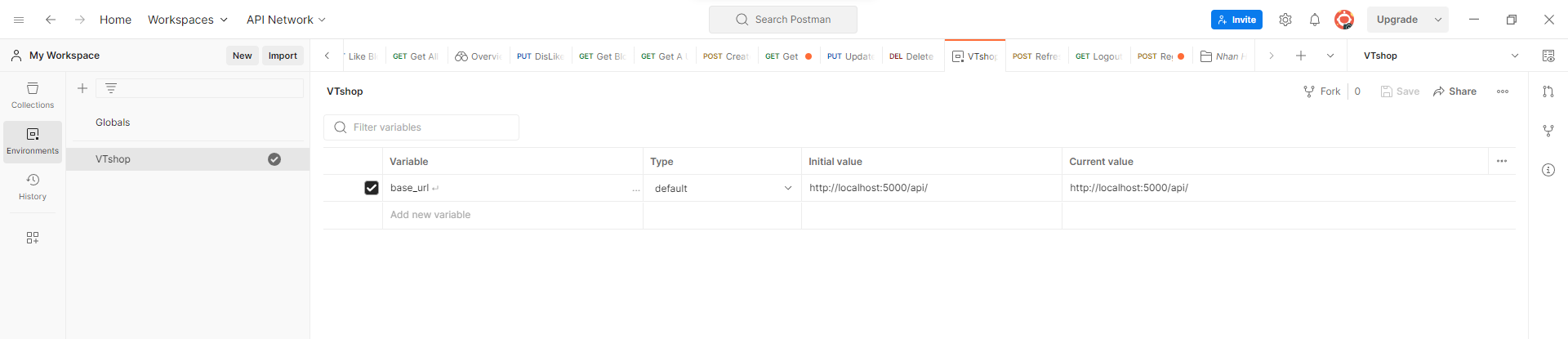
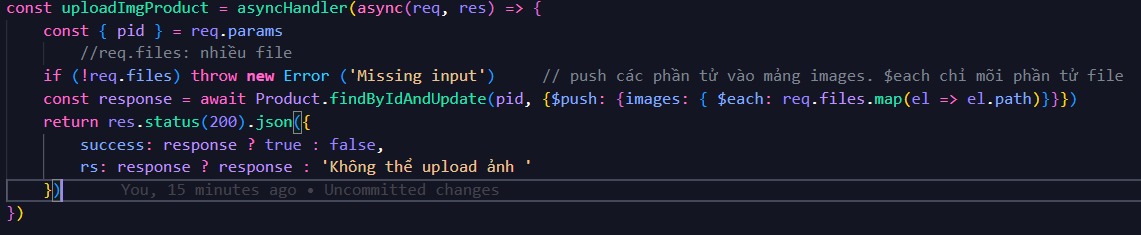
A screenshot of a computer

Description automatically generated



//-----------controllers:

* product.js:



1. Sử dụng phương thức findByIdAndUpdate của model Product để tìm và cập nhật thông tin về sản phẩm. Sử dụng pid làm điều kiện tìm kiếm và { $push: { images: { $each: req.files.map(el => el.path) } } } để thêm các đường dẫn hình ảnh mới vào mảng images của sản phẩm. Sử dụng $each và req.files.map(el => el.path) để truyền một mảng các đường dẫn hình ảnh vào phương thức $push.

* giamgia.js:

const themgiamgia = asyncHandler(async(req, res) => {

    const {name, discount, expiry} = req.body

    if(!name || !discount || !expiry) throw **new** Error('Missing input')

    const response = await GiamGia.create({

        ...req.body,

        // set ngay het han

        expiry: Date.now() + expiry \* 24 \* 60 \* 60 \* 1000

    })

    return res.json({

        success: response ? true : false,

        rs: response ? response: 'không thể tạo giảm giá'

    })

})

