**Tugas Individu**

**Email Protokol**

***“Untuk Memenuhi Salah Satu Tugas”***

**Praktikum Pemograman Jaringan**

****

**Dosen Pengampu :**

**RANDI PROSKA SANDRA, S.Pd., M.Sc.**

**Disusun Oleh :**

**SALSABILLA AHMAD**

**NIM : 21343072**

**Seksi : 202313430091**

**Program Studi S1 Teknik Informatika**

**Fakultas Teknik**

**Universitas Negeri Padang**

**Tahun 2023**

1. **Latar Belakang Program Aplikasi**

Aplikasi email, singkatan dari electronic mail, merupakan suatu sistem komunikasi elektronik yang memungkinkan pengguna untuk mengirim dan menerima pesan secara elektronik melalui internet. Protokol email, atau sering disebut sebagai protokol komunikasi, adalah serangkaian aturan dan standar yang mengatur cara data dikirim, diterima, dan disimpan dalam konteks layanan email. Salah satu protokol email yang paling umum digunakan adalah Simple Mail Transfer Protocol (SMTP) untuk pengiriman pesan, dan Post Office Protocol (POP) atau Internet Message Access Protocol (IMAP) untuk menerima dan mengakses pesan.

Latar belakang pengembangan protokol email melibatkan evolusi teknologi komunikasi dan pertumbuhan pengguna internet, dimulai dari penggunaan email dalam jaringan lokal hingga menjadi salah satu alat komunikasi paling penting dan universal di seluruh dunia, mendukung pertukaran informasi secara cepat dan efisien

1. **Langkah – Langkah Pembuatan Aplikasi**
2. Sourcode Server.js

const express = require('express');

const appRoute = require('./routes/route.js') ( mengimpor modul express dan modul route.js dari folder routes. Modul express digunakan untuk membuat server, sementara appRoute adalah objek yang berisi definisi rute-rute HTTP yang akan digunakan oleh server. )

const app = express(); ( Membuat objek aplikasi Express. Objek ini akan digunakan untuk mengkonfigurasi dan menjalankan server web. )

const PORT = process.env.PORT || 5000;

app.use(express.json()); ( untuk memungkinkan server memahami dan menguraikan data JSON yang diterima dari permintaan HTTP. )

/\*\* routes \*/

app.use('/api', appRoute); ( rute yang didefinisikan di dalam appRoute pada path '/api'. Ini berarti semua permintaan yang datang ke server pada path yang dimulai dengan '/api' akan diarahkan ke rute-rute yang didefinisikan di dalam appRoute. )

app.listen(5000, () => {

console.log('Server is running on http://localhost:${PORT}')

}) ( Menjalankan server pada port yang telah ditentukan sebelumnya (PORT). Saat server dimulai, pesan 'Server is running on <http://localhost:PORT>' akan dicetak ke konsol. )

****

1. Sourcode package.json

{

"name": "mailerr",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"start": "nodemon server.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"express": "^4.18.2",

"mailgen": "^2.0.28",

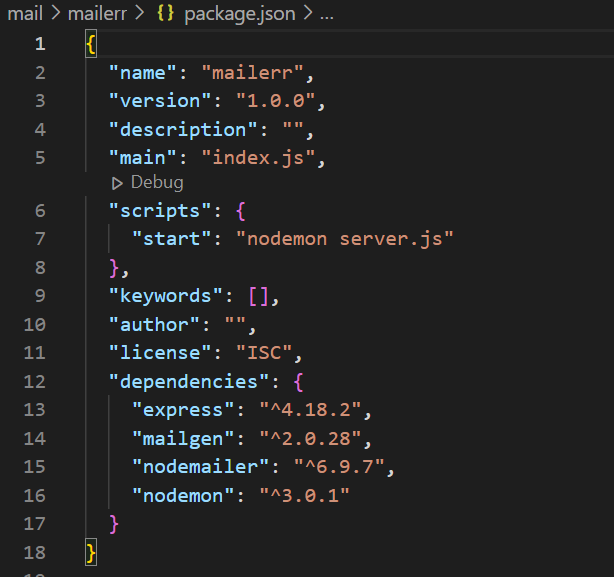
"nodemailer": "^6.9.7",

"nodemon": "^3.0.1"

}

}

package.json adalah file konfigurasi yang digunakan dalam proyek Node.js. File ini menyimpan metadata proyek, daftar dependensi, skrip build, dan konfigurasi lainnya. Kegunaan codingan pada package.json sangat penting untuk memahami dan mengelola proyek Node.js



1. Sourcode app.Controller.js

const nodemailer = require('nodemailer');

const Mailgen = require('mailgen');

const { EMAIL, PASSWORD } = require('../env.js');

const signup = async (req, res) => {

let testAccount = await nodemailer.createTestAccount();

// create reusable transporter object using the default SMTP transport

let transporter = nodemailer.createTransport({

host: "smtp.ethereal.email",

port: 587,

secure: false, // true for 465, false for other ports

auth: {

user: testAccount.user, // generated ethereal user

pass: testAccount.pass, // generated ethereal password

},

});

let message = {

from: '"Fred Foo 👻" <foo@example.com>', // sender address

to: "bar@example.com, baz@example.com", // list of receivers

subject: "Salsabilla Ahmad", // Subject line

text: "Email Caca", // plain text body

html: "Halooo Cacaca", // html body

}

transporter.sendMail(message).then((info) => {

return res.status(201)

.json({

msg: "Terima Email Caca",

info : info.messageId,

preview: nodemailer.getTestMessageUrl(info)

})

}).catch(error => {

return res.status(500).json({ error })

})

// res.status(201).json("Signup Successfully...!");

}

/\*\* send mail \*/

const getbill = (req, res) => {

const { userEmail } = req.body;

let config = {

service : 'gmail',

auth: {

user: EMAIL,

pass: PASSWORD

}

}

let transporter = nodemailer.createTransport(config)

let MailGenerator = new Mailgen({

theme: "default",

product : {

name: "Mailgen",

link : 'https://mailgen.js/'

}

})

let response = {

body: {

name : "Salsabilla Ahmad",

intro: "your bill has arrived!",

table : {

data : [

{

item : "Nodemailer Stack Book",

description: "A Backend application",

price: "$10.99",

}

]

},

outro: "Looking forwward to do more business"

}

}

let mail = MailGenerator.generate(response)

let message = {

from : EMAIL,

to : userEmail,

subject: "Place Order",

html: mail

}

transporter.sendMail(message).then(() => {

return res.status(201).json({

msg: "you should receive an email"

})

}).catch(error => {

return res.status(500).json({ error })

