МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО

«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Информатика и программное обеспечение»

«У Т В Е Р Ж Д А Ю»
Зав. кафедрой «И и ПО», к.т.н., доцент
_____ Подвесовский А.Г.
«____» _____ 2021 г.

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО КОМПЛЕКСА УПРАВЛЕНИЯ НАДЕЖНОЙ ОТПРАВКОЙ ЭЛЕКТРОННЫХ ПИСЕМ С РЕАЛИЗАЦИЕЙ ОСНОВНОГО МЕХАНИЗМА РАССЫЛКИ И ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Д И П Л О М Н А Я Р А Б О Т А

Документы текстовые

Всего ____ листов в папке

Руководитель
_____ к.т.н., доц. Трубаков А.О.
«___»_____2021 г.
Консультанты:
по экономической части
_____ к.т.н., доц. Титарев Д.В.
«___»_____2021 г.
по организационной части
__ст. преп. Зяблова Е.С.
«___»_____2021 г.

Нормоконтролер
_____ к.т.н., доц. Трубаков А.О.
«___»_____2021 г.

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

БРЯНСК 2021

АННОТАЦИЯ

Данная дипломная работа посвящена вопросам проектирования и разработки программного комплекса, служащего для управления и контроля надежной отправки электронных писем.

На данный момент в спроектированном комплексе выполнена реализация следующих модулей:

1) интерфейсы пользовательской части для просмотра доступной информации;
2) компоненты пользовательской части, отвечающие за обработку действий пользователя и отправку запросов на сервер;
3) web-API [2] для обработки получаемых сервером запросов;
4) фоновая служба для асинхронной отправки сообщений;
5) база данных для хранения основных сущностей.

В анализе требований представлен обзор процесса отправки электронного сообщения, программ-аналогов, диаграммы вариантов использования, IDEF0-диаграммы. Описана диаграмма BPMN, рассмотрен бизнес-процесс от создания сообщения пользователем до отправки этого сообщения получателям.

Экономический анализ представляет собой определение организационной структуры, необходимой для создания программного комплекса, составление календарного плана и расчёт себестоимости проекта.

В рамках проектирования и разработки программного комплекса были выбраны основные инструменты разработки, разработана архитектура программного комплекса, спроектированы и разработаны основные интерфейсы. Спроектирована база данных, а особенности основных ее сущностей отражены на ER-диаграмме. Также разработаны диаграмма потоков данных и диаграмма классов для службы отправки сообщений.

Тестирование программного комплекса содержит перечень видов испытаний, их краткое описание, а также результаты проведенных испытаний.

Организационная часть описывает вредные и опасные факторы, которым подвергаются пользователи программного комплекса.

СОДЕРЖАНИЕ

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.

Электронное сообщение представляет из себя информацию, переданную или полученную пользователем сети Интернет. Электронным документом признается электронное сообщение, подписанное цифровой подписью.

Многие крупные компании используют электронную почту для обмена документами. В таких ситуациях крайне важно, чтобы сообщение было успешно доставлено и прочитано получателем.

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.

Тема работы: «Проектирование программного комплекса управления надежной отправкой электронных писем с реализацией основного механизма рассылки и пользовательского интерфейса».

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

Целью дипломной работы является проектирование программного комплекса управления рассылкой электронных сообщений с возможностью последующего внедрения в системы, требующие высокой 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 информационную систему ПланФакт.

Объектом исследования является процесс рассылки электронных сообщений.

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

1. АНАЛИЗ ТРЕБОВАНИЙ

Разработка программного комплекса 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.

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.

Контакт – сохраненный получатель пользователя.

Статус доставки – информация о результате доставки сообщения получателю.

Фоновая служба доставки – процесс, запущенный на сервере и функционирующий на протяжении всего времени работы программного комплекса. Отвечает за получение сообщения из очереди и передачу этого сообщения доступным сервисам доставки.

Авторизация – предоставление пользователю прав на выполнение определенных действий.

Авторизованный пользователь – пользователь, который успешно прошел этап авторизации и получил доступ к программному комплексу.

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

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

Электронная цифровая подпись (ЭЦП) [3] – 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.

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

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.

Процесс создания и отправки сообщения схематично изображен на рис. 1.1.

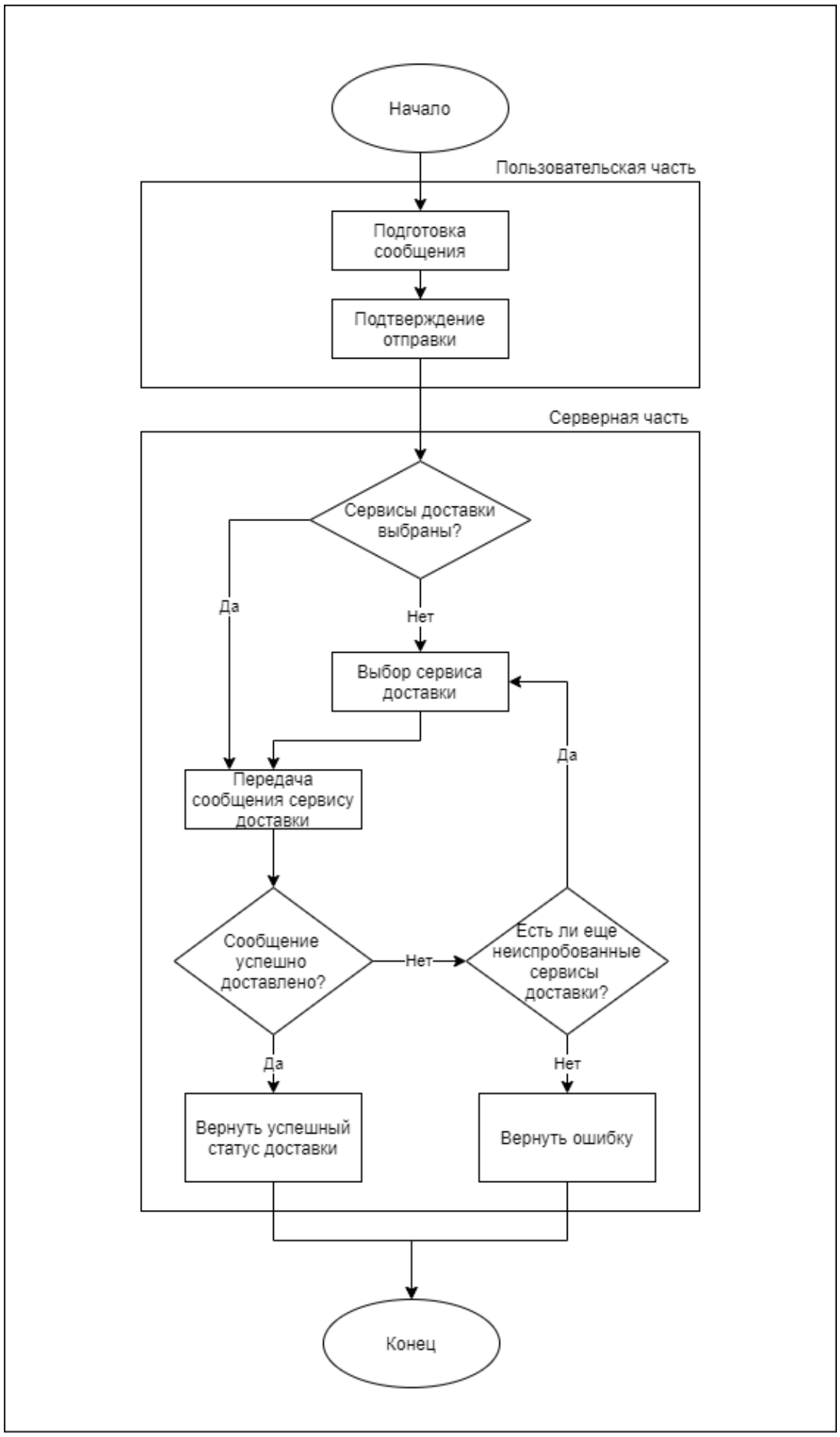

Рис. 1.1. Процесс отправки сообщения

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 [5];
2) Sendpulse [27];
3) Tin-cat [7].

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. Sendpulse

Sendpulse [27] – сервис, с помощью которого можно создавать и распространять рассылки E-mail и SMS-сообщений.

Сервис предлагает следующие возможности:

- возможность рассылать различные типы сообщений: по электронной почте, через мессенджеры, в виде смс или Push-сообщений;
- интегрирование работы с мессенджерами Facebook и Viber;
- удобный конструктор, который сочетает богатые возможности с простотой использования;
- 130 шаблонов для рассылаемых писем;
- возможность автоматизации сложных процедур;
- средства для проведения сегментации и персонализации;
- возможность работы с 40 сервисами;
- продуманная и удобная админ-панель;
- бесплатный тариф, который подойдет тем, кто делает первые шаги.

Достоинства сервиса:

- возможность автоматизации сложных процессов работы;
- удобный встроенный блочный конструктор;
- много встроенных возможностей;
- работа с различными типами рассылок;
- гибкие и недорогие тарифы, возможность бесплатной работы.

Недостатки SendPulse:

- мало подписчиков на бесплатном тарифе;
- медленная работа техподдержки;
- часто изменяющееся API, из-за которого может страдать обратная совместимость;
- мало возможностей для создания писем-триггеров;
- если получатель пометил сообщение как «Спам», то сервис может блокировать аккаунт отправителя и приостанавливать дальнейшую рассылку.

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	Sendpulse	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 ненадежность API (Sendpulse). Важно: если сообщение не удается отправить, данные сервисы не предпринимают попыток отправить его другим способом, что отрицательно сказывается на надежности рассылки.

1.3. Диаграмма вариантов использования

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 отправка сообщения и просмотр истории сообщений.

При отправке сообщений оператор должен заполнить поле «Кому». Дополнительно он может добавить получателей, 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, что сделано для упрощения отправки сообщений и повышения гибкости разрабатываемого программного комплекса.

Оператор может сохранять контакты. Контакт – запись, представляющая собой пару из имени получателя и его почтового адреса. И затем в дальнейшем выбирать получателей из существующих аккаунтов (в форме создания сообщения при вводе получателя появляются контекстные подсказки из списка контактов).

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

Также оператор может просмотреть список доступных на данный момент сервисов доставки и особенности выбранного сервиса 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.

На данный момент разработан модуль авторизации, но все зарегистрированные и авторизованные пользователи будут обладать правами оператора.

В целях визуализации описанного выше функционала составлена диаграмма вариантов использования. Она представлена на рис. 1.2.

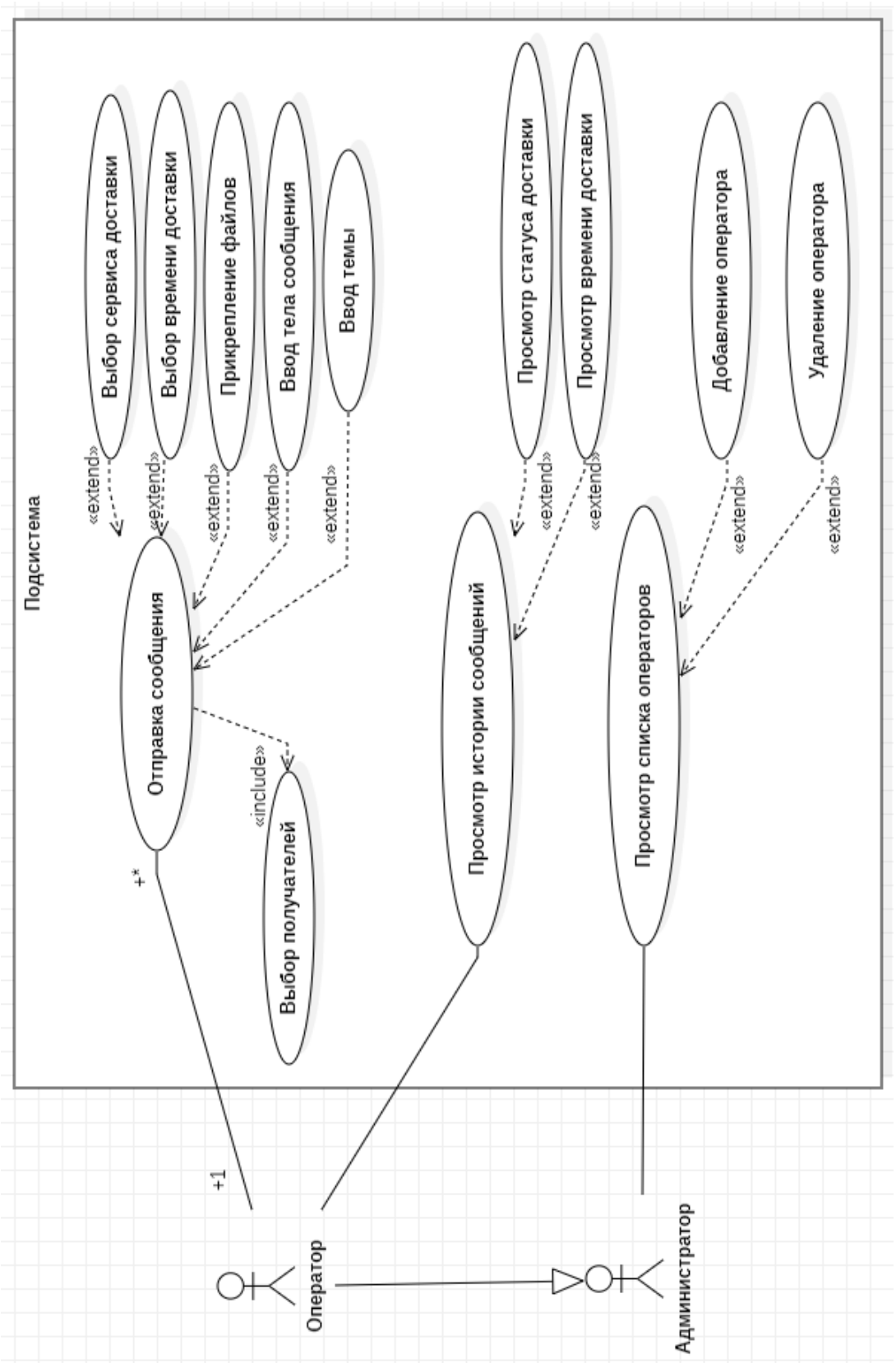

Рис. 1.2. Диаграмма вариантов использования

1.4. IDEF0-Диаграмма

Разработана IDEF0-диаграмма для процесса отправки сообщения.

На контекстной диаграмме (рис. 1.3) определен функциональный блок «Отправить сообщение через систему», механизм, предписание, входные и выходные данные.

На диаграмме первого уровня (рис. 1.4) 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. 1.5) и «Инициировать отправку сообщения» (рис. 1.6)

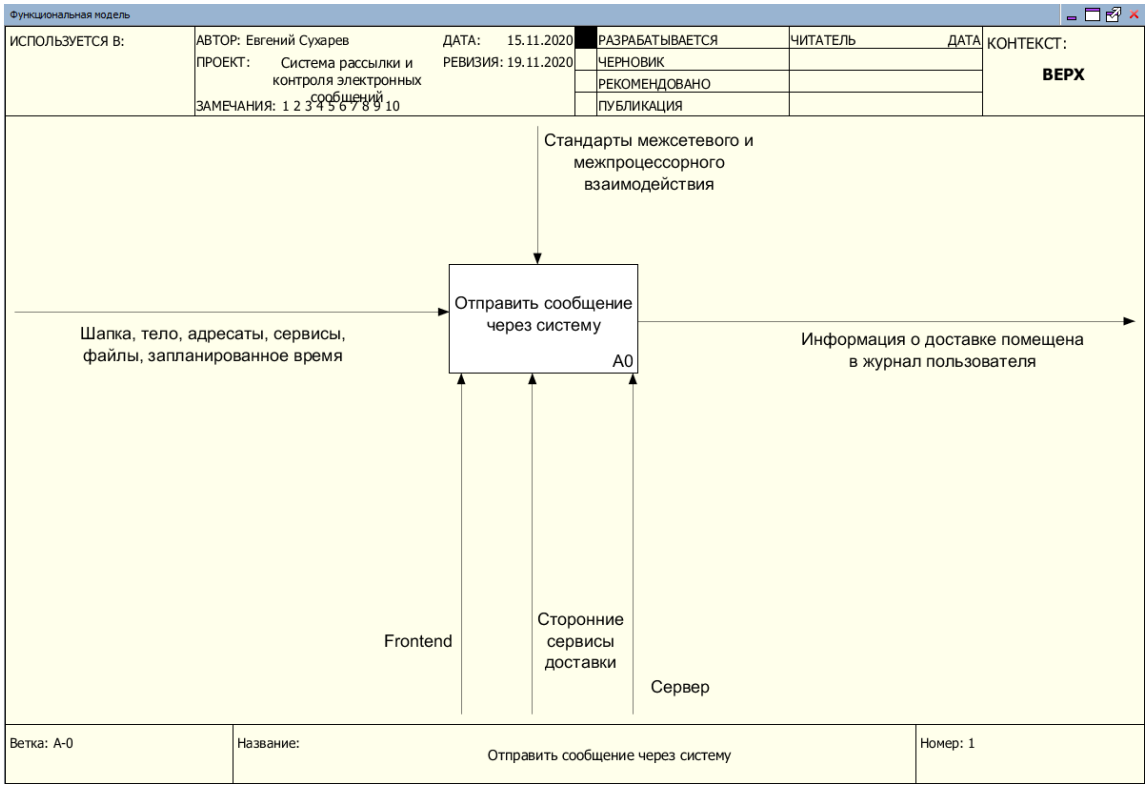

Рис. 1.3. Контекстная диаграмма функциональной модели «Поиск объектов»

