PERANCANGAN BACKEND APLIKASI PENGELOLA KELAS KHUSUS TENAGA AJAR BERBASIS WEBSITE DENGAN NODEJS

Ali Ridho[1*)], Ir. Aghus Sofwan, S.T., M.T., Ph.D., IPU.[2] dan Yuli Christyono, S.T., M.T.[3]

[1]Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro, Semarang, Indonesia

E-mail: ridhoali20@students.undip..ac.id

Abstrak

Transformasi digital dalam dunia pendidikan menuntut adanya sistem informasi yang mampu menunjang efektivitas pembelajaran dan administrasi, khususnya bagi tenaga pengajar. Beban administratif yang tinggi dan keterbatasan literasi teknologi masih menjadi tantangan nyata yang dihadapi oleh banyak guru, terutama di daerah dengan akses teknologi yang terbatas. Oleh karena itu, diperlukan solusi berbasis teknologi informasi yang tidak hanya efektif, tetapi juga mudah diakses dan ramah pengguna. Tugas akhir ini bertujuan mengembangkan backend aplikasi pengelola kelas berbasis web yang dirancang khusus untuk tenaga pengajar. Sistem dibangun menggunakan Node.js, Express, TypeScript, dan MongoDB, dengan pendekatan arsitektur modular Model-Service-Controller (MSC). Backend ini menyediakan fitur manajemen pengguna, kelas, siswa, tugas, jurnal pembelajaran, serta otentikasi dan otorisasi berbasis token. Struktur API mengikuti standar RESTful yang konsisten dan mudah diintegrasikan dengan antarmuka frontend. Hasil pengembangan menunjukkan bahwa sistem backend mampu menangani kebutuhan administrasi kelas secara efisien dan fleksibel. Dengan adanya sistem ini, diharapkan guru dapat terbantu dalam mengurangi beban administratif, meningkatkan efektivitas pengelolaan kelas, serta mendukung peningkatan profesionalisme dalam pembelajaran berbasis digital.

***Kata kunci**: Backend, Node.js, RESTful API, Guru, Manajemen Kelas, Aplikasi Web*

Abstract

The digital transformation in education demands information systems that support effective teaching and administrative processes, especially for educators. High administrative workloads and limited digital literacy remain significant challenges for many teachers, particularly in areas with limited access to technology. Therefore, a technology-based solution is needed—one that is not only effective but also accessible and user-friendly.This final project focuses on developing the backend of a web-based classroom management application specifically designed for educators. The system is built using Node.js, Express, TypeScript, and MongoDB, applying a modular Model-Service-Controller (MSC) architecture. The backend provides features such as user management, class and student handling, assignment tracking, learning journals, and secure authentication and authorization via tokens. The API structure follows RESTful standards, ensuring consistency and ease of integration with frontend interfaces.The implementation demonstrates that the backend system can efficiently address classroom administrative needs while remaining flexible and scalable. This solution is expected to reduce teachers' administrative burden, improve classroom management, and support professional growth through digital learning tools.

***Keywords**: backend, node.js, restful api, teacher, classroom management, web application*

1. Pendahuluan

Pemanfaatan teknologi informasi telah menjadi elemen penting dalam transformasi dunia pendidikan modern. Sistem berbasis web, khususnya aplikasi pengelolaan kelas dan administrasi pembelajaran, telah berkontribusi secara signifikan dalam memperluas akses informasi, mempercepat pelaporan, serta meningkatkan keterlibatan baik dari pendidik maupun peserta didik[1]. Transformasi digital ini sejatinya membuka gerbang menuju efisiensi dan inovasi dalam proses belajar mengajar, meskipun implementasinya masih menemui tantangan.

