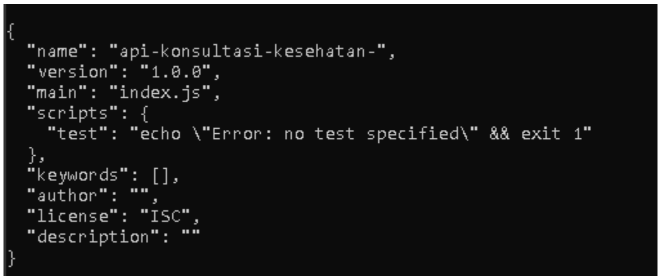
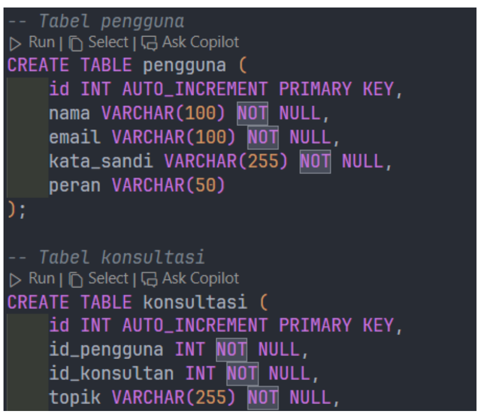
DOKUMENTASI

1. Instalasi ntuk membangun dan menjalankan server backend.

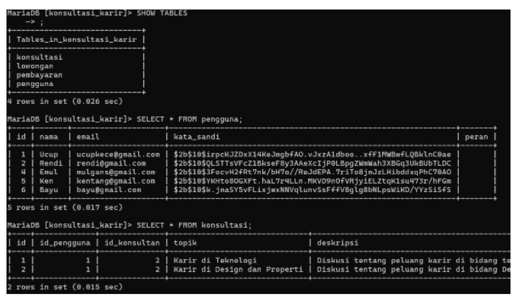


```
{
  "name": "api-konsultasi-kesehatan-",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

2. Membuat Database konsultasi_karir

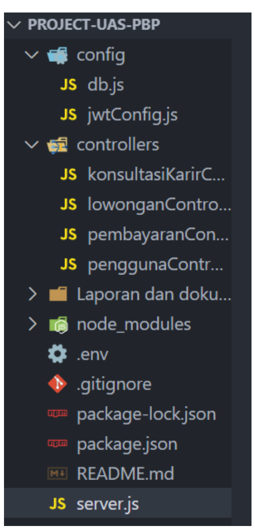


```
-- Tabel pengguna
▷ Run | Select | Ask Copilot
CREATE TABLE pengguna (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    kata_sandi VARCHAR(255) NOT NULL,
    peran VARCHAR(50)
);

-- Tabel konsultasi
▷ Run | Select | Ask Copilot
CREATE TABLE konsultasi (
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_pengguna INT NOT NULL,
    id_konsultan INT NOT NULL,
    topik VARCHAR(255) NOT NULL,
```



```
MariaDB [konsultasi_karir]> SHOW TABLES
    -> ;
+----------------------------+
| Tables_in_konsultasi_karir |
+----------------------------+
| konsultasi                 |
| lowongan                   |
| pembayaran                 |
| pengguna                   |
+----------------------------+
4 rows in set (0.026 sec)

