**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN HANDLING CORS IN WEB APPLICATIONS: JSONP VS PROXY SERVER VS CORS 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](#_Toc176602150)

[A. IDENTIFIKASI PROBLEM 1](#_Toc176602151)

[B. DESKRIPSI PROBLEM 1](#_Toc176602152)

[C. METODOLOGI EKSPERIMEN 1](#_Toc176602153)

[D. PELAKSANAAN EKSPERIMEN 2](#_Toc176602154)

[E. ANALISIS HASIL EKSPERIMEN 4](#_Toc176602155)

# **IDENTIFIKASI PROBLEM**

Cross-Origin Resource Sharing (CORS) adalah mekanisme keamanan penting dalam pengembangan aplikasi web yang memungkinkan atau membatasi permintaan antara sumber yang berbeda. Dalam pengaturan standar, browser web mencegah aplikasi dari mengakses sumber daya di domain yang berbeda, yang dapat menjadi masalah ketika aplikasi perlu mengonsumsi API dari domain yang berbeda. Untuk mengatasi ini, terdapat beberapa pendekatan umum: JSONP, Proxy Server, dan CORS Headers. Tujuan dari eksperimen ini adalah untuk menganalisis efektivitas dan kinerja dari ketiga pendekatan ini dalam menangani permintaan cross-origin.

# **DESKRIPSI PROBLEM**

Eksperimen ini mengevaluasi tiga pendekatan dalam menangani permintaan cross-origin:

1. **JSONP (JSON with Padding):** Teknik lama yang memungkinkan permintaan GET cross-origin melalui tag <script>.
2. **Proxy Server:** Teknik yang menggunakan server proxy untuk mengambil data dari server target dan meneruskannya ke klien, sehingga menghindari masalah CORS.
3. **CORS Headers:** Metode modern yang mengandalkan header HTTP untuk mengizinkan atau membatasi permintaan cross-origin dari browser.

Eksperimen ini akan membandingkan ketiga metode dalam hal:

1. **Kemudahan Implementasi:** Seberapa mudah implementasi setiap pendekatan pada aplikasi.
2. **Keamanan:** Potensi risiko keamanan yang terkait dengan masing-masing metode.
3. **Kompatibilitas Browser:** Dukungan di berbagai browser web.
4. **Kinerja:** Latensi dan waktu respons saat melakukan permintaan.

# **METODOLOGI EKSPERIMEN**

**Variabel Terikat:** Kemudahan implementasi, keamanan, kompatibilitas browser, dan kinerja. **Variabel Bebas:** Metode yang digunakan (JSONP, Proxy Server, CORS Headers).

**Metode Pengukuran:**

* **Kemudahan Implementasi:** Diukur dengan melihat jumlah kode dan kompleksitas yang dibutuhkan untuk mengimplementasikan setiap metode.
* **Keamanan:** Dievaluasi berdasarkan potensi risiko yang teridentifikasi (misalnya, risiko XSS dengan JSONP).
* **Kompatibilitas Browser:** Diuji di berbagai browser modern dan versi yang lebih lama.
* **Kinerja:** Diukur berdasarkan latensi dan waktu respons saat melakukan permintaan cross-origin.

**Skenario Pengujian:**

* **Implementasi JSONP:** Uji permintaan GET cross-origin dengan menggunakan teknik JSONP.
* **Setup Proxy Server:** Implementasikan server proxy menggunakan Node.js dan uji respons dari test-proxy.html.
* **Konfigurasi CORS Headers:** Setup server dengan CORS headers dan uji dengan berbagai metode HTTP dari test-cors.html.

**Alat dan Teknologi:**

* **Node.js dan Express:** Untuk mengimplementasikan Proxy Server dan CORS Headers.
* **Browser (Chrome):** Untuk menguji kompatibilitas dan keamanan.
* **JavaScript (fetch API):** Untuk melakukan permintaan HTTP.

# **PELAKSANAAN EKSPERIMEN**

**Tahap 1: Setup Lingkungan**

1. **Setup Server JSONP:**
   * Implementasikan server yang mendukung JSONP untuk permintaan GET cross-origin.
   * Siapkan file HTML untuk menguji permintaan JSONP dari berbagai browser.
2. **Setup Server Proxy:**
   * Implementasikan server proxy menggunakan Node.js untuk meneruskan permintaan ke server target (proxy-server.js).
   * Siapkan file HTML (test-proxy.html) untuk menguji server proxy.
3. **Setup Server CORS:**
   * Implementasikan server dengan CORS headers menggunakan Node.js untuk mengizinkan permintaan cross-origin (cors-server.js).
   * Siapkan file HTML (test-cors.html) untuk menguji server CORS dengan berbagai metode HTTP.

