|  |  |
| --- | --- |
| https://lh7-us.googleusercontent.com/QuBaagabzZYLr7U3ZbYOtMusd5FbWWxIvMUTN8jrLHzSLMg534z9gXRTIG1Us4i_lOwmWlaBxKedNt-SQ26dm4WmyqwjGDmEO6z8GE3QrZosqvHM88J2EFeVf1u0GzyCZQlhWmp1Zeo85tKo4LJVXQ | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» |

Институт № 3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 «Вычислительные машины, системы и сети»

**Отчет по лабораторной работе**

по учебной дисциплине «Web-технологии»

на тему «Почтовый сервер»

Руководители работы:

доц. каф.304, канд. тех. наук Титов Юрий Павлович

Ассистент Борисов Артём Ильич

Исполнители:

обучающиеся группы М3О-125БВ-24

Тихонов Дмитрий Сергеевич

Москва 2025

СОДЕРЖАНИЕ

[1. ЗАДАНИЕ №6. ПОЧТОВЫЙ СЕРВЕР 3](#_Toc196447799)

[2. РАЗРАБОТКА КЛИЕНТА ДЛЯ РАБОТЫ С ПОЧТОВЫМ СЕРВЕРОМ 4](#_Toc196447800)

[2.1. КОНФИГУРАЦИЯ ПРИЛОЖЕНИЯ 4](#_Toc196447801)

[2.2. РЕАЛИЗАЦИЯ ОСНОВНОГО МОДУЛЯ 5](#_Toc196447802)

[2.3. РЕАЛИЗАЦИЯ МЕХАНИЗМОВ ОТПРАВКИ ПИСЕМ 7](#_Toc196447803)

[2.4. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПОДСИСТЕМЫ МОДУЛЯ ПРОВЕРКИ ПОЧТЫ 10](#_Toc196447804)

[2.5. РЕАЛИЗАЦИЯ МОДУЛЯ ПОЛУЧЕНИЯ ПОЛНОГО ПИСЬМА 13](#_Toc196447805)

[2.6. СЦЕНАРИЙ С УСПЕШНОЙ РАБОТЫ КЛИЕНТА 15](#_Toc196447806)

[2.7. АНАЛИЗ ТРАФИКА В WIRESHARK. 17](#_Toc196447807)

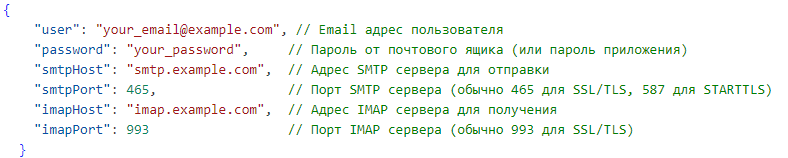
[2. ВЫВОД 19](#_Toc196447808)