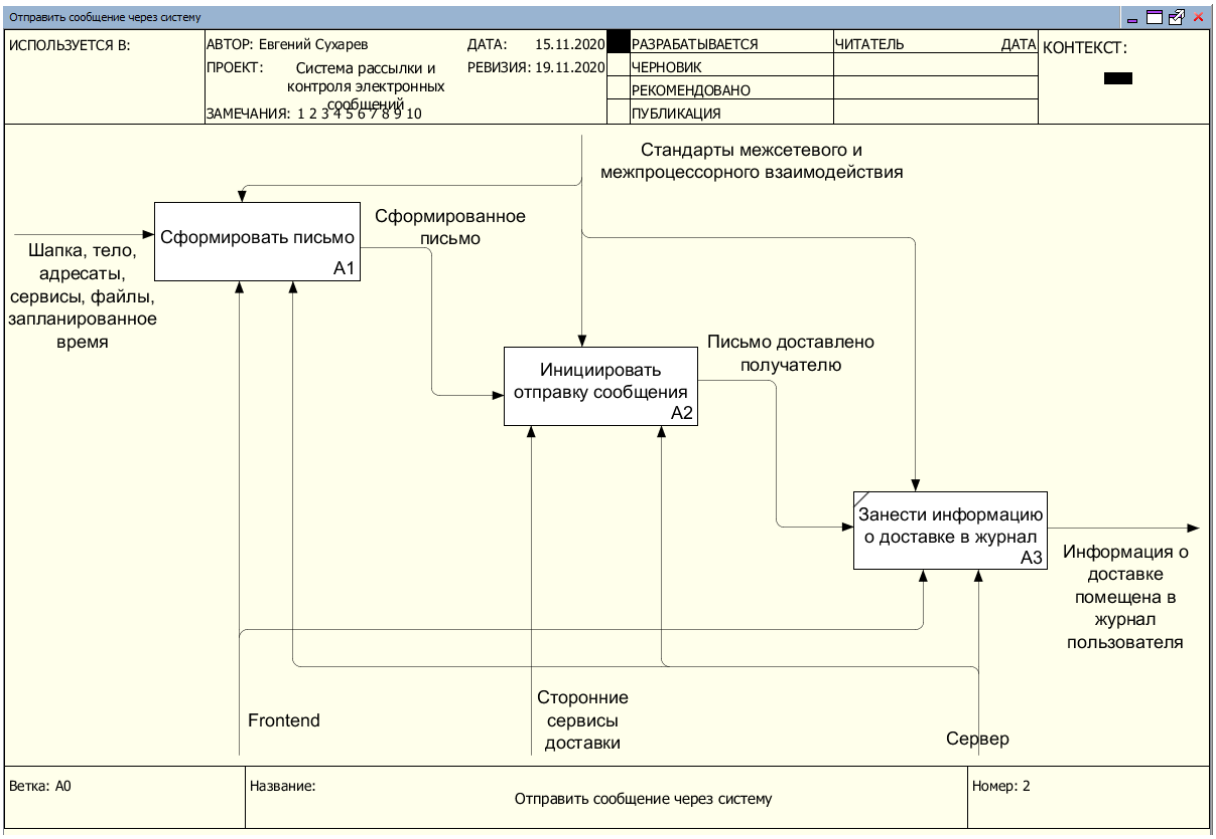

Рис. 1.4. Диаграмма первого уровня функциональной модели

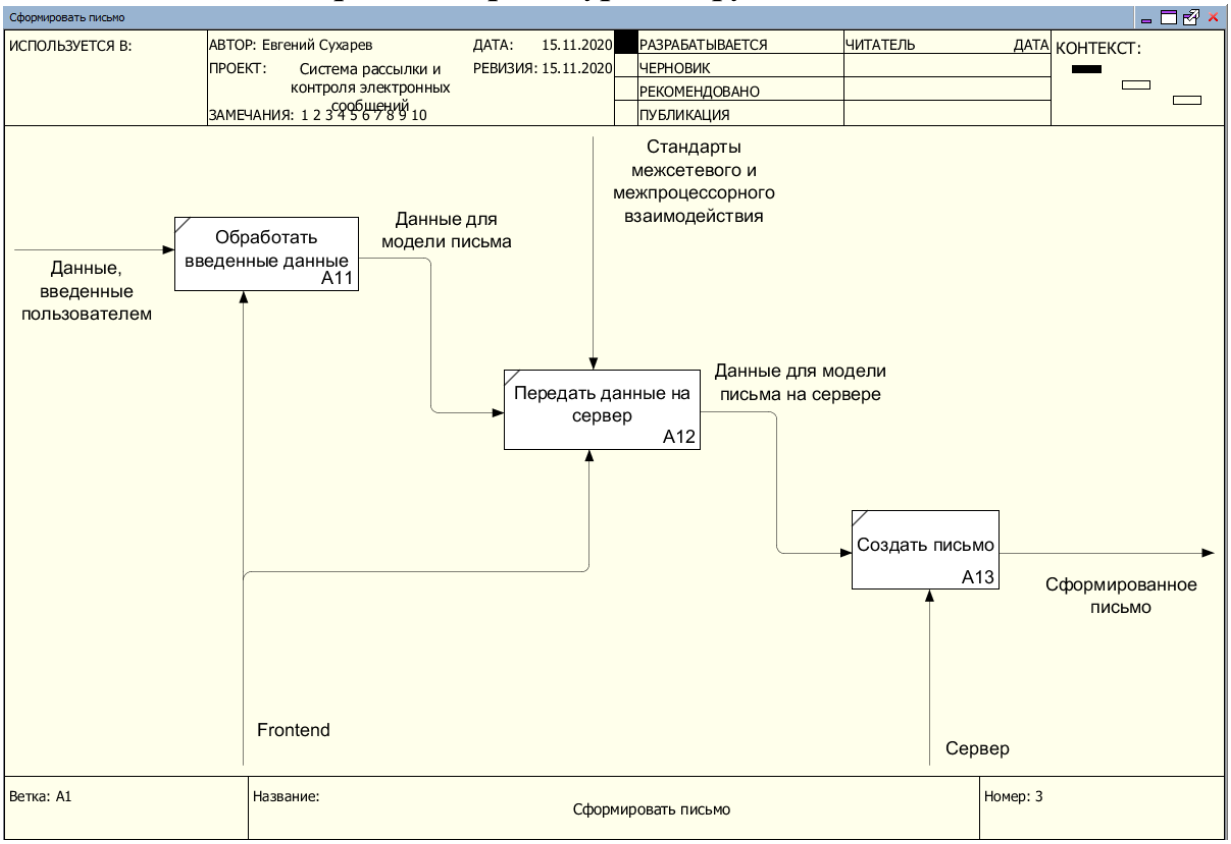

Рис. 1.5. Диаграмма второго уровня для блока «Сформировать письмо»

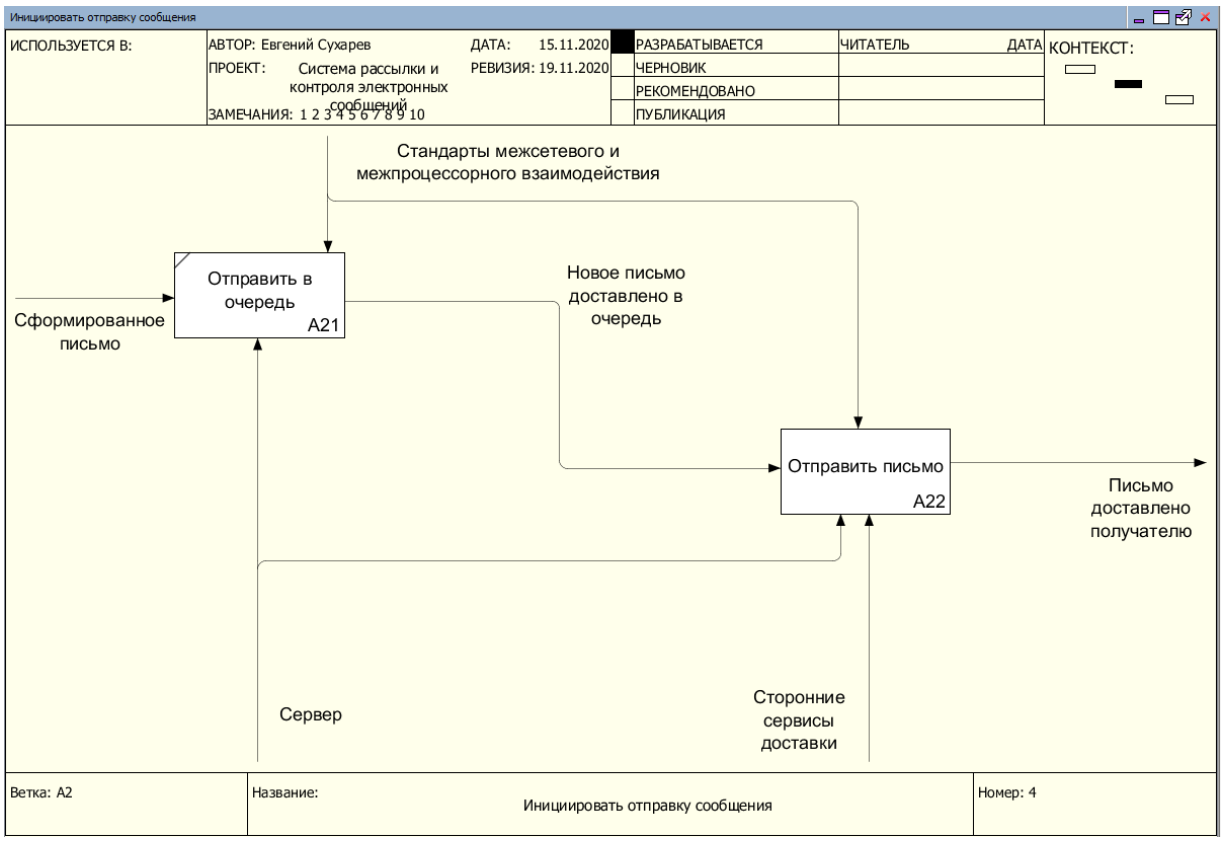

Рис. 1.6. Диаграмма второго уровня для блока «Инициировать отправку сообщения»

1.5. BPMN

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. Фрагменты модели бизнес-процесса представлены на рис. 1.7-1.8.

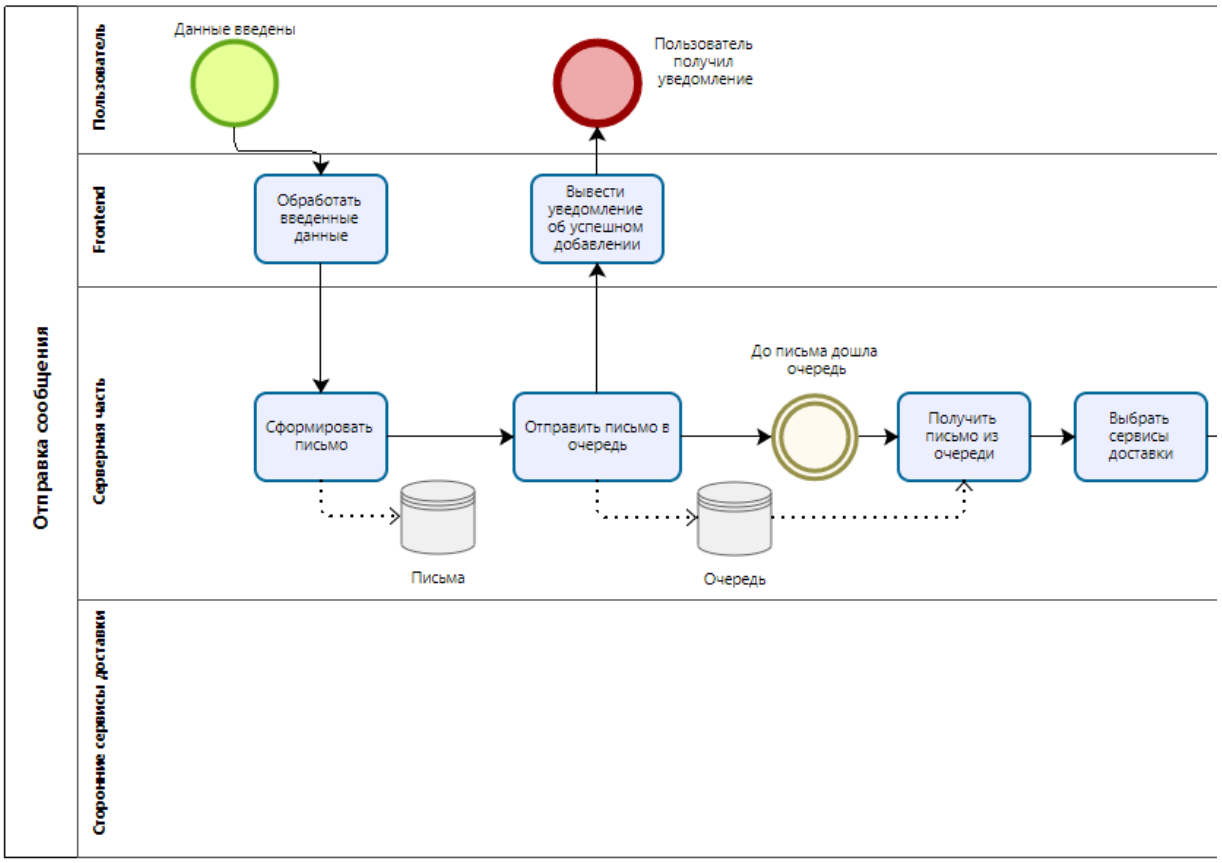

Рис. 1.7. Фрагмент диаграммы модели бизнес-процесса

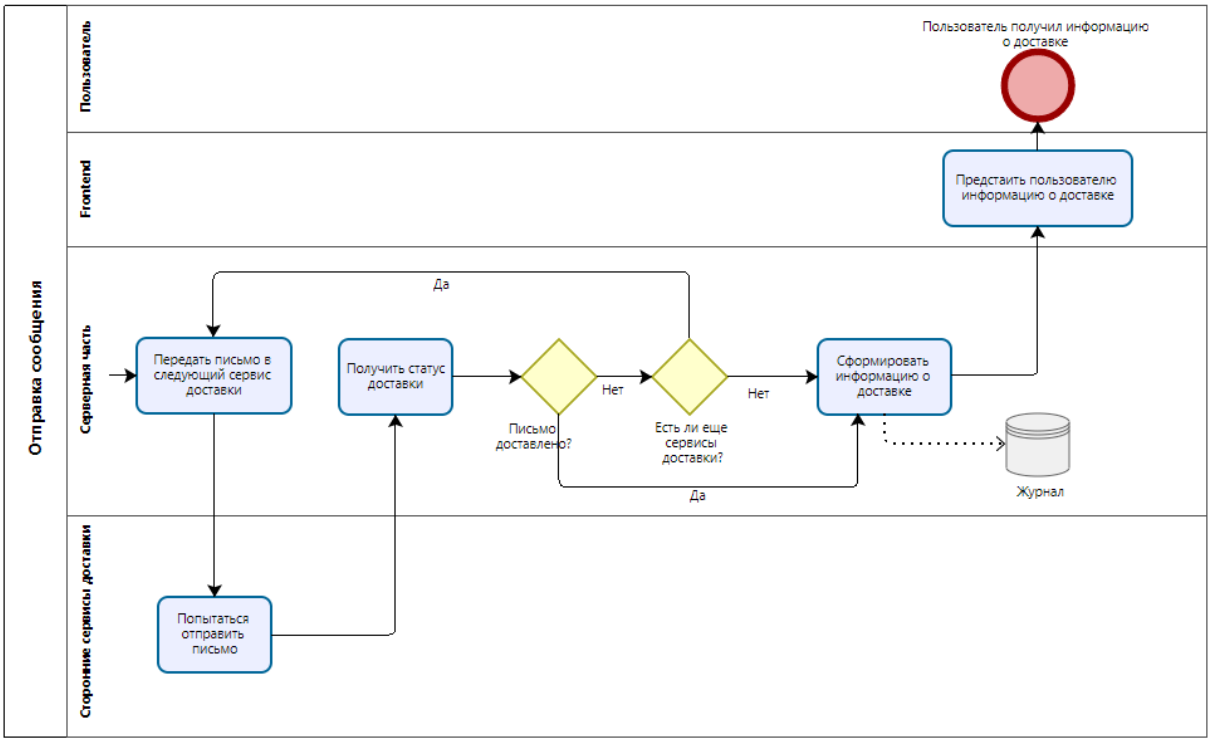

Рис. 1.8. Фрагмент диаграммы модели бизнес-процесса

1.6. Выводы

В результате проведенного анализа была изучена предметная область (процесс отправки электронных сообщений), проведено сравнение и выявлены ключевые недостатки существующих программ-аналогов, описаны варианты использования программного комплекса, разработаны IDEF0-диаграмма и BPMN.

Тема работы: «Проектирование программного комплекса управления надежной отправкой электронных писем с реализацией основного механизма рассылки и пользовательского интерфейса».

Так как 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 отправки сообщений.

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

2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Наименование программы: «Программный комплекс управления надежной отправкой электронных писем».

2.1. Основание для разработки

Основанием для разработки подсистемы управления надежностью отправки электронных писем является задание на дипломную работу, выданное доцентом Трубаковым А.О.

2.2. 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.

2.3. Требования к программному комплексу

1.6.1. Требования к функциональным характеристикам

Реализуемый программный комплекс должен включать в себя нижеупомянутые функциональные особенности:

1. Авторизация пользователей в web-интерфейсе для администрирования:
   Роли:
   - оператор – может управлять рассылкой сообщений, просматривать соответствующую историю сообщений;
   - администратор – имеет все возможности оператора, но также может просматривать список операторов, создавать новых операторов и удалять существующих.
2. После осуществления входа в программный комплекс пользователь может:

- отправлять сообщения на существующие электронные почтовые адреса;
- выбирать сервисы доставки из списка доступных на данный момент;
- подключать свою электронную почту;
- прикреплять файлы к сообщению;
- организовывать отправку сообщений по расписанию;
- просматривать список доставленных сообщений;
- просматривать подробную информацию о доставке сообщений.

3. Разрабатываемый комплекс должен обладать следующими функциями:
   - по умолчанию в первую очередь осуществлять попытку отправлять сообщение с помощью SMTP-сервера;
   - возможность повторной отправки недоставленного сообщения, но уже посредством другого доступного сервиса;
   - возможность отправлять сообщения посредством минимум **трех** различных сервисов для отправки сообщений;
   - в случае неудачной попытки отправить сообщение посредством каждого доступного сервера сообщение помещается в конец очереди;
   - наличие API, с помощью которого разрабатываемый программный комплекс можно будет встраивать в различные сервисы, использующие рассылку сообщений;
   - независимость от пользовательского интерфейса – возможность выполнять все функции с помощью методов WEB API;
   - возможность прикреплять файлы к сообщению (до 5 файлов в сообщении общим объемом до 10 Мб). *Доступные* расширения: .doc, .docx, .xlsx, .csv, .png, .svg, .rar;
   - возможность хранить историю сообщений, т.е. хранить ранее отправленные сообщения и давать доступ к просмотру информации об их доставке;

- возможность скачать ранее прикрепленные файлы;
- 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.

