

1. Apa peran npm dalam pengembangan aplikasi Node.js?

- a. Membuat tampilan dari suatu website
- b. Menangani permintaan HTTP
- c. Menangani permintaan HTTPS
- x. Mengelola dependensi dan paket

2. Bagaimana cara membuat route endpoint GET dalam Express.js

dengan kaidah penulisan REST conventions yang tepat?

(Contoh untuk mengambil sekumpulan data dari produk)

- a. `app.get('/product, function)`
- b. `app.get('/get-products', function)`
- x. `app.get('/products', function)`
- d. `app.post('/products, function)`

3. Apa perbedaan antara PUT dan POST dalam konteks REST API?

- a. PUT digunakan untuk membuat data baru, sedangkan POST digunakan untuk mengupdate data.
- x. PUT digunakan untuk mengupdate data, sedangkan POST digunakan untuk membuat data baru.
- c. PUT dan POST keduanya dapat digunakan untuk membuat data baru.
- d. PUT dan POST Keduanya dapat digunakan untuk mengupdate data.

4. Bagaimana cara mengkonfigurasi middleware di Express.js? Diasumsikan app adalah variable yang dideklarasikan untuk express.

- x. `app.use(middlewareFunction)`
- b. `app.configure(middlewareFunction)`
- c. `middlewareFunction.use(app)`
- d. `middlewareFunction.configure(app)`

5. Apa itu middleware di Express.js dan bagaimana Anda dapat mengimplementasikannya untuk validasi data input sebelum menyimpannya ke database?

- a. Middleware digunakan hanya untuk menghubungkan aplikasi Express.js dengan server basis data.
- b. Middleware hanya dapat digunakan untuk manajemen session pengguna.
- x. Middleware adalah fungsi yang dapat memodifikasi objek request atau response.
- d. Middleware adalah metode standar untuk mengelola route di Express.js.

6. Berikut ini yang bukan untuk menangani autentikasi pengguna dalam pembuatan REST API adalah?

- x. OAuth
- b. Bcrypt
- c. JWT
- d. Semua Jawaban Salah

7. Berikut ini yang bukan digunakan untuk mengimplementasikan pengujian (testing) otomatis untuk REST API yang dibangun dengan Node.js adalah?

- x. Unit testing dengan JUnit dan Chai
- b. Pengujian integrasi dengan Supertest
- c. Unit testing dengan Mocha dan Chai
- d. Semua Jawaban Benar

8. Bagaimana Anda dapat mengoptimalkan kinerja aplikasi Node.js yang menangani lalu lintas HTTP yang tinggi? Jelaskan teknik dan strategi yang dapat digunakan.

- a. Gunakan teknik serverless computing untuk mengelola beban lalu lintas yang tinggi dan menjamin ketersediaan aplikasi.
- b. Hindari penggunaan clustering, karena dapat menyebabkan overhead yang tidak perlu, dan fokus pada pengoptimalan kode sumber.
- c. Tingkatkan penggunaan synchronous programming untuk memastikan konsistensi data, dan gunakan alat monitoring untuk mendeteksi bottleneck.
- x. Gunakan caching untuk hasil query yang sering digunakan dan pertimbangkan penggunaan reverse proxy.

9. Jelaskan konsep "JWT (JSON Web Token)" dan cara penggunaannya dalam otentikasi dan otorisasi aplikasi Node.js.

- a. JWT hanya dapat digunakan untuk otentikasi, dan tidak dapat digunakan untuk otorisasi. Penggunaannya melibatkan pertukaran token antara server dan klien.
- b. JWT adalah protokol otorisasi yang digunakan untuk memverifikasi identitas pengguna. Penggunaannya melibatkan pertukaran token antara server dan pengguna.
- c. JWT adalah mekanisme otentikasi yang menyediakan alur kerja otentikasi dalam server. Penggunaannya melibatkan penyimpanan token dalam cookie pada server maupun klien pengguna nya.
- x. JWT adalah format token terstruktur yang dapat digunakan untuk mentransmisikan informasi pengguna. Penggunaannya melibatkan pembuatan token dan verifikasi pada setiap permintaan.

