**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО  
«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**Кафедра «Информатика и программное обеспечение»**

**ПОДСИСТЕМА УПРАВЛЕНИЯ НАДЕЖНОСТЬЮ**

**ОТПРАВКИ ЭЛЕКТРОННЫХ ПИСЕМ**

**КУРСОВАЯ РАБОТА**

Документы текстовые

Всего \_\_\_\_ листов в папке

**Руководитель**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. Лагерев Д.Г.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

**Студент**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

**БРЯНСК 2021**

**АННОТАЦИЯ**

Данная курсовая работа посвящена организации создания встраиваемого сервиса, служащего для управления и контроля надежности отправки электронных писем.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc67515150)

[1. Анализ требований 8](#_Toc67515151)

[1.1. Обзор предметной области 8](#_Toc67515152)

[1.1.1. Общие сведения 8](#_Toc67515153)

[1.1.2. Основные понятия 8](#_Toc67515154)

[1.1.3. Процесс отправки сообщения 9](#_Toc67515155)

[1.1.4. Основные проблемы 11](#_Toc67515156)

[1.2. Обзор программ-аналогов 12](#_Toc67515157)

[1.2.1. Amazon SES 12](#_Toc67515158)

[1.2.2. Sendgrid 13](#_Toc67515159)

[1.2.3. Tin-cat email queue 14](#_Toc67515160)

[2. Техническое задание 16](#_Toc67515161)

[2.1. Основание для разработки 16](#_Toc67515162)

[2.2. Назначение и область применения 16](#_Toc67515163)

[2.3. Требование к программному комплексу 16](#_Toc67515164)

[2.3.1. Требования к функциональным характеристикам 16](#_Toc67515165)

[2.3.2. Требования к надежности 19](#_Toc67515166)

[2.4. Условия эксплуатации 20](#_Toc67515167)

[2.4.1. Климатические условия эксплуатации 20](#_Toc67515168)

[2.4.2. Требования к квалификации и численности персонала 21](#_Toc67515169)

[2.4.3. Требования к составу и параметрам технических средств 21](#_Toc67515170)

[2.4.4. Требования к информационной и программной совместимости 21](#_Toc67515171)

[2.5. Программная документация 22](#_Toc67515172)

[2.5.1. Предварительный состав программной документации 22](#_Toc67515173)

[2.6. Стадии и этапы разработки 22](#_Toc67515174)

[2.6.1. Стадии разработки 22](#_Toc67515175)

[2.6.2. Содержание работ по этапам 22](#_Toc67515176)

[2.7. Порядок контроля и приемки 23](#_Toc67515177)

[2.7.1. Виды испытаний 23](#_Toc67515178)

[2.7.2. Общие требования к приемке работы 24](#_Toc67515179)

[3. Проектирование 25](#_Toc67515180)

[3.1. ER-диаграмма 25](#_Toc67515181)

[3.2. DF-диаграмма 29](#_Toc67515182)

[3.3. IDEF0-Диаграмма 31](#_Toc67515183)

[3.4. BPMN 34](#_Toc67515184)

[3.4.1. Пользователь и клиентская часть 34](#_Toc67515185)

[3.4.2. Серверная часть и сервисы доставки 34](#_Toc67515186)

[3.4.3. Диаграмма модели бизнес-процесса 35](#_Toc67515187)

[4. Проектирование архитектуры ПС 36](#_Toc67515188)

[4.1. Клиентская часть 38](#_Toc67515189)

[4.2. Серверная часть 38](#_Toc67515190)

[4.3. Хранилище данных 39](#_Toc67515191)

[5. технико-экономическая часть 41](#_Toc67515192)

[5.1. Организационная структура проекта 41](#_Toc67515193)

[5.2. Календарный план проекта 41](#_Toc67515194)

[5.3. Расчёт затрат на разработку продукта 46](#_Toc67515195)

[5.3.1. Расчёт заработной платы исполнителей работ по созданию программного продукта 46](#_Toc67515196)

[1.1.1. Расчёт отчислений на социальные нужды (страховые взносы) 47](#_Toc67515197)

[5.3.2. Арендные платежи за производственные (офисные) помещения 48](#_Toc67515198)

[5.3.3. Амортизация используемых основных средств и нематериальных активов 48](#_Toc67515199)

[5.3.4. Расходы на модернизацию и приобретение основных средств 49](#_Toc67515200)

[5.3.5. Расходы на приобретение необходимого ПО 50](#_Toc67515201)

[5.3.6. Расходы на интернет и связь 50](#_Toc67515202)

[5.3.7. Расходы на канцелярские товары и расходные материалы 50](#_Toc67515203)

[5.3.8. Прочие расходы 50](#_Toc67515204)

[5.3.9. Расчёт себестоимости программного продукта 50](#_Toc67515205)

[6. Экспериментальная часть 53](#_Toc67515206)

[6.1. Тестирование корректности работы WEB API методом черного ящика 53](#_Toc67515207)

[6.2. Модульное тестирование 56](#_Toc67515208)

[6.3. Кроссбраузерное тестирование интерфейсов пользовательской части 58](#_Toc67515209)

[7. Список литературы 61](#_Toc67515210)

# 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: «подсистема управления надежностью отправки электронных писем».

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

Целью курсовой 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 курсовой работе являются методы и средства улучшения надежности и контроля рассылки электронных сообщений.