Kondisi lapangan menunjukkan banyak guru masih bergulat dengan tumpukan pekerjaan administratif yang mengurangi waktu dan energi yang seharusnya dialokasikan untuk pengembangan profesional diri dan interaksi langsung dengan peserta didik[2]. Khusunya guru di daerah yang belum memiliki tingkat literasi teknologi

yang tinggi. Dilaporkan bahwa lebih dari 60% waktu guru dihabiskan untuk tugas administratif, seperti penyusunan Rencana Pelaksanaan Pembelajaran (RPP) dan berbagai laporan[3]. Kondisi ini secara drastis mengurangi waktu yang tersedia untuk pengajaran yang esensial, persiapan materi, serta kolaborasi profesional, yang pada akhirnya berdampak pada kualitas pembelajaran

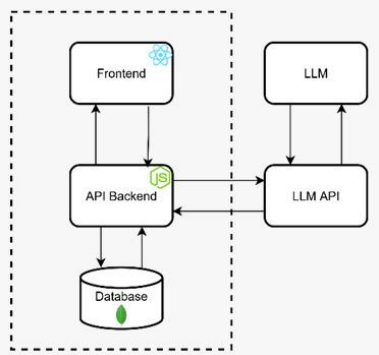
Untuk mengatasi permasalahan tersebut, dibutuhkan solusi berbasis teknologi yang tidak hanya efektif, tetapi juga mudah diakses oleh semua kalangan pendidik, termasuk mereka yang memiliki keterbatasan dalam penggunaan teknologi. Platform digital yang mendukung kolaborasi dan mentoring telah terbukti mampu meningkatkan profesionalisme guru melalui peer learning, pemantauan tugas, serta pengelolaan administrasi yang lebih efisien[4], [5]. Oleh karena itu, pengembangan sistem berbasis web yang intuitif, ramah pengguna, dan terjangkau menjadi sangat penting untuk mendukung kebutuhan guru, terutama dalam mengurangi beban administratif dan meningkatkan keterampilan digital mereka secara bertahap[6].

Berdasarkan latar belakang tersebut, tugas akhir ini difokuskan pada pengembangan backend aplikasi pengelola kelas untuk tenaga pengajar dengan menggunakan teknologi Node.js[7]. Pemilihan Node.js sebagai basis pengembangan backend dilandasi oleh kemampuannya dalam membangun aplikasi yang skalabel, efisien, dan mendukung komunikasi real-time[8], [9]. Melalui pengembangan ini, diharapkan dapat dibangun sebuah fondasi sistem yang tangguh dan responsif, yang tidak hanya membantu mengurangi beban administratif guru, tetapi juga mendorong peningkatan profesionalisme dan kualitas pendidikan secara keseluruhan.

2. Metode

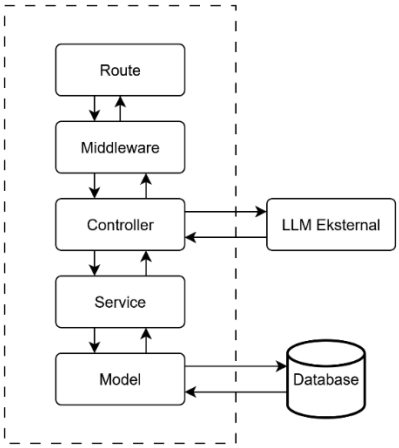
2.1. Deskripsi Sistem

Secara umum, arsitektur sistem aplikasi pengelola kelas mengadopsi pendekatan client-server yang memungkinkan pemisahan jelas antara antarmuka pengguna dan logika bisnis pada sisi server[10]. Dalam konfigurasi ini, aplikasi akan terdiri dari setidaknya tiga komponen utama yang saling berinteraksi, yaitu frontend, backend, dan basis data.



Gambar 2. 1 Arsitektur sistem

Backend berperan sebagai inti pemrosesan data dan logika bisnis aplikasi. Arsitektur backend yang digunakan mengikuti pola arsitektur bertingkat atau sering juga disebut *N-tier Architecture*, yang membagi sistem menjadi beberapa lapisan[11]. Pendekatan ini dipilih untuk meningkatkan modularitas, pemisahan tanggung jawab, dan memudahkan pemeliharaan kode.



Gambar 2. 2 Arsitektur backend aplikasi pengelola kelas

2.2. Kebutuhan Fungsional

Analisis kebutuhan fungsional untuk aplikasi pengelola kelas berbasis website mencakup berbagai fitur dan fungsi yang diperlukan untuk memenuhi kebutuhan pembimbingan dan pengajaran dalam suatu kelompok belajar. Berdasarkan deskripsi sistem yang akan dibuat, terdapat beberapa kebutuhan fungsional yang diidentifikasi yaitu:

1. *User* dapat melakukan register dan login
2. *User* dapat membuat, mengedit dan menghapus kelas
3. *User* dapat mengundang user lain untuk berkolaborasi dalam suatu kelas
4. *User* dapat memasukkan dan mengedit nilai
5. *User* dapat membuat dan mengedit jurnal harian kelas
6. *User* dapat membuat rencana pembelajaran
7. *User* dapat membuat rencana pembimbingan personal
8. *User* dapat melihat dan mengunduh rapor

2.3. Kebutuhan Non Fungsional

Analisis non-fungsional keseluruhan sistem informasi melibatkan aspek-aspek mempengaruhi kinerja, tampilan antarmuka, dan pemeliharaan. Berikut adalah analisis non-fungsional berdasarkan deskripsi sistem yang perlu dipertimbangkan:

A. Kebutuhan Operasional

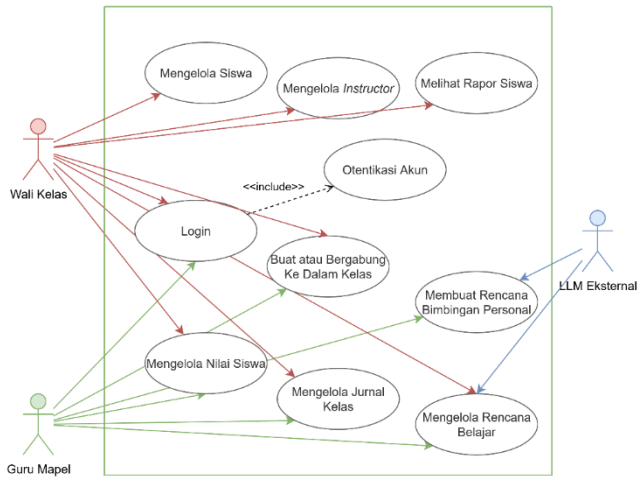
- Sistem informasi dapat beroperasi dalam beberapa web browser seperti: Google, Chrome, Mozilla, dan lain-lain.
- Diperlukan koneksi internet untuk menggunakan web maupun aplikasi.
- Sistem dirancang dengan bahasa pemrograman Typescript dengan framework MERN.
- Desain antarmuka pengguna (UI) sistem informasi web dan aplikasi dibuat agar sederhana dan mudah dipahami oleh pengguna.

B. Kebutuhan Keamanan

- Sistem wajib memiliki mekanisme autentikasi yang kuat untuk memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses.
- Sistem harus dapat menyimpan data-data ke dalam database yang aman.

2.4. Perancangan Diagram Use Case

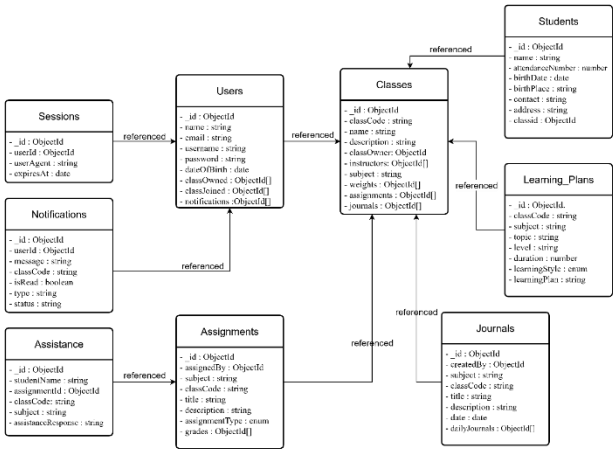
Berdasarkan analisis kebutuhan sistem, maka disusunlah diagram use case sebagaimana terlihat pada Gambar 2.3. Pengguna aplikasi pengelola kelas berbasis website dapat memiliki 2 role atau tugas berbeda, yaitu wali kelas dan guru mata pelajaran (mapel). Pembagian role ini bertujuan untuk membagi akses wali kelas dan guru mapel di dalam kelas. Sebagai wali kelas, pengguna dapat membuat dan mengedit data kelas, melihat nilai dan jurnal seluruh mapel, serta melihat dan mengunduh rapor. Sedangkan sebagai guru mapel, pengguna dapat memasukkan dan melihat nilai siswa, mengisi dan melihat jurnal kelas, serta mengakses LLM eksternal untuk membuat rencana belajar. Selain dari yang disebutkan sebelumnya, pengguna dapat melakukan login ke dalam aplikasi dan mengedit profil akunnya



