Kelompok 1

1. Ananda Noviesta Wijaya
2. Jefri Tri Muji Pangestu
3. Azhar Adyatama Pratama
4. Rizka Khairunnisa Faridatuz Zahra
5. Aditia Prasasti

Tugas Kelompok Pembuatan CRUD API

1. Membuat CRUD API untuk model USER
   - File User.Controller.js

```
let express = require('express')
let router = express.Router()
let userService = require('./user.services')

router.post('/', async(req, res) => {
    try {
        let userData = req.body
        let newUser = await userService.createUser(userData)
        res.status(201).json(newUser)
    } catch (error) {
        res.status(400).send(error.message)
    }
})

router.get('/', async(req, res) => {

    try {
        let users = await userService.getAllUsers()
        res.send(users)
    } catch (error) {
        res.status(500).send(error.message)
    }
})

router.get('/:id', async(req, res) => {

    try {
        let userId = parseInt(req.params.id)
        let user = await userService.getUserByUserId(userId)
        res.send(user)
    } catch (error) {
        res.status(400).send(error.message)
    }
})

router.patch('/:id', async(req, res) => {

    try {
        let userId = parseInt(req.params.id)
        let userData = req.body
        let updatedUser = await userService.editUserByUserId(userId, userData)

        delete updatedUser.password
        res.status(200).send({data: updatedUser, message: 'User updated succesfully!'})
    } catch (error) {
        res.status(400).send(error.message)
    }
})

router.delete('/:id', async(req, res) => {

    try {
        let userId = parseInt(req.params.id)
        await userService.deleteUserByUserId(userId)
        res.status(200).json({message: 'User Deleted'})
    } catch (error) {
        res.status(400).send(error.message)
    }
})

module.exports = router
```

- Create

Membuat router dengan method post dan endpoint ‘/’, yang di dalamnya ada try and catch, lalu membuat variabel userData yang memiliki value yang didapat dari req.body setelah itu membuat variabel newUser yang digunakan untuk memanggil function createUser dengan argumen userData dari file user.service.js.

Jika berhasil maka akan mengirim respon status 201 beserta format json dari variabel newUser, jika gagal maka akan mengirim respon status 400 beserta pesan error

- Read

a. Membuat router dengan method get dan endpoint '/', yang di dalamnya ada try and catch, lalu membuat variabel users yang digunakan untuk memanggil function getAllUsers dari file user.service.js. Jika berhasil maka akan memberikan respon berisi data users, jika gagal maka akan mengirim respon status 500 beserta pesan error

b. Membuat router dengan method get dan endpoint '/:id', yang di dalamnya ada try and catch, lalu membuat variabel userId untuk menampung nilai id dari req.params.id yang telah dijadikan integer. Lalu membuat variabel user yang digunakan untuk memanggil function getUserById dari file user.service.js dengan argumen userId. Jika berhasil maka akan memberikan respon berisi data user, jika gagal maka akan mengirim respon status 400 beserta pesan error

- Update

Membuat router dengan method patch dan endpoint '/:id', yang di dalamnya ada try and catch, lalu membuat variabel userId untuk menampung nilai id dari req.params.id yang telah dijadikan integer dan variabel userData yang menampung nilai dari req.body, setelah itu membuat variabel updatedData yang digunakan untuk memanggil function editUserById dari file user.service.js dengan argumen userId dan userData. Lalu password yang dikembalikan dari variabel updatedUser akan disembuyikan saat diberikan menjadi respon. Jika berhasil maka akan memberikan respon status 200 dan data updatedUser serta pesan 'user deleted' dengan format json, jika gagal maka akan mengirim respon status 400 beserta pesan error

- Delete

Membuat router dengan method delete dan endpoint '/:id', yang di dalamnya ada try and catch, lalu membuat variabel userId untuk menampung nilai id dari req.params.id yang telah dijadikan integer, lalu memanggil function deleteUserById dari file user.service.js dengan argumen userId. Jika berhasil maka akan memberikan respon status 200 dan pesan 'User Deleted' dengan format json, jika gagal maka akan mengirim respon status 400 beserta pesan error

- File user. Repository

```
let prisma = require('../db')

async function insertUser(userData) {

    let newUser = await prisma.user.create({
        data: {
            username: userData.username,
            email: userData.email,
            password: userData.password
        }
    })
    return newUser
}

async function findUsers() {
   let users = await prisma.user.findMany({
        select: {
            user_id: true,
            username: true,
            email: true,
            role: true,
            created_at: true
        }
   })
   return users
}

async function findUserByUserId(user_id) {
    let user = await prisma.user.findUnique({
        where: {
            user_id: parseInt(user_id)
        },
        select: {
            user_id: true,
            username: true,
            email: true,
            role: true,
            created_at: true
        }
    })
    return user
}


async function editUser(user_id, userData) {
    let updatedUser = await prisma.user.update({
        where: {
            user_id: parseInt(user_id)
        },
        data: {
            username: userData.username,
            email: userData.email,
            password: userData.password,
            role: userData.role,
        }
    })
    return updatedUser
}

async function deleteUser(user_id) {
    await prisma.user.delete({
        where: {
            user_id: parseInt(user_id)
        }
    })
}


module.exports = {insertUser, findUsers, findUserByUserId, editUser, deleteUser}
```

Terdapat function yang terdiri dari 4 operasi Create, Read, Update, Delete.

- Create

Membuat function insertUser yang memiliki parameter userData, lalu membuat variabel newUser yang digunakan untuk membuat data user baru yang kolomnya diisi berdasarkan parameter userData, seperti username, email, password, dan role

- Read

a. Membuat function findUsers, lalu membuat variabel users yang digunakan untuk memanggil seluruh data di tabel user berupa kolom id, username, email, password, role, dan createdAt

b. Membuat function findUserById yang memiliki parameter id, lalu membuat variabel user yang digunakan untuk memanggil data user berdasarkan id yang didapat dari parameter id dan memanggil kolom berupa kolom id, username, email, password, role, dan createdAt

- Update

Membuat function editUser yang memiliki parameter id dan userData, lalu membuat variabel updatedUser yang digunakan untuk memanggil data user berdasarkan id yang didapat dari parameter id dan mengupdate nya dengan data baru yang didapatkan dari parameter userData, seperti kolom id, username, email, password, dan role

- Delete

Membuat function deleteUser yang memiliki parameter id, lalu memanggil data user berdasarkan id yang didapat dari parameter id lalu menghapus datanya

- Fitur user service

```
let bcrypt = require('bcrypt')
let { insertUser, findUsers, findUserByUserId, editUser, deleteUser } = require('./user.repository')

async function createUser(newUserData) {

    let hashedPassword = await bcrypt.hash(newUserData.password, 10)

    newUserData.password = hashedPassword
    let newUser = await insertUser(newUserData)
    return newUser
}

async function getAllUsers() {
    let users = await findUsers()
    return users
}

async function getUserByUserId(user_id) {

    let user = await findUserByUserId(user_id)
    if(!user){
        throw Error('User not found')
    }
    return user
}

async function editUserByUserId(user_id, userData) {
    if(userData.password){
        let hashedPassword = await bcrypt.hash(userData.password, 10)
        userData.password = hashedPassword
    }
    await getUserByUserId(user_id)
    let updatedUser = await editUser(user_id, userData)
    return updatedUser
}

async function deleteUserByUserId(user_id) {
    await getUserByUserId(user_id)
    await deleteUser(user_id)
}

module.exports = { createUser, getAllUsers, getUserByUserId, editUserByUserId, deleteUserByUserId }
```

- Create

Membuat function createUser yang memiliki parameter newUserData, lalu membuat variabel hashed password untuk melakukan hashing pada password dalam parameter newUserData, lalu isi dari password dalam newUserData akan diganti dengan password yang telah di hashed. Setelah itu membuat variabel newUser yang akan memanggil function insertUser dari file user.repository.js dengan argumen newUser data

- Read

a. Membuat function getAllUsers, lalu membuat variabel users yang akan memanggil function findUsers yang ada di file user.repository.js

b. Membuat function getUserById yang memiliki parameter id, lalu membuat variabel user yang digunakan untuk memanggil function findUserById dari file user.repository.js dengan argumen id

