

ГУАП

КАФЕДРА №51

КУРСОВАЯ РАБОТА (ПРОЕКТ)  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

доцент, канд.техн.наук

Линский Е.М.

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ)

Почтовый сервис ZenMail

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ II

РАБОТУ ВЫПОЛНИЛИ

Щипило А.М.  
Заболотный А.В.  
Костин С.О.

СТУДЕНТЫ ГР. № 5511

подпись, дата

инициалы, фамилия

Санкт-Петербург 2017

# Содержание

<b>1 Функциональная спецификация</b>	<b>2</b>
1.1 Цель . . . . .	2
1.2 Основные функции . . . . .	2
1.3 Технологии . . . . .	3
1.4 Команда . . . . .	3
<b>2 Особенности реализации</b>	<b>4</b>
2.1 Backend . . . . .	4
2.1.1 Тестирование . . . . .	7
2.2 Mail Server . . . . .	7
2.3 Frontend . . . . .	10
<b>3 Описание архитектуры проекта</b>	<b>11</b>
<b>4 Руководство пользователя</b>	<b>12</b>
4.1 Авторизация . . . . .	12
4.2 Входящие сообщения . . . . .	15
4.3 Отправка сообщения . . . . .	17
<b>5 Заключение</b>	<b>20</b>

# 1 Функциональная спецификация

## 1.1 Цель

В качестве основной цели данной курсовой работы была поставлена разработка почтового сервиса с выделенной базой данных, почтовым сервером и функционально полным веб-приложением, позволяющим принимать и отправлять письма зарегистрированным пользователям.

Следовательно, в данном случае необходимо было заняться многими направлениями разработки, и разбить большой объем работ над веб-приложением. Так сформировалась команда из трех человек: Андрей Щипило, Артем Заболотный и Сергей Костин.

Тогда весь объем работы разбивается на три главных компонента:

- **Интерактивная часть веб-приложения** — интерактивная оболочка, созданная при помощи фреймворка Angular 2+. (далее frontend)
- **Программно-аппаратная часть веб-приложения** — программа на фреймворке Spring с архитектурой REST, управляющая данными с frontend. (далее backend)
- **Почтовый сервер с базой данных** — сервер, отвечающий за отправку, прием и хранение писем. (далее mailserver)

**Frontend:** это интерфейс взаимодействия между пользователем и программно-аппаратной частью веб-приложения, использующий веб-технологии для обработки информации полученной от пользователя на клиенте или отправляя информацию на *Backend*, предоставляя пользователю полный контроль над своим почтовым ящиком, включая регистрацию, аутентификацию, а также просмотр и отправку сообщений.

**Backend:** это программа, исполняющаяся на предполагаемом сервере, которая обрабатывает информацию и отвечает на запросы от *Frontend*, выполняя необходимые действия (прим. добавление нового пользователя в базу данных). Для улучшения опыта использования веб-приложения сервер должен своевременно и быстро обрабатывать запросы от нескольких пользователей одновременно. Именно с этого приложения происходит запросы почтовому серверу на отправку и чтение писем, по протоколам IMAP, POP3 и SMTP.

**Mail Server:** это специально настроенный сервер, размещенный на сервере, который может быть использован большую часть времени (прим. облачное хранилище). На машине в облаке установлены и конфигурированы программы для отправки и принятия писем, проверки писем на спам, вирусы, а также для аутентификации и шифрования соединений. Причем именно на этом сервер хранится и база данных, с которой активно взаимодействует *Frontend* часть веб-приложения.

## 1.2 Основные функции

Основные функции программы:

- Возможность подключения к почтовому серверу не только с реализованного в курсовой работе веб-приложения
- Регистрация новых пользователей через форму.
- Вход в почтовый ящик по логину и паролю.
- Просмотр пришедших писем.
- Поиск по письмам.
- Создание нового письма с базовым редактированием.

Component	Technology
Frontend	Angular 4+ and Covalent
Backend (REST)	SpringBoot (Java)
Security	Token
Persistence	JPA (Using Spring Data)
Client Build Tools	angular-cli, Webpack, npm
Server Build Tools	Maven (Java)

Таблица 1: Стэк технологий

### 1.3 Технологии

Кроме основного стэка технологий будут использоваться дополнительные сторонние разработки и программы для обеспечения быстроты, удобства разработки и использования, а также общей безопасности системы:

- Ubuntu Server 16.04 LTS
- Postfix, MTA
- Dovecot
- MySQL
- Rspamd
- Amazon EC2

### 1.4 Команда

Так как существовали установленные сроки сдачи курсовой работы, необходимо было распределить курс работ равномерно между всеми участниками проекта, чтобы максимизировать участие каждого из участников в проекте и минимизировать время затраченное в ожидании результатов одного из участников. В конечном итоге было предложено такое распределение по обязанностям:

- Щипило Андрей: занимается общей архитектурой веб-приложения SpringFramework, правильной инициализацией процессов приложения через SpringBoot и Maven, графическим и пользовательским представлением веб-приложения, используя возможности Angular 4, менеджментом проекта, а также настройкой и поддержанием почтового сервера в облачном хранилище.
- Костин Сергей: занимается обработкой данных почтового сервера и веб-приложения, используя MariaDB и Spring Data, представлением данных в программе, а отвечает за безопасную аутентификацию пользователей через токены и фильтры безопасности системы, используя Spring Security и защищённые протоколы связи.
- Заболотный Артем: занимается связью между программной и графической частями приложения, используя нативный для Angular 4 язык программирования TypeScript, из Spring Security, является менеджером глобального тестирования веб-приложения, а также за защищенную связь через почтовые протоколы IMAPS и SMTPS, позволяющую безопасно связать программную часть веб-приложения и почтовый сервер.

## 2 Особенности реализации

Для более подробного описания, а также документации лучше всего обратиться на официальную страницу приложения на сервисе github.

<https://github.com/ProjectZenMail/zenmail>

The screenshot displays the GitHub profile for the ZenMail project. At the top, it shows the repository name 'ZenMail' and its URL 'https://github.com/ProjectZenMail/zenmail'. Below this are standard GitHub navigation links for issues, pull requests, and releases. The main content area includes:

- Repository Statistics:** 83 commits, 5 branches, 0 releases, 3 contributors.
- Recent Activity:** A list of recent commits from 'walker2' showing contributions to 'java\_report', 'src', '.gitignore', 'README.md', 'package-lock.json', and 'pom.xml'.
- README.md:** A section containing the project's description: 'Application for receiving and sending emails built with Java and Angular 4+'.
- Technology Stack:** A table mapping project components to their technologies:

Component	Technology
Frontend	Angular 4+ and Covalent
Backend (REST)	SpringBoot (Java)
Security	Token Based (Spring Security and JWT )
In Memory DB	H2
Persistence	JPA (Using Spring Data)

Рис. 1: Страница проекта на сервисе GitHub

### 2.1 Backend

Для разработки такого, достаточно масштабного, проекта необходимо было найти некий фреймворк, который облегчил поставленную задачу. В качестве такого фреймворка был выбран самый популярный на данный момент – Spring Framework, который позволяет упростить настройку и конфигурацию сервера и обеспечить все необходимые компоненты для разработки программной части полноценного веб-приложения.

