Здравствуйте уважаемый председатель, уважаемые члены государственной экзаменационной комиссии.

Вашему вниманию предоставляется выпускная квалификационная работа на тему «Проектирование программного комплекса управления надежной отправкой электронных писем с реализацией основного механизма рассылки и пользовательского интерфейса».

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

Основанием для разработки является задание на дипломную работу, выданное к.т.н., доцентом Трубаковым А.О., а также заявка на разработку программного комплекса управления надежной отправкой электронных писем, полученная от компании ООО «Максимум Веб».

**Цель** работы: проектирование указанного программного комплекса и дальнейшая интеграция с системами, требующими высокую надежность отправки.

**Объектом** исследования являются процессы отправки и доставки электронных сообщений.

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

Полученное задание было разбито на ряд этапов, представленных на слайде: анализ предметной области, сравнительный анализ аналогов, подготовка ТЗ, проектирование, разработка, тестирование и внедрение.

В рамках анализа аналогов были рассмотрены следующие сервисы: Amazon SES, Sendpulse, tin-cat’s emailqueue. Описание представлено в записке на стр. 13-16. Кратко о имеющихся недостатках.

При использовании Amazon SES скорость отправки может быть снижена из-за особого алгоритма определения пути, а также имеет место относительная ненадежность – при неудачной доставке будет пересчитан путь доставки.

Sendpulse позволяет отправлять сообщения через SMTP и через API. Например, в моем сервисе присутствует отправка через API. При этом API часто меняется, из-за чего может страдать обратная совместимость.

Tin-cat/emailqueue – пример очереди для асинхронной отправки сообщений. Главный недостаток – это ненадежность доставки.

Для разработки были выбраны следующие технологии: для серверной части - .net core 5, для клиентской части – Angular с языком TS. Модель данных – MS SQL, а мобильная разработка планируется на языках Java и Swift.

На данном слайде представлена архитектура программного комплекса. Как видно из схемы выбрана трехуровневая модель. Посмотреть схему поближе можно на стр. 37. Пользователь взаимодействует с серверной часть через Веб-API (запросы отправляются из клиентской части или через мобильное приложение). Функционал клиентской части дублируются в мобильном приложении.

Серверная часть взаимодействует с базой данных посредством Entity Framework Core. В серверной части реализованы все модули кроме модуля балансировки нагрузки и модуля планирования доставки.

На схеме показаны сторонние сервисы доставки. В рамках дипломной работы спроектирована работа с сервисами Amazon SES, Sendpulse, Sendgrid и Unisender и реализована отправка сообщений через SMTP и Sendpulse.

База данных состоит из трех частей – модуль бизнес-логики, модуль авторизации и модуль балансировки.

Рассмотрим модель данных подробнее. Модуль бизнес-логики включает в себя основные бизнес-сущности (такие как сообщения, файлы и т.д.), модуль авторизации - информацию о пользователях, ролях и доступах, а модуль балансировки – информацию о доступных серверах и их ролях.

Приведенная на слайде диаграмма вариантов использования описывает весь доступный операторам и администраторам функционал. Как видно из диаграммы, оператору доступны просмотр истории сообщений и отправка сообщения, а администратору – всё, что доступно оператору, плюс управление операторами – добавление и удаление операторов.

Рассмотрим бизнес-процесс создания и отправки сообщение подробнее. В этом нам поможет модель бизнес-процесса. Пользователь создает сообщение, клиентская часть отправляет соответствующий запрос на сервер. На сервере формируется письмо, и это письмо помещается в очередь, после чего пользователю возвращается соответствующее сообщение.

Далее фоновая служба, запущенная на сервере, получает это письмо из очереди и пытается отправить его с помощью доступных сервисов доставки. После чего меняется статус сообщения.

На данном слайде представлена диаграмма классов фоновой службы. WorkerMain – основной класс, в котором находится цикл отправки. На каждой итерации происходит получение письма из очереди и передача его объекту класса SenderService. Тот в свою очередь пытается отправить сообщение посредством каждого доступного сервиса доставки (через интерфейс ISender) и возвращает ответ в виде объекта класса SendingResult.

Расскажу о балансировке нагрузки. Подробнее можно посмотреть на стр.53. Есть распределительный сервер – он принимает запросы от пользователя, передает их серверам, обрабатывающим запросы, получает от них ответ и перенаправляет их обратно пользователю. А вот как распределять запросы по серверам – это определяет алгоритм балансировки. Программный комплекс предоставляет несколько на выбор: Round Robin, Weighted Round Robin, Least Connections.

Перейдем к экономической части. На данном слайде представлена организационная структура проекта. Подробнее можно посмотреть на стр. 65. На схеме представлены следующие роли: системный аналитик, бизнес-аналитик, дизайнер, разработчик, тестировщик, а также специалист по внедрению.

Перед вами табличное представление диаграммы Ганта. Подробнее можно прочитать на стр. 66.

На данном слайде представлена структура себестоимости программного продукта. Наибольшую долю трат (56,2%) занимает общая заработная плата исполнителей, а за ней (23,39%) – прочие расходы.

Подведем вывод:

1. Программный комплекс спроектирован и частично разработан.
2. Произведена интеграция с веб-приложением ПланФакт.
3. Статья с материалами проектирования программного комплекса представлена на Всероссийской конференции «Автоматизация и моделирование в проектировании и управлении» 19 мая 2021 года и вышла в печать.

С вашего позволения перейду к демонстрации работы приложения.

(После демонстрации) На этом у меня всё. Благодарю вас за внимание.