ГУАП

КАФЕДРА №51

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

_____	_____	_____
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ)

Почтовый сервис ZenMail

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ II

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ ГР. № 5511

_____	_____
подпись, дата	инициалы, фамилия

Санкт-Петербург 2017

Содержание

1 Функциональная спецификация **2**
1.1 Цель . . . 2
1.2 Основные функции . . . 2
1.3 Технологии . . . 3
1.4 Команда . . . 3

2 Руководство пользователя **4**
2.1 Авторизация . . . 4
2.2 Входящие сообщения . . . 6
2.3 Отправка сообщения . . . 8

3 Описание архитектуры проекта **11**

4 Особенности реализации **12**

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 зарегестрированным 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 взаимодейтсвия между пользователем и программно-аппаратной частью веб-приложения, использующий веб-технологии для обработки информации полученной от пользователя на клиенете или отправляя информацию на *Backend*, предоставляя пользователю полный контроль над своим почтовым ящиком, включая регистрации, аутентификацию, а также просмотр и отправку сообщений.

Backend: это программа, исполняющаяся на пердполагаемом 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
In Memory DB	H2
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 защищеные протоколы связи.
- Заболотный Артем: занимается связью между программной и графической частями приложения, используя нативный для Angular 4 язык программированния 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 Руководство пользователя

