SMK Telkom Malang

JOB SHEET

NODE JS

Sequelize Bagian IV

Disusun Oleh:
TIM MGMP RPL
SMK TELKOM MALANG

RESTFUL API USING SEQUELIZE

"BORROW" FEATURE

Pada modul sebelumnya kita telah membahas tentang pembuatan proses CRUD untuk data member, admin, dan buku. Selanjutnya kita akan membahas proses peminjaman buku yang melibatkan data member yang meminjam, buku yang dipinjam, dan admin yang melayani peminjaman. Dengan kata lain proses ini melibatkan relasi dari tiga tabel. Ikuti langkah-langkah praktikum ini dengan penuh kecermatan!

A. CRUD Peminjaman Buku

Dalam fitur peminjaman buku pada project ini berkonsep bahwa di tiap peminjaman buku, member dapat meminjam satu atau lebih buku. Oleh karena itu proses CRUD ini melibatkan 2 table sekaligus yaitu data peminjaman akan kita simpan pada table "borrows" dan daftar buku yang dipinjam akan disimpan pada tabel "details_of_borrow". Adapun proses CRUD 2 table ini adalah sebagai berikut.

1. Buat file controller baru dengan nama "borrow.controller.js"!
2. Muat model untuk table "borrow" dan "details_of_borrow" serta load library Operator dari Sequelize!

```
/** load model for `borrow` table */
const borrowModel = require(`../models/index`).borrow

/** load model for `details_of_borrow` table */
const detailsOfBorrowModel =
require(`../models/index`).details_of_borrow

/** load Operator from  Sequelize  */
const Op = require(`sequelize`).Op
```

3. Buat fungsi untuk menambahkan data peminjaman baru!

```
/** create function for add book borrowing */
exports.addBorrowing = async (request, response) => {
    /** prepare data for borrow's table */
    let newData = {
        memberID: request.body.memberID,
        adminID: request.body.adminID,
        date_of_borrow: request.body.date_of_borrow,
        date_of_return: request.body.date_of_return,
        status: request.body.status
    }
```

```
    /** execute for inserting to borrow's table */
    borrowModel.create(newData)
        .then(result => {
            /** get the latest id of book borrowing */
            let borrowID = result.id

            /** store details of book borrowing from request
             * (type: array object)
             */

            let detailsOfBorrow = request.body.details_of_borrow

            /** insert borrowID to each item of detailsOfBorrow
*/
            for (let i = 0; i < detailsOfBorrow.length; i++) {
                detailsOfBorrow[i].borrowID = borrowID
            }

            /** insert all data of detailsOfBorrow */
            detailsOfBorrowModel.bulkCreate(detailsOfBorrow)
                .then(result => {
                    return response.json({
                        success: true,
                        message: `New Book Borrowed has been
inserted`
                    })
                })
                .catch(error => {
                    return response.json({
                        success: false,
                        message: error.message
                    })
                })

        })
        .catch(error => {
            return response.json({
                success: false,
                message: error.message
            })
        })
}
```

4. Buat fungsi untuk mengubah data peminjaman buku!

```
/** create function for update book borrowing */
exports.updateBorrowing = async (request, response) => {
    /** prepare data for borrow's table */
    let newData = {
        memberID: request.body.memberID,
        adminID: request.body.adminID,
        date_of_borrow: request.body.date_of_borrow,
        date_of_return: request.body.date_of_return,
        status: request.body.status
    }

    /** prepare parameter Borrow ID */
    let borrowID = request.params.id

    /** execute for inserting to borrow's table */
    borrowModel.update(newData, { where: { id: borrowID } })
        .then(async result => {

            /** delete all detailsOfBorrow based on borrowID */
            await detailsOfBorrowModel.destroy(
                { where: { borrowID: borrowID } }
            )

            /** store details of book borrowing from request
             * (type: array object)
             */

            let detailsOfBorrow = request.body.details_of_borrow

            /** insert borrowID to each item of detailsOfBorrow
*/
            for (let i = 0; i < detailsOfBorrow.length; i++) {
                detailsOfBorrow[i].borrowID = borrowID
            }

            /** re-insert all data of detailsOfBorrow */
            detailsOfBorrowModel.bulkCreate(detailsOfBorrow)
                .then(result => {
                    return response.json({
                        success: true,
                        message: `Book Borrowed has been
updated`
                    })
                })
                .catch(error => {
                    return response.json({
                        success: false,
```

```
                        message: error.message
                    })
                })

        })
        .catch(error => {
            return response.json({
                success: false,
                message: error.message
            })
        })
}
```

5. Buat fungsi untuk menghapus data peminjaman buku berdasarkan id peminjamanannya!

```
/** create function for delete book borrowing's data */
exports.deleteBorrowing = async (request, response) => {
    /** prepare borrowID that as paramter to delete */
    let borrowID = request.params.id

    /** delete detailsOfBorrow using model */
    detailsOfBorrowModel.destroy(
        { where: { borrowID: borrowID } }
    )
        .then(result => {
            /** delete borrow's data using model */
            borrowModel.destroy({ where: { id: borrowID } })
                .then(result => {
                    return response.json({
                        success: true,
                        message: `Borrowing Book's has deleted`
                    })
                })
                .catch(error => {
                    return response.json({
                        success: false,
                        message: error.message
                    })
                })
        })
        .catch(error => {
            return response.json({
                success: false,
                message: error.message
```

```
            })
        })
}
```

