

**JOB SHEET**

**NODE JS**

**Sequelize Bagian III**

Disusun Oleh:  
**TIM MGMP RPL**  
**SMK TELKOM MALANG**

## RESTFUL API WITH UPLOAD FILE USING SEQUELIZE

Pada modul sebelumnya kita telah membuat proses CRUD pada tabel member dan admin. Pada modul ini kita akan membahas pembuatan CRUD yang di dalamnya terdapat proses unggah file.

### A. CRUD Tabel Books.

1. Install library multer

Pada tahap ini kita akan menginstall library multer yang digunakan untuk membaca data *request* bertipe file. Untuk install library tersebut menggunakan perintah seperti berikut.

```
D:\Nodejs\school_library>npm install multer
```

Setelah itu buatlah file controller baru untuk proses CRUD table books dengan nama **book.controller.js**.

2. Buat code untuk memanggil model untuk tabel "book", object Operation dari library Sequelize, library path, dan library filestream seperti berikut!

```
/** load model for table 'books' */
const bookModel = require(`../models/index`).book

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

/** load library 'path' and 'filestream' */
const path = require(`path`)
const fs = require(`fs`)
```

3. Buat fungsi untuk mendapatkan semua data buku seperti berikut!

```
/** create function to read all data */
exports.getAllBooks = async (request, response) => {
  /** call findAll() to get all data */
  let books = await bookModel.findAll()
  return response.json({
    success: true,
    data: books,
    message: `All Books have been loaded`
  })
}
```

4. Buat fungsi untuk pencarian data member berdasarkan keyword yang ditentukan seperti berikut!

```

/** create function for filter */
exports.findBook = async (request, response) => {
  /** define keyword to find data */
  let keyword = request.body.keyword

  /** call findAll() within where clause and operation
   * to find data based on keyword */
  let books = await bookModel.findAll({
    where: {
      [Op.or]: [
        { isbn: { [Op.substring]: keyword } },
        { title: { [Op.substring]: keyword } },
        { author: { [Op.substring]: keyword } },
        { category: { [Op.substring]: keyword } },
        { publisher: { [Op.substring]: keyword } }
      ]
    }
  })
  return response.json({
    success: true,
    data: books,
    message: `All Books have been loaded`
  })
}

```

5. Sebelum kita membuat fungsi untuk penambahan data buku baru kita akan membuat fungsi untuk proses upload file terlebih dahulu. Buat folder baru dengan nama "cover" pada root project! Folder tersebut digunakan untuk menyimpan file yang diunggah..
6. Pada folder controllers, buat file baru dengan nama "**upload-cover.js**" dan buat kode konfigurasi upload file di dalamnya seperti berikut.

```

/** load library 'multer' and 'path' */
const multer = require(`multer`)
const path = require(`path`)

/** storage configuration */
const storage = multer.diskStorage({
  /** define storage folder */
  destination: (req, file, cb) => {
    cb(null, `./cover`)
  },

  /** define filename for upload file */
  filename: (req, file, cb) => {

```

```

        cb(null, `cover-${
Date.now()}${path.extname(file.originalname)}`)
    }
})

const upload = multer({
    /** storage configuration */
    storage: storage,
    /** filter uploaded file */
    fileFilter: (req, file, cb) => {
        /** filter type of file */
        const acceptedType = [`image/jpg`, `image/jpeg`,
`image/png`]
        if (!acceptedType.includes(file.mimetype)) {
            cb(null, false) /** refuse upload */
            return cb(`Invalid file type (${file.mimetype})`)
        }

        /** filter size of file */
        const fileSize = req.headers[`content-length`]
        const maxSize = (1 * 1024 * 1024) /** max: 1MB */
        if(fileSize > maxSize){
            cb(null, false) /** refuse upload */
            return cb(`File size is too large`)
        }

        cb(null, true) /** accept upload */
    }
})
module.exports = upload

```

7. Pada file book.controller.js, buat code untuk memuat file "upload-cover.js" yang telah dibuat sebelumnya!

```

/** load function from `upload-cover`
 * single(`cover`) means just upload one file
 * with request name `cover`
 */
const upload = require(`./upload-cover`).single(`cover`)

```

8. Pada file book.controller.js, buatlah fungsi untuk menambah data buku baru dengan code seperti berikut ini!

```

/** create function to add new book */