2.1 Авторизация

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

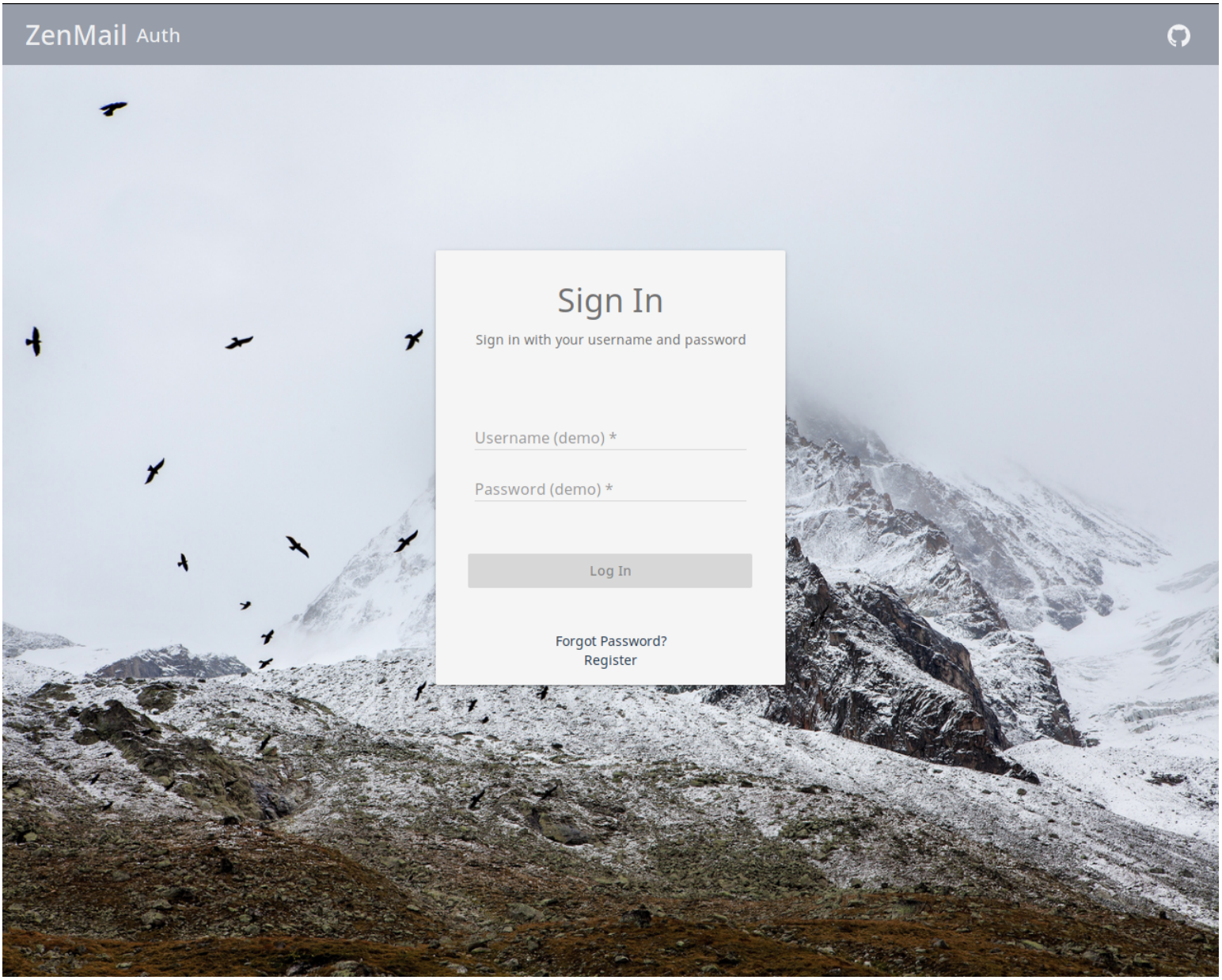

Рис. 1: Страница авторизации

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