10. Jelaskan perbedaan antara autentikasi dan otorisasi dalam konteks pembuatan REST API!

- a. Autentikasi dan otorisasi adalah konsep yang digunakan untuk menyimpan data dalam cache pada server.
- x. Autentikasi adalah proses verifikasi identitas pengguna, sementara otorisasi adalah proses memberikan hak akses berdasarkan identitas yang telah terverifikasi.
- c. Autentikasi adalah proses memberikan hak akses berdasarkan identitas pengguna, sementara otorisasi adalah proses verifikasi identitas.
- d. Autentikasi dan otorisasi adalah istilah yang dapat digunakan secara bergantian, karena keduanya memiliki makna yang serupa.

11. Bagaimana cara membuat tabel baru dengan ketentuan dua kolom dengan kolom pertama digunakan untuk menyimpan angka dan kolom kedua digunakan untuk menyimpan sebuah nama dalam MySQL?

- x. `CREATE TABLE new_table (column1 INT, column2 VARCHAR(255));`
- b. `CREATE new_table (column1 INT, column2 VARCHAR(255));`
- c. `CREATE new_table ADD COLUMN (column1 VARCHAR(255), column2 INT);`
- d. `CREATE TABLE new_table (column1 VARCHAR(255), column2 INT);`

12. Apa yang dimaksud dengan foreign key dalam SQL dan bagaimana cara menggunakannya?

- a. Foreign Key digunakan untuk mengurutkan data dalam suatu tabel berdasarkan kolom tertentu.
- b. Foreign Key digunakan untuk memberikan nilai default pada kolom-kolom tertentu.
- x. Foreign Key digunakan untuk menghubungkan dua tabel bersama berdasarkan kolom tertentu.
- d. Foreign Key digunakan untuk mengidentifikasi catatan unik dalam tabel.

13. Bagaimana cara menghapus semua data dari sebuah tabel di MySQL tanpa menghapus struktur tabel itu sendiri?

- a. `DELETE * FROM table_name;`
- b. `REMOVE FROM table_name;`
- x. `TRUNCATE table_name;`
- d. `DROP table_name;`

14. Jelaskan konsep normalisasi dalam desain basis data dan mengapa itu penting?

- x. Normalisasi adalah proses mengurangi redundansi dan dependensi data dalam basis data. Ini penting untuk memastikan data konsisten dan mengurangi risiko anomali.
- b. Normalisasi adalah proses mengelompokkan data dalam basis data berdasarkan tipe datanya. Ini penting untuk memastikan penyimpanan data yang efisien.
- c. Normalisasi adalah proses mengkonversi data menjadi bentuk yang lebih sederhana. Ini penting untuk memudahkan pembacaan data dalam basis data.
- d. Normalisasi tidak penting dalam desain basis data karena dapat meningkatkan kompleksitas query SQL.

15. Apa itu nested subquery dan bagaimana cara Anda dapat menggunakannya dalam SQL?

- a. Nested Subquery adalah subquery yang ditempatkan di dalam HAVING clause utama. Menggunakannya dengan menerapkan fungsi agregat pada hasil subquery.
- b. Nested Subquery adalah subquery yang ditempatkan di dalam SELECT clause utama. Menggunakannya dengan mengelompokkan hasil subquery untuk penggunaan dalam SELECT clause utama.
- x. Nested Subquery adalah subquery yang ditempatkan di dalam FROM clause utama. Menggunakannya dengan merujuk pada hasil subquery sebagai tabel sementara dalam FROM clause.
- d. Nested Subquery adalah subquery yang ditempatkan di dalam WHERE clause utama. Menggunakannya dengan menyaring hasil utama berdasarkan kriteria subquery.

