



UNIVERSITAS INDONESIA

**Pengimplementasian Arsitektur *Event-Driven* untuk Pengembangan
Sistem *Earthquake Early Warning System* Menggunakan Model
*Deep-Learning***

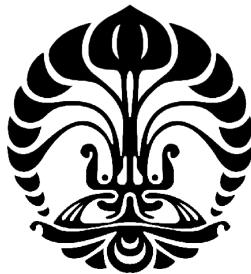
SKRIPSI

MUHAMMAD AGIL GHIFARI - 2006595835

TAUFIK PRAGUSGA - 2006595980

ZIDAN KHARISMA ADIDARMA - 2006463881

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
2023**



UNIVERSITAS INDONESIA

**Pengimplementasian Arsitektur *Event-Driven* untuk Pengembangan
Sistem *Earthquake Early Warning System* Menggunakan Model
*Deep-Learning***

SKRIPSI

**Diajukan sebagai salah satu syarat untuk
memperoleh gelar Sarjana Ilmu Komputer**

MUHAMMAD AGIL GHIFARI - 2006595835

TAUFIK PRAGUSGA - 2006595980

ZIDAN KHARISMA ADIDARMA - 2006463881

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
2023**

HALAMAN PERSETUJUAN

Judul : Pengimplementasian Arsitektur *Event-Driven* untuk Pengembangan Sistem *Earthquake Early Warning System* Menggunakan Model *Deep-Learning*

Nama Penulis 1 : Muhammad Agil Ghifari

NPM Penulis 1 : 2006595835

Nama Penulis 2 : Taufik Pragusga

NPM Penulis 2 : 2006595980

Nama Penulis 3 : Zidan Kharisma Adihdarma

NPM Penulis 3 : 2006463881

Laporan Skripsi ini telah diperiksa dan disetujui.

18 Desember 2023

Ari Wibisono, S.Kom., M.Kom
(Pembimbing 1 Skripsi)

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya kami sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama Penulis 1 : Muhammad Agil Ghifari

NPM Penulis 1 : 2006595835

Nama Penulis 2 : Taufik Pragusga

NPM Penulis 2 : 2006595980

Nama Penulis 3 : Zidan Kharisma Adihdarma

NPM Penulis 3 : 2006463881

Tanda Tangan Penulis 1 :



Tanda Tangan Penulis 2 :



Tanda Tangan Penulis 3 :



Tanggal : **18 Desember 2023**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama/NPM Mahasiswa 1 : Muhammad Agil Ghifari/2006595835

Nama/NPM Mahasiswa 2 : Taufik Pragusga/2006595980

Nama/NPM Mahasiswa 3 : Zidan Kharisma Adihdarma/2006463881

Program Studi : Sarjana Ilmu Komputer

Judul : Pengimplementasian Arsitektur *Event-Driven* untuk Pengembangan Sistem *Earthquake Early Warning System* Menggunakan Model *Deep-Learning*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Sarjana Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia

DEWAN PENGUJI

Pembimbing 1 : Ari Wibisono,
S.Kom., M.Kom

Penguji : Muhammad Hafizhuddin
Hilman, S.Kom., M.Kom., Ph.D.

Penguji : Heri Kurniawan,
S.Kom., M.Kom.

Ditetapkan di : Depok, Jawa Barat

Tanggal :

KATA PENGANTAR

Penulis ingin mengungkapkan rasa terima kasih kepada Tuhan Yang Maha Kuasa atas segala berkah dan anugerah-Nya, yang memungkinkan penyelesaian skripsi ini. Skripsi ini disusun sebagai bagian dari persyaratan untuk meraih gelar Sarjana Komputer di Fakultas Ilmu Komputer, Universitas Indonesia. Keberhasilan dalam menuntaskan skripsi ini tak lepas dari kontribusi, nasihat, serta dukungan berbagai pihak, baik yang terlibat langsung maupun tidak. Untuk itu, penulis ingin menyampaikan rasa syukur dan terima kasih kepada:

1. Allah SWT yang telah memberikan kesehatan, kemampuan, dan juga nikmat-nikmat lainnya untuk penulis sehingga dapat menyelesaikan penelitian ini.
2. Orang tua dan keluarga penulis yang selalu mendoakan dan mendukung penulis dari awal perkuliahan sampai penulisan tugas akhir ini
3. Bapak Ari Wibisono, S.Kom., M.Kom. sebagai dosen pembimbing tugas akhir ini yang sudah menyempatkan waktu, tenaga dan pemikirannya agar penulis dapat menyelesaikan tugas akhir ini.
4. Bapak Dr. Ade Azurat, S.Kom., selaku pembimbing akademik dari penulis 1 dan penulis 2 yang telah memberikan bimbingan selama masa perkuliahan di Fakultas Ilmu Komputer Universitas Indonesia.
5. Muhammad Agil Ghifari, selaku penulis 1 yang telah dengan baik mengerjakan tugas akhir dan banyak membantu penulis 2 dan 3 saat dalam kesulitan.
6. Taufik Pragusga, selaku penulis 2, dalam penelitian ini mengerjakan tugas akhir dengan baik, menjadi teman baik penulis 1 dan 2 selama kuliah.
7. Zidan Kharisma Adidharma, selaku penulis 3, dalam penelitian ini banyak membantu penulis 2 dalam mengerjakan tugas akhir serta menjadi teman kos dari penulis 1

8. Teman-teman penulis yang sudah memberikan dukungan dan menemani dalam menyelesaikan tugas akhir ini.
9. Kelompok Aesthetic yang banyak membantu penulis 1, 2, dan 3 selama kuliah dalam mengerjakan tugas dan menghibur dengan candaan-candaannya

Penulis mengakui bahwa penelitian ini masih memiliki ruang untuk perbaikan dan terdapat beberapa aspek yang belum optimal. Dengan demikian, penulis terbuka untuk menerima segala bentuk kritik dan masukan yang dapat meningkatkan kualitas penelitian ini. Penulis berkeinginan agar penelitian ini dapat berguna dan bermanfaat bagi banyak orang.

Depok, 18 Desember 2023



(Muhammad Agil Ghifari)



(Taufik Pragusga)



(Zidan Kharisma Adidharma)

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama Penulis 1 : Muhammad Agil Ghifari
NPM Penulis 1 : 2006595835
Nama Penulis 2 : Taufik Pragusga
NPM Penulis 2 : 2006595980
Nama Penulis 3 : Zidan Kharisma Adidharma
NPM Penulis 3 : 2006463881
Program Studi : Ilmu Komputer
Fakultas : Ilmu Komputer
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

Pengimplementasian Arsitektur *Event-Driven* untuk Pengembangan Sistem *Earthquake Early Warning System* Menggunakan Model *Deep-Learning*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok, Jawa Barat

Pada Tanggal : 18 Desember 2023

Yang menyatakan



(Muhammad Agil Ghifari)



(Taufik Pragusga)



(Zidan Kharisma Adidharma)

ABSTRAK

Nama Penulis : Muhammad Agil Ghifari, Taufik Pragusga, Zidan Kharisma Adidharma
Program Studi : Ilmu Komputer
Judul : Pengimplementasian Arsitektur *Event-Driven* untuk Pengembangan Sistem *Earthquake Early Warning System* Menggunakan Model *Deep-Learning*
Pembimbing 1 : Ari Wibisono. S.Kom., M.Kom

Kata Kunci: Arsitektur *Event-Driven*, Sistem Pendekripsi Dini Gempa, *Deep Learning*, Seismik, Hiposenter, Magnitudo, Visualisasi Data, Sistem Terdistribusi.

Penelitian ini berfokus pada pengembangan sistem peringatan dini gempa bumi yang memanfaatkan arsitektur *event-driven* dan model *deep-learning*. Tujuannya adalah untuk memodelkan data seismik guna mendekripsi gelombang awal, hiposenter, magnitudo, dan kedalaman gempa. Penulis mengumpulkan data dari ratusan titik seismograf dan mengolahnya dengan model *deep-learning* untuk menghasilkan prediksi yang akurat. Sistem ini dirancang untuk memberikan visualisasi dan informasi yang mendukung Badan Meteorologi, Klimatologi, dan Geofisika (BMKG) dalam mendekripsi aspek-aspek kritis gempa. Selain itu, penulis mengembangkan sistem terdistribusi untuk mengelola permintaan dan pengolahan data skala besar dengan efisiensi tinggi. Antarmuka pemrograman aplikasi (API) juga disajikan untuk memungkinkan prediksi data yang mudah diakses dan dipahami. Terakhir, integrasi antara model machine learning dengan backend dan frontend dirancang untuk memberikan tampilan yang ramah pengguna. Penelitian ini berkontribusi dalam mengembangkan sistem peringatan dini gempa yang lebih canggih dan responsif, sehingga dapat meningkatkan kesiapan dan keamanan masyarakat dalam menghadapi bencana alam.

ABSTRACT

Name : Muhammad Agil Ghifari, Taufik Pragusga, Zidan Kharisma Adidharma
Study Program : Computer Science
Title : Implementation of Event-Driven Architecture for Developing an Earthquake Early Warning System using Deep-Learning Model
Counsellor1 : Ari Wibisono. S.Kom., M.Kom

Keywords: Event-Driven Architecture, Earthquake Early Warning System, Deep Learning, Seismic Data, Hypocenter, Magnitude, Data Visualization, Distributed Systems.

This study focuses on the development of an earthquake early warning system utilizing event-driven architecture and deep-learning models. The aim is to model seismic data to detect initial waves, hypocenters, magnitude, and depth of earthquakes. Data from hundreds of seismograph points were collected and processed using deep-learning models to generate accurate predictions. The system is designed to provide visualizations and information to support the Meteorology, Climatology, and Geophysics Agency (BMKG) in detecting critical earthquake aspects. Additionally, a distributed system was developed to manage large-scale data requests and processing efficiently. An Application Programming Interface (API) is also presented for accessible and understandable data predictions. Finally, the integration of machine learning models with backend and frontend is designed to offer a user-friendly display. This research contributes to the development of a more sophisticated and responsive early warning system, enhancing public preparedness and safety in the face of natural disasters.

DAFTAR ISI

| | |
|---|----------|
| HALAMAN JUDUL..... | i |
| LEMBAR PERSETUJUAN..... | ii |
| LEMBAR PERNYATAAN ORISINALITAS..... | iii |
| LEMBAR PENGESAHAN..... | iv |
| KATA PENGANTAR..... | v |
| LEMBAR PERSETUJUAN PUBLIKASI ILMIAH..... | vii |
| ABSTRAK..... | ix |
| DAFTAR ISI..... | xi |
| DAFTAR GAMBAR..... | xv |
| DAFTAR TABEL..... | xii |
| 1. PENDAHULUAN..... | 1 |
| 1.1 Latar Belakang..... | 1 |
| 1.2 Rumusan Masalah..... | 3 |
| 1.3 Batasan Masalah..... | 3 |
| 1.4 Tujuan Penelitian..... | 4 |
| 1.5 Basis Penelitian..... | 4 |
| 1.6 Posisi Penelitian..... | 6 |
| 1.7 Sistematika Penulisan..... | 8 |
| 2. TINJAUAN PUSTAKA..... | 9 |
| 2.1 Earthquake Early Warning System (EEWS)..... | 9 |
| 2.2 Deep Learning..... | 9 |
| 2.2.1 Neural Network..... | 10 |
| 2.2.2 Convolutional Layer..... | 11 |
| 2.2.1 LSTM..... | 11 |
| 2.2.1 U-Net..... | 12 |
| 2.3 Seismologi..... | 13 |
| 2.3.1 Gelombang P dan Gelombang S..... | 13 |
| 2.3.2 Parameter-Parameter Gempa..... | 14 |
| 2.4 Sistem Back-End..... | 14 |
| 2.4.1 HTTP..... | 15 |
| 2.4.2 JSON..... | 15 |
| 2.4.3 API..... | 16 |
| 2.4.4 REST API..... | 17 |
| 2.4.5 WebSocket..... | 17 |
| 2.4.6 Database..... | 18 |
| 2.4.7 Microservices..... | 19 |
| 2.4.8 Arsitektur Event-Driven..... | 20 |

| | |
|---|-----------|
| 2.4.9 Message Broker..... | 21 |
| 2.4.10 Docker..... | 24 |
| 2.5 Data Seismik dan Sumber Data..... | 25 |
| 2.5.1 FDSN Web Service..... | 26 |
| 2.5.2 SeedLink..... | 26 |
| 2.5.3 ObsPy..... | 27 |
| 2.6 SeisComP | 27 |
| 2.7 Web-Based Application..... | 29 |
| 2.7.1 User Interface and Experience..... | 30 |
| 3. METODOLOGI..... | 31 |
| 3.1 Pembagian Ruang Lingkup Penelitian..... | 31 |
| 3.2 Metodologi Umum..... | 32 |
| 3.2.1 Tahapan Penelitian..... | 33 |
| 4. IMPLEMENTASI..... | 39 |
| 4.1 Implementasi Machine Learning..... | 39 |
| 4.1.1 Requirements Model..... | 40 |
| 4.1.2 Deskripsi Dataset..... | 42 |
| 4.1.3 Preprocessing Data..... | 44 |
| 4.1.4 Labelling Data..... | 52 |
| 4.1.5 Arsitektur Model Deep Learning..... | 54 |
| 4.1.6 Proses Training..... | 58 |
| 4.1.7 Output Hasil Inferensi..... | 60 |
| 4.1.8. Postprocessing Hasil..... | 61 |
| 4.1.9. Deployment Model ML..... | 65 |
| 4.1.10 Hipotesis Penulis tentang Karakteristik Data yang telah di-Preprocess... | 67 |
| 4.2 Implementasi Backend..... | 68 |
| 4.2.1 Pemilihan Teknologi..... | 70 |
| 4.2.1 Arsitektur Sistem Backend..... | 72 |
| 4.2.2 Desain Data JSON..... | 76 |
| 4.2.3 Desain Topik Kafka..... | 77 |
| 4.2.4 Implementasi Layanan Producer..... | 78 |
| 4.2.5 Implementasi Layanan Queue..... | 81 |
| 4.2.6 Implementasi Layanan Picker..... | 83 |
| 4.2.6 Implementasi Layanan REST API dan WebSocket..... | 86 |
| 4.3 Implementasi Frontend..... | 89 |
| 4.3.1 Pemilihan Teknologi..... | 90 |
| 4.3.2 Arsitektur Aplikasi Web..... | 91 |
| 4.3.3 Komunikasi dengan Backend..... | 94 |
| 4.3.4 Manajemen State atau Data..... | 96 |

| | |
|--|------------|
| 4.3.5 Visualisasi Data..... | 98 |
| 4.3.6 Hasil Implementasi Aplikasi Web..... | 100 |
| 5. EVALUASI SISTEM..... | 106 |
| 5.1 Evaluasi Machine Learning..... | 106 |
| 5.1.1 Metodologi Evaluasi..... | 105 |
| 5.1.2 Hasil Evaluasi Offset Picking..... | 106 |
| 5.1.3 Hasil Evaluasi F1 Score..... | 107 |
| 5.1.4 Hasil Evaluasi Jumlah Parameter..... | 107 |
| 5.1.5 Hasil Evaluasi Model Parameter Gempa..... | 108 |
| 5.2 Evaluasi Sistem Backend..... | 108 |
| 5.2.1 Metodologi Evaluasi..... | 110 |
| 5.2.2 Hasil Evaluasi Layanan Producer..... | 111 |
| 5.2.3 Hasil Evaluasi Layanan Queue..... | 112 |
| 5.2.4 Hasil Evaluasi Layanan Picker..... | 113 |
| 5.2.5 Hasil Evaluasi Layanan WebSocket dan REST API..... | 115 |
| 5.2.6 Hasil Evaluasi Keseluruhan..... | 116 |
| 5.3 Evaluasi Frontend..... | 117 |
| 5.3.1 Hasil Evaluasi Uji Skenario..... | 118 |
| 5.3.2 Hasil Evaluasi Penilaian BMKG..... | 119 |
| 6. KESIMPULAN DAN SARAN..... | 121 |
| 6.1 Kesimpulan..... | 121 |
| 6.2 Saran..... | 122 |
| DAFTAR REFERENSI..... | 123 |

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 1.1: Diagram Basis Penelitian..... | 5 |
| Gambar 1.2: Pembagian Tugas Sistem EEWs..... | 7 |
| Gambar 2.1: Struktur Neural Network Dense Dasar..... | 10 |
| Gambar 2.2: Ekstraksi Fitur dari CNN Layer..... | 11 |
| Gambar 2.3: Struktur dari Satu Unit LSTM..... | 12 |
| Gambar 2.4: Struktur Dasar U-Net..... | 13 |
| Gambar 2.5: Cara Kerja REST API..... | 17 |
| Gambar 2.6: Ilustrasi Cara Kerja WebSocket..... | 18 |
| Gambar 2.7: Ilustrasi Arsitektur Microservices..... | 20 |
| Gambar 2.8: Ilustrasi Arsitektur Event-Driven..... | 21 |
| Gambar 2.9: Ilustrasi Cara Kerja Message Broker..... | 22 |
| Gambar 2.10: Arsitektur Apache Kafka..... | 22 |
| Gambar 2.11: Akuisisi Data Real-Time, Pemrosesan Data, dan Analisa Data dalam SeisComP..... | 28 |
| Gambar 2.12: Modul SCRTTV..... | 29 |
| Gambar 4.1: Tahapan Implementasi model Machine Learning..... | 39 |
| Gambar 4.2: Distribusi Gempa Dataset ETHZ..... | 42 |
| Gambar 4.3: Distribusi Kejadian Gempa dalam Dataset ETHZ..... | 43 |
| Gambar 4.4: Distribusi Magnitudo Gempa dalam Dataset ETHZ..... | 43 |
| Gambar 4.5: Pipeline Preprocessing Data..... | 45 |
| Gambar 4.6: Data yang Diinputkan ke dalam Pipeline..... | 45 |
| Gambar 4.7: Output fungsi Slope dalam pipeline..... | 45 |
| Gambar 4.8: Output fungsi Square dalam pipeline..... | 46 |
| Gambar 4.9: Output fungsi AddChannels dalam pipeline..... | 47 |
| Gambar 4.10: Output fungsi Log1P dalam pipeline..... | 48 |
| Gambar 4.11: Output fungsi MultiExponentialSmoothing dalam pipeline..... | 49 |
| Gambar 4.12: Output fungsi PairwiseRatio dalam pipeline..... | 50 |
| Gambar 4.13: Output salah satu channel dari fungsi SlidingWindow dalam pipeline.... | 51 |
| Gambar 4.14: Pelabelan Mask P & S Wave Arrival..... | 53 |
| Gambar 4.15: Arsitektur Model Pendekripsi Kedatangan Gelombang P & S..... | 55 |
| Gambar 4.16: Unit Encoder..... | 56 |
| Gambar 4.17: Bagian Tengah..... | 57 |
| Gambar 4.18: Unit Decoder..... | 57 |
| Gambar 4.19: Arsitektur Model Aproksimasi Parameter Gempa..... | 58 |
| Gambar 4.20: Proses Training..... | 59 |
| Gambar 4.21: Hasil Inferensi Model Deteksi Kedatangan Gelombang P dan S..... | 60 |
| Gambar 4.22: Ilustrasi Penentuan Lokasi Episentrum Gempa dari Tiga Stasiun..... | 63 |

| | |
|--|-----|
| Gambar 4.23: Ilustrasi Algoritma Pencari Episentrum..... | 64 |
| Gambar 4.24: Penentuan Perwakilan dari Lingkaran yang Tidak Berpotongan..... | 64 |
| Gambar 4.25: Karakteristik Distribusi Nilai di Sliding Windows untuk Setiap Fasa.... | 67 |
| Gambar 4.26: Proses Pembuatan Sistem Backend..... | 69 |
| Gambar 4.27: Rancangan Arsitektur EEWS..... | 73 |
| Gambar 4.28: Alur Kerja Sistem..... | 74 |
| Gambar 4.29: Interaksi antar Layanan Melalui Topik Kafka..... | 76 |
| Gambar 4.30: Alur Kerja Layanan Producer..... | 79 |
| Gambar 4.31: Pendistribusian Data oleh Producer..... | 80 |
| Gambar 4.32: Alur Kerja Layanan Queue..... | 81 |
| Gambar 4.33: Proses Konsumsi Data oleh Queue..... | 82 |
| Gambar 4.34: Proses Distribusi Data oleh Queue..... | 82 |
| Gambar 4.35: Alur Kerja Layanan Picker..... | 84 |
| Gambar 4.36: Proses Konsumsi Data oleh Picker..... | 85 |
| Gambar 4.37: Alur Teknis Pemrograman Aplikasi Web..... | 90 |
| Gambar 4.38: Tampilan Low Fidelity Aplikasi Web..... | 92 |
| Gambar 4.39: Ilustrasi User Flow Aplikasi Web..... | 93 |
| Gambar 4.40: Ilustrasi Komunikasi Data Aplikasi Web..... | 95 |
| Gambar 4.41: Ilustrasi Komunikasi Data Websocket Aplikasi Web..... | 95 |
| Gambar 4.42: Ilustrasi Manajemen State pada React Component Tree..... | 97 |
| Gambar 4.43: Peta Interaktif Tanpa Data..... | 98 |
| Gambar 4.44: Line Chart..... | 99 |
| Gambar 4.45: Ilustrasi Intersection Observer..... | 100 |
| Gambar 4.46: Halaman Web Tanpa Data Waveform..... | 101 |
| Gambar 4.47: Halaman Web dengan Data Waveform..... | 101 |
| Gambar 4.48: Popup Informasi Stasiun (kiri) dan Parameter Gempa (kanan)..... | 102 |
| Gambar 4.49: Panel Player Playback (kiri) dan Panel Player Live (kanan)..... | 102 |
| Gambar 4.50: Panel Arrival Pick (kiri) dan Panel Arrival Event (kanan)..... | 103 |
| Gambar 4.51: Panel Stations (kiri) dan Panel Statistics (kanan)..... | 104 |
| Gambar 4.52: Line Chart dengan Prediksi P-Wave..... | 105 |

DAFTAR TABEL

| | |
|--|-----|
| Tabel 3.1: Pembagian Identifikasi Masalah..... | 31 |
| Tabel 3.2: Tahapan Penelitian Secara Umum..... | 33 |
| Tabel 4.1: Spesifikasi Input dan Output Model..... | 40 |
| Tabel 4.2: Skema Data JSON Endpoint /predict..... | 66 |
| Tabel 4.3: Skema Data JSON Endpoint /approx_earthquake_statistics..... | 66 |
| Tabel 4.4: Skema Data JSON Endpoint /recalculate..... | 67 |
| Tabel 4.5: Skema Data JSON..... | 76 |
| Tabel 4.6: Deskripsi Topik Kafka..... | 77 |
| Tabel 4.7: Topik, Partisi, dan Replika..... | 78 |
| Tabel 4.8: Endpoint Producer..... | 78 |
| Tabel 4.9: Endpoint REST API..... | 86 |
| Tabel 4.10: WebSocket Response..... | 88 |
| Tabel 4.11: Penjelasan State Global..... | 96 |
| Tabel 5.1: Hasil Evaluasi End-to-end..... | 106 |
| Tabel 5.2: Hasil Evaluasi Offset Picking..... | 106 |
| Tabel 5.3: Hasil Evaluasi F1 Score..... | 107 |
| Tabel 5.4: Hasil Jumlah Parameter..... | 107 |
| Tabel 5.5: Hasil Evaluasi RMSE Setiap Model Parameter..... | 108 |
| Tabel 5.6: Spesifikasi Server Penguji..... | 110 |
| Tabel 5.7: Hasil Evaluasi Producer..... | 111 |
| Tabel 5.8: Hasil Evaluasi Queue..... | 112 |
| Tabel 5.9: Hasil Evaluasi Picker..... | 113 |
| Tabel 5.10: Hasil Evaluasi WebSocket dan REST API..... | 115 |
| Tabel 5.11: Hasil Uji Skenario..... | 118 |

BAB 1

PENDAHULUAN

Pada bab pendahuluan dijelaskan latar belakang penelitian, termasuk topik penelitian, masalah yang akan dipecahkan, batasan masalah, tujuan akhir yang ingin dicapai dari penelitian yang dilakukan, posisi penelitian. Kemudian, penulis menjelaskan mengenai pendekatan penulisan yang metodis yang digunakan dalam laporan ini.

1.1 Latar Belakang

Gempa bumi merupakan salah satu bencana alam yang memiliki dampak yang sangat signifikan terhadap kehidupan manusia dan lingkungan. Indonesia, sebagai negara yang terletak pada Cincin Api Pasifik, merupakan salah satu daerah yang paling rentan terhadap risiko gempa bumi. Dalam beberapa tahun terakhir, Indonesia telah menjadi saksi dari beberapa peristiwa gempa besar yang mengakibatkan kerugian besar baik dalam hal kerusakan fisik maupun korban manusia. Oleh karena itu, pengembangan earthquake early warning system (EEWS) menjadi sangat penting untuk meminimalkan dampak negatif dari gempa bumi.

Sayangnya, hingga saat ini, Indonesia belum memiliki sistem EEWS yang dapat memberikan peringatan dini kepada masyarakat sebelum terjadinya gempa. Meskipun telah ada upaya dari peneliti sebelumnya untuk mengembangkan sistem pendekripsi gempa dengan informasi kedatangan gempa, kedalaman, magnitudo, dan lokasi secara real-time, tetapi masih terdapat keterbatasan-keterbatasan tertentu dalam penelitian tersebut.

Penulis melanjutkan penelitian sebelumnya untuk membuat sistem prediksi P wave, magnitudo, lokasi, dan kedalaman gempa (Wahidi et al., 2022). Salah satu kendala utama dari penelitian sebelumnya adalah jumlah titik seismograf yang digunakan untuk pendekripsi dini. Penelitian sebelumnya hanya memanfaatkan tiga titik seismograf dalam analisisnya, padahal sebenarnya Indonesia memiliki sekitar lima ratus titik seismograf yang tersebar di seluruh wilayahnya. Hal ini mengakibatkan kurangnya representasi data yang akurat dan komprehensif dalam analisis gempa bumi. Selain itu, proses preprocessing data dan pendekripsi P wave (gelombang primer yang pertama kali terdeteksi saat gempa) dilakukan secara terpisah, sehingga tidak

menggambarkan kondisi sesungguhnya karena data harus diproses secara real-time dari data kotor yang berbentuk MSEED

Oleh karena itu, diperlukan upaya untuk memperbaiki dan menyempurnakan sistem pendekripsi dini gempa yang ada. Penelitian ini akan mengambil langkah lebih maju dengan mengintegrasikan konsep machine learning dalam sistem pendekripsi gempa. Selain itu, penelitian ini juga akan mengadopsi pendekatan arsitektur microservice dalam pembangunan sistem, untuk mengatasi tantangan dalam mengelola volume besar permintaan (request) per detik serta memastikan sistem dapat dipelihara dengan baik.

Dalam konteks tersebut, penelitian ini bertujuan untuk mengembangkan sebuah sistem pendekripsi-dini gempa berbasis web yang mengintegrasikan teknologi microservice, machine learning, dan penggunaan data seismik MSEED (MiniSEED) dalam memprediksi terjadinya gempa bumi secara lebih akurat dan real-time. Dengan melakukan penyempurnaan atas keterbatasan-keterbatasan penelitian sebelumnya, diharapkan hasil penelitian ini dapat memberikan kontribusi signifikan dalam memitigasi dampak buruk dari gempa bumi di Indonesia serta dapat menjadi referensi bagi perkembangan teknologi serupa di negara-negara lain yang juga rentan terhadap risiko gempa.

Perancangan sistem didasarkan pada penggunaan microservices untuk mencapai skalabilitas yang dibutuhkan untuk menangani lebih dari 500 sensor seismograf. Adapun untuk menangani itu, dibutuhkan pendekatan event-driven yang bisa menghubungkan antar microservices yang bekerja. Arsitektur event-driven dapat melepas ketergantungan antar sistem dengan menggunakan event dan message broker sebagai mediator untuk komunikasi (Stopford, 2018). Sebuah event akan dikirimkan dari microservice ke dalam message broker untuk mensimulasikan stasiun seismograf. Data tersebut akan diterima oleh sekumpulan microservice yang melakukan teknik preprocessing dan interpolasi. Kemudian, akan dikirim lagi ke message broker ke servis yang mendekripsi P wave dan mengirimkan sinyal ke frontend melalui websocket. Apabila terdeteksi P wave, maka akan dilakukan analisa oleh model Machine Learning untuk memperoleh data kedalaman, lokasi, dan magnitude.

1.2 Rumusan Masalah

Berdasarkan informasi yang telah diuraikan sebelumnya, penulis mengambil kesimpulan mengenai beberapa isu yang akan diinvestigasi dalam penelitian ini, antara lain:

1. Bagaimana cara membuat sebuah model *machine learning* yang dapat mengidentifikasi kedatangan gelombang p dan s, magnitudo, jarak, dan kedalaman gempa dari data seismogram?
2. Bagaimana performansi model yang dibuat jika dibandingkan dengan model PhaseNet dalam melakukan picking pada gelombang p dan s?
3. Bagaimana cara mendapatkan data seismik untuk diproses dengan arsitektur *event-driven* sehingga dapat ditampilkan kepada pengguna akhir?
4. Bagaimana sistem dapat menentukan parameter gempa, seperti episenter, kedalaman, dan magnitudo?
5. Bagaimana cara membangun sistem pendekripsi gempa dengan arsitektur *event-driven*?
6. Bagaimana cara menampilkan data gelombang gempa dari banyak seismograf dengan format mudah dimengerti dan efisien?

1.3 Batasan Masalah

Cakupan penelitian ini didefinisikan sebagai berikut:

1. Dataset yang digunakan untuk proses training adalah dataset gempa ETHZ yang terdiri atas kumpulan gempa yang terjadi di Swiss pada rentang tahun 2013 hingga 2021.
2. Data yang diproses oleh sistem berasal dari 26 stasiun yang merupakan berasal dari jaringan GEOFON. Masing-masing stasiun memiliki tiga *channel*, yaitu BHN, BHE, dan BHZ.
3. Penelitian ini menghasilkan Sistem Bukti Konsep (Proof of Concept, PoC) untuk deteksi awal gelombang, magnitudo, kedalaman, dan lokasi hiposenter gempa bumi secara *real-time*.

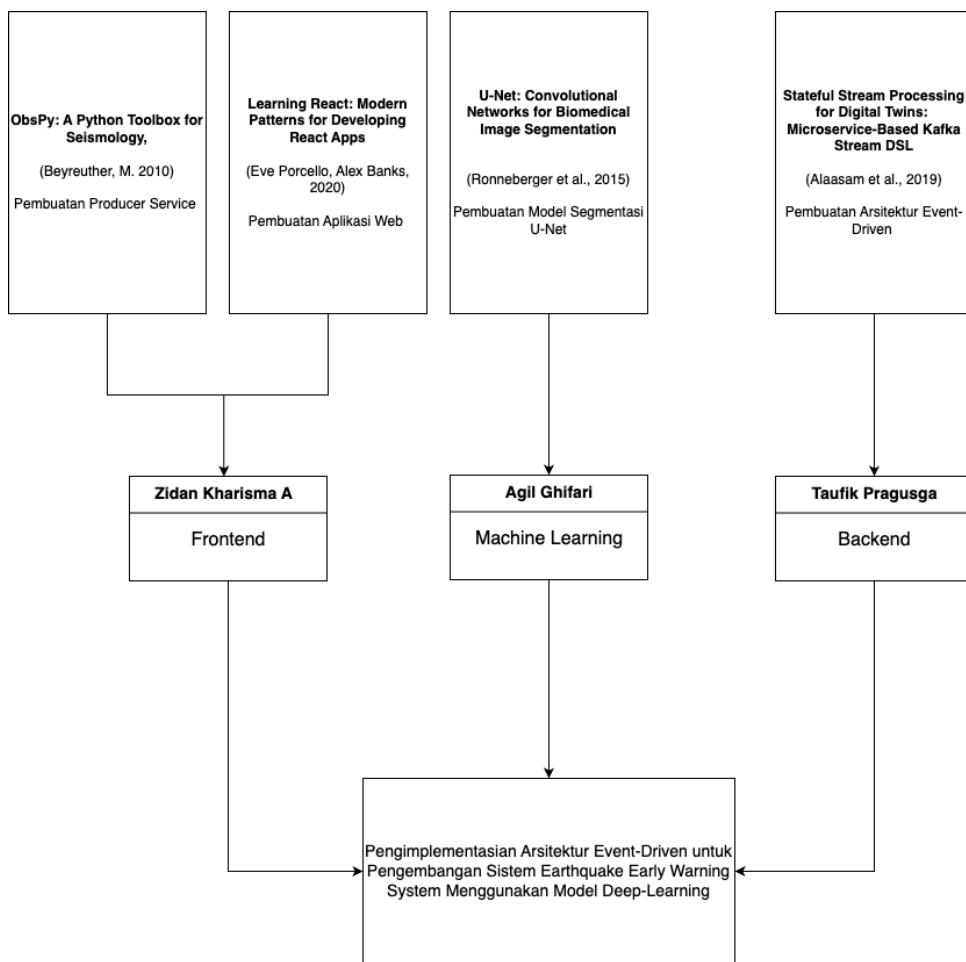
1.4 Tujuan Penelitian

Pelaksanaan penelitian dalam tugas akhir ini bertujuan akhir sebagai berikut:

1. Memodelkan data seismik untuk mendeteksi gelombang awal, hiposenter, magnitudo, dan kedalaman berdasarkan data dari ratusan titik seismograf.
2. Menciptakan visualisasi dan menyediakan informasi yang dapat memberikan dukungan kepada Badan Meteorologi, Klimatologi, dan Geofisika (BMKG) dalam hal mendeteksi gelombang awal, hiposenter, magnitudo, dan kedalaman gempa. Informasi ini akan disajikan dengan cara yang mudah dipahami, penuh informasi, dan interaktif.
3. Membangun sebuah sistem terdistribusi guna menangani permintaan dan pengolahan data dalam skala besar dengan kecepatan yang tinggi untuk kebutuhan sistem EEWS.
4. Menyajikan antarmuka pemrograman aplikasi (API) yang memungkinkan prediksi data yang diberikan oleh Badan Meteorologi, Klimatologi, dan Geofisika (BMKG).
5. Mengintegrasikan model machine learning dengan *backend*, serta menampilkan hasil prediksi data kepada frontend dalam tampilan yang ramah pengguna

1.5 Basis Penelitian

Sesuai dengan diagram pada Gambar 1.1, dalam pengembangan model ML, terkhusus untuk model deteksi kedatangan gelombang p/s, penulis mengacu pada sebuah penelitian yang dilakukan oleh Ronneberger et al. (2015) mengenai arsitektur yang mereka gunakan untuk mensegmentasi citra-citra medis yang disebut sebagai arsitektur U-Net. Penelitian ini menjelaskan tentang bagaimana konsep dasar dari bentuk arsitektur U-Net yang mereka buat. Model U-Net ternyata memiliki kemampuan yang sangat bagus dalam menjalankan tugas segmentasi, dan hingga saat ini banyak dari algoritma-algoritma segmentasi machine learning yang menggunakan U-Net sebagai basis dari arsitektur modelnya. Selain itu, data yang digunakan penulis untuk melatih model machine learning adalah data dari ETHZ Swiss yang mengandung informasi mengenai event gempa yang terjadi di Swiss dan sekitarnya pada rentang 2010 hingga 2020.



Gambar 1.1: Diagram Basis Penelitian

Sumber: (Olahan Penulis, 2023)

Dalam pengembangan arsitektur *event-driven* untuk EEWS, penulis mengacu pada penelitian oleh Alaasam et al. (2019) yang membahas pengolahan data *stream* berbasis *stateful* dalam konteks Digital Twins dengan menggunakan Kafka Stream DSL. Penelitian ini menawarkan wawasan mendalam mengenai pembangunan *microservices* yang *stateful*, yang esensial dalam menganalisis data manufaktur secara *real-time*. Studi ini menyoroti pentingnya arsitektur yang *loosely-coupled*, *fault tolerance*, *latency* dalam pengolahan, dan skalabilita. Hal tersebut merupakan aspek kunci dalam desain sistem EEWS penulis. Pendekatan *microservices* memberi penulis kerangka kerja untuk membangun sistem yang dapat menyesuaikan diri dengan bebas.

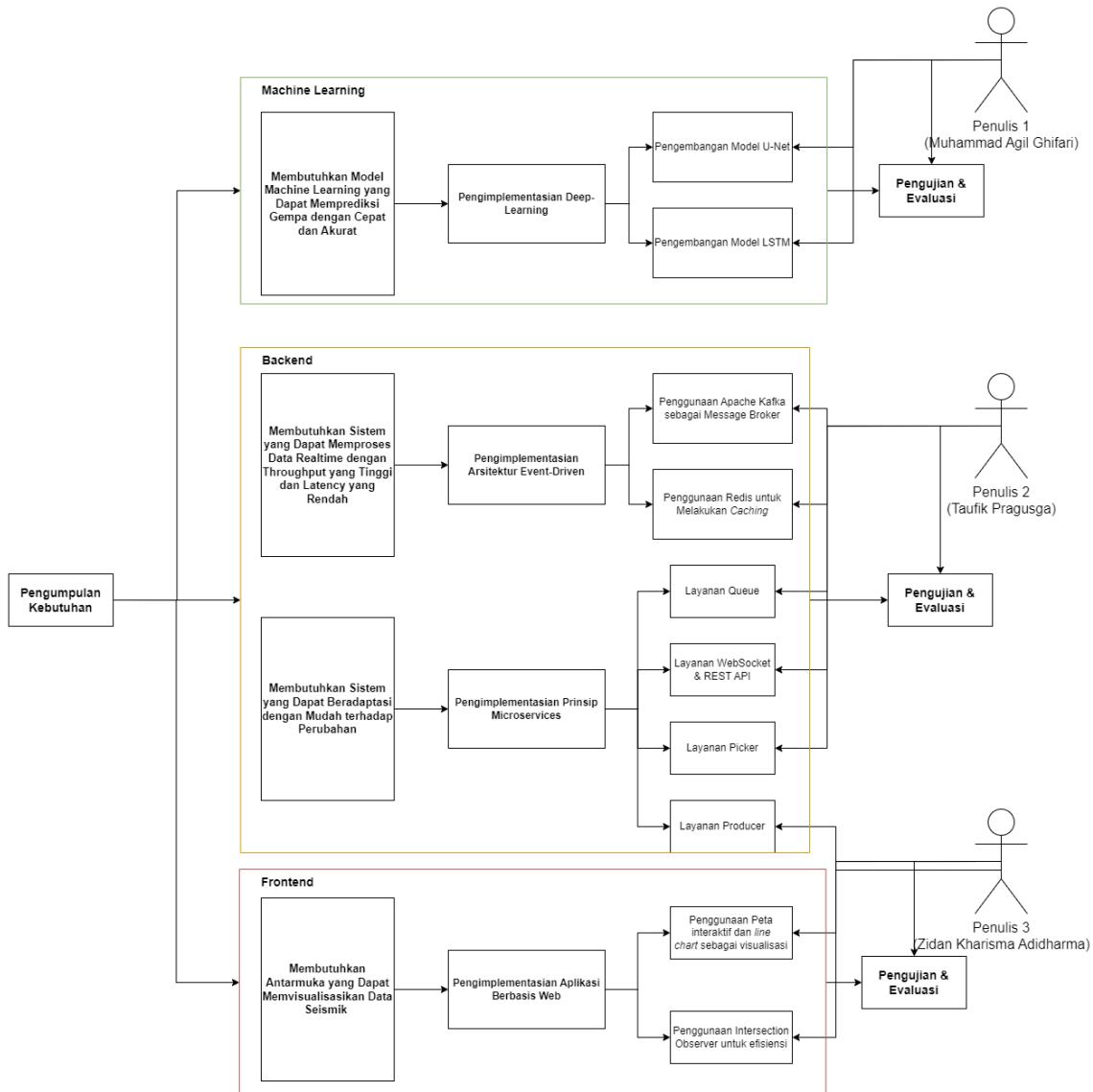
Dalam pengembangan producer service, dibutuhkan menghubungkan service tersebut dengan protokol standar SEEDLINK dan standar FDSN Web Service. Penulis mengacu pada penelitian dan proyek oleh Beyreuther, M. et al (2010) yang menjelaskan mengenai pengembangan library ObsPy. Penelitian, atau hasil dari penelitian ini, memberi penulis kemudahan untuk mengimplementasikan berbagai kebutuhan untuk berbagi dan memproses data seismologis hanya dengan satu *library*.

Untuk pengembangan aplikasi berbasis web, penulis mengacu pada buku karya Eve Porcello, dan Alex Banks (2020). Pada buku ini dijelaskan mengenai pengembangan aplikasi dengan React.js memanfaat design pattern dan best practices yang dapat diterapkan dalam penelitian ini

1.6 Posisi Penelitian

Terdapat banyak bagian yang perlu dikembangkan dan diimplementasikan dalam membuat sistem EEWs ini. Untuk dapat mengembangkan setiap bagian tersebut, diperlukan pembagian tugas yang jelas dalam mengerjakan komponen-komponen yang ada di sistem ini seperti digambarkan pada Gambar 1.2. Berikut ini adalah pembagian tugas antara para penulis dalam konteks studi literatur.

1. Muhammad Agil Ghifari sebagai Penulis 1, berfokus pada pengembangan model machine learning, membangun dan mengevaluasi model LSTM dan U-Net untuk interpretasi data seismik yang akurat.
2. Taufik Pragusga sebagai Penulis 2, mengambil tanggung jawab dalam desain dan implementasi *arsitektur event-driven backend* menggunakan Apache Kafka, pengintegrasian Redis untuk caching, serta pembuatan layanan Queue, Picker, WebSocket, dan REST API.
3. Zidan Kharisma Adidharma sebagai Penulis 3, bertugas mengembangkan frontend, termasuk pemanfaatan peta interaktif dan *line chart* untuk visualisasi data seismik, serta optimasi dengan *intersection observer* untuk efisiensi. Kemudian, juga bertanggung jawab mengembangkan layanan Producer



Gambar 1.2: Pembagian Tugas Sistem EEWs

Sumber: (Olahan Penulis, 2023)

Setiap penulis akan melakukan pengujian dan evaluasi terhadap bagian besar yang menjadi tanggung jawabnya. Setiap penulis memiliki peran spesifik dalam pengembangan sistem EEWs, dengan fokus yang terdefinisi baik dalam penelitian dan implementasi.

1.7 Sistematika Penulisan

Dalam penelitian ini, terdapat pembatasan pada cakupan penelitian yang dilakukan, yang meliputi:

1. Bab 1 yang berjudul Pendahuluan mengulas tentang latar belakang dilakukannya penelitian ini, termasuk merinci pertanyaan-pertanyaan penelitian, tujuan, serta pembatasan-pembatasan yang ada dalam penelitian.
2. Bab 2, Tinjauan Pustaka, mengulas penjelasan tentang konsep-konsep yang diterapkan serta mendefinisikan istilah-istilah yang digunakan.
3. Bab 3, Metodologi, membahas tentang berbagai metode yang dipakai untuk mengatasi masalah yang dibahas dalam penelitian ini serta metodologi yang diaplikasikan secara umum.
4. Bab 4, Implementasi membahas secara detail pengimplementasian komponen-komponen sistem, yaitu *machine learning*, *backend*, dan *frontend*
5. Bab 5, Evaluasi, menampilkan hasil evaluasi dari setiap komponen setelah dilakukan pengujian dengan berbagai metrik
6. Bab 6, Kesimpulan dan Saran, berisi kesimpulan dari penelitian serta saran untuk penelitian sistem pendeksi dini gempa di masa depan

BAB 2

TINJAUAN PUSTAKA

Sebelum memulai suatu penelitian, perlu dibangun kerangka teori yang sesuai untuk menyelesaikan masalah tersebut. Untuk membentuk kerangka teori yang tepat, harus terkait dengan temuan dari tinjauan literatur yang telah dilakukan. Oleh karena itu, temuan dari tinjauan literatur yang berkaitan dengan parameter penelitian ini dijelaskan dalam bab ini.

2.1 *Earthquake Early Warning System (EEWS)*

Earthquake early warning system atau sistem peringatan dini gempa bumi adalah sistem yang memberi peringatan kepada orang-orang sebelum terjadi gempa (Allen et al., 2009). Untuk memungkinkan respon cepat dalam mengurangi kerusakan akibat gempa, sistem ini bekerja dengan cepat mendeteksi energi yang dipancarkan dari patahan gempa dan menilai guncangan tanah yang akan terjadi.

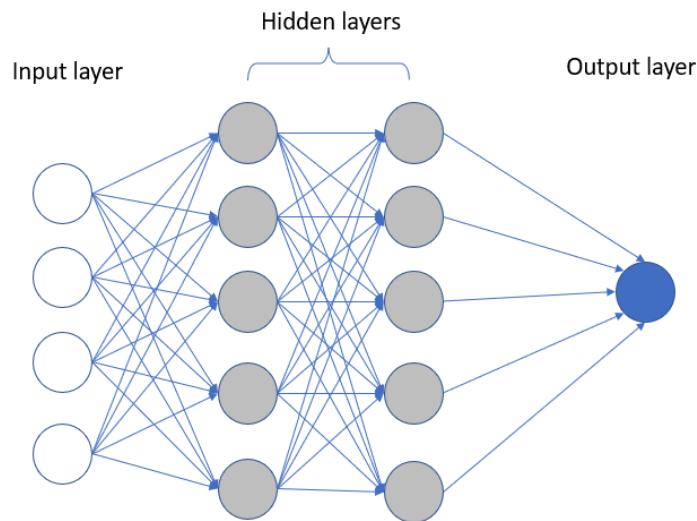
Sistem ini sangat penting bagi orang-orang yang tinggal di tempat-tempat rawan gempa karena menyediakan detik atau menit penting bagi orang-orang dan sistem untuk bersiap menghadapi guncangan yang akan datang, yang mungkin dapat mencegah kematian dan meminimalkan kerusakan properti. Sistem ini bekerja dengan cepat mendeteksi energi yang dipancarkan dari patahan gempa dan menilai guncangan tanah yang akan terjadi, memungkinkan respons cepat dalam mengurangi kerusakan akibat gempa.

2.2 Deep Learning

Deep learning adalah sekelompok algoritma pembelajaran mesin yang memanfaatkan rangkaian unit pemrosesan non-linear yang disebut sebagai neuron yang disusun dalam arsitektur tertentu (biasanya dalam bentuk lapisan/*layer*) untuk mengekstraksi dan mentransformasi fitur pada tingkat abstraksi yang berbeda-beda (L. Deng et al, 2014). Setiap lapisan berturut-turut menggunakan hasil keluaran dari lapisan sebelumnya sebagai input dan akan menghasilkan nilai baru yang merupakan abstraksi yang lebih *high-level* dari lapisan sebelumnya.

2.2.1 Neural Network

Sebuah *neural network* (NN), atau sering juga disebut sebagai *artificial neural network* (ANN) adalah sebuah model yang menirukan struktur dan tingkah laku dari jaringan saraf biologis. Seperti pada Gambar 2.1, *neural network* terdiri dari sekelompok *neuron* yang saling terhubung dengan satu sama-lain (D'Addona, 2014). sebuah neuron dapat menerima satu tensor input, melakukan transformasi linear terhadap vektor input, dan memasukkan setiap nilai dari vektor hasil transformasi ke dalam fungsi non-linear sehingga dihasilkan sebuah output.



Gambar 2.1: Struktur Neural Network Dense Dasar

Sumber: (Olahan Penulis, 2023)

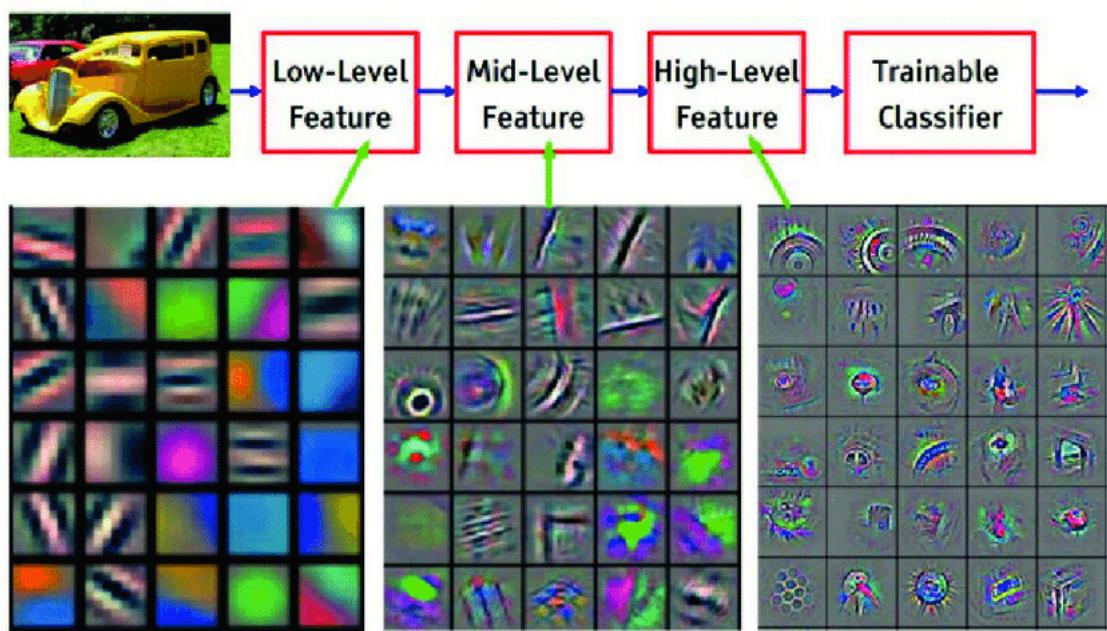
Secara matematis, sebuah neuron dapat dinyatakan sebagai sebuah fungsi $z(x)$ yang didefinisikan sebagaimana persamaan di bawah ini.

$$z(x) = f(x \cdot w^T)$$

Untuk x adalah sebuah input vektor, w adalah sebuah vektor beban, dan f adalah sebuah fungsi aktivasi non-linear. Kombinasi dari neuron-neuron ini dapat menghasilkan fungsi yang lebih kompleks untuk lapisan yang lebih dalam.

2.2.2 Convolutional Layer

Convolutional layer adalah sebuah arsitektur *neural network* yang menggunakan operasi konvolusi pada vektor input dengan menggunakan sebuah kernel yang biasanya berukuran jauh lebih kecil daripada ukuran input (O'Shea, 2015). Dalam layer ini, kernel adalah hal yang dicari tahu. *Convolutional layer* seringkali digunakan untuk ekstraksi fitur pada data-data tak terstruktur seperti gambar dan audio. Kernel-kernel dari *convolutional layer* ini dapat dilatih pada proses training sehingga kernel-kernel ini dapat menangkap pola-pola tertentu pada data.



Gambar 2.2: Ekstraksi Fitur dari CNN Layer

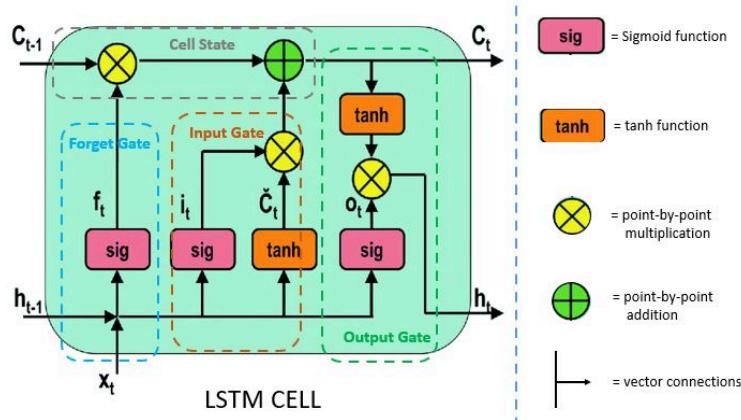
Sumber: (<https://www.researchgate.net/profile/Terje-Midtbo>, 2023)

Susunan layer ini dapat menghasilkan ekstraksi fitur yang makin *high-level* pada setiap layer-nya.

2.2.1 LSTM

LSTM (*long-short term memory*) adalah sebuah unit yang dapat mengolah data yang berbentuk sekuensial. Sesuai dengan Gambar 2.3, LSTM terdiri dari atas beberapa komponen yang disebut sebagai *gate*, yaitu *forget gate*, *input gate*, dan *output gate* (Greff et al. 2017). Dalam layer ini, pada suatu waktu di sebuah data sekuensial, nilai yang dikeluarkan oleh lapisan LSTM ini adalah dependen terhadap nilai-nilai yang

berada di sebelumnya. Agar bisa dihasilkan luaran yang bisa memiliki tingkah laku sesuai seperti yang kita inginkan, maka kita harus melatih nilai dari tensor-tensor yang terdapat di dalam setiap gate yang ada di dalam unit LSTM.

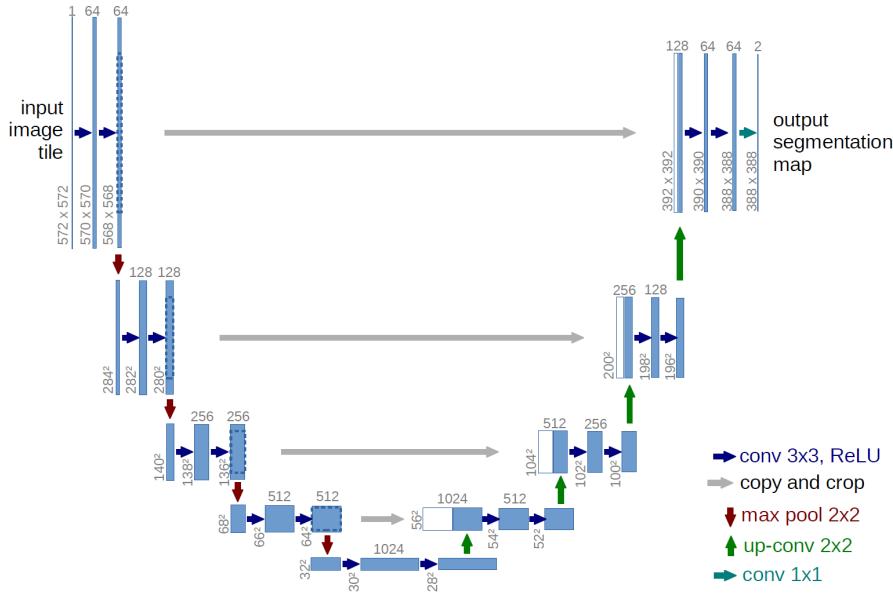


Gambar 2.3: Struktur dari Satu Unit LSTM

Sumber: (<https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn>, 2023)

2.2.1 U-Net

Dalam menghadapi *task* segmentasi citra medis, penelitian terus dilakukan untuk mengembangkan jaringan saraf tiruan. Salah satu inovasi signifikan adalah U-Net, diperkenalkan oleh Ronneberger et al. pada tahun 2015. U-Net awalnya didesain khusus untuk tugas segmentasi citra medis namun juga menunjukkan keunggulan dalam aplikasi umum. Arsitektur U-Net terdiri dari encoder dan decoder, dengan jalur langsung antara keduanya untuk mempertahankan informasi spasial tingkat tinggi seperti diilustrasikan pada Gambar 2.4. Struktur U-Net melibatkan lapisan konvolusi kontraktif untuk ekstraksi fitur dan lapisan konvolusi ekspansif untuk merekonstruksi citra. Keunggulan U-Net termasuk kemampuan untuk dilatih dengan dataset relatif kecil.



Gambar 2.4: Struktur Dasar U-Net

Sumber: (<https://lmb.informatik.uni-freiburg.de/>, 2015)

Kontribusi utama dari paper Ronneberger et al. adalah pengembangan arsitektur efektif untuk segmentasi citra medis, menunjukkan hasil superior pada saat penerbitan. Pendekatan U-Net mengatasi masalah hilangnya informasi selama segmentasi dan memfasilitasi pelatihan dengan dataset yang terbatas. Oleh karena itu, pemahaman konsep U-Net dari paper ini menjadi dasar teoretis yang kuat untuk pengembangan metode segmentasi citra medis dalam penelitian ini.

2.3 Seismologi

Pada sub bab ini, penulis akan menjelaskan aspek seismologi yang digunakan dalam pengembangan sistem ini. Aspek tersebut meliputi gelombang P dan gelombang S yang terjadi saat gempa, dan parameter gempa seperti lokasi, magnitudo, dan kedalaman.

2.3.1 Gelombang P dan Gelombang S

Dalam konteks gempa bumi, gelombang seismik dikelompokkan menjadi dua jenis utama: gelombang primer (P) dan gelombang sekunder (S). Gelombang P, atau gelombang primer, merupakan gelombang longitudinal yang merambat melalui material bumi. Gelombang ini dapat merambat melalui berbagai jenis material, termasuk

padatan, cairan, dan gas. Gelombang P ditandai oleh perubahan tekanan dalam medium, dan karena sifat longitudinalnya, gelombang ini dapat merambat melalui inti bumi.

Sementara itu, gelombang S, atau gelombang sekunder, merupakan gelombang transversal yang merambat melalui material bumi. Gelombang ini menyebabkan partikel tanah bergerak secara lateral atau melintang terhadap arah perambatannya. Gelombang S hanya dapat merambat melalui material padat, dan karena itu, tidak dapat melewati inti bumi yang cair. Gelombang S memiliki kecepatan yang lebih lambat daripada gelombang P.

2.3.2 Parameter-Parameter Gempa

Dalam konteks penelitian ini, parameter gempa melibatkan data-data statistik yang memberikan gambaran menyeluruh tentang suatu kejadian gempa. Magnitudo merupakan salah satu parameter utama yang mengukur besarnya energi yang dilepaskan oleh gempa bumi. Magnitudo dapat memberikan indikasi seberapa besar dampak yang mungkin terjadi, dan umumnya diukur dengan skala Richter atau skala moment.

Selain magnitudo, parameter lokasi gempa mencakup koordinat geografis lintang dan bujur. Koordinat ini memberikan informasi tentang lokasi episenter gempa, yaitu titik permukaan bumi tepat di atas sumber gempa. Koordinat geografis ini penting untuk memetakan persebaran gempa dan menentukan wilayah yang mungkin terdampak.

Kedalaman gempa adalah parameter yang mengukur jarak vertikal dari pusat gempa ke permukaan bumi. Kedalaman ini berperan dalam menentukan seberapa dalam energi gempa bumi merambat dan sejauh mana dampaknya dapat dirasakan di permukaan. Kedalaman gempa dapat bervariasi, dan pemahaman terhadap nilai kedalaman ini penting dalam mengevaluasi potensi risiko dan dampak gempa terhadap suatu wilayah.

2.4 Sistem *Back-End*

Back-end yang terdiri dari beberapa bagian seperti basis data dan layanan web merupakan area dari aplikasi web yang menangani permintaan pengguna (Saldamli et al., 2021). Bagian ini mengelola logika, penyimpanan data, dan tugas-tugas sisi server

yang diperlukan agar program berjalan dengan lancar dan berfungsi sesuai dengan yang diharapkan

2.4.1 HTTP

HTTP (*Hypertext Transfer Protocol*) adalah teknologi penting untuk mentransfer data di internet. Dalam kelompok protokol internet, HTTP adalah protokol tingkat aplikasi yang dibuat untuk sistem informasi *hypermedia* yang terdistribusi dan bekerja sama (Berners-Lee et al., 1996). Karena HTTP adalah protokol umum yang tidak menyimpan data (*stateless*), ini memungkinkan pengguna mengambil dokumen web yang saling terhubung dan sumber lainnya.

HTTP sangat penting dalam pengembangan World Wide Web. Karena sifatnya yang tidak menyimpan data, server menangani setiap permintaan dari pengguna secara terpisah dan menghapus data sesi dalam prosesnya.

2.4.2 JSON

JSON (JavaScript Object Notation), yang awalnya berasal dari standar bahasa pemrograman ECMAScript, adalah format pertukaran data yang sederhana, berbasis teks, dan tidak tergantung pada bahasa pemrograman tertentu (Crockford, 2006). JSON terkenal karena mudah dipahami, sederhana, dan fleksibel dalam berbagai situasi pemrograman. Formatnya membuat serialisasi dan komunikasi data menjadi sederhana dan dapat dibaca baik oleh mesin maupun manusia seperti pada Kode 2.1. Format ini sangat populer untuk pertukaran data antara pengguna dan server dalam aplikasi web.

```
{
  "code": "TNTI",
  "lat": "0.7718",
  "long": "127.3667",
  "elevation": "43",
  "name": "GEOFON Station GEOFON Station Ternate, Indonesia",
}
```

Kode 2.1: Contoh Struktur JSON

Sumber: (Olahan Penulis, 2023)

Dua ide utama dari JSON adalah daftar nilai yang terurut (mirip dengan *array* di banyak bahasa pemrograman) dan kumpulan pasangan *key-value* (mirip dengan objek dalam berbagai bahasa pemrograman). Karena strukturnya yang sederhana dan mirip

dengan konstruksi yang dikenal dari bahasa-bahasa pemrograman, JSON mudah dibaca oleh mesin dan manusia.

Kemampuan JSON untuk merepresentasikan struktur data yang kompleks secara ringkas dan terorganisir adalah salah satu keunggulannya. Oleh karena itu, JSON menjadi alat yang sangat berharga dalam pengembangan web saat ini, di mana mengelola data dengan efektif sangat penting. Karena integrasinya yang mulus dengan berbagai teknologi web dan bahasa pemrograman, JSON telah menjadi standar industri untuk pertukaran data dalam aplikasi internet.

2.4.3 API

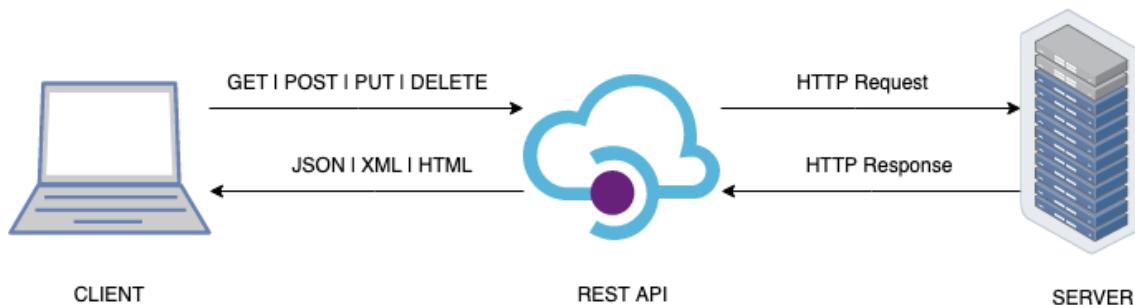
API adalