

## **PENGEMBANGAN WEB (TEORI)**

### **LAPORAN EKSPERIMEN MENGENAI PROBLEM SINKRONISASI TABEL SECARA REAL-TIME ANTARA FRONTEND DAN BACKEND**

*Laporan ini disusun untuk memenuhi tugas 1 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

<b>Asri Husnul Rosadi</b>	<b>221524035</b>
Faris Abulkhoir	221524040
Mahardika Pratama	221524044
Muhamad Fahri Yuwan	221524047
Najib Alimudin Fajri	221524053
Septyana Agustina	221524058
Sarah	221524059

Dosen Pengampu:  
Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA  
PROGRAM STUDI D4 TEKNIK INFORMATIKA  
POLITEKNIK NEGERI BANDUNG  
2024**

## DAFTAR ISI

<b>DAFTAR ISI</b> .....	i
<b>A. IDENTIFIKASI PROBLEM</b> .....	1
<b>B. DESKRIPSI PROBLEM</b> .....	1
<b>C. METODOLOGI EKSPERIMEN</b> .....	1
<b>D. PELAKSANAAN EKSPERIMEN</b> .....	2
<b>E. ANALISIS HASIL EKSPERIMEN</b> .....	3
<b>F. KESIMPULAN</b> .....	3

Link chatGPT : <https://chatgpt.com/share/5785b2d1-59bc-47bb-a3bb-9111ca4bb14f>

## A. IDENTIFIKASI PROBLEM

Dalam pengembangan aplikasi web modern, sering kali diperlukan fitur untuk memperbarui data secara real-time agar pengguna dapat melihat data terbaru tanpa harus me-refresh halaman secara manual. Tantangannya adalah bagaimana mengimplementasikan pembaruan data secara otomatis di frontend ketika terjadi perubahan di backend, dengan efisiensi dan responsivitas tinggi.

## B. DESKRIPSI PROBLEM

Aplikasi yang sedang dikembangkan memerlukan fitur untuk menyinkronkan tabel data di frontend (React.js) dengan data yang diperbarui di backend (Laravel) secara real-time. Pengguna harus dapat melihat perubahan data langsung setelah data tersebut diperbarui di server. Hal ini mencakup:

- Menyediakan form inputan di frontend untuk mengirim data.
- Menampilkan tabel yang otomatis memperbarui data tanpa perlu refresh halaman.
- Menggunakan teknologi yang memungkinkan sinkronisasi data real-time antara frontend dan backend.

## C. METODOLOGI EKSPERIMEN

### 1. Pemilihan Teknologi:

- **Pusher:** Untuk menyiarkan event secara real-time ke frontend.
- **Laravel Broadcasting:** Untuk mengelola dan menyiarkan event dari backend.
- **Inertia.js:** Untuk mengintegrasikan frontend dan backend tanpa memerlukan API tambahan.

### 2. Konfigurasi dan Implementasi:

- **Backend (Laravel):** Konfigurasi Pusher di Laravel, buat event broadcasting, dan update controller untuk memicu event.
- **Frontend (React):** Buat form inputan untuk mengirim data dan tabel untuk menampilkan data. Integrasikan Pusher untuk mendengarkan event dan memperbarui UI.

## D. PELAKSANAAN EKSPERIMEN

### 1. Setup Pusher dan Laravel Broadcasting:

- **Konfigurasi:** Tambahkan kredensial Pusher di file .env dan config/broadcasting.php.
- **Event:** Buat event DataUpdated yang mengimplementasikan ShouldBroadcast di Laravel. Event ini disiarkan setiap kali data diperbarui.
- **Controller:** Modifikasi DataController untuk memicu event DataUpdated setelah menyimpan data dan gunakan Inertia.js untuk merender halaman Dashboard.

