

# **LAPORAN TUGAS UAS BACKEND MEMBUAT WEBSITE TEMA WALLPAPER**



**Dibuat Oleh:**

**Achmad Aldino/3124101290**

**Miftahul Jannah/3124101289**

**Program Studi D3 Manajemen Informatika  
STIKOM PGRI Banyuwangi  
Tahun 2025**

## A. Struktur Folder Project Website Wallpaper

### Backend

- └─ configs
  - | └─ db.js
- └─ controllers
  - | └─ authController.js
  - | └─ wallpaperController.js
- └─ middleware
  - | └─ auth.js
  - | └─ upload.js
  - | └─ verifyToken.js
- └─ models
  - | └─ user.js
  - | └─ wallpaper.js
- └─ profile
  - | └─ photo-1752291099237.jpg
  - | └─ photo-1752291496353.jpeg
- └─ routes
  - | └─ auth.js
  - | └─ download.js
  - | └─ wallpapers.js
- └─ sql
  - | └─ wallpaper\_site (1).sql
- └─ uploads
  - | └─ resized-image-1752291701010.jpg
  - | └─ resized-image-1752292830347.jpeg
- └─ .env
- └─ package-lock.json
- └─ package.json
- └─ [server.js](#)

## B. Bahasa Pemrograman dan Tools yang dipakai

Backend dari aplikasi ini Menggunakan FrameWork Express JS dan untuk Bagian Front End Menggunakan Next JS + Tailwind CSS + Daisy UI dan Database nya menggunakan MYSQL

## C. Penjelasan Masing-Masing Folder dan isi File

- a. penjelasan folder configs ini berfungsi sebagai menyimpan file koneksi Database MYSQL isi dari folder configs:

Gambar Code :

```
1 // configs/db.js
2 const { Sequelize } = require("sequelize");
3 require("dotenv").config();
4
5 const sequelize = new Sequelize(
6   process.env.DB_NAME,
7   process.env.DB_USER,
8   process.env.DB_PASSWORD,
9   {
10     host: process.env.DB_HOST,
11     port: process.env.DB_PORT,
12     dialect: "mysql",
13     logging: false,
14   }
15 );
16
17 module.exports = sequelize;
```

- b. Selanjutnya penjelasan folder controllers yang memiliki 2 file di dalam nya yaitu [authController.js](#) dan [wallpaperController.js](#) kedua file memiliki fungsi nya masing masing auth berfungsi sebagai mengatur fungsi autentikasi yaitu Register dan Login sedangkan wallpaperController berfungsi yang mengatur data lebih tepat nya melakukan aksi CRUD (Create Read Update Delete)

Potongan Gambar Code:

[authController.js](#)

```
authController.js
Backend > controllers > authController.js > ...
1 const bcrypt = require("bcryptjs");
2 const jwt = require("jsonwebtoken");
3 const User = require("../models/user");
4
5 // REGISTER
6 exports.register = async (req, res) => {
7   try {
8     console.log("REGISTER req.body:", req.body);
9     console.log("REGISTER req.file:", req.file);
10
11     const { username, email, password, confirmPassword, role } = req.body;
12
13     // Cek semua field
14     if (!username || !email || !password || !confirmPassword || !role) {
15       console.warn("Field tidak lengkap:", req.body);
16       return res.status(400).json({ message: "Semua field wajib diisi." });
17     }
18
19     // Cek panjang password
20     if (password.length < 6 || password.length > 8) {
21       console.warn("Password tidak memenuhi panjang minimum/maksimum.");
22       return res.status(400).json({ message: "Password harus 6-8 karakter." });
23     }
24
25     if (password !== confirmPassword) {
26       console.warn("Password & Konfirmasi tidak sama.");
27       return res
28         .status(400)
29         .json({ message: "Password dan konfirmasi tidak sama." });
30     }
31
32     const existing = await User.findOne({ where: { email } });
33     if (existing) {
34       console.warn("Email sudah terdaftar:", email);
35       return res.status(400).json({ message: "Email sudah terdaftar." });
36     }
37   } catch (error) {
38     console.error(error);
39     return res.status(500).json({ message: "Terjadi kesalahan server." });
40   }
41
42   // Register successful
43   const hashedPassword = await bcrypt.hash(password, 10);
44   const newUser = await User.create({
45     username,
46     email,
47     password: hashedPassword,
48     confirmPassword,
49     role,
50   });
51
52   const token = jwt.sign({ id: newUser.id, role: newUser.role }, process.env.JWT_SECRET, {
53     expiresIn: process.env.JWT_EXPIRES_IN,
54   });
55
56   return res.status(201).json({
57     message: "User registered successfully.",
58     token,
59     user: {
60       id: newUser.id,
61       username: newUser.username,
62       email: newUser.email,
63       role: newUser.role,
64     },
65   });
66 }
```