16. Apa yang dimaksud dengan Docker Compose? Bagaimana cara menggunakannya dalam pengelolaan aplikasi yang kompleks?

- a. Docker Compose adalah layanan yang digunakan hanya untuk membuat dan mengelola jaringan pada Docker. Ini dapat digunakan untuk mengatur komunikasi antara kontainer.
- b. Docker Compose adalah perintah untuk mengkompilasi dan mengkonfigurasi Docker Image. Ini dapat digunakan untuk membangun dan menyimpan Docker Image.
- x. Docker Compose adalah layanan web untuk mengelola kontainer Docker. Ini dapat menyederhanakan implementasi dan orkestrasi aplikasi yang terdiri dari beberapa layanan.
- d. Docker Compose adalah alat untuk menyusun dan mengkompilasi kode sumber dalam aplikasi Docker. Ini dapat membantu dalam manajemen proyek pengembangan sebuah aplikasi secara terstruktur.

17. Bagaimana Docker mengelola volume dan apa manfaatnya dalam konteks persistensi data?

- a. Docker menggunakan volume untuk menyimpan snapshot dari kontainer. Manfaatnya adalah untuk membuat cadangan data.
- b. Tidak ada jawaban yang benar.
- x. Docker menggunakan volume untuk menyediakan ruang penyimpanan yang persisten di luar kontainer. Manfaatnya adalah untuk mempertahankan dan membagikan data di antara kontainer.
- d. Docker menggunakan volume untuk meningkatkan keamanan data di dalam kontainer. Manfaatnya adalah melindungi data dari akses yang tidak sah.

18. Apa yang dimaksud dengan mocking dalam konteks pengujian unit Express.js?

- a. Mocking adalah proses membuat salinan data yang identik untuk keperluan pengujian.
- b. Mocking adalah langkah dalam proses optimasi kinerja Express.js untuk meningkatkan waktu respon server.
- x. Mocking adalah teknik untuk menggantikan komponen-komponen eksternal dengan implementasi palsu untuk mengisolasi unit yang diuji.
- d. Mocking adalah proses membuat replika server Express.js untuk menggantikan server asli selama pengujian unit.

19. Apa perbedaan antara beforeEach dan before dalam pengujian unit dengan Jest pada Express.js?

- a. beforeEach digunakan untuk konfigurasi pengujian sebelumnya, sedangkan before digunakan untuk konfigurasi pengujian setelahnya.
- b. beforeEach dan before bersifat sama sehingga dapat digunakan secara bergantian.
- x. beforeEach digunakan untuk menjalankan suatu blok kode sebelum setiap tes dijalankan, sedangkan before menjalankan blok kode hanya sekali sebelum semua tes dijalankan.\
- d. before digunakan untuk menjalankan suatu blok kode sebelum setiap tes dijalankan, sedangkan beforeEach menjalankan blok kode hanya sekali sebelum semua tes dijalankan.

20. Jika kita berhasil menambahkan sebuah record baru ke database kita, HTTP response code mana yang paling tepat kita gunakan? Apabila kita tidak mendapatkan akses yang tepat dalam mengakses suatu request, response code apa yang harus diberikan?