4. 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.
5. В целях улучшения надежности разрабатываемый комплекс должен включать в себя кластерную систему серверов.
6. Требования к интерфейсу

Реализуемый 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:

1) тему;
2) тело сообщения;
3) список прикрепленных файлов;
4) список адресов, на которые нужно отправить сообщения;
5) выбранный сервис доставки, посредством которого необходимо доставить сообщение;
6) запланированное время доставки или диапазоны времени, в пределах которых сообщение должно быть доставлено.

Выходные данные. Система предоставляет пользователю историю сообщений с информацией о доставке, список контактов, список доступных сервисов доставки.

Отправка данных. Система обрабатывает полученные от пользователя данные через WEB API [2] и передает их указанным сервисам доставки.

1.6.2. 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.

2.4. Условия эксплуатации

1.6.3. Климатические условия эксплуатации

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

1.6.4. Требования к квалификации и численности персонала

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

1.6.5. 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;
- SSD диск.

Сервер баз данных:

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

1.6.6. Требования к 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, или Opera 12;
- антивирусное программное обеспечение.

Дополнительные требования к защите программного обеспечения и информации не предъявляются.

2.5. Программная документация

1.6.7. Предварительный состав программной документации

Состав программной документации:

- техническое задание;

- программа и методики испытаний;
- руководство оператора.

2.6. Стадии и этапы разработки

1.6.8. 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 программного комплекса.

1.6.9. 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.

2.7. Порядок контроля и приемки

1.6.10. Виды испытаний

Для проверки корректности работы разработанного программного комплекса должны быть проведены следующие виды испытаний:

1) тестирование WEB-API методом черного ящика;
2) модульное тестирование;
3) кроссбраузерное тестирование.

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

Кроссбраузерное 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.

Приемо-сдаточные испытания программы должны проводиться согласно разработанной Исполнителем и согласованной Заказчиком Программы и методик испытаний.

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

1.6.11. Общие требования к приемке работы

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

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1. При входе в веб-сервис под учетной записью пользователь должен иметь возможность:
    - просматривать список сообщений;
    - создавать новые сообщения;
    - выбирать несколько получателей;

- прикреплять до 5 файлов общим объемом до 10 МБ с расширением: .doc, .docx, .xlsx, .csv, .rar, .pdf, .txt.

7. Программный комплекс должен хранить сведения о:
    - сообщениях: дата доставки, тема сообщения, текст сообщения, размер, запланировано ли сообщение, запланированная дата доставки, отправлено ли сообщение, идентификатор очереди, в которой находится сообщение, выбранный пользователем сервис доставки, сервис доставки, посредством которого сообщение доставлено, идентификатор отправителя, статус доставки, дата создания сообщения, является ли текст сообщения html-страницей;
    - адресах получателей: электронный адрес получателя, сообщение, отправляемое на этот адрес;
    - очередях доставки: интервал доставки, дата последней отправки;
    - прикрепленных файлах: название файла, содержание файла, тип файла, сообщение, к которому файл прикреплен;
    - пользователях: имя и фамилия пользователя, электронный адрес почты пользователя, хэш пароля, статус активности пользователя, роль;
    - контактах: электронный адрес почты контакта, имя контакта, идентификатор пользователя, которому принадлежит данный контакт;
    - доступах: дата начала действия доступа, дата окончания действия доступа, идентификатор действующего тарифа, идентификатор пользователя, которому принадлежит данный доступ;
    - тарифах: название тарифа, цена;
    - скидках: наименование скидки, значение.

3. ЭКОНОМИЧЕСКИЙ АНАЛИЗ

Структура себестоимости программного продукта отражена в табл. 3.1 и представлена на рис. 3.1. Расчеты себестоимости приведены в прилож. 1.

Таблица 3.1

Структура себестоимости программного продукта

№	Элементы себестоимости	Сумма (руб.)	% в общ. сумме себестоимости
1	Общая заработная плата исполнителя	786 125	56,20
3	Отчисления на социальные нужды (страховые взносы)	95 702	6,84
4	Арендные платежи за производственные (офисные) помещения	134 572	9,62
5	Амортизация используемых основных средств и нематериальных активов	78 986	5,65
6	Расходы на модернизацию и приобретение основных средств	-	-
7	Расходы на приобретение необходимого ПО	-	-
8	Расходы на интернет, связь	-	-
9	Расходы на канцелярские товары и расходные материалы	4 200	0,30
10	Прочие расходы	299 114	21,39
	Итого:	**1 398 699**	**100**

Рис. 3.1. Структура себестоимости программного продукта

4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА

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. 4.1.

4.1. Клиентская часть

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

1) React [18];
2) Angular [17];
3) Vue [19].

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 [21]. Для вёрстки будет применяться следующий набор инструментов: HTML + CSS [20], а также адаптивные компоненты Material и некоторые стили Bootstrap. Bootstrap и Material уже входят в Angular, поэтому отдельно подключать их не требуется.

4.2. 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.

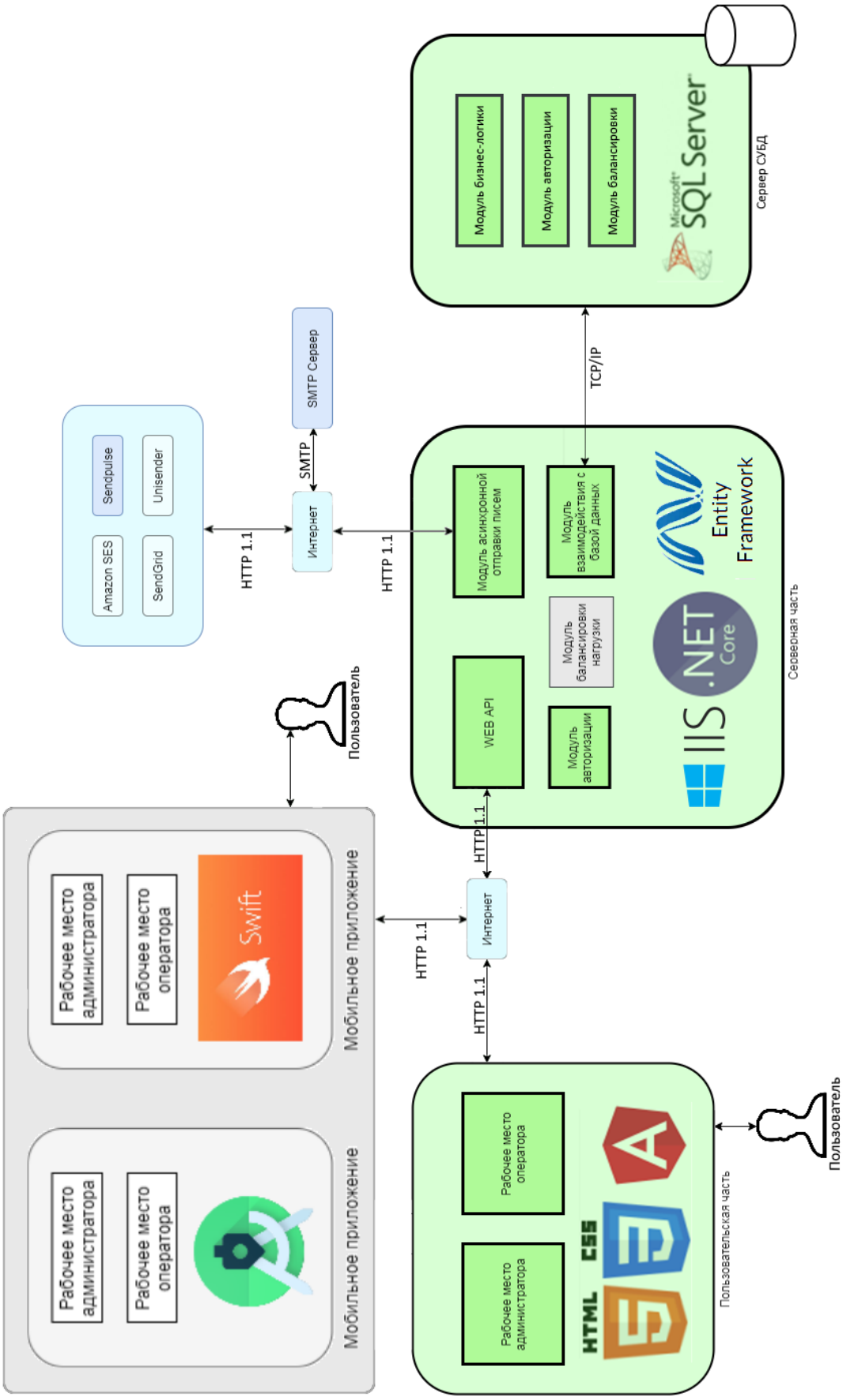

Рис. 4.1. Схема архитектуры ПС

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, собирает статистику запросов, задавать подходящий метод распределения нагрузки. Подробная

Модуль асинхронной отправки писем является фоновой службой и 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# [22], а также благодаря детальной технической документации от официального разработчика [9].

4.3. Модель данных

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

a) MySQL [25];

b) MS SQL [24];

c) PostgreSQL [26].

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) [24].

База данных условно разделена на ряд модулей:

1) модуль бизнес-логики;
2) модуль авторизации;
3) модуль балансировки.

В базе данных будут храниться все основные сущности системы:

1) сообщения;
2) пользователи;
3) ссылки на прикрепленные файлы;
4) контакты пользователей;
5) очереди доставки;
6) доступы пользователей;
7) тарифы;
8) скидки.

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

- Пользователи – Доступы;
- Тарифы – Скидки.

Модуль бизнес-логики включает в себя таблицы основных бизнес-сущностей (таких как сообщения, файлы и другие) и отношения между ними.

Модуль авторизации включает в себя таблицы, в которых хранится информация о зарегистрированных пользователях, их ролях и доступах.

Модуль балансировки состоит из таблиц доступных серверов и их приоритетов.

4.4. Низкоуровневое проектирование

1.6.12. 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:

1. Messages – таблица для хранения сообщений:
    - messageId (long) – суррогатный первичный ключ;
    - userId (long) – id отправителя;
    - destinationDate (dateTime) – дата и время успешной доставки;
    - theme (nvarchar(max)) – тема сообщения;
    - body (text) – текст 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;
    - deliveryQueueId (int) – id очереди, в которой содержится данное сообщение;
    - chosendDeliveryService (int) – id выбранного пользователем сервиса доставки;
    - usedDeliveryService (int) – id сервиса доставки, использованного для отправки;
    - deliveryStatus (int) – id статуса доставки;
    - creationDate (dateTime) – дата создания сообщения;

- isHtml (bit) – логическое поле – является ли текст html-страницей

2. AttachedFiles – таблица для хранения ссылок на прикрепленные файлы:
    - id (long) – суррогатный первичный ключ;
    - title (int) – название файла;
    - fileType – тип файла;
    - messageId – сообщение, к которому файл прикреплен.
3. Users – таблица для хранения информации о пользователях:
    - userId (long) – суррогатный первичный ключ;
    - userName (nvarchar(max)) – имя пользователя;
    - userSecondName (nvarchar(max)) – фамилия пользователя;
    - email (nvarchar(max)) – почтовый адрес пользователя;
    - isActive (bit) – логическое поле – является ли пользователь активным
    - passwordHash (nvarchar(max)) – хэш, полученный из пароля пользователя;
    - role (int) – текущая роль пользователя.
4. UsersAccesses – таблица для связи таблиц пользователей и доступов
    - userId (long) – суррогатный первичный ключ пользователя;
    - accessId (long) – суррогатный первичный ключ доступа
5. Accesses – таблица для хранения информации о доступах
    - accessId (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.
6. 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.

7. DeliveryQueues – таблица для хранения информации об очередях
    - id (int) – суррогатный первичный ключ;
    - sendingIntervalSec (int) – интервал отправки сообщений.
8. Tariffs – таблица для хранения информации о тарифах
    - id (int) – суррогатный первичный ключ;
    - tariffName (nvarchar(255)) – название тарифа;
    - cost (int) – стоимость тарифа.
9. TariffsSales – таблица для связи таблиц тарифов и скидок
    - tariffId (int) – суррогатный первичный ключ тарифа;
    - saleId (int) – суррогатный первичный /ключ скидки.
10. 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. 4.2.

Рис. 4.2. ER диаграмма

1.6.13. Диаграмма потоков данных

Разработана модель потоков данных для процесса создания сообщения и отправки сообщения.

На контекстной диаграмме (рис. 4.3) определены базовый блок «Система рассылки и учета электронных сообщений» и внешняя сущность «Пользователь», хранилища данных «Журнал», «Очередь», «Письма», а также информационные потоки между ними.

На диаграмме первого уровня (рис. 4.4) выполнена декомпозиция базового процесса на три подпроцесса. На диаграмме второго уровня проведена детализация процесса «Инициировать отправку письма» (рис. 4.5). На диаграммах третьего уровня проведена детализация процесса «Отправить письмо» (рис. 4.6).