})

// res.status(201).json("getBill Successfully...!");

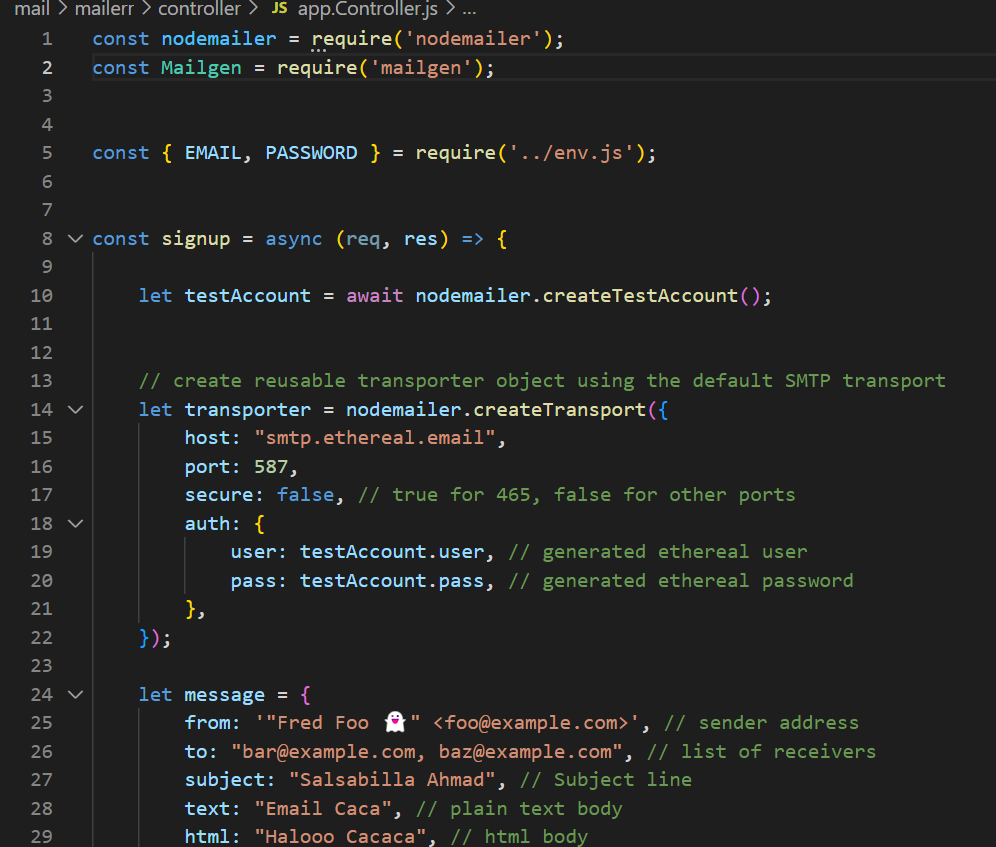
}

module.exports ={

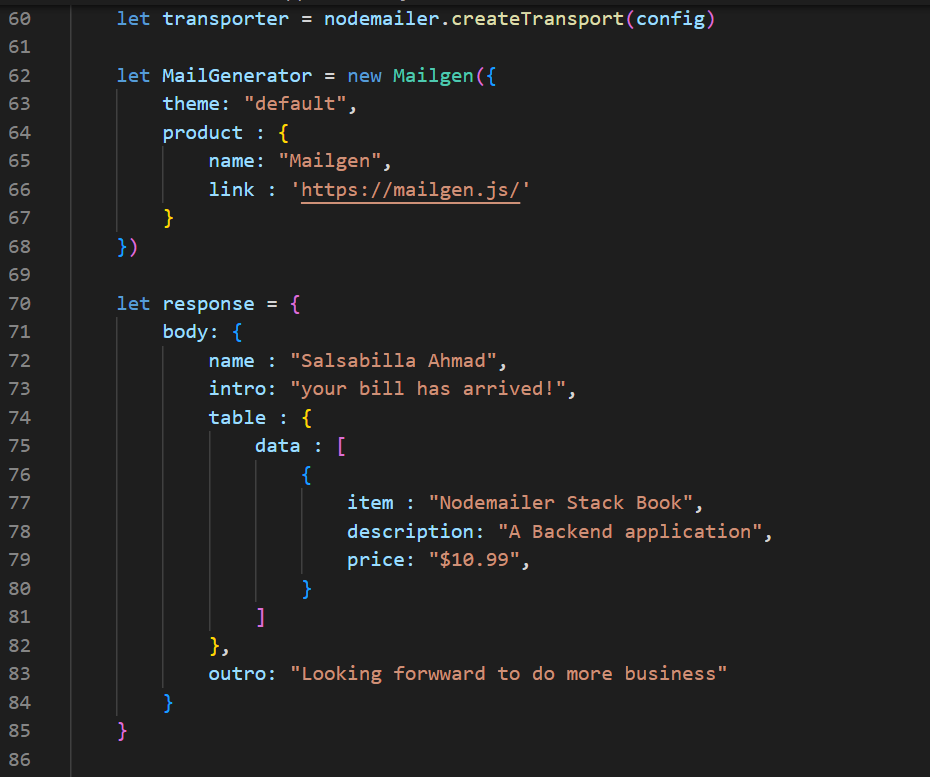
signup,

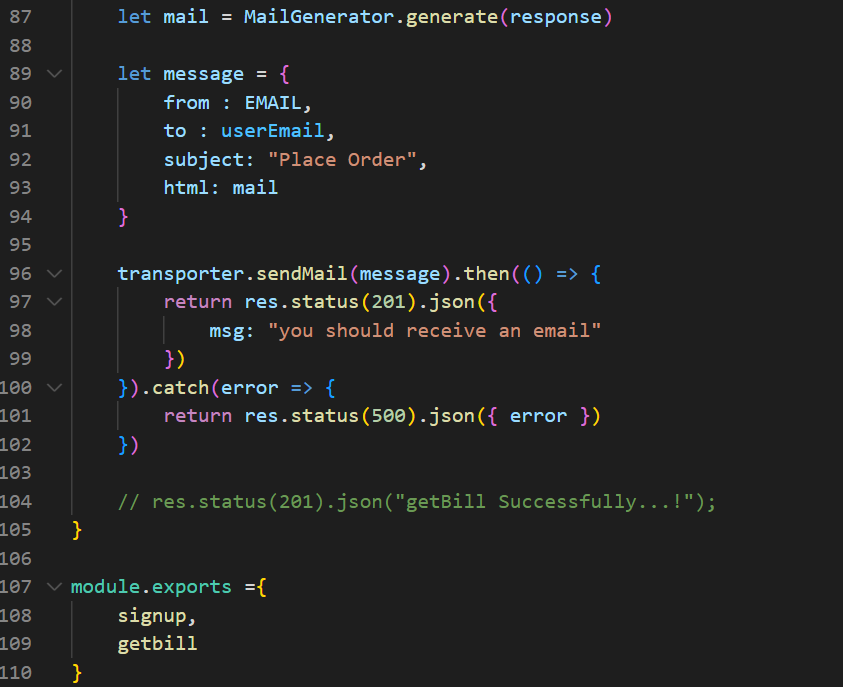
getbill

}









1. Sourcode route.js

const router = require('express').Router();

const { signup, getbill } = require('../controller/app.Controller.js')