[wallpaperController.js](#)

```

wallpaperController.js A X
Backend > controllers > wallpaperController.js > ...
1  const path = require("path");
2  const sharp = require("sharp");
3  const fs = require("fs");
4  const Wallpaper = require("../models/wallpaper");
5
6  // ✅ Pastikan controller termuat
7  console.log("✅ wallpaperController.js TERLOAD");
8
9  // ✅ Tambah Wallpaper
Windsurf: Refactor | Explain | X
10 exports.addWallpaper = async (req, res) => {
11   console.log("🔥 Controller addWallpaper dipanggil");
12
13   try {
14     const { uploader, name, description } = req.body;
15     const file = req.file;
16
17     // 🕒 Log detail file
18     console.log("📁 File diterima:", file);
19
20     if (!file) {
21       console.warn("⚠️ Tidak ada file dikirim!");
22       return res.status(400).json({ error: "File gambar wajib diupload" });
23     }
24
25     // Cek apakah file ada secara fisik
26     if (!fs.existsSync(file.path)) {
27       console.error("❌ File tidak ditemukan:", file.path);
28       return res.status(500).json({ error: "File upload gagal disimpan" });
29     }
30

```

- c. Folder middleware berisi fungsi-fungsi yang dijalankan sebelum permintaan diproses oleh route handler. File `auth.js` digunakan untuk memvalidasi login user dan hak akses berdasarkan role (user, admin, atau uploader). File `upload.js` mengatur proses upload gambar, memastikan file yang diunggah sesuai aturan (jenis file, ukuran, dan folder tujuan). Sedangkan file `verifytoken.js` memverifikasi apakah token JWT yang dikirim oleh user valid, sehingga hanya user yang sudah login yang bisa mengakses fitur tertentu.

Gambar Potongan Code:

[auth.js](#)

```

auth.js A X
Backend > middleware > auth.js > ...
1  const jwt = require("jsonwebtoken");
2
3  // hanya untuk user yang sudah login
Windsurf: Refactor | Explain | X
4  exports.verifyToken = (req, res, next) => {
5    const authHeader = req.headers["authorization"];
6    const token = authHeader && authHeader.split(" ")[1]; // Bearer <token>
7
8    if (!token) {
9      return res
10       .status(401)
11       .json({ message: "Akses ditolak. Token tidak ditemukan." });
12    }
13
14    try {
15      const decoded = jwt.verify(token, process.env.JWT_SECRET);
16      req.user = decoded; // { id, role }
17      next();
18    } catch (err) {
19      console.error(err);
20      res.status(403).json({ message: "Token tidak valid." });
21    }
22  };
23

```

[upload.js](#)

```

upload.js A x
Backend > middleware > upload.js > ...
1  const multer = require("multer");
2  const path = require("path");
3  const fs = require("fs");
4
5  // 📁 Folder tujuan: profile (berdiri sendiri di root backend)
6  const uploadDir = path.join(__dirname, "..", "profile");
7
8  if (!fs.existsSync(uploadDir)) {
9      fs.mkdirSync(uploadDir, { recursive: true });
10     console.log("📁 Folder profile dibuat di:", uploadDir);
11 } else {
12     console.log("📁 Folder profile sudah ada di:", uploadDir);
13 }
14
15 // 📁 Konfigurasi penyimpanan file
16 const storage = multer.diskStorage({
17     // Windsurf: Refactor | Explain | Generate JSDoc | x
18     destination: (req, file, cb) => {
19         cb(null, uploadDir);
20     },
21     // Windsurf: Refactor | Explain | Generate JSDoc | x
22     filename: (req, file, cb) => {
23         const ext = path.extname(file.originalname).toLowerCase();
24         const filename = `${file.fieldname}-${Date.now()}${ext}`;
25         cb(null, filename);
26     },
27 });

```

## [verifytoken.js](#)

```

verifyToken.js A x
Backend > middleware > verifyToken.js > ...
1  const jwt = require("jsonwebtoken");
2
3  // Windsurf: Refactor | Explain | Generate JSDoc | x
4  module.exports = function (req, res, next) {
5      const authHeader = req.headers.authorization;
6
7      if (!authHeader || !authHeader.startsWith("Bearer ")) {
8          return res.status(401).json({ message: "Akses ditolak. Token tidak ada." });
9      }
10
11     const token = authHeader.split(" ")[1];
12
13     try {
14         const decoded = jwt.verify(token, process.env.JWT_SECRET);
15         req.user = decoded; // menyimpan payload ke req.user
16         next();
17     } catch (err) {
18         return res.status(403).json({ message: "Token tidak valid." });
19     }
20 };

```

- d. Folder models berisi definisi skema tabel database dengan Sequelize. File user.js mendefinisikan model User untuk menyimpan data user ke tabel users. File wallpaper.js mendefinisikan model Wallpaper untuk menyimpan data wallpaper ke tabel wallpapers. Kedua model ini mempermudah proses pengolahan data pada database melalui kode program.