6. Buat fungsi untuk proses pengembalian data peminjaman buku!

```
/** create function for return borrowed book */
exports.returnBook = async (request, response) => {
    /** prepare borrowID that will be return */
    let borrowID = request.params.id

    /** prepare current time for return's time */
    let today = new Date()
    let currentDate = `${today.getFullYear()}-${today.getMonth()
+ 1}-${today.getDate()}`

    /** update status and date_of_return from borrow's data */
    borrowModel.update(
        {
            date_of_return: currentDate,
            status: true
        },
        {
            where: { id: borrowID }
        }
    )
        .then(result => {
            return response.json({
                success: true,
                message: `Book has been returned`
            })
        })
        .catch(error => {
            return response.json({
                success: false,
                message: error.message
            })
        })
}
```

7. Buat fungsi untuk mendapatkan semua data peminjaman buku!

```
/** create function for get all borrowing data */
exports.getBorrow = async (request, response) => {
    let data = await borrowModel.findAll(
```

```
        {
            include: [
                `member`, `admin`,
                {
                    model: detailsOfBorrowModel,
                    as: `details_of_borrow`,
                    include: ["book"]
                }
            ]
        }
    )

    return response.json({
        success: true,
        data: data,
        message: `All borrowing book have been loaded`
    })
}
```

B. Route CRUD Peminjaman Buku

1. Buat file baru pada folder "routes" dengan nama "borrow.route.js"!
2. Buat code untuk memanggil dan inisiasi library "express" seperti berikut!

```
/** load library express */
const express = require(`express`)

/** initiate object that instance of express */
const app = express()
```

3. Buat code untuk membuka akses membaca data request yang bertipe JSON seperti berikut ini!

```
/** allow to read 'request' with json type */
app.use(express.json())
```

4. Buat code untuk memanggil file controller dari member seperti berikut ini!

```
/** load borrow's controller */
const borrowController =
require(`../controllers/borrow.controller`)
```

5. Buat code untuk menentukan route pada proses penambahan data peminjaman buku!

```
/** create route to add new borrowing book */
app.post("/", borrowController.addBorrowing)
```

6. Buat code untuk menentukan route pada pengubahan data peminjaman buku berdasarkan ID yang ditentukan!

```
/** create route to update borrowed book based on ID */
app.put("/:id", borrowController.updateBorrowing)
```

7. Buat code untuk menentukan route pada proses penghapusan data peminjaman buku berdasarkan ID yang ditentukan!

```
/** create toute to delete borrowed book based on ID */
app.delete("/:id", borrowController.deleteBorrowing)
```

8. Buat code untuk menentukan route pada proses pengembalian buku berdasarkan ID yang ditentukan!

```
/** create route to return book */
app.get("/return/:id", borrowController.returnBook)
```

9. Buat code untuk menentukan route pada proses pengambilan data peminjaman buku!

```
/** create route to get all borrowed book */
app.get("/", borrowController.getBorrow)
```

10. Buat code untuk mengekspor objek "app" agar dapat dimuat pada file lain!

```
/** export app in order to load in another file */
module.exports = app
```

11. Pada file "server.js" tambahkan code untuk memuat file route untuk proses peminjaman!

```
const borrowRoute = require(`./routes/borrow.route`)
```

12. Tambahkan code untuk menentukan prefix pada route peminjaman buku!

```
app.use(`/borrow`, borrowRoute)
```

C. Tes API Peminjaman Buku

1. Jalankan server backend dengan "npm start"!

2. Buka Aplikasi Postman dan tambahkan folder "borrowed" pada collection sebelumnya!
3. Tambahkan *request* baru untuk menambah data peminjaman baru, pilih method POST dan tuliskan *request URL* dengan **http://localhost:8000/borrow**. Kemudian tambahkan data request pada tab **Body**, pilih option "**raw**" dan pilih format **JSON**. Sebagai contoh data *request* yang ditambahkan adalah sebagai berikut.

```
{
    "memberID":1,
    "adminID":1,
    "date_of_borrow":"2022-05-05",
    "date_of_return":null,
    "status":false,
    "details_of_borrow":[
        { "bookID": 1, "qty": 2}
    ]
}
```

   Klik Send untuk mendapatkan *response*. Jangan lupa untuk "***Save***" request tersebut! Pada request di atas, key "details_of_borrow" bertipe array agar dapat menampung banyak data buku yang dipinjam.
4. Tambahkan *request* baru untuk menampilkan seluruh data peminjaman buku, pilih method GET dan tuliskan *request URL* dengan **http://localhost:8000/borrow**. Klik Send untuk mendapatkan *response*. Jangan lupa untuk "***Save***" request tersebut!
5. Tambahkan *request* baru untuk mengubah status data peminjaman menjadi status telah dikembalikan berdasarkan id peminjaman yang dipilih, pilih method GET dan tuliskan *request URL* dengan **http://localhost:8000/borrow/return/1**. Klik Send untuk mendapatkan *response*. Jangan lupa untuk "***Save***" request tersebut! Pastikan untuk melihat perubahan status data peminjaman dengan mengakses request untuk penampilan data!
6. Tambahkan *request* baru untuk menghapus data peminjaman berdasarkan id peminjaman yang dipilih, pilih method DELETE dan tuliskan *request URL* dengan **http://localhost:8000/borrow/1**. Klik Send untuk mendapatkan *response*. Jangan lupa untuk "***Save***" request tersebut!

TUGAS PRAKTIKUM

Buatlah endpoint untuk memfilter data peminjaman buku berdasarkan member yang meminjamnya!