Так как SpringFramework – это популярный комплексный фреймфорк для разработки веб-приложений, то у него большое количество настроек. Для упрощения запуска и отладки нашего веб-приложения используется модуль **SpringBoot**: с помощью аннотации `@SpringBootApplication` для нашего запускающего класса SpringBoot освобождает нас от слишком глубокой настройки приложения и позволяет собирать, размещать и запускать приложение с помощью одной команды – `springboot:run`. Данная команда за нас компилирует код,

запускает движок серверов Tomcat, создает соответствия для REST API и загружает необходимые серверы.

Но у нас все также остается возможность модифицировать необходимые конфигурационные переменные через изменение файла application.properties, так например мы можем задать адрес нашей базы данных, указать порт для запуска сервера или отключить дополнительные модули для тестирования.

Кроме Spring'a были использованы библиотеки, облегчающие некоторые аспекты. Например, для уменьшения кода при написании приватных полей классов модели был использована библиотека **Lombok**. Так же для отправки и принятия сообщений была использована библиотека **javax.Mail**. Для сборки и загрузки модулей, а также для тестирования и генерации jar файла был использован Maven.

Для обеспечения безопасности аутентификации был использован модуль SpringFramework – **Spring Security**. А именно были созданы фильтры, запрещающие доступ к скрытым данным через запросы, создана аутентификация на токенах и фильтры Spring Security для проверки подлинности токенов пользователей. Диаграмма работы аутентификации через токены приведена ниже:

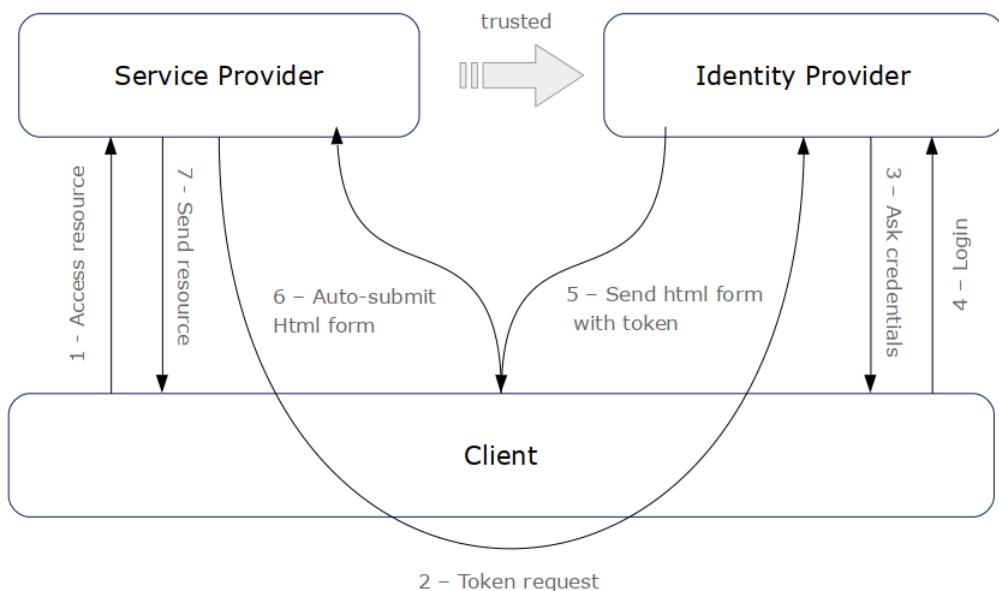


Рис. 2: Аутентификация через токены

Для простоты и элегантности кода было решено использовать **JPA** из SpringFramework, чтобы создать Repository слой, предназначенный для CRUD функций взаимодействия с базой данных.

Из-за большого количества модульных компонентов необходимо было упростить сборку и тестирование отдельных компонентов. Для данной задачи был использован **Apache Maven** – фреймворк для автоматизации сборки проектов. С его помощью можно было не заботится о скачивании и установки новых модулей, так как Maven берет их со своих серверов и подключает к приложению. Для запуска всего модуля Backend необходимо было ввести лишь одну команду – mvn clean install, а затем запустить сгенерированный jar файл с помощью java -jar ./target/zenmail-0.0.1.jar.

Некоторые из важных методов Backend'a:

1. MainController – класс, который возвращает index.html, когда требуется mapping '/'. Нужен, чтобы выдавать экран загрузке при запуске веб-приложения.
2. UserService – класс, который организует доступ в свои репозитории (данные из базы данных) и предоставляет возможность работы с ними.

- `addNewUser(User user)` – добавляет нового пользователя в репозитории UserRepository
  - `getLoggedInUser()` – возвращает всю информацию, в виде класса User, о залогиненном пользователе.
3. UserController – класс, который использует класс UserService, для возвращения необходимых значений, связывая вызовы GET и POST по пути /user.
- `getUserInformation()` – вызывает `getLoggedInUser()` класса UserService и возвращает результат в виде UserResponse, который хранит в себе информацию о пользователе в виде класса User.
  - `getLoggedInUser()` – возвращает всю информацию, в виде класса User, о залогиненном пользователе.
4. MessagesService – класс, который хранит необходимые сведения и методы для отправки писем
- `getMessages(User user)` – возвращает все доступные сообщения конкретного пользователя в качестве вектора IMAPMessage.
  - `sendMessage(String addressTo, String subject, String text, User user)` – отправляет сообщение по указанному адресу, с указанной темой, текстом от определенного пользователя.
5. MessagesController – класс, который использует класс MessagesService, для возвращения необходимых значений, связывай вызовы GET и POST по пути /messages.
- `getUserInformation()` – вызывает `getMessages()` класса MessagesService и возвращает результат в виде MessagesResponse, который хранит в себе вектор всех сообщений.
  - `sendMessage()` – возвращает всю информацию, в виде класса User, о залогиненном пользователе.

Для более наглядной и функциональной документации API были использованы **Swagger** и **ReDoc**.