/\*\* HTTP Request \*/

router.post('/user/signup', signup );

router.post('/product/getBill', getbill );

module.exports = router;

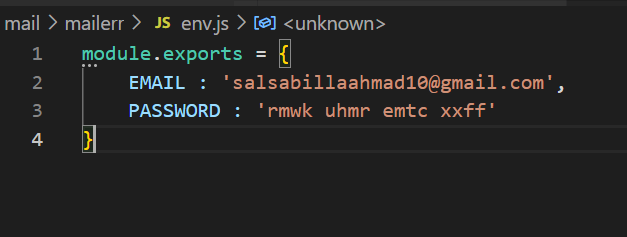
1. Sourcode env.js

module.exports = {

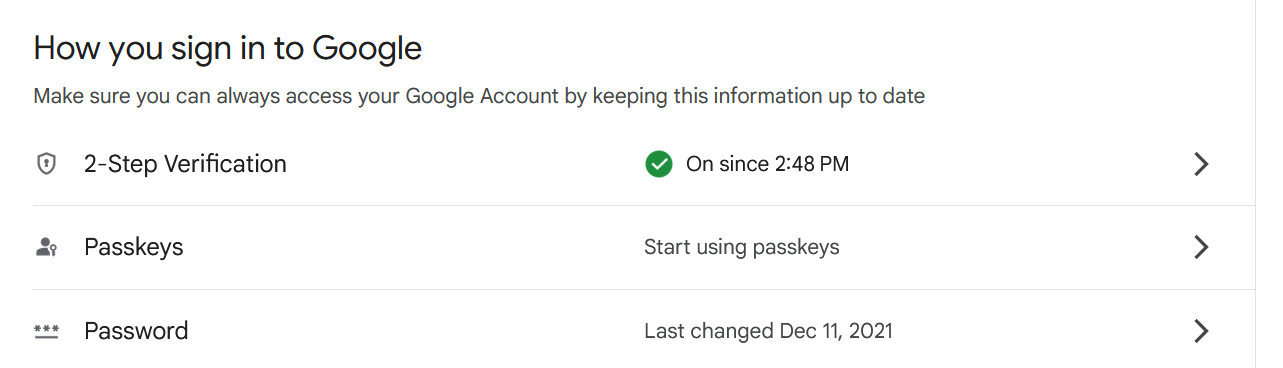
EMAIL : 'salsabillaahmad10@gmail.com',

PASSWORD : 'rmwk uhmr emtc xxff'