- Update

Membuat function editUserById yang memiliki parameter id dan userData, lalu membuat pengkondisian jika password dalam parameter userData juga diganti maka akan dilakukan hashing terlebih dahulu dan digantikan dengan password yang telah dihashing. Lalu memanggil function getUserById dengan argumen id, dan membuat variabel updatedUser yang digunakan untuk memanggil function editUser dari file user.repository.js dengan argumen id dan userData

- Delete

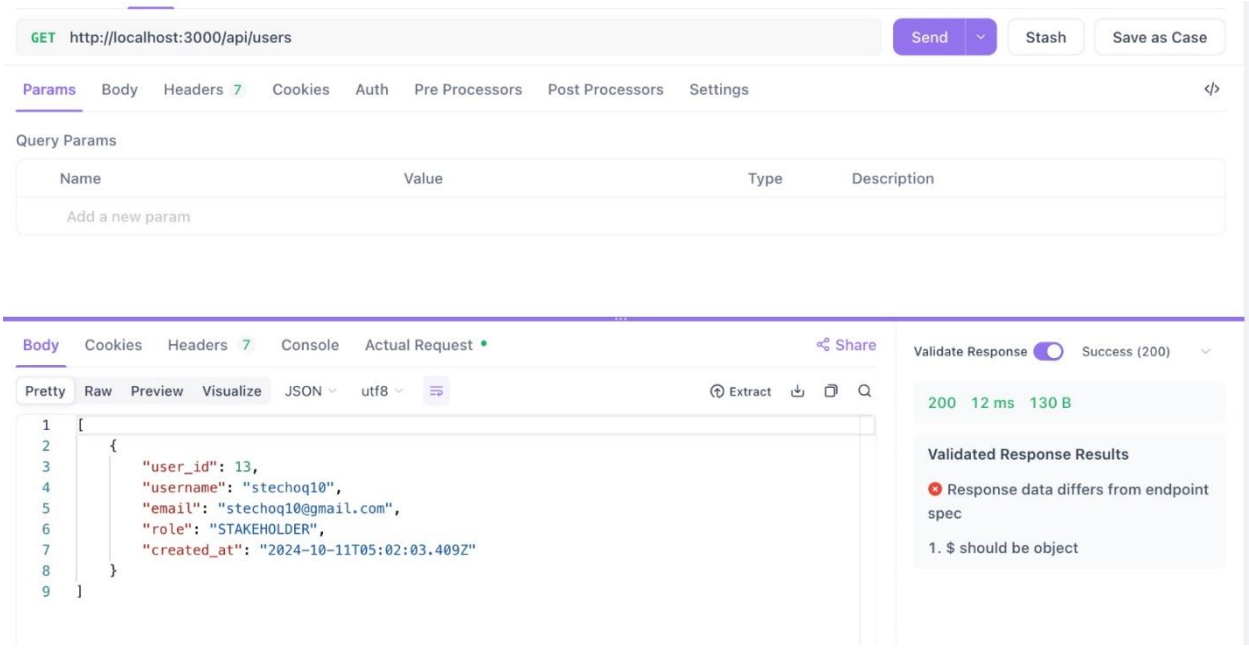
Membuat function deleteUserId yang memiliki parameter id, lalu memanggil function getUserById dan getUserById dengan argumen id

File ini bertugas untuk mendefinisikan endpoint (rute) yang menangani permintaan HTTP terkait entitas User. File ini menangani metode POST, GET, PATCH, dan DELETE untuk operasi yang berbeda.

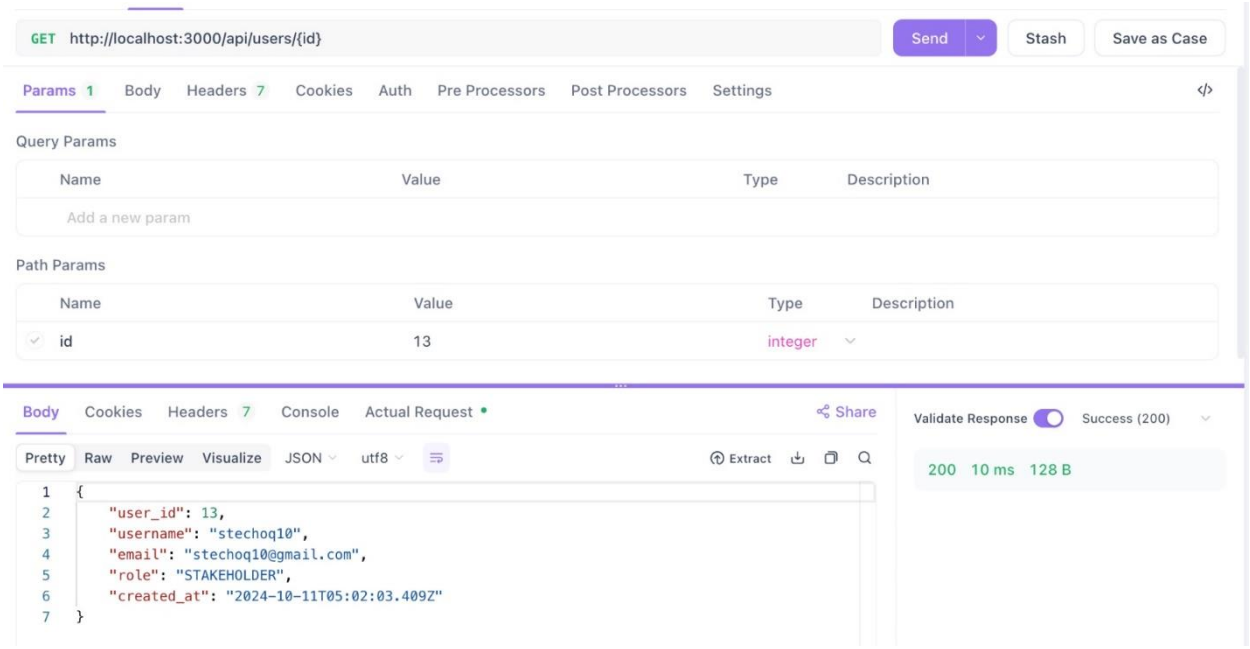
Fungsi Utama:

- POST /: Membuat pengguna baru.
- GET /: Mengambil semua pengguna.
- GET /:id: Mengambil detail pengguna berdasarkan ID.
- PATCH /:id: Memperbarui data pengguna berdasarkan ID.
- DELETE /:id: Menghapus pengguna berdasarkan ID.

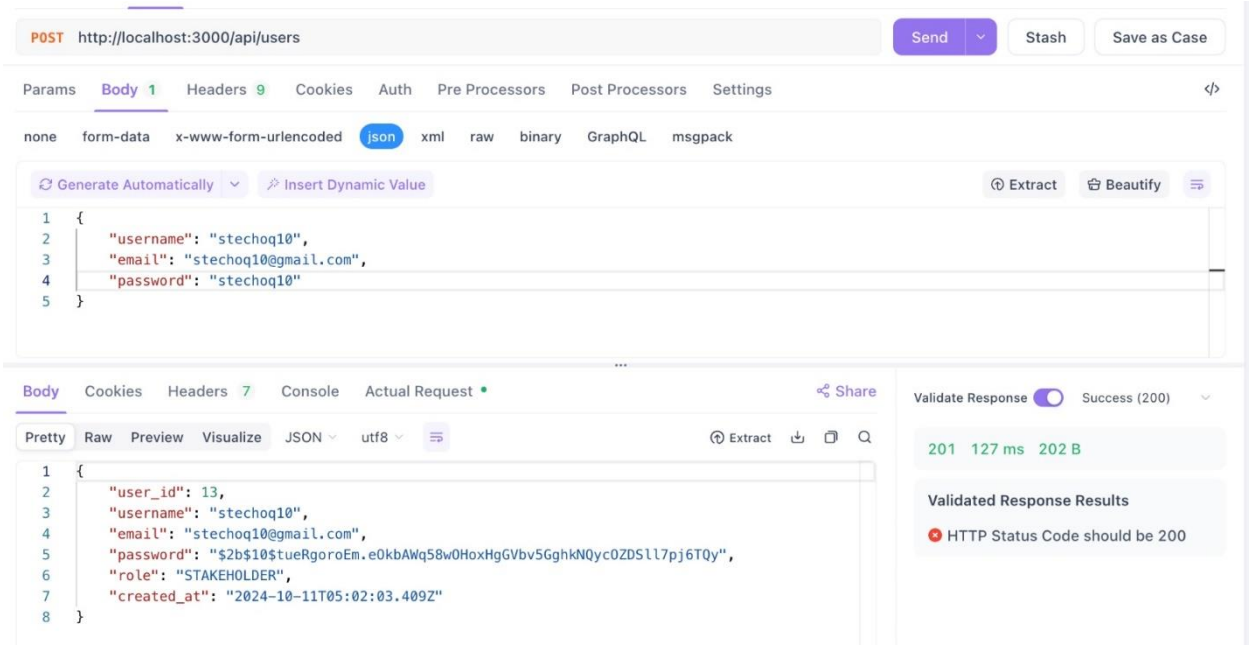
A. Get User
Deskripsi: Endpoint ini digunakan untuk mengambil semua data pengguna yang ada di dalam sistem.