# Анализ требований

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

## Обзор предметной области

### 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 сервисом заключается в выборе подходящих сервисов доставки, передаче сообщения этим сервисам и последующей проверке того, доставлено и прочитано сообщение или нет.

### Основные понятия

Ниже представлены определения основным понятиям, необходимым в дальнейшей работе.

*Сообщение –* определенная информация, которую необходимо передать.

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

*Отправленное письмо –* письмо, переданное сервису доставки.

*Получатель* – лицо, которому адресовано отправляемое сообщение.

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

*Отправитель* – лицо, запрашивающее отправку сообщения.

*Статус доставки –* информация о результате доставки сообщения получателю.

*WEB-интерфейс* – веб-страница или несколько веб-страниц, позволяющие взаимодействовать с нужным сайтом, сервисом.

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

*Электронная цифровая подпись (ЭЦП)* – идентификатор документа, полученный в результате криптографического преобразования информации с использованием закрытого ключа подписи и позволяющий проверить отсутствие искажения информации в электронном документе, а также принадлежность подписи владельцу сертификата ключа подписи.

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

### Процесс отправки сообщения

Изучим процесс отправки сообщения (рис. 1.1).



Рис. 1.1. Процесс отправки сообщения

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 жмет 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.

### Основные проблемы

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.

## 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.

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

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

### Amazon SES

Amazon Simple Email Service [1] – это экономичный, гибкий и масштабируемый сервис электронной почты, с помощью которого разработчики могут отправлять электронные письма из любого приложения. Вы можете быстро настроить Amazon SES и выбрать несколько вариантов использования электронной почты, включая отправку транзакций, маркетинговых писем или выполнение массовой рассылки. Amazon SES включает различные возможности развертывания IP-адресов и аутентификации по электронной почте, которые позволяют повысить эффективность доставки и защитить репутацию отправителя, а также предоставляет аналитику, с помощью которой можно проанализировать эффективность каждого отправленного письма. Используйте Amazon SES для безопасной отправки электронной почты по всему миру.

Данный сервис 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.

Данный сервис не лишен минусов:

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

### Sendgrid

Sendgrid [2] запущен в 2017 году. Заменяет пользователям индивидуальные почтовые серверы, анализирует репутацию почтовых рассылок, не требует дополнительных затрат для масштабирования инфраструктуры.

Сервис Sendgrid предлагает каждому пользователю стать партнёром:

* Партнёрам агентства предоставляются инструменты для управления почтовыми программами клиентов с одной платформы;
* Партнёры по рынку, после согласования с администрацией, добавляют собственные инструменты и продают их более чем 60-тысячной «горячей» аудитории;
* OEM-партнёры могут рассылать письма без подписи «via sendgrid.net»;
* Партнёры-посредники занимаются перепродажей услуг Sendgrid своим пользователям, экономя собственные ресурсы.

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

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

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

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

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

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

### Tin-cat email queue

Tin-cat email queue [3] – система очереди для отправки сообщений. При попытке отправить сообщение, 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.

Рассмотренные программные комплексы, разработанные сторонними компаниями, не реализуют все указанные функции или реализуют их не в полной мере. Данные ПО позволяют лишь частично контролировать отправку сообщений. Также важным моментом является высокая цена использования некоторых сервисов (таких как Sendgrid), ненадежность API. Важно: если сообщение не удается отправить, данные сервисы не предпринимают попыток отправить его другим способом, что отрицательно сказывается на стабильности рассылки.

Из вышеперечисленного можно сделать вывод о необходимости разработки программного продукта, который должен решать все поставленные задачи.

# Техническое задание

Техническое задание включает в себя требования, необходимые для реализации программного продукта.

Название программного комплекса «Подсистема управления рассылкой электронных писем и их учёта».

## Основание для разработки

Основанием для разработки подсистемы управления надежностью отправки электронных писем является задание на курсовую работу, выданное доцентом Трубаковым А.О. на основании приказа по Брянскому государственному техническому университету № 428-3 от 27 мая 2020 г.