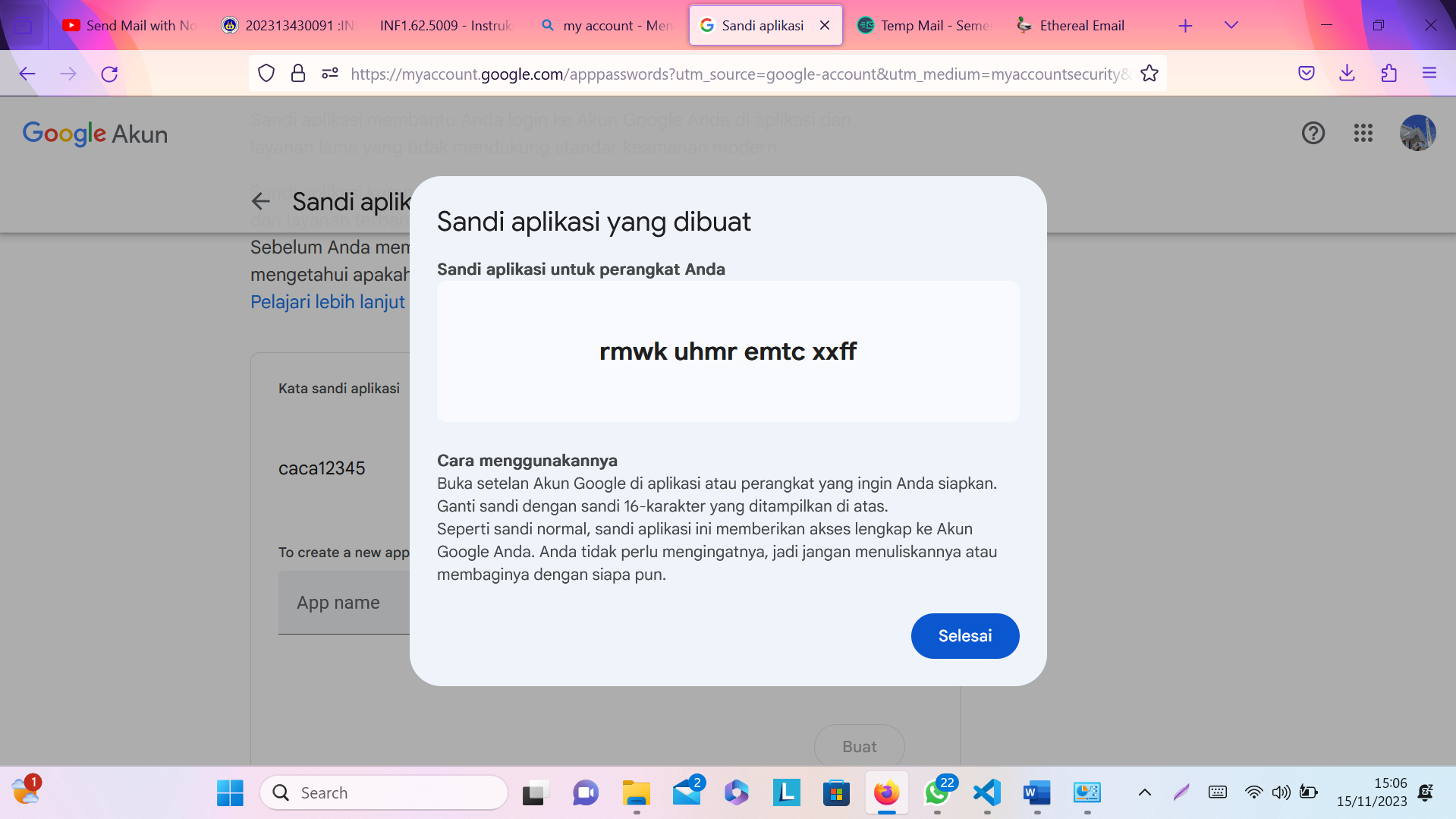
}



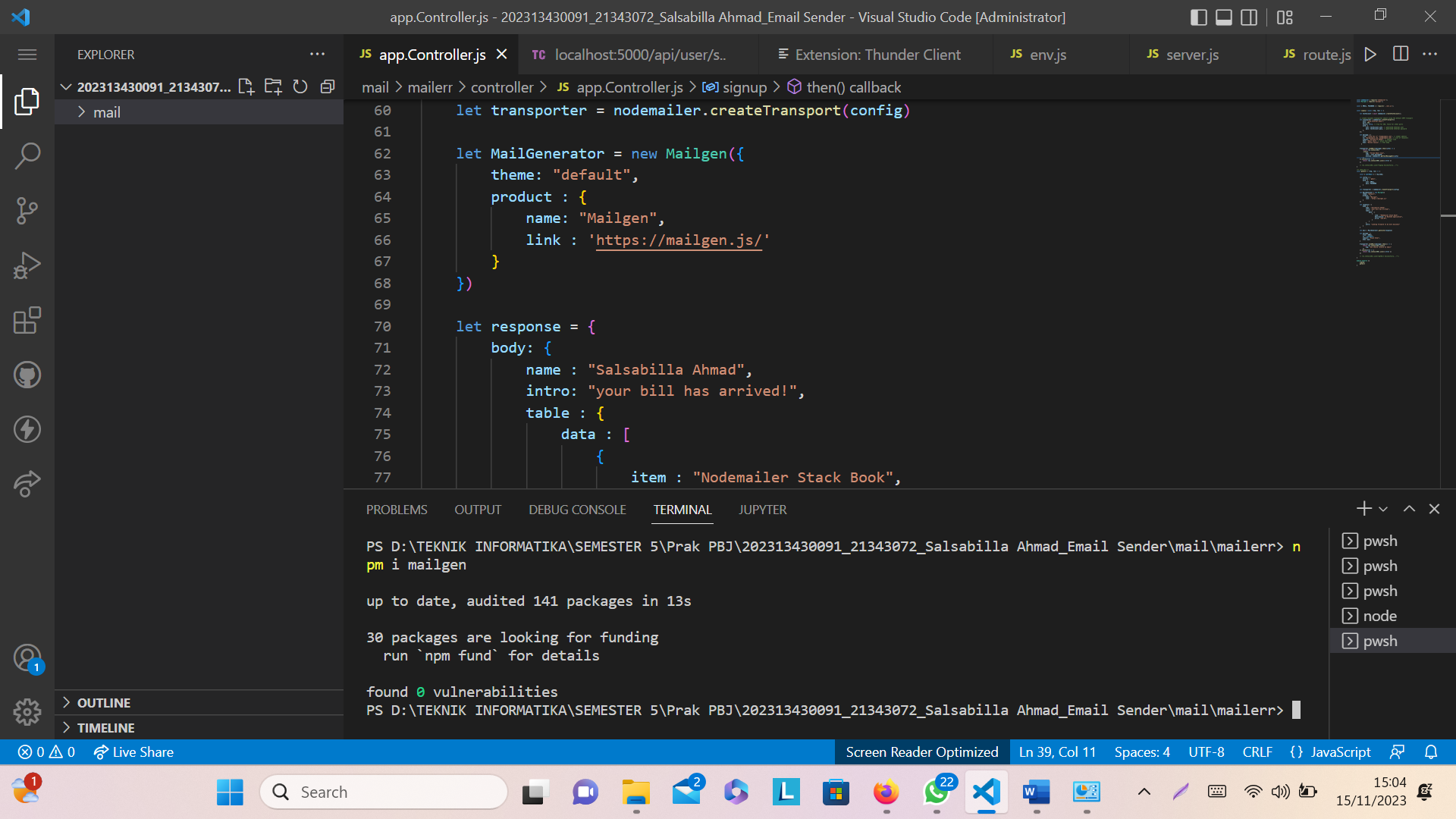
1. Pertama bukak di browser my account > security
2. Lalu bukak . 2-Step Verification