1. Lấy các thuộc tính name, discount, expiry từ phần thân yêu cầu (req.body).
2. Kiểm tra xem name, discount, và expiry có tồn tại hay không. Nếu một trong các thuộc tính này không tồn tại, ném ra một lỗi.
3. Sử dụng phương thức create của model GiamGia để tạo mới một đối tượng mã giảm giá trong cơ sở dữ liệu. Đối tượng được tạo mới có các thuộc tính được sao chép từ req.body. Ngoài ra, thuộc tính expiry được tính toán bằng cách cộng thời gian hết hạn (được cung cấp trong expiry) với thời điểm hiện tại (Date.now()) nhân với số mili giây trong một ngày.
4. Hàm trả về một đối tượng JSON chứa thông tin về thành công (success) và kết quả tạo mới (rs). Nếu tạo mới thành công, success sẽ là true và rs sẽ chứa thông tin về mã giảm giá đã được tạo mới. Nếu không thành công, success sẽ là false và rs sẽ là chuỗi 'không thể tạo giảm giá'.

=> Điều này cho phép bạn sử dụng API này để tạo mới một mã giảm giá bằng cách gửi yêu cầu HTTP với phương thức POST và cung cấp thông tin về tên (name), giảm giá (discount), và thời gian hết hạn (expiry) trong phần thân yêu cầu (req.body).

const suagiamgia = asyncHandler(async(req, res) => {

    const { gid } = req.params

    if(Object.keys(req.body).length === 0) throw **new** Error ('Missing input')

        if (req.body.expiry) req.body.expiry = Date.now() + req.body.expiry \* 24 \* 60 \* 60 \* 1000

    const response = await GiamGia.findByIdAndUpdate(gid, req.body, {new: true})

    return res.json({

        success: response ? true : false,

        rs: response ? response: 'không thể sửa giảm giá'

    })

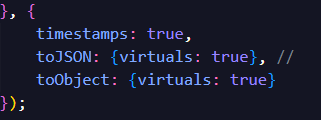
})

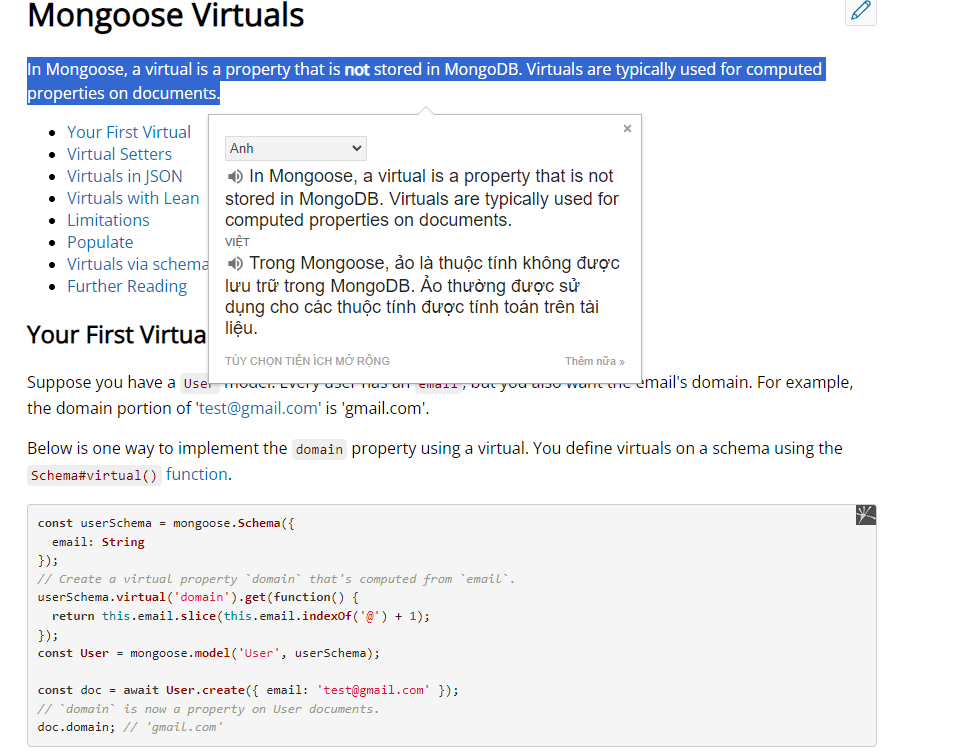
1. Lấy gid từ tham số đường dẫn (req.params), đại diện cho ID của mã giảm giá cần cập nhật.
2. Kiểm tra xem req.body có chứa dữ liệu đầu vào hay không. Nếu không có dữ liệu, ném ra một lỗi.
3. Kiểm tra xem expiry có tồn tại trong req.body hay không. Nếu tồn tại, cập nhật giá trị của expiry bằng cách tính toán thời gian hết hạn dựa trên số ngày (req.body.expiry) được cung cấp. Thời gian hết hạn tính bằng số mili giây từ thời điểm hiện tại (Date.now()) cộng với số ngày nhân với số giây trong một ngày.
4. Sử dụng phương thức findByIdAndUpdate của model GiamGia để tìm và cập nhật thông tin về mã giảm giá. Sử dụng gid làm điều kiện tìm kiếm và req.body làm dữ liệu cập nhật. Tùy chọn {new: true} được sử dụng để trả về đối tượng đã được cập nhật sau khi thực hiện cập nhật.
5. Hàm trả về một đối tượng JSON chứa thông tin về thành công (success) và kết quả cập nhật (rs). Nếu cập nhật thành công, success sẽ là true và rs sẽ chứa thông tin về mã giảm giá đã được cập nhật. Nếu không thành công, success sẽ là false và rs sẽ là chuỗi 'không thể sửa giảm giá'.