- a. 200 dan 404
- b. 200 dan 401
- c. 201 dan 404
- x. 201 dan 401

21. Perhatikan kode TypeScript di atas yang menggunakan Express.js untuk routing dan middleware. Apa yang dilakukan oleh fungsi loggerMiddleware dalam aplikasi Express.js ini?



```
Index.ts

import express, { Request, Response, NextFunction } from 'express';

const app = express();
const port = 3000;

// Middleware
const loggerMiddleware = (req: Request, res: Response, next: NextFunction) => {
  console.log(`${new Date().toISOString()} ${req.method} ${req.url}`);
  next();
};

app.use(loggerMiddleware);

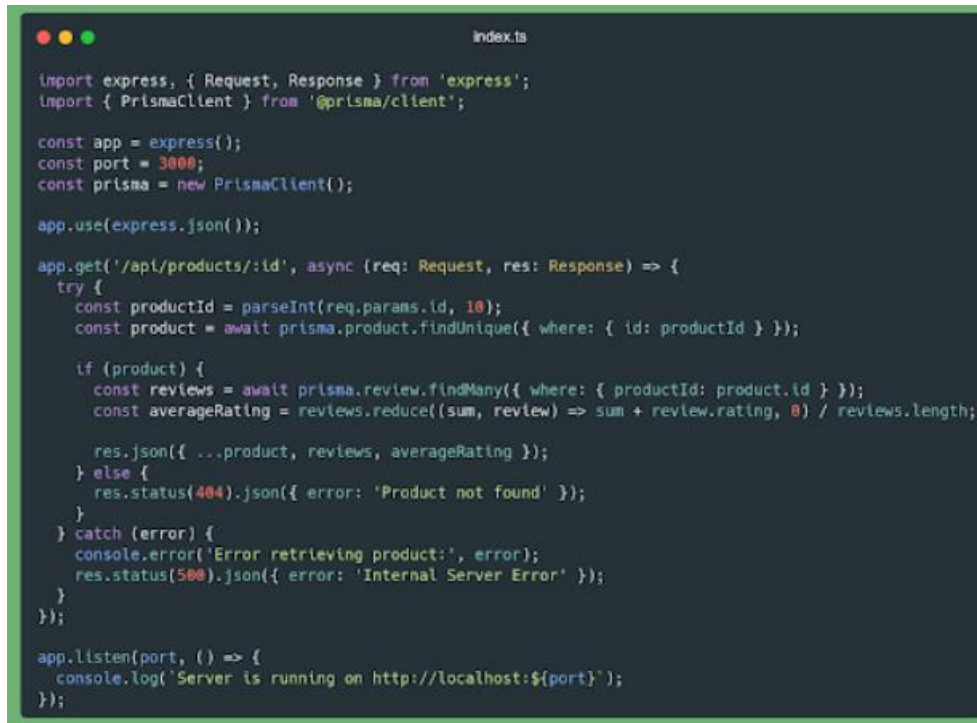
// Routes
app.get('/', (req: Request, res: Response) => {
  res.send('Hello, Express.js in TypeScript!');
});

app.get('/api/user/:id', (req: Request, res: Response) => {
  const userId = req.params.id;
  res.json({ userId, message: 'User details fetched successfully' });
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

- a. Mencatat timestamp, metode HTTP, dan URL setiap response keluar ke console.
- x. Mencatat timestamp, metode HTTP, dan URL setiap permintaan masuk ke console.
- c. Mencatat timestamp, metode HTTP, dan URL setiap permintaan masuk ke dalam file.
- d. Menangani otentikasi untuk protected route dengan menggunakan middleware.

22. Perhatikan kode TypeScript berikut yang menggunakan Express.js, TypeScript, dan PrismaJS untuk mengambil data dari tabel Products dan Reviews. Apa fungsi `parseInt(req.params.id, 10)`; pada baris `const productId = parseInt(req.params.id, 10)`; dan apa yang akan terjadi jika ada kesalahan dalam pengambilan data dari tabel Reviews?



```
index.ts

import express, { Request, Response } from 'express';
import { PrismaClient } from '@prisma/client';

const app = express();
const port = 3000;
const prisma = new PrismaClient();

app.use(express.json());

app.get('/api/products/:id', async (req: Request, res: Response) => {
  try {
    const productId = parseInt(req.params.id, 10);
    const product = await prisma.product.findUnique({ where: { id: productId } });