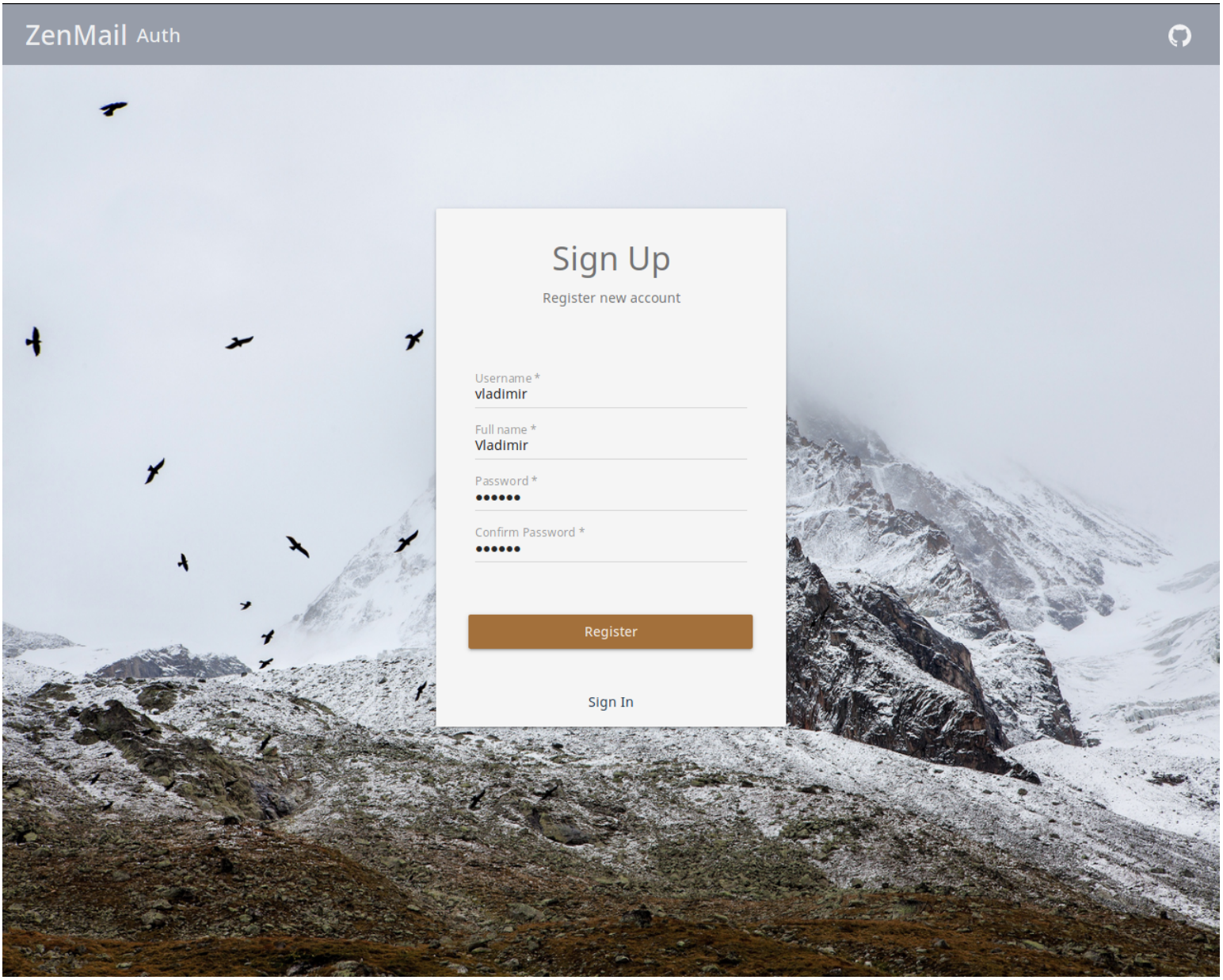

Рис. 2: Страница регистрации

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

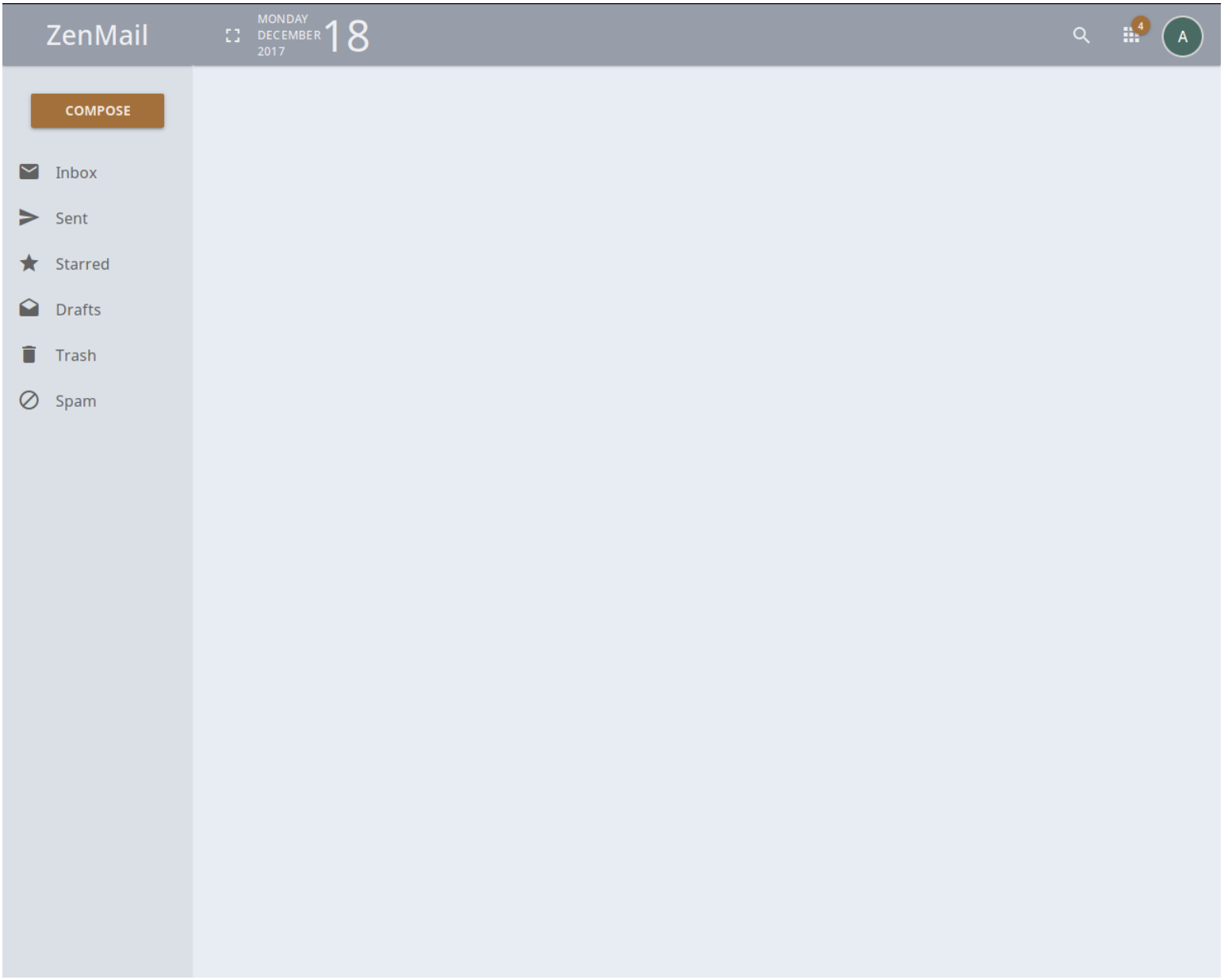

Рис. 3: Домашняя страница авторизированного аккаунта

