**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN EKSPERIMEN PERBANDINGAN IMPLEMENTASI API VERSIONING: URL PATH VS QUERY PARAMETERS VS CUSTOM HEADERS**

*Laporan ini disusun untuk memenuhi tugas 3 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

Asri Husnul Rosadi 221524035

Faris Abulkhoir 221524040

Mahardika Pratama 221524044

Muhamad Fahri Yuwan 221524047

Najib Alimudin Fajri 221524053

Sarah 221524059

Septyana Agustina 221524060

Dosen Pengampu:

Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# **DAFTAR ISI**

[DAFTAR ISI i](#_Toc176537927)

[A. IDENTIFIKASI PROBLEM 1](#_Toc176537928)

[B. DESKRIPSI PROBLEM 1](#_Toc176537929)

[C. METODOLOGI EKSPERIMEN 1](#_Toc176537930)

[D. PELAKSANAAN EKSPERIMEN 2](#_Toc176537931)

[E. ANALISIS HASIL EKSPERIMEN 3](#_Toc176537932)

# **IDENTIFIKASI PROBLEM**

API versioning adalah praktik yang digunakan untuk mengelola perubahan pada API tanpa mengganggu pengguna yang ada. Terdapat beberapa metode untuk mengimplementasikan versioning, seperti melalui URL Path, Query Parameters, dan Custom Headers. Tujuan eksperimen ini adalah untuk membandingkan efektivitas dan efisiensi ketiga pendekatan tersebut dalam konteks kinerja, caching, dan kemudahan penggunaan dalam manajemen perubahan untuk beralih ke versi yang lebih baru.

# **DESKRIPSI PROBLEM**

Untuk memahami perbedaan antara ketiga metode API versioning ini, eksperimen ini mengevaluasi beberapa metrik penting:

1. Kinerja Caching: Efektivitas caching respons API untuk setiap metode versioning.
2. Kemudahan Penggunaan: Bagaimana setiap metode mempengaruhi struktur URL dan header permintaan.
3. Dukungan Standar: Seberapa mudah setiap metode diterapkan dan diadopsi dalam sistem yang sudah ada.
4. Keamanan: Bagaimana setiap metode melindungi versi API dari potensi eksploitasi atau penggunaan yang salah.

# **METODOLOGI EKSPERIMEN**

**Variabel Terikat:**

* + Kinerja caching (kemampuan cache respons API).
  + Kemudahan penggunaan (kompleksitas dalam mengkonsumsi API).
  + Efisiensi (latensi respons API).

**Variabel Bebas:** Metode versioning yang digunakan (URL Path, Query Parameters, Custom Headers).

**Metode Pengukuran:**

* + Caching Efektivitas: Diukur dengan menggunakan cache kontrol di server dan observasi perubahan respons caching.
  + Kinerja Permintaan API: Diukur menggunakan alat pengujian seperti Postman atau Curl untuk melihat kecepatan dan waktu respons.

**Skenario Pengujian:**

* + Uji respons API dengan metode versioning melalui URL Path.
  + Uji respons API dengan metode versioning melalui Query Parameters.
  + Uji respons API dengan metode versioning melalui Custom Headers.
  + Observasi hasil caching dan kinerja untuk setiap metode.

**Alat dan Teknologi:**

* + Express.js: Digunakan untuk membuat server API.
  + Postman atau Curl: Alat untuk menguji permintaan API.
  + Cache-Control Middleware: Digunakan untuk mengatur caching pada server.

**Langkah-langkah Eksperimen:**

* + Implementasikan API dengan tiga metode versioning berbeda.
  + Lakukan pengujian caching dan kinerja menggunakan Postman.
  + Observasi hasil untuk setiap metode dan bandingkan efektivitasnya.

# **PELAKSANAAN EKSPERIMEN**

**Tahap 1: Setup Lingkungan**

* 1. **Setup Aplikasi:** Menggunakan Express.js untuk mengimplementasikan tiga metode versioning: URL Path (/v1/users, /v2/users, dll.), Query Parameters (/users?version=1), dan Custom Headers (/users dengan header api-version).
  2. **Cache-Control Implementasi:** Tambahkan middleware cache-control untuk mengatur caching respons API selama 5 menit.