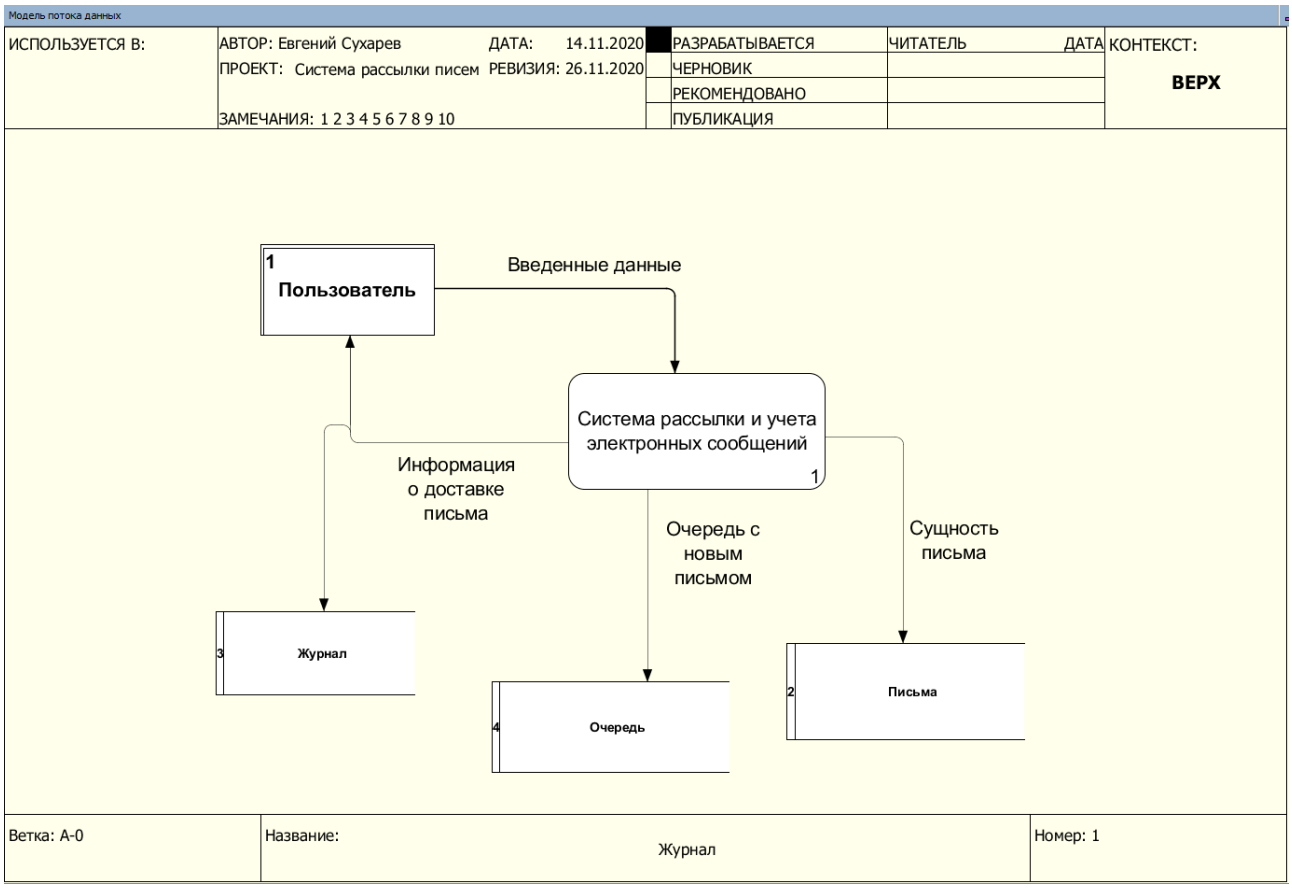

Рис. 4.3. Контекстная диаграмма

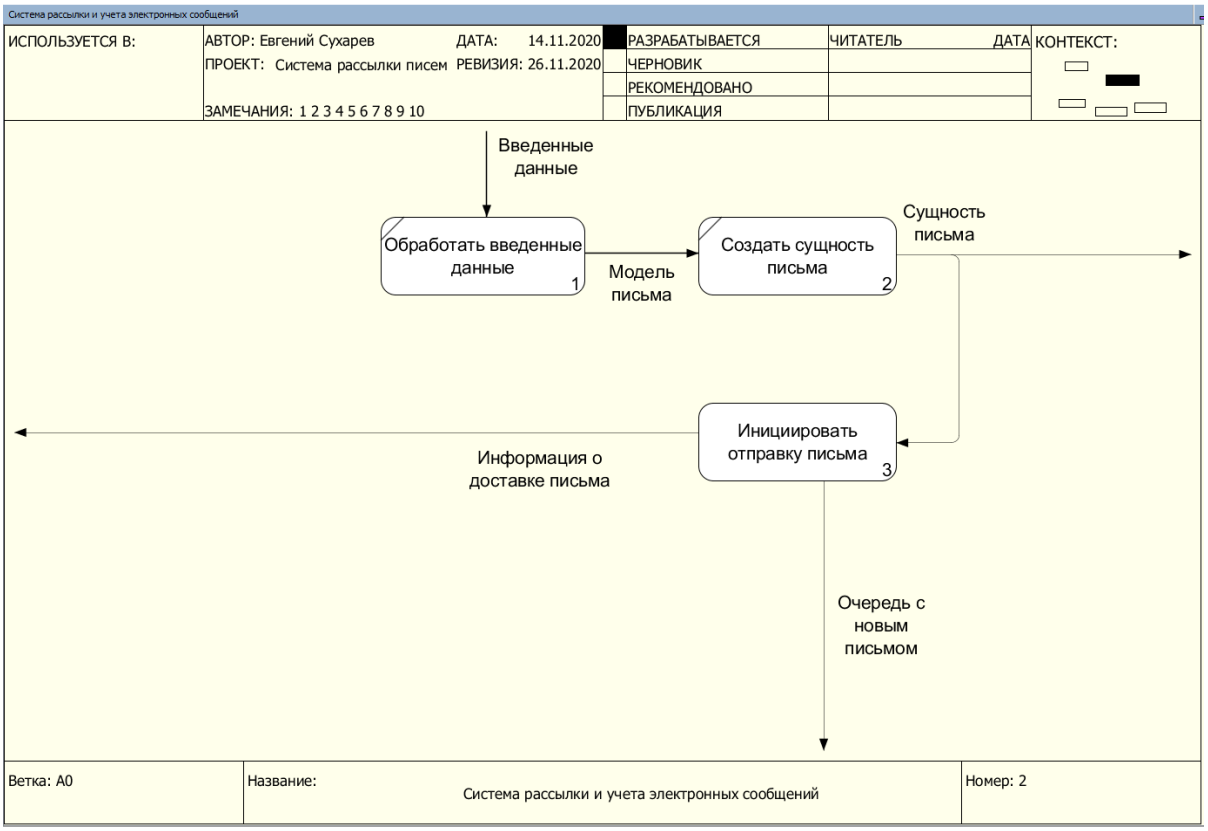

Рис. 4.4. Диаграмма первого уровня модели потоков данных

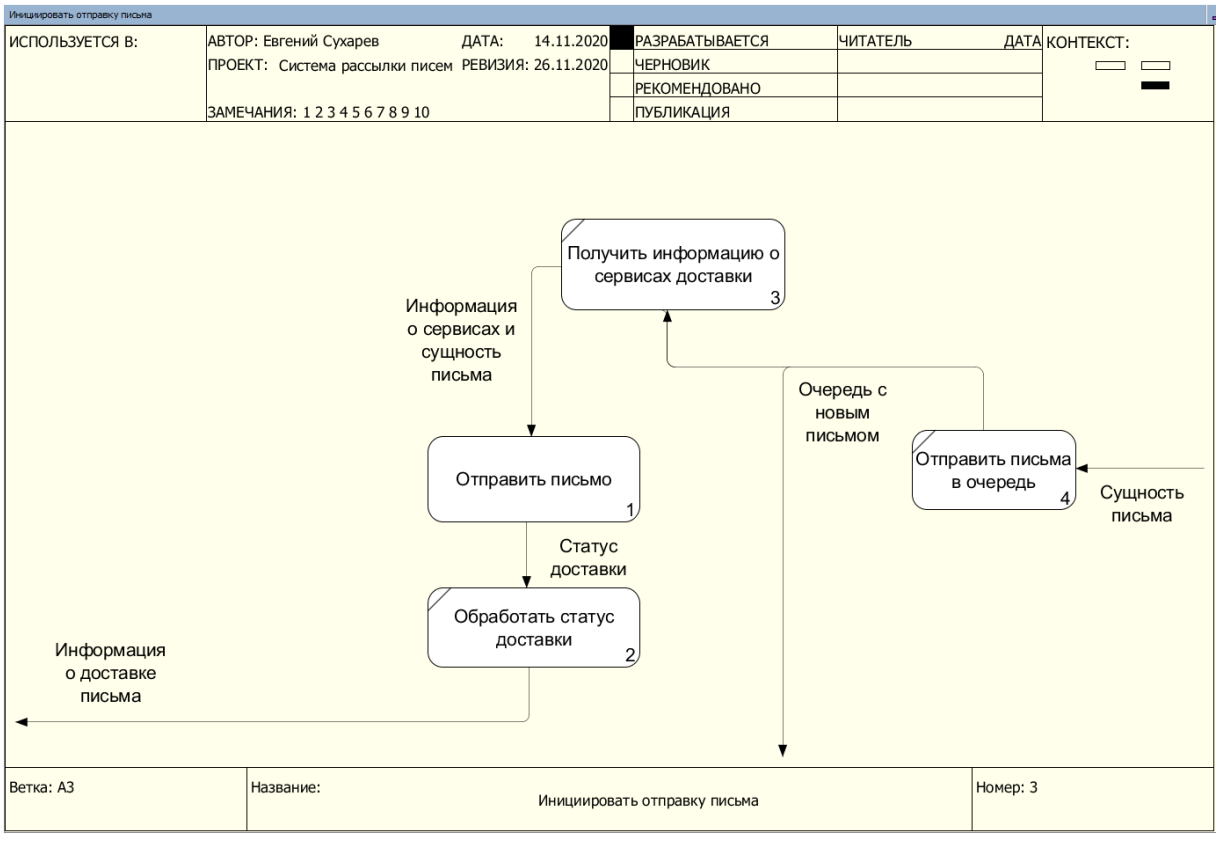

Рис. 4.5. Диаграмма второго уровня модели потоков данных. Процесс «Инициировать отправку письма»

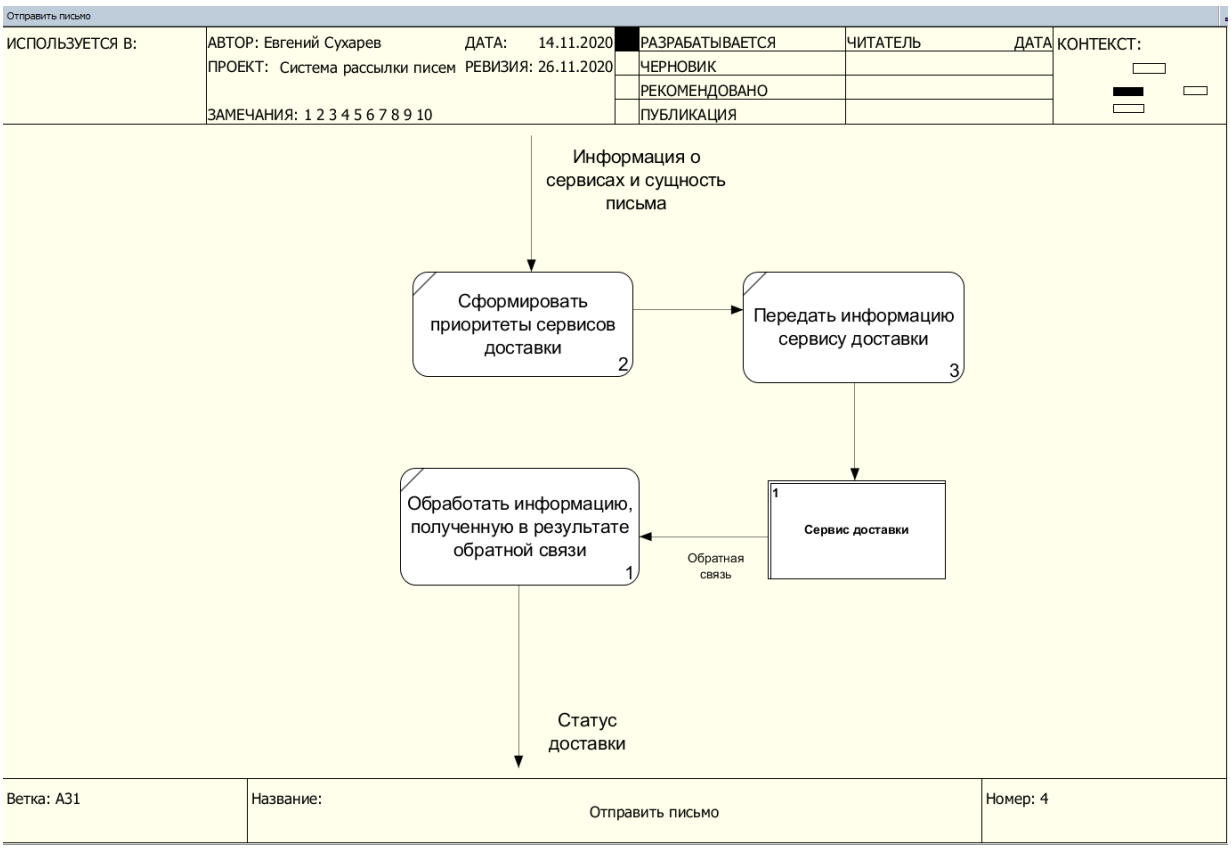

Рис. 4.6. Диаграмма третьего уровня потоков данных. Процесс «Отправить письмо»

1.6.14. Диаграмма классов для службы отправки сообщений

Служба отправки сообщений разрабатываемого программного комплекса представляет собой фоновый процесс, запущенный на сервере.

Этот процесс циклически получает из базы данных сообщение из очереди и пытается отправить его с помощью интерфейса ISenderService. Класс SenderService реализует указанный интерфейс и последовательно передает сообщение классам через интерфейс ISender.

Классы, реализующие ISender:

1) SmtpSender;
2) SendpulseSender;
3) SendGrideSender.

В зависимости от результата SenderService формирует и возвращает соответствующий объект класса SendingResult.

Если результат положительный, то WorkerMain обновляет письмо: удаляет его из очереди и меняет его статус.

Диаграмма классов представлена на рис. 4.7.

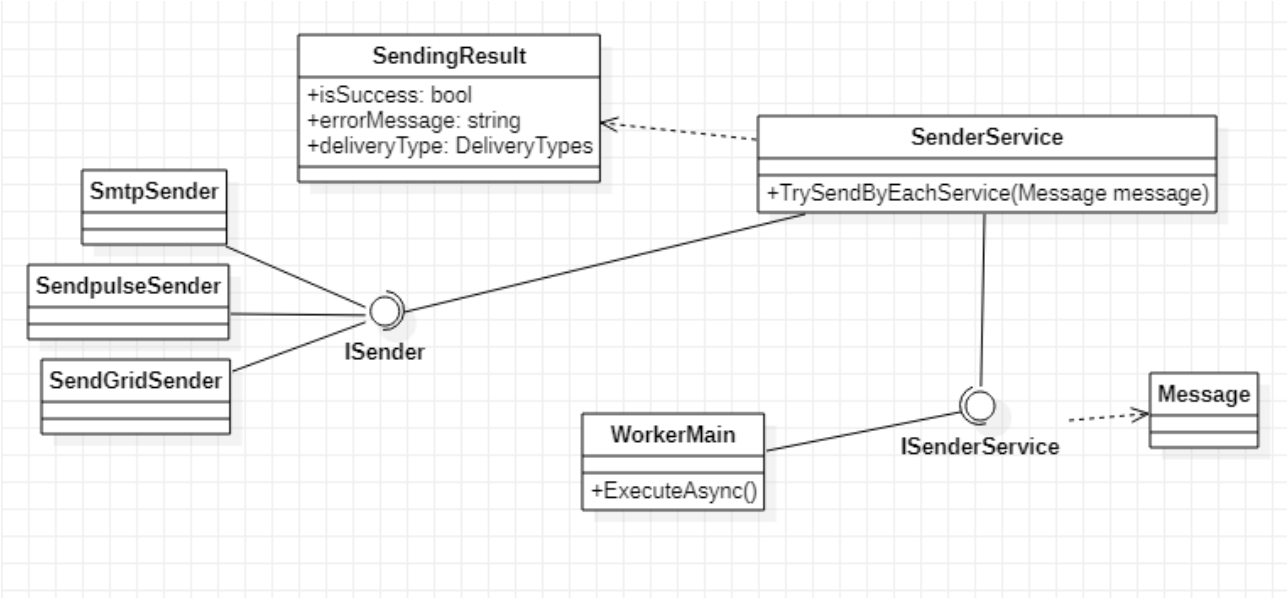

Рис. 4.7. Диаграмма классов

4.5. 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 интерфейсов создания сообщения, списка сообщений и списка доступных сервисов доставки представлены на рис. 4.8-4.11.