```

```
exports.addBook = (request, response) => {
  /** run function upload */
  upload(request, response, async error => {
    /** check if there are error when upload */
    if (error) {
      return response.json({ message: error })
    }

    /** check if file is empty */
    if (!request.file) {
      return response.json({ message: `Nothing to Upload`
    })
  })

  /** prepare data from request */
  let newBook = {
    isbn: request.body.isbn,
    title: request.body.title,
    author: request.body.author,
    publisher: request.body.publisher,
    category: request.body.category,
    stock: request.body.stock,
    cover: request.file.filename
  }

  /** execute inserting data to book's table */
  bookModel.create(newBook)
    .then(result => {
      /** if insert's process success */
      return response.json({
        success: true,
        data: result,
        message: `New book has been inserted`
      })
    })
    .catch(error => {
      /** if insert's process failed */
      return response.json({
        success: false,
        message: error.message
      })
    })
  })
}
```

9. Buat fungsi untuk mengubah data buku seperti berikut ini!

```
/** create function to update book */
exports.updateBook = async (request, response) => {
  /** run upload function */
  upload(request, response, async error => {
    /** check if there are error when upload */
    if (error) {
      return response.json({ message: error })
    }

    /** store selected book ID that will update */
    let id = request.params.id

    /** prepare book's data that will update */
    let book = {
      isbn: request.body.isbn,
      title: request.body.title,
      author: request.body.author,
      publisher: request.body.publisher,
      category: request.body.category,
      stock: request.body.stock
    }

    /** check if file is not empty,
     * it means update data within reupload file
     */
    if (request.file) {
      /** get selected book's data */
      const selectedBook = await bookModel.findOne({
        where: { id: id }
      })

      /** get old filename of cover file */
      const oldCoverBook = selectedBook.cover

      /** prepare path of old cover to delete file */
      const pathCover = path.join(__dirname, `../cover`,
oldCoverBook)

      /** check file existence */
      if (fs.existsSync(pathCover)) {
        /** delete old cover file */
        fs.unlink(pathCover, error =>
console.log(error))
      }
    }
  })
}
```

```

        /** add new cover filename to book object */
        book.cover = request.file.filename
    }

    /** execute update data based on defined id book */
    bookModel.update(book, { where: { id: id } })
    .then(result => {
        /** if update's process success */
        return response.json({
            success: true,
            message: `Data book has been updated`
        })
    })
    .catch(error => {
        /** if update's process fail */
        return response.json({
        })
    })
})
}

```

10. Buat fungsi untuk menghapus data buku seperti berikut ini!

```

/** create function to delete book */
exports.deleteBook = async (request, response) => {
    /** store selected book's ID that will be delete */
    const id = request.params.id

    /** -- delete cover file -- */
    /** get selected book's data */
    const book = await bookModel.findOne({ where: { id: id } })
    /** get old filename of cover file */
    const oldCoverBook = book.cover

    /** prepare path of old cover to delete file */
    const pathCover = path.join(__dirname, `../cover`,
oldCoverBook)

    /** check file existence */
    if (fs.existsSync(pathCover)) {
        /** delete old cover file */
        fs.unlink(pathCover, error => console.log(error))
    }
    /** -- end of delete cover file -- */
}

```

```
/** execute delete data based on defined id book */
bookModel.destroy({ where: { id: id } })
  .then(result => {
    /** if update's process success */
    return response.json({
      success: true,
      message: `Data book has been deleted`
    })
  })
  .catch(error => {
    /** if update's process fail */
    return response.json({
      success: false,
      message: error.message
    })
  })
})
```

11. Buat file route baru pada folder routes dengan nama "book.route.js"!
12. 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()
```

13. 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())
```

14. Buat code untuk memanggil file controller dari book seperti berikut ini!

```
/** load book's controller */
const bookController = require(`../controllers/book.controller`)
```

15. Buat code untuk menentukan route proses mendapatkan data semua buku seperti berikut!

```
/** create route to get data with method "GET" */
app.get("/", bookController.getAllBooks)
```



16. Buat code untuk menentukan route proses mendapatkan data buku berdasarkan keyword pencarian seperti berikut!

```
/** create route to find book
 * using method "POST" and path "find" */
app.post("/find", bookController.findBook)
```

17. Buat code untuk menentukan route proses penambahan data buku baru yang didalamnya terdapat proses upload file cover buku seperti berikut!

```
/** create route to add new book using method "POST"
 *
 */
app.post("/", [upload.single('cover')], bookController.addBook)
```

18. Buat code untuk menentukan route proses pengubahan data buku baru yang didalamnya terdapat proses upload file cover buku seperti berikut!

```
/** create route to update book
 * using method "PUT"
 * and define parameter for "id" */
app.put("/:id", bookController.updateBook)
```

19. Buat code untuk menentukan route proses penghapusan data buku seperti berikut ini!

```
/** create route to delete book
 * using method "DELETE" and define parameter for "id" */
app.delete("/:id", bookController.deleteBook)
```

20. Buat code untuk mengekspor object "app" agar dapat dipanggil pada file lain!

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

21. Pada file server.js, tambahkan code untuk memuat route CRUD book seperti berikut ini!

```
const bookRoute = require(`./routes/book.route`)
```