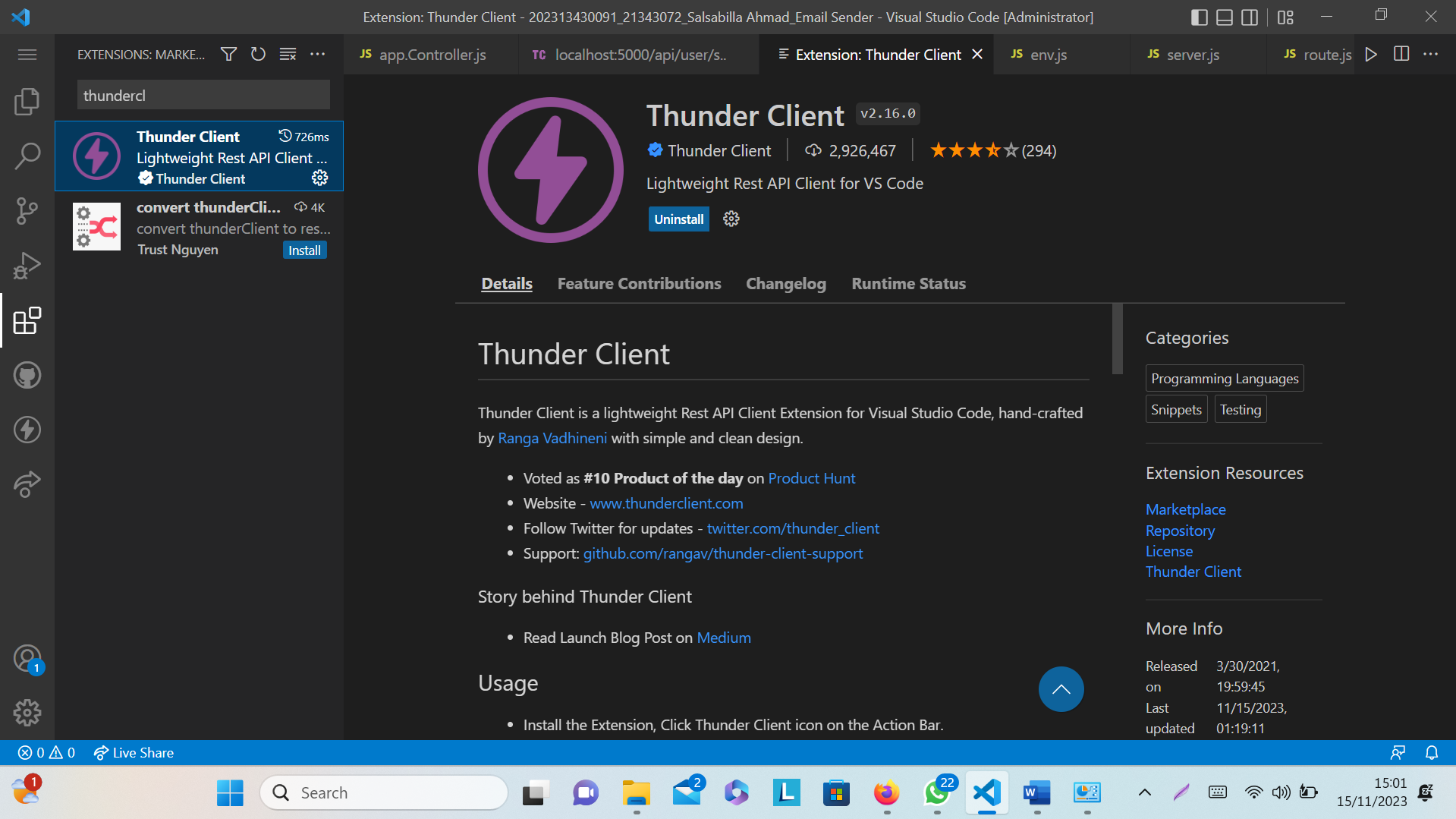
1. Kemudian buatkan katasandi aplikasi



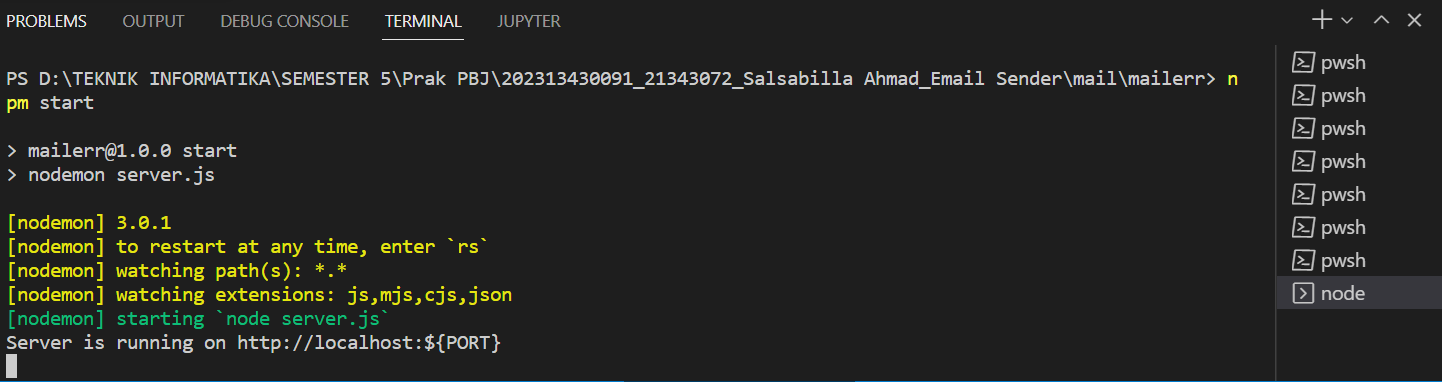
1. Lalu salin sandi aplikasi baru tersebut, lalu paste pada file env.js
2. Install package > **npm I milgen**



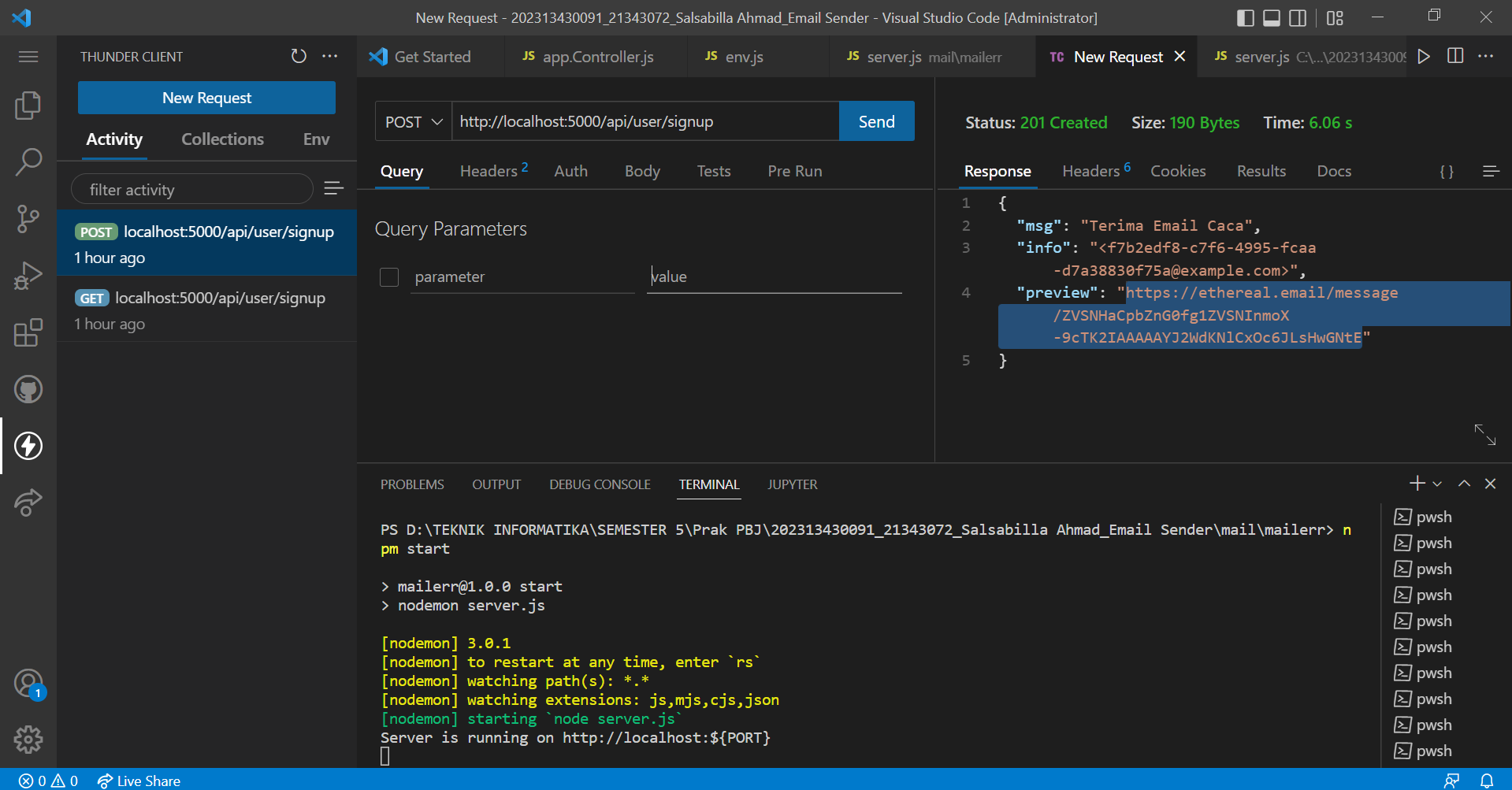
1. Lalu install thunder client pada extension