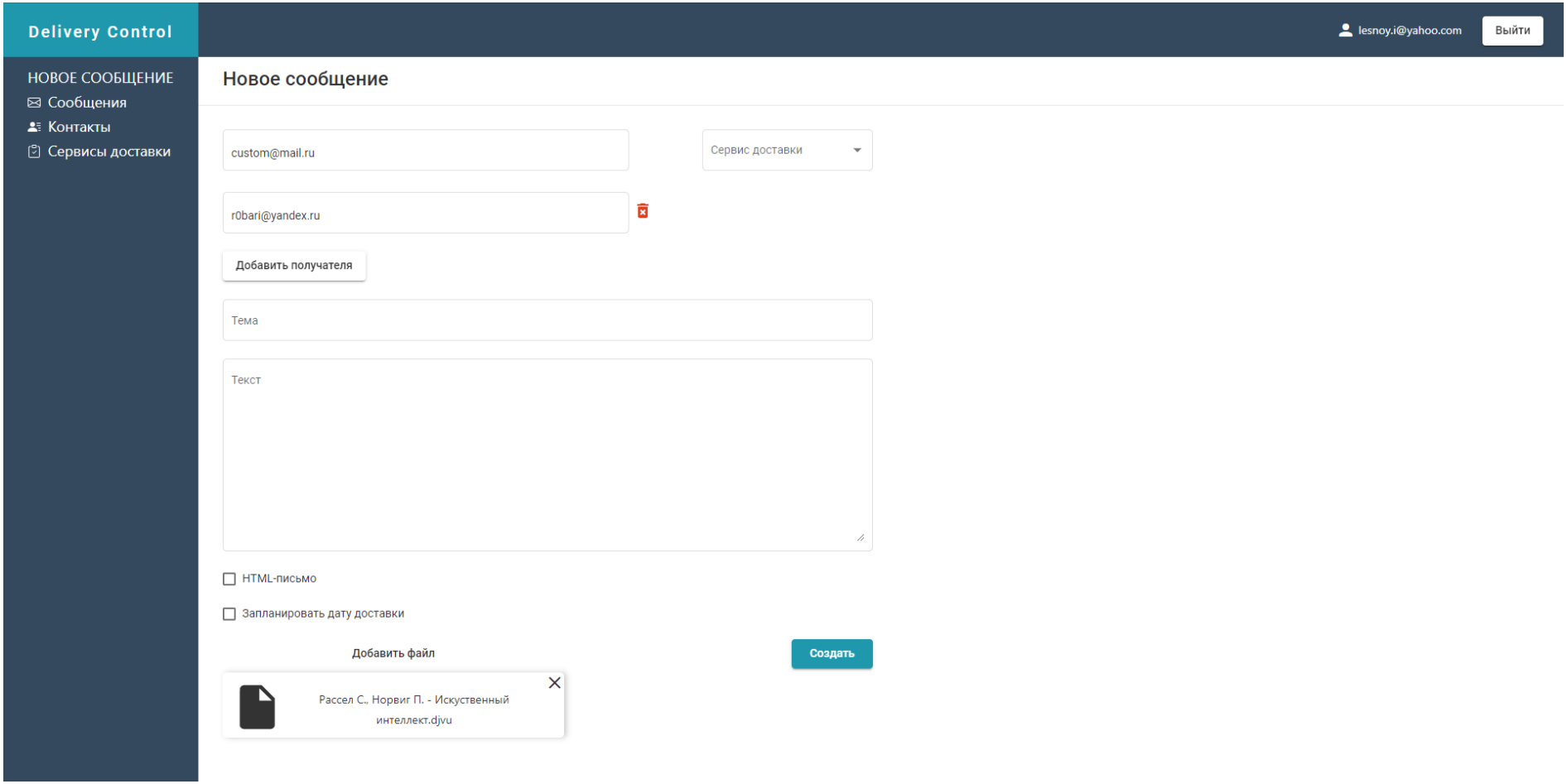

Рис. 4.8. Страница с доступными сервисами доставки

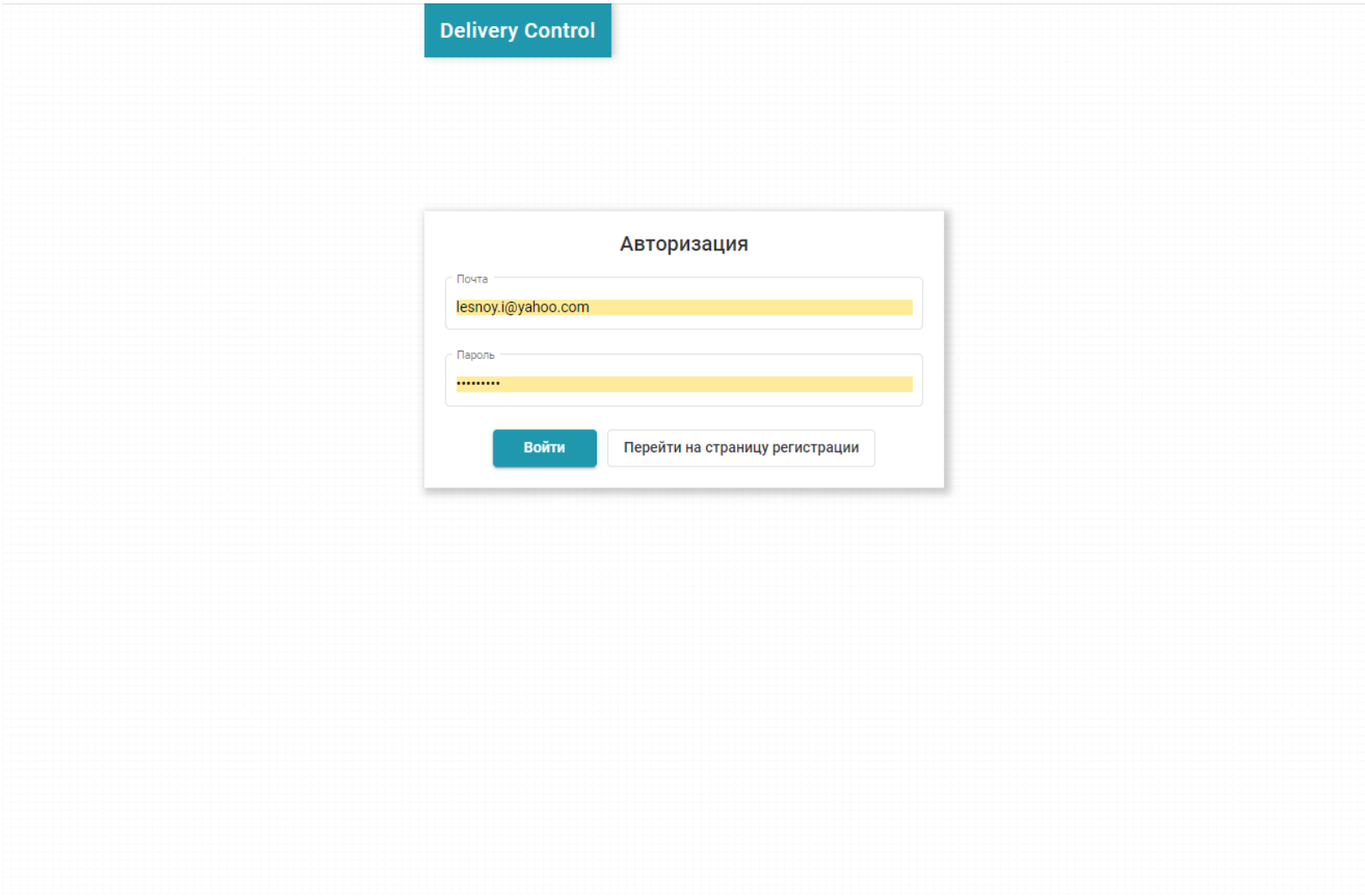

Рис. 4.9. Форма авторизации

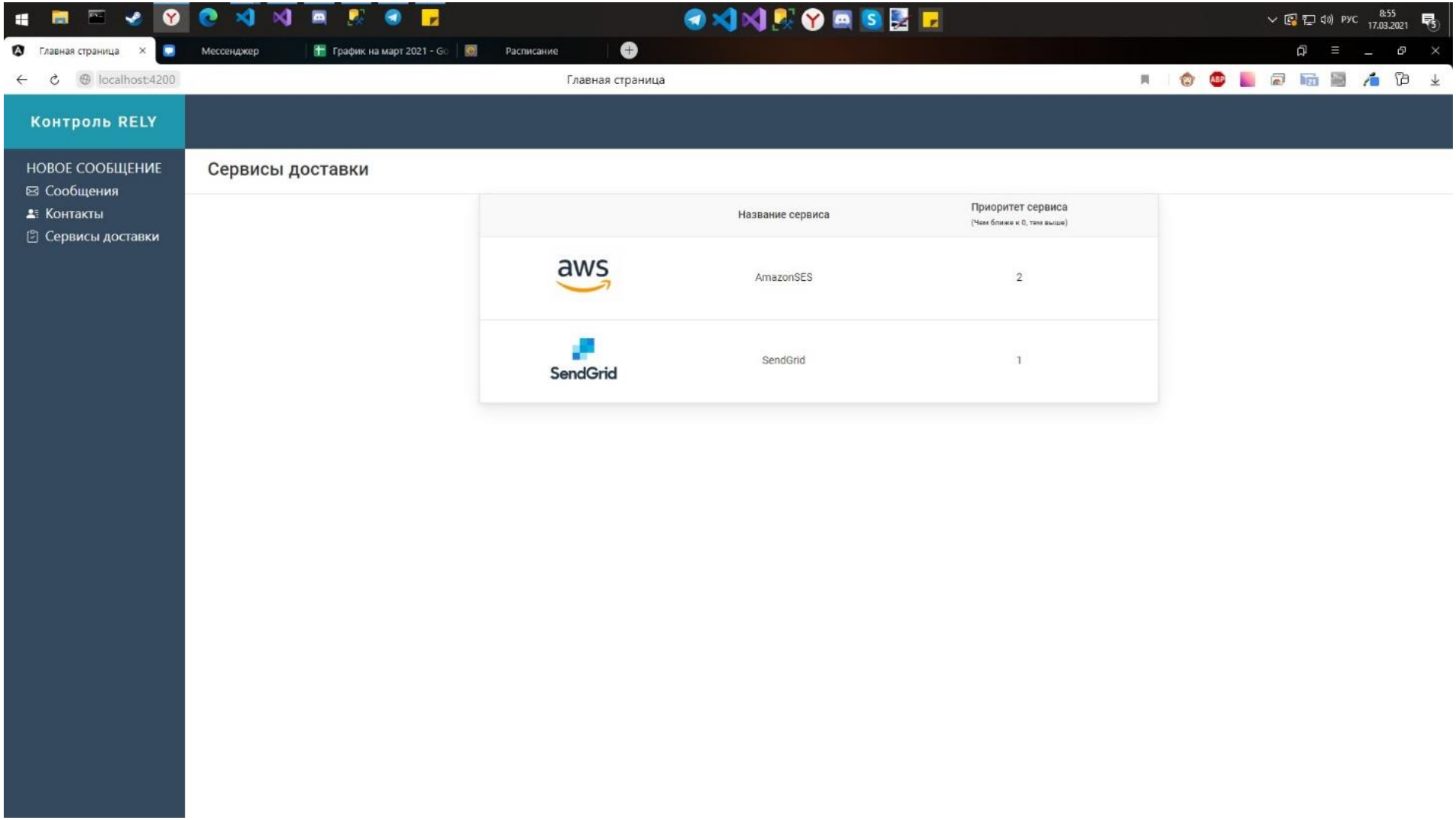

Рис. 4.10. Страница с доступными сервисами доставки

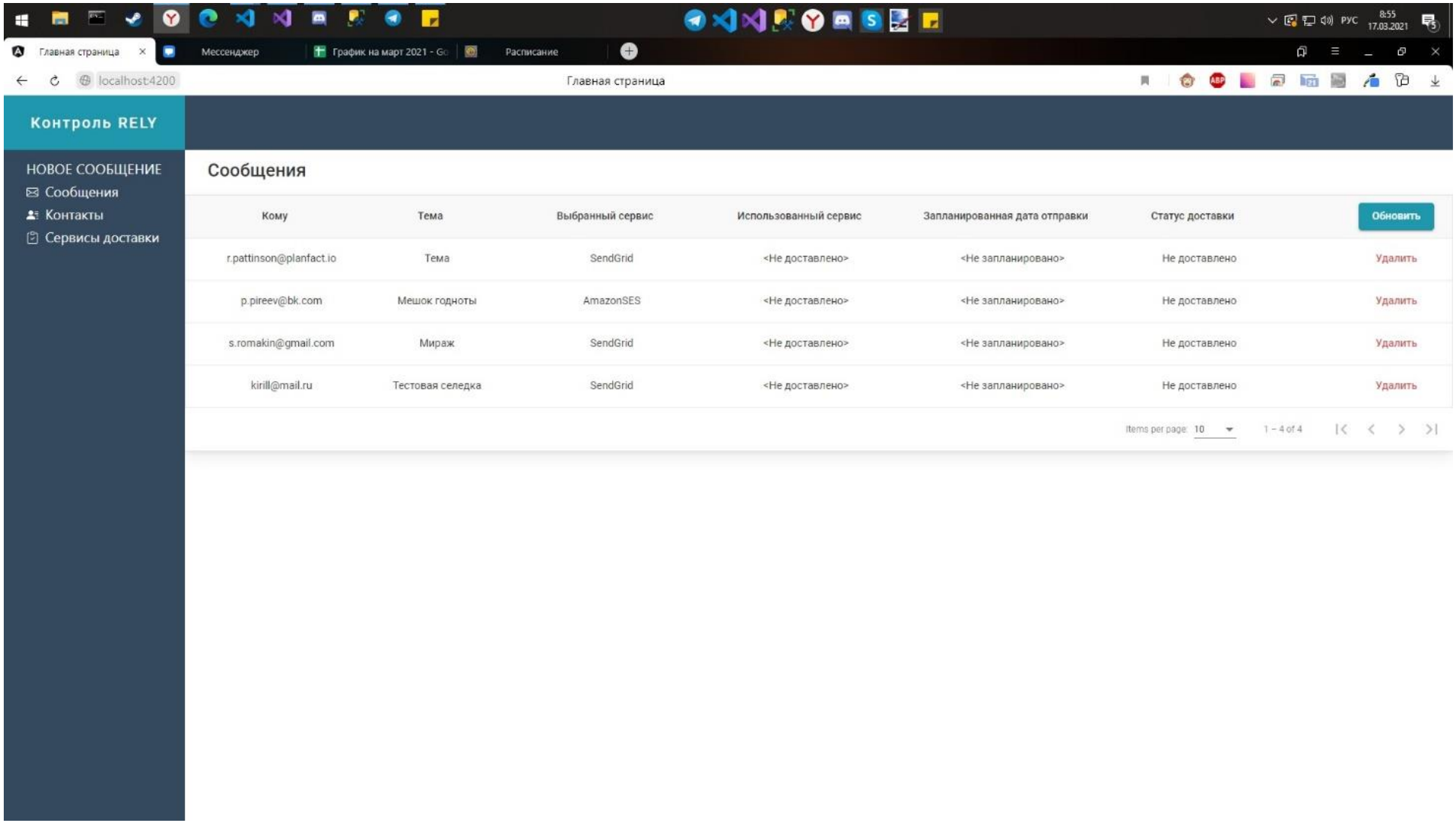

Рис. 4.11. Журнал сообщений

4.6. Проектирование системы балансировки нагрузки

В случае, когда на сервер одновременно поступает большое количество запросов, время обработки запросов закономерно вырастает, что делает использование программного комплекса менее удобным. Чтобы избежать этой проблемы, можно использовать несколько идентичных серверов и распределять по ним поступающие запросы по определенному правилу.

В таком случае один из серверов отвечает за прием запросов от пользователей и перенаправление запросов подходящим серверам для обработки (распределительный сервер).

Сервера, обрабатывающие запросы, принимают запрос от распределительного сервера, обрабатывают полученный запрос и возвращают ответ распределительному серверу. А тот в свою очередь возвращает полученный ответ пользователю.

Распределительный сервер выбирает, какому из серверов передать запрос на обработку по определенному алгоритму. Рассмотрим несколько доступных алгоритмов балансировки нагрузки.

Round Robin. Самый простой алгоритм, при котором поступающие запросы распределяются по серверам поочередно и циклически. Таким образом, каждый сервер будет обрабатывать равное количество запросов. Возможна ситуация, когда один сервер получит ряд очень простых запросов, и его ресурсы будут простаивать, а другой сервер получит такое же количество очень длительных запросов, из-за чего будет перегружен.

Weighted Round Robin. Модификация предыдущего алгоритма, в которой каждому обрабатывающему запросы серверу присваивается какой-то удельный вес. И распределительный сервер передает запросы серверам пропорционально их весам.

Least Connections. Алгоритм балансировки, при котором распределительный сервер передает очередной запрос тому серверу, на который в данный момент времени назначено наименьшее число запросов.

В целях улучшения гибкости и конфигурируемости в разрабатываемом программном комплексе возможно переключение между тремя описанными выше алгоритмами.

Схема взаимодействия серверов представлена на рис. 4.12.

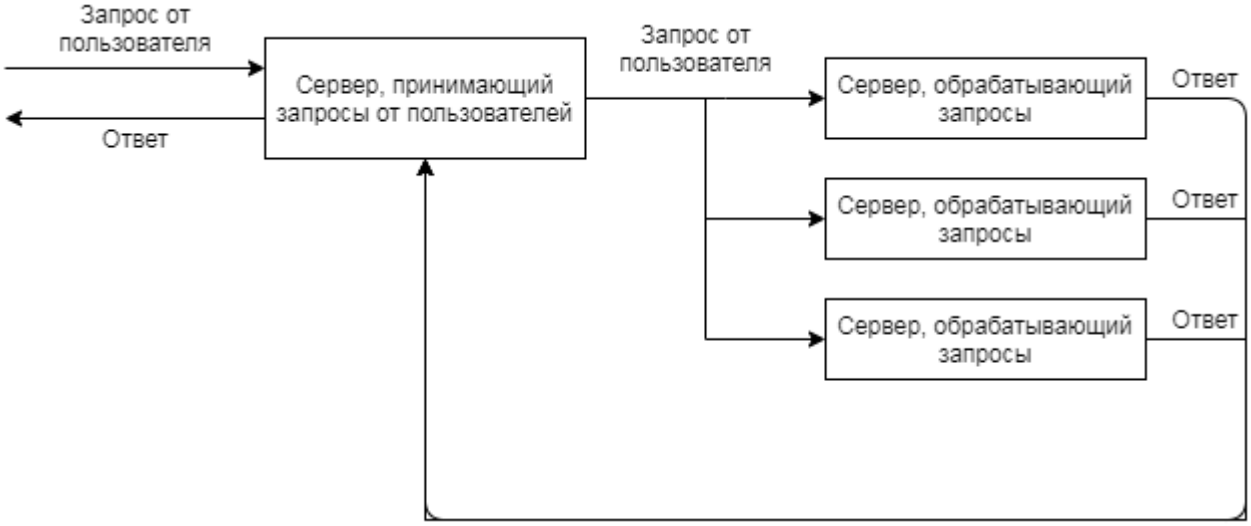

Рис. 4.12. Схема взаимодействия серверов

5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО КОМПЛЕКСА

Для проверки корректности разработанного программного комплекса планируется проведение следующих видов тестирования:

4) тестирование корректности работы WEB API методом черного ящика;
5) модульное тестирование;
6) кроссбраузерное тестирование интерфейсов пользовательской части.

5.1. Тестирование корректности работы 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 будут проверены следующие контроллеры WEB API:

1) MessagesController;
2) UsersController;
3) ContactsController;
2. DeliveryQueuesController;
3. AttachedFilesController;
4. AuthController.

Пример тестов представлены в таблице 5.1.

Таблица 5.1.

Тесты для тестирования методом черного ящика

№ 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

№ теста	Входные данные	Ожидаемый результат	Соответствие полученного результата ожидаемому

2	Запрос:
	POST
	https://localhost:44306/api/auth/sign-in
	Body:
	{
	"email": "r0bari@yandex.ru",
	"password": "Eviguf@77"
	}
		{
		"data": {
		"userId": 4,
		"userName": "Evgeny Sukharev",
		"email": "r0bari@yandex.ru",
		"isActive": true,
		"role": 2
		},
		"isSuccess": true,
		"errorMessage": null
		}
			Да

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

5.2. 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 будут 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 прилож. 3 в листинге 3.6.

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 прилож. 3 в листинге 3.7.

5.3. Кроссбраузерное тестирование интерфейсов пользовательской части

Тестирование кроссбраузерности — вид тестирования, направленный на поддержку и правильное полное отображение программного продукта в разных браузерах, мобильных устройствах, планшетах, экранах различного размера [14].

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. 5.1) и Opera (рис. 5.2).

По итогам данного тестирования были выявлены определенные особенности отображения интерфейсов в некоторых браузерах. Например, последний столбец в компоненте таблицы из библиотеки Material отображался некорректно в браузере Mozilla Firefox, из-за чего таблица становилась шире, в браузере появлялась горизонтальная полоса прокрутки, и не было видно столбца с доступными действиями.

Данная ошибка была исправлена изменением стилей таблицы, после чего поведение таблицы во всех браузерах стало идентичным и корректным.