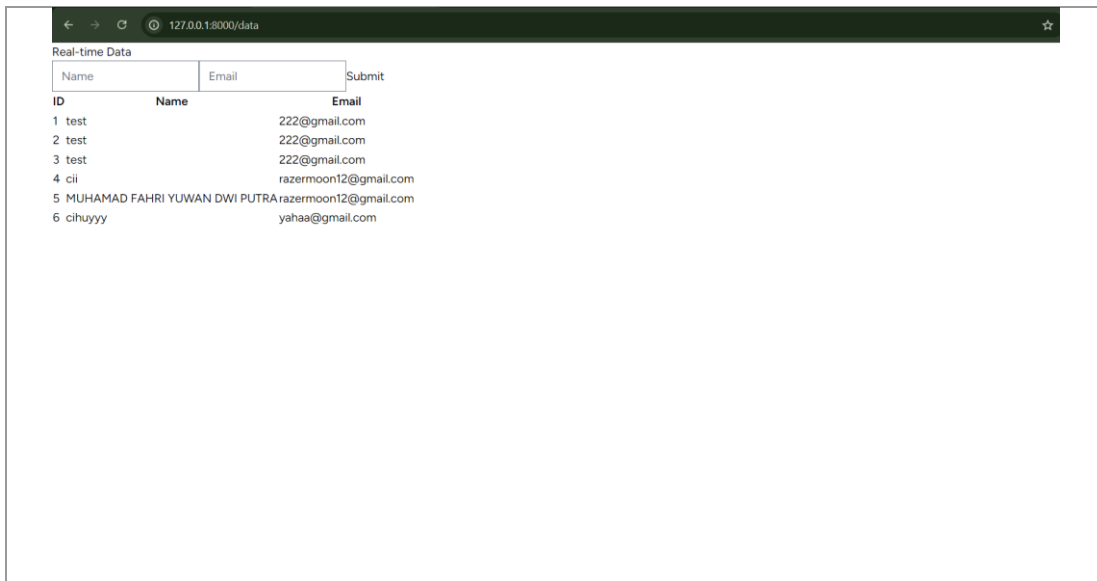
### 2. Implementasi di Frontend (React):

- **Form Input:** Buat form DataForm untuk mengirim data ke backend menggunakan Inertia.post.
- **Tabel Data:** Buat komponen DataTable untuk menampilkan data. Integrasikan Pusher untuk mendengarkan event DataUpdated dan memperbarui tabel data secara otomatis.

### 3. Testing dan Debugging:

- **Verifikasi:** Pastikan form mengirimkan data dengan benar dan event diterima di frontend.
- **Pemantauan:** Uji apakah data di tabel frontend diperbarui sesuai dengan data yang diterima dari backend.

## E. ANALISIS HASIL EKSPERIMEN



ID	Name	Email
1	test	222@gmail.com
2	test	222@gmail.com
3	test	222@gmail.com
4	cii	razermoon12@gmail.com
5	MUHAMAD FAHRI YUWAN DWI PUTRA	razermoon12@gmail.com
6	cihuyyy	yahaa@gmail.com

### 1. Kelebihan:

- **Real-Time Data:** Data di frontend diperbarui secara otomatis dan langsung setelah ada perubahan di backend, memberikan pengalaman pengguna yang responsif.
- **Integrasi Seamless:** Inertia.js memudahkan integrasi antara frontend React dan backend Laravel tanpa memerlukan API tambahan.
- **Mudah Digunakan:** Pusher menyediakan API yang sederhana untuk implementasi real-time messaging.

### 2. Kekurangan:

- **Biaya:** Pusher adalah layanan berbayar, dan biaya dapat meningkat tergantung pada skala penggunaan.
- **Kompleksitas:** Mengatur konfigurasi Pusher, Laravel Broadcasting, dan Inertia.js memerlukan pemahaman mendalam dan bisa menjadi kompleks.
- **Ketergantungan Eksternal:** Bergantung pada layanan eksternal seperti Pusher dapat mempengaruhi ketersediaan dan performa.

## F. KESIMPULAN

Implementasi fitur real-time data update menggunakan Pusher, Laravel Broadcasting, dan Inertia.js berhasil mencapai tujuan untuk memperbarui data secara otomatis di frontend tanpa memerlukan refresh halaman manual. Meskipun ada beberapa tantangan terkait biaya dan kompleksitas konfigurasi, teknologi yang digunakan memberikan solusi yang efisien dan responsif untuk sinkronisasi data real-time. Penggunaan Pusher untuk broadcasting event dan Inertia.js untuk integrasi frontend dan backend menunjukkan hasil yang positif dalam meningkatkan pengalaman pengguna dengan pembaruan data yang cepat dan akurat.