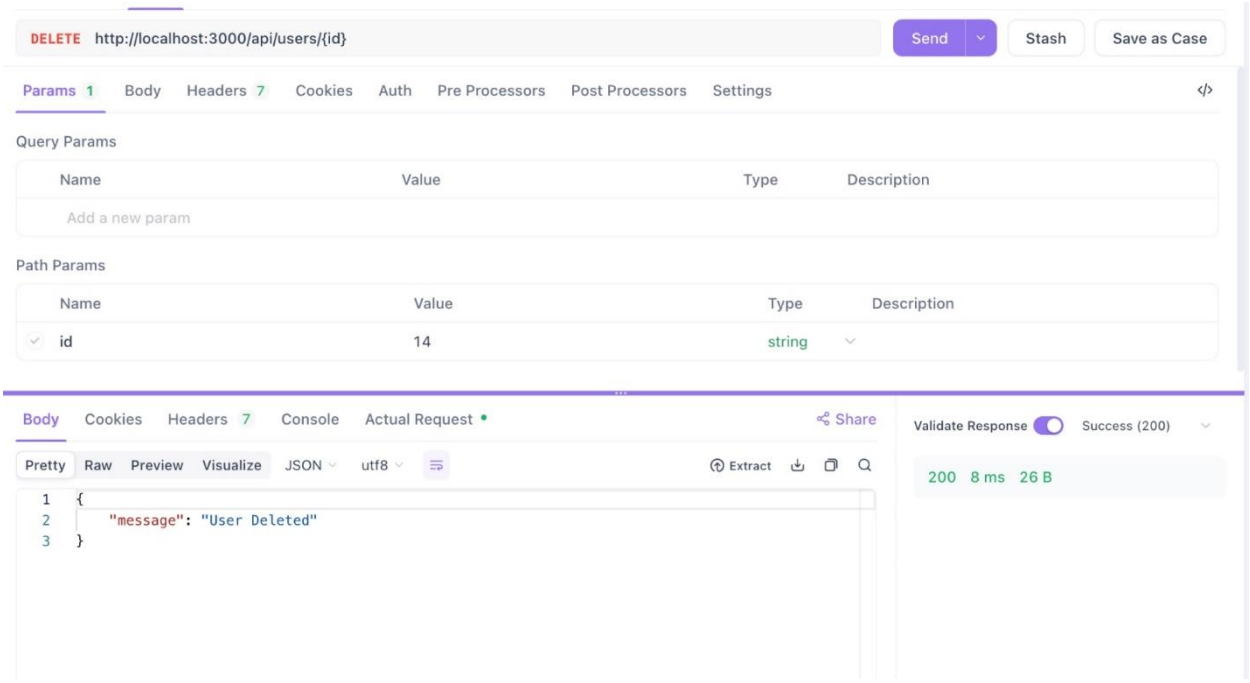
B. Get User By Id
Deskripsi: Endpoint ini digunakan untuk mengambil data pengguna berdasarkan ID pengguna tertentu.



C. Post User
Deskripsi: Endpoint ini digunakan untuk menambahkan pengguna baru ke dalam sistem.



D. Delete User
Deskripsi: Endpoint ini digunakan untuk menghapus pengguna dari sistem berdasarkan ID pengguna.



2. Membuat CRUD API untuk model Product

   Produk.Repository.js

   -Inisialisasi Prisma: Mengimpor koneksi database Prisma dari file db.js.
   -Fungsi insertProduct(productData): Menyimpan produk baru dan jumlahnya ke database, lalu mengembalikan produk yang baru dibuat.

-Fungsi findProducts(): Mengambil semua produk dan jumlahnya dari database dan mengembalikannya.
-Fungsi findProductById(product_id): Mencari produk berdasarkan ID dan mengembalikan informasi produk serta jumlahnya.
-Fungsi editProduct(product_id, productData): Memperbarui informasi produk dan jumlahnya berdasarkan ID dan mengembalikan produk yang diperbarui.
-Fungsi deleteProduct(product_id): Menghapus jumlah produk dan produk itu sendiri dari database berdasarkan ID.
-Fungsi findQuantityById(product_id): Mencari jumlah produk berdasarkan ID dan mengembalikannya.
-Fungsi updateProductQuantity(product_id, newQuantity): Memperbarui jumlah produk berdasarkan ID dengan nilai baru.
-Ekspor Modul: Mengekspor semua fungsi agar bisa digunakan di file lain.

Kode ini mengelola operasi dasar (CRUD) untuk produk dan jumlahnya di database.

```
let prisma = require('../db')

async function insertProduct(productData) {

    let newProduct = await prisma.master_Data.create({
        data: {
            product_name: productData.product_name,
            price: productData.price,
            user_id: productData.user_id
        }
    })
    let newQuantity = await prisma.quantity.create({
        data: {
            quantity_of_product: productData.quantity,
            product_id: parseInt(newProduct.product_id)
        }
    })
    return newProduct
}

async function findProducts() {

    let products = await prisma.master_Data.findMany({
        include: {
            Quantity: {
                select: {
                    quantity_of_product: true
                }
            }
        }
    })
    return products
}

async function findProductById(product_id) {

    let products = await prisma.master_Data.findUnique({
        where: {
            product_id: parseInt(product_id)
        },
        include: {
            Quantity: {
                select: {
                    quantity_of_product: true
                }
            }
        }
    })
    return products
}

async function editProduct(product_id, productData) {

    let updatedProduct = await prisma.master_Data.update({
        where: {
            product_id: parseInt(product_id)
        },
        data: {
            product_name: productData.product_name,
            price: productData.price,
            user_id: productData.user_id
        }
    })

    let updateQuantity = await prisma.quantity.update({
        where: {
            product_id: parseInt(product_id)
        },
        data: {
            quantity_of_product: productData.quantity
        }
    })
    return updatedProduct
}

async function deleteProduct(product_id) {

    await prisma.quantity.delete({
        where: {
            product_id: parseInt(product_id)
        }
    })
    await prisma.master_Data.delete({
        where: {
            product_id: parseInt(product_id)
        }
    })
}

async function findQuantityById(product_id) {
    let quantity =  await prisma.quantity.findUnique({
        where: {
            product_id: parseInt(product_id)
        }
    })
    return quantity
}

async function updateProductQuantity(product_id, newQuantity) {
    await prisma.quantity.update({
        where: {
            product_id: parseInt(product_id)
        },
        data: {
            quantity_of_product: parseInt(newQuantity)
        }
    })
}


module.exports = {insertProduct, findProducts, findProductById, editProduct, deleteProduct, updateProductQuantity, findQuantityById}
```

Product.Service.js

-Mengimpor Fungsi: Mengimpor fungsi dari product.repository untuk digunakan dalam pengelolaan produk.

-Fungsi createProduct(newProductData): Menggunakan insertProduct untuk menambahkan produk baru ke database dan mengembalikan produk yang baru dibuat.

-Fungsi getAllProducts(): Mengambil semua produk dari database dengan memanggil findProducts dan mengembalikannya.