**Tahap 2: Pengujian Implementasi**

1. **Pengujian JSONP:**
   * Kirim permintaan GET menggunakan JSONP dari file HTML yang disiapkan.
   * Akses http://localhost:3001/data?callback=yourCallbackFunction untuk melihat respons JSONP.
   * Amati dan catat respons.
2. **Pengujian Proxy Server:**
   * Kirim permintaan GET, POST, PUT, dan DELETE ke server proxy.
   * Akses http://localhost:3002/proxy untuk melihat respons Proxy Server.
   * Amati dan catat respons.
3. **Pengujian CORS Headers:**
   * Kirim permintaan GET, POST, PUT, dan DELETE ke server dengan CORS headers.
   * Akses http://localhost:3003/data untuk melihat respons CORS headers.
   * Amati dan catat respons.

**Tahap 3: Analisis dan Evaluasi**

1. **Analisis Kemudahan Implementasi:**
   * Evaluasi jumlah kode dan kompleksitas implementasi setiap metode.
   * Identifikasi kemudahan atau kesulitan dalam setup dan konfigurasi.
2. **Analisis Keamanan:**
   * Evaluasi potensi risiko keamanan untuk masing-masing metode (misalnya, XSS dengan JSONP).
   * Identifikasi langkah-langkah mitigasi untuk risiko yang teridentifikasi.
3. **Analisis Kompatibilitas Browser:**
   * Uji respons di berbagai browser modern dan lama.
   * Identifikasi masalah kompatibilitas yang muncul pada browser tertentu.
4. **Analisis Kinerja:**
   * Ukur latensi dan waktu respons untuk permintaan cross-origin.
   * Bandingkan hasil kinerja antara JSONP, Proxy Server, dan CORS Headers.

# **ANALISIS HASIL EKSPERIMEN**

**Hasil Eksperimen:**

1. **Kemudahan Implementasi:**
   * JSONP: Mudah untuk permintaan GET tetapi tidak mendukung metode lain seperti POST, PUT, atau DELETE.
   * Proxy Server: Sedikit lebih kompleks untuk disetup, tetapi dapat menangani semua jenis permintaan HTTP.
   * CORS Headers: Paling mudah diimplementasikan dengan middleware di server dan mendukung semua metode HTTP.
2. **Keamanan:**
   * JSONP: Berisiko tinggi terhadap serangan XSS karena penggunaan tag <script>.
   * Proxy Server: Lebih aman karena klien hanya berkomunikasi dengan server local tetapi memerlukan pengaturan dan pengelolaan tambahan.
   * CORS Headers: Relatif aman jika dikonfigurasi dengan benar untuk mengizinkan domain tertentu.
3. **Kompatibilitas Browser:**
   * JSONP: Kompatibel dengan semua browser modern dan lama, tetapi terbatas pada permintaan GET.
   * Proxy Server: Tidak bergantung pada browser, sehingga sangat kompatibel.
   * CORS Headers: Didukung di semua browser modern, namun memerlukan pengaturan yang benar di server.
4. **Kinerja:**
   * JSONP: Cepat untuk permintaan GET sederhana, tetapi rentan terhadap penundaan jika data besar.
   * Proxy Server: Tambahan latensi minimal, namun dapat mengkonsumsi lebih banyak sumber daya server.
   * CORS Headers: Performa terbaik tanpa overhead tambahan di server.

**Analisis Keseluruhan:**

* JSONP efektif untuk permintaan GET sederhana tetapi tidak aman dan terbatas fungsionalitasnya.
* Proxy Server memberikan fleksibilitas penuh untuk semua jenis permintaan, meskipun memerlukan pengaturan tambahan.
* CORS Headers merupakan pendekatan yang paling modern dan fleksibel, mudah diimplementasikan, dan aman jika dikonfigurasi dengan benar.

**Pros dan Cons:**

**Pros JSONP:**

* Sederhana untuk diimplementasikan.
* Didukung oleh hampir semua browser.

**Cons JSONP:**

* Terbatas hanya pada permintaan GET.
* Risiko tinggi terhadap serangan XSS.

**Pros Proxy Server:**

* Mendukung semua jenis permintaan HTTP.
* Menyediakan kontrol penuh atas permintaan, seperti menangani kebutuhan autentikasi dan keamanan tambahan.
* Menyembunyikan identitas asli klien dari server tujuan.

**Cons Proxy Server:**

* Memerlukan pengaturan dan pengelolaan server tambahan.
* Menambah sedikit overhead di server proxy.

**Pros CORS Headers:**

* Mendukung semua metode HTTP.
* Mudah diimplementasikan dengan konfigurasi server.
* Aman dan didukung oleh semua browser modern.

**Cons CORS Headers:**

* Bergantung pada konfigurasi server yang benar.
* Tidak didukung oleh browser lama.

**Kesimpulan**

Untuk aplikasi modern, CORS Headers adalah metode yang paling direkomendasikan karena mendukung semua jenis permintaan, aman, dan mudah diimplementasikan. Proxy Server juga merupakan opsi yang baik untuk skenario yang memerlukan kontrol yang lebih besar atau ketika server tidak mendukung CORS, sedangkan JSONP hanya cocok untuk permintaan GET, memiliki risiko keamanan tinggi, dan hanya digunakan untuk kompatibilitas dengan browser lama.