## 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.

## Требование к программному комплексу

### Требования к функциональным характеристикам

Реализуемый программный комплекс должен включать в себя нижеупомянутые функциональные особенности:

1. Авторизация пользователей в web-интерфейсе для администрирования:

Роли:

1. Оператор – может управлять рассылкой сообщений, просматривать соответствующую историю сообщений;
2. Администратор – имеет все возможности оператора, но также может просматривать список операторов, создавать новых операторов и удалять существующих.
3. После входа в Web-сервис пользователь может:
4. отправлять сообщения на электронные почтовые ящики;
5. выбирать сервисы доставки из списка доступных на данный момент;
6. прикреплять файлы к сообщению;
7. организовывать отправку сообщений по расписанию;
8. просматривать подробную информацию о доставке сообщений.
9. Разрабатываемый комплекс должен обладать следующими функциями:
10. возможность повторной отправки недоставленного сообщения, но уже посредством другого доступного сервиса доставки при наличии Интернет-соединения;
11. возможность отправлять сообщения посредством минимум **трех** различных сервисов для отправки сообщений;
12. наличие API, с помощью которого подсистему можно будет встраивать в различные сервисы, использующие рассылку сообщений;
13. возможность прикреплять файлы к сообщению;
14. возможность хранить историю сообщений, т.е. хранить определенное число ранее отправленных сообщений и давать доступ к просмотру информации об их доставке;
15. 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.
16. 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.
17. В целях улучшения надежности разрабатываемый комплекс должен включать в себя кластерную систему серверов.
18. 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. прикрепленные файлы;
4. список адресов, на которые нужно отправить сообщения;
5. список сервисов доставки, посредством которых 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.

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

### 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.
   * + 1. 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.

* + - 1. 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.

## Условия эксплуатации

### Климатические условия эксплуатации

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

### Требования к квалификации и численности персонала

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

### Требования к составу и параметрам технических средств

Подсистема 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.

### Требования к информационной и программной совместимости

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

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

Дополнительные требования к защите программного обеспечения и информации не предъявляются.

## Программная документация

### Предварительный состав программной документации

Состав программной документации:

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

## Стадии и этапы разработки

### Стадии разработки

Разработка подсистемы состоит из следующих этапов:

* разработка технического задания;
* проектирование программного комплекса;
* разработка программного комплекса;
* тестирование разработанного программного комплекса;
* написание программной документации;
* внедрение разработанного сервиса.

### 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.

## Порядок контроля и приемки

### 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

### Общие требования к приемке работы

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

# Проектирование

## ER-диаграмма

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

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

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

* **Messages**
  + id (long) – суррогатный первичный ключ;
  + userId (long) – id отправителя;
  + destinationDate (dateTime) – дата и время успешной доставки;
  + theme (nvarchar(max)) – тема сообщения;
  + body (text) – тело сообщения;
  + size (int) – размер сообщения в единицах (байт);
  + isScheduled (int) – логическое поле – запланирована ли отправка сообщения;
  + scheduleDate (dateTime) – запланированная дата и время, когда необходимо отправить сообщение;
  + isSent (int) – логическое поле – успешно ли доставлено сообщение;
  + queueId (int) – id очереди, в которой содержится данное сообщение.
* **MessagesAttachedFiles**
  + messageId (long) – суррогатный первичный ключ сообщения;
  + attachedFileId (long) – суррогатный первичный ключ файла.
* **AttachedFiles**
  + attachedFileId (long) – суррогатный первичный ключ;
  + line (nvarchar(max)) – ссылка на файл, хранящийся на сервере;
  + size (int) – размер файла, на который указывает ссылка в поле link;
  + typeId – тип файла, на который указывает ссылка в поле link.
* **Users**
  + id (long) – суррогатный первичный ключ;
  + userName (nvarchar(max)) – имя пользователя;
  + email (nvarchar(max)) – почтовый адрес пользователя;
  + passwordHash (nvarchar(max)) – хэш, полученный из пароля пользователя;
  + isActive (int) – логическое поле – статус доступа пользователя;
  + queueId (int) – id очереди, в которую помещаются сообщения данного пользователя.
* **UsersAccesses**
  + userId (long) – суррогатный первичный ключ пользователя;
  + accessId (long) – суррогатный первичный ключ доступа
* **Accesses**
  + id (long) – суррогатный первичный ключ;
  + userId (long) – id пользователя, которому принадлежит данный доступ
  + startDate (dateTime) – дата и время начала подписки пользователя;
  + endDate (dateTime) – дата и время конца подписки пользователя;
  + tariffId (int) – id тарифа пользователя.
* **Contacts**
  + id (long) – суррогатный первичный ключ;
  + contactName (nvarchar(MAX)) – имя контакта;
  + contactEmail (ncarchar(MAX)) – электронный адрес контакта;
  + userId (long) – id пользователя, которому принадлежит эта запись.
* **DeliveryServices**
  + id (int) – суррогатный первичный ключ;
  + serviceName (nvarchar(max)) – название сервиса доставки;
  + standartPriority (int) – значение приоритета данного сервиса доставки при переборе списка сервисов во время отправки по умолчанию;
  + connectionString (nvarchar(max)) – строка, необходимая для взаимодействия с данным сервисом.
* **DeliveryQueues**
  + id (int) – суррогатный первичный ключ;
  + sendingIntervalSec (int) – интервал отправки сообщений.
* **Tariffs**
  + id (int) – суррогатный первичный ключ;
  + tariffName (nvarchar(255)) – название тарифа;
  + cost (int) – стоимость тарифа.
* **TariffsSales**
  + tariffId (int) – суррогатный первичный ключ тарифа;
  + saleId (int) – суррогатный первичный /ключ скидки.
* **Sales**
  + id (int) – суррогатный первичный ключ;
  + name (nvarchar(255)) – название скидки на тариф;
  + value (int) – размер скидки на тариф.