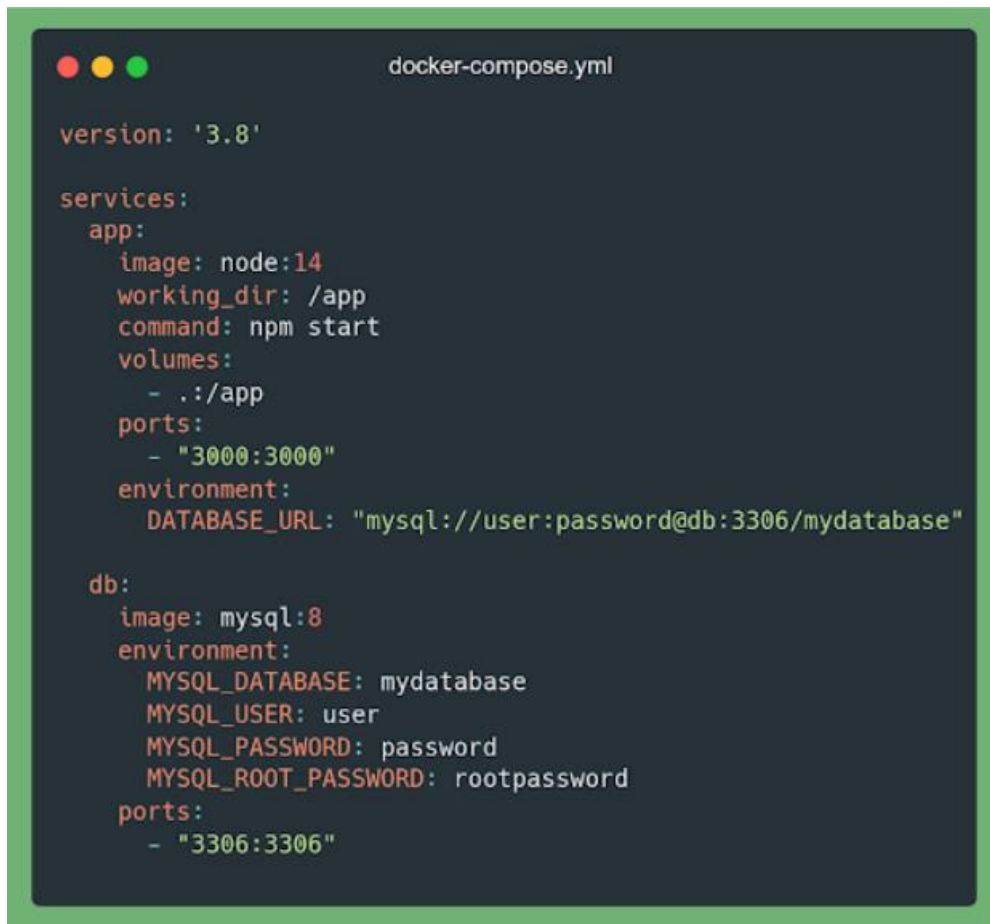
    if (product) {
      const reviews = await prisma.review.findMany({ where: { productId: product.id } });
      const averageRating = reviews.reduce((sum, review) => sum + review.rating, 0) / reviews.length;

      res.json({ ...product, reviews, averageRating });
    } else {
      res.status(404).json({ error: 'Product not found' });
    }
  } catch (error) {
    console.error('Error retrieving product:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

app.listen(port, () => {
  console.log('Server is running on http://localhost:${port}');
});
```

- a. Baris tersebut mengubah `req.params.id` menjadi bilangan bulat. Jika review tidak ditemukan maka akan menyebabkan kesalahan runtime dan crash.
- b. Baris tersebut menerapkan parsing desimal pada `req.params.id`. Jika review tidak ditemukan maka server akan memberikan respons dengan status kesalahan 500
- x. Baris tersebut mengubah `req.params.id` menjadi bilangan bulat. Jika review tidak ditemukan maka server akan memberikan respons dengan status kesalahan 500
- d. Baris tersebut menerapkan parsing desimal pada `req.params.id`. Jika review tidak ditemukan maka server akan memberikan respons dengan status 200

23. Services bernama “app” dalam file docker-compose.yml dimaksudkan untuk?



```
docker-compose.yml

version: '3.8'

services:
  app:
    image: node:14
    working_dir: /app
    command: npm start
    volumes:
      - ./app
    ports:
      - "3000:3000"
    environment:
      DATABASE_URL: "mysql://user:password@db:3306/mydatabase"

  db:
    image: mysql:8
    environment:
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: rootpassword
    ports:
      - "3306:3306"
```

- a. Layanan untuk mengelola proxy Nginx.
- b. Layanan untuk membuat images pada docker.
- x. Layanan untuk menjalankan aplikasi NodeJs
- d. Layanan untuk membuat network pada docker.

24. Berdasarkan ketiga tabel tersebut, pilihlah jawaban dibawah ini untuk mendapatkan nama-nama siswa yang memiliki nilai di atas rata-rata untuk semua kursus yang diikuti?

