в современном мире. Для быстрого обмена информацией существует множество различных способов, и электронная почта до сих пор занимает лидирующие позиции в сфере коммуникации.

Рассылка сообщений включает в себя любое взаимодействие сервиса с пользователем через его электронную почту, например — подтверждение о регистрации или служебные уведомления.

Для отправки и принятия электронных сообщений существует ряд протоколов электронной почты: POP3, IMAP и SMTP.

Как правило, крупные развлекательные сервисы сами не реализуют механизм отправки сообщения через протокол SMTP, а пользуются услугами готовых сервисов рассылки сообщений. К сожалению, большинство почтовых клиентов не предоставляют информации о том, доставлено ли и прочитано ли сообщение клиентов, а также не всегда предоставляют необходимый уровень надежности.

Многие крупные компании используют электронную почту для взаимодействия с партнерами, клиентами, обмениваясь электронными документами. В таких ситуациях крайне важно, чтобы сообщение было успешно доставлено и прочитано получателем. И цена такого недоставленного сообщения крайне высока. Поэтому в таких случаях именно надежность доставки становится во главу угла.

Некоторые почтовые сервисы реализуют специальные предложения для бизнеса, но часто даже в рамках этих предложений улучшается лишь объем трафика и скорость доставки, а надежность так и оставляет желать лучшего.

1. АНАЛИЗ ТРЕБОВАНИЙ

Практически любой IT-проект начинается со сбора информации о деловых процессах, подлежащих автоматизации, среде их функционирования и т.п. Цель этого процесса заключается в том, чтобы все заинтересованные стороны договорились относительно общего толкования конечного результата проекта. Поскольку, участники проекта имеют различный уровень подготовки в области работы с информацией, то и материалы должны быть представлены в разнообразных (но заранее определенных) формах, доступных для понимания разными группами. Другими словами, с разной степенью детализации, уровня абстракции, применения нотаций и других характеристик.

Разработка требований и документирование проекта для сложных продуктов — это длительный процесс, к тому же очень кропотливый и требующий слаженной работы коллектива. Поэтому, с самого начала необходимо продумать и подготовить ландшафт (среду обитания), в которой это процесс будет проистекать. На мой взгляд, обеспечение комфортной среды для работы с артефактами проекта, а также для коммуникации участников между собой – это один из ключевых аспектов его (проекта) успешного завершения.

Анализ требований включает в себя данные, полученные в результате сбора требований к ПО, их систематизации, документирования, анализа, выявления неполноты и разрешения конфликтов.

1.1. Обзор процесса отправки сообщения

1.1.1. *Общие сведения*

Обмен сообщениями является неотъемлемой частью любого взаимодействия. Если важна скорость взаимодействия, чаще всего стороны используют мессенджеры. Электронная почта же из-за своей специфики обязывает вкладывать достаточно смысла в каждое сообщение. Более того, электронная почта является важным звеном в регистрации и защите аккаунтов.

При взаимодействии крупных компаний, передающих важные документы посредством электронной почты, надежность доставки является одним из ключевых параметров. В таких случаях цена недоставленного сообщения может быть крайне высока.

Процесс отправки электронного сообщения разрабатываемым программным комплексом заключается в выборе подходящих сервисов доставки, передаче сообщения этим сервисам и последующей проверке того, доставлено и прочитано сообщение или нет.

1.1.2. ***Основные понятия***

Ниже представлены определения основным понятиям, необходимым в дальнейшей работе.

Сообщение – определенная информация, которую необходимо передать.

Сервис доставки – стороннее ПО, предоставляющее функционал для рассылки сообщений. Чаще всего использование таких сервисов бесплатно до определенного объема трафика.

Отправленное письмо – письмо, переданное сервису доставки.

Получатель – лицо, которому адресовано отправляемое сообщение.

Доставленное письмо – отправленное письмо, полученное получателем и не помеченное как спам.

Отправитель – лицо, запрашивающее отправку сообщения.

Статус доставки – информация о результате доставки сообщения получателю.

Web-интерфейс – веб-страница или несколько веб-страниц, позволяющие взаимодействовать с нужным сайтом, сервисом [1].

API (Application programming interface) – набор функций, описывающих способы взаимодействия с серверной частью разрабатываемого программного комплекса [2].

Электронная цифровая подпись (ЭЦП) – идентификатор документа, полученный в результате криптографического преобразования информации с

использованием закрытого ключа подписи и позволяющий проверить отсутствие искажения информации в электронном документе, а также принадлежность подписи владельцу сертификата ключа подписи [3].

Сертификат ключа подписи – документ, содержащий открытый ключ, информацию о владельце ключа, подтверждающий принадлежность открытого ключа владельцу [4].

1.1.3. ***Процесс отправки сообщения***

Процесс отправки сообщения представлен на рис. 1.1.

Отправка сообщения происходит асинхронно, т.е. отправитель вовлечен в процесс только до момента подтверждения отправки.

На первом шаге отправитель составляет сообщение, может прикрепить к нему какие-либо файлы, выбирает получателя или список получателей, при желании может выбрать предпочтительный сервис отправки или разные сервисы отправки для разных получателей, может отложить отправку на запланированное время, либо указать предпочтительные диапазоны времени отправки. По окончании конфигурирования сообщения пользователь нажимает кнопку отправить.

Далее сервис пытается отправить сообщение получателям посредством выбранных отправителем сервисов или сервисами, выбранными по умолчанию для адресов выбранных получателей. Сервер отдает всю информацию о доставке сервисам доставки и ожидает от них ответ. Если сообщение не доставлено, сообщение передается следующему в очереди сервису доставки, и от него ожидается ответ. Данный цикл продолжается, пока сервер не получит ответ с успешным статусом доставки или пока не перепробует все доступные сервисы доставки.

На заключительном этапе формируется подробная информация о доставке. Эта информация заносится в журнал отправителя.

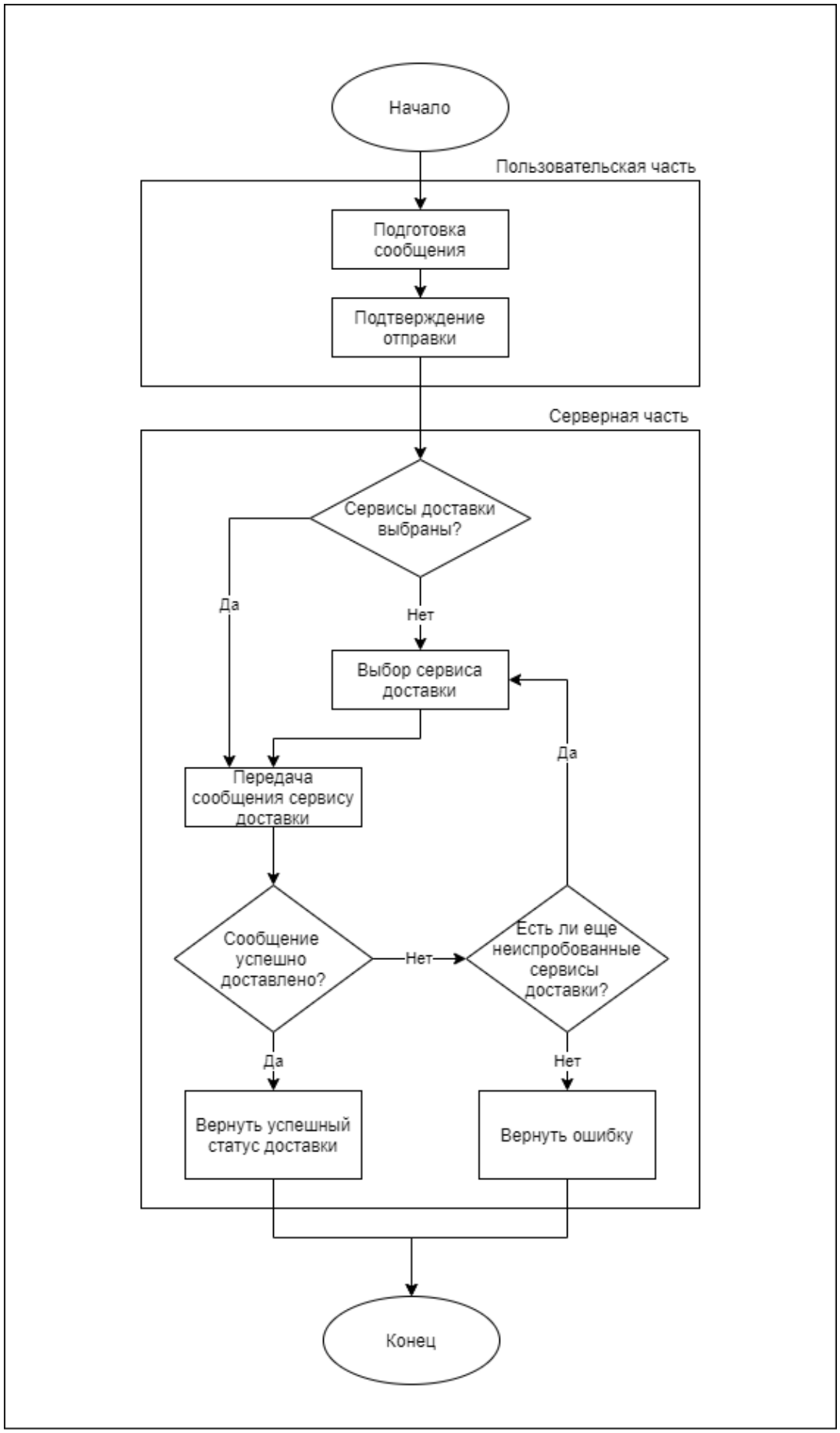

Рис. 1.1. Процесс отправки сообщения

1.1.4. *Основные трудности при отправке сообщения*

Выделим основные, часто встречаемые проблемы, связанные с процессом отправки сообщений.

К техническим проблемам отправки сообщения можно отнести следующие:

- на стороне сервера отправителя (указаны неверные реквизиты, сервер не настроен);
- на стороне сервера получателя (письмо считается спамом; письмо – не спам, но всё равно было отклонено, несуществующий адрес, почтовый ящик получателя переполнен).

Как уже отмечалось выше, сервисы для отправки сообщений предоставляют свой функционал бесплатно до достижения определенного объема трафика (например, определенное количество отправленных сообщений в месяц). Таким образом, для использования некоторых сервисов придется приобретать один из платных тарифов.

1.2. **Обзор программ-аналогов**

В ходе обзора аналогов учитывались следующие функции:

1) отправка сообщения;
2) возможность конфигурирования способа доставки;
3) наличие повторной отправки недоставленного сообщения;
4) наличие и объем информации о доставке.

Полных программ-аналогов найдено не было. Поэтому были рассмотрены сервисы, частично реализующие рассмотренные функции:

1) Amazon Simple Email Service;
2) SendGrid;
3) Tin-cat.

1.2.1. ***Amazon SES***

Amazon Simple Email Service [5] – это экономичный, гибкий и масштабируемый сервис электронной почты, с помощью которого разработчики могут отправлять электронные письма из любого приложения. Amazon SES предоставляет гибкую конфигурацию, в том числе возможность выбрать несколько вариантов использования электронной почты, включая отправку транзакций, маркетинговых писем или выполнение массовой рассылки.