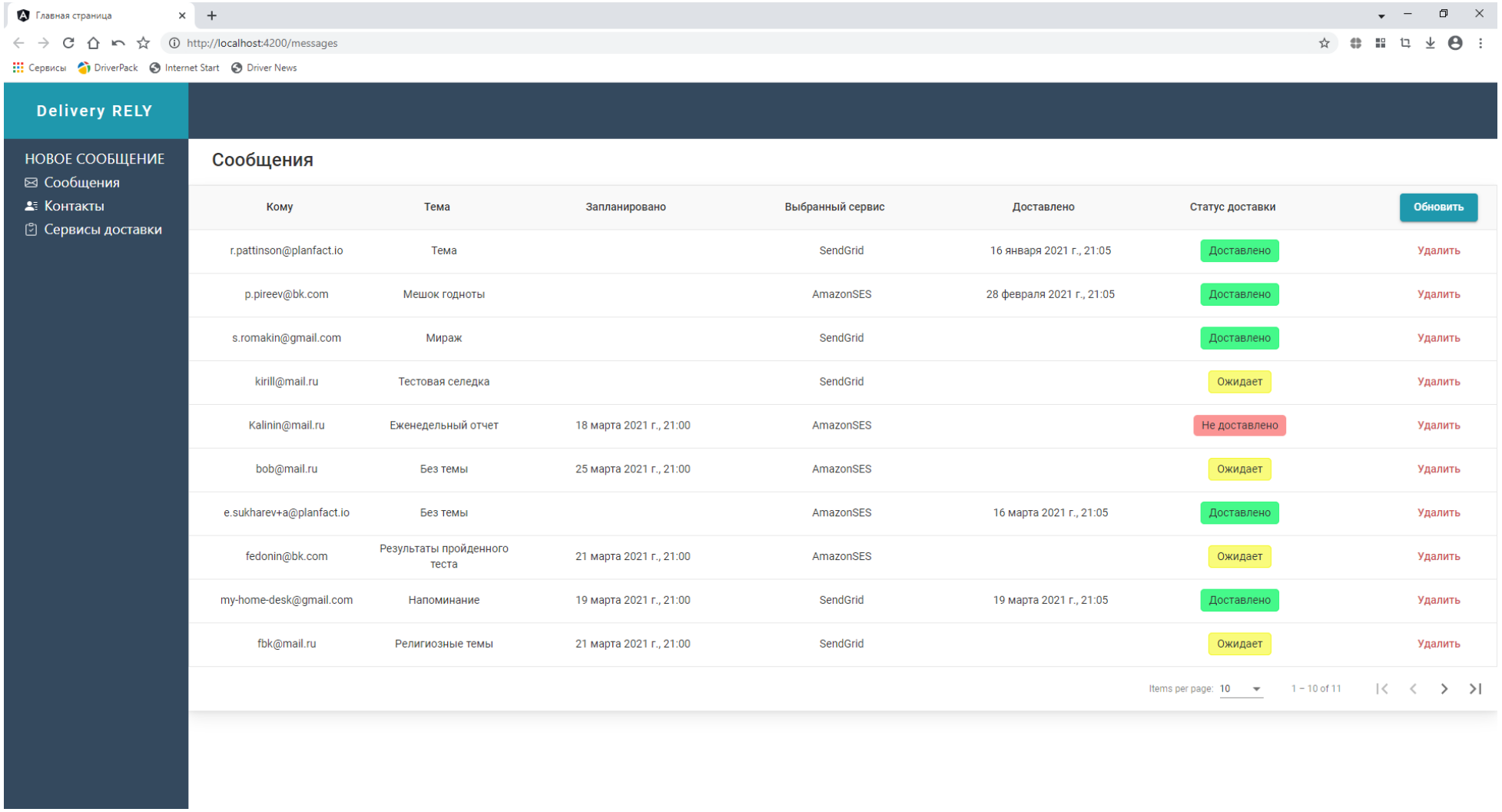

Рис. 5.1. Список сообщений в Google Chrome

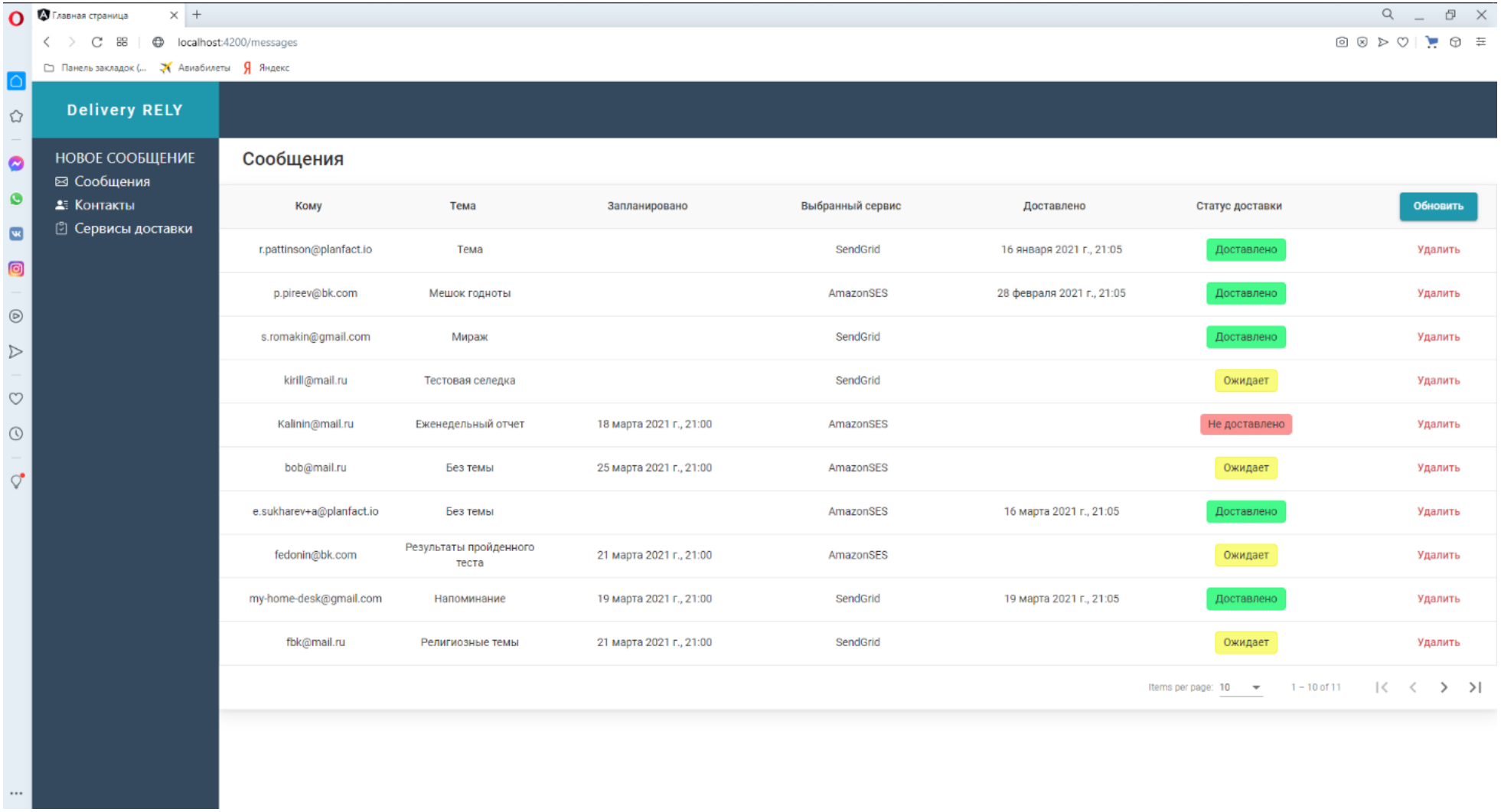

Рис.5.2. Страница сообщений в Opera

6. ОРГАНИЗАЦИОННАЯ ЧАСТЬ

Тема дипломной работы связана с разработкой программного комплекса управления надежной отправки электронных писем, предусматривающей эксплуатацию вычислительной техники и решение задач в помещениях, оборудованных ПЭВМ.

Все требования к помещению и используемому компьютерному оборудованию отражены в Санитарных правилах и нормах СанПиН 2.2.2/2.4.1340-03 "Гигиенические требования к персональным электронно-вычислительным машинам и организации работы" [28] и приведены в прилож. 2.

При проектировании рабочего места должна быть решена проблема освещения как искусственного, так и естественного.

Заданные параметры:

1) тип лампы – люминесцентная;
2) тип светильника – ЛД;
3) освещенность – E = 300 лк;
4) высота помещения – H = 3,3 м;
5) длина помещения – a = 9 м;
6) ширина помещения – b = 4 м;
7) высота подвеса светильника – hp = 2,5 м.

Результаты расчетов освещения обобщены в таблице 6.1, схема расположения светильников представлена на рис. 6.1, а сами расчеты представлены в прилож. 3.

Таблица 6.1

Результаты расчётов параметров осветительной установки

№ п/п	Тип лампы	Световой поток лампы **Ф**, лм	Количество светильников		Отклонение $\mathbf{n_{пр}}$ от $\mathbf{n_{расч}}$, %	Мощность лампы, Вт	Полная мощность **N**, Вт
			расчётное $\mathbf{n_{расч}}$	принятое $\mathbf{n_{пр}}$			
1	ЛД-65	3390	6,37	6	-5,80	65	780
2	ЛД-80	3865	5,58	6	7,53	80	960

Вывод: оптимальным вариантом осветительной установки является ЛД-65, так как значение её полной потребляемой мощности является меньше, у осветительной установки ЛД-80.

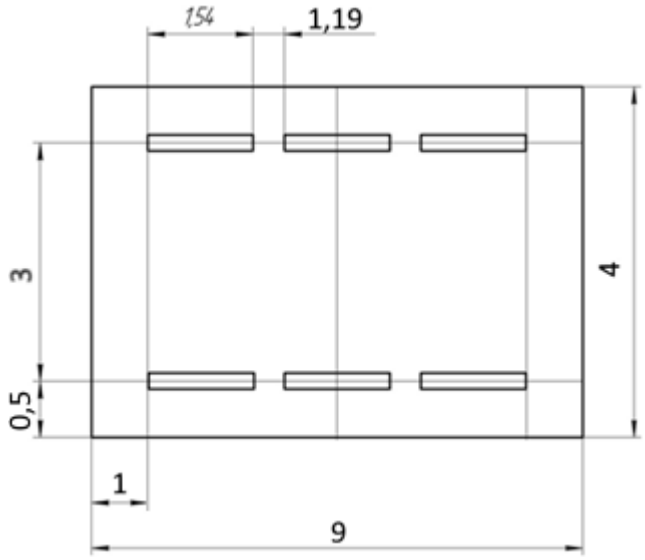

Рис. 6.1. Схема расположения светильников

СПИСОК ЛИТЕРАТУРЫ

1. Что такое web-интерфейс: [Электронный ресурс]. URL: https://semantica.in/blog/veb-interfejs.html/.
2. Проектирование веб-API / Перевод с английского Д.А. Беликова: [Текст] / Арно Лоре – Оформление, издание, ДМК Пресс, 2020.
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. UML основы. Краткое руководство по стандартному языку объектного проектирования: [Текст] / Мартин Фаулер – Санкт-Петербург, 2005.
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. Кроссбраузерное 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. React в действии: [Текст] / Марк Тиленс Томас – СПб.: Питер, 2019.
19. Vue.js в действии [Текст] / Эрик Хэнчетт, Бенджамин Листоун – СПб: Питер, 2019.
20. Большая книга CSS: [Текст] / Дэвид Макфарланд – СПб.: Питер, 2016.
21. Современный учебник JavaScript: [Электронный ресурс]. URL: https://learn.javascript.ru/
22. Язык программирования C# 7 и платформы .NET и .NET Core [Текст] / Троелсен, Джепикс – 2018.
23. Software Engineering for Internet Applications by Eve Andersson [Текст] / Philip Greenspun, Andrew Grumet – 2006.
24. Microsoft SQL Server: [Текст] / Душан Петкович – Санкт-Петербург «БХВ-Петербург» – 2013.
25. My SQL 5.0. Библиотека программиста: [Текст] / Виктор Гольцман – Питер, Санкт-Петербург, 2010.
26. Техническая документация по PostgreSQL: [Электронный ресурс]. URL: https://postgrespro.ru/docs/postgresql
27. Официальный сайт Senpulse: [Электронный ресурс]. URL: https://sendpulse.com/ru
28. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. СанПиН 2.2.2/2.4.1340-03 – М.: ДЕАН, 2003. – 128 с.

ПРИЛОЖЕНИЯ

Приложение 1

Организационная структура проекта

Перед началом разработки любой программной системы необходимо выяснить, насколько целесообразна ее разработка, как с точки зрения полезности, так и с точки зрения экономической эффективности. В технико-экономической части работы рассматриваются вопросы организации работ по созданию и внедрению программной системы, а также приводится расчёт ее себестоимости.

Организационная структура проекта (OBS) приведена на рис.1.1.

Рис. 1.1. Организационная структура проекта

Календарный план проекта

Для оценки расходов на реализацию проекта в числе прочих необходимо определить временные затраты на его реализацию. Для определения временных затрат проекта необходимо разработать календарный план проекта. Реализуемый проект является типовым для Компании, исходя из этого, был сформирован состав работ проекта, определена их длительность, а также распределение ресурсов по ним. При разработке календарного плана были учтены ограничения и допущения, накладываемые на проект Заказчиком.

Для реализации проекта необходимо выполнить ряд работ, представленных ниже.

1. Сбор требований Заказчика к разрабатываемому ПО.
2. Разработка и согласование ТЗ.
3. Разработка технического проекта.
4. Разработка моделей бизнес-процессов.
5. Проектирование интерфейсов.
6. Проектирование иерархии пользовательских интерфейсов.

7. Проектирование архитектуры серверной части.
8. Определение методов WEB API и их сигнатур.
9. Разработка тестов для серверной части.
10. Разработка тестов для клиентской части.
11. Разработка серверной части.
12. Разработка клиентской части.
13. Тестирование серверной части.
14. Тестирование взаимодействия серверной части и клиентской части.
15. Тестирование работы клиентской части.
16. Кроссбраузерное тестирование интерфейсов.
17. Разработка пользовательской документации.
18. Внедрение ПО.

Распределение человеческих ресурсов по работам проекта и степень их загрузки приведены в табл. 1.1.

Таблица 1.1

Структура общего времени на создание программного продукта

№ этапа	Этап работ	Ответственные исполнители (занятость на этапе)	Длительность, дней
1	Сбор требований	• Менеджер проекта [40%] • Бизнес-аналитик [100%]	10
2	Разработка и согласование ТЗ	• Менеджер проекта [50%] • Бизнес-аналитик [100%] • Системный аналитик [40%]	10
3	Разработка технического проекта	• Менеджер проекта [30%] • Бизнес-аналитик [60%] • Системный аналитик [100%]	4
4	Разработка моделей бизнес-процессов	• Бизнес-аналитик [100%]	7
5	Проектирование интерфейсов	• Менеджер проекта [30%] • Дизайнер [100%] • Бизнес-аналитик [40%]	5
6	Проектирование иерархии пользовательских интерфейсов	• Менеджер проекта [30%] • Бизнес-аналитик [100%] • Дизайнер [50%]	2

№ этапа	Этап работ	Ответственные исполнители (занятость на этапе)	Длительность, дней
7	Проектирование архитектуры серверной части	• Системный аналитик [100%] Менеджер проекта [30%]	5
8	Определение методов WEB API и их сигнатур	• Менеджер проекта [20%] • Системный аналитик [100%]	2
9	Разработка тестов для серверной части	• Менеджер проекта [30%] • Тестировщик [50%] • Backend-разработчик[100%]	5
10	Разработка тестов для клиентской части	• Менеджер проекта [30%] • Тестировщик [50%] • Frontend-разработчик[100%]	5
11	Разработка серверной части	• Менеджер проекта [20%] • Backend-разработчик [100%]	20
12	Разработка клиентской части	• Менеджер проекта [20%] • Frontend-разработчик [100%]	30
13	Тестирование серверной части	• Менеджер проекта [20%] • Тестировщик [100%] • Backend-разработчик [30%]	10
14	Тестирование взаимодействия серверной части и клиентской части	• Менеджер проекта [20%] • Тестировщик [100%]	5
15	Тестирование работы клиентской части	• Менеджер проекта [20%] • Тестировщик [100%] • Frontend-разработчик [60%]	10
16	Кроссбраузерное тестирование интерфейсов	• Менеджер проекта [20%] • Тестировщик [100%] • Frontend-разработчик [60%]	10
17	Разработка пользовательской документации	• Бизнес-аналитик [100%] • Менеджер проекта [20%] • Системный аналитик [100%]	4
18	Внедрение ПО	• Менеджер проекта [50%] • Специалист по внедрению [100%]	3

При реализации данного проекта работы выполняются последовательно. Диаграмма Ганта приведена на рис. 1.2. Графическое представление диаграммы Ганта показано на рис. 1.3.

	Название задачи	Длительность	Начало	Окончание	Пр
1	Сбор требований	10 дней	Пт 01.01.21	Чт 14.01.21	
2	Разработка и согласование ТЗ	10 дней	Пт 15.01.21	Чт 28.01.21	1
3	Разработка технического проекта	4 дней	Пт 29.01.21	Ср 03.02.21	2
4	Разработка моделей бизнес-процессов	7 дней	Чт 04.02.21	Пт 12.02.21	3
5	Проектирование интерфейсов	5 дней	Пн 15.02.21	Пт 19.02.21	4
6	Проектирование иерархии пользовательских интерфейсов	2 дней	Пн 22.02.21	Вт 23.02.21	5
7	Проектирование архитектуры серверной части	5 дней	Пн 15.02.21	Пт 19.02.21	4
8	Определение методов WEB API и их сигнатур	2 дней	Пн 22.02.21	Вт 23.02.21	7
9	Разработка тестов для серверной части	5 дней	Ср 24.02.21	Вт 02.03.21	8
10	Разработка тестов для клиентской части	5 дней	Ср 03.03.21	Вт 09.03.21	9
11	Разработка серверной части	20 дней	Ср 03.03.21	Вт 30.03.21	9
12	Разработка клиентской части	25 дней	Ср 10.03.21	Вт 13.04.21	10
13	Тестирование серверной части	10 дней	Ср 31.03.21	Вт 13.04.21	11
14	Тестирование взаимодействия серверной части и клиентской части	5 дней	Ср 14.04.21	Вт 20.04.21	12;11
15	Тестирование работы клиентской части	10 дней	Ср 21.04.21	Вт 04.05.21	14
16	Кроссбраузерное тестирование интерфейсов	10 дней	Ср 05.05.21	Вт 18.05.21	15
17	Разработка пользовательской документации	4 дней	Ср 19.05.21	Пн 24.05.21	16;13
18	Внедрение ПО	10 дней	Вт 25.05.21	Пн 07.06.21	17

Рис. 1.2. Табличное представление Диаграммы Ганта

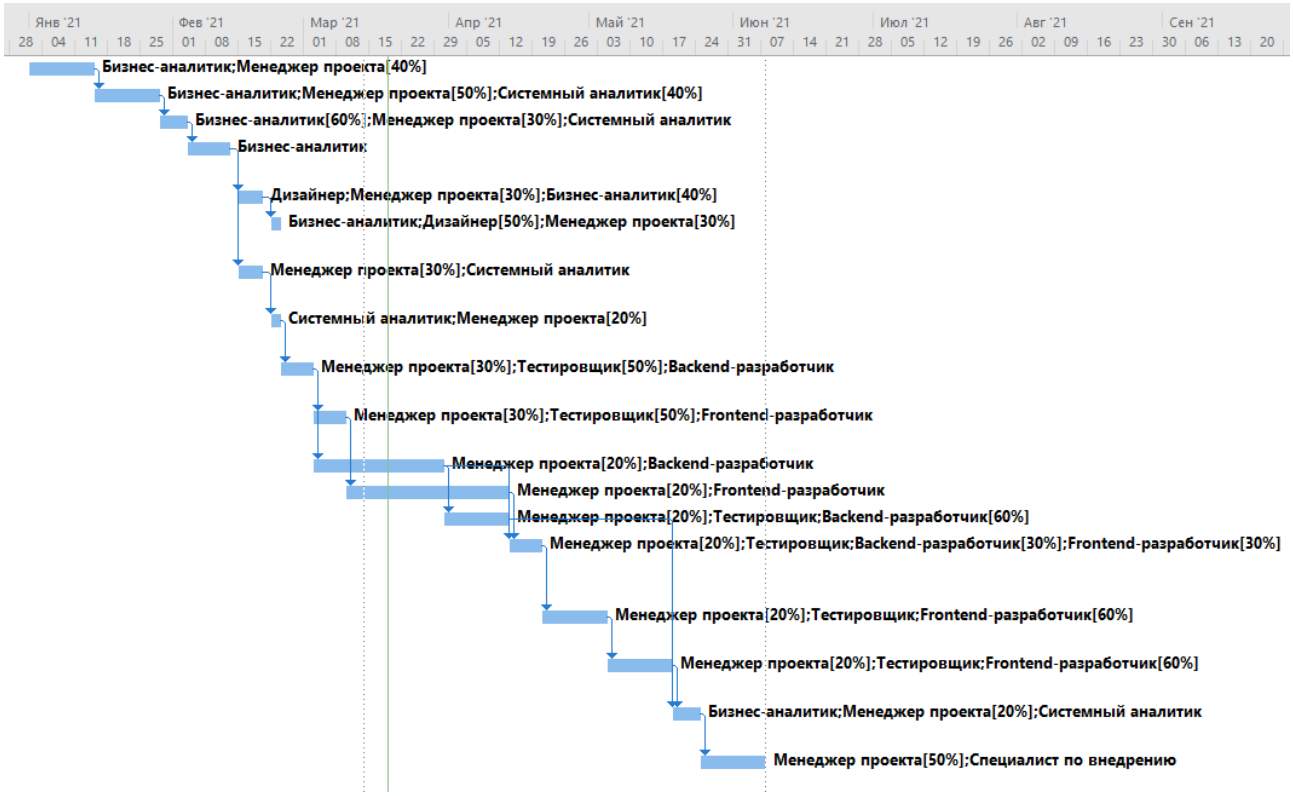

Рис. 1.3. Графическое представление Диаграммы Ганта

Исходя из длительности работ и коэффициента загрузки членов проектной команды, определим их трудозатраты при реализации проекта (табл. 1.2).