Gambar Potongan Code:

[user.js](#)

```
user.js A X
Backend > models > user.js > [0] User
1  const { DataTypes } = require("sequelize");
2  const sequelize = require("../configs/db");
3
4  const User = sequelize.define(
5    "User",
6    {
7      id: {
8        type: DataTypes.INTEGER,
9        primaryKey: true,
10       autoIncrement: true,
11      },
12      username: {
13        type: DataTypes.STRING,
14        allowNull: false,
15        unique: true,
16        validate: {
17          notEmpty: { msg: "Username tidak boleh kosong" },
18        },
19      },
20      email: {
21        type: DataTypes.STRING,
```

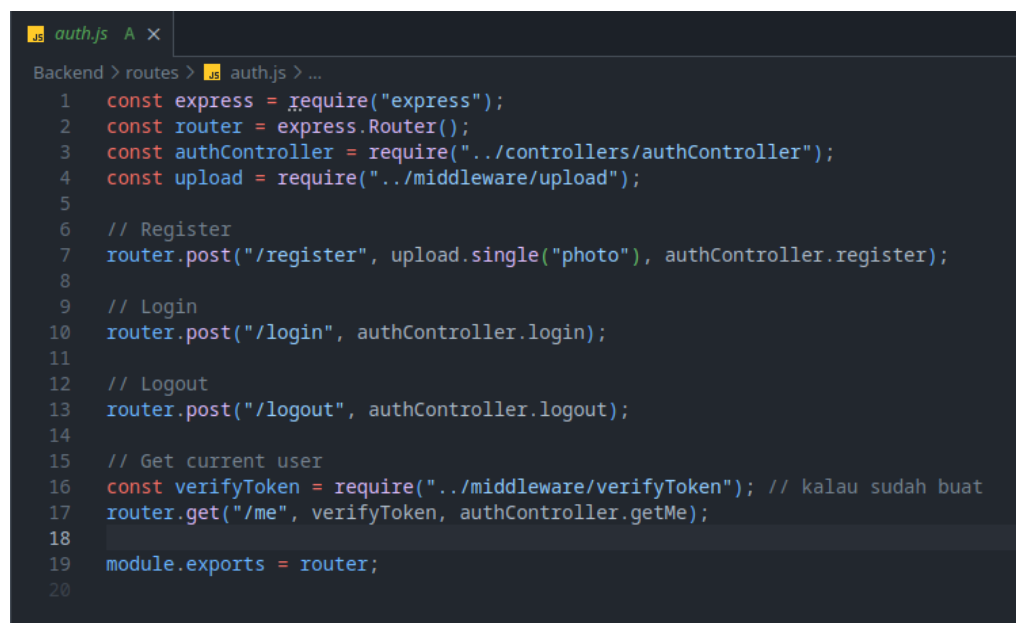
[wallpaper.js](#)

```
wallpaper.js A X
Backend > models > wallpaper.js > ...
1  // models/wallpaper.js
2  const { DataTypes } = require("sequelize");
3  const sequelize = require("../configs/db");
4
5  const Wallpaper = sequelize.define(
6    "Wallpaper",
7    {
8      id: {
9        type: DataTypes.INTEGER,
10       primaryKey: true,
11       autoIncrement: true,
12      },
13      uploader: {
14        type: DataTypes.STRING(100),
15        allowNull: false,
16      },
17      name: {
18        type: DataTypes.STRING(150),
19        allowNull: false,
20      },
21      description: {
```

- e. Selanjutnya Folder profile yang berfungsi sebagai penampung photo profile dari para pengguna yaitu admin dan users
- f. Selanjutnya, pada folder **routes** terdapat tiga buah file utama yaitu auth.js, wallpaper.js, dan download.js. File auth.js berfungsi untuk menangani rute-rute yang berkaitan dengan proses autentikasi, seperti endpoint /login untuk login, /register untuk mendaftarkan user baru, dan /logout untuk logout. Kemudian, file wallpaper.js digunakan untuk menangani rute-rute CRUD pada data wallpaper, di antaranya endpoint untuk mendapatkan semua data wallpaper, mendapatkan detail wallpaper berdasarkan ID, menambahkan wallpaper baru, mengedit, dan menghapus wallpaper. Sedangkan file download.js berfungsi untuk menangani proses pengunduhan file wallpaper melalui endpoint /download/:filename, sehingga user dapat mengunduh wallpaper yang diinginkan. Semua rute ini diatur secara terpisah agar kode lebih rapi, terstruktur, dan mudah dipelihara.