Amazon SES обладает целым рядом сильных сторон:

1) быстрая интеграция;
2) эффективная отправка сообщений;
3) оптимизация доставки;
4) безопасное масштабирование.

Данный сервис широко применяется для отправки мгновенных сообщений в ответ на действия пользователя, например, для подтверждения регистрации или восстановления пароля, а также хорошо подходит для массовых рассылок.

При этом Amazon SES не лишен недостатков:

1) скорость отправки сообщений снижена из-за сложного алгоритма определения пути доставки;
2) относительная ненадежность (в случае неудачной доставки сервис лишь попытается поменять маршрут доставки).

1.2.2. ***Sendgrid***

Sendgrid [6] запущен в 2017 году. Заменяет пользователям индивидуальные почтовые серверы, анализирует репутацию почтовых рассылок, не требует дополнительных затрат для масштабирования инфраструктуры.

Сервис Sendgrid предлагает каждому пользователю стать партнёром:

- партнёрам агентства предоставляются инструменты для управления почтовыми программами клиентов с одной платформы;

- партнёры по рынку, после согласования с администрацией, добавляют собственные инструменты и продают их более чем 60-тысячной «горячей» аудитории;
- OEM-партнёры могут рассылать письма без подписи «via sendgrid.net»;
- партнёры-посредники занимаются перепродажей услуг Sendgrid своим пользователям, экономя собственные ресурсы.

В бесплатной версии сервиса пользователю предоставляют 40 тыс. бесплатных электронных писем на 30 дней. По истечению месяца, можно использовать бесплатную версию сервиса с ограничением 100 писем в день.

Итоговая стоимость услуг зависит от выбранного плана и количества отправленных писем.

Для увеличения производительности можно приобрести дополнительные IP-адреса за дополнительную плату, эта функция доступна и будет полезной только на плане PRO.

Отправлять сообщения можно сразу после подтверждения почты и интеграции с Sendgrid напрямую через SMTP или же с помощью API. У SMTP больше функций, но его сложнее настроить. API рекомендуется большинству пользователей сервиса благодаря простоте кодирования.

Для интеграции через API нужно его сгенерировать, для каждого приложения или сервиса нужен отдельный API.

Важным недостатком Sendgrid является значительная стоимость его использования. Также отмечаются проблемы с нотификацией – SendGrid не запрашивает URL в некоторых случаях. Отмечаются частые обновления в API, а также проблемы в работе поддержки, что иногда приводит к проблемам с обратной совместимостью.

1.2.3. ***Tin-cat email queue***

Tin-cat email queue [7] – система очереди для отправки сообщений. При попытке отправить сообщение, это сообщение отправляется в очередь. При этом

каждую минуту система проверяет наличие сообщений в очереди и отправляет их.

Tin-cat позволяет регулировать частоту проверки очереди, а также количество сообщений, отправляемых за один период (за одну проверку).

Данная система является примером асинхронной отправки сообщений.

Недостатком системы является ее надежность. Если сообщение не будет доставлено, будет осуществлена попытка отправить его тем же способом, что не является эффективным способом. Также стоит отметить недостаточно гибкую конфигурацию данного сервиса (например, невозможность, регулировать интервал отправки сообщений).

1.2.4. ***Сравнение***

При анализе программ-аналогов были выделены ключевые недостатки рассматриваемого ПО, которые необходимо избежать при проектировании разрабатываемого программного комплекса.

Итоговое сравнение программ-аналогов представлено в таблице 1.1.

Знаком «+» обозначены преимущества соответствующих программ-аналогов, а знаком «-» недостатки (в рамках темы курсовой работы).

Таблица 1.1.

Сравнение программ-аналогов

Критерий	Amazon SES	Sendgrid	Tin-cat email queue
Цена	+	-	+
Асинхронная отправка	+	+	+
Гибкость	+	+	-
Надежность	-	-	-

Рассмотренные программы-аналоги, разработанные сторонними компаниями, не реализуют все указанные функции или реализуют их не в полной

мере. Данные ПО позволяют лишь частично контролировать отправку сообщений. Также важным моментом является высокая цена использования некоторых сервисов (таких как Sendgrid), ненадежность API. Важно: если сообщение не удается отправить, данные сервисы не предпринимают попыток отправить его другим способом, что отрицательно сказывается на стабильности рассылки.

1.3. **Функциональная модель**

Так как в программном комплексе существует разделение пользователей по ролям (оператор и администратор), рассмотрим доступный функционал для каждой из ролей.

1.3.1. ***Оператор***

Оператору доступны следующие действия:

1) Отправка сообщения;
2) Просмотр истории сообщений.

При отправке сообщений оператор должен заполнить поле «Получатели». **Дополнительно** он может выбрать сервис доставки, время доставки или период, во время которого необходимо доставить сообщение. Также он может прикрепить к сообщению файлы. Из диаграммы видно, что ввод темы сообщения, а также ввод текста *опциональны*.

Также оператору доступен просмотр сообщений в своём журнале сообщений. Здесь оператор может проверить статус доставки, чтобы убедиться, что сообщение доставлено, и просмотреть время доставки.

1.3.2. ***Администратор***

Администратор наследуется от оператора, т.е. ему доступен весь доступный оператору функционал.

Отличие администратора от оператора заключается в возможности просмотра списка операторов, где администратор может добавить или удалить оператора.

Составлена диаграмма вариантов использования, представленная на рис. 1.2.

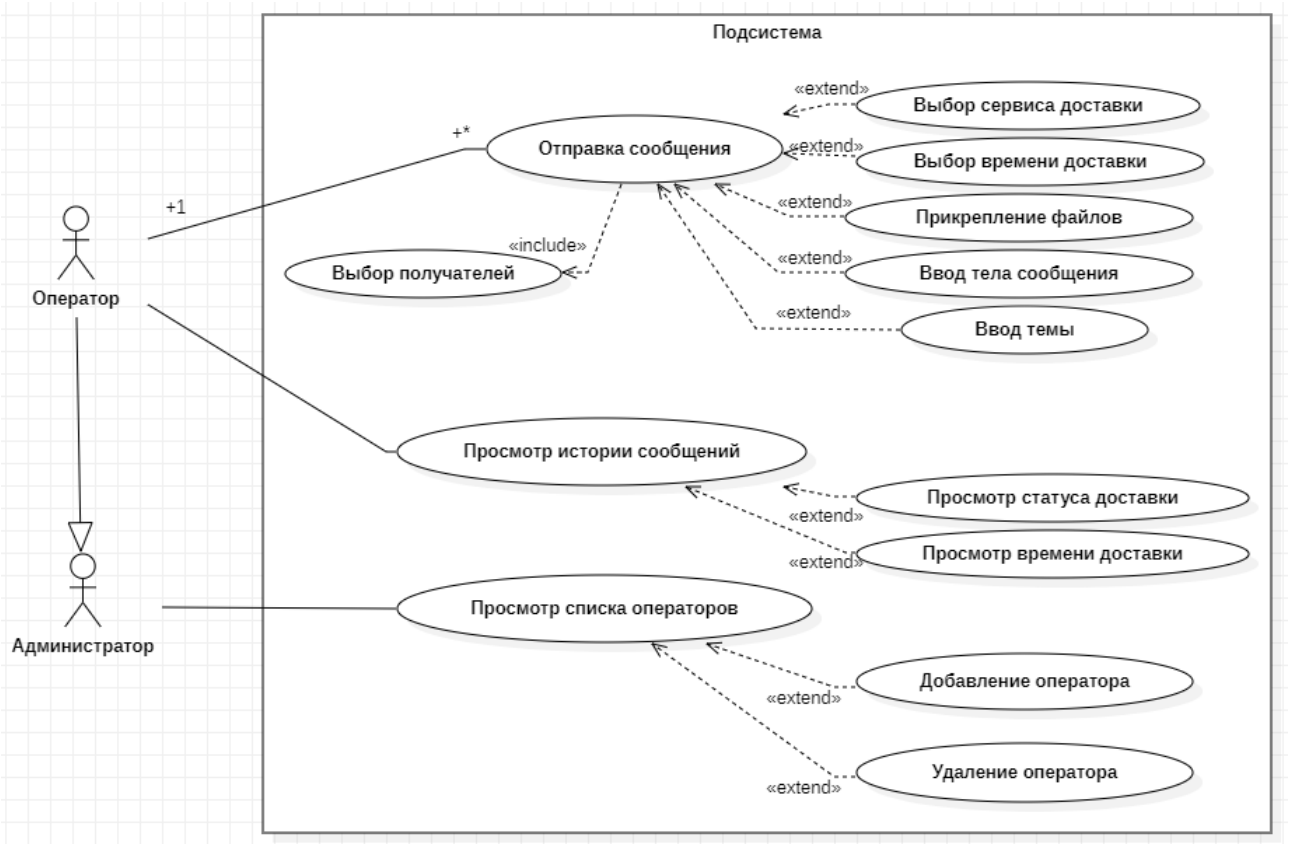

Рис. 1.2. Диаграмма вариантов использования

1.4. **Выводы**

Тема работы: «Проектирование программного комплекса управления надежностью отправки электронных писем».

Тема курсовой работы актуальна в виду того, что сервисы рассылки электронных сообщений не лишены разноплановых недостатков, поэтому выбор подходящего сервиса становится нетривиальной задачей.

Так как главным критерием является надежность, а все рассмотренные программы-аналоги не предоставляют требуемый уровень надежности доставки сообщений, можно сделать вывод о необходимости разработки программного продукта, который должен решать все поставленные задачи.

Целью курсовой работы является проектирование программного комплекса управления рассылкой электронных сообщений с возможностью последующего внедрения в системы, требующие повышенной надежности отправки.

Поставленная цель достигается путем решения следующих основных задач:

1) анализ процесса отправки электронного письма;
2) сравнительный анализ уже имеющихся систем и платформ для обучения;
3) разработка и анализ требований;
4) проектирование программного комплекса;
5) программная реализация базы данных, API, серверной части, WEB-интерфейса;
6) тестирование разработанного программного комплекса;
7) внедрение в существующую информационную систему.

Объектом исследования является процесс рассылки электронных сообщений.

Предметом исследования в курсовой работе являются методы и средства улучшения надежности и контроля рассылки электронных сообщений.

2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1. **Введение**

Техническое задание (ТЗ) является исходным материалом для создания программного комплекса. ТЗ устанавливает назначение разрабатываемого программного комплекса, его технические характеристики, показатели качества и технико-экономические требования, предписанные по выполнению необходимых стадий создания документации и её состав, а также специальные требования.

2.2. **Основание для разработки**

Основанием для разработки программного комплекса управления надежностью отправки электронных писем является задание на курсовую работу.

2.3. **Назначение и область применения**

Разрабатываемый программный комплекс должен выполнять следующие задачи:

- асинхронная рассылка сообщений выбранным адресам с помощью выбранных сервисов доставки;
- обеспечение повышенной надежности доставки;
- возможность получения и просмотра подробной информации о статусе доставки сообщения.