ER диаграмма представлена на рис. 3.1.

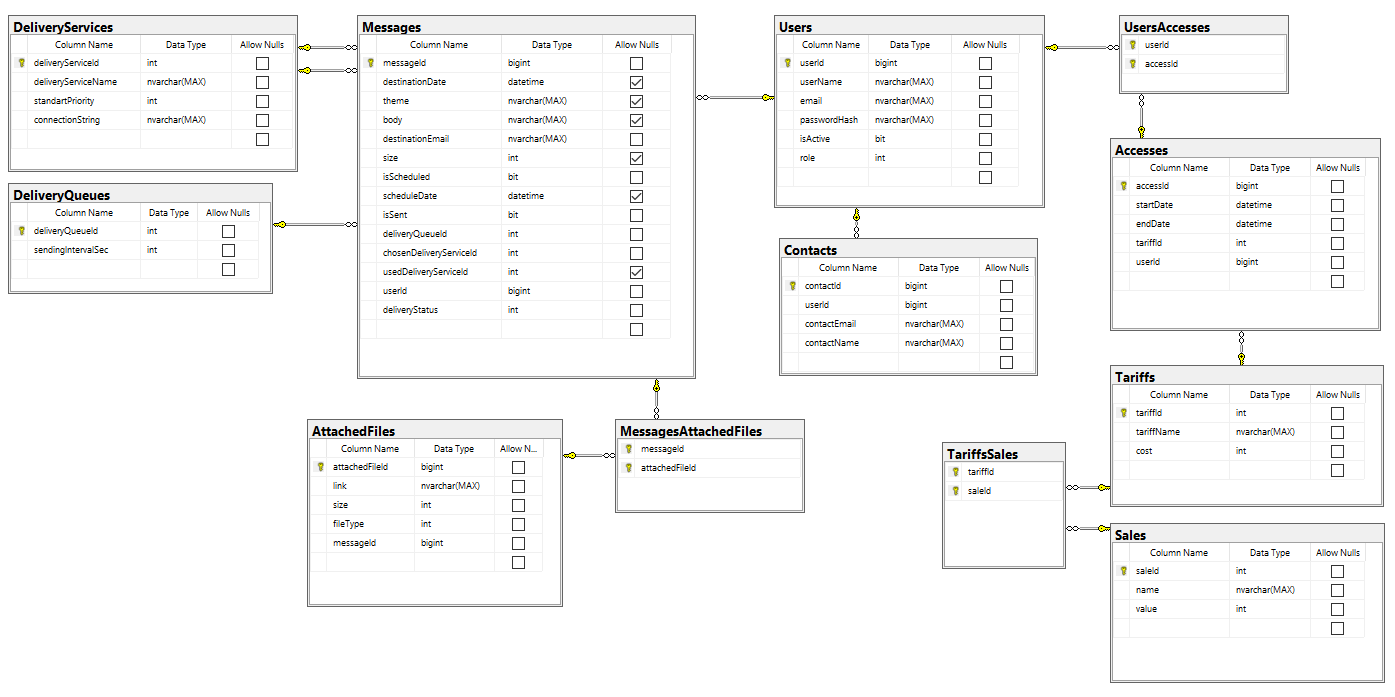


Рис. 3.1. ER диаграмма

## DF-диаграмма

Поставлена задача: разработать модель потоков данных **для программного комплекса управления рассылкой электронных сообщений**

На контекстной диаграмме (рис. 3.2) 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. 3.3) выполнена декомпозиция базового процесса на три подпроцесса. На диаграмме второго уровня проведена детализация процесса «Инициировать отправку письма» (рис. 3.4). На диаграммах третьего уровня проведена детализация процесса «Отправить письмо» (рис. 3.5).



Рис. 3.2. Контекстная диаграмма



Рис. 3.3. Диаграмма первого уровня модели потоков данных



Рис. 3.4. Диаграмма второго уровня модели потоков данных. Процесс «Инициировать отправку письма»



Рис. 3.5. Диаграмма третьего уровня потоков данных. Процесс «Отправить письмо»

## IDEF0-Диаграмма

Поставлена задача: *отправить сообщение.* Необходимо определить функциональные требования к программной системе.