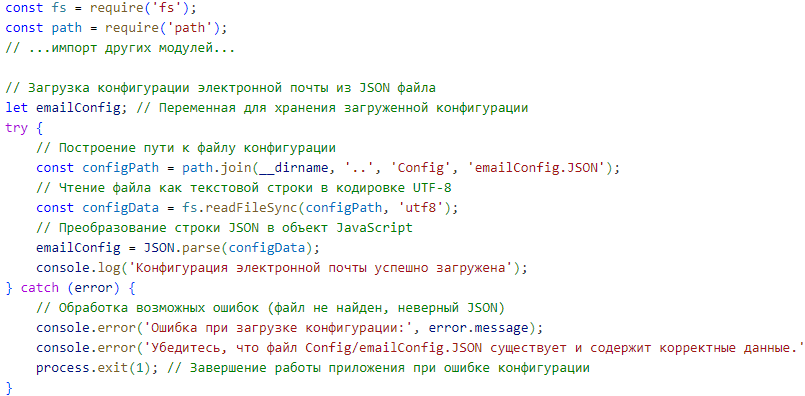
# Задание №6. Почтовый сервер

1. Добавить возможность комментирования задач в REST API сервисе. Пользователь может оставлять комментарии к задачам и видеть комментарии других пользователей.  
   Используя почтовые протоколы, отправить письмо на почтовый ящик одногруппника. В почтовом сообщении написать: тема сообщения – название лабораторной работы, в тексте сообщения написать ФИО и группу отправителя, ФИО и группу получателя и время отправки сообщения.   
   Установить флажок «Важное» для данного письма.
2. Используя протокол, получить письмо, зайти на свой ящик. Получить информацию (и привести в отчете) о количестве писем на ящике. Привести в отчете заголовки некоторых писем (не более 5 шт.). Получить письмо, присланное одногруппником из почтового ящика. Привести его в отчете.
3. Получить письмо с вложенным файлом, получить вложенный файл.
4. Провести анализ передаваемых команд и данных с помощью программы Wireshark.

# Разработка КЛИЕНТА ДЛЯ РАБОТЫ С почтовым сервером

## Конфигурация приложения

Для взаимодействия с почтовым сервером (как для отправки, так и для получения) необходимы учетные данные пользователя (логин, пароль) и адреса/порты соответствующих серверов (SMTP и IMAP). Эти чувствительные данные хранятся в формате JSON в файле emailConfig.JSON для легкого чтения и изменения без модификации основного кода.  


Загрузка этой конфигурации осуществляется при запуске приложения в client.js. Используется модуль fs для синхронного чтения содержимого файла (fs.readFileSync) и модуль path для построения корректного пути к файлу конфигурации относительно текущего файла (\_\_dirname). Содержимое файла затем парсится из строки JSON в объект JavaScript с помощью JSON.parse. Предусмотрена обработка ошибок на случай отсутствия файла или некорректного JSON-формата.

## Реализация основного модуля

Этот файл служит точкой входа и оркестратором приложения.

Используется модуль readline для создания интерактивного интерфейса (rl). readline.createInterface настраивает чтение из стандартного потока ввода (process.stdin) и запись в стандартный поток вывода (process.stdout).

Основная логика заключена в асинхронной функции main. Она выводит приветственное сообщение и использует rl.question для запроса у пользователя выбора действия (отправить или проверить почту).

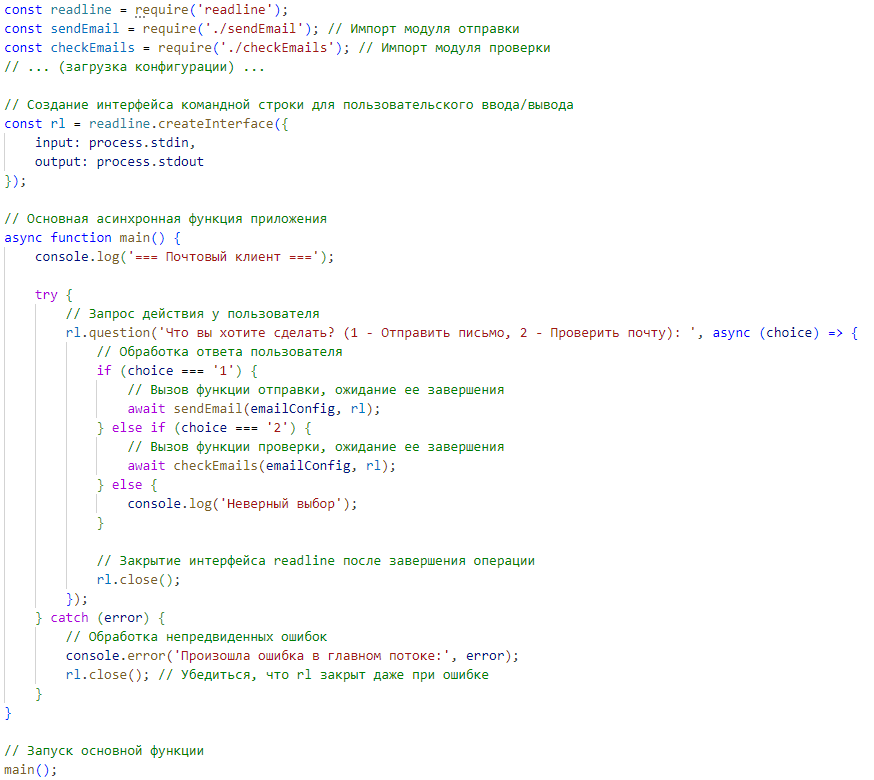
rl.question принимает строку запроса и функцию обратного вызова (callback), которая выполняется после ввода пользователя. Внутри callback анализируется ответ (choice).