2.4. **Требование к программному комплексу**

2.4.1. ***Требования к функциональным характеристикам***

Разрабатываемый программный комплекс должен включать в себя нижеупомянутые функциональные особенности:

1. Авторизация пользователей в web-интерфейсе для администрирования:

Роли:

1) оператор – может управлять рассылкой сообщений, просматривать соответствующую историю сообщений;
2) администратор – имеет все возможности оператора, но также может просматривать список операторов, создавать новых операторов и удалять существующих.

2. После входа в web-сервис пользователь может:
   1) отправлять сообщения на электронные почтовые ящики;
   2) выбирать сервисы доставки из списка доступных на данный момент;
   3) прикреплять файлы к сообщению;
   4) организовывать отправку сообщений по расписанию;
   5) просматривать подробную информацию о доставке сообщений;
   6) управлять доступом других пользователей (только для администратора).
3. Разрабатываемый программный комплекс должен предоставлять следующий уровень надежности:
   - если хотя бы один из доступных сервисов доставки способен доставить указанное сообщение указанному адресату, то это сообщение должно быть доставлено.
4. Разрабатываемый программный комплекс должен обладать следующими функциями:
   1) возможность повторной отправки недоставленного сообщения, но уже посредством другого доступного сервиса доставки при наличии Интернет-соединения;
   2) возможность отправлять сообщения посредством минимум **трех** различных сервисов для отправки сообщений;
   3) наличие API, с помощью которого программный комплекс можно будет встраивать в различные сервисы, использующие рассылку сообщений;

4) возможность прикреплять файлы к сообщению совокупным объемом не более 30МБ;
5) возможность хранить историю сообщений, т.е. хранить определенное число ранее отправленных сообщений и давать доступ к просмотру информации об их доставке;
6) возможность получения обратной связи - подсистема должна оповещать пользователя об удачной доставке или о причине неудачной отправки сообщения (*опционально:* уведомлять о прочтении доставленного сообщения), хранить эту информацию и давать возможность ее просматривать.

5. Разрабатываемый комплекс предоставляет асинхронную отправку сообщений.

Когда оператор или администратор отправляет сообщение, сообщение отправляется в очередь на удаленном сервере. Система с настраиваемой периодичностью проверяет эту очередь и передает сообщения соответствующим сервисам доставки.

6. В целях улучшения надежности разрабатываемый комплекс должен включать в себя кластерную систему серверов.
7. Требования к интерфейсу.

Разрабатываемый программный комплекс должен включать в себя нижеупомянутые функциональные особенности:

1) интерфейс для работы с подсистемой (интерфейс администратора) – web-страница, включает в себя список со всей информацией об отправленных сообщениях, предоставляет возможность вручную отправить сообщение через выбранный сервис, настраивать почтовые шлюзы;
2) необходимо обеспечить совместимость с основными браузерами (последние версии Chrome, Firefox, Safari, Opera; IE начиная с 10 версии). Верстка должна быть адаптивной и рассчитана на минимальное разрешение экрана 1280×720;

3) цветовая гамма приложения не должна быть излишне яркой, но сочетания цветов должны быть контрастными. Дизайн должен быть простым и понятным, нужно избегать непонятных иконок. Списки, с помощью которых также может быть реализовано меню, не должны превышать 5-9 элементов;
4) если время загрузки страницы составляет более 5с., необходимо обеспечить пользователю уверенность в том, что процесс действительно происходит при помощи индикатора выполнения процесса;
5) на длинных страницах необходимо применять ссылки, возвращающие пользователя в верхнюю часть страницы.

Получение данных. Система получает на вход:

- тему;
- тело сообщения;
- прикрепленные файлы;
- список адресов, на которые нужно отправить сообщения;
- список сервисов доставки, посредством которых необходимо доставить сообщение;
- запланированное время доставки или диапазоны времени, в пределах которых сообщение должно быть доставлено.

Выходные данные. Система предоставляет пользователю историю сообщений с информацией о доставке.

Отправка данных. Система обрабатывает полученные от пользователя данные через REST API и передает их указанным сервисам доставки.

2.4.2. ***Требования к надежности***

1. Требования к обеспечению надежного функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1) организацией бесперебойного питания технических средств;
2) использованием лицензионного программного обеспечения;
3) регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
4) регулярным выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов.

2. Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 30-ти минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

3. Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с системой.

Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу оператора без предоставления ему административных привилегий, что реализовано посредством разделения ролей.

2.5. **Условия эксплуатации**

2.5.1. ***Климатические условия эксплуатации***

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

2.5.2. ***Требования к квалификации и численности персонала***

Минимальное количество персонала, требуемого для работы программного комплекса, должно составлять не менее 1 человека: администратор, управляющий настройкой подсистемы.

2.5.3. ***Требования к составу и параметрам технических средств***

Программный комплекс функционирует в составе уже имеющегося сервиса и не предъявляет дополнительных требований к составу и параметру технических средств. Подсистема должна работать на тех же самых устройствах, что и вся система учета.

Сервер приложения:

- наличие веб-сервера IIS.
- процессор: 3,1 ГГц и выше;
- ОЗУ: 16Гб и выше;
- SSD диск.

Сервер баз данных:

- процессор: 3,1 ГГц и выше;
- ОЗУ: 16Гб и выше;
- SSD диск.

Клиент:

- процессор Intel Pentium 4 или более поздней версии с поддержкой SSE3;

- ОЗУ: 1Гб и выше.

2.5.4. *Требования к информационной и программной совместимости*

В перечень программного обеспечения, работающего на стороне сервера входит следующее ПО:

- лицензионная версия операционной системы «Microsoft Windows Server 2019»;
- базы данных «Microsoft SQL Server 2019»;
- СУБД «SQL Server Management Studio»;
- web-браузер Google Chrome 89.0 или Safari 14, или Mozilla Firefox 91.6 ESRЮ, или Opera 12;
- антивирусное программное обеспечение.

Дополнительные требования к защите программного обеспечения и информации не предъявляются.

2.6. **Программная документация**

2.6.1. *Предварительный состав программной документации*

Состав программной документации:

- техническое задание;
- программа и методики испытаний;
- руководство оператора.

2.7. **Стадии и этапы разработки**

2.7.1. *Стадии разработки*

Разработка подсистемы состоит из следующих этапов:

- разработка технического задания;
- проектирование программного комплекса;
- разработка программного комплекса;

- тестирование разработанного программного комплекса;
- написание программной документации;
- внедрение разработанного сервиса.

2.7.2. *Содержание работ по этапам*

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- постановка задачи;
- определение и уточнение требований к техническим средствам;
- определение требований к программе;
- определение стадий, этапов и сроков разработки программы;
- согласование и утверждение технического задания.

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

Этап тестирования включает в себя выполнение тестов по указанным заранее шагам и сверку полученных результатов с указанными заранее ожидаемыми результатами.

Этап внедрения подразумевает развертывание разработанного программного комплекса на выделенном сервере с дальнейшим приведением подсистемы в стабильно работающее состояние. Результатом данного заключительного этапа является подписанный Заказчиком акт о внедрении.

2.8. **Порядок контроля и приемки**

2.8.1. *Виды испытаний*

Для проверки корректности работы разработанного программного комплекса должны быть проведены следующие виды испытаний: тестирование web API методом черного ящика, модульное тестирование, кросс-браузерное тестирование.

В результате проведенного тестирования методом черного ящика и модульного тестирования все установленные заранее тесты должны быть пройдены успешно.

Кросс-браузерное тестирование должно подтвердить идентичное поведение разработанного программного комплекса в указанных в разделе «Требования к информационной и программной совместимости» браузерах.

Приемо-сдаточные испытания должны проводиться на объекте Заказчика в оговоренные сроки.

Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель документируют в Протоколе проведения испытаний

2.8.2. ***Общие требования к приемке работы***

На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывает акт о внедрении.

3. ПРОЕКТИРОВАНИЕ

3.1. **Проектирование архитектуры**

Поставлена задача: **разработать архитектуру для программного комплекса управления рассылкой электронных сообщений.**

В процессе проектирования было принято следующее: система должна быть централизованной, т.е. все данные должны располагаться в центральном хранилище. Система должна иметь трехуровневую архитектуру, состоящую из следующих уровней: первый - клиент, второй - сервер, третий – хранилище в связи с ее масштабируемостью, гибкой настройкой, высокой безопасностью и надежностью, а также наличием большого объема документации.

Схема архитектуры ПС представлена на рис. 3.1.

3.1.1. ***Клиентская часть***

В процессе анализа было рассмотрено несколько альтернативных инструментов разработки web-интерфейса:

a) React;
b) Angular;
c) Vue.

Среди выбранных фреймворков предпочтение было отдано Angular благодаря его декларативности, наличию развитого сообщества, модульности и MVC из коробки [17]. В качестве скриптового языка из-за ряда преимуществ над JavaScript был выбран TypeScript [19]. Для вёрстки будет применяться следующий набор инструментов: HTML + CSS [18], а также адаптивные компоненты Material и некоторые стили Bootstrap. Bootstrap и Material уже входят в Angular, поэтому отдельно подключать их не требуется.

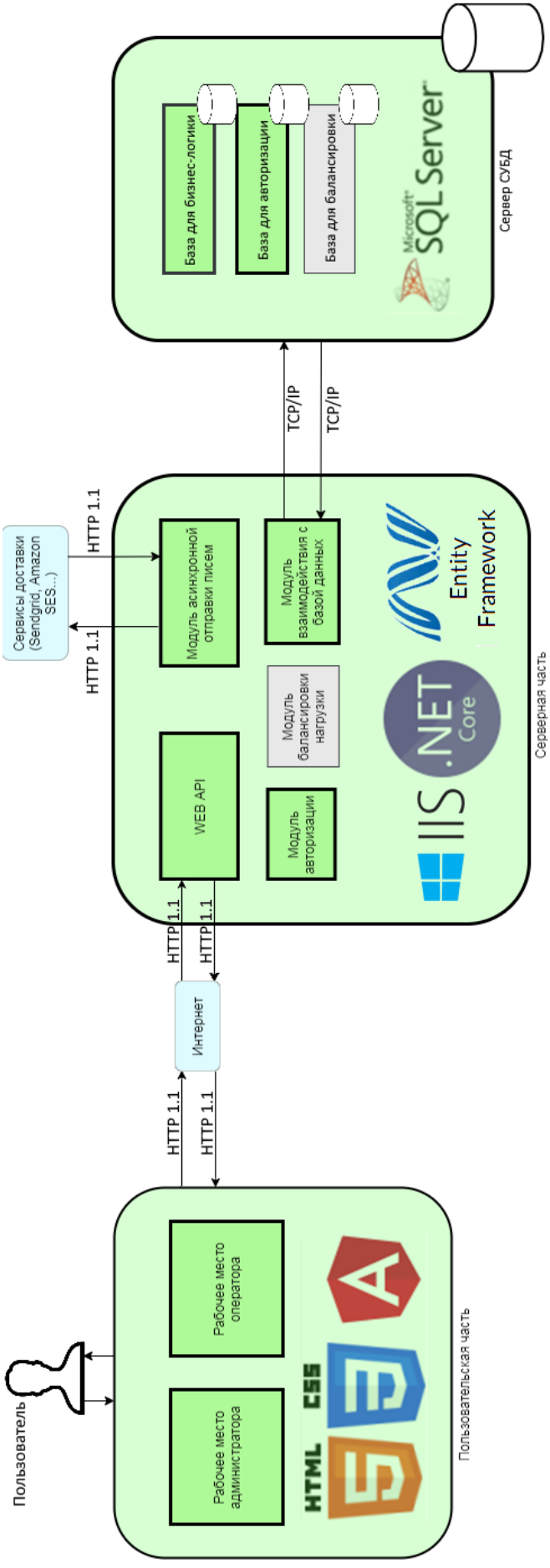

Рис. 3.1. Схема архитектуры ПС

3.1.2. ***Серверная часть***

Серверная часть включает в себя API, используемый для обработки запросов от клиентской части, набор функций для взаимодействия с хранилищем данных, модуль асинхронной отправки писем.