На контекстной диаграмме (рис. 3.6) определен функциональный блок «Отправить сообщение через систему», механизм, предписание, входные и выходные данные.

На диаграмме первого уровня (рис. 3.7) выполнена декомпозиция базового блока «Отправить сообщение через систему». Этот блок содержит четыре подфункции: «Сформировать письмо», «Инициировать отправку сообщения», «Занести информацию о доставке в журнал». Проведена детализация функций «Сформировать письмо» (рис. 3.8) и «Инициировать отправку сообщения» (рис. 3.9)



Рис. 3.6. Контекстная диаграмма функциональной модели «Поиск объектов»



Рис. 3.7. Диаграмма первого уровня функциональной модели



Рис. 3.8. Диаграмма второго уровня для блока «Сформировать письмо»



Рис. 3.9. Диаграмма второго уровня для блока «Инициировать отправку сообщения»

## BPMN

Поставлена задача: разработать модель бизнес-процесса **для программного комплекса управления рассылкой RELY.**

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 хранилище данных и отправляет письмо в очередь.
4. Сторонние сервисы доставки.

### Пользователь и клиентская часть

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 и сервисы доставки

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

Если сервис доставки возвращает ответ о том, что письмо успешно доставлено, формируется информация о доставке. Эта информация заносится в хранилище данных и возвращается в ответ на запрос от клиентской части.

Если сервис доставки возвращает информацию о том, что письмо не доставлено или попало в спам, то предпринимается попытка заново отправить письмо, но посредством других доступных сервисов доставки. В крайнем случае, когда письмо не удалось доставить ни одним из доступных сервисов, соответствующая информация заносится в хранилище данных и возвращается клиентской части.

### Диаграмма модели бизнес-процесса

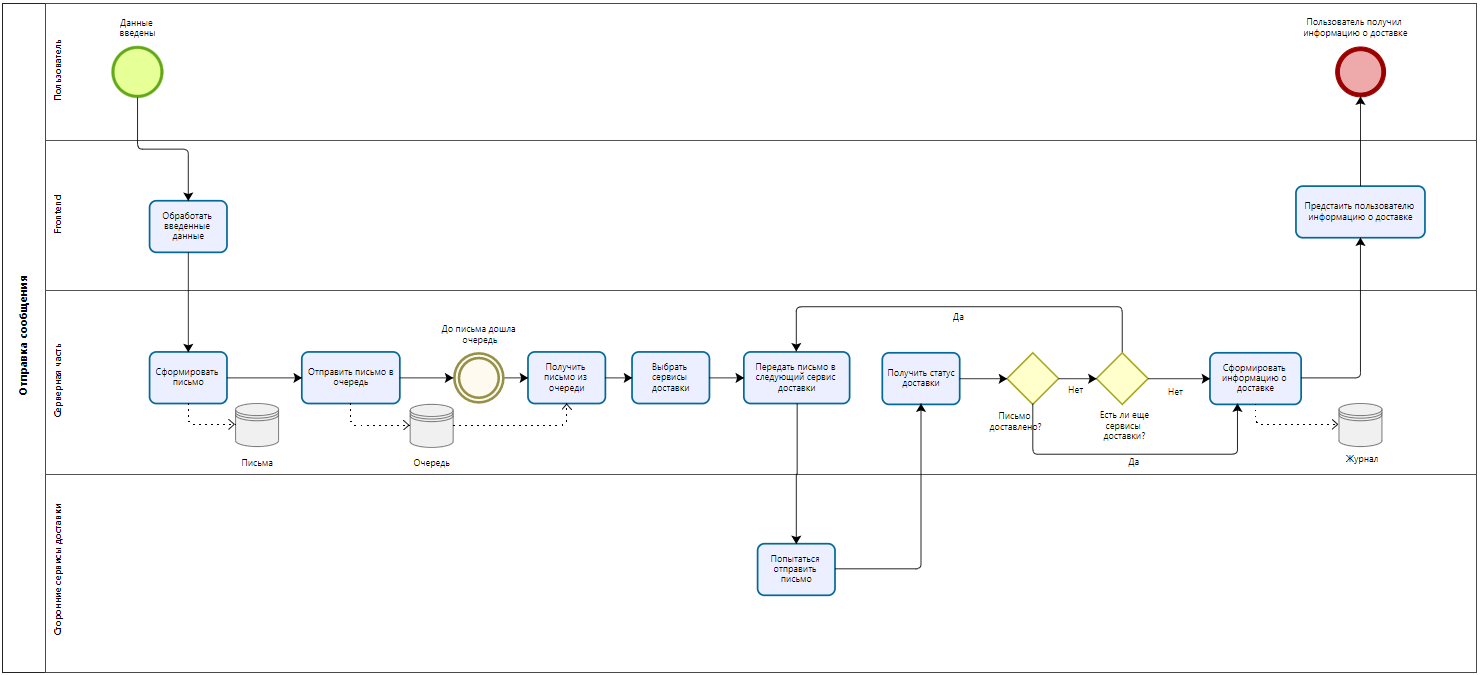


Рис. 3.10. Диаграмма модели бизнес-процесса

# Проектирование архитектуры ПС

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

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

Схема архитектуры ПС представлена на рис. 4.1.