Gambar 2. 3 *Use case diagram* apikasi pengelola kelas

2.5. Perancangan Basis Data

Perancangan backend memerlukan adanya struktur basis data yang menunjukkan entitas-entitas data yang terllibat. Pada tahap perancangan ini, MongoDB digunakan sebagai sistem manajemen basis data yang fleksibel dan berbasis dokumen[12]. Pendekatan berbasis dokumen ini dipilih karena sesuai dengan kebutuhan sistem yang dinamis dan terstruktur secara modular. Meskipun MongoDB merupakan basis data NoSQL, yang mana mendukung skema data yang bebas karena bukan termasuk basis data relasional, namun tetap perlu dilakukan pemodelan data untuk memberikan gambaran umum dan membantu dalam pengembangan basis data[13], [14]. Pemodelan basis data untuk aplikasi pengelola kelas diperlihatkan pada Gambar 2.4.



Gambar 2. 4 Desain model basis data apikasi pengelola kelas

Pemodelan basis data yang ditunjukkan pada Gambar 2.4 memperlihatkan entitas apa saja yang ada dalam sistem, yaitu *users, sessions, notifications, classes, students, assignments, journals, learning_plans*, dan *assistance*. Entitas *users* menyimpan data pengguna yang mana terhubung juga dengan *classes* menggunakan referensi pada field *classOwned* dan *classJoined*. Selain itu, *users* juga terhubung dengan entitas *sessions* yang menyimpan data sesi login pengguna dan *notifications* yang menyimpan undangan pengguna ke dalam kelas yang dikirim oleh pengguna lain.

Entitas *classes* berperan sebagai wadah yang menyimpan dan menghubungkan semua entitas yang berkaitan dengan aktifitas yang terjadi dalam kelas seperti *students, assignment, journals, learning_plans,* dan *assistance*. Entitas *students* menyimpan data siswa yang berada dalam kelas. Selain itu, *assignments* dan *journals* berfungsi mencatat data yang berkaitan dengan data kehadiran dan nilai siswa. Entitas *learning_plans* menyimpan rencana belajar yang berdasarkan field kelas dan mata pelajaran. Dan terakhir, entitas *assistance* menyimpan rencana bimbingan untuk siswa berdasarkan nilai siswa yang

terhubung dengan entitas *assignments* menggunakan referensi pada field *assignmentId*.

3. Hasil dan Pembahasan

3.1. Implementasi

3.1.1. Implementasi Basis Data

Implementasi basis data dilakukan dengan mengacu pada hasil perancangan model yang telah dibahas pada bab sebelumnya. Basis data dikembangkan menggunakan MongoDB, sebuah sistem manajemen basis data NoSQL berbasis dokumen yang menawarkan fleksibilitas dalam pengelolaan data semi-terstruktur. Setiap entitas dalam sistem, seperti pengguna, kelas, siswa, tugas, dan jurnal, direpresentasikan dalam bentuk dokumen yang memiliki skema tersendiri menggunakan library Mongoose.

3.1.2. Implementasi API Backend

Implementasi *Application Programming Interface* (API) pada sisi *backend* dikembangkan menggunakan Node.js dengan *framework* Express.js. API ini mengadopsi arsitektur RESTful untuk memfasilitasi komunikasi antara *frontend* (aplikasi website) dan basis data. Untuk manajemen basis data, digunakan Mongoose sebagai *Object Data Modeling* (ODM) yang terhubung dengan basis data MongoDB.

Setiap fungsionalitas aplikasi diimplementasikan sebagai endpoint API yang spesifik. Berikut disebutkan beberapa endpoint yang telah diterapkan:

- GET /users/me
- POST /auth/signin
- POST /students/:classid/create
- PATCH /assignments/:classId/giveScore
- DELETE /class/:id

3.2. Pengujian

Pengujian API adalah tahapan krusial untuk memastikan komunikasi data yang mulus dan fungsionalitas aplikasi yang baik. Tujuan utamanya adalah memverifikasi bahwa data yang dikirim dan diterima sudah sesuai ekspektasi, serta untuk mengidentifikasi potensi masalah teknis yang bisa memengaruhi kinerja aplikasi secara keseluruhan[15]. API yang tidak teruji dengan baik berisiko menyebabkan bug, kegagalan fungsionalitas, atau bahkan kehilangan data penting, sehingga pengujian dini sangat vital untuk memastikan interaksi aplikasi dengan backend berjalan lancar dan memberikan pengalaman pengguna terbaik.