Gambar Potongan Code:

[auth.js](#)



```
auth.js A x
Backend > routes > auth.js > ...
1  const express = require("express");
2  const router = express.Router();
3  const authController = require("../controllers/authController");
4  const upload = require("../middleware/upload");
5
6  // Register
7  router.post("/register", upload.single("photo"), authController.register);
8
9  // Login
10 router.post("/login", authController.login);
11
12 // Logout
13 router.post("/logout", authController.logout);
14
15 // Get current user
16 const verifyToken = require("../middleware/verifyToken"); // kalau sudah buat
17 router.get("/me", verifyToken, authController.getMe);
18
19 module.exports = router;
20
```

[wallpaper.js](#)

```
JS wallpapers.js A X
Backend > routes > JS wallpapers.js > ...
1  const express = require("express");
2  const router = express.Router();
3
4  // ✓ Import multer dan langsung gunakan `.single("image")`
5  const uploadMiddleware = require("../middleware/upload");
6
7  const {
8    addWallpaper,
9    updateWallpaper,
10   deleteWallpaper
11 } = require("../controllers/wallpaperController");
12
13 const Wallpaper = require("../models/wallpaper");
14
15 // ✓ Ambil semua wallpaper
16 router.get("/", async (req, res) => {
17   try {
18     const wallpapers = await Wallpaper.findAll({
19       order: [["id", "DESC"]],
20     });
21     res.json(wallpapers);
22   } catch (err) {
23     console.error(err);
24   }
25 });
```

## [download.js](#)

```
JS download.js A X
Backend > routes > JS download.js > ...
1  const express = require("express");
2  const path = require("path");
3  const router = express.Router();
4
5  router.get("/:filename", (req, res) => {
6    const filename = req.params.filename;
7    const filepath = path.join(__dirname, "..", "uploads", filename);
8
9    res.download(filepath, filename, (err) => {
10      if (err) {
11        console.error("Download error:", err);
12        res.status(404).send("File tidak ditemukan");
13      }
14    });
15  });
16
17 module.exports = router;
18
```



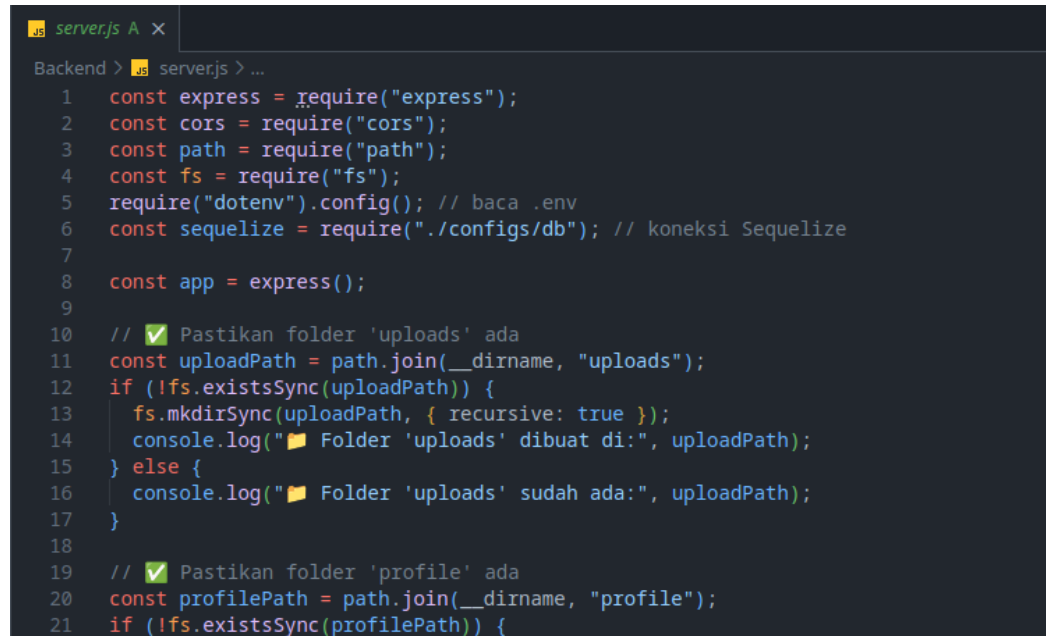
- g. Selanjutnya folder sql yang menyimpan query database website wallpaper
- h. Selanjutnya folder uploads yang berfungsi menyimpan semua gambar wallpaper yang di upload oleh user dan admin
- i. Selanjutnya terdapat file **.env** yang berfungsi sebagai tempat penyimpanan variabel-variabel environment untuk konfigurasi aplikasi. Di dalam file ini biasanya berisi informasi penting seperti nama database, username dan password database, host dan port database, serta JWT\_SECRET yang digunakan untuk mengenkripsi dan memverifikasi token JWT. Dengan adanya file .env, informasi konfigurasi yang bersifat rahasia dapat dipisahkan dari kode program sehingga lebih aman dan mudah diubah tanpa perlu mengubah kode.
- j. Selain itu, terdapat file **package-lock.json** yang secara otomatis dihasilkan ketika menjalankan perintah npm install. File ini berfungsi untuk mengunci versi pasti dari semua dependensi (dan dependensi turunannya) yang terpasang di proyek, sesuai dengan saat instalasi. Tujuannya adalah untuk memastikan bahwa setiap orang yang menginstal proyek ini akan mendapatkan versi paket yang sama persis, sehingga menghindari masalah ketidakcocokan versi di lingkungan yang berbeda. File ini bekerja berdampingan dengan package.json dan tidak perlu diubah secara manual.
- k. Selanjutnya ada file **package.json** yang berfungsi untuk menyimpan informasi tentang proyek beserta daftar dependensi yang digunakan. File ini juga mencatat metadata proyek seperti nama, versi, deskripsi, skrip yang bisa dijalankan (misalnya start, dev), serta dependensi dan devDependencies yang telah diinstal. File ini sangat penting karena digunakan oleh npm untuk mengatur, menginstal, dan membagikan proyek beserta semua paket yang dibutuhkan.
- l. Selanjutnya ada file **server.js** yang berperan sebagai pusat atau otak dari backend. File ini bertanggung jawab untuk menjalankan server Express, menghubungkan ke database, serta mengatur penggunaan semua route, middleware, dan konfigurasi lainnya yang sudah dibuat. Semua file yang