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

2.2 Входящие сообщения

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

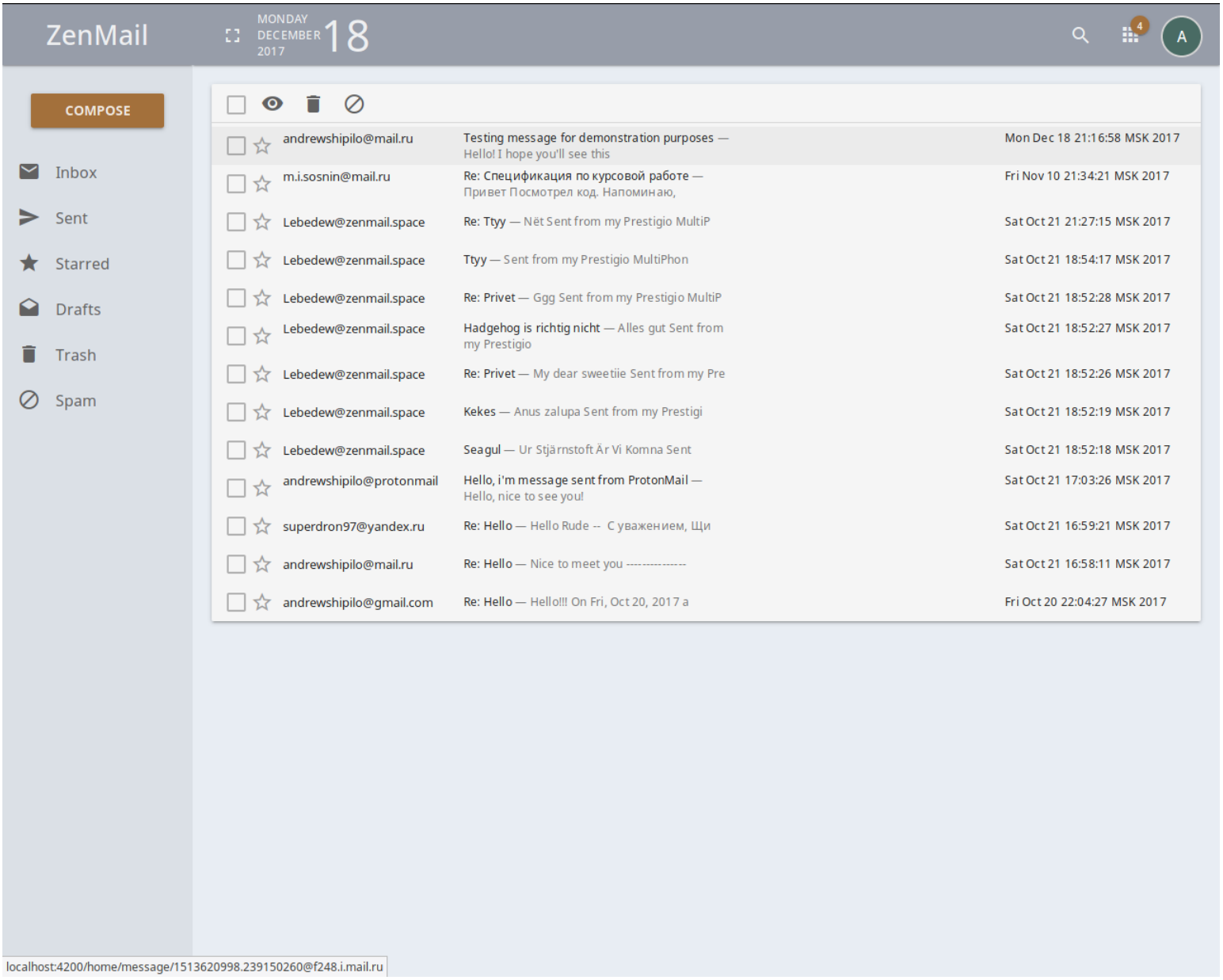