Серверная часть условно поделена на ряд модулей:

a) web API;
b) модуль авторизации;
c) модуль балансировки нагрузки;
d) модуль асинхронной отправки писем;
e) модуль взаимодействия с базой данных.

Web API служит для взаимодействия пользовательской части и серверной части. Он включает в себя методы получения, добавления, изменения и удаления пользователей, сообщений и других сущностей [8].

Модуль авторизации отвечает за процедуру подтверждения подлинности конкретного человека, блокировку пользователей, а также за управление ролями зарегистрированных в системе пользователей.

Модуль балансировки нагрузки обеспечивает корректную работу кластерной системы серверов: координирует работу нескольких серверов, позволяет проверять их работоспособность и эффективность их работы, назначать определенным единицам выбранные роли, собирает статистику запросов, позволяет задавать подходящий метод распределения нагрузки.

Модуль асинхронной отправки писем служит для взаимодействия с доступными сервисами доставки и организации отправки писем.

Модуль взаимодействия с базой данных предоставляет интерфейсы для взаимодействия с таблицами сущностей. Для разработки данного модуля был выбран Entity Framework Core.

В качестве инструмента разработки серверной части была выбрана технология .NET Core, благодаря ее кроссплатформенности и удобству языка C#, а также благодаря детальной технической документации от официального разработчика [9].

3.1.3. ***Сервер СУБД***

В ходе работы был проанализирован ряд различных СУБД, а именно:

a) MySQL;
b) MS SQL;
c) PostgreSQL.

Предпочтение было отдано MS SQL в связи с его масштабируемостью и надежностью, высокой скоростью создания решений, а также возможностью обработки вычислений в оперативной памяти (in-memory OLTP) [22].

В базе данных будут храниться все основные сущности системы:

a) сообщения;
b) пользователи;
c) ссылки на прикрепленные файлы;
d) контакты пользователей;
e) сервисы доставки;
f) очереди доставки;
g) доступы пользователей;
h) тарифы;
i) скидки.

Также для осуществления отношения «многие ко многим» необходимы промежуточные таблицы для взаимодействия следующих таблиц:

- Сообщения – Файлы;
- Пользователи – Доступы;
- Тарифы – Скидки.

3.2. **Низкоуровневое проектирование**

3.2.1. ***ER-диаграмма***

Разработана модель данных для программного комплекса управления рассылкой электронных сообщений.

Оператор передает информацию о сообщении в пользовательской части и при помощи API передает его на сервер, где информация о сообщении формализуется, заносится в базу данных и передается сервисам доставки.

В результате анализа процесс обучения можно обозначить несколько сущностей. Вот описание некоторых основных сущностей:

- Messages – таблица для хранения сообщений
    - id (long) – суррогатный первичный ключ;
    - userId (long) – id отправителя;
    - destinationDate (dateTime) – дата и время успешной доставки;
    - theme (nvarchar(max)) – тема сообщения;
    - body (text) – тело сообщения;
    - size (int) – размер сообщения в единицах (байт);
    - isScheduled (int) – логическое поле – запланирована ли отправка сообщения;
    - scheduleDate (dateTime) – запланированная дата и время, когда необходимо отправить сообщение;
    - isSent (int) – логическое поле – успешно ли доставлено сообщение;
    - queueId (int) – id очереди, в которой содержится данное сообщение.
- MessagesAttachedFiles – таблица для связи таблиц сообщений и прикрепленных файлов
    - messageId (long) – суррогатный первичный ключ сообщения;
    - attachedFileId (long) – суррогатный первичный ключ файла.
- AttachedFiles – таблица для хранения ссылок на прикрепленные файлы
    - attachedFileId (long) – суррогатный первичный ключ;
    - line (nvarchar(max)) – ссылка на файл, хранящийся на сервере;
    - size (int) – размер файла, на который указывает ссылка в поле link;
    - typeId – тип файла, на который указывает ссылка в поле link.
- Users – таблица для хранения информации о пользователях

    - id (long) – суррогатный первичный ключ;
    - userName (nvarchar(max)) – имя пользователя;
    - email (nvarchar(max)) – почтовый адрес пользователя;
    - passwordHash (nvarchar(max)) – хэш, полученный из пароля пользователя;
    - isActive (int) – логическое поле – статус доступа пользователя;
    - queueId (int) – id очереди, в которую помещаются сообщения данного пользователя.
- UsersAccesses – таблица для связи таблиц пользователей и доступов
    - userId (long) – суррогатный первичный ключ пользователя;
    - accessId (long) – суррогатный первичный ключ доступа
- Accesses – таблица для хранения информации о доступах
    - id (long) – суррогатный первичный ключ;
    - userId (long) – id пользователя, которому принадлежит данный доступ
    - startDate (dateTime) – дата и время начала подписки пользователя;
    - endDate (dateTime) – дата и время конца подписки пользователя;
    - tariffId (int) – id тарифа пользователя.
- Contacts – таблица для хранения контактов пользователей
    - id (long) – суррогатный первичный ключ;
    - contactName (nvarchar(MAX)) – имя контакта;
    - contactEmail (ncarchar(MAX)) – электронный адрес контакта;
    - userId (long) – id пользователя, которому принадлежит эта запись.
- DeliveryServices – таблица для хранения информации о сервисах доставки
    - id (int) – суррогатный первичный ключ;
    - serviceName (nvarchar(max)) – название сервиса доставки;
    - standartPriority (int) – значение приоритета данного сервиса доставки при переборе списка сервисов во время отправки по умолчанию;

    - connectionString (nvarchar(max)) – строка, необходимая для взаимодействия с данным сервисом.
- DeliveryQueues – таблица для хранения информации об очередях
    - id (int) – суррогатный первичный ключ;
    - sendingIntervalSec (int) – интервал отправки сообщений.
- Tariffs – таблица для хранения информации о тарифах
    - id (int) – суррогатный первичный ключ;
    - tariffName (nvarchar(255)) – название тарифа;
    - cost (int) – стоимость тарифа.
- TariffsSales – таблица для связи таблиц тарифов и скидок
    - tariffId (int) – суррогатный первичный ключ тарифа;
    - saleId (int) – суррогатный первичный /ключ скидки.
- Sales – таблица для хранения информации о скидках
    - id (int) – суррогатный первичный ключ;
    - name (nvarchar(255)) – название скидки на тариф;
    - value (int) – размер скидки на тариф.

ER диаграммы применяются для моделирования и проектирования реляционных баз данных, причем как в плане логических и бизнес-правил (логические модели данных), так и в плане внедрения конкретных технологий (физические модели данных). В сфере разработки программного обеспечения ER-диаграмма, как правило, служит первым шагом в определении требований проекта по созданию информационных систем.

Спроектированная ER диаграмма находится в четвертой нормальной форме, так как находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей [10]. ER диаграмма представлена на рис. 3.2.

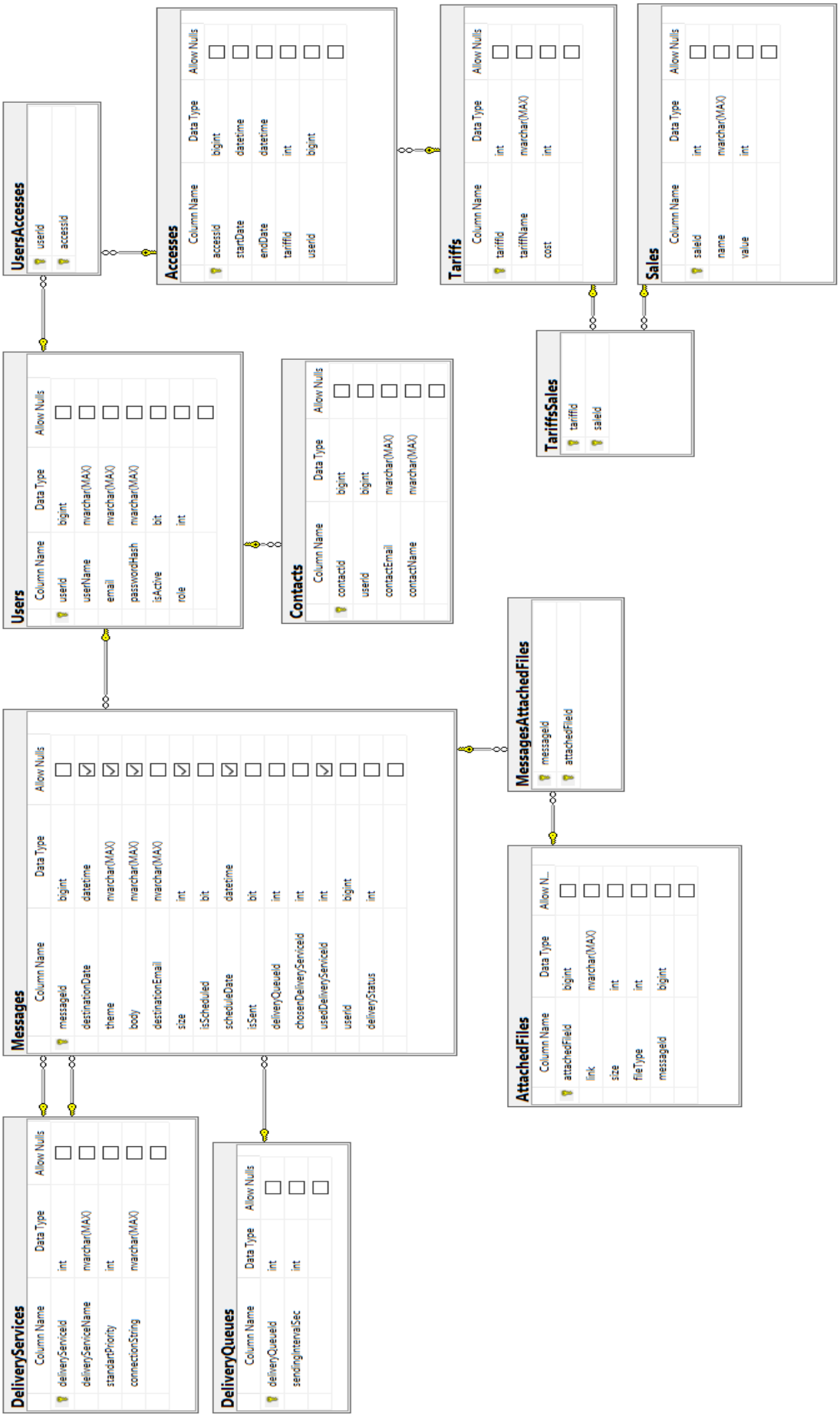

Рис. 3.2. ER диаграмма

3.2.2. *IDEF0-Диаграмма*

В рамках проектирования данной диаграммы были определены функциональные требования к программной системе в рамках процесса отправки сообщения.