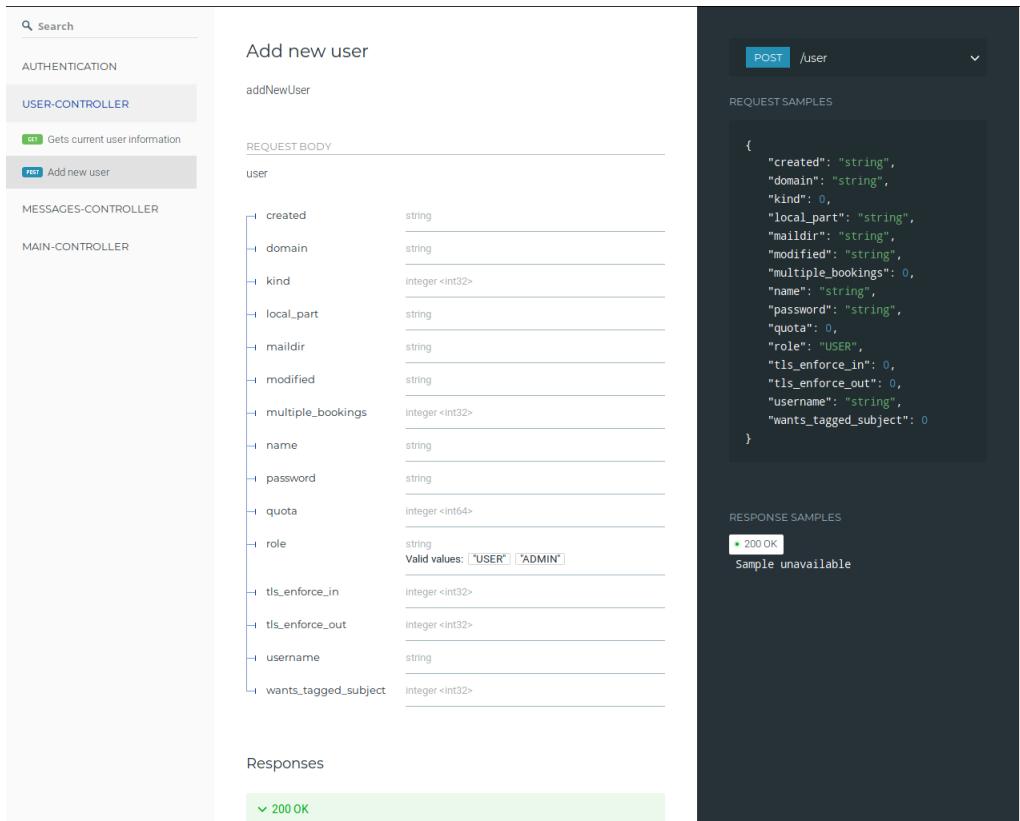


Рис. 3: Документация API

### 2.1.1 Тестирование

Для тестирования был использованы библиотеки **Spring Test** и **JUnit 4**. Были написаны тестирующие программы для всех основных задач api, а именно регистрацию, авторизацию, принятие и отправку сообщений. Для создания имитационных объектов (mock objects) был использован фреймворк **Mockito**.

Покрытие кода выведено с помощью **Jacoco**.

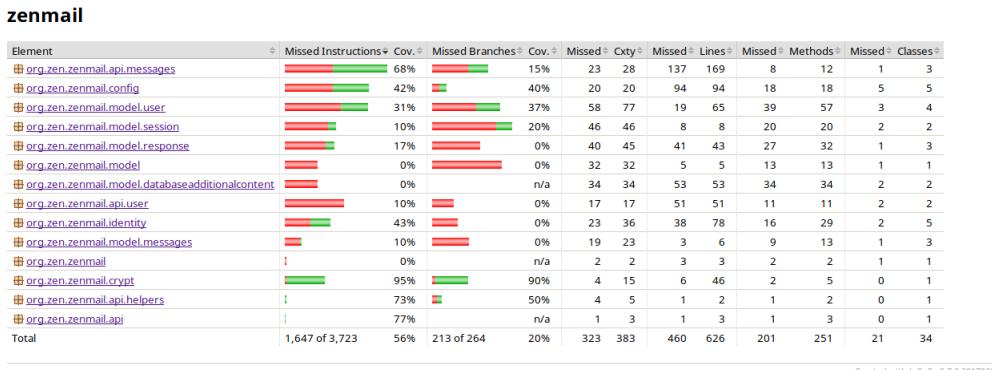


Рис. 4: Покрытие тестами кода

### 2.2 Mail Server

Для корректной работы приложения, а именно для базовой авторизации, необходимо было предоставить место для хранения данных пользователей. И, так как почтовый сервер необходимо было разместить на компьютере с открытыми необходимыми портами для различных почтовых протоколов. Было решено использовать реляционную базу данных на основе системы управления базами данных **MariaDB**.

Схема базы данных mail приведена ниже:

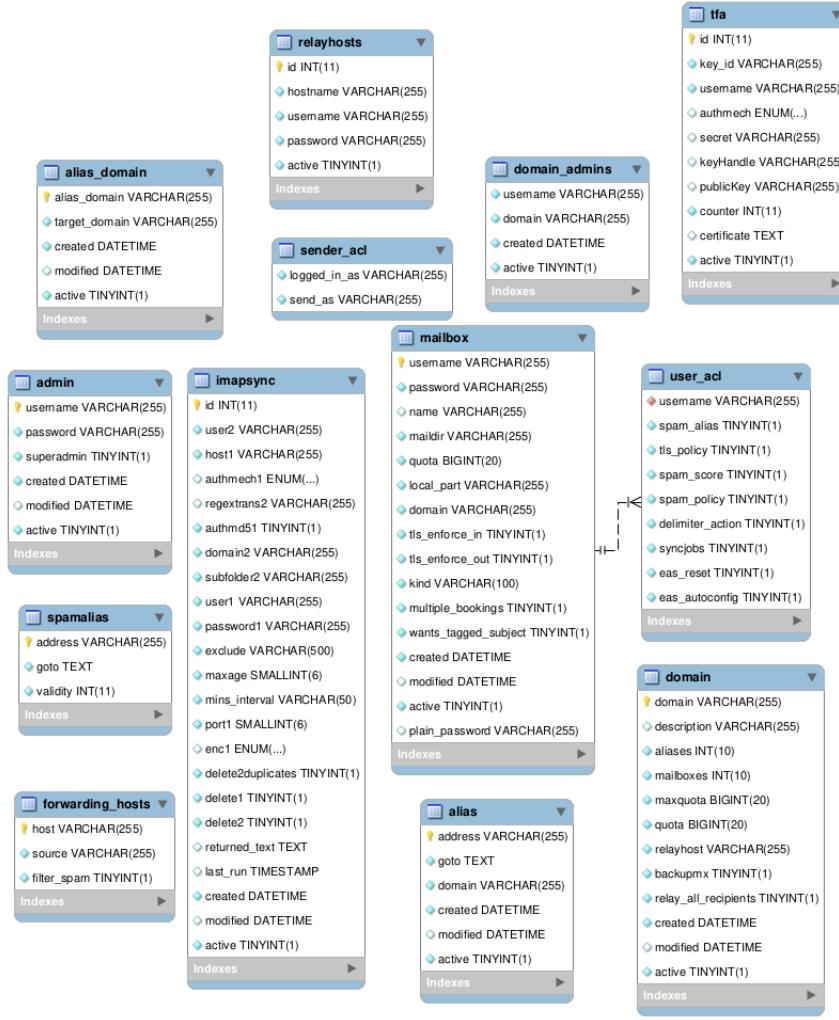


Рис. 5: Схема базы данных

Основными таблицами, с которыми осуществляется взаимодействие программы являются *mailbox* и *user*, предоставляющие всю необходимую информацию о пользователе.