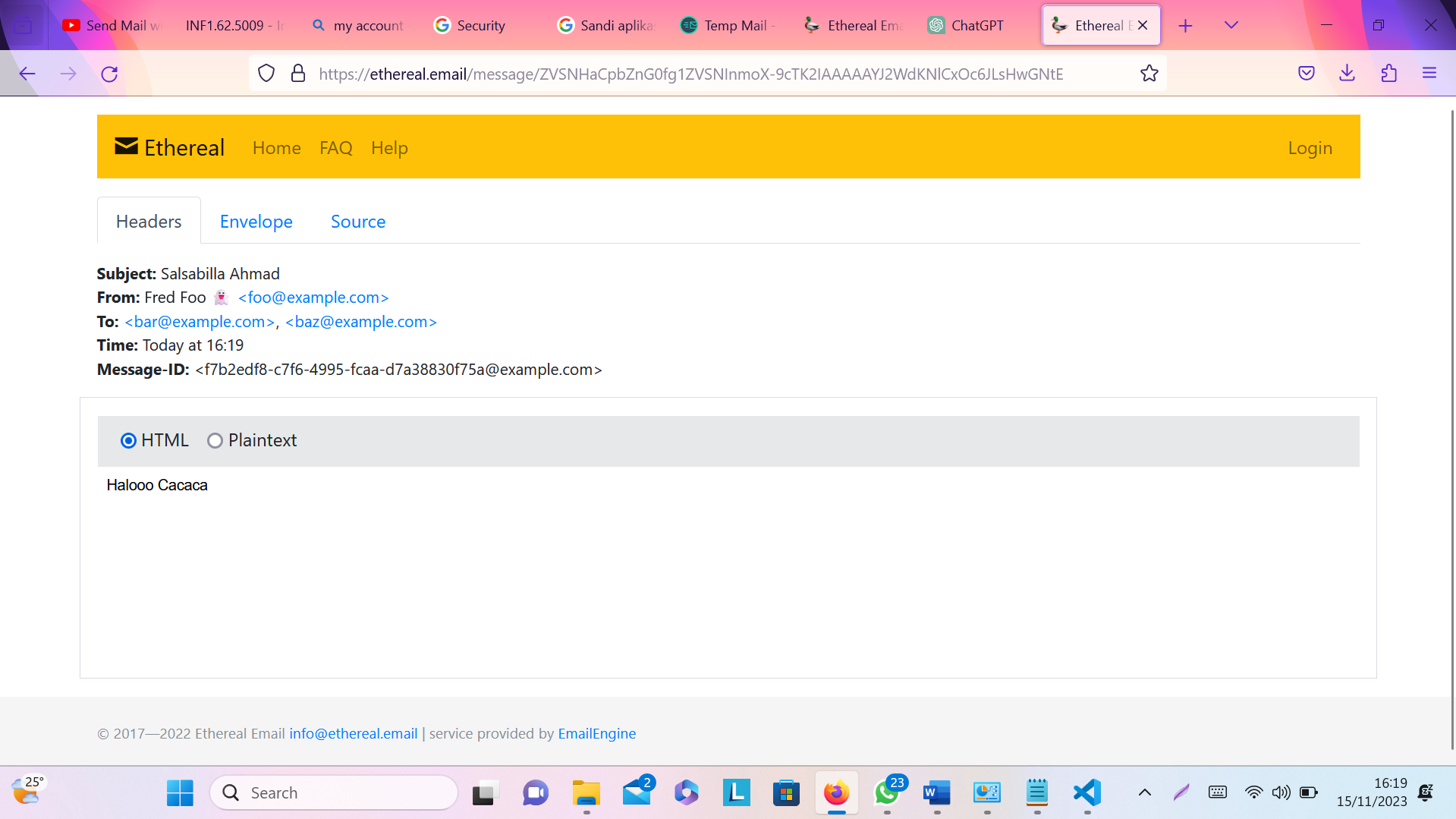
1. Lalu jalankan npm start pada terminal > **npm start**