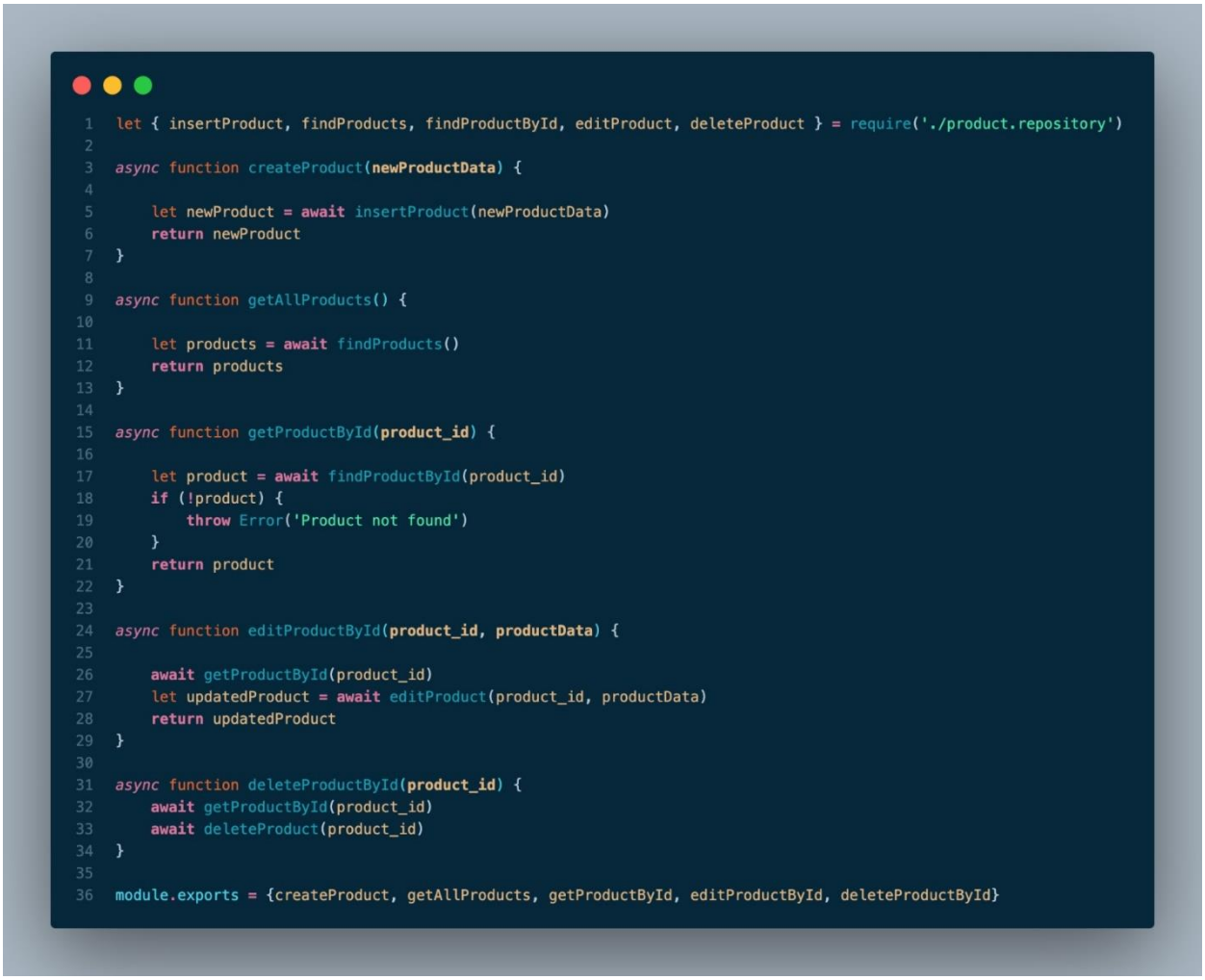
-Fungsi getProductById(product_id): Mencari produk berdasarkan ID dan melemparkan error jika produk tidak ditemukan, lalu mengembalikan produk yang ditemukan.

-Fungsi editProductById(product_id, productData): Memastikan produk ada dengan memanggil getProductById, kemudian memperbarui produk menggunakan editProduct dan mengembalikan produk yang diperbarui.

-Fungsi deleteProductById(product_id): Memastikan produk ada dengan memanggil getProductById, lalu menghapus produk tersebut dengan memanggil deleteProduct.

-Ekspor Modul: Mengekspor semua fungsi agar bisa digunakan di file lain.

Kode ini mengelola operasi dasar (CRUD) untuk produk dengan menggunakan fungsi yang diimpor dari repository.



```
let { insertProduct, findProducts, findProductById, editProduct, deleteProduct } = require('./product.repository')

async function createProduct(newProductData) {

    let newProduct = await insertProduct(newProductData)
    return newProduct
}

async function getAllProducts() {

    let products = await findProducts()
    return products
}

async function getProductById(product_id) {

    let product = await findProductById(product_id)
    if (!product) {
        throw Error('Product not found')
    }
    return product
}

async function editProductById(product_id, productData) {

    await getProductById(product_id)
    let updatedProduct = await editProduct(product_id, productData)
    return updatedProduct
}

async function deleteProductById(product_id) {
    await getProductById(product_id)
    await deleteProduct(product_id)
}

module.exports = {createProduct, getAllProducts, getProductById, editProductById, deleteProductById}
```

Product.controller.js

-Mengimpor Modul: Mengimpor express dan fungsi dari product.services untuk routing.

-Inisialisasi Router: Membuat instance router dengan Express.

-Endpoint POST /: Menerima data produk baru dan menggunakan createProduct untuk menambahkannya, mengembalikan produk baru dengan status 201, atau mengembalikan error dengan status 400 jika gagal.

-Endpoint GET /: Mengambil semua produk dengan getAllProducts, mengembalikan daftar produk dengan status 200, atau mengembalikan error dengan status 500 jika gagal.

-Endpoint GET /:id: Mencari produk berdasarkan ID dari parameter URL dengan getProductById, mengembalikan produk dengan status 200, atau mengembalikan error dengan status 400 jika tidak ditemukan.

-Endpoint PUT /:id: Memperbarui produk berdasarkan ID dan data yang diberikan dengan editProductById, mengembalikan produk yang diperbarui, atau mengembalikan error dengan status 400 jika gagal.

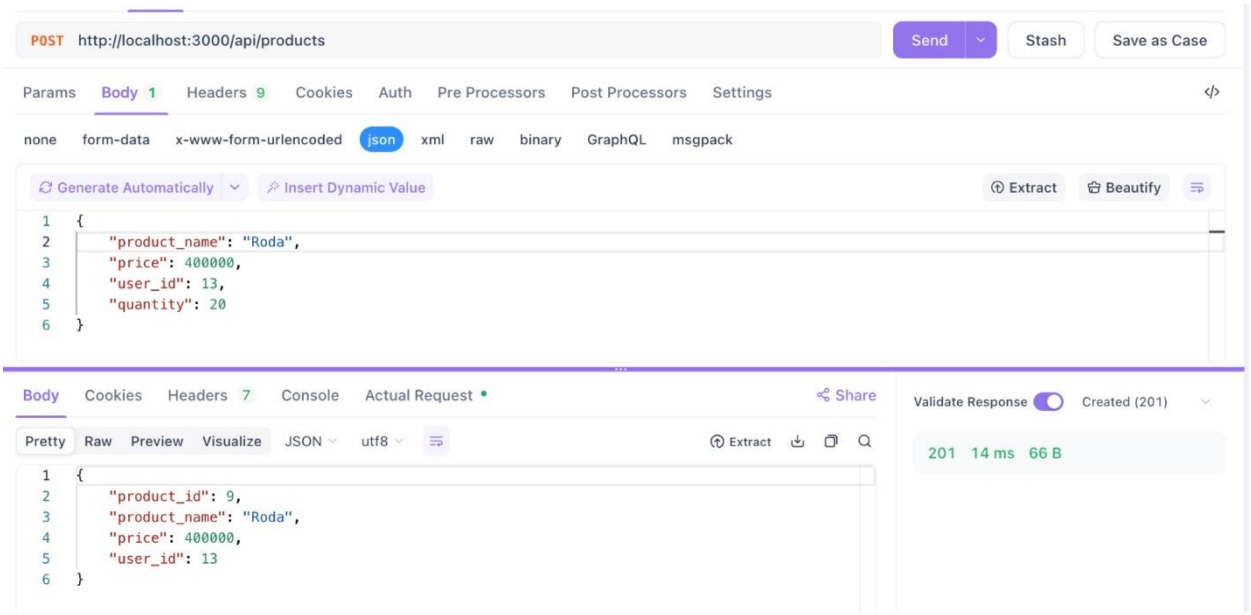
-Endpoint DELETE /:id: Menghapus produk berdasarkan ID dengan deleteProductById, mengembalikan pesan bahwa produk telah dihapus dengan status 200, atau mengembalikan error dengan status 400 jika gagal.