В зависимости от выбора ('1' или '2'), асинхронно вызывается (await) соответствующая функция из импортированных модулей (sendEmail или checkEmails), передавая ей объект конфигурации (emailConfig) и интерфейс readline (rl) для дальнейшего взаимодействия с пользователем внутри этих модулей.

После выполнения выбранного действия или при неверном вводе, интерфейс readline закрывается (rl.close()), что завершает работу приложения. Предусмотрен блок try...catch для отлова общих ошибок на верхнем уровне.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Контент, сгенерированный ИИ, может содержать ошибки.



## Реализация механизмов отправки писем

Этот модуль инкапсулирует всю логику, связанную с отправкой писем по протоколу SMTP.

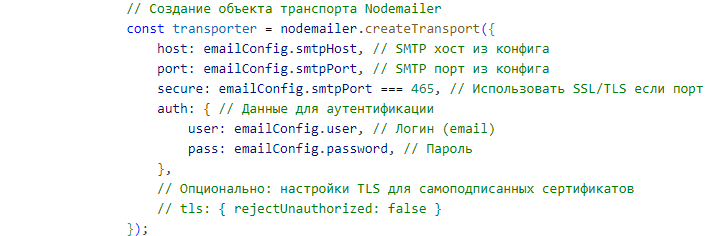
Используется библиотека nodemailer для упрощения взаимодействия с SMTP-сервером.

Функция sendEmail обернута в Promise, чтобы обеспечить асинхронное выполнение и возможность ожидания (await) в client.js. Она принимает объект конфигурации и интерфейс readline.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.Внутри промиса последовательно используются rl.question для сбора необходимой информации от пользователя: email получателя, тема, ФИО отправителя/получателя, группа, и опционально путь к файлу для вложения. Каждый следующий вопрос задается в callback предыдущего, создавая цепочку асинхронных запросов.

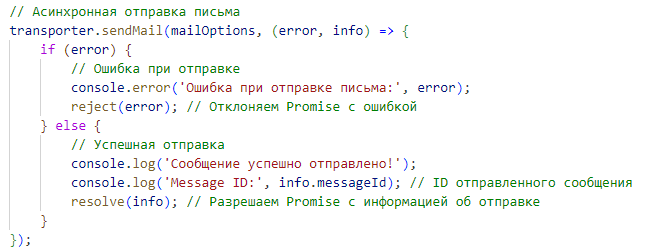
После сбора всех данных создается объект transporter с помощью nodemailer.createTransport. Этот объект конфигурируется данными SMTP-сервера (хост, порт, безопасность) и учетными данными пользователя из [emailConfig](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "). Параметр secure: (true) используется, если порт 465 (SSL/TLS).



Далее формируется объект mailOptions, содержащий все параметры письма: from (от кого, с форматированием имени), to (кому), subject (тема), text (тело письма, сформированное из введенных данных), priority: 'high' (установка флага "Важное"), и опционально headers для пользовательских заголовков.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.Реализована проверка наличия пути к вложению. Если путь указан и файл существует (fs.existsSync), он добавляется в массив attachments объекта mailOptions. Используется path.basename для получения имени файла из пути.

Наконец, вызывается метод transporter.sendMail(mailOptions, callback). Он асинхронно отправляет письмо. Функция обратного вызова (callback) получает error (если отправка не удалась) или info (при успехе). В зависимости от результата вызывается reject(error) или resolve(info) для завершения промиса, созданного в начале функции sendEmail. Результат выводится в консоль.  


## Проектирование и реализация подсистемы модуля проверки почты

Этот модуль отвечает за подключение к почтовому ящику по протоколу IMAP и получение информации о письмах.