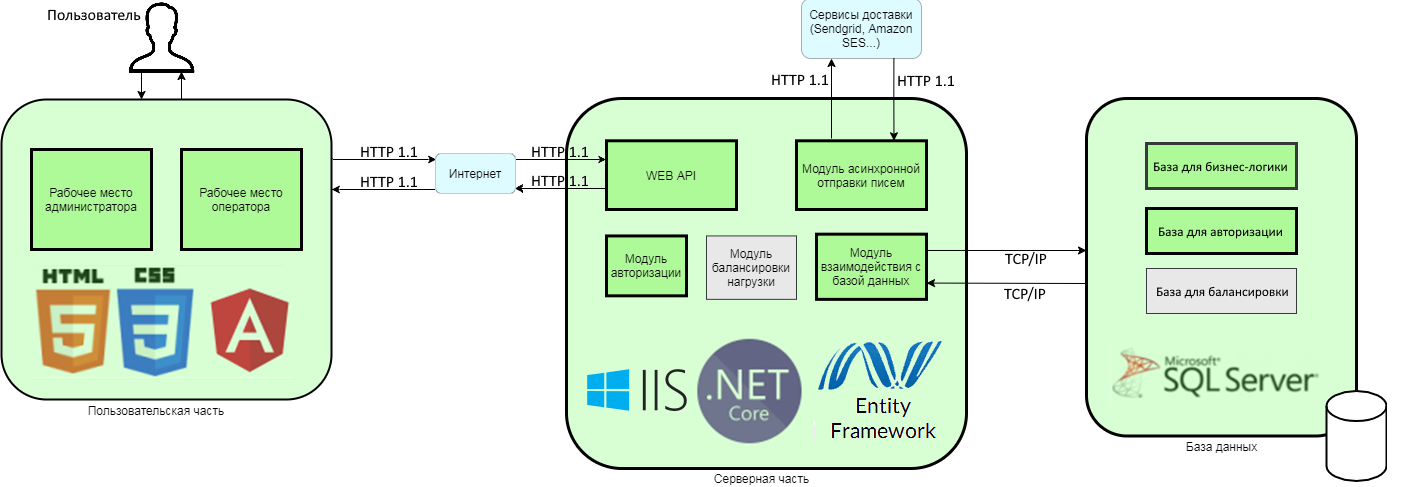


Рис. 4.1. Схема архитектуры ПС

## Клиентская часть

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

1. React;
2. Angular;
3. Vue.

Среди выбранных фреймворков предпочтение было отдано Angular благодаря его декларативности, наличию развитого сообщества, модульности и MVC из коробки. В качестве скриптового языка из-за ряда преимуществ над JavaScript был выбран TypeScript.

Для вёрстки будет применяться классический набор инструментов: HTML + CSS.

## Серверная часть

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

Серверная часть условно поделена на ряд модулей:

1. WEB API;
2. Модуль авторизации;
3. Модуль балансировки нагрузки;
4. Модуль асинхронной отправки писем;
5. 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.

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

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

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

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

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

## Хранилище данных

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

1. MySQL;
2. MS SQL;
3. PostgreSQL.

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

В базе данных будут храниться все основные сущности системы:

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

# технико-экономическая часть

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

Организационная структура проекта (OBS) приведена на рис. 5.1.



Рис. 5.1. 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 следующие работы:

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. 5.1.

**Таблица 5.1**

***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 |
| 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. 5.2. Графическое представление диаграммы Ганта показано на рис. 5.3.



Рис. 5.2. Табличное представление Диаграммы Ганта



Рис. 5.3. Графическое представление Диаграммы Ганта

Исходя из длительности работ и коэффициента загрузки членов проектной команды, определим их трудозатраты при реализации проекта (табл. 5.2).

|  |  |  |
| --- | --- | --- |
| **№** | **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 |

**Таблица 5.2**

Трудозатраты членов проектной команды

## 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 – *ЗПосн*;
* дополнительная заработная плата *ЗПдоп*;
* отчисления на социальные нужды (страховые взносы) – *Нзп*;
* арендные платежи за производственные (офисные) помещения – *Апм*;
* амортизация используемых основных средств и нематериальных активов – *А*;
* расходы на модернизацию и приобретение основных средств – *Рмод*;
* расходы на приобретение необходимого ПО – *РПО*;
* расходы на интернет, связь – *Ртел*;
* расходы на канцелярские товары и расходные материалы – *Рр.м.*;
* прочие расходы – *Пр.р.*.