На контекстной диаграмме (рис. 3.3) определен функциональный блок «Отправить сообщение через систему», механизм, предписание, входные и выходные данные.

На диаграмме первого уровня (рис. 3.4) выполнена декомпозиция базового блока «Отправить сообщение через систему». Этот блок содержит четыре подфункции: «Сформировать письмо», «Инициировать отправку сообщения», «Занести информацию о доставке в журнал». Проведена детализация функций «Сформировать письмо» (рис. 3.5) и «Инициировать отправку сообщения» (рис. 3.6)

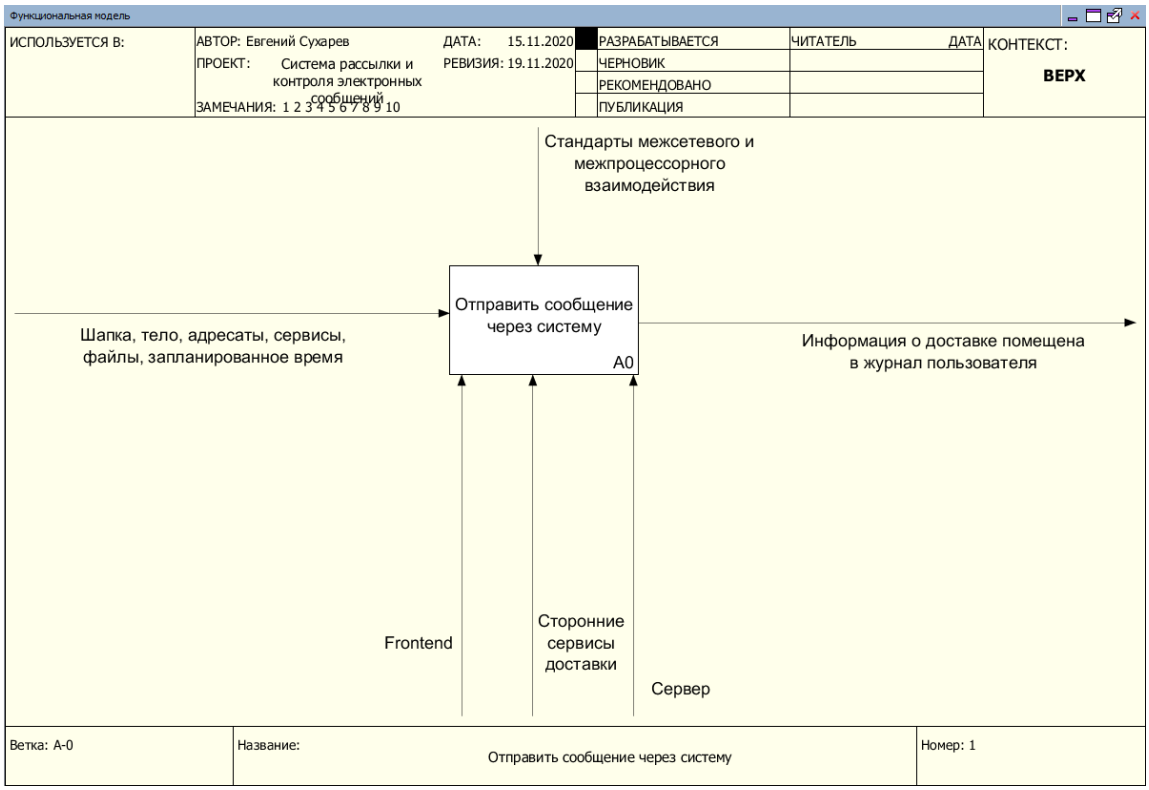

Рис. 3.3. Контекстная диаграмма функциональной модели «Поиск объектов»

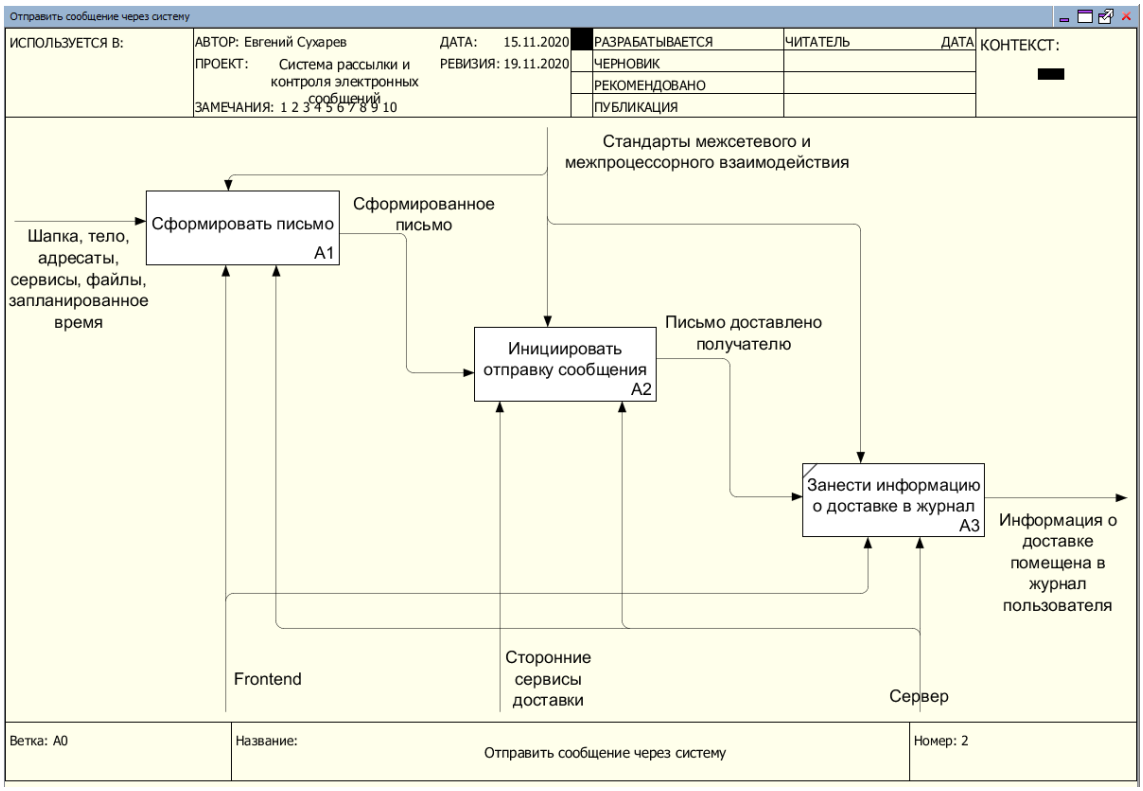

Рис. 3.4. Диаграмма первого уровня функциональной модели

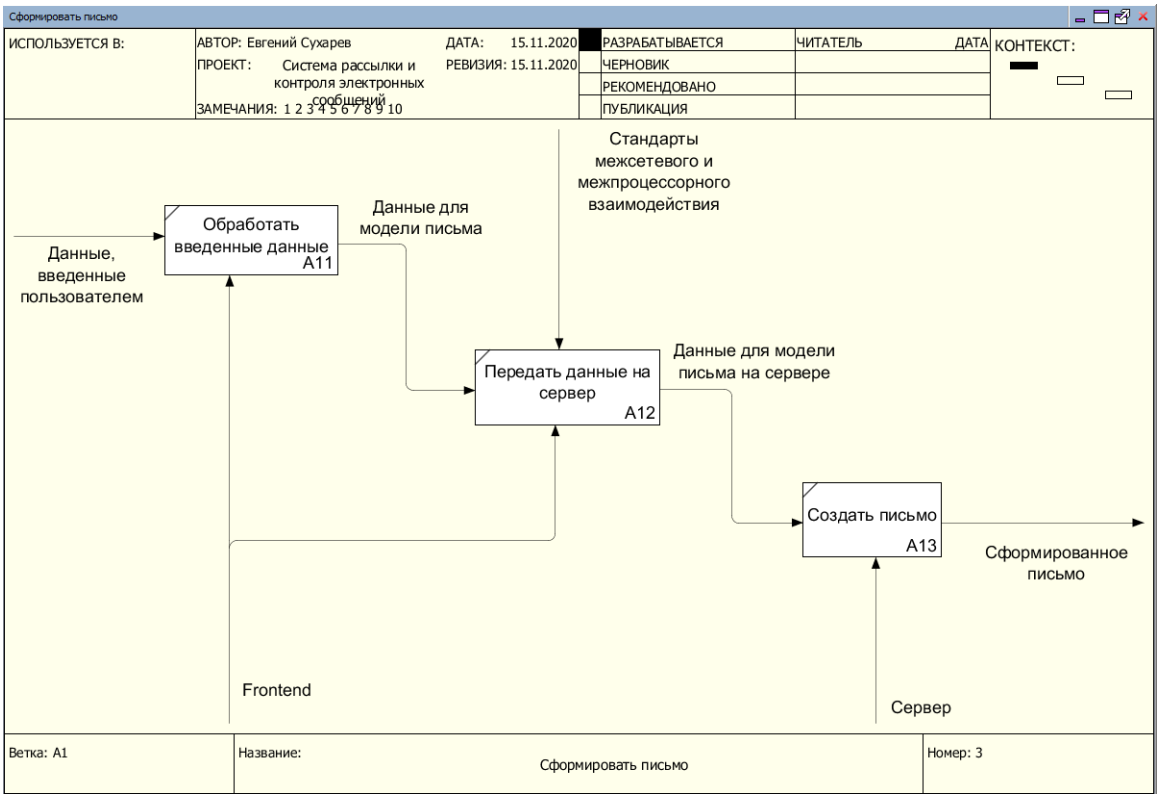

Рис. 3.5. Диаграмма второго уровня для блока «Сформировать письмо»