Рис. 4: Входящие сообщения

При нажатии на любое из сообщений мы можем открыть его

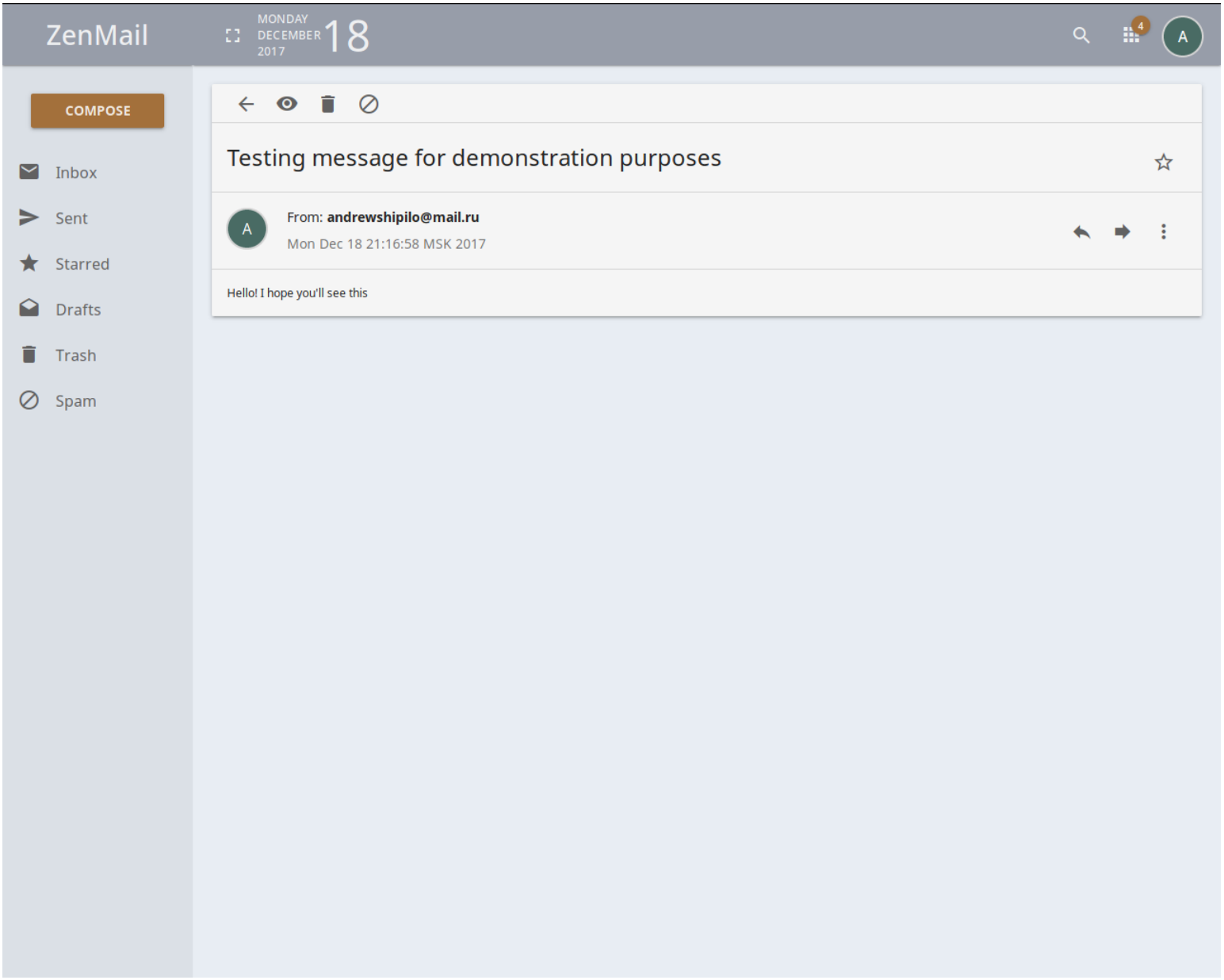

Рис. 5: Содержимое сообщения

2.3 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.