Pengujian dilakukan untuk memastikan setiap API endpoint berfungsi dengan benar dan stabil. Metode yang digunakan adalah black box testing, yang berfokus pada validasi fungsionalitas tanpa memeriksa struktur internal kode. Pengujian ini mencakup skenario positif (input valid) dan negatif (input tidak valid) untuk memastikan API merespons dengan tepat dalam berbagai kondisi.

Pengujian API dalam proyek ini akan mencakup seluruh endpoint yang digunakan oleh aplikasi. Proses pengujian akan dilakukan menggunakan Postman, sebuah alat yang umum digunakan untuk menguji API. Permintaan akan dikirimkan dengan format body JSON, dan untuk endpoint yang memerlukan otentikasi, header akan menyertakan token yang diperoleh setelah pengguna berhasil masuk (login).

Tabel 3. 1 Pengujian endpoint API login

Input	Kasus uji	Ekspektasi	Hasil
body:{Email, password}	login dengan email dan *password* yang benar.	status: 200 ok	sesuai
	login dengan email benar, tapi password salah	status: 400 bad request	sesuai
	login dengan email yang tidak terdaftar.	status: 404 not found	sesuai
body:{Email, password}	login dengan format email yang tidak valid.	status: 400 bad request	sesuai
body:{Email/password}	login tanpa mengisi email atau *password*.	status: 400 bad request	sesuai

Tabel 3. 2 Pengujian endpoint API membuat kelas

Input	Kasus uji	Ekspektasi	Hasil
header: userid body:{name}	sudah login, mengisi form dengan data yang valid.	status: 201 created	sesuai
header: userid body:{name}	sudah login, tidak mengisi form dengan data yang valid.	status: 400 bad request	sesuai
header: userid body:{name}	belum login	status: 401 unauthorized	sesuai

Tabel 3. 3 Pengujian endpoint API mengisi nilai siswa

Input	Kasus uji	Ekspektasi	Hasil
header: userid param: classid body: {grades..}	sudah login, memiliki mata pelajaran terkait, semua data terisi dan valid.	status: 201 created	sesuai
	sudah login, memiliki mata pelajaran terkait, ada data yang tidak terisi atau tidak valid.	status: 400 bad request	sesuai
	sudah login, memasukkan nilai untuk mata pelajaran yang tidak valid	status: 404 not found	sesuai
	sudah login, tidak terdaftar sebagai guru dalam kelas	status: 404 not found	sesuai

Tabel 3. 4 Pengujian endpoint API melihat dan mengunduh rapor

Input	Kasus uji	Ekspektasi	Hasil
header: userid param: classid, studentid	sudah login; memiliki hak akses *classOwner*	status: 200 ok	sesuai
	sudah login; memiliki hak akses *classOwner*, mencoba melihat rapor untuk classid yang tidak valid atau tidak ditemukan	status: 404 not found	sesuai
header: userid param: classid, studentid	sudah login; memiliki hak akses *classOwner*, mencoba melihat rapor untuk studentid yang tidak valid atau tidak ditemukan.	status: 404 not found	sesuai
	belum login	status: 401 unauthorized	sesuai

Tabel 3. 5 Pengujian endpoint API membuat rencana belajar

Input	Kasus uji	Ekspektasi	Hasil
header: userid body: {prompt, parameter}	sudah login, membuat rencana belajar dengan semua parameter valid.	status: 200 ok	sesuai
	sudah login, membuat rencana belajar dengan parameter yang kosong atau tidak valid.	status: 400 bad request	sesuai
body: {prompt, parameter}	belum login	status: 401 unauthorized	sesuai