Используется библиотека imap. simpleParser из mailparser импортирован, но не используется в текущей версии для получения заголовков (он понадобится для разбора полного тела письма).

Функция checkEmails также возвращает Promise для асинхронной работы. Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Создается экземпляр Imap, конфигурируемый данными IMAP-сервера и учетными данными из emailConfig. Важно установить tls: true для защищенного соединения и tlsOptions: { rejectUnauthorized: false }, если сервер использует самоподписанный сертификат (в рабочих системах это обычно не требуется).

Устанавливаются обработчики событий для объекта imap:

imap.once('ready', callback): Срабатывает после успешного подключения и аутентификации. Внутри этого callback происходит основная работа с почтовым ящиком.

imap.once('error', callback): Срабатывает при ошибке подключения или другой ошибке IMAP. Вызывает reject промиса.

imap.once('end', callback): Срабатывает при закрытии соединения.

После установки обработчиков вызывается imap.connect() для инициации подключения.

Внутри обработчика ready вызывается imap.openBox('INBOX', false, callback), чтобы открыть основную папку входящих сообщений ('INBOX'). false указывает, что папка открывается не в режиме "только для чтения".

В callback функции openBox (если нет ошибки) выводится общее количество сообщений (box.messages.total).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.Затем используется imap.seq.fetch для запроса данных писем. Диапазон ${Math.max(1, box.messages.total - 4)}:${box.messages.total} выбирает последние 5 писем (или меньше, если их всего меньше 5). Опция bodies: 'HEADER.FIELDS (FROM TO SUBJECT DATE)' указывает, что нужно загрузить только указанные поля заголовка для экономии трафика и времени.

Объект f (результат imap.seq.fetch) генерирует события:

f.on('message', function(msg, seqno){...}): Срабатывает для каждого полученного сообщения в запрошенном диапазоне. msg - объект сообщения, seqno - его порядковый номер.

Внутри обработчика message устанавливается обработчик msg.on('body', function(stream, info){...}), который получает поток (stream) с данными запрошенного заголовка (HEADER.FIELDS (...)).

Данные из потока читаются по частям (stream.on('data', ...)) и собираются в buffer.

Когда поток завершается (stream.once('end', ...)), накопленный buffer парсится с помощью Imap.parseHeader. Результат (объект с полями from, subject, date и др.) используется для формирования упрощенного объекта заголовка, который добавляется в массив headers.  
Изображение выглядит как текст, снимок экрана, документ, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

## Реализация модуля получения полного письма

Этот модуль инкапсулирует логику получения полного содержимого конкретного электронного письма по его порядковому номеру ([seqno](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) из почтового ящика, его разбора и отображения деталей, включая обработку вложений.

В начале проверяется, что IMAP-соединение готово к выполнению запросов (imap.state === 'authenticated'). Запрос полного тела: Выполняется imap.seq.fetch(seqno, { bodies: '' }). Опция bodies: '' указывает библиотеке imap загрузить всё содержимое письма (заголовки и тело в формате MIME).  


Данные письма поступают асинхронно через события. Обработчик [f.on('message', ...)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) и вложенные обработчики [msg.on('body', ...)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) и [stream.on('data', ...)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) собирают все части письма в строку [fullEmailData](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

 Установлен обработчик [f.once('error', ...)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) для отлова ошибок на этапе загрузки письма с сервера. После завершения загрузки ([f.once('end', ...)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)), строка [fullEmailData](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") передается в асинхронную функцию [simpleParser](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) из библиотеки mailparser.

Изображение выглядит как текст, Шрифт, снимок экрана, рукописный текст

Контент, сгенерированный ИИ, может содержать ошибки.