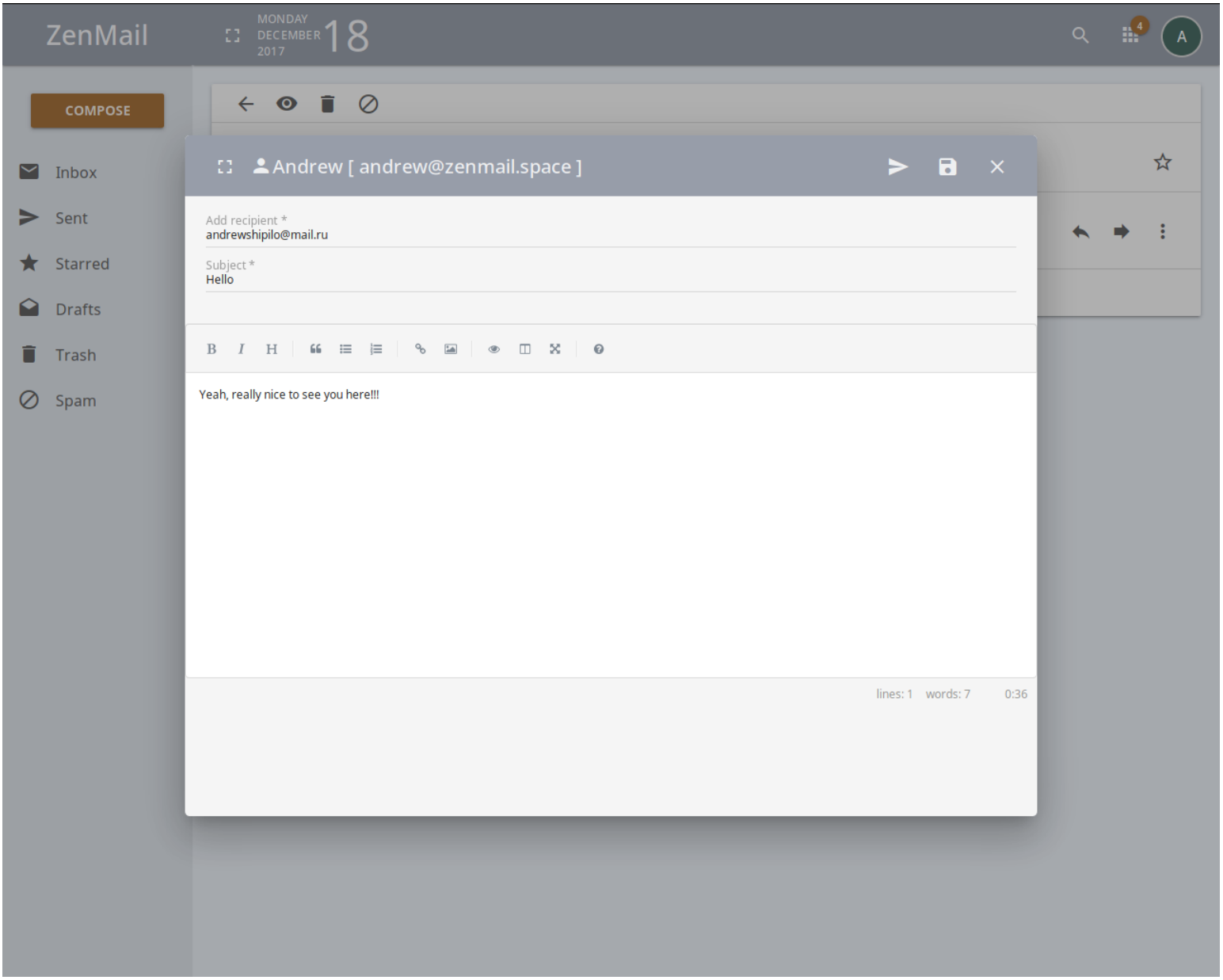

Рис. 6: Форма отправки сообщения

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

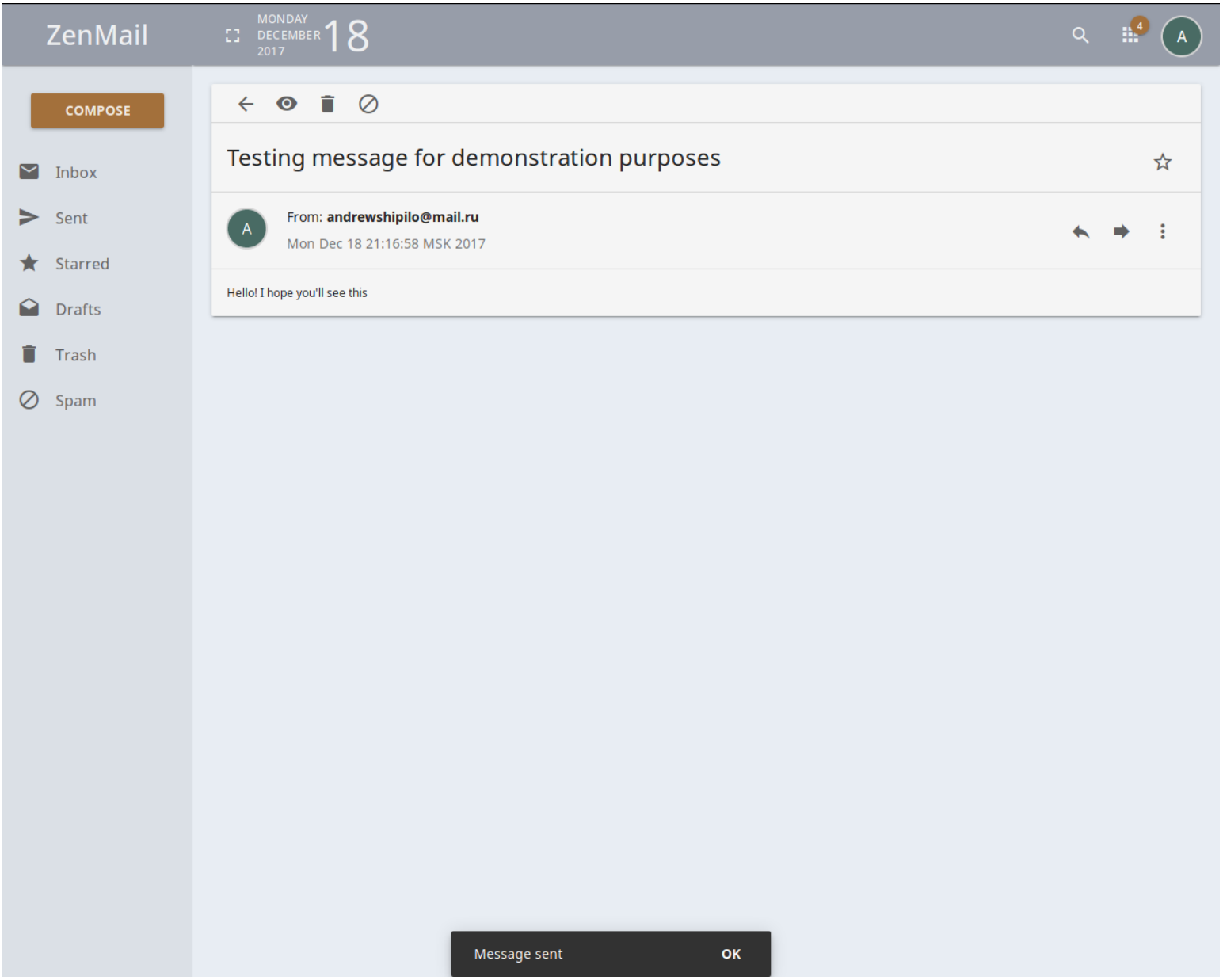

Рис. 7: Уведомление об отправке сообщения

3 Описание архитектуры проекта

Проект, как говорилось раньше, можно разделить на три главные составляющие: Frontend, Backend и MailServer. В общем виде Frontend посылает запрос Backend'у, и в большинстве случаев Backend посылает запрос базе, получает ответ, обрабатывает и отсылает на Frontend.
Общее взаимодействие между основными компонентами проекта можно наблюдать на схеме ниже.