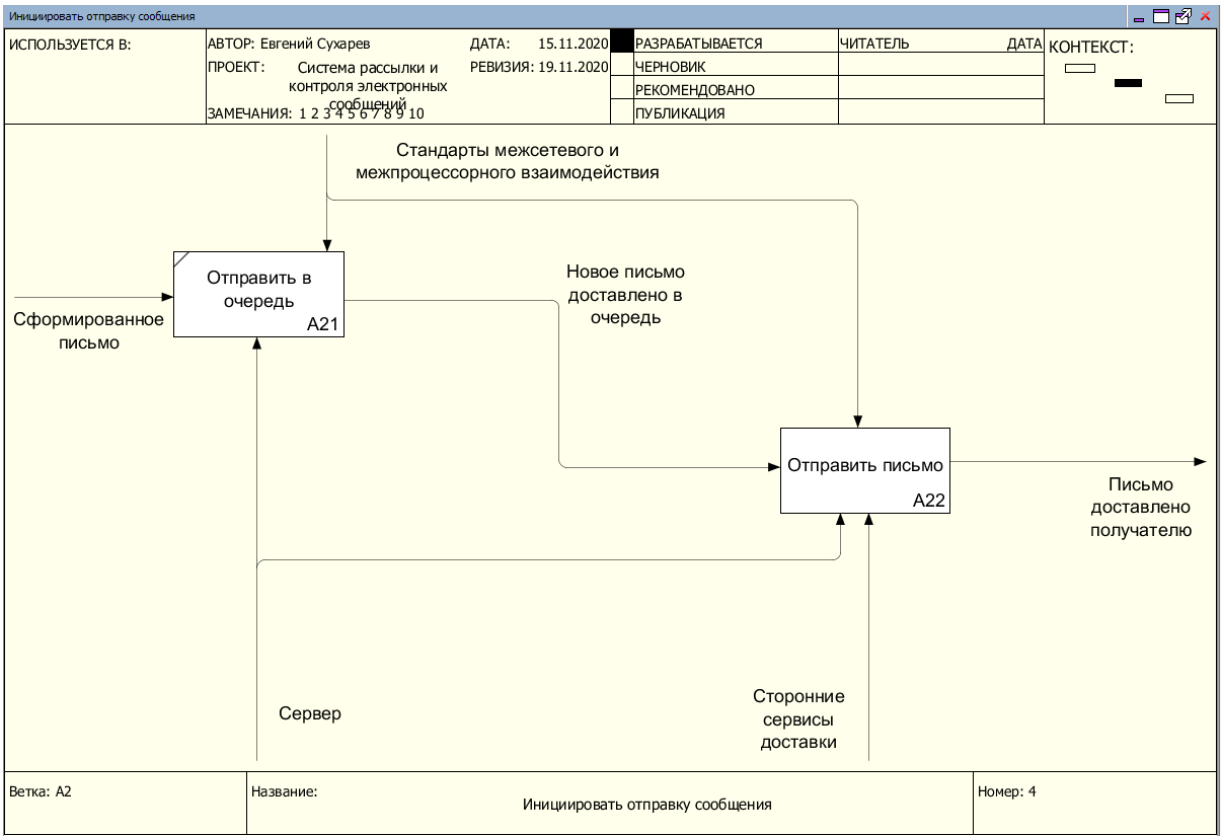

Рис. 3.6. Диаграмма второго уровня для блока «Инициировать отправку сообщения»

3.2.3. ***BPMN***

Поставлена задача: разработать модель бизнес-процесса для программного комплекса управления надежностью отправки электронных писем**.**

В качестве бизнес-процесса был рассмотрен процесс отправки сообщения.

В данном процессе участвуют четыре исполнителя:

1. Пользователь как инициатор процесса.
2. Клиентская часть, обрабатывающая введенные пользователем данные и отправляющая запрос на сервер.
3. Серверная часть принимает запрос от клиентской части, формирует письмо, заносит его в базу данных и отправляет письмо в очередь.
4. Сторонние сервисы доставки.

Клиентская часть обрабатывает введенные пользователем данные и отправляет запрос на сервер. Также клиентская часть получает ответ от сервера и представляет пользователю обратную связь в удобном виде.

Серверная часть принимает запрос от клиентской части, формирует письмо, заносит его в базу данных и отправляет письмо в очередь. Когда очередь доходит до данного письма, происходит выбор сервера доставки и передача письма выбранному сервису.

При получении неудовлетворительного ответа предпринимается попытка заново отправить письмо, но посредством других доступных сервисов доставки.

Диаграмма модели бизнес-процесса представлена на рис. 3.7.

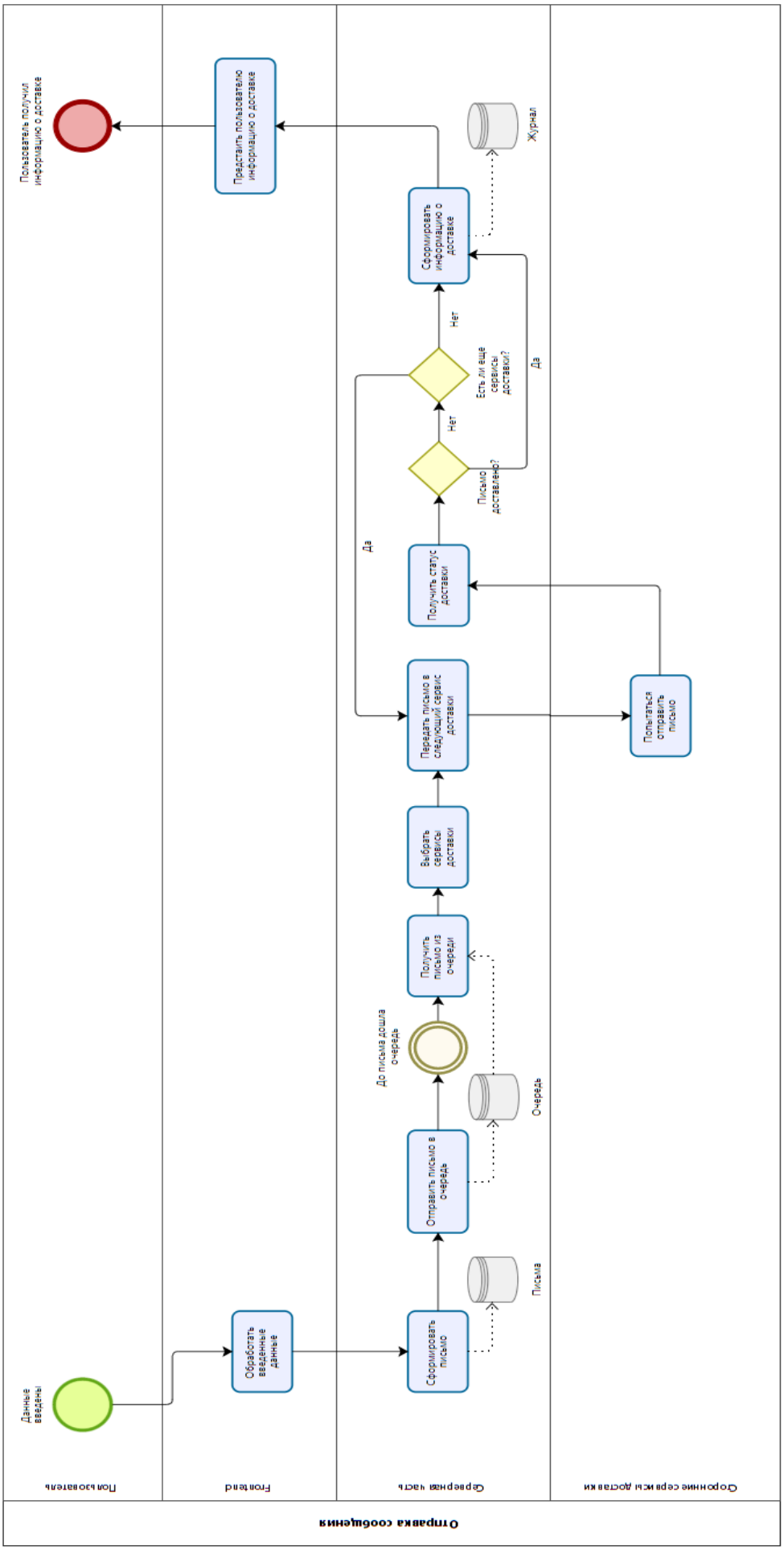

Рис. 3.7. Диаграмма модели бизнес-процесса

3.3. **Проектирование интерфейса**

При проектировании интерфейса должны быть учтены следующие критерии эргономики:

1) скорость работы пользователя.
2) количество человеческих ошибок.
3) скорость обучения.
4) субъективная удовлетворенность пользователя.

Макеты основных интерфейсов представлены на рис. 3.8-3.10.