Остальные же таблицы необходимы для работы программ, отвечающих за работу почтового сервера.

Одной из главных таких программ является **Dovecot** – это свободный IMAP- и POP3-сервер с упором на безопасность. Из-за возможности гибкой настройки программы с помощью конфигурационных файлов можно кардинально поменять принцип работы программы. В проекте используется IMAP аутентификация пользователя, так как она требует меньше дополнительных затрат ресурсов на создание копии ящика пользователя на клиенте, и просто каждый раз забирает почту по TCP протоколу, что, конечно, накладывает некие ограничения на скорость работы.

Именно с этой программой связывается Backend часть приложения и запрашивает определенную информацию (напр. входящие письма), авторизовавшись по логину и паролю. Dovecot настроен так, чтобы хранить письма пользователей в виртуальных папках системы, в которой он установлен. Это, а также каким способом авторизоваться программе, по какой таблице определять письма и прочее описывается в конфигурационных файлах Dovecot.

Второй программой является агент передачи почты (MTA) **Postfix**. Как можно понять из названия, именно через эту программу отправляются письма через протокол SMTP. Для данной программы тоже существует

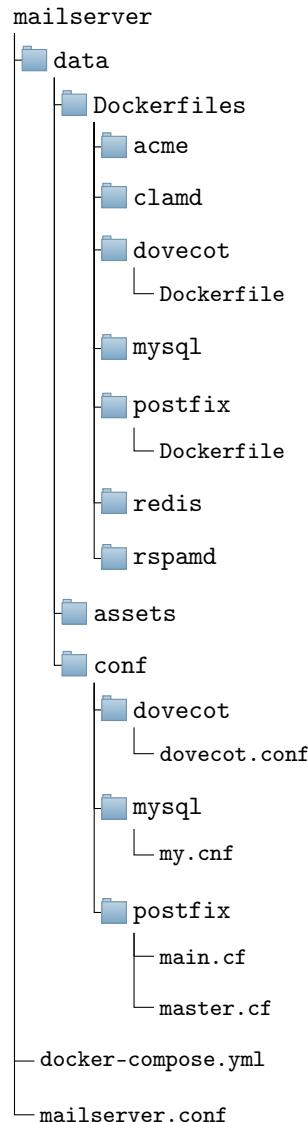
множество настроек, которые задаются в конфигурационных файлах.

Также установлены и другие программы, необходимые для фильтрования спама и осуществления передачи через TLS протоколы (напр. `rspamd`).

Так как все эти программы требуют портов, которые обычно закрыты на роутерах обычных компьютеров, и сама суть сервера подразумевает постоянную работу всех этих приложений, то было принято решение использовать облачные вычисления от Amazon Web Services – *EC<sup>2</sup>*. Вся настройка сервера происходила через сетевой протокол SSH, который позволяет подключаться к операционной системе удаленных систем с помощью приватного и публичного ключа.

Так как требовалось большое количество испытаний прежде чем был бы найден оптимальный вариант настройки конфигурационных файлов приложений, а некоторые из них достаточно сложно найти в установленной системе и при изменении одной строчки конфигурационного файла могла лечь вся программа, то было принято решение использовать **docker**. Docker позволяет автоматизировать развертывание и управление приложениями в виртуализированной среде. Так как нам нужно было использовать много контейнеров для различных приложений, то был использован **Docker Compose**, который позволяет запускать мультиконтейнерные приложения через конфигурационный файл и команду `docker-compose up`.

Ниже представлена структура папок docker



## 2.3 Frontend

Для того, чтобы конечный пользователь мог удобно взаимодействовать с описанным выше почтовым сервером, необходимо создать оболочку веб-приложения.

Для этой цели был использован фреймворк веб-приложений **Angular2+**, который позволяет упростить создание интерактивных сайтов, при этом добавляя множество функций. Одна из главных – это возможность написания на строго типизированном языке программирования **TypeScript**,

Ниже приведена схема архитектуры фреймворка Angular2+

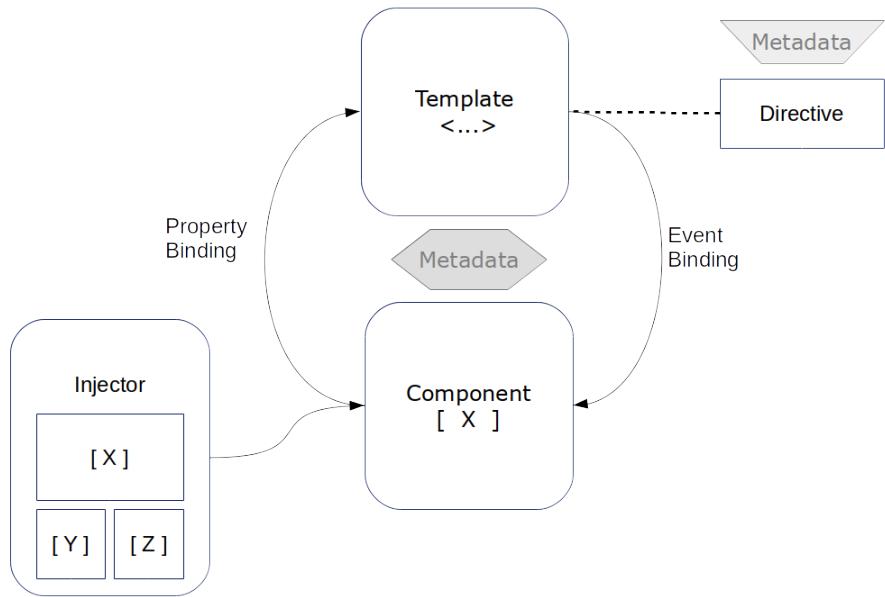


Рис. 6: Архитектура фреймворка Angular2+

Для упрощения графическо-дизайнерской стороны был использован модуль angular-material, который предоставляет набор различных элементов для придания веб-приложению вида material-design.

Для сборки веб-приложения был использован модуль **angular-cli**. Из-за большого количества остальных модулей, от которых зависят многие вышеописанные модули, был использован менеджер пакетов - **npm**, который скачивает и подсоединяет модули к проекту.

### 3 Описание архитектуры проекта

Проект, как говорилось раньше, можно разделить на три главные составляющие: Frontend, Backend и MailServer. В общем виде Frontend посыпает запрос Backend'у, и в большинстве случаев Backend посыпает запрос базе, получает ответ, обрабатывает и отсыпает на Frontend.

Общее взаимодействие между основными компонентами проекта можно наблюдать на схеме ниже.