sudah disiapkan (seperti routes, middleware, dan models) di-*import* dan digunakan di sini sehingga server bisa berjalan dan melayani permintaan dari client.

Gambar

Potongan

Code:

A screenshot of a code editor window titled 'server.js A x'. The editor shows a JavaScript file named 'server.js' with the following code:

```
Backend > .js server.js > ...
1  const express = require("express");
2  const cors = require("cors");
3  const path = require("path");
4  const fs = require("fs");
5  require("dotenv").config(); // baca .env
6  const sequelize = require("../configs/db"); // koneksi Sequelize
7
8  const app = express();
9
10 // ✅ Pastikan folder 'uploads' ada
11 const uploadPath = path.join(__dirname, "uploads");
12 if (!fs.existsSync(uploadPath)) {
13   fs.mkdirSync(uploadPath, { recursive: true });
14   console.log("📁 Folder 'uploads' dibuat di:", uploadPath);
15 } else {
16   console.log("📁 Folder 'uploads' sudah ada:", uploadPath);
17 }
18
19 // ✅ Pastikan folder 'profile' ada
20 const profilePath = path.join(__dirname, "profile");
21 if (!fs.existsSync(profilePath)) {
```

## D. Hasil CRUD dan AUTENTIKASI

### 1. Tampilan Form Create atau Tambah data

#### a. Tampilan mengisi form tambah data

### Form Tambah Wallpaper

Silakan isi form berikut untuk menambahkan wallpaper baru.

**Nama Uploader**

Jery

**Nama Wallpaper**

Tikus Putih

**Deskripsi**

Tikus Putih

**Resolusi**


1920x1080

**Gambar**

CHOOSE FILE FAUNA-Tikus-Putih-Teman-Peneliti-Bereksperimen.jpg

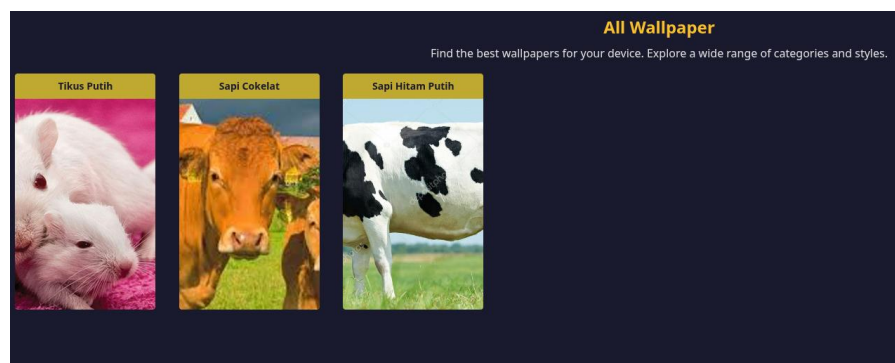
\*Gambar harus berformat .jpg atau .png

Preview Gambar:



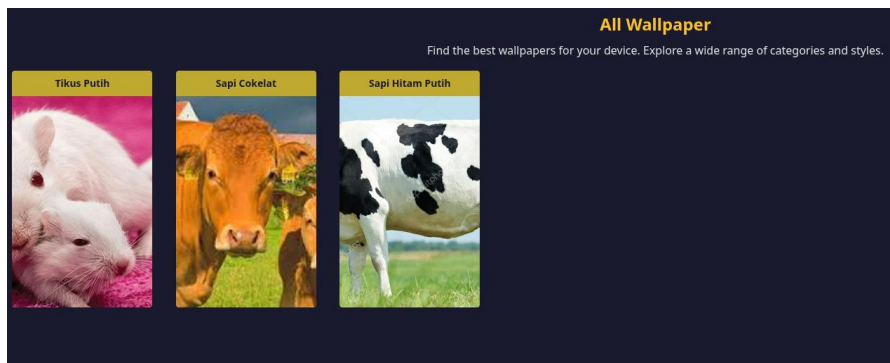
KIRIM

#### b. Tampilan Berhasil Bertambah

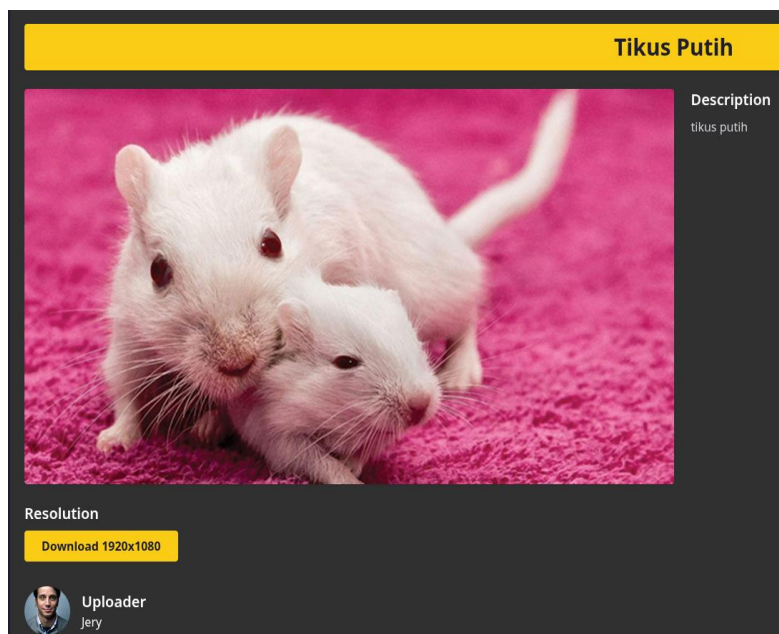


## 2. Tampilan Get Data atau Read data

### a. Get all data



### b. get by id



## 1. Edit atau Update data

### a. tampilan sebelum di edit

### Edit Wallpaper: Tikus Putih

Update informasi wallpaper di bawah ini.

Nama Uploader

Jery

Nama Wallpaper

Tikus Putih

Deskripsi

tikus putih

Resolusi

1920x1080


Gambar

CHOOSE FILE

No file chosen

\*Gambar harus berformat .jpg atau .jpeg

Preview Gambar:



SIMPAN PERUBAHAN

tampilan mengisi form data edit

### Edit Wallpaper: Tikus Putih

Update informasi wallpaper di bawah ini.

Nama Uploader

Bagus

Nama Wallpaper

Tikus Putih Dengan Background Putih

Deskripsi

Tikus putih, umumnya dikenal sebagai tikus albino, adalah jenis tikus (biasanya *Rattus norvegicus*) yang tidak memiliki pigmen warna pada kulit dan rambutnya, sehingga tampak berwarna putih atau merah muda. Tikus putih sering digunakan sebagai hewan percobaan dalam penelitian ilmiah karena beberapa alasan, termasuk kemudahan pemeliharaan, siklus reproduksi cepat, dan kemiripan fisiologis dengan manusia.