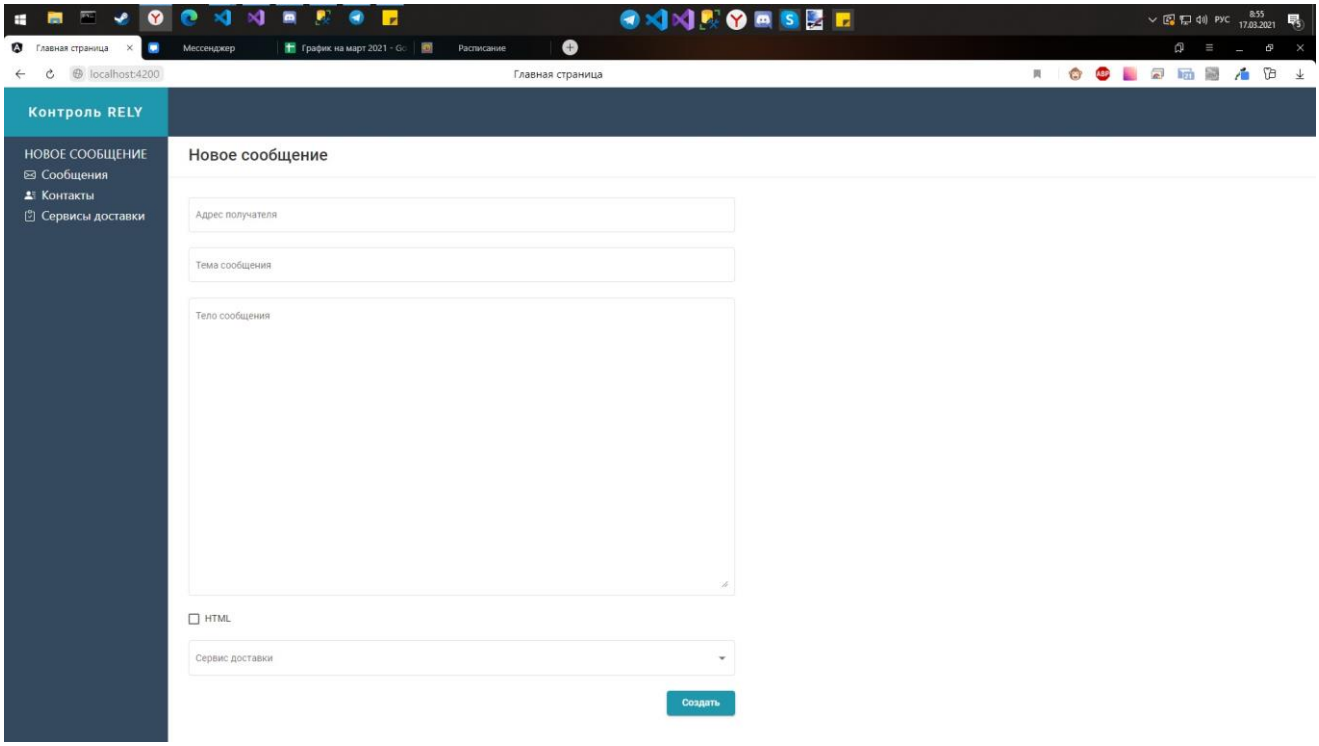

Рис. 3.8. Форма создания письма

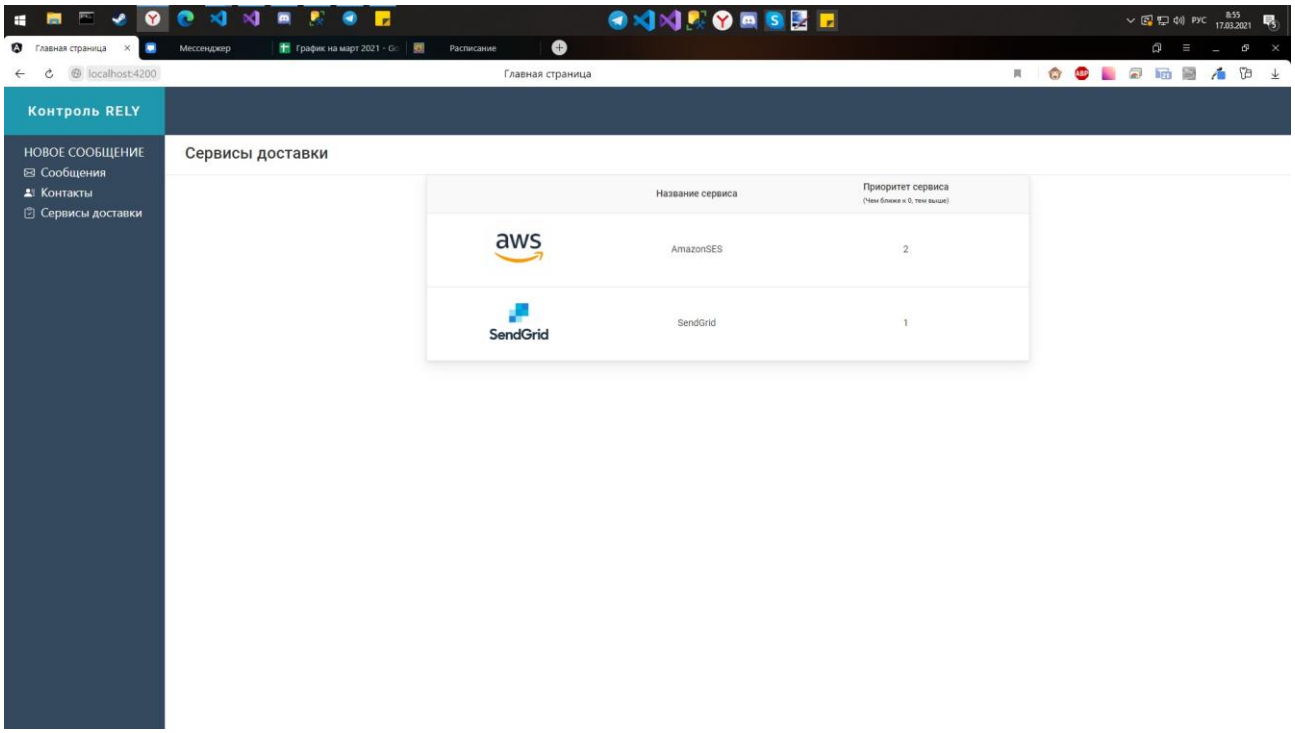

Рис. 3.9. Страница с доступными сервисами доставки

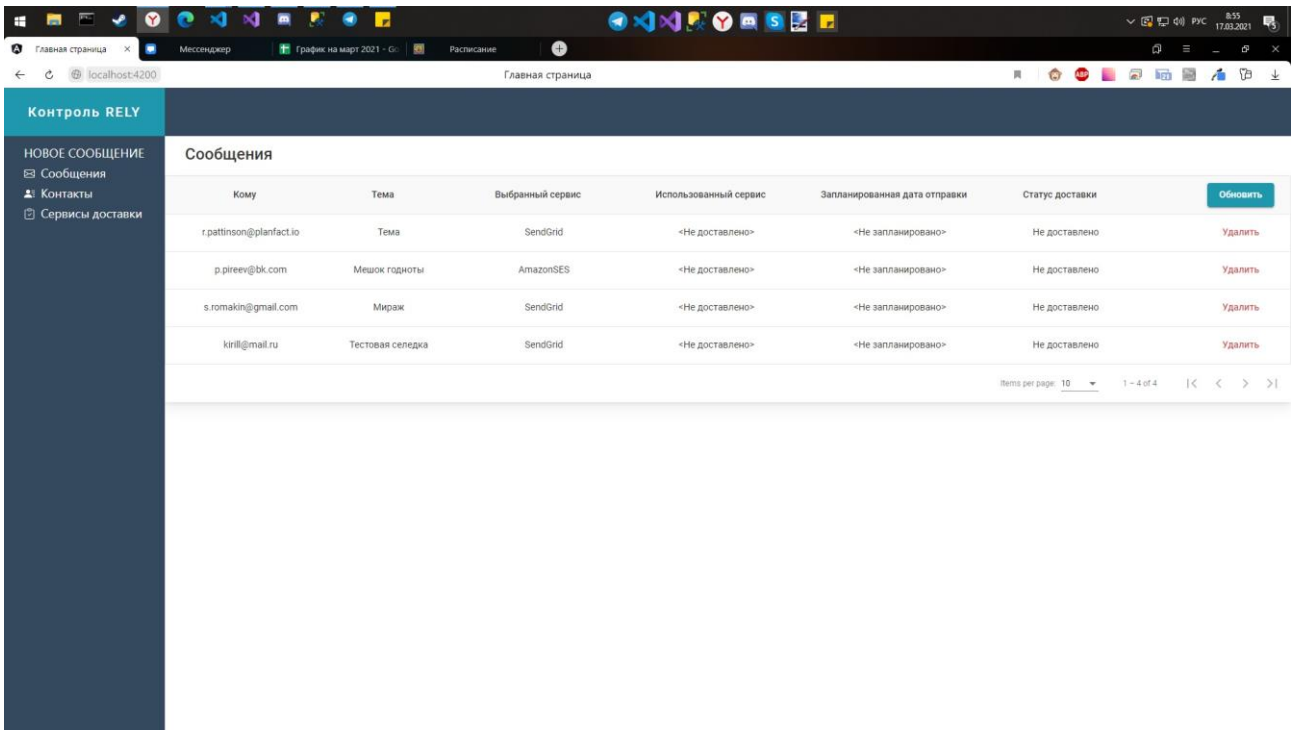

Рис. 3.10. Журнал сообщений

4. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Для проверки корректности разработанного программного комплекса проводятся следующие виды тестирования:

a) тестирование корректности работы web API методом черного ящика;
b) модульное тестирование;
c) кросс-браузерное тестирование интерфейсов пользовательской части.

4.1. **Тестирование web API методом черного ящика**

Тестирование методом черного ящика – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы [12]. Метод тестирования «черного ящика» – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство.

В рамках данного тестирования проверены следующие контроллеры web API:

a) messages;
b) users;
c) contacts;
d) files;
e) delivery-services;
f) delivery-queues;
g) accesses;
h) sales.

Примеры уже выполненных тестов представлены в таблице 4.1.

Таблица 4.1.

Тесты для тестирования методом черного ящика

Номер теста	Входные данные	Ожидаемый результат	Соответствие полученного результата ожидаемому

1	Запрос: GET https://localhost:44306/api/messages/list HTTP/1.1	{
		"attachedFiles": null,
		"deliveryQueue": {
		"deliveryQueueId": 2,
		"sendingIntervalSec": 40
		},
		"chosenDeliveryService": {
		"deliveryServiceId": 3,
		"deliveryServiceName": "SendGrid",
		"standartPriority": 1,
		"connectionString": "'sendGrid road'"
		},
		"usedDeliveryService": null,
		"user": {
		"userId": 4,
		"userName": "Evgeny Sukharev",
		"email": "r0bari@yandex.ru",
		"isActive": true,
		"role": 2
		},
		"messageId": 7,
		"theme": "Тема",
		"body": "Обычный текст",
		"destinationDate": "2021-01-16T21:05:00",
		"destinationEmail": "r.pattinson@planfact.io",
		"size": 50,
		"isScheduled": false,
		"scheduleDate": null,
		"isSent": false,
		"deliveryQueueId": 2,
		"chosenDeliveryServiceId": 3,
		"usedDeliveryServiceId": null,
		"deliveryStatus": 1,
		"userId": 4
		}
			Да

2	Запрос: POST https://local	{
		"data": {
		"userId": 4,
		"userName": "Evgeny Sukharev",
		"email": "r0bari@yandex.ru",
		"isActive": true,
			Да

	host:44306/api/auth/sign-in
	Body:
	{
	"email": "r0bari@yandex.ru",
	"password": "Eviguf@77"
	}
		"role": 2
		},
		"isSuccess": true,
		"errorMessage": null
		}	

3	Запрос:
	GET
	https://localhost:44306/api/contacts/3
		{
		"data": {
		"user": {
		"userId": 4,
		"userName": "Evgeny Sukharev",
		"email": "r0bari@yandex.ru",
		"isActive": true,
		"role": 2
		},
		"contactId": 3,
		"userId": 4,
		"contactEmail": "maxonfjvipon@yandex.ru",
		"contactName": "Максим Трунников"
		},
		"isSuccess": true,
		"errorMessage": null
		}
			Да

4.2. **Модульное тестирование**

Модульное тестирование, или юнит-тестирование (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы [13].

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

В рамках модульного тестирования созданы новые проекты DeliveryRely.ServiceLayer.Tests и DeliveryRely.Domain.Tests.

В проекте Delivery.Domain.ServiceLayer.Tests содержатся тесты для методов класса Hash и других вспомогательных классов, таких как AutoMapper. Пример модульного теста из данного проекта представлен в листинге 4.1.

Листинг 4.1. Модульный тест механизма хэширования

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace DeliveryRely.ServiceLayer.Tests
{
    [TestClass]
    public class HashSHA512Tests
    {
        [TestMethod]
        public void MakeHashTest()
        {
            string password = "Eviguf@77";
            string passwordHash = "dba23bd80b85edfd9ea2f28e4cc4bb953" +
                                  "86df6b0183567ca3179cb156120f5c4" +
                                  "0312fae40f7ad74431ca921ce4c2c27" +
                                  "d4c92b604051bf090947649801dc54e5e";
            string actualPasswordHash = new HashSHA512().MakeHash(password);
            Assert.Equals(passwordHash, actualPasswordHash);
        }
    }
}
```

В проекте DeliveryRely.Domain.Tests содержатся тесты для методов получения (методы Get…), добавления (методы Insert…), изменения (методы Update…) и удаления (методы Delete…) класса EFRepository. Пример теста представлен в листинге 4.2.

Листинг 4.2. Модульный тест контактов