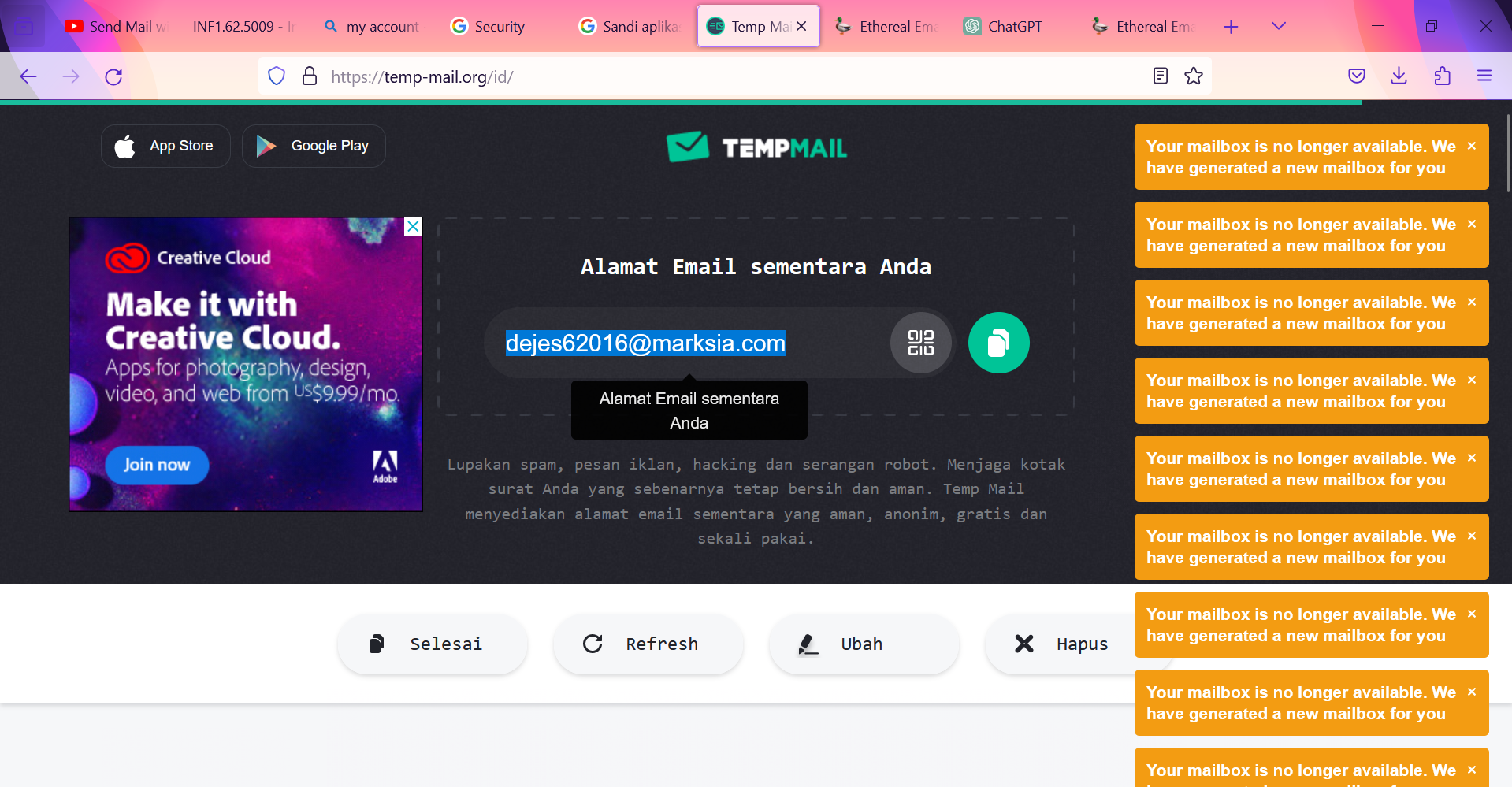
1. Kemudian klik thunder client > new request > ketikkan url > <http://localhost:5000/api/user/signup> > send
2. Hasil pada terminal



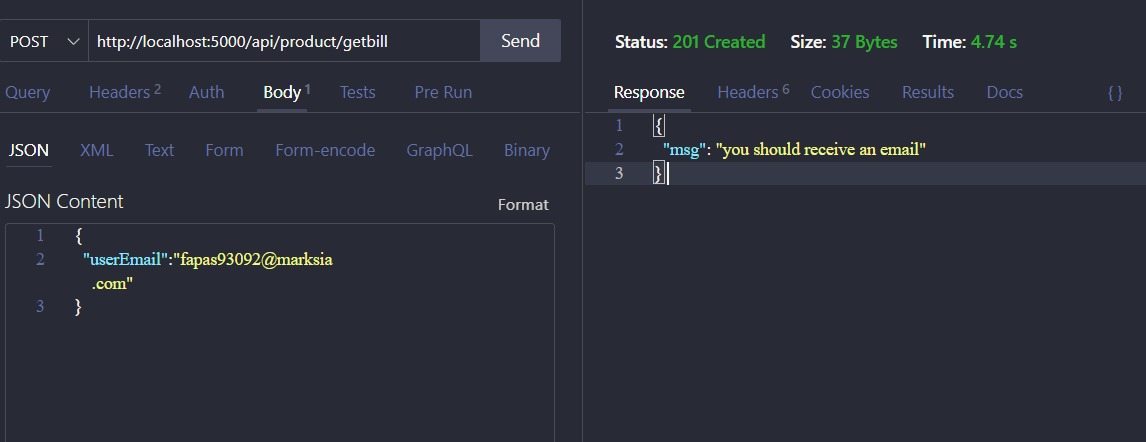
1. Hasil pada Ethereal Email



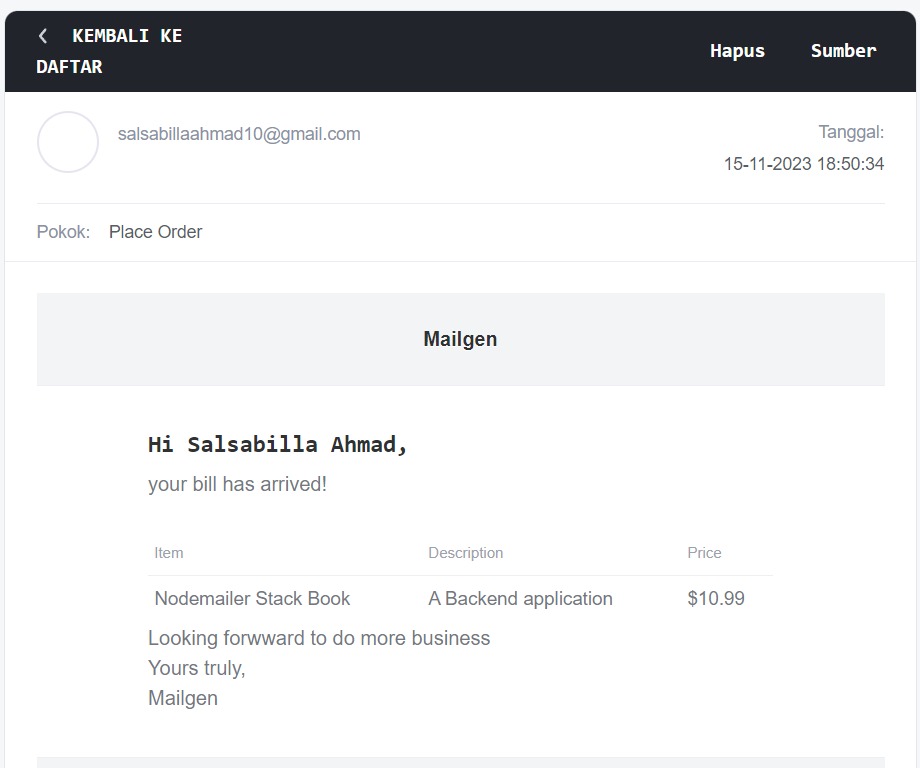
1. Bukak TempMail > Salin Email tersebut