Resolusi

1920x1080

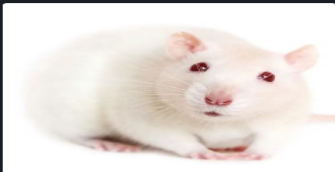
Gambar

CHOOSE FILE

Tikus-Putih.jpg

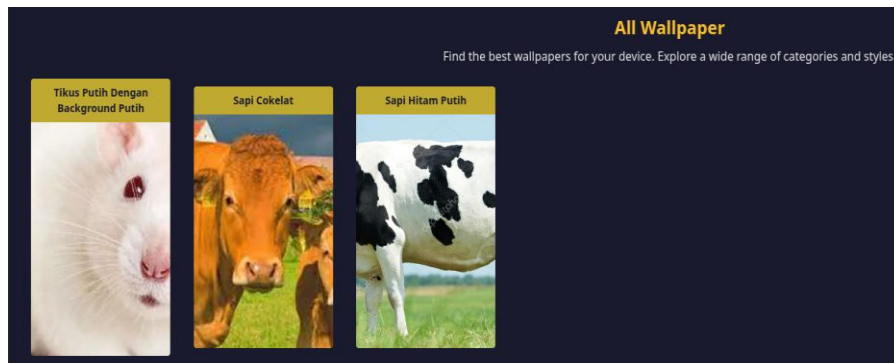
\*Gambar harus berformat .jpg atau .jpeg

Preview Gambar:



SIMPAN PERUBAHAN

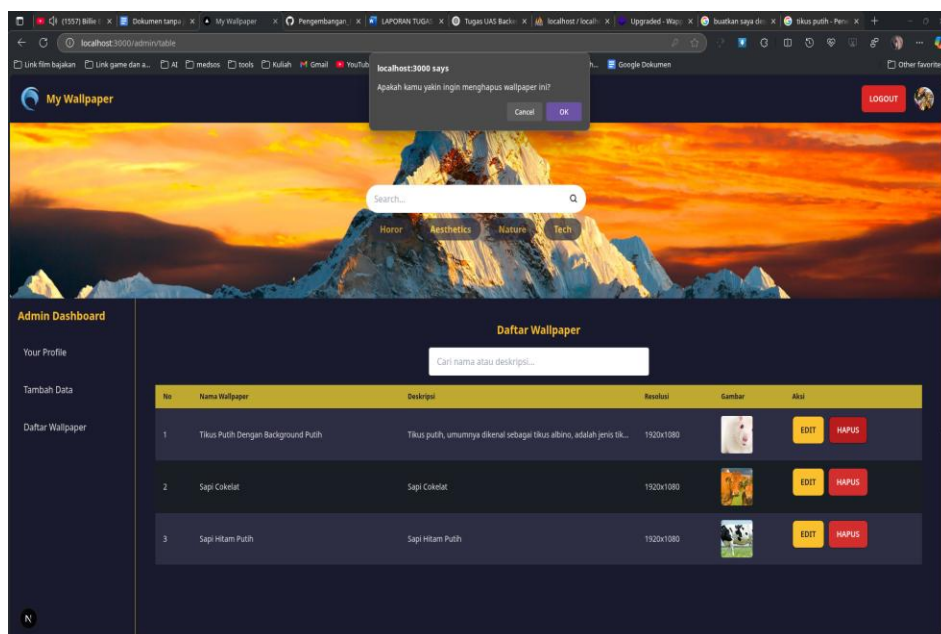
b. tampilan setelah berhasil di edit



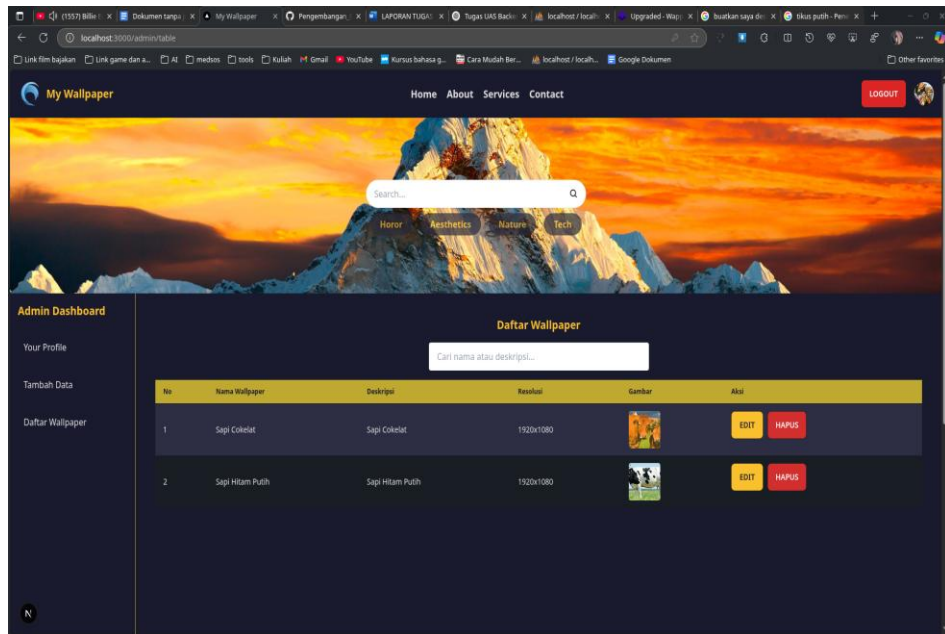
### c. tampilan setelah berhasil di edit bagian detail