Результат парсинга ([parsed](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")) содержит удобные поля: [parsed.from](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [parsed.to](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [parsed.subject](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [parsed.date](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [parsed.text](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [parsed.html](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [parsed.attachments](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Основные поля и текстовое содержимое ([parsed.text](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")) выводятся в консоль. Если текстовой части нет, выводится уведомление.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Проверяется наличие массива [parsed.attachments](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "). Если вложения есть, информация о них (имя файла, тип, размер) выводится в консоль. Реализовано сохранение файла: используется [fs.writeFileSync(attachment.filename, attachment.content)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") для записи содержимого вложения ([attachment.content](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")) в файл с оригинальным именем ([attachment.filename](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")) в текущую рабочую директорию. Предусмотрена обработка ошибок записи файла.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

В блоке finally после try...catch (гарантирует выполнение) происходит закрытие IMAP-соединения ([imap.end()](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)), чтобы освободить ресурсы сервера, независимо от того, успешно ли прошло получение и разбор письма или произошла ошибка. Промис разрешается ([resolve()](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) при успехе или отклоняется ([reject(err)](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) при ошибке.

## Сценарий с успешной работы клиента

Приложение запускается командой node App/client.js. Пользователь выбирает опцию "1" для отправки письма.  
Изображение выглядит как текст, Шрифт, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

После выбора опции "1" управление передается модулю [sendEmail.js](vscode-file://vscode-app/c:/Users/oisa0/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Приложение последовательно запрашивает у пользователя все необходимые данные для отправки письма: email получателя, тему, ФИО и группу отправителя, ФИО и группу получателя, а также путь к вложению.  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

После ввода всех данных приложение использует nodemailer для подключения к SMTP-серверу (согласно emailConfig.JSON) и отправки сформированного письма с указанными параметрами и высоким приоритетом. В случае успеха выводится подтверждающее сообщение и идентификатор отправленного письма (Message ID).

Проверяем, что пользователю действительно пришло письмо.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Запустив приложение и выбрав опцию проверки почты, пользователь видит заголовки последних писем. Затем пользователь вводит порядковый номер нужного письма. Приложение загружает полное содержимое этого письма, разбирает его структуру и выводит основную информацию (отправитель, тему, текст). Если в письме обнаружены вложения, их детали отображаются, а сами файлы сохраняются в рабочую директорию приложения. Выводится подтверждение сохранения файлов, после чего соединение закрывается, и приложение завершает работу.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

## Анализ трафика в Wireshark.

Для проверки корректности и безопасности сетевого взаимодействия клиента с почтовыми серверами был проведен анализ трафика с использованием Wireshark. В ходе анализа были зафиксированы следующие ключевые моменты:

Установление TCP-соединения: Клиент успешно устанавливает TCP-соединения с IP-адресами, соответствующими доменным именам smtp.mail.ru и imap.mail.ru, на порты 465 (SMTPS) и 993 (IMAPS) соответственно.

Изображение выглядит как электроника, текст, снимок экрана, компьютер

Контент, сгенерированный ИИ, может содержать ошибки.

Инициация TLS-шифрования: непосредственно после установления TCP-соединения наблюдается обмен пакетами TLS Handshake (Client Hello, Server Hello и т.д.). Это подтверждает, что клиент корректно инициирует шифрование канала связи согласно требованиям протоколов SMTPS и IMAPS

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

 Весь последующий обмен данными, включая аутентификационные данные, команды протоколов и содержимое писем, происходит внутри зашифрованного TLS-канала (пакеты Application Data). Анализ не выявил передачи какой-либо чувствительной информации в открытом виде.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Результаты анализа сетевого трафика подтверждают, что клиент обеспечивает безопасное соединение с почтовыми серверами, используя шифрование TLS для защиты передаваемых данных.

# Вывод

В процессе выполнения лабораторной работы был успешно разработан консольный почтовый клиент на платформе Node.js, что позволило освоить ключевые аспекты взаимодействия с электронной почтой на программном уровне.

В ходе работы были получены практические навыки в следующих областях:

* Реализация функционала отправки электронных писем по протоколу SMTP, включая обработку вложений.
* Осуществление подключения к почтовому ящику и получение информации о письмах по протоколу IMAP.
* Извлечение и отображение заголовков писем, а также получение полного содержимого выбранного сообщения.

Полученные в процессе выполнения лабораторной работы компетенции формируют важную основу для разработки сетевых приложений и понимания механизмов работы стандартных интернет-протоколов.