-Ekspor Router: Mengekspor router agar bisa digunakan di file lain.

Kode ini mengatur rute untuk operasi CRUD pada produk menggunakan Express.

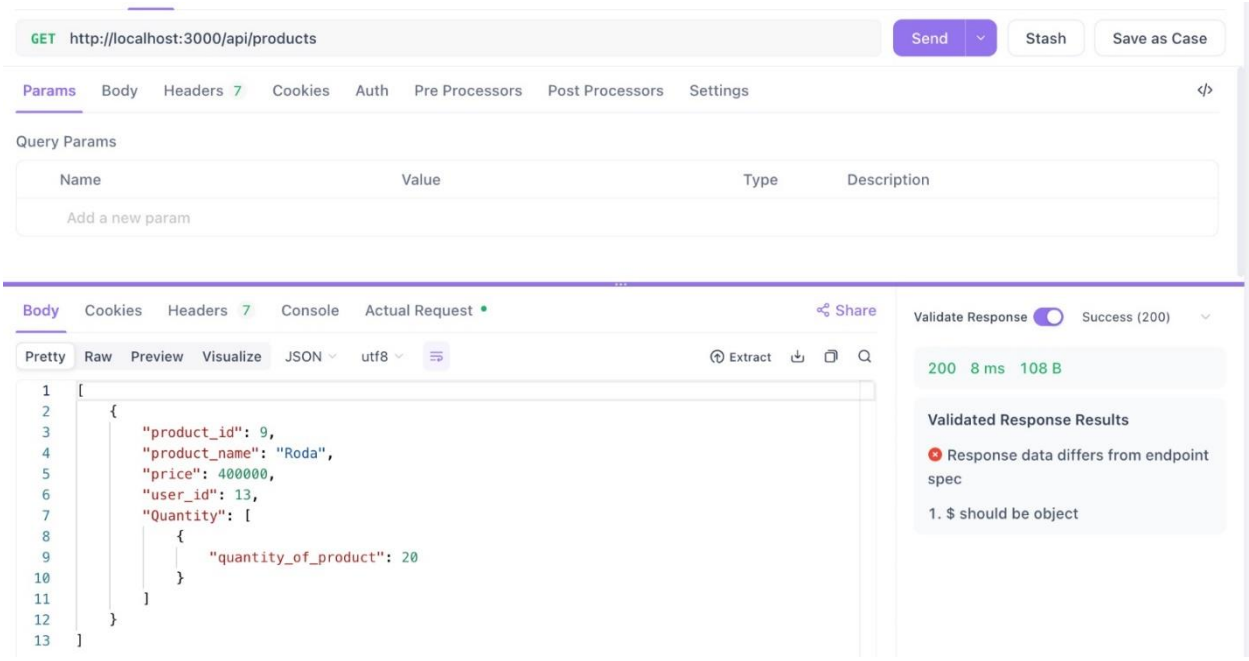
Create Product

Deskripsi: Endpoint ini digunakan untuk menambahkan produk baru ke dalam sistem.

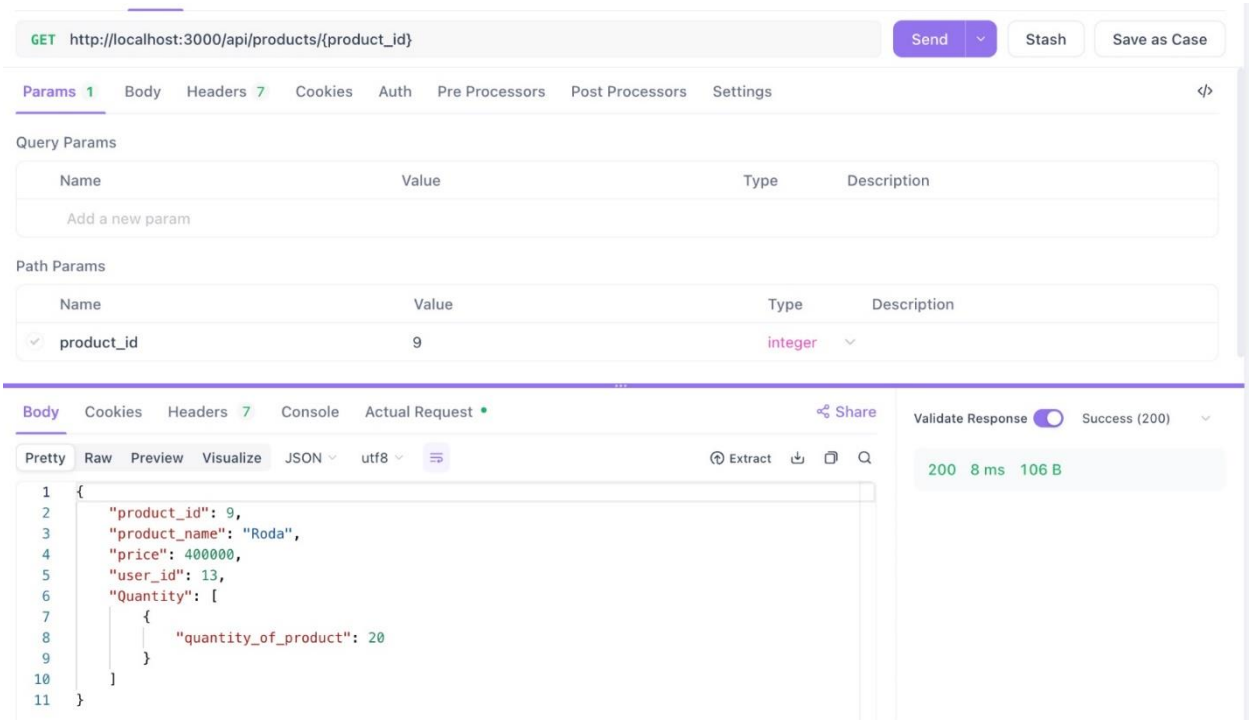


Get Product

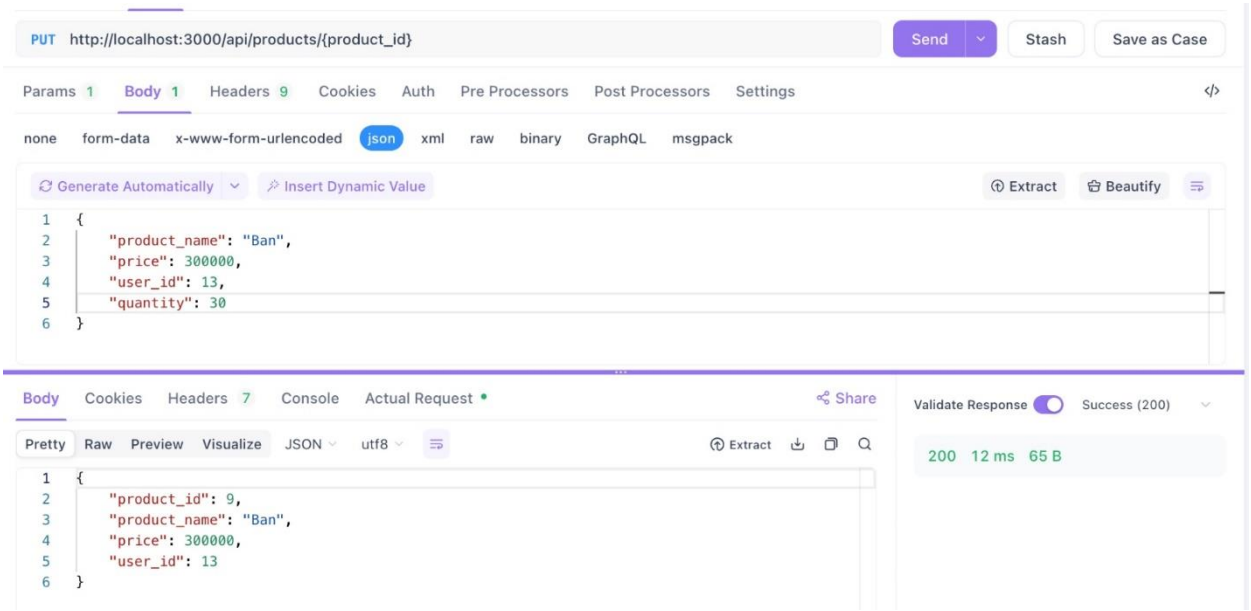
Deskripsi: Endpoint ini digunakan untuk mengambil semua data produk yang ada di dalam sistem.



