Asynchronous Programming Dengan Promise di Node.js

Pengertian Promise:

Promise adalah objek yang merepresentasikan keberhasilan atau kegagalan dari operasi asynchronous. Promise memiliki tiga state, yaitu Pending, yang merupakan state awal di mana operasi belum selesai dan belum dalam kondisi fulfilled atau rejected. Fulfilled menandakan bahwa operasi telah berhasil diselesaikan, sedangkan Rejected berarti operasi mengalami kegagalan.

Karakteristik Promise di nodejs**:**

1. Chainable: Promise dapat dirantai menggunakan .then() dan .catch()
2. Error Handling: Menggunakan .catch() untuk menangani error
3. Promise.all(): Menjalankan beberapa Promise secara parallel
4. Lebih mudah dibaca dibanding callback

Deskripsi Modul:

Pada praktikum ini, mahasiswa akan melanjutkan pengembangan aplikasi backend sistem reservasi restoran menggunakan Node.js dengan fokus pada manajemen meja menggunakan Promise. Modul ketiga ini akan membahas implementasi Promise pattern dalam mengelola meja restoran, termasuk pembuatan meja baru, pengecekan ketersediaan, dan reservasi meja.

Tujuan Pembelajaran:

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

1. Memahami konsep Promise dan perbedaannya dengan callback dan Async/Await.
2. Mengimplementasikan operasi asynchronous menggunakan Promise dalam konteks manajemen meja restoran.
3. Membuat endpoint untuk manajemen meja (create, read, update) dengan menggunakan Promise.
4. Menerapkan error handling menggunakan .catch() dan memahami cara menjalankan beberapa Promise secara parallel dengan Promise.all().
5. Mengintegrasikan manajemen meja dengan modul lain dalam sistem reservasi restoran.

Kesimpulan:

Modul ini dirancang untuk membantu mahasiswa memahami dan mengimplementasikan Promise dalam konteks manajemen meja restoran. Dengan mengikuti modul ini, mahasiswa akan memahami cara kerja Promise, mampu membuat endpoint untuk manajemen meja, menerapkan error handling dengan baik, serta mengintegrasikan manajemen meja dengan modul lain dalam sistem reservasi restoran. Dengan demikian, mahasiswa akan memiliki pemahaman yang kuat tentang pengembangan aplikasi backend menggunakan Node.js dan Promise.

Praktikum 3: Implementasi Meja Management dengan Promise

Langkah 1: Setup Model Meja

1. Buat file mejaModel.js di folder src/models/mejaModel.js
2. Tambahkan kode berikut di dalam file mejaModel.js

```
const mongoose = require('mongoose');


const tableSchema = new mongoose.Schema({
    tableNumber: {
        type: Number,
        required: true,
        unique: true
    },
    capacity: {
        type: Number,
        required: true
    },
    isOccupied: {
        type: Boolean,
        default: false
    },
    status: {
        type: String,
        enum: ['available', 'reserved'],
        default: 'available'
    },
    customerName: {
        type: String,
        default: ''
    },
    createdAt: {
        type: Date,
        default: Date.now
    }
});


module.exports = mongoose.model('Meja', tableSchema);
```

Langkah 2: Implementasi Meja Controller

1. Buat file mejaController.js di dalam folder src/controller/mejaController.js
2. Tambahkan kode created meja berikut di file mejaController.js

```
const createMeja = (req, res) => {
    const { tableNumber, capacity } = req.body;
    return Meja.create({ tableNumber, capacity })
        .then(meja => {res.status(201).json({
                success: true,
                data: meja});
        })
        .catch(error => {res.status(400).json({
                success: false,
                error: error.message});
        });
};
```

3. Tambahkan kode get all meja berikut di file mejaController.js

```
const getAllMeja = (req, res) => {
    return Meja.find()
        .then(meja => {res.status(200).json({
                success: true,
                data: meja});
        })
        .catch(error => {res.status(400).json({
                success: false,
                error: error.message});
        });
};
```

4. Tambahkan kode reserve meje berikut pada file mejaController.js

```
const reserveMeja = (req, res) => {
    const { tableNumber } = req.params;
    const { customerName } = req.body;
    // Validasi nama pelanggan
    if (!customerName) {
        return res.status(400).json({
            success: false,
            error: 'Nama pelanggan harus diisi'
        });
    }
    return Meja.findOneAndUpdate(
        { tableNumber, status: 'available' },
        { status: 'reserved',customerName},
        { new: true }
    )
        .then(meja => {
            if (!meja) {
                return res.status(404).json({
                    success: false,
                    error: 'Meja tidak tersedia'
                });
            }
            res.status(200).json({
                success: true,
                data: meja
            });
        })
        .catch(error => {
            res.status(400).json({
                success: false,
                error: error.message
            });
        });
};
```

5. Ekspor tiap fungsi secara individual

Langkah 3: Implementasi Routes

1. Buat file mejaRoute.js di dalam folder src/routes/mejaRoute.js
2. Tambahkan kode berikut di file mejaRoute.js

```
const express = require('express');

const router = express.Router();

const mejaController = require('../controllers/mejaController');


router.post('/createMeja', mejaController.createMeja);

router.get('/meja', mejaController.getAllMeja);

router.put('/meja/:tableNumber/reserve', mejaController.reserveMeja);


module.exports = router;
```