MariaDB [konsultasi_karir]> SELECT * FROM pengguna;
+----+-------+--------------------+--------------------------------------------------------------+-------+
| id | nama  | email              | kata_sandi                                                   | peran |
+----+-------+--------------------+--------------------------------------------------------------+-------+
|  1 | Ucup  | ucupkece@gmail.com | $2b$10$irpcKJZDxX14KeJmgbfAO.vJxzA1dboo..xfF1MWBwfLQBklnC0ae |       |
|  2 | Rendi | rendi@gmail.com    | $2b$10$QLSTTsVFcZ1BkseF8y3AAeXcIjP0LBpgZWmWah3XBGq3UkBUbTLDC |       |
|  4 | Emul  | mulgans@gmail.com  | $2b$10$3FocvH2fRt7nk/bH7o//ReJdEPA.7riTo8jmJzLHibddxqPhC70AO |       |
|  5 | Ken   | kentang@gmail.com  | $2b$10$YKHto8OGXFt.haL7r4LLn.MKVD9nOfVRjyiELZtqK1su473r/hFGm |       |
|  6 | Bayu  | bayu@gmail.com     | $2b$10$k.jnaSY5vFLixjwxNNVqlunvSsFffVBglg8bNLpsWiKD/YYzSiSfS |       |
+----+-------+--------------------+--------------------------------------------------------------+-------+
5 rows in set (0.017 sec)

MariaDB [konsultasi_karir]> SELECT * FROM konsultasi;
+----+-------------+--------------+------------------------------+----------------------------------------
| id | id_pengguna | id_konsultan | topik                        | deskripsi
+----+-------------+--------------+------------------------------+----------------------------------------
|  1 |           1 |            2 | Karir di Teknologi           | Diskusi tentang peluang karir di bidang te
|  2 |           1 |            2 | Karir di Design dan Properti | Diskusi tentang peluang karir di bidang De
+----+-------------+--------------+------------------------------+----------------------------------------
2 rows in set (0.015 sec)
```

3. Struktur Direktori Proyek



Implementasi Rest API

1. Konfigurasi Koneksi Database

```
config > JS db.js > ...
const mysql = require('mysql2');  782.8k (gzipped: 345.6k)

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'konsultasi_karir' // Mengubah nama database agar sesuai dengan tema baru
});

connection.connect((err) => {
  if (err) {
    console.error('Kesalahan koneksi ke database: ', err);
    return;
  }
  console.log('Koneksi ke database jasa pencari lowongan pekerjaan dan konsultasi bimbingan pra kerja berhasil!');
});

module.exports = connection;
```

2. Setup Server

```
JS server.js > ...
const express = require('express');
const bodyParser = require('body-parser');  487.5k (gzipped: 212.1k)
const penggunaController = require('./controllers/penggunaController');
const lowonganController = require('./controllers/lowonganController');
const konsultasiKarirController = require('./controllers/konsultasiKarirController');
const pembayaranController = require('./controllers/pembayaranController');
require("dotenv").config();  6.3k (gzipped: 2.8k)

const app = express();
const PORT = 5000;

app.use(bodyParser.json());

app.use('/api', penggunaController);
app.use('/api', lowonganController);
app.use('/api', konsultasiKarirController);
app.use('/api', pembayaranController);

app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT} untuk layanan pencarian kerja dan bimbingan karir`);
});

//Kelompok 1
```

3. Membuat Rute dan Operasi CRUID

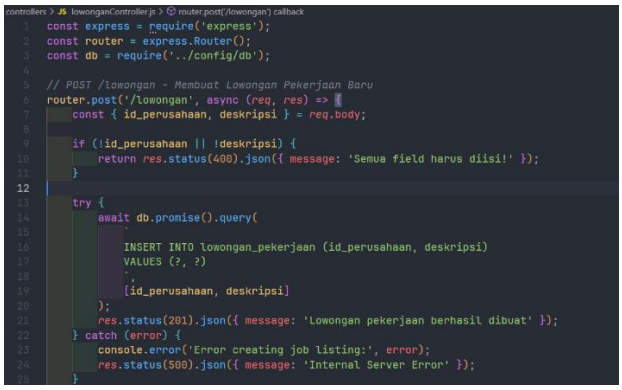
a. penggunaController.js

```
controllers > JS penggunaController.js > router.get('/pengguna') callback
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Gunakan db dari config.js
const bcrypt = require('bcrypt'); Calculating...
const jwt = require('jsonwebtoken');  53.2k (gzipped: 15.9k)
require('dotenv').config(); // Pastikan menggunakan environment variable  6.3k (gzipped: 2.8k)

const SECRET_KEY = process.env.SECRET_KEY; // Gunakan SECRET_KEY dari environment variable