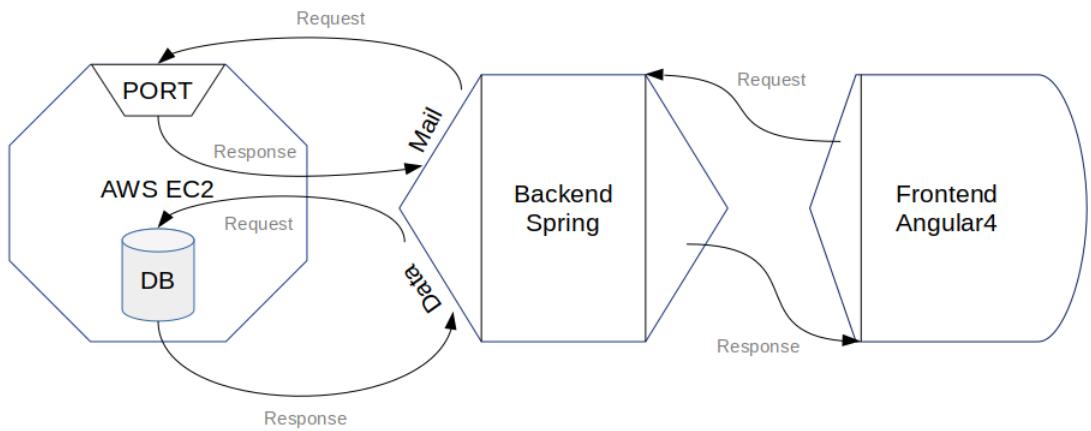


Рис. 7: Взаимодействие составляющих проекта

Веб-приложение организовано с помощью паттерна проектирования Model-View-Control (MVC), где моделями являются различные внутренние представления (напр. User, у которого есть имя, директории, логин и т.д.), control – это сервисы и контроллеры для управления моделями (напр. у MessagesController есть метод sendMessage, который отсылает сообщение по IMAP, используя сервис MessagesService), view – это отображение наших данных (напр. сообщений) с использованием фреймворка Angular 4.

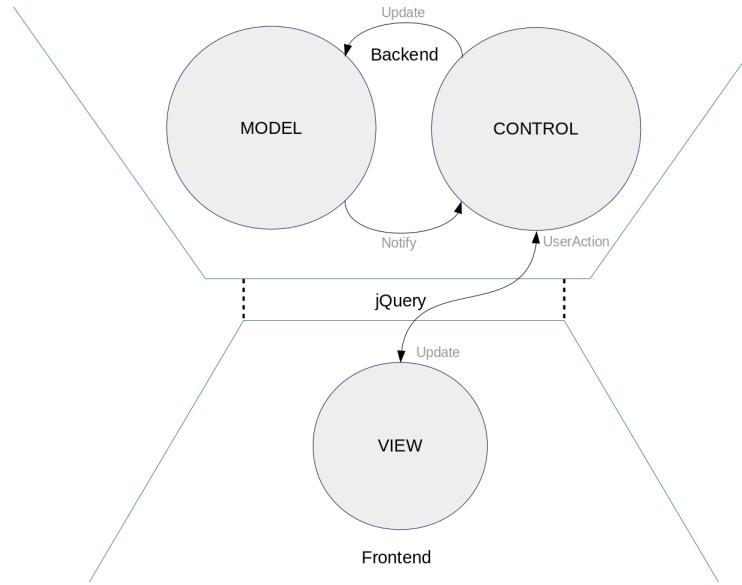


Рис. 8: Взаимодействие составляющих проекта

## 4 Руководство пользователя

### 4.1 Авторизация

После перехода на адрес веб-приложения пользователь перенаправят на страницу авторизации.

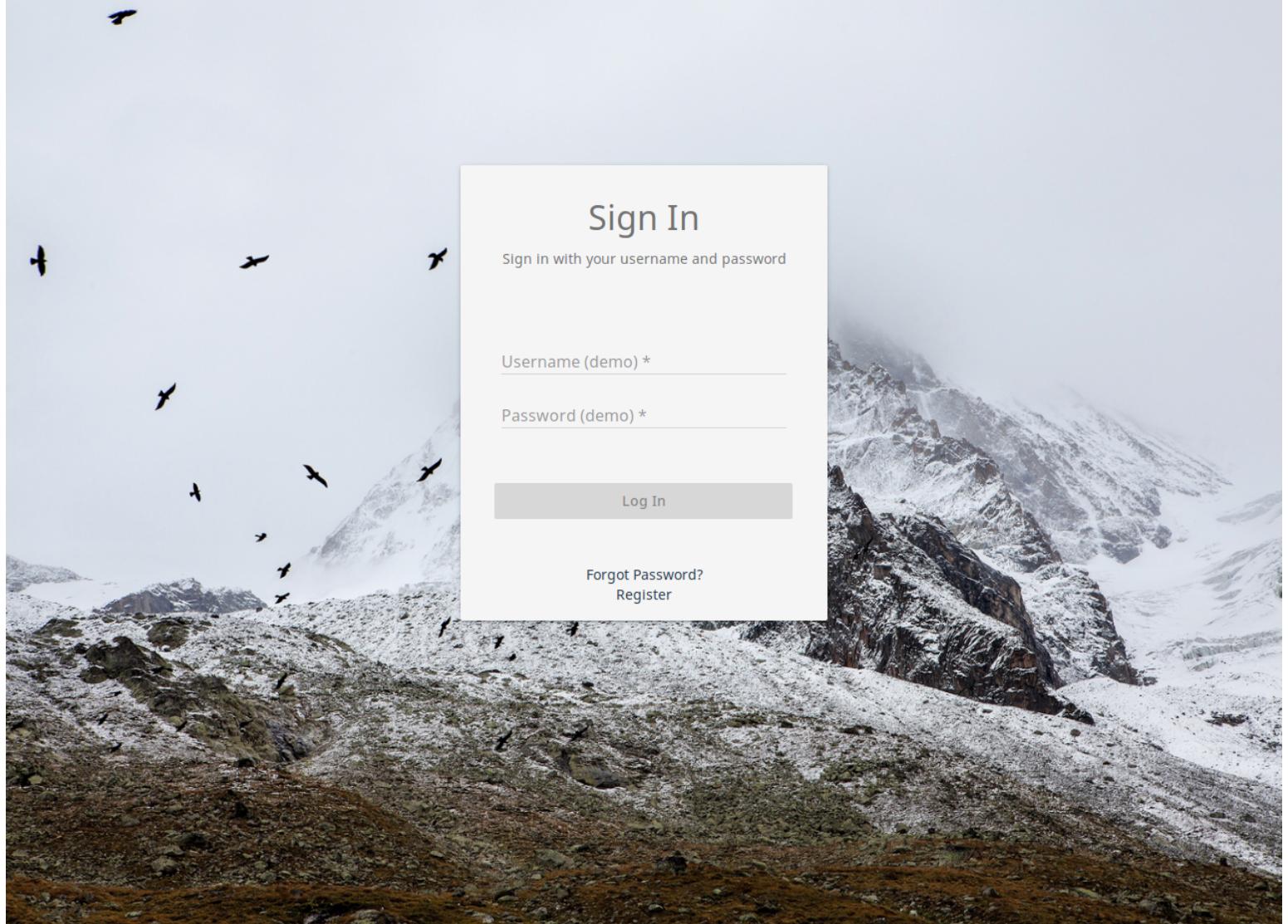


Рис. 9: Страница авторизации

Если у пользователя нет действующей почты он может перейти на страницу регистрации, нажав на ссылку Register.