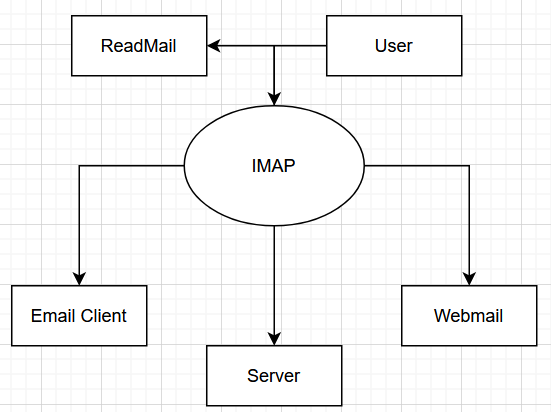
1. Pengetesan POST menggunakan TempMail



1. Hasil Run tampilan di Temp mail



1. **Diagram Terkait Aplikasi**
2. IMAP

****

* Penjelasan

1. User

Memanfaatkan email client atau webmail untuk berinteraksi dengan server email menggunakan protokol IMAP.

1. Email Client

Menyediakan antarmuka pengguna untuk berinteraksi dengan email, memungkinkan pengguna untuk membaca, mengirim, dan mengelola pesan email.

1. WebMail

Memberikan akses online ke akun email pengguna tanpa memerlukan instalasi perangkat lunak tambahan.

1. Server

Menangani permintaan dari email client atau webmail, menyimpan pesan email, mengatur folder, dan memastikan keamanan dan integritas data.

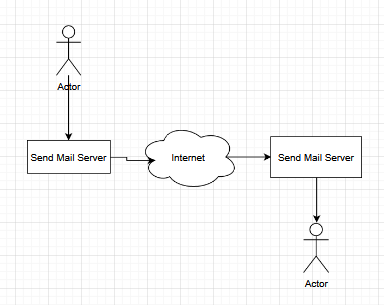
1. Imap

Memungkinkan email client atau webmail untuk berkomunikasi dengan server email, memungkinkan pengguna untuk membaca, mengirim, dan mengelola pesan email tanpa harus mengunduh pesan secara loka

1. ReadMail

Saat pengguna membaca email menggunakan email client atau webmail, permintaan dibuat melalui protokol IMAP untuk mengambil pesan email dari server dan menampilkannya kepada pengguna.

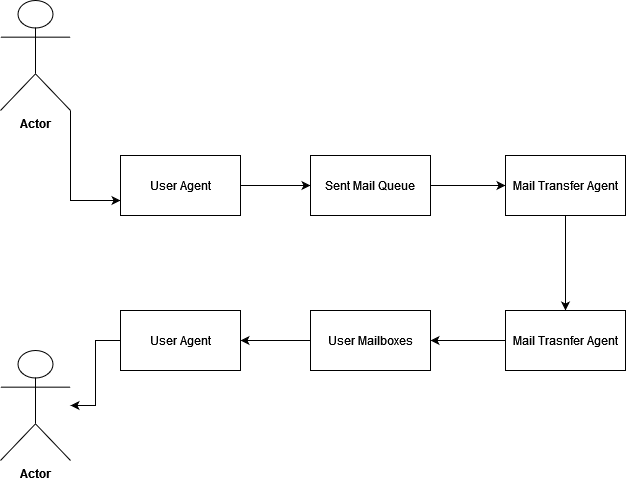
1. POP3



* Penjelasan:

Dalam diagram urutan ini, tergambar dengan jelas operasi pertukaran email menggunakan protokol POP3 (Post Office Protocol). Semua tahap interaksi pesan antara klien POP3 dan server telah rinci dijelaskan, fokus utamanya adalah pada proses pengambilan satu pesan email dari kotak surat POP3. Alur pesan yang diuraikan mencerminkan dengan akurat komunikasi yang terjadi antara klien POP3 dan server, memberikan pemahaman menyeluruh tentang bagaimana proses ini berlangsung

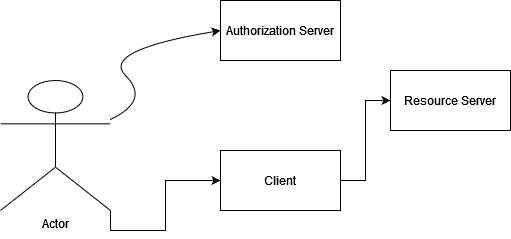
1. SMTP



* Penjelasan:

Cara kerja SMTP melibatkan pembukaan koneksi TCP ke server SMTP dan pengiriman email melalui port 25. Proses SMTP mencakup beberapa perintah, seperti HELO untuk mengidentifikasi klien ke server, MAIL untuk memulai transfer pesan, RCPT untuk mengidentifikasi penerima, dan DATA untuk mengirim data pesan secara bertahap. Selain itu, ada perintah EHLO yang digunakan oleh klien ESMTP untuk membuka transmisi dengan server. SMTP mengadopsi arsitektur client-server dan merupakan protokol email yang paling umum digunakan.

1. Oauth 2.0



* Penjelasan:

Diagram ini secara visual dapat memperlihatkan langkah-langkah di mana aplikasi klien meminta akses ke sumber daya yang dilindungi oleh server otorisasi, cara pengguna memberikan izin untuk mengakses sumber daya tersebut, dan bagaimana aplikasi klien menggunakan token akses yang diberikan untuk mengakses sumber daya yang dilindungi.

**D. Referensi**

<https://youtu.be/lBRnLXwjLw0?si=uV94O8SmLsdxSDMW>

<https://www.bing.com/search?pc=COSP&ptag=D060422-N9997AD0447C6901&form=CONBDF&conlogo=CT3335361&q=my%20account&ntref=1>

<https://myaccount.google.com/apppasswords?utm_source=google-account&utm_medium=myaccountsecurity&utm_campaign=tsv-settings&rapt=AEjHL4OmPrA0okl62UJTjk6AKd5TtA5QlKFoEpB8-BxGz-CygSThZDyoOjrrSz2KwmUNZxRq3VWrtbEzcbTcdfFsv8rd3tNw9Bp7ulUceQRN9XNVgxQzqk0>

<https://ethereal.email/message/ZVSNHaCpbZnG0fg1ZVSNInmoX-9cTK2IAAAAAYJ2WdKNlCxOc6JLsHwGNtE>

<https://temp-mail.org/id/>

<https://app.diagrams.net/>