// Registrasi pengguna untuk platform pencari lowongan kerja dan konsultasi pra-kerja
router.post('/register', async (req, res) => {
    const { nama, email, kata_sandi, peran } = req.body;

    if (!nama || !email || !kata_sandi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        const [existingUser] = await db.promise().query('SELECT * FROM pengguna WHERE email = ?', [email]);
        if (existingUser.length > 0) {
            return res.status(400).json({ message: 'Email sudah digunakan!' });
        }

        const hashedPassword = await bcrypt.hash(kata_sandi, 10);

        await db.promise().query('INSERT INTO pengguna (nama, email, kata_sandi, peran) VALUES (?, ?, ?, ?)', [
            nama,
            email,
            hashedPassword,
            peran || 'pelamar'
        ]);
```

b. konsultasiKarirController.js

```
controllers > JS konsultasiKarirController.js > ...
const express = require('express');
const router = express.Router();
const db = require('../config/db');

// POST /konsultasiKarir - Membuat janji konsultasi bimbingan pra kerja baru
router.post('/karir', async (req, res) => {
    const { id_pelamar, id_mentor, jadwal_konsultasi } = req.body;

    if (!id_pelamar || !id_mentor || !jadwal_konsultasi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        await db.promise().query(
            `
            INSERT INTO janji_konsultasi (id_pelamar, id_mentor, jadwal_konsultasi)
            VALUES (?, ?, ?)
            `,
            [id_pelamar, id_mentor, jadwal_konsultasi]
        );
        res.status(201).json({ message: 'Janji konsultasi bimbingan pra kerja berhasil dibuat' });
    } catch (error) {
        console.error('Error creating appointment:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
});

// GET /konsultasiKarir - Mengambil daftar janji konsultasi untuk pelamar tertentu
router.get('/karir', async (req, res) => {
    const { id_pelamar } = req.query;
```

c. lowonganController.js



```
controllers > JS lowonganController.js > router.post('/lowongan') callback
const express = require('express');
const router = express.Router();
const db = require('../config/db');

// POST /lowongan - Membuat Lowongan Pekerjaan Baru
router.post('/lowongan', async (req, res) => {
    const { id_perusahaan, deskripsi } = req.body;

    if (!id_perusahaan || !deskripsi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        await db.promise().query(
            `
            INSERT INTO lowongan_pekerjaan (id_perusahaan, deskripsi)
            VALUES (?, ?)
            `,
            [id_perusahaan, deskripsi]
        );
        res.status(201).json({ message: 'Lowongan pekerjaan berhasil dibuat' });
    } catch (error) {
        console.error('Error creating job listing:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
```

d. pembayaranController.js

```
controllers > JS pembayaranController.js > router.post('/pembayaran') callback
const express = require('express');
const router = express.Router();
const db = require('../config/db');

// POST /pembayaran - Membuat pembayaran untuk layanan bimbingan pra kerja
router.post('/pembayaran', async (req, res) => {
    const { id_konsultasi_karir, jumlah, status } = req.body;

    if (!id_konsultasi_karir || !jumlah) {
        return res.status(400).json({ message: 'Field id_konsultasi_karir dan jumlah harus diisi!' });
    }

    try {
        await db.promise().query(
            `
            INSERT INTO pembayaran (id_konsultasi_karir, jumlah, status)
            VALUES (?, ?, ?)
            `,
            [id_konsultasi_karir, jumlah, status || 'tertunda']
        );
        res.status(201).json({ message: 'Pembayaran untuk konsultasi bimbingan pra kerja berhasil dibuat' });
    } catch (error) {
        console.error('Error creating pembayaran:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
});

// GET /pembayaran - Mengambil semua pembayaran layanan bimbingan pra kerja
router.get('/pembayaran', async (req, res) => {
    try {
        const [results] = await db.promise().query(
```

4. Pengujian API