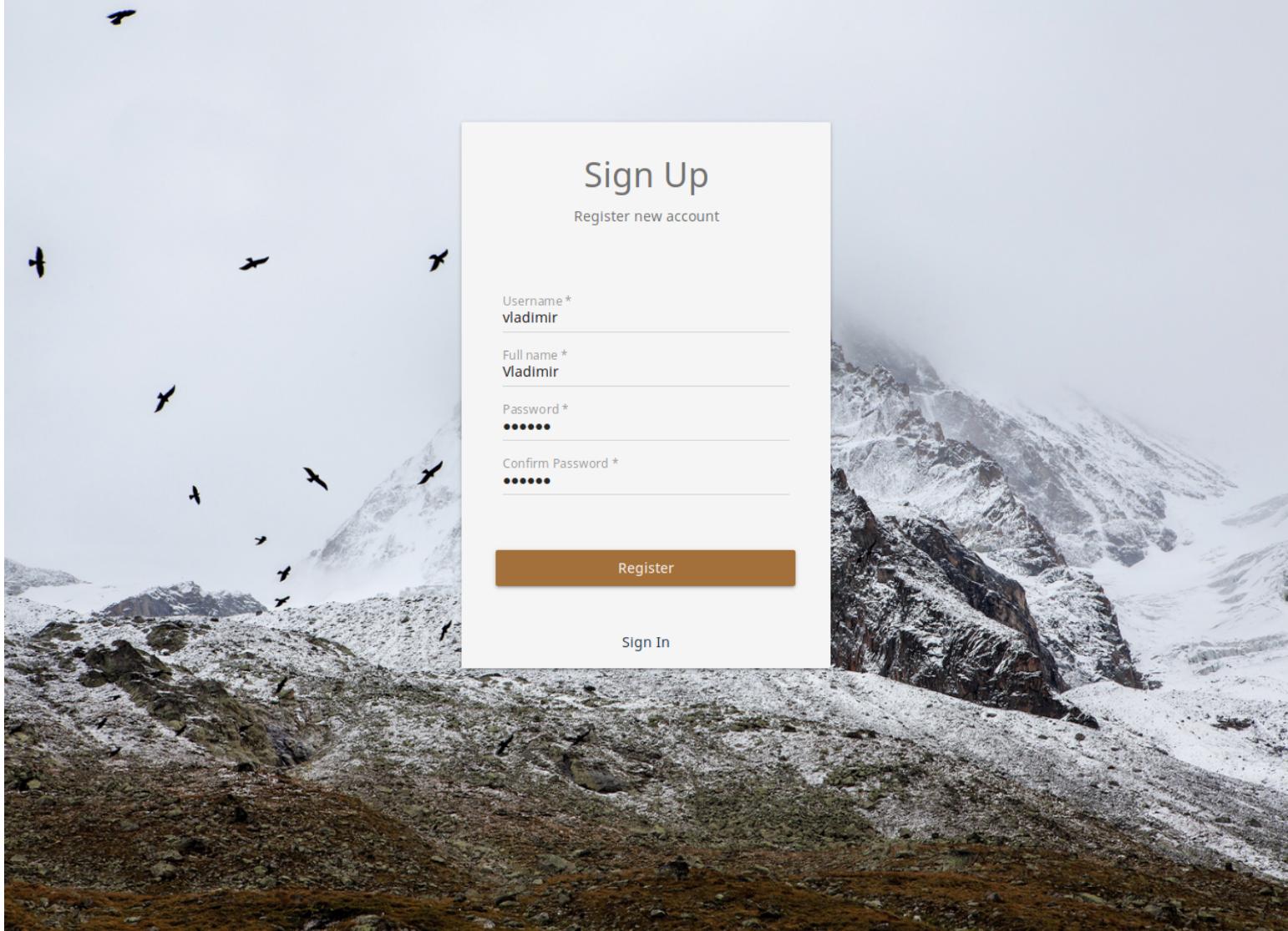


Рис. 10: Страница регистрации

После верного заполнения всех полей пользователя перенаправит на главную страницу с письмами.