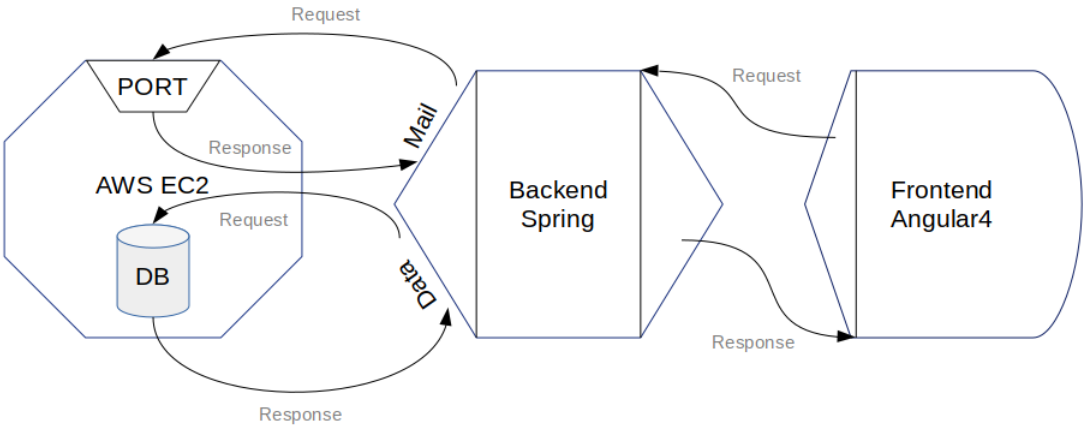

Рис. 8: Взаимодействие составляющих проекта

Веб-приложение организованно с помощью паттерна проектирования Model-View-Control (MVC), где моделями являются разлинчные 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.

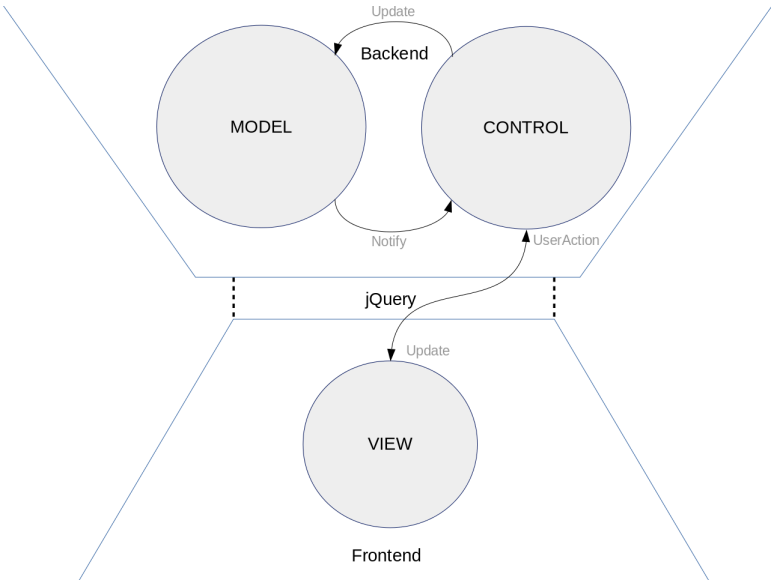

Рис. 9: Взаимодействие составляющих проекта

4 Особенности реализации

1. MainController – класс, который возвращает index.html, когда требуется mapping '/'. Нужен, чтобы выдавать экран загрузке при запуске веб-приложения.

2. UserService – класс, который огранизует 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, ввиде 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, ввиде класса User, о залогиненном пользователе.