### Расчёт заработной платы исполнителей работ по созданию программного продукта

Основная ЗП определяется по формуле:

|  |  |
| --- | --- |
|  | (5.1) |

где *M* – месячная зарплата (*руб.*), *T* – общие трудозатраты (*чел.-ч*), *Чр* – число рабочих дней в месяц, *tр.д. –* продолжительность рабочего дня в часах, *П* – процент премии. В данной работе *Чр =* 21 *день*, *tр.д.=*8*ч*, *П*=0.

Значение месячной заработной платы (М), суммарные трудозатраты членов, а также рассчитанная по формуле 1.1 основная заработная плата проектной команды приведены в табл. 1.3.

**Таблица 5.3**

Основная заработная плата членов проектной команды

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Исполнитель** | **Месячная заработная плата (М), руб.** | **Трудозатраты, человеко-часов** | **, 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.

### Расчёт отчислений на социальные нужды (страховые взносы)

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 5.4**

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, *Нзп=* 95 702 (*руб.*).

### Арендные платежи за производственные (офисные) помещения

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

Стоимость аренды составляет 500 руб/м2 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 м2 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 *Апм* = .

### Амортизация используемых основных средств и нематериальных активов

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:

*(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:

*(руб.).*

Суммарные амортизационные отчисления составят: А= 78 986 руб.

### Расходы на модернизацию и приобретение основных средств

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

### Расходы на приобретение необходимого ПО

При реализации проекта не планируется приобретение ПО.

### Расходы на интернет и связь

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.

### Расходы на канцелярские товары и расходные материалы

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

### Прочие расходы

Прочие расходы составляют 30% от суммы следующих элементов структуры затрат: *ЗПосн*, *ЗПдоп*, *Нзп*, *Апм*, *А*, *Рмод*, *РПО*, *Ртел* и *Рр.м.*.

|  |  |
| --- | --- |
|  |  |

Таким образом, *Пр.р.=* 299 114 (*руб.*).

### Расчёт себестоимости программного продукта

В себестоимость программного продукта входят следующие элементы: *ЗПосн*, *ЗПдоп*, *Нзп*, *Апм*, *А*, *Рмод*, *РПО*, *Ртел*, *Рр.м.* и *Пр.р.*.

Сложив все элементы, можно определить себестоимость программного продукта и услуг по его внедрению: *Сп.п.*= 1 398 699 (*руб.*).

Структура себестоимости программного продукта отражена в табл. 5.5 и представлена на рис. 5.4.

**Таблица 5.5**

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** |

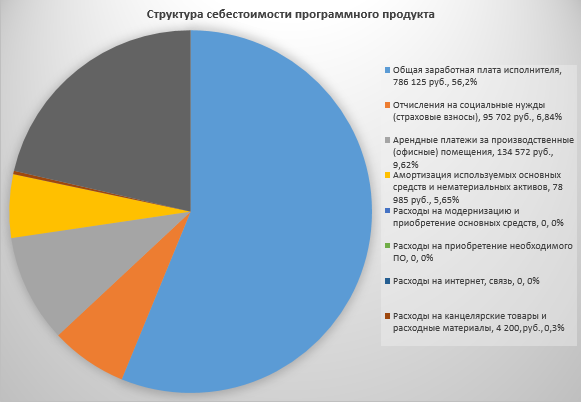


Рис. 5.4. Структура себестоимости программного продукта

# Экспериментальная часть

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

1. тестирование корректности работы WEB API методом черного ящика;
2. модульное тестирование;
3. кроссбраузерное тестирование интерфейсов пользовательской части.

## Тестирование корректности работы WEB API методом черного ящика

Тестирование методом черного ящика – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы [?]. 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. messages;
2. users;
3. contacts;
4. files;
5. delivery-services;
6. delivery-queues;
7. accesses;
8. sales.

Пример тестов представлены в таблице 6.1.

**Таблица 6.1.**

Тесты для тестирования методом черного ящика

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

## Модульное тестирование

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

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 листинге 6.1.

Листинг 6.1. Модульный тест механизма хэширования

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

В проекте DeliveryRely.Domain.Tests содержатся тесты для методов получения (методы Get…), добавления (методы Insert…), изменения (методы Update…) и удаления (методы Delete…) класса EFRepository. Пример теста представлен в листинге 6.2.