Таблица 1.2

Трудозатраты членов проектной команды

№	Исполнитель	Трудозатраты, человеко-часов
1	Менеджер проекта	280,4
2	Бизнес-аналитик	230,4
3	Системный аналитик	144
4	Тестировщик	256
5	Специалист по внедрению	80
6	Дизайнер	48
7	Frontend-разработчик	348
8	Backend-разработчик	260

Расчёт затрат на разработку продукта

Расчет затрат на создание и внедрение ПО включает следующие составляющие с последующим их графическим представлением в виде круговой диаграммы:

- заработная плата исполнителей работ по проекту – $ЗП_{осн}$;
- дополнительная заработная плата $ЗП_{доп}$;
- отчисления на социальные нужды (страховые взносы) – $Н_{зп}$;
- арендные платежи за производственные (офисные) помещения – $А_{пм}$;
- амортизация используемых основных средств и нематериальных активов – $А$;
- расходы на модернизацию и приобретение основных средств – $Р_{мод}$;
- расходы на приобретение необходимого ПО – $Р_{ПО}$;
- расходы на интернет, связь – $Р_{тел}$;
- расходы на канцелярские товары и расходные материалы – $Р_{р.м.}$;
- прочие расходы – $П_{р.р.}$.

Далее представлен расчёт заработной платы исполнителей работ по созданию программного продукта.

Основная ЗП определяется по формуле:

$$ЗП_{осн} = \frac{М \cdot Т}{Ч_р \cdot t_{р.д.}}\left(1+\frac{П}{100}\right), \text{ руб,} \qquad (1.1.)$$

где $М$ – месячная зарплата (*руб.*), $Т$ – общие трудозатраты (*чел.-ч*), $Ч_р$ – число рабочих дней в месяц, $t_{р.д.}$ – продолжительность рабочего дня в часах, $П$ – процент премии. В данной работе $Ч_р$ = 21 *день*, $t_{р.д.}$=8*ч*, $П$=0.

Значение месячной заработной платы (М), суммарные трудозатраты членов, а также рассчитанная по формуле 1 основная заработная плата проектной команды приведены в табл. 1.3.

Таблица 1.3

Основная заработная плата членов проектной команды

№	Исполнитель	Месячная заработная плата (М), руб.	Трудозатраты, человеко-часов	$ЗП_{осн}$, руб.
1	Менеджер проекта	56 000	280,4	93 466
2	Бизнес-аналитик	75 000	230,4	102 857
3	Системный аналитик	75 000	144	64 285
4	Тестировщик	53 000	256	80 762
5	Специалист по внедрению	50 000	80	23 810
6	Дизайнер	41 000	48	11 714
7	Frontend-разработчик	85 000	348	176 071
8	Backend-разработчик	85 000	260	131 547

Суммарное значение основной заработной платы проектной команды на период реализации проекта составит 683 587 (руб.).

Так как дополнительную заработную плату берем за 15% от основной, общая заработная плата составит 786 125 (руб.).

Проект реализуется в небольшой ИТ-компании, где доля вспомогательного и административного персонала по отношению к основному персоналу не велика. Большая часть административного персонала задействована в проектной деятельности в качестве руководителей проекта. Кадровый учет, бухгалтерский и налоговый учет в Компании отдан на аутсорсинг. Затраты на аутсорсинг войдут в прочие расходы. В связи с этим примем заработную плату обслуживающего персонала равной 0 руб.

Теперь можно рассчитать величину отчислений на социальные нужды (страховые взносы), которые начисляются на заработную плату и в 2021 г. для организаций, осуществляющих деятельность в области информационных технологий, составляют 14% по выплатам в пределах 568 тыс. руб. Структура отчислений на социальные нужды (страховые взносы) приведена в табл. 1.4.

Таблица 1.4

Структура отчислений на социальные нужды (страховые взносы)

Пенсионный фонд Российской Федерации	8,0%
для лиц 1966 года рождения и старше	
страховые взносы на страховую часть трудовой пенсии	8,0%
для лиц 1967 года рождения и моложе	
страховые взносы на страховую часть трудовой пенсии	2,0%
страховые взносы на накопительную часть трудовой пенсии	6,0%
Фонд социального страхования Российской Федерации	2,0%
Федеральный фонд обязательного медицинского страхования	4,0%

Таким образом, $H_{зп}$= 95 702 (*руб.*).

Далее приведены расчеты по арендным платежам за производственные (офисные) помещения.

Компания, реализующая проект по разработке и внедрению ПО арендует офисные помещения в г. Брянск.

Стоимость аренды составляет 500 руб/м^2 в месяц.

Арендная плата включает в себя оплату как площади занимаемых Компанией помещений, так и электроэнергии, отопления, водоснабжения, кондиционирования и уборки помещений, вывоза и утилизации технико-бытовых отходов, парковочных мест на автостоянке.

На каждого члена проектной команды приходится 4,5 м^2 арендуемого офисного помещения. На период данного проекта члены проектной команды в других проектах не задействованы.

Исходя из изложенного выше, затраты на аренду помещений, отнесенные на проект составят $A_{пм}$ = *500*8*4,5/21*157=134 572 (руб.).*

При реализации проекта по разработке и внедрению ПО для автоматизации внутри складской логистики задействованы следующие основные средства:

- 7 персональных компьютеров в сборе первоначальной стоимостью 45 000 (руб.) каждый;
- 1 персональный компьютер в сборе первоначальной стоимостью 60 000 (руб.)

- Срок полезного использования для задействованных в проекте основных средств определен в 3 года. Метод начисления амортизации – линейный.

Амортизационные отчисления для персонального компьютера на 1 месяц составят

45 000 / 36 = 1 250 (руб.)

И

60 000 / 36 = 1 667 (руб.)

Амортизационные отчисления по ОС, относящиеся на проект составят:

$$A_{\text{ос}} = 7{,}48 \cdot (7 \cdot 1250 + 1667) = 72\,919 \text{ (руб.)}.$$

Данное ПО принимается Компанией к учету как расходы будущих периодов со сроком списания 3 года. Метод списания – линейный.

В качестве ОС используется платное ПО Windows 10 – 8 лицензий общей стоимостью 31 200 (руб.). В качестве сервера БД используется платное ПО MS SQL Server. Данное ПО уже было приобретено ранее и полностью списано до начала проекта.

Амортизационные отчисления по РБП, относящиеся на проект составят:

$$A_{\text{РБП}} = 31\,200 \cdot 7{,}48 / 36 = 6067 \text{ (руб.)}.$$

Суммарные амортизационные отчисления составят: А= 78 986 руб.

При реализации проекта по разработке и внедрению ПО не планируется приобретение новых и модернизация существующих основных средств.

При реализации проекта не планируется приобретение ПО.

Так как в Компании, реализующей проект не производится биллинг и тарификация телекоммуникационных услуг в разрезе сотрудников, затраты на интернет и связь войдут в прочие затраты, рассчитываемые как процент от прямых затрат.

Затраты на расходные материалы берутся по факту и составляют $P_{р.м.}$= 4 200 (*руб.*). К данным затратам относятся затраты на канцтовары, тонер и бумагу для принтера и т.д.

Прочие расходы составляют 30% от суммы следующих элементов структуры затрат: $ЗП_{осн}$, $ЗП_{доп}$, $Н_{зп}$, $А_{пм}$, $А$, $Р_{мод}$, $Р_{ПО}$, $Р_{тел}$ и $Р_{р.м.}$.

$$П_{р.р.} = 0.3(ЗП_{осн} + ЗП_{доп} + Н_{зп} + А_{пм} + А + Р_{мод} + Р_{ПО} + Р_{тел} + Р_{р.м})$$

Таким образом, $П_{р.р.}$ = 299 114 (*руб.*).

Расчёт себестоимости программного продукта

В себестоимость программного продукта входят следующие элементы: $ЗП_{осн}$, $ЗП_{доп}$, $Н_{зп}$, $А_{пм}$, $А$, $Р_{мод}$, $Р_{ПО}$, $Р_{тел}$, $Р_{р.м.}$ и $П_{р.р.}$. В итоге:

$С_{п.п.}$ = 786125 + 95702 + 134572 + 78986 + 4200 + 299114 = 1 398 699 (*руб.*).

Приложение 2

Вредные факторы при использовании программного комплекса

Работа оператора ПЭВМ относится к категории работ, связанных с опасными и вредными условиями труда. В процессе труда на оператора ПЭВМ оказывают действие следующие опасные и вредные производственные факторы:

1. Физические**:**
    - повышенные уровни электромагнитного, рентгеновского, ультрафиолетового и инфракрасного излучения;
    - повышенный уровень статического электричества;
    - повышенные уровни запыленности воздуха рабочей зоны;
    - пониженная или повышенная влажность и подвижность воздуха рабочей зоны;
    - повышенный уровень шума;
    - повышенный или пониженный уровень освещенности;
    - повышенный уровень прямой и отраженной блесткости;
    - повышенный уровень ослепленности;
    - неравномерность распределения яркости в поле зрения;
    - повышенная яркость светового изображения;
    - повышенный уровень пульсации светового потока;
    - повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека.
2. Химические**:**
    - повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола, формальдегида и полихлорированных бифенилов.
    - Психофизиологические**:**
    - напряжение зрения и внимания;
    - интеллектуальные и эмоциональные нагрузки;
    - длительные статические нагрузки и монотонность труда;

- большой объем информации обрабатываемой в единицу времени;
- нерациональная организация рабочего места.

3. Биологические**:** повышенное содержание в воздухе рабочей зоны микроорганизмов.

Освещенность рабочего места

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Допускается установка светильников местного освещения для подсветки документов. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ).

В зависимости от типа светильника выбираем коэффициент λ. Он определяет такое соотношение максимального расстояния между светильниками $L_{св(max)}$ и высотой их подвеса над рабочей поверхностью h_p, которое обеспечит равномерность освещения в помещении.

$$L_{св(max)} = \lambda \cdot h_p = 1{,}4 \cdot 2{,}5 = 3{,}5\text{м}.$$

Определяем расстояние $L_{1(max)}$ от стены до первого ряда светильников: – при наличии рабочих мест у стены:

$$L_{1(max)} = (0{,}2 \ldots 0{,}3) \cdot L_{св(max)} = 0{,}25 \cdot 3{,}5 = 0{,}875\text{м}.$$

Определяем общее число рядов светильников (по ширине помещения):

$$n_{ш(min)} = \frac{b - 2 \cdot L_{1(max)}}{L_{св(max)}} + 1 = \frac{4 - 2 \cdot 0{,}875}{3{,}5} + 1 = 1{,}64 \approx 2;$$

Определяем число светильников в ряду (по длине помещения):

$$n_{д(min)} = \frac{a - 2 \cdot L_{1(max)}}{L_{св(max)}} + 1 = \frac{9 - 2 \cdot 0{,}875}{3{,}5} + 1 = 3.07 \approx 3;$$

где a – длина, b – ширина помещения, для которого рассчитывается система освещения.

Полученные результаты округляем до ближайшего целого числа, после чего определяем общее расчётное минимальное количество светильников, которое необходимо разместить в помещении:

$$n_{общ\ (min)} = n_{ш\ (min)} \cdot n_{д\ (min)} = 2 \cdot 3 = 6;$$

$$S = a \cdot b = 9 \cdot 4 = 36.$$

По площади помещения S и высоте подвески светильника h_p определяем показатель помещения i:

$$i = \frac{S}{h_{\text{р}} \cdot (\text{a} + \text{b})} = \frac{36}{2{,}5 \cdot (9 + 4)} = 1{,}10 \approx 1{,}25.$$

Находим значения коэффициентов отражения потолка ρ_n, стен ρ_c и полов $\rho_{пол}$ помещения, для которого рассчитывается осветительная установка.

$$\rho_n = 50\%;$$

$$\rho_c = 30\%;$$

$$\rho_{пол} = 10\%.$$

В зависимости от типа светильника и вида лампы определяем коэффициент использования светового потока η_u по показателю помещения i и коэффициентам отражения потолка ρ_n, стен ρ_c и полов $\rho_{пол}$.

$$\eta_u = 0{,}45.$$

Определяем коэффициент запаса k**,** учитывающий снижение уровня освещённости из-за неблагоприятных условий эксплуатации осветительной установки: наличия дыма, копоти, пыли, повышенной концентрации химических веществ и т. д.; из-за старения и выхода из строя ламп.

$$k = 1{,}5.$$

Решаем сколько источников света x будет в светильнике.

$$x = 2.$$

Назначаем коэффициент z, характеризующий неравномерность освещённости (коэффициент отношения средней освещённости к максимальной):

z = 1,2 – для люминесцентных ламп.

Рассчитываем требуемый световой поток одной лампы:

$$\Phi_{расч} = \frac{E \cdot S \cdot k \cdot z}{\eta_и \cdot n_{общ(min)} \cdot x} = \frac{300 \cdot 36 \cdot 1{,}5 \cdot 1{,}2}{0{,}45 \cdot 6 \cdot 2} = 3600 лм.$$

По рассчитанному световому потоку лампы $\Phi_{расч}$ подбираем стандартную лампу со световым потоком $\Phi_{табл}$, значение которого близко к значению $\Phi_{расч}$ (желательно в пределах – 10…+20%).

После выбора стандартных ламп рассчитываем число светильников, необходимых для обеспечения заданной освещённости E. Полученное число $n_{расч}$ округляют до ближайшего целого значения $n_{пр}$, при этом отклонение между принятым количеством светильников $n_{пр}$ и расчётным $n_{расч}$ допускается в пределах от –10 до +20%.

$$\Phi_{1табл} = 3390;$$

$$n_{1пр} = 6;$$

$$n_{1расч} = \frac{E \cdot S \cdot k \cdot z}{\Phi_{табл} \cdot \eta_и \cdot x} = \frac{300 \cdot 36 \cdot 1{,}5 \cdot 1{,}2}{3390 \cdot 0{,}45 \cdot 2} = 6{,}37;$$

$$\Delta_1 = \frac{n_{пр} - n_{расч}}{n_{расч}} \cdot 100\% = -5{,}80\%.$$

Рассчитывают полную мощность проектируемой системы освещения:

$$N_1 = n_{пр} \cdot x \cdot N_{лампы} = 6 \cdot 2 \cdot 65 = 780;$$

$$\Phi_{2табл} = 3865;$$

$$n_{2расч} = \frac{E \cdot S \cdot k \cdot z}{\Phi_{табл} \cdot \eta_и \cdot x} = \frac{300 \cdot 36 \cdot 1{,}5 \cdot 1{,}2}{3865 \cdot 0{,}45 \cdot 2} = 5{,}58;$$

$$n_{2пр} = 6;$$

$$\Delta_2 = \frac{n_{пр} - n_{расч}}{n_{расч}} \cdot 100\% = 7{,}53\%.$$

Рассчитывают полную мощность проектируемой системы освещения:

$$N_2 = n_{пр} \cdot x \cdot N_{лампы} = 6 \cdot 2 \cdot 80 = 960.$$

Микроклимат

Вычислительная техника является источником тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться необходимые параметры микроклимата.

Поскольку работа с ЭВМ при использовании разрабатываемого программного комплекса является основной, а также связана с нервно-эмоциональным напряжением, таким образом необходимо обеспечить оптимальные параметры микроклимата для работ 1а и 1б (табл. 2.1). Для этого рабочее помещение оборудуется системами отопления и кондиционирования.

Таблица 2.1

Оптимальные параметры микроклимата

Период года	Категория работ по уровню энергозатрат	Температура воздуха, (°С)	Относительная влажность	Скорость движения воздуха, (м/с)
Холодный (ниже +10°С)	Iа (до 139) Iб (140-174)	22-24 21-23	40-60	0,1
Теплый (+10°С и выше)	Iа (до 139) Iб (140-174)	23-25 22-24	40-60	0,1

Влияние ЭВМ на организм

Рассмотрим основные вредные факторы, действующие на человека за компьютером:

1) сидячее положение в течение длительного времени;
2) воздействие электромагнитного излучения монитора;
3) утомление глаз, нагрузка на зрение;
4) перегрузка суставов кистей;
5) стресс при потере информации.

Одним из наиболее распространенных источников электромагнитного излучения является компьютер. Наибольшее излучение не со стороны монитора, а со стороны задней стенки.

Изучение возможных последствий воздействия на организм человека находится еще в начале своего пути, однако имеется довольно много убедительных доказательств об опасности для здоровья, особенно электромагнитных полей низкой частоты.

Многим пользователям ПК, обратившимся к грамотному окулисту с жалобами на "ухудшение зрения от работы за компьютером", всего лишь не

хватало в организме витамина А, который есть практически в любой аптеке. Симптомы отсутствия витамина А общеизвестны - это чрезмерная чувствительность к яркому свету и, особенно, ухудшение сумеречного зрения. Эти симптомы принято связывать с компьютером в основном из-за того, что компьютер (при неправильном использовании) повышает потребность глаз в витамине А. Когда пользователь набивает большие объема текста не "вслепую", а глядя то на слабо освещенную клавиатуру, то на сильно освещенный монитор (пусть даже самый безопасный), то для глаз пользователя это является большим испытанием. Зрачки постоянно то сужаются, то расширяются и не успевают настроиться под имеющееся количество света, поэтому и глаза вынуждены работать в "разогнанном режиме".

Характерные для программистов боли в пояснице и в основании шеи запросто могут привести к болезням вен и суставов конечностей. "Синдром программиста" (боли между лопатками) представляет опасность для сердца и легких. Он обычно сопровождается спазмом трапециевидных мышц, которые в попытках спасти позвоночник пережимают артерии, идущие к мозгу (помните давящие боли в затылке). Чуть выше может защемиться нерв, идущий к лицу и среди прочего контролирующий глаза, именно так и появляется временное ухудшение зрения, которое не лечится очками, но проходит после работы с позвоночником на специальном тренажере. Боли в середине спины, на стыке грудного и поясничного отделов, обещают пользователю гастрит, а то и язву желудка, но задолго до этого обеспечивают беспричинным "общим утомлением".

В целях предотвращения негативных последствий рекомендуется раз в 40-45 минут сделать перерыв. Необходимо встать, пройтись, подвигаться, потянуться (наклоны особенно хороши), выполнить гимнастические упражнения.

Для снятия статического и нервно-эмоционального напряжения можно использовать обычные физические упражнения, преимущественно для верхней части туловища (рывки руками, повороты, «рубка дров» и т.д. Для снятия напряжения зрения рекомендуется зрительная гимнастика. Даже при небольшой

ее продолжительности (1 мин), но регулярном проведении, она является эффективным мероприятием профилактики утомления. Эффективность зрительной гимнастики объясняется тем, что при выполнении специальных упражнений (описаны ниже) обеспечивается периодическое переключение зрения с ближнего на дальнее, снимается напряжение с цилиарной мышцы глаза, активизируются восстановительные процессы аккомодационного аппарата глаза, в результате чего функция зрения нормализуется. Кроме того, есть специальное упражнение (с меткой на стекле), предназначенное для тренировки и развития аккомодационной функции глаза.