4. Kesimpulan

Rancangan diagram aplikasi pengelola kelas berhasil dibuat dengan baik dan jelas berdasarkan hasil analisis kebutuhan yang telah dilakukan, sehingga mampu merepresentasikan alur sistem secara menyeluruh. Dalam mendukung penyimpanan data aplikasi, sistem basis data dirancang menggunakan MongoDB, yang mengandalkan pendekatan berbasis dokumen dengan format penyimpanan data dalam bentuk JSON dan struktur data yang bersifat dinamis. Basis data ini kemudian diintegrasikan dengan backend guna memenuhi kebutuhan penyimpanan data secara efisien dan fleksibel. Selain itu, implementasi fungsi yang memanfaatkan layanan sistem eksternal, khususnya model kecerdasan buatan dari OpenAI, juga telah berhasil dibangun dan diintegrasikan dalam fitur pembuatan rencana belajar serta bimbingan, yang menjadi nilai tambah dalam sistem. Pengujian terhadap API menunjukkan bahwa setiap endpoint pada fitur-fitur fungsional telah berjalan sesuai dengan yang

diharapkan dan mampu memberikan respon yang sesuai dengan spesifikasi yang telah ditentukan.

Referensi

[1] P. BRUGLIERA, "The Effectiveness of Digital Learning Platforms in Enhancing Student Engagement and Academic Performance," *Journal of Education, Humanities, and Social Research*, vol. 1, no. 1, pp. 26–36, Dec. 2024, doi: 10.70088/xq3gy756.

[2] S. S. Sabon, "PROBLEMATIK PEMENUHAN BEBAN KERJA GURU DAN ALTERNATIF PEMENUHANNYA (Studi Kasus di Kota Depok Provinsi Jawa Barat)," *Jurnal Penelitian Kebijakan Pendidikan*, vol. 13, no. 1, pp. 27–44, Sep. 2020, doi: 10.24832/jpkp.v13i1.345.

[3] M. H. Napitupulu, A. Muddin, B. Bagiya, S. Diana, and N. S. Rosyidah, "Teacher Professional Development in the Digital Age: Strategies for Integrating Technology and Pedagogy," *Global International Journal of Innovative Research*, vol. 2, no. 10, pp. 2382–2396, Nov. 2024, doi: 10.59613/global.v2i10.334.

[4] R. Pesina, "Mentoring Software in Education and Its Impact on Teacher Development: An Integrative Literature Review," Feb. 2025.

[5] N. Yulin and S. D. Danso, "Assessing Pedagogical Readiness for Digital Innovation: A Mixed-Methods Study," Feb. 2025.

[6] A. A. Zulfa and O. Arifudin, "PERAN SISTEM INFORMASI AKADEMIK BERBASIS WEB DALAM UPAYA MENINGKATKAN EFEKTIVITAS DAN EFISIENSI PENGELOLAAN AKADEMIK DI PERGURUAN TINGGI," 2025.

[7] P. Sekhar Emmanni, "The Role of TypeScript in Enhancing Development with Modern JavaScript Frameworks," *International Journal of Science and Research (IJSR)*, vol. 10, no. 2, pp. 1738–1741, Feb. 2021, doi: 10.21275/sr24401234212.

[8] B. Basumatary and N. Agnihotri, "Benefits and Challenges of Using NodeJS," *International Journal of Innovative Research in Computer Science & Technology*, pp. 67–70, May 2022, doi: 10.55524/ijircst.2022.10.3.13.

[9] S. Verma, "International Journal of Research Publication and Reviews Express.Js and its Usage in Web Development," 2024. [Online]. Available: www.ijrpr.com

[10] D. Sereda, "Creating a Multi-tier Architecture for Web Applications: Design and Implementation," 2025, [Online]. Available: https://asrjetsjournal.org/index.php/American_Scientific_Journal/index

[11] N. S. Filho, "Separation of Responsibilities in Practice: Comparison between N-Tier Architecture and Clean Architecture," 2025, doi: 10.5281/zenodo.15164674.

[12] MongoDB Inc., "MongoDB Manual." Accessed: Apr. 17, 2025. [Online]. Available: https://www.mongodb.com/docs/manual/.

[13] A. Meier and M. Kaufmann, *SQL & NoSQL Databases*, 1st ed. Springer Vieweg Wiesbaden, 2019.

[14] P. Atzeni, F. Bugiotti, L. Cabibbo, R. Torlone, and R. Tre, "Data Modeling in the NoSQL World Data Modeling in the NoSQL World 6," vol. 67, 2020, doi: 10.1016/j.csi.2016.10.003ï.

[15] IBM, "What is API Testing?" Accessed: Jul. 21, 2025. [Online]. Available: https://www.ibm.com/think/topics/api-testing?