“Build your imagination with Naratass Software”

Backend Dasar dengan Node.js + Express Bagian 3

Mengimplementasikan operasi CRUD (Create, Read, Update, Delete) untuk data Catatan

Pertemuan 14— Belajar JavaScript Tingkat Basic
Disampaikan oleh: Aries Indrabayu Yunandar

Tujuan Pembelajaran

- Dasar JavaScript untuk backend
- Cara kerja Express dan routing
- Struktur proyek backend modern
- Membaca dan memodifikasi controller & router
- Validasi dan migrasi database

Struktur Folder

```
catatan-api/
├── config/
│   └── db.js
├── controller/
│   └── catatanController.js
├── middlewares/
│   └── validate.js
├── migrations/
├── node_modules/
├── routes/
│   └── catatan.js
├── validators/
│   ├── catatanValidator.js
│   ├── paramValidator.js
│   └── queryValidator.js
├── .env
├── knexfile.js
├── package-lock.json
├── package.json
└── server.js
```

controller/catatanController.js

```
// CREATE
const createCatatan = async (req, res) => {
  try {
    const { isi, kategori } = req.body;
    const [result] = await db.query(
      `INSERT INTO catatan (isi, kategori) VALUES (?, ?)` ,
      [isi, kategori ?? true]
    );
    res.status(201).json({ status: "success", id: result.insertId });
  } catch (err) {
    res.status(500).json({ status: "error", message: err.message });
  }
};

// UPDATE
const updateCatatan = async (req, res) => {
  try {
    const { isi, kategori } = req.body;
    const [result] = await db.query(
      `UPDATE catatan SET isi = ?, kategori = ? WHERE id = ?` ,
      [isi, kategori, req.params.id]
    );
    res.json({ status: result.affectedRows ? "success" : "not_found"
});
  } catch (err) {
    res.status(500).json({ status: "error", message: err.message });
  }
};
```

```
// DELETE
const DeleteCatatan = async (req, res) => {
  try {
    const [result] = await db.query(`DELETE FROM catatan WHERE id
= ?`, [
      req.params.id,
    ]);
    if (result.affectedRows) {
      res.json({
        status: "success",
        message: "Catatan berhasil dihapus permanen" ,
      });
    } else {
      res
        .status(404)
        .json({ status: "not_found", message: "Data tidak
ditemukan" });
    }
  } catch (err) {
    res.status(500).json({ status: "error", message: err.message
});
  }
};

module.exports = {
  getAllCatatan,
  getCatatanById,
  createCatatan,
  updateCatatan,
  DeleteCatatan,
};
```

Validasi → express-validator

express-validator adalah **middleware** untuk **validasi dan sanitasi** input di aplikasi Express.js. Library ini dibangun di atas validator.js, dan sangat membantu untuk:

- Memastikan **data yang dikirim user valid**, misalnya: email valid, password tidak kosong, usia harus angka, dsb.
- Membersihkan input dari karakter-karakter tak aman (sanitasi), contohnya menghindari XSS.

Perintah untuk menambahkan express-validator:

npm install express-validator

✅ **Kapan Harus Pakai express-validator?**

Gunakan ketika:

- Membuat API yang menerima data dari user.
- Ingin pastikan input valid sebelum diproses.
- Ingin hindari validasi manual yang panjang dan rawan kesalahan.

validators/catatanValidator.js

```
const { body } = require("express-validator");

// Validasi untuk CREATE
const createCatatanRules = () => [
  body("isi")
    .notEmpty()
    .withMessage("Isi tidak boleh kosong")
    .isLength({ min: 3 })
    .withMessage("Isi minimal 3 karakter"),

  body("kategori")
    .optional()
    .isLength({ max: 50 })
    .withMessage("Kategori maksimal 50 karakter"),
];
```

```
// Validasi untuk UPDATE (semua optional, tapi tetap
divalidasi jika dikirim)
const updateCatatanRules = () => [
  body("isi")
    .optional()
    .isLength({ min: 3 })
    .withMessage("Isi minimal 3 karakter"),

  body("kategori")
    .optional()
    .isLength({ max: 50 })
    .withMessage("Kategori maksimal 50 karakter"),
];

module.exports = {
  createCatatanRules,
  updateCatatanRules,
};
```

validators/paramValidator.js

```
const { param } = require("express-validator");

const validateIdParam = () => [
  param("id")
    .exists()
    .withMessage("ID wajib disertakan")
    .bail()
    .isInt({ gt: 0 })
    .withMessage("ID harus berupa angka lebih dari 0"),
];

module.exports = { validateIdParam };
```

middlewares/validate.js

```
const { validationResult } = require("express-validator");

const validate = (req, res, next) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(422).json({
      status: "validation_error",
      errors: errors.array(),
    });
  }
  next();
};

module.exports = validate;
```

routes/catatan.js

```
const express = require("express");
const router = express.Router();
const controller =
require("../../controllers/catatanController");
const {
  createCatatanRules,
  updateCatatanRules,
} = require("../../validators/catatanValidator");
const validate = require("../../middlewares/validate");

const { validateIdParam } =
require("../../validators/paramValidator");


// buat route GET Catatan --> membaca semua catatan
router.get("/catatan", controller.getAllCatatan);


// buat route membaca by ID
router.get("/catatan/:id", controller.getCatatanById);



// CREATE → Validasi penuh
router.post(
  "/catatan",
  createCatatanRules(),
  validate,
  controller.createCatatan
);
```

```
// UPDATE → Validasi fleksibel
router.put(
  "/catatan/:id",
  validateIdParam(),
  updateCatatanRules(),
  validate,
  controller.updateCatatan
);

// DELETE
router.delete(
  "/catatan/:id",
  validateIdParam(),
  validate,
  controller.DeleteCatatan
);

module.exports = router;
```

server.js

```
const express = require("express");
const app = express();
const catatanRoutes = require("./routes/catatan");
app.use(express.json());
app.get("/", (req, res) => {
  res.send("API Catatan Siap!");
});

// Daftarkan routing catatan
// Prefix api
app.use("/api", catatanRoutes);
app.listen(5000, () => console.log("Server jalan di http://localhost:5000"));
```

Thank you