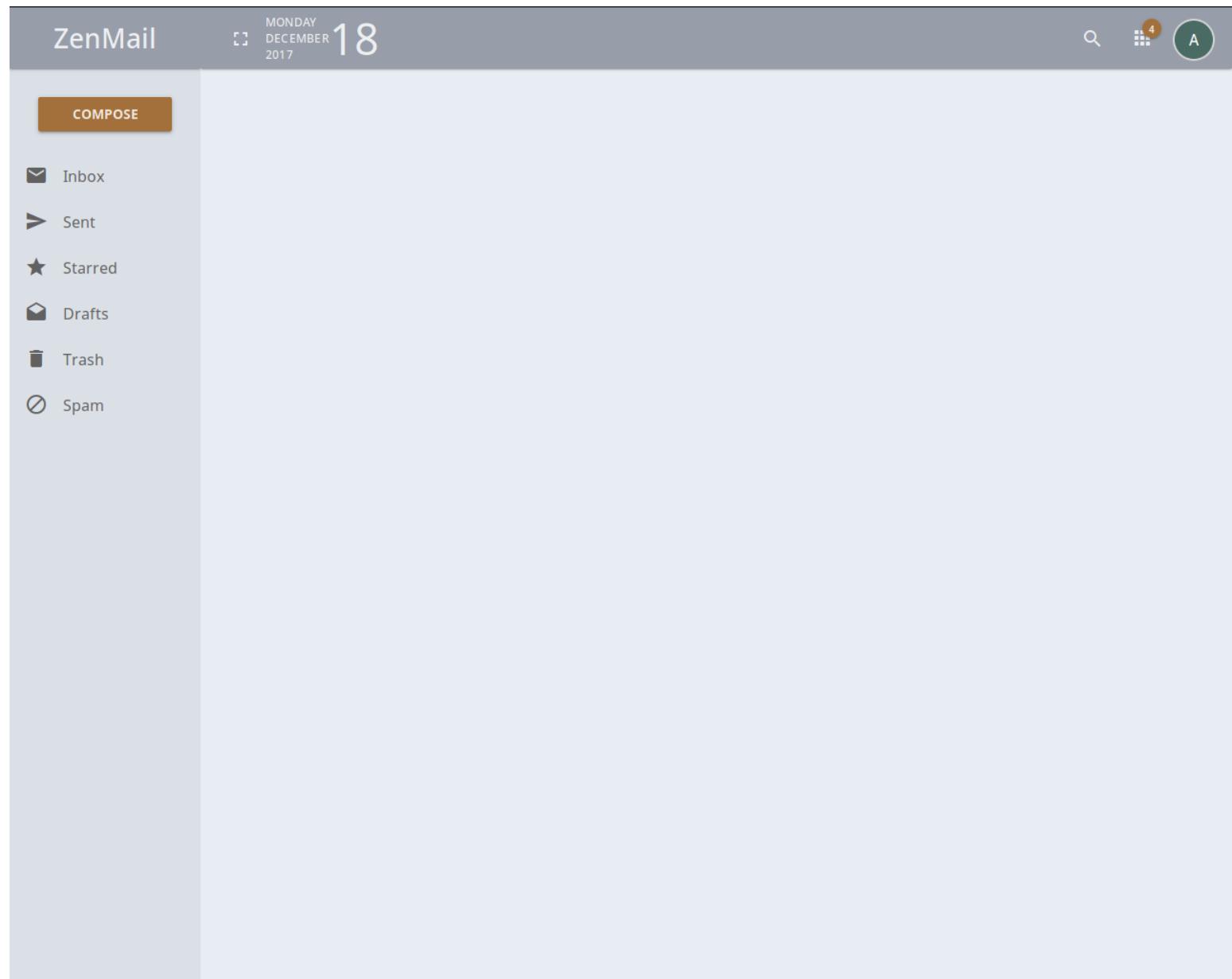


Рис. 11: Домашняя страница авторизированного аккаунта

После перехода на домашнюю страницу пользователь может управлять своим почтовым аккаунтом.

## 4.2 Входящие сообщения

При нажатии на кнопку Inbox открывается страница со всеми входящими сообщениями на данном аккаунте.

## COMPOSE

- Inbox
- Sent
- Starred
- Drafts
- Trash
- Spam

<input type="checkbox"/>		<input type="checkbox"/>	
<input type="checkbox"/>		andrewshipilo@mail.ru	Testing message for demonstration purposes — Hello! I hope you'll see this
<input type="checkbox"/>		m.i.sosnin@mail.ru	Re: Спецификация по курсовой работе — Привет Посмотрел код. Напоминаю,
<input type="checkbox"/>		Lebedew@zenmail.space	Re: Tty — Nät Sent from my Prestigio MultiP
<input type="checkbox"/>		Lebedew@zenmail.space	Tty — Sent from my Prestigio MultiPhon
<input type="checkbox"/>		Lebedew@zenmail.space	Re: Privet — Ggg Sent from my Prestigio MultiP
<input type="checkbox"/>		Lebedew@zenmail.space	Hedgehog is richtig nicht — Alles gut Sent from my Prestigio
<input type="checkbox"/>		Lebedew@zenmail.space	Re: Privet — My dear sweetie Sent from my Pre
<input type="checkbox"/>		Lebedew@zenmail.space	Seagul — Ur Stjärnstoft Är Vi Komna Sent
<input type="checkbox"/>		andrewshipilo@protonmail	Hello, i'm message sent from ProtonMail — Hello, nice to see you!
<input type="checkbox"/>		superdrond97@yandex.ru	Re: Hello — Hello Rude -- С уважением, Щи
<input type="checkbox"/>		andrewshipilo@mail.ru	Re: Hello — Nice to meet you -----
<input type="checkbox"/>		andrewshipilo@gmail.com	Re: Hello — Hello!!! On Fri, Oct 20, 2017 a
			Mon Dec 18 21:16:58 MSK 2017
			Fri Nov 10 21:34:21 MSK 2017
			Sat Oct 21 21:27:15 MSK 2017
			Sat Oct 21 18:54:17 MSK 2017
			Sat Oct 21 18:52:28 MSK 2017
			Sat Oct 21 18:52:27 MSK 2017
			Sat Oct 21 18:52:26 MSK 2017
			Sat Oct 21 18:52:18 MSK 2017
			Sat Oct 21 17:03:26 MSK 2017
			Sat Oct 21 16:59:21 MSK 2017
			Sat Oct 21 16:58:11 MSK 2017
			Fri Oct 20 22:04:27 MSK 2017