=> Điều này cho phép bạn sử dụng API này để cập nhật thông tin về một mã giảm giá bằng cách gửi yêu cầu HTTP với phương thức PUT hoặc PATCH, cung cấp ID (gid) và thông tin cập nhật trong phần thân yêu cầu (req.body).

Models:

blog.js:





1. timestamps: true: Đây là một tùy chọn trong cấu hình schema để tự động thêm hai trường createdAt và updatedAt vào các tài liệu (documents) tương ứng trong cơ sở dữ liệu. Trường createdAt sẽ được tự động gán giá trị thời gian khi một tài liệu được tạo, và trường updatedAt sẽ được tự động cập nhật giá trị thời gian khi một tài liệu được cập nhật.
2. Tùy chọn toJSON: {virtuals: true} được sử dụng trong cấu hình của một schema trong MongoDB khi sử dụng thư viện Mongoose. Đây là cách để đảm bảo rằng các thuộc tính ảo (virtual properties) cũng được bao gồm trong kết quả của phương thức toJSON() khi chuyển đổi một tài liệu thành đối tượng JSON.

Trong MongoDB và Mongoose, các thuộc tính ảo là những thuộc tính không được lưu trữ trong cơ sở dữ liệu, mà được tính toán dựa trên các thuộc tính khác của tài liệu. Ví dụ, trong một schema đại diện cho một tài liệu về người dùng, bạn có thể định nghĩa một thuộc tính ảo là fullName để kết hợp firstName và lastName thành một chuỗi đầy đủ.

Khi bạn gọi phương thức toJSON() trên một tài liệu, nó sẽ chuyển đổi tài liệu đó thành một đối tượng JSON. Mặc định, chỉ các thuộc tính thực (non-virtual properties) sẽ được bao gồm trong đối tượng JSON kết quả. Tuy nhiên, bằng cách sử dụng tùy chọn toJSON: {virtuals: true} trong cấu hình schema, bạn đảm bảo rằng các thuộc tính ảo cũng được bao gồm trong đối tượng JSON đó.

Với tùy chọn này, khi bạn gọi toJSON() trên một tài liệu, đối tượng JSON kết quả sẽ bao gồm cả các thuộc tính thực và các thuộc tính ảo. Điều này giúp bạn có được toàn bộ thông tin cần thiết từ tài liệu khi chuyển đổi thành đối tượng JSON.

1. toOject tương tự toJSON