22. Buat code untuk menentukan prefix route book seperti berikut ini!

```
app.use(`/book`, bookRoute)
```

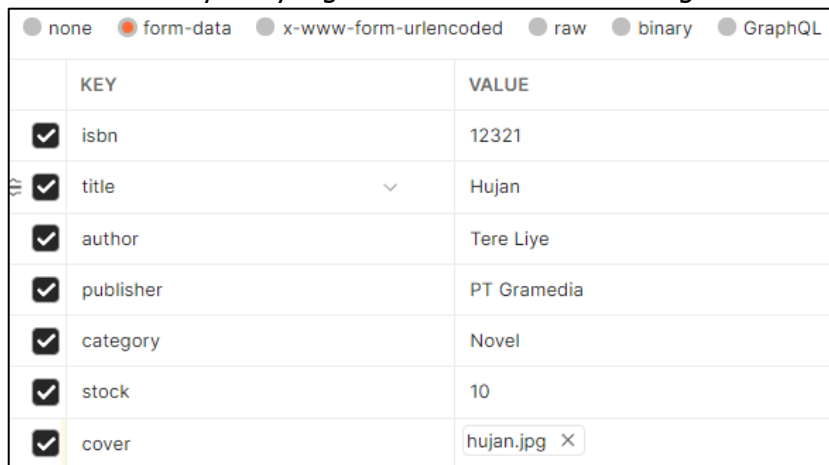
23. Buat code untuk membuka route pemanggilan file cover!

```
/** route to access uploaded file */
app.use(express.static(__dirname))
```

Simpan file dan saatnya mencoba REST API CRUD Book yang telah dibuat!

## B. Testing API CRUD Book

1. Buka Aplikasi Postman dan tambahkan folder baru untuk menyimpan request proses CRUD table Book!
2. Tambahkan *request* baru untuk menambah data buku baru, pilih method POST dan tuliskan *request URL* dengan <http://localhost:8000/book>. Kemudian tambahkan data request pada tab **Body**, pilih option "**form-data**". Sebagai contoh data *request* yang ditambahkan adalah sebagai berikut.



	KEY	VALUE
<input checked="" type="checkbox"/>	isbn	12321
<input checked="" type="checkbox"/>	title	Hujan
<input checked="" type="checkbox"/>	author	Tere Liye
<input checked="" type="checkbox"/>	publisher	PT Gramedia
<input checked="" type="checkbox"/>	category	Novel
<input checked="" type="checkbox"/>	stock	10
<input checked="" type="checkbox"/>	cover	hujan.jpg

Klik Send untuk mendapatkan *response*. Jangan lupa untuk "**Save**" request tersebut.

3. Tambahkan *request* baru untuk mengubah data buku misal dengan ID Buku 1, pilih method PUT dan tuliskan *request URL* dengan <http://localhost:8000/book/1> (kode 1 adalah id buku yang akan diubah). Kemudian tambahkan data request pada tab **Body**, pilih option "**form-data**". Sebagai contoh data *request* perubahan data buku adalah sebagai berikut.

KEY	VALUE
<input checked="" type="checkbox"/> isbn	12321
<input checked="" type="checkbox"/> title	Bumi
<input checked="" type="checkbox"/> author	Tere Liye
<input checked="" type="checkbox"/> publisher	PT Gramedia
<input checked="" type="checkbox"/> category	Novel
<input checked="" type="checkbox"/> stock	10
<input checked="" type="checkbox"/> cover	<input type="text" value="bumi.jpg"/>

Klik Send untuk mendapatkan *response*. Jangan lupa untuk "**Save**" request tersebut.

4. Tambahkan *request* baru untuk menghapus data buku, pilih method DELETE dan tuliskan *request URL* dengan <http://localhost:8000/book/1> (kode 1 adalah ID buku yang akan dihapus). Klik Send untuk mendapatkan *response*. Jangan lupa untuk "**Save**" request tersebut!
5. Tambahkan *request* baru untuk mengambil semua data buku, pilih method GET dan tuliskan *request URL* dengan <http://localhost:8000/book> dan klik Send untuk mendapatkan *response*. Jangan lupa untuk "**Save**" request tersebut.
6. Tambahkan *request* baru untuk mencari data buku, pilih method POST dan tuliskan *request URL* dengan <http://localhost:8000/book/find>. Kemudian tambahkan data request pada tab **Body**, pilih option "**raw**" dan pilih format **JSON**. Sebagai contoh data *request* yang ditambahkan adalah sebagai berikut.

```
{
  "keyword": "Hujan"
}
```

Klik Send untuk mendapatkan *response*. Jangan lupa untuk "**Save**" request tersebut.

### Tugas Praktikum!

Buatlah fitur unggah file image pada data member!