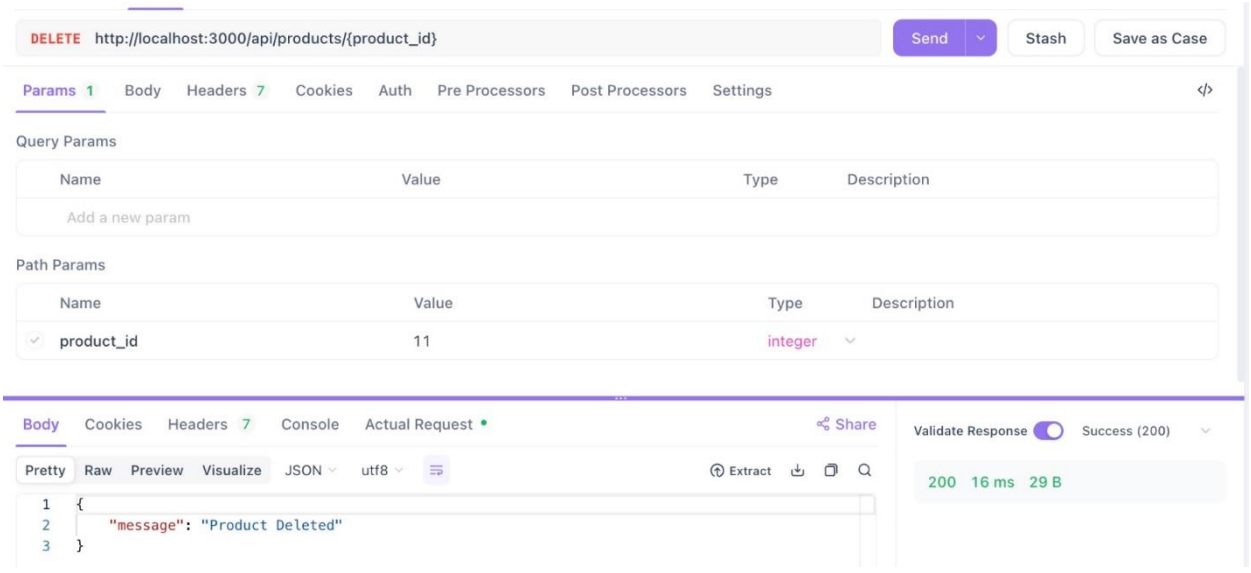
Get Product By Id
Deskripsi: Endpoint ini digunakan untuk mengambil data produk yang ada di dalam sistem berdasarkan ID



Update Product
Deskripsi: Endpoint ini digunakan untuk mengupdate data produk yang ada di dalam sistem berdasarkan value yang dituju



Delete Product
Deskripsi: Endpoint ini digunakan untuk menghapus data produk yang ada di dalam sistem berdasarkan value yang dituju



3. Membuat CRUP API untuk model Order

Order.repository.js

-Mengimpor Prisma: Mengimpor koneksi database Prisma dari file db.js.
-Fungsi createOrder(product_id, user_id, quantity):
    *Membuat pesanan baru dengan status "PENDING" menggunakan data produk, pengguna, dan jumlah.
    *Mengembalikan pesanan yang baru dibuat atau melempar error jika gagal.
-Fungsi findOrders():
    *Mengambil semua pesanan dari database dan menyertakan nama produk dari tabel terkait.
    *Mengembalikan daftar pesanan atau melempar error jika gagal.
-Fungsi findOrdersByUserId(user_id):
    *Mengambil pesanan berdasarkan ID pengguna dan menyertakan nama produk dari tabel terkait.
    *Mengembalikan daftar pesanan atau melempar error jika gagal.
-Fungsi findOrderById(order_id):
    *Mencari pesanan berdasarkan ID dan mengembalikannya.
-Fungsi updateOrderStatus(order_id, status, timeStampField):
    *Memperbarui status pesanan berdasarkan ID dan menambahkan cap waktu jika parameter tersebut diberikan.
    *Mengubah status pesanan dan melempar error jika gagal.

Ekspor Modul: Mengekspor semua fungsi agar dapat digunakan di file lain.

```
let prisma = require('../db')

async function createOrder(product_id, user_id, quantity) {

    try {
        let newOrder = await prisma.order.create({
            data: {
                product_id,
                user_id,
                quantity,
                status: "PENDING"
            }
        })
        return newOrder
    } catch (error) {
        throw new Error('Failed to create order')
    }
}

async function findOrders() {

    try {
        let orders = await prisma.order.findMany({
            include: {
                Master_Data: {
                    select: {
                        product_name: true
                    }
                }
            }
        })
        return orders
    } catch (error) {
        throw new Error('Failed to fetch order')
    }
}


async function findOrdersByUserId(user_id) {

    try {
        let orders = await prisma.order.findMany({
            where: {
                user_id: parseInt(user_id)
            },
            include: {
                Master_Data: {
                    select: {
                        product_name: true
                    }
                }
            }
        })
        return orders
    } catch (error) {
        throw new Error('Failed to fetch order by User Id')
    }
}

async function findOrderById(order_id) {

    let order = await prisma.order.findUnique({
        where: {
            order_id: parseInt(order_id)
        }
    })
    return order
}

async function updateOrderStatus(order_id, status, timeStampField) {

    try {
        let updateData = {status}

        if(timeStampField){
            updateData[timeStampField] = new Date()
        }

        await prisma.order.update({
            where: {
                order_id: parseInt(order_id)
            },
            data: updateData
        })
    } catch (error) {
        throw new Error('Failed to update transaction status')
    }
}

module.exports = {createOrder, findOrders, findOrdersByUserId, findOrderById, updateOrderStatus}
```

Order.service.js

-Mengimpor Repository: Mengimpor orderRepository untuk mengelola pesanan dan productRepository untuk mengelola produk.

-Fungsi createOrder(user_id, product_id, quantity):

*Menggunakan orderRepository untuk membuat pesanan baru dengan ID pengguna, ID produk, dan jumlah yang diberikan.

*Mengembalikan pesanan yang baru dibuat.

-Fungsi getAllOrders():

*Mengambil semua pesanan dari orderRepository dan mengembalikannya.

-Fungsi getOrdersByUserId(user_id):

*Mengambil pesanan berdasarkan ID pengguna dari orderRepository dan mengembalikannya.

-Fungsi getOrdersById(order_id):

*Mencari pesanan berdasarkan ID dari orderRepository dan mengembalikannya.

-Fungsi verifyOrder(order_id, status):

*Mencari pesanan berdasarkan ID, melempar error jika tidak ditemukan, lalu memperbarui status pesanan.

*Jika status baru adalah "ON_PROCESS", memeriksa ketersediaan produk dan memperbarui jumlah produk jika cukup tersedia.

*Mengembalikan error jika produk tidak ditemukan atau jumlah tidak mencukupi.

-Fungsi rejectOrder(order_id):

*Mencari pesanan berdasarkan ID, melempar error jika tidak ditemukan atau jika status bukan "ON_PROCESS".