Рис. 12: Входящие сообщения

При нажатии на любое из сообщений мы можем открыть его

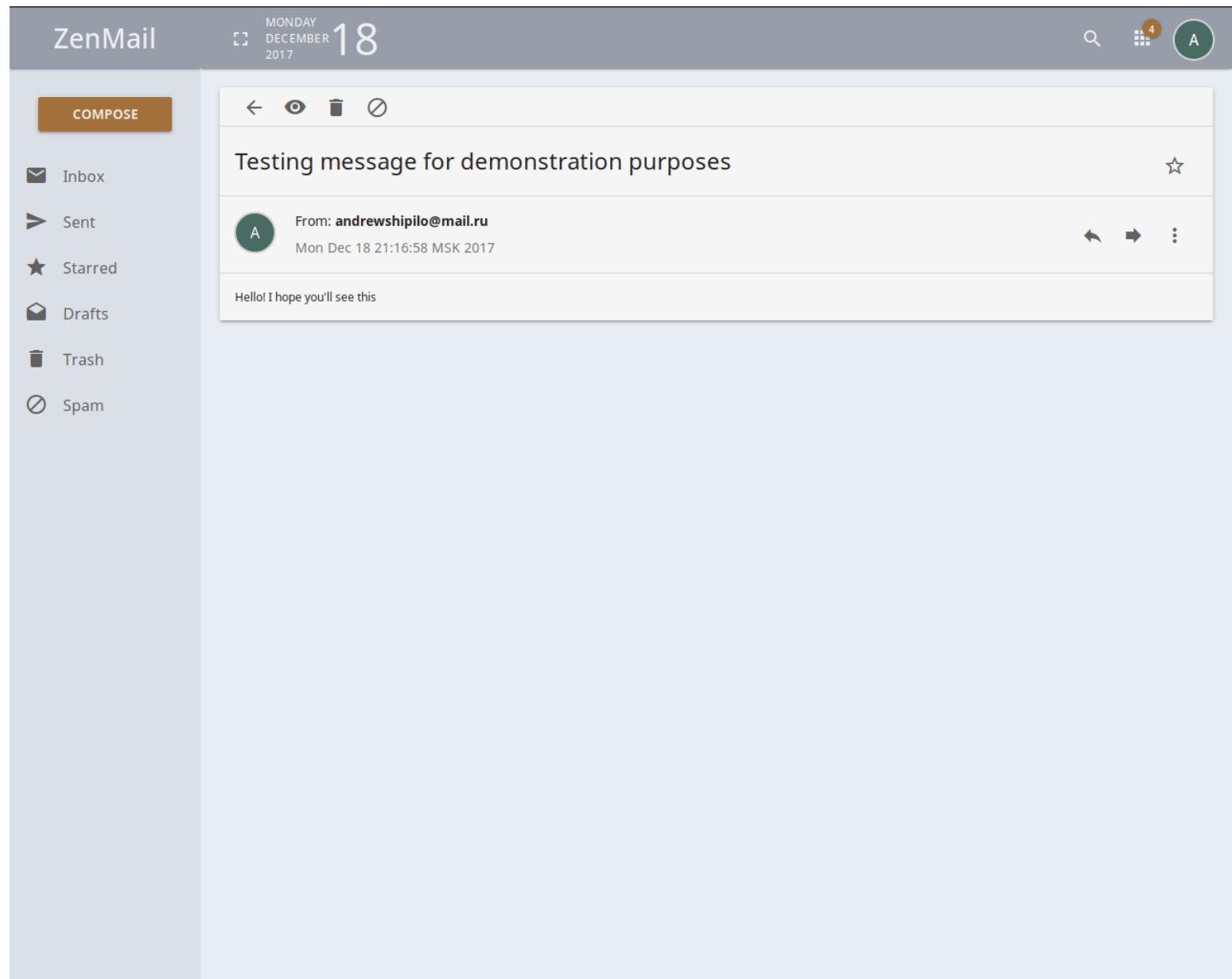


Рис. 13: Содержимое сообщения

### 4.3 Отправка сообщения

Для того, чтобы отправить сообщение нужно нажать на кнопку COMPOSE, после этого появится форма отправки сообщения, где можно указать получателя, тему и текст сообщения. Эту форму можно закрыть, нажав на крестик в правом углу. Для того, чтобы отправить сообщение необходимо нажать на самолетик.

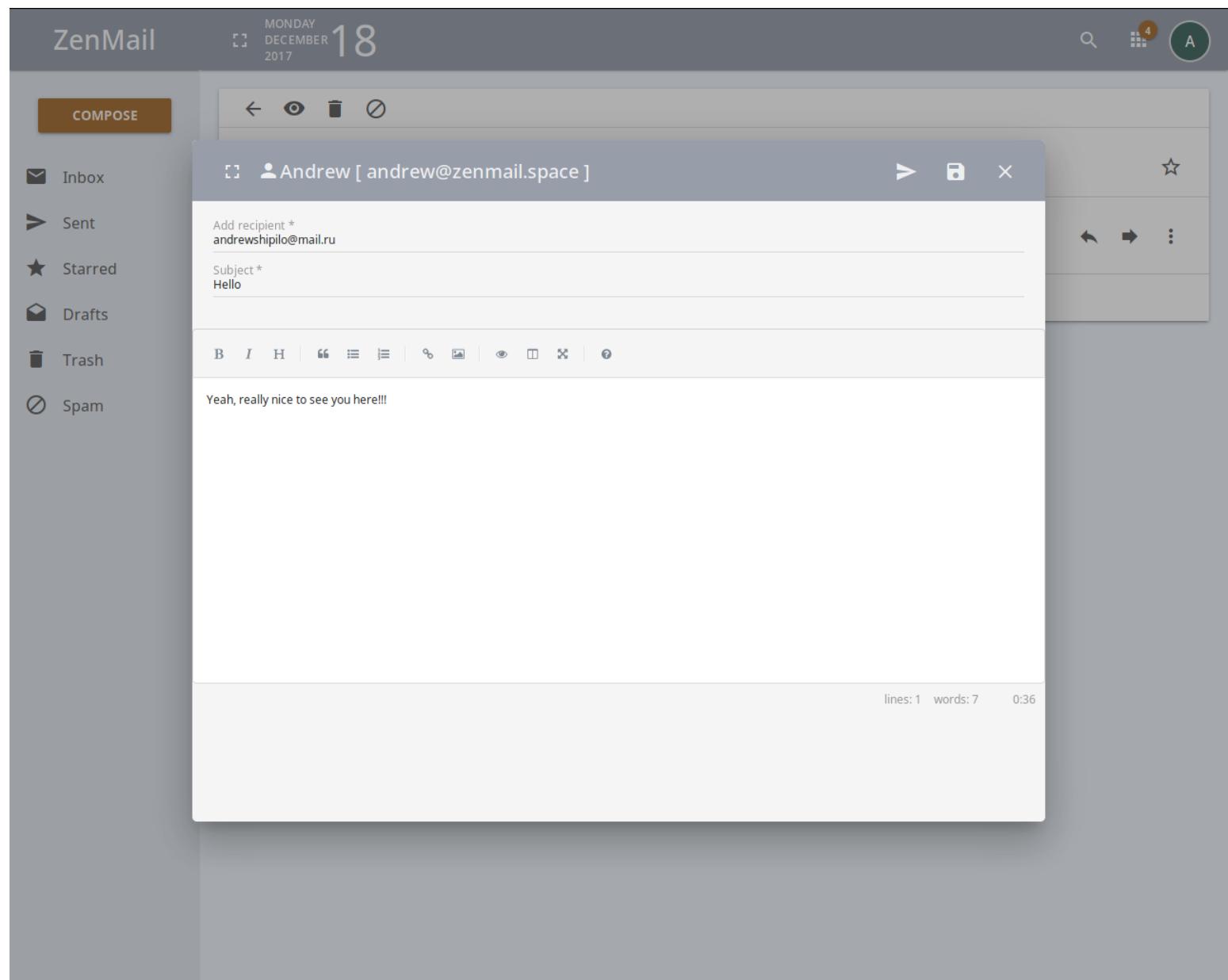


Рис. 14: Форма отправки сообщения

При отправке сообщения форма закроется и появится уведомление с результатом отправки сообщения.

**COMPOSE**

Inbox

Sent

Starred

Drafts

Trash

Spam

## Testing message for demonstration purposes



From: andrewshipilo@mail.ru  
Mon Dec 18 21:16:58 MSK 2017



Hello! I hope you'll see this

Message sent

OK

Рис. 15: Уведомление об отправке сообщения

## 5 Заключение

Итак, разработав почтовый сервис, включающий в себя веб-приложение, серверное приложение и почтовый сервер, мы проделали большую исследовательскую и производственную работу.

Был получен глубокий взгляд на разработку веб-приложений, а именно на RESTfull архитектуру.

Получены навыки работы с облачными серверами, а также получены базовые знания о работе реалиционных базах данны. Была разработана инфраструктура для запуска почтового сервиса на основе docker-compose. Также была исследована работа различных протоколов связи (SMTP, IMAP) и общая структура почтовой связи.

Получены навыки работы в команде: обсуждение, брейн-штурминг, тайм и таск менеджмент, использована система контроля версий git для кооперирования разработки. Были изучены и имплементированы базовые навыки test driven development.