3. Import directory mejaRoute.js ke file app.js

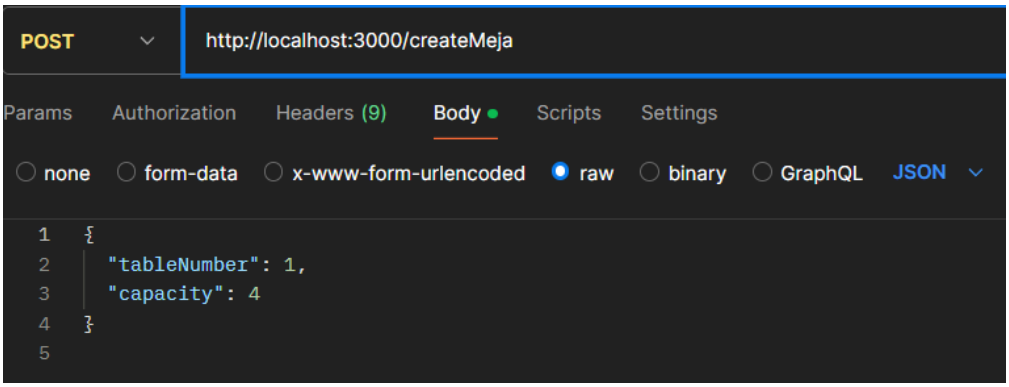
```
const mejaRoutes = require('./src/routes/mejaRoutes');
```

4. Selanjutnya tambahkan kode berikut di file app.js

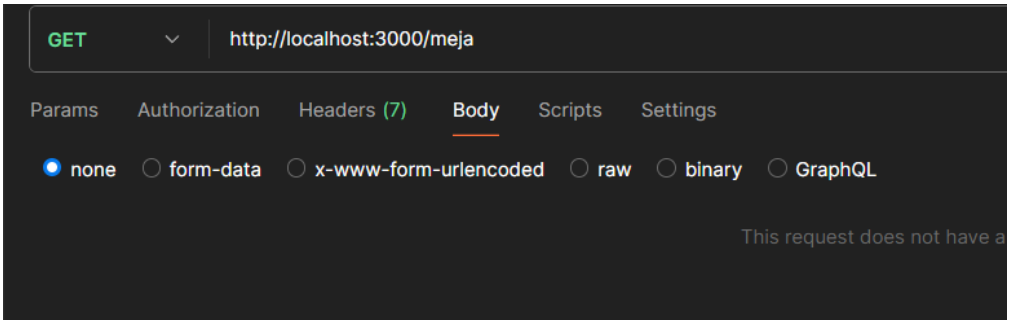
```
app.use('/', mejaRoutes);
```

Langkah 4: Testing dengan Postman

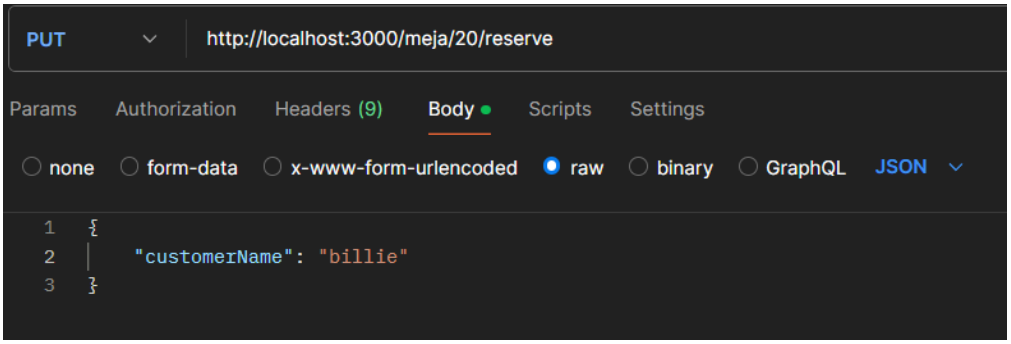
1. Create Meja (POST/ http://localhost:{port}/add/meja)



2. Get All meja (GET/ http://localhost:{port}/meja)



3. Reservasi meja (PUT/http://localhost:{port}/meja/:tableNumber/reserve/)



Tase Case

1. Buatlah API untuk mengcancel reservasi dengan clue berikut

Controllers

```
const cancelReservation = (req, res) => {
    const { tableNumber } = _____________; // TODO: Ambil nomor meja dari parameter URL
    return Meja.findOneAndUpdate(
        {
            tableNumber,
            status: _____________ // TODO: hanya meja yang "reserved" yang bisa dibatalkan
        },
        {
            status: _____________, // TODO: Ubah status menjadi "available"
           customerName: _____________, // TODO
            updatedAt: _____________ // TODO: Simpan waktu pembatalan reservasi
        },
        { new: true }
    )
    .then(meja => {
        if (!meja) {
            return res.status(404).json({
                success: false,
                error: 'Table not found or not currently reserved'
            });
        }
        res.status(200).json({
            success: true,
            message: `Reservation for table ${tableNumber} has been cancelled`,
            data: _____________ // TODO: Kirimkan data meja yang sudah diperbarui
        });
    })
    .catch(error => {
        res.status(400).json({
            success: false,
            error: _____________ // TODO: Kirimkan pesan error jika terjadi kesalahan
        });
    });
};
```

Routes

```
router.get("/meja/:__________/cancel", mejaController.__________);
// TODO: Lengkapi parameter dan fungsi yang dipanggil
```