*Memperbarui status pesanan menjadi "REJECT" dan mengembalikan jumlah produk yang dipesan ke dalam stok.

Ekspor Modul: Mengekspor semua fungsi agar dapat digunakan di file lain.

```
// let { item } = require('../db')
let orderRepository = require('./order.repository')
let productRepository = require('../product/product.repository')

async function createOrder(user_id, product_id, quantity) {
    let newOrder = await orderRepository.createOrder(user_id, product_id, quantity)
    return newOrder
}

async function getAllOrders() {
    let orders = await orderRepository.findOrders()
    return orders
}

async function getOrdersByUserId(user_id) {
    let orders = await orderRepository.findOrdersByUserId(user_id)
    return orders
}

async function getOrdersById(order_id) {
    let order = await orderRepository.findOrderById(order_id)
    return order
}

async function verifyOrder(order_id, status) {
    let order = await orderRepository.findOrderById(order_id)
    if(!order){
        throw new Error('Order not found')
    }

    await orderRepository.updateOrderStatus(order_id, status, status === 'ON_PROCESS' ? 'updated_at' : null)

    if(status === 'ON_PROCESS'){
        let product = await productRepository.findProductById(order.product_id)
        if(!product){
            throw new Error('Product not found')
        }
        let quantity = await productRepository.findQuantityById(order.product_id)
        let newQuantity = quantity.quantity_of_product - order.quantity
        if(newQuantity < 0){
            throw new Error('Insuficient quantity')
        }

        await productRepository.updateProductQuantity(quantity.product_id, newQuantity)
    }
}

async function rejectOrder(order_id) {
    let order = await orderRepository.findOrderById(order_id)

    if(!order){
        throw new Error('Order not found')
    }
    if(order.status !== 'ON_PROCESS'){
        throw new Error('Cannot reject order. Order status is not On Process')
    }

    await orderRepository.updateOrderStatus(order_id, 'REJECT', 'updated_at')

    let quantity = await productRepository.findQuantityById(order.product_id)
    let newQuantity = quantity.quantity_of_product + order.quantity
    await productRepository.updateProductQuantity(quantity.product_id, newQuantity)
}


module.exports = {createOrder, getAllOrders, getOrdersByUserId, getOrdersById, verifyOrder, rejectOrder}
```

Order.controller.js

-Mengimpor Modul: Mengimpor express untuk routing dan orderServices untuk mengelola operasi pesanan.

-Inisialisasi Router: Membuat instance router menggunakan Express.

-Endpoint POST /order:

*Menerima data produk, pengguna, dan jumlah dari permintaan (req.body).

*Menggunakan createOrder dari orderServices untuk membuat pesanan baru dan mengembalikannya dengan status 201.

*Jika gagal, mengembalikan status 400 dengan pesan error.

-Endpoint GET /:

*Mengambil semua pesanan dengan memanggil getAllOrders dari orderServices dan mengembalikannya.

*Jika gagal, mengembalikan status 500 dengan pesan error.

-Endpoint GET /user:

*Menerima ID pengguna dari permintaan (req.body) dan menggunakan getOrdersByUserId untuk mengambil pesanan berdasarkan ID pengguna.

*Mengembalikan pesanan dengan status 200 atau status 500 jika gagal.

-Endpoint PATCH /verify/:order_id:

*Menerima ID pesanan dari parameter URL dan status dari permintaan (req.body).

*Menggunakan verifyOrder dari orderServices untuk memverifikasi pesanan dan mengembalikan pesan sukses dengan status 200.

*Jika gagal, mengembalikan status 400 dengan pesan error.

-Endpoint POST /return/:order_id:

*Menerima ID pesanan dari parameter URL dan ID pengguna dari permintaan (req.body).

*Mengecek apakah pengguna yang mengajukan permintaan adalah pemilik pesanan; jika tidak, mengembalikan status 403 dengan pesan "Unauthorized".

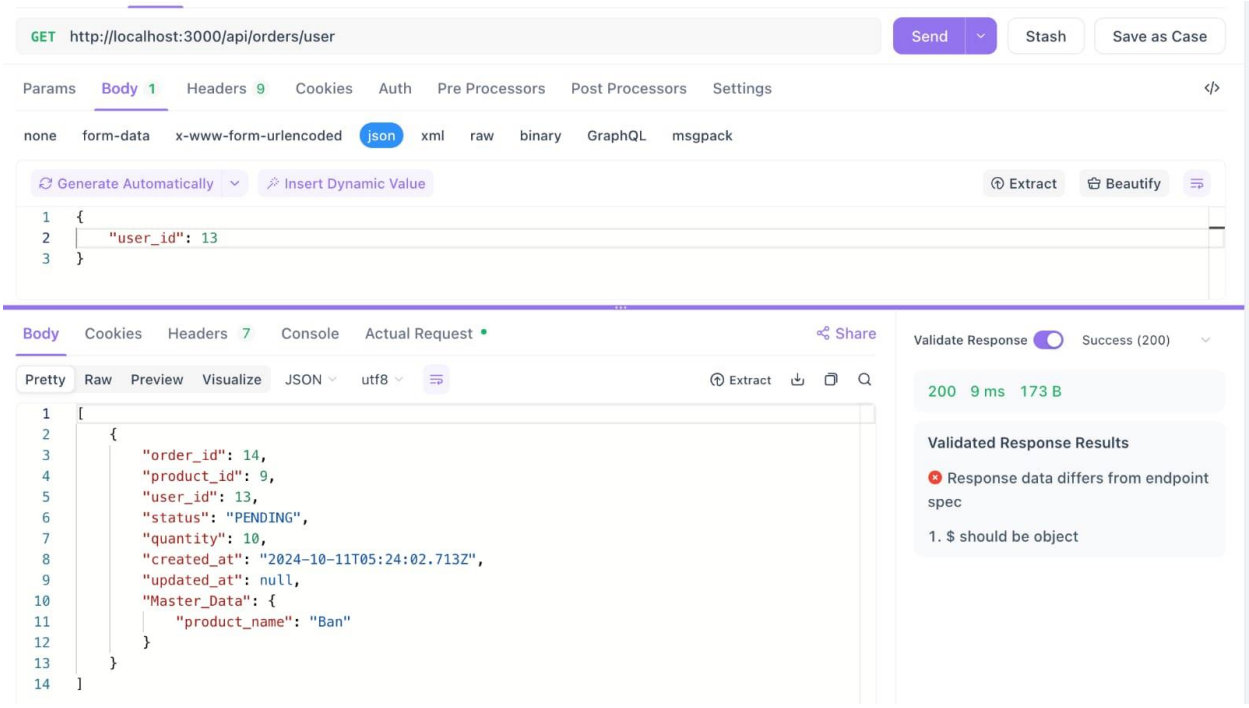
*Menggunakan rejectOrder untuk menolak pesanan dan mengembalikan pesan sukses dengan status 200. Jika gagal, mengembalikan status 400 dengan pesan error.

-Ekspor Router: Mengekspor router agar dapat digunakan di file lain.

```
let express = require('express')
let router = express.Router()
let orderServices = require('./order.services')

router.post('/order', async(req, res) => {

    try {
        let {product_id, user_id, quantity} = req.body
        let newOrder = await orderServices.createOrder(product_id, user_id, quantity)
        res.status(201).json(newOrder)
    } catch (error) {
        res.status(400).send(error.message)
    }
})


router.get('/', async(req, res) => {

    try {
        let orders = await orderServices.getAllOrders()
        res.send(orders)
    } catch (error) {
        res.status(500).send(error.message)
    }
})


router.get('/user', async(req, res) => {

    let {user_id} = req.body
    try {
        let orders = await orderServices.getOrdersByUserId(user_id)
        res.status(200).send(orders)
    } catch (error) {
        res.status(500).send(error.message)
    }
})

router.patch('/verify/:order_id', async(req, res) => {

    try {
        let {order_id} = req.params
        let {status} = req.body
        await orderServices.verifyOrder(order_id, status)
        res.status(200).json({message: 'Order verified successfully'})
    } catch (error) {
        res.status(400).send(error.message)
    }
})

router.post('/return/:order_id', async(req, res) => {

    try {
        let {order_id} = req.params
        let {user_id} = req.body

        let order = await orderServices.getOrdersById(order_id)
        if(order.user_id !== user_id){
            return res.status(403).json({message: 'Unauthorized'})
        }

        await orderServices.rejectOrder(order_id)
        res.status(200).json({message: 'Order Rejected'})
    } catch (error) {
        res.status(400).send(error.message)
    }
})

module.exports = router
```