Упражнения выполняются сидя или стоя, отвернувшись от экрана при ритмичном дыхании, с максимальной амплитудой движения глаз.

Упражнения для глаз:

1) закрыть глаза, сильно напрягая глазные мышцы, на счет 1 - 4, затем раскрыть глаза, расслабив мышцы глаз, посмотреть вдаль на счет 1 - 6. Повторить 4 - 5 раз;
2) посмотреть на переносицу и задержать взор на счет 1 - 4. До усталости глаза не доводить. Затем открыть глаза, посмотреть вдаль на счет 1 - 6. Повторить 4 - 5 раз;
3) не поворачивая головы, посмотреть направо и зафиксировать взгляд на счет 1 - 4, затем посмотреть вдаль прямо на счет 1 - 6. Аналогичным образом проводятся упражнения, но с фиксацией взгляда влево, вверх и вниз. Повторить 3 - 4 раза;
4) перенести взгляд быстро по диагонали: направо вверх - налево вниз, потом прямо вдаль на счет 1 - 6; затем налево вверх направо вниз и посмотреть вдаль на счет 1 - 6. Повторить 4 - 5 раз.

Упражнение для снятия локального утомления:

1) и. п. - о. с. 1 - 2 - встать на носки, руки вверх-наружу, потянуться вверх за руками, 3 - 4 - дугами в стороны руки вниз и расслабленно скрестить перед грудью, голову наклонить вперед. Повторить 6 - 8 раз. Темп быстрый;

2) и. п. - стойка ноги врозь, руки вперед, 1 - поворот туловища направо, мах левой рукой вправо, правой назад за спину. 2 и. п. 3 - 4 - то же в другую сторону. Упражнения выполняются размашисто, динамично. Повторить 6 - 8 раз. Темп быстрый;
3) и. п. 1 - согнуть правую ногу вперед и, обхватив голень руками, притянуть ногу к животу. 2 - приставить ногу, руки вверх-наружу, 3 - 4 - то же другой ногой. Повторить 6 - 8 раз. Темп средний.

Упражнение для улучшения кровообращения мозгового кровообращения:

1) и. п. - о. с. 1 - руки за голову; локти развести, голову наклонить назад. 2 - локти вперед, 3 - 4 - руки расслабленно вниз, голову наклонить вперед. Повторить 4 - 6 раз. Темп медленный;
2) и. п. - стойка ноги врозь, кисти в кулаках. 1 - мах левой рукой назад, правой вверх - назад. 2 - встречными махами переменить положение рук. Махи заканчивать рывками руками назад. Повторить 6 - 8 раз. Темп средний;
3) и. п. - сидя на стуле. 1 - 2 отвести голову назад и плавно наклонить назад. 3 - 4 - голову наклонить вперед, плечи не поднимать. Повторить 4 - 6 раз. Темп медленный.

Наклоны и повороты головы оказывают механическое воздействие на стенки шейных кровеносных сосудов, повышают их эластичность; раздражение вестибулярного аппарата вызывают расширение кровеносных сосудов головного мозга. Дыхательные упражнения, особенно дыхание через нос, изменяют их кровенаполнение. Все это усиливает мозговое кровообращение, повышает его интенсивность и облегчает умственную деятельность.

Пожарная безопасность

В процессе эксплуатации ПК существует опасность возникновения пожара, в связи с чем существует необходимость проведения мер по его предотвращению. Пожарная безопасность представляет собой комплекс действий по предупреждению опасности возникновения пожаров и взрывов, а также, в случае их возникновения, мер по их ликвидации.

Источником пожара могут быть электрические схемы ЭВМ, устройства электропитания, расположенные в непосредственной близости друг от друга коммуникационные кабели и прочее. В современных ЭВМ очень высокая плотность размещения элементов электронных схем. В непосредственной близости друг от друга располагаются соединительные провода, кабели. При протекании по ним электрического тока выделяется значительное количество теплоты. При этом возможно оплавление изоляции. Для отвода избыточной теплоты от ЭВМ служат системы вентиляции и кондиционирования воздуха. При постоянном действии эти системы представляют собой дополнительную пожарную опасность. Также нельзя располагать ПК вблизи источников тепла. Устанавливать системные блоки необходимо так, чтобы задняя и боковые стенки стояли не менее чем на 0,2 м от других предметов. На экраны дисплеев не должны падать прямые солнечные лучи.

Важную роль в обеспечении пожарной безопасности играет обучение и инструктирование персонала при прохождении пожарно-технического минимума (ПТМ). Для этого приказом руководителя необходимо определить порядок и сроки прохождения противопожарного инструктажа и пожарно-технического минимума, а также назначить лиц, ответственных за их проведение. Проводя инструктаж по ПТМ следует: рассказать сотрудникам о мерах предотвращения пожаров и загораний, указать место для курения, проинформировать о действиях при пожаре, свойствах горючих материалов, ознакомить с местом нахождения и инструкцией использования огнетушителя, показать ближайший телефон и прочее.

В офисном помещении обязательно на видном месте должна располагаться «Схема эвакуации людей при пожаре», регламентирующая действия персонала при возникновении очага-возгорания. В помещении нельзя использовать установки для тушения пожара с применением воды, пены, сухих химических порошков. В обязательном порядке должен присутствовать углекислотный огнетушитель (ОУ-2), относящийся к первичному средству огнетушения. Их использование предназначено для тушения загораний электрооборудования, находящегося под напряжением до 1000 В.

Средством обнаружения и оповещения при пожаре являются датчики, которые устанавливаются на потолке и в вытяжных воздуховодах.

Выводы

В рамках организационной части был проанализирован ряд источников опасностей при работе с ПЭВМ.

Рассчитан оптимальный уровень освещения в помещении, где проходит процесс разработки программного комплекса. Для освещения необходимо 6 ламп ЛД-80, расположенных в 2 ряда по три лампы в каждом.

Проанализированы основные показатели микроклимата в теплые и холодные периоды года. В холодный период года температура воздуха должны быть в диапазоне 22-24 градусов по Цельсию для работ типа Iа и 21-23 – для работ типа Iб. В теплый период года соответственно 23-25 и 22-24 градусов по Цельсию. Относительная влажность 40-60%. Скорость движения воздуха не более 0,1м/с.

Отдельно рассмотрены отрицательное влияние ЭВМ на организм человека и способы минимизации этого влияния.

Проанализированы основные аспекты пожарной безопасности и описаны необходимые средства для минимизации риска появления пожара.

Приложение 3

Листинг 3.1

Получение и отправка сообщения

```
protected override async Task ExecuteAsync(CancellationToken
stoppingToken)
        {
            while (!stoppingToken.IsCancellationRequested)
            {
                Thread.Sleep(DelayInMs);
                try
                {
                    var message = await _repository
                        .GetCurrentMessageFromBusiestQueue()
                        .ConfigureAwait(false);
                    if (message is null)
                    {
                        _logger.LogInformation("{DateTime}   Queue   is
empty", DateTime.Now);
                        continue;
                    }

                    _logger.LogInformation("[{DateTime}]    Try    send
message",
                        DateTime.Now);
                    var result = await _senderService
                        .TrySendByEachService(message)
                        .ConfigureAwait(false);

                    if (result.IsSuccess)
                    {
                        message.MarkSuccessful(result.DeliveryType);
                        message.DeleteFromQueue();
                        await UpdateMessage(message)
                            .ConfigureAwait(false);
                        _logger.LogInformation("[{DateTime}]    Message
successfully sent by {Service}",
                            DateTime.Now, result.DeliveryType);
                        continue;
                    }

                    message.MarkFailed(result.ErrorMessage);
                    message.DeleteFromQueue();
                    await UpdateMessage(message)
                        .ConfigureAwait(false);
                    _logger.LogInformation("[{DateTime}]   Sending   via
each delivery service failed", DateTime.Now);
                }
                catch (Exception exception)
                {
                    _logger.LogError("[{DateTime}] {Message}",
```

```
                        DateTime.Now, exception.Message);
                }
            }
            _logger.LogInformation("Worker stopped");
        }

        private async Task UpdateMessage(Message message) =>
            await _repository
                .UpdateMessage(message)
                .ConfigureAwait(false);
```

Листинг 3.2

Интерфейс ISender

```
using System.Threading.Tasks;
using DeliveryRely.Domain.Models.Messages;

namespace DeliveryRely.SendingService.Senders
{
    public interface ISender
    {
        Task<SendingResult> TrySend(Message message);
    }
}
```

Листинг 3.3

Класс SmtpSender, реализующий интерфейс ISender

```
public class SmtpSender : ISender
    {
        private const string filesPath =
"C:/Users/r0bar/Documents/GitHub/DeliveryRely/DeliveryRely.ServiceLayer
/Files";

        public async Task<SendingResult> TrySend(Message message)
        {
            try
            {
                var mailMessages = PrepareMailMessages(message);
                var smtpClient = PrepareSmtpClient(message.User);

                foreach (var mailMessage in mailMessages)
                {
                    if (message.AttachedFiles?.Count > 0
                        && Directory.Exists(Path.Combine(filesPath,
message.MessageId.ToString())))
                    {
                        AttachFiles(mailMessage,
message.AttachedFiles);
                    }
```

```
                    await smtpClient.SendMailAsync(mailMessage);
                    DisposeAttachments(mailMessage);
                }

                return SendingResult.Successful(DeliveryTypes.Smtp);
            }
            catch (Exception exception)
            {
                return SendingResult.Failed(exception.Message);
            }
        }

        protected virtual void AttachFiles(MailMessage mailMessage,
IEnumerable<AttachedFile> files)
        {
            foreach (var file in files)
            {
                var path = Path.Combine(filesPath,
file.MessageId.ToString(), file.Title);
                if (!File.Exists(path))
                {
                    continue;
                }

                mailMessage.Attachments.Add(new Attachment(path,
file.FileType));
            }
        }

        protected virtual void DisposeAttachments(MailMessage message)
        {
            message.Attachments.ForAll(a => a.Dispose());
        }


        private static List<MailMessage> PrepareMailMessages(Message
message)
        {
            var mailMessages = new List<MailMessage>();
            message.DestinationEmails.ForEach(destinationEmail =>
            {
                var from = new MailAddress(message.User.Email,
                    $"{message.User.UserName}
{message.User.UserSecondName}");
                var to = new MailAddress(destinationEmail.Email);
                mailMessages.Add(new MailMessage(from, to)
                {
                    Subject = message.Theme,
                    Body = message.Body
                });
            });
```

```
            return mailMessages;
        }

        private static SmtpClient PrepareSmtpClient(UserBase user) =>
new()
        {
            Host = SmtpOptions.Host,
            Port = SmtpOptions.Port,
            EnableSsl = SmtpOptions.EnableSSL,
            DeliveryMethod = SmtpDeliveryMethod.Network,
            UseDefaultCredentials = false,
            Credentials = new NetworkCredential(SmtpOptions.GmailEmail,
SmtpOptions.GmailPassword)
        };
    }
```

Листинг 3.4

Контроллер MessagesController

```
[Authorize]
    [Route(Constants.ApiPrefix + "messages")]
    public class MessagesController : ControllerBase
    {
        private readonly IRepository _repository;
        private readonly IMapper _mapper;

        public MessagesController(IRepository repository, IMapper
mapper)
        {
            _repository = repository;
            _mapper = mapper;
        }

        // GET api/messages/list
        [Route("list")]
        [HttpGet]
        public async Task<ApiResponse<IEnumerable<MessageProjection>>>
GetList()
        {
            try
            {
                var messageProjections = await
_repository.MessagesIncludeAll
                    .Select(message =>
_mapper.Map<MessageProjection>(message))
                    .ToListAsync();
                return
ApiResponse.Successful<IEnumerable<MessageProjection>>(messageProjectio
ns);
            }
            catch (Exception exception)
```

```
            {
                return
ApiResponse.Failed<IEnumerable<MessageProjection>>(exception.Message);
            }
        }


        // GET api/messages/id
        [Route("{id}")]
        [HttpGet]
        public async Task<ApiResponse<MessageProjection>> Get(long id)
        {
            try
            {
                var message = await _repository.MessagesIncludeAll
                    .AsNoTracking()
                    .Where(m => m.MessageId == id)
                    .FirstOrDefaultAsync();

                var messageProjection = _mapper
                    .Map<MessageProjection>(message ?? throw new
Exception("Message not found"));

                return ApiResponse.Successful(messageProjection);
            }
            catch (Exception exception)
            {
                return
ApiResponse.Failed<MessageProjection>(exception.Message);
            }
        }


        // GET api/messages/queue/id
        [Route("queue/{id}")]
        [HttpGet]
        public async Task<ApiResponse<IEnumerable<MessageProjection>>>
GetMessageForQueue(int id)
        {
            try
            {
                var messages = await
_repository.GetMessagesForQueue(id);
                var messageProjections = messages
                    .Select(m => _mapper.Map<MessageProjection>(m));
                return ApiResponse.Successful(messageProjections);
            }
            catch (Exception exception)
            {
                return
ApiResponse.Failed<IEnumerable<MessageProjection>>(exception.Message);
            }
```

```
        }

        // GET api/messages/user/id
        [Route("user/{id}")]
        [HttpGet]
        public async Task<ApiResponse<IEnumerable<MessageProjection>>>
GetListForUser(long id)
        {
            try
            {
                var messageProjections = (await _repository
                        .GetMessagesForUser(id))
                    .Select(m => _mapper.Map<MessageProjection>(m));
                return ApiResponse.Successful(messageProjections);
            }
            catch (Exception exception)
            {
                return
ApiResponse.Failed<IEnumerable<MessageProjection>>(exception.Message);
            }
        }

        // GET api/messages/next
        [Route("next")]
        [HttpGet]
        public async Task<ApiResponse<Message>> Get()
        {
            try
            {
                var message = await
_repository.GetCurrentMessageFromBusiestQueue();
                if (message is null)
                {
                    throw new Exception("All queues are empty");
                }
                message.AttachedFiles = await
_repository.GetAttachedFilesForMessage(message.MessageId);

                return ApiResponse.Successful(message);
            }
            catch (Exception exception)
            {
                return ApiResponse.Failed<Message>(exception.Message);
            }
        }

        // POST api/messages
        [HttpPost]
        public async Task<ApiResponse> Post([FromBody]Message message)
        {
            try
            {
```

```
                var result = await _repository.InsertMessage(message);
                if (result <= 0)
                {
                    throw new Exception("Message inserting failed");
                }
                return ApiResponse.Successful();
            }
            catch (Exception exception)
            {
                return ApiResponse.Failed(exception.Message);
            }
        }

        // PUT api/messages/id
        [Route("{id}")]
        [HttpPut]
        public async Task<ApiResponse> Put([FromBody] Message message)
        {
            try
            {
                await _repository.UpdateMessage(message);
                return ApiResponse.Successful();
            }
            catch (Exception exception)
            {
                return ApiResponse.Failed(exception.Message);
            }
        }

        // DELETE api/messages/id
        [Route("{id}")]
        [HttpDelete]
        public async Task<ApiResponse> Delete(long id)
        {
            try
            {
                await _repository.DeleteMessage(id);
                return ApiResponse.Successful();
            }
            catch (Exception exception)
            {
                return ApiResponse.Failed(exception.Message);
            }
        }
    }
```

Листинг 3.5

Интерфейс для взаимодействия с БД IRepository

```
public interface IRepository : IDisposable
```

```
    {
        DbSet<Access> Accesses { get; set; }
        IIncludableQueryable<Access, User> AccessesIncludeAll { get; }
        DbSet<AttachedFile> AttachedFiles { get; set; }
        DbSet<Contact> Contacts { get; set; }
        DbSet<DeliveryQueue> DeliveryQueues { get; set; }
        DbSet<DestinationEmail> DestinationEmails { get; set; }
        DbSet<Message> Messages { get; set; }
        IIncludableQueryable<Message, User> MessagesIncludeAll { get; }
        DbSet<Sale> Sales { get; set; }
        DbSet<Tariff> Tariffs { get; set; }
        DbSet<User> Users { get; set; }

        Task<Access> InsertAccess(Access access);
        Task<Access> UpdateAccess(Access access, long id);
        Task<Access> DeleteAccess(long id);
        Task<IEnumerable<Access>> GetActiveAccessesForUser(long id);

        Task<int> InsertAttachedFiles(IEnumerable<AttachedFile>
attachedFiles);
        Task<int> UpdateAttachedFile(long id, AttachedFile file);
        Task<int> DeleteAttachedFile(long id);
        Task<List<AttachedFile>> GetAttachedFilesForMessage(long
messageId);

        Task<Contact> InsertContact(Contact contact);
        Task<Contact> UpdateContact(Contact contact, long id);
        Task<Contact> DeleteContact(long id);
        Task<IEnumerable<Contact>> GetContactsForUser(long id);

        Task<DeliveryQueue> InsertDeliveryQueue(DeliveryQueue
deliveryQueue);
        Task<DeliveryQueue> UpdateDeliveryQueue(DeliveryQueue queue,
int id);
        Task<DeliveryQueue> DeleteDeliveryQueue(int id);
```

```
        Task<int> InsertDestinationEmail(DestinationEmail
destinationEmail);
        Task<int> UpdateDestinationEmail(DestinationEmail
destinationEmail);
        Task<int> DeleteDestinationEmail(long id);
        Task<Message> GetCurrentMessageFromBusiestQueue();
        Task<int> InsertMessage(Message message);
        Task<Message> UpdateMessage(Message message);
        Task<Message> DeleteMessage(long id);
        Task<IEnumerable<Message>> GetMessagesForQueue(int queueId);
        Task<IEnumerable<Message>> GetMessagesForUser(long userId);

        Task<Sale> InsertSale(Message message);
        Task<Sale> UpdateSale(Message message, int id);
        Task<Sale> DeleteSale(int id);

        Task<Tariff> InsertTariff(Tariff tariff);
        Task<Tariff> UpdateTariff(Tariff tariff, int id);
        Task<Tariff> DeleteTariff(int id);

        Task<User> InsertUser(User user);
        Task<User> UpdateUser(User user, long id);
        Task<User> DeleteUser(long id);
    }
```

Листинг 3.6

Модульный тест механизма хэширования

```
    [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);
        }
    }
```

Листинг 3.7

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);
        }
    }
}
```