Листинг 6.2. Модульный тест контактов

|  |
| --- |
| using Domain.Models.Contacts;  using Domain.Models.Users;  using Microsoft.VisualStudio.TestTools.UnitTesting;  namespace DeliveryRely.Domain.Tests  {  [TestClass]  public class ContactTest  {  [TestMethod]  public void FieldsTest()  {  var contact = new Contact  {  ContactId = 1,  ContactName = "contact",  ContactEmail = "contact@yandex.ru",  UserId = 4,  User = new User  {  Email = "user@yandex.ru"  }  };  Assert.AreEqual("contact", contact.ContactName);  Assert.AreEqual("contact@yandex.ru", contact.ContactEmail);  Assert.AreEqual(4, contact.UserId);  Assert.AreEqual("user@yandex.ru", contact.User.Email);  }  }  } |

## 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[?].

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. 6.1) и Opera (рис. 6.2).



Рис. 6.1. Список сообщений в Google Chrome

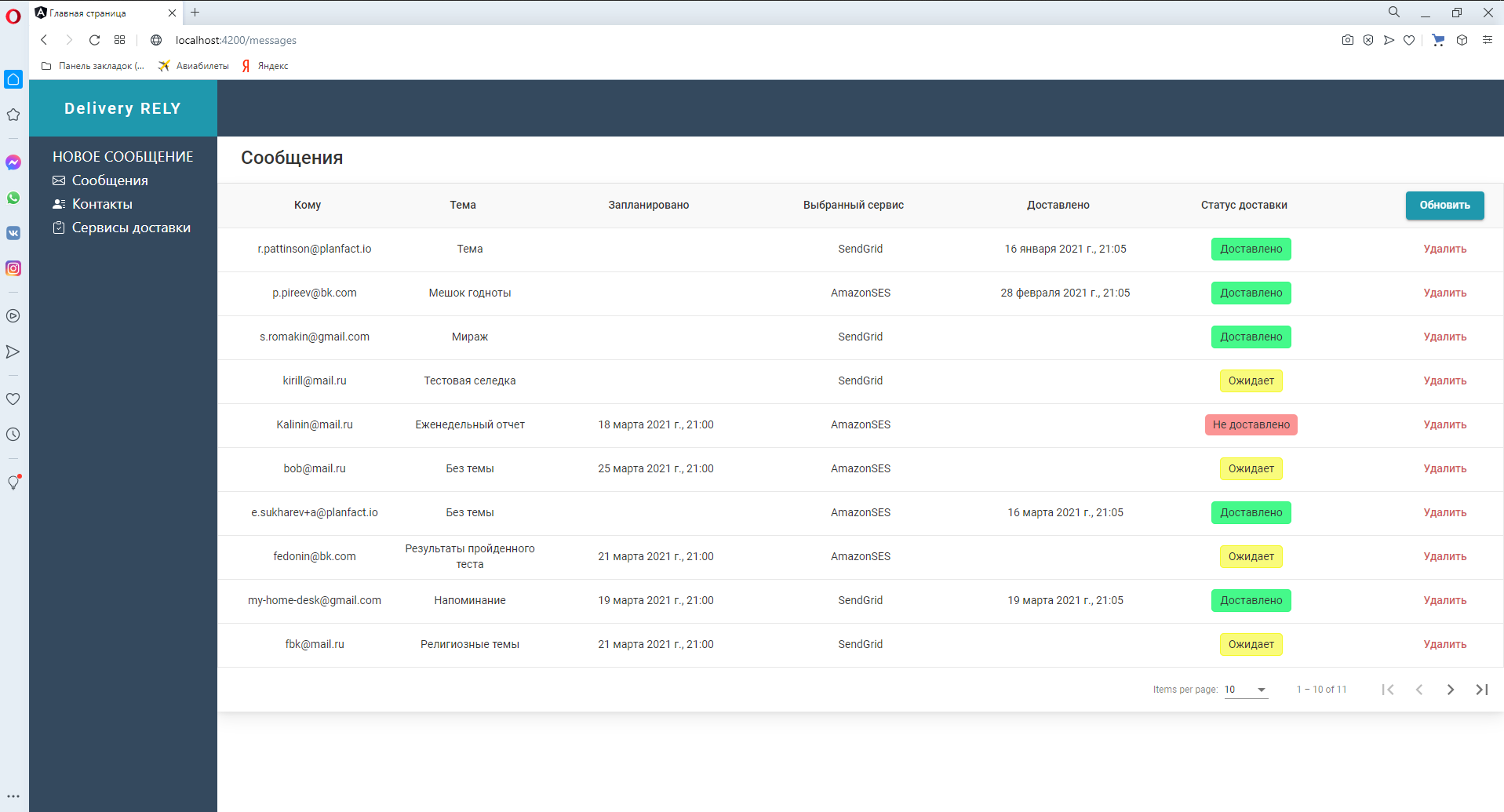


Рис.6.2. Страница сообщений в Opera

# Список литературы

1. Официальный сайт Amazon SES – режим доступа: https://aws.amazon.com/ru/ses/
2. Официальный сайт Sendgrid – режим доступа: https://sendgrid.com
3. Проект Email queue пользователя tin-cat на Github – режим доступа: <https://github.com/tin-cat/emailqueue>