```
CREATE TABLE Students (  
  student_id INT PRIMARY KEY,  
  student_name VARCHAR(255) NOT NULL,  
  date_of_birth DATE NOT NULL  
);  
  
CREATE TABLE Courses (  
  course_id INT PRIMARY KEY,  
  course_name VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE Enrollments (  
  enrollment_id INT PRIMARY KEY,  
  student_id INT,  
  course_id INT,  
  grade DECIMAL(3, 2),  
  FOREIGN KEY (student_id) REFERENCES Students(student_id),  
  FOREIGN KEY (course_id) REFERENCES Courses(course_id)  
);
```

x. SELECT DISTINCT s.student_name FROM Students s INNER JOIN Enrollments e ON s.student_id = e.student_id INNER JOIN Courses c ON e.course_id = c.course_id WHERE e.grade > (SELECT AVG(grade) FROM Enrollments);

b. SELECT s.student_name FROM Students s INNER JOIN Enrollments e ON s.student_id = e.student_id INNER JOIN Courses c ON e.course_id = c.course_id GROUP BY s.student_name WHERE AVG(e.grade) > (SELECT AVG(grade) FROM Enrollments);

c. SELECT s.student_name FROM Students s INNER JOIN Enrollments e ON s.student_id = e.student_id INNER JOIN Courses c ON e.course_id = c.course_id WHERE e.grade > (SELECT AVG(grade) FROM Enrollments GROUP BY course_id);

d. SELECT s.student_name FROM Students s INNER JOIN Enrollments e ON s.student_id = e.student_id INNER JOIN Courses c ON e.course_id = c.course_id WHERE e.grade > (SELECT AVG(grade) FROM Enrollments GROUP BY student_id);

25. Berdasarkan ketiga tabel tersebut, bagaimana cara untuk menampilkan nama proyek, nama karyawan, dan total jam kerja yang dikerjakan oleh setiap karyawan pada proyek "Web Development". Urutkan hasil berdasarkan total jam kerja secara menurun.

```
CREATE TABLE Employees (  
  employee_id INT PRIMARY KEY,  
  employee_name VARCHAR(255) NOT NULL,  
  position VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Projects (  
  project_id INT PRIMARY KEY,  
  project_name VARCHAR(255) NOT NULL,  
  start_date DATE NOT NULL,  
  end_date DATE NOT NULL  
);  
  
CREATE TABLE Assignments (  
  assignment_id INT PRIMARY KEY,  
  employee_id INT,  
  project_id INT,  
  hours_worked INT NOT NULL,  
  FOREIGN KEY (employee_id) REFERENCES Employees(employee_id),  
  FOREIGN KEY (project_id) REFERENCES Projects(project_id)  
);
```

a. SELECT p.project_name, e.employee_name, MAX(a.hours_worked) AS total_hours_worked FROM Projects p JOIN Assignments a ON p.project_id = a.project_id JOIN Employees e ON a.employee_id = e.employee_id WHERE p.project_name = 'Web Development' GROUP BY p.project_name, e.employee_name ORDER BY total_hours_worked DESC;

b. SELECT p.project_name, e.employee_name, AVG(a.hours_worked) AS total_hours_worked FROM Projects p JOIN Assignments a ON p.project_id = a.project_id JOIN Employees e ON a.employee_id = e.employee_id WHERE p.project_name = 'Web Development' GROUP BY p.project_name, e.employee_name ORDER BY total_hours_worked DESC;

c. SELECT p.project_name, e.employee_name, COUNT(a.hours_worked) AS total_hours_worked FROM Projects p JOIN Assignments a ON p.project_id = a.project_id JOIN Employees e ON a.employee_id = e.employee_id WHERE p.project_name = 'Web Development' GROUP BY p.project_name, e.employee_name ORDER BY total_hours_worked DESC;

x. SELECT p.project_name, e.employee_name, SUM(a.hours_worked) AS total_hours_worked FROM Projects p JOIN Assignments a ON p.project_id = a.project_id JOIN Employees e ON a.employee_id = e.employee_id WHERE p.project_name = 'Web Development' GROUP BY p.project_name, e.employee_name ORDER BY total_hours_worked DESC;