### d. tampilan Delete



#### e. Tampilan Setelah Berhasil di delete



### 4. tampilan registrasi

#### a. isi form registrasi

### Register

Buat akun baru untuk mulai menjelajahi wallpaper.

Username

Email

Password

\* Password harus 6-8 karakter

Konfirmasi Password

Role

Admin

Foto Profil

Choose File 1-intro-photo-final.jpg

Preview Foto:

Register

**b. setelah berhasil daftar dan masuk ke tabel user**

		id	username	email	password	role	photo	createdAt	updatedAt
		1	bless	budi@gmail.com	\$2b\$10\$Rpv6NSIPOQCADNzhNnykP.1nULQEf75qQp9OuQCdVS...	admin	photo-1752291496353.jpeg	2025-07-12 03:38:16	2025-07-12 03:38:16
		3	badrul	badrul@gmail.com	\$2b\$10\$DsPE.xLPbR/s5Gnni8lv.s75EOdYgc3m2ldyNoW/...	admin	photo-1752372856291.jpg	2025-07-13 02:14:16	2025-07-13 02:14:16

**5. tampilanLogin**

**a. tampilan isi form login**

## Login

Email

badrul@gmail.com

Password

.....

\* Password harus 6-8 karakter

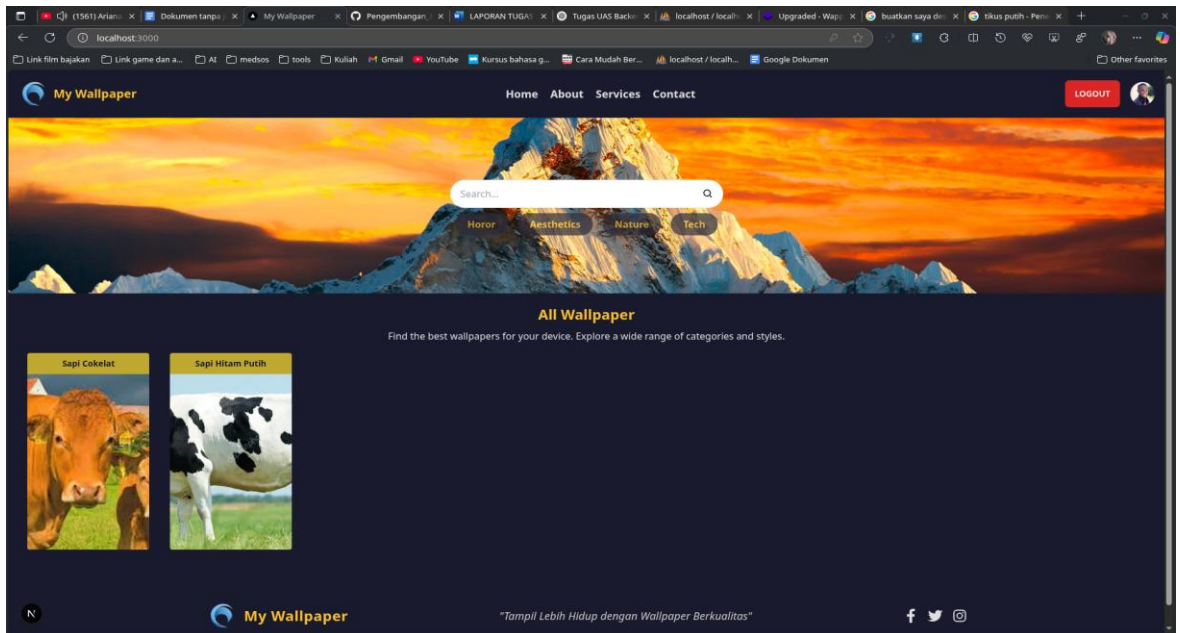
Login

[Lupa password?](#)

Belum punya akun? [Daftar di sini](#)



## b. tampilan berhasil login



## c. tampilan di admin dashboard