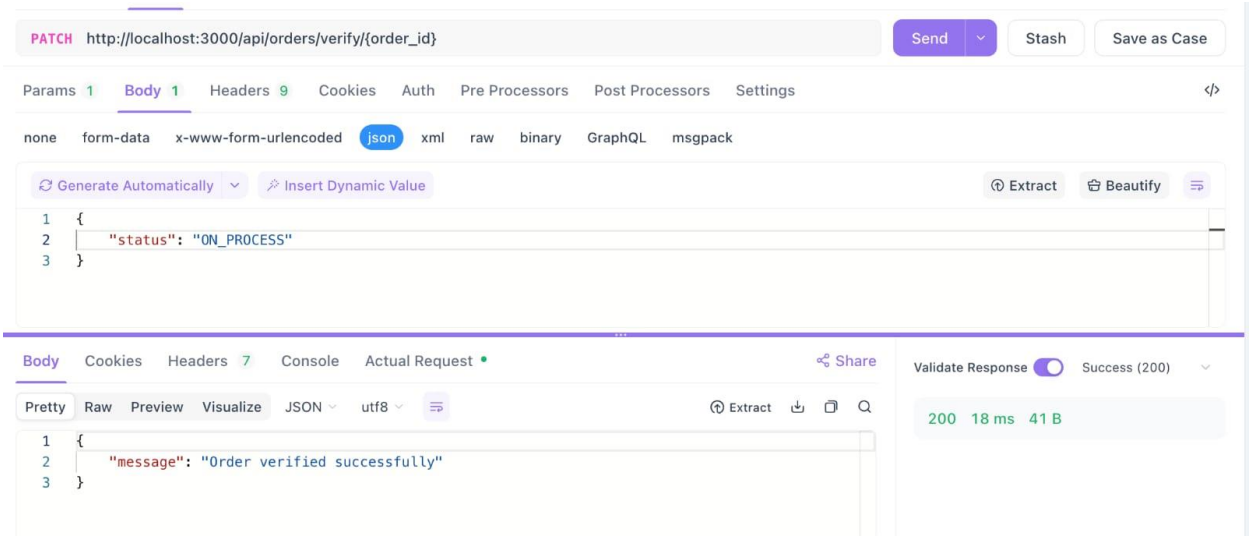
GET Order

Deskripsi: Endpoint ini digunakan untuk mengambil detail pesanan tertentu berdasarkan ID pengguna



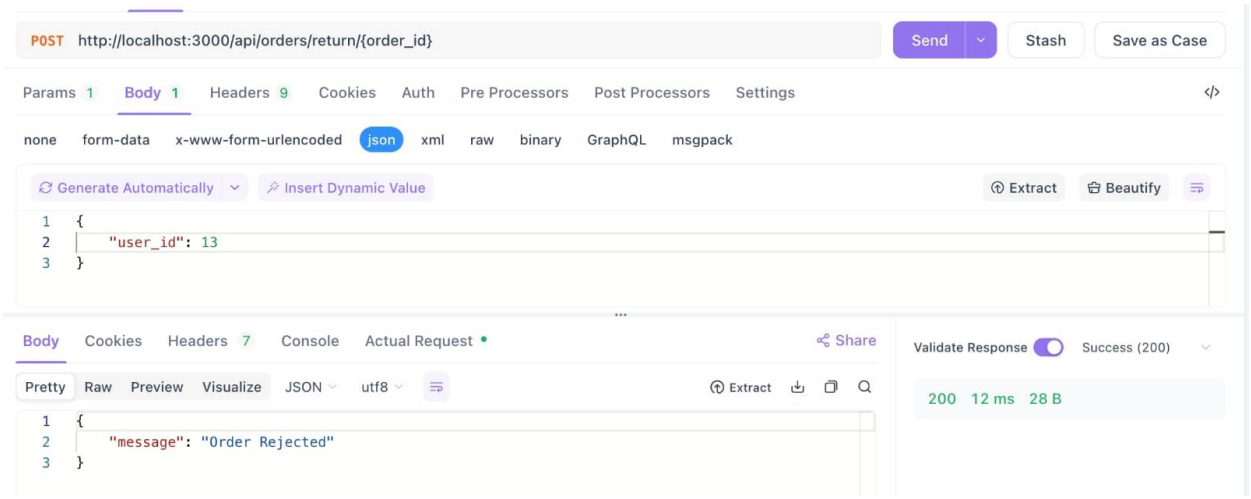
PATCH ORDER

Deskripsi: Endpoint ini dirancang untuk memperbarui status sebuah pesanan (order) tertentu.



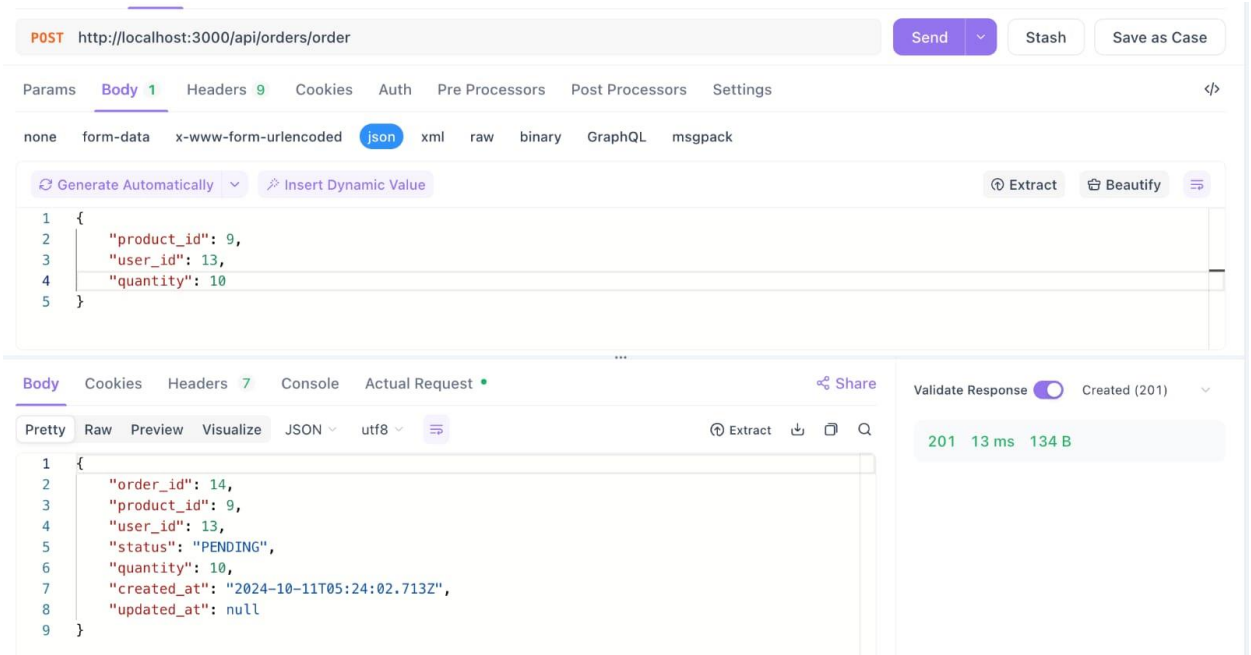
POST ORDER BY ID

Deskripsi: Endpoint ini digunakan untuk **mencatat atau memproses permintaan pengembalian pesanan**. Ketika mengirimkan permintaan ke endpoint ini, sebenarnya sedang memberi tahu aplikasi bahwa Anda ingin mengembalikan pesanan dengan nomor tertentu.



POST ORDER

Deskripsi : Endpoint ini dirancang khusus untuk **membuat pesanan baru** dalam sebuah sistem.



Berikut ini merupakan link Github tugas Kelompok https://github.com/anandanwijaya/supply-chain