**Tahap 2: Pengujian Kinerja dan Caching**

1. **Pengujian API:**

Kirim permintaan menggunakan Postman ke tiga endpoint berbeda:

* 1. URL Path Versioning: /v1/users, /v2/users, dll.
  2. Query Parameters Versioning: /users?version=1, /users?version=2, dll.
  3. Custom Headers Versioning: /users dengan header api-version: 1, api-version: 2, dll.

1. **Observasi Caching:**
   1. Catat respons caching untuk setiap permintaan menggunakan Postman.
   2. Lihat apakah respons API di-cache sesuai dengan konfigurasi Cache-Control yang diatur.

# **ANALISIS HASIL EKSPERIMEN**

**Hasil Eksperimen:**

* 1. **Kinerja Caching:**
* URL Path Versioning: Efektivitas caching lebih tinggi karena cache server dapat dengan mudah mengenali endpoint yang berbeda berdasarkan URL.
* Query Parameters Versioning: Caching bekerja dengan baik, tetapi beberapa server cache mungkin memerlukan konfigurasi tambahan untuk menangani query parameters.
* Custom Headers Versioning: Caching kurang efektif karena cache server biasanya tidak memeriksa header untuk menentukan cache yang relevan.
  1. **Kemudahan Penggunaan:**
* URL Path Versioning: Mudah digunakan dan dipahami karena versi API terlihat langsung di URL.
* Query Parameters Versioning: Sedikit lebih kompleks karena memerlukan parameter tambahan pada URL.
* Custom Headers Versioning: Memerlukan konfigurasi tambahan di klien API (seperti Postman atau browser) untuk menambahkan header khusus.
  1. **Dukungan Standar:**
  + URL Path Versioning: Didukung secara luas oleh kebanyakan server dan alat cache.
  + Query Parameters Versioning: Dukungan luas tetapi memerlukan konfigurasi tambahan untuk beberapa server cache.
  + Custom Headers Versioning: Kurang standar dan tidak didukung secara default oleh beberapa alat caching.
  1. **Keamanan:**
  + URL Path Versioning dan Query Parameters: Rentan terhadap eksploitasi jika URL atau parameter API terekspos ke publik.
  + Custom Headers Versioning: Lebih aman karena header tidak terlihat langsung di URL, tetapi masih bisa dicatat di log server.
  1. **Analisis Keseluruhan**
  + URL Path Versioning adalah metode yang paling efektif untuk caching, mudah digunakan, dan didukung secara luas.
  + Query Parameters Versioning sedikit lebih kompleks untuk caching tetapi menawarkan fleksibilitas tambahan.
  + Custom Headers Versioning memberikan lebih banyak kontrol dan keamanan, tetapi kurang efektif untuk caching dan memerlukan lebih banyak konfigurasi.
  1. **Pros dan Cons**

**Pros URL Path Versioning:**

* + Mudah dipahami dan diimplementasikan.
  + Efektif untuk caching.
  + Mendukung standar REST API.

**Cons URL Path Versioning:**

* Tidak fleksibel jika Anda ingin mengubah versi API tanpa mengubah URL.

**Pros Query Parameters Versioning:**

* + Fleksibel dalam mengelola beberapa versi API.
  + Tidak perlu mengubah URL utama.

**Cons Query Parameters Versioning:**

* + Caching mungkin memerlukan konfigurasi tambahan.
  + Membuat URL lebih kompleks.

**Pros Custom Headers Versioning:**

* + Lebih aman karena versi API tersembunyi di dalam header.
  + Fleksibel untuk pengaturan versi tanpa mengubah URL.

**Cons Custom Headers Versioning:**

* + Kurang efektif untuk caching.
  + Memerlukan konfigurasi tambahan di klien dan server.

**Kesimpulan**

* + **URL Path Versioning** direkomendasikan untuk penggunaan umum karena kemudahan implementasi dan caching yang efektif.
  + **Query Parameters Versioning** dapat digunakan untuk fleksibilitas tambahan, tetapi membutuhkan lebih banyak pengaturan untuk caching.
  + **Custom Headers Versioning** ideal untuk situasi di mana keamanan adalah prioritas, tetapi kurang efisien dalam hal caching.