```
using Domain.Models.Contacts;
using Domain.Models.Users;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace DeliveryRely.Domain.Tests
{
    [TestClass]
    public class ContactTest
    {
        [TestMethod]
        public void FieldsTest()
        {
            var contact = new Contact
            {
                ContactId = 1,
                ContactName = "contact",
                ContactEmail = "contact@yandex.ru",
                UserId = 4,
                User = new User
                {
                    Email = "user@yandex.ru"
                }
            };
            Assert.AreEqual("contact", contact.ContactName);
            Assert.AreEqual("contact@yandex.ru", contact.ContactEmail);
            Assert.AreEqual(4, contact.UserId);
            Assert.AreEqual("user@yandex.ru", contact.User.Email);
        }
    }
}
```

4.3. **Кросс-браузерное тестирование пользовательской части**

Кросс-браузерность – это способность веб-ресурса отображаться одинаково и работать во всех популярных браузерах, без перебоев в функционировании и ошибок в верстке, а также с одинаково корректной читабельностью контента. Из-за стремительного развития технологий, сайт рекомендуется делать кросс-браузерным только по отношению к новейшим версиям браузеров [11].

В последние годы оптимизаторы все меньше сталкиваются с проблемами кросс-браузерности, потому что разработчики программного обеспечения перестают поддерживать старые версии своих браузеров. Так, например, компания Microsoft уже в открытую заявляет о необходимости использования

обновленного Internet Explorer, призывая отказываться от устаревших версий продукта.

Стоит отметить, что элементы веб-ресурса не должны быть абсолютно идентичны во всех браузерах, которые только существуют на сегодняшний день. Соответствующим можно считать сайт, если:

- информация на странице читабельная;
- сохранена структура;
- нет ошибок в верстке;
- текст не «накладывается» поверх другого текста и изображений, если только это не предусмотрено автором контента.

Тестирование кросс-браузерности — вид тестирования, направленный на поддержку и правильное полное отображение программного продукта в разных браузерах, мобильных устройствах, планшетах, экранах различного размера.

В рамках данного тестирования в самых распространенных браузерах были проверены основные интерфейсы клиентской части.

Список проверяемых браузеров:

1) Google Chrome 89.0;
2) Safari 14;
3) Mozilla Firefox 91.6 ESR;
4) Opera 12.

Список сравниваемых интерфейсов:

1) страница авторизации;
2) главная страница;
3) список сервисов доставки;
4) список сообщений;
5) список контактов;
6) страница создания сообщения.

В качестве примера приводятся скриншоты интерфейсов страницы сообщений в браузерах Google Chrome (рис. 4.1) и Opera (рис. 4.2).

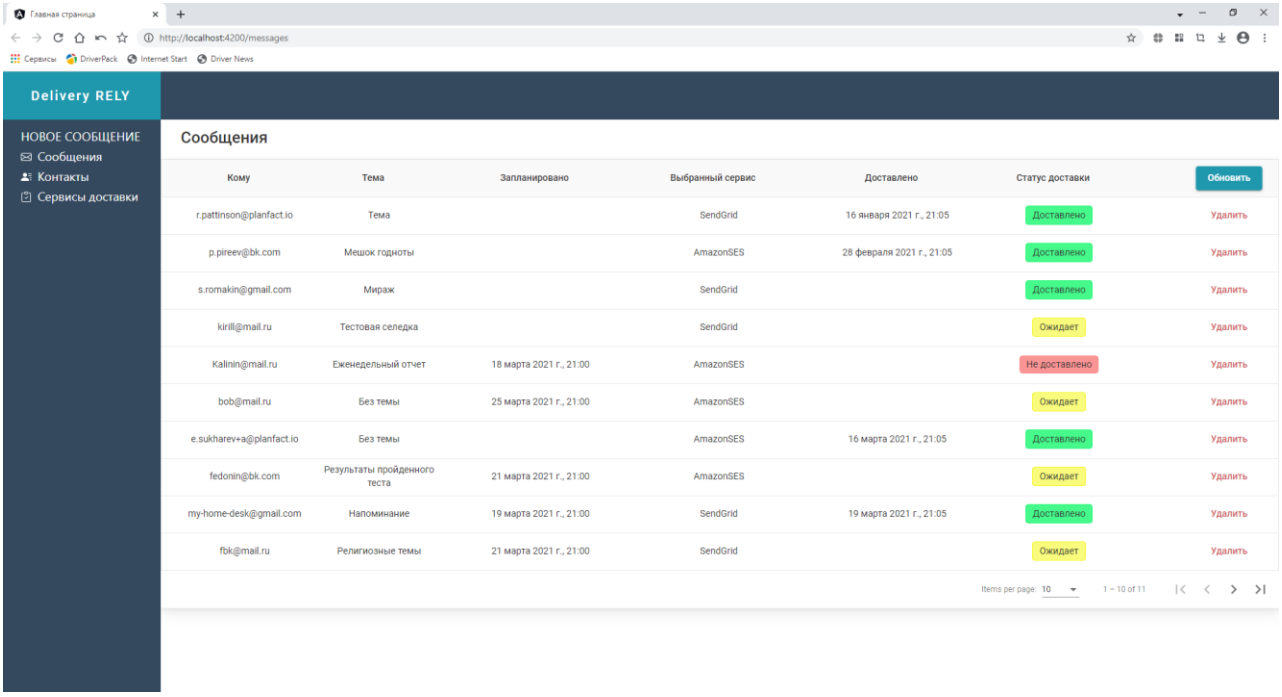

Рис. 4.1. Список сообщений в Google Chrome

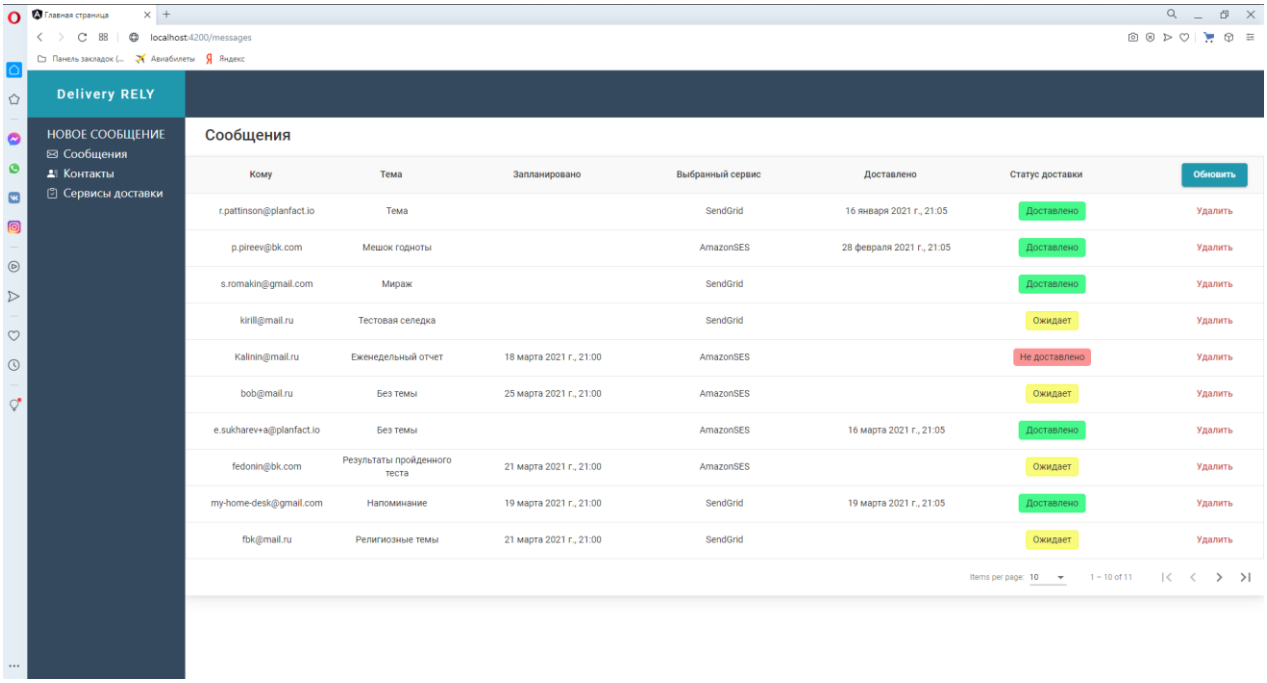

Рис. 4.2. Страница сообщений в Opera

В результате проведения описанных выше тестов критических ошибок найдено не было. Все разработанные испытания в тестировании методом черного ящика и тесты в модульном тестировании пройдены успешно, а все разработанные интерфейсы успешно проверены на кросс-браузерность и могут применяться в дальнейшей разработке программного комплекса.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы на тему «Проектирование программного комплекса управления надежностью отправки электронных писем» был выполнен ряд работ, связанных с планированием, сбором и анализом требований, проектированием, разработкой тестов и выполнением части из них.

Было разработано техническое задание для разрабатываемого программного комплекса.

В рамках проектирования были рассмотрены основные бизнес-процессы и спроектированы соответствующие UML-диаграммы, а также ERD, DFD, IDEF0-диаграмма и BPMN.

Также была спроектирована архитектура программного комплекса, на которой отмечены основные модули подсистемы.

В экспериментальной части описаны виды тестирований, применяемых для проверки качества спроектированного продукта, а также представлены результаты некоторых уже разработанных тестов.

СПИСОК ЛИТЕРАТУРЫ

1. Что такое web-интерфейс: [Электронный ресурс]. URL: https://semantica.in/blog/veb-interfejs.html/.
2. Что такое API: [Электронный ресурс]. URL: https://skillbox.ru/media/code/chto_takoe_api/.
3. Виды электронных подписей: [Электронный ресурс]. URL: http://www.consultant.ru/document/cons_doc_LAW_112701/d9cd621c949a3c9efef51c2884c247e18ab9908b/.
4. Сертификат ключа подписи: [Электронный ресурс]. URL: https://uc.iitrust.ru/articles/chto-takoe-sertifikat-klyucha-proverki-elektronnoy-podpisi/.
5. Официальный сайт Amazon SES: [Электронный ресурс]. URL: https://aws.amazon.com/ru/ses/.
6. Официальный сайт Sendgrid: [Электронный ресурс]. URL: https://sendgrid.com.
7. Проект Email queue пользователя tin-cat на Github: [Электронный ресурс]. URL: https://github.com/tin-cat/emailqueue.
8. Введение в Web API: [Электронный ресурс]. URL: https://metanit.com/sharp/mvc/12.1.php.
9. Документация по ASP.NET: [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-3.1.
10. Нормализация сущностей: [Электронный ресурс]. URL: https://habr.com/ru/post/254773.
11. Основы UML – диаграммы использования (use-case): [Электронный ресурс]. URL: https://pro-prof.com/archives/2594.
12. Тестирование методом черного ящика: [Электронный ресурс]. URL: https://habr.com/ru/post/462837/.
13. Модульное тестирование или юнит-тестирование: [Электронный ресурс]. URL: https://habr.com/ru/post/169381/.

14. Кросс-браузерное тестирование: [Электронный ресурс]. URL: https://imajor.ru/razrabotka/verstka/krossbrauzernost-sayta.
15. Язык UML. Руководство пользователя: [Текст] / Буч Г., Рамбо Д., Якобсон И. – 2-е издание: Пер. с англ Мухин Н., М.: ДМК Пресс. – 496 с.: ил.
16. Документация по C#: [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/
17. Официальная страница Angular: [Электронный ресурс]. URL: https://angular.io/?developerstash.
18. Большая книга CSS: [Текст] / Дэвид Макфарланд - СПб.: Питер, 2016.
19. Современный учебник JavaScript: [Электронный ресурс]. URL: https://learn.javascript.ru/
20. Язык программирования C# 7 и платформы .NET и .NET Core [Текст] / Троелсен, Джепикс – 2018.
21. Software Engineering for Internet Applications by Eve Andersson [Текст] / Philip Greenspun, Andrew Grumet – 2006.
22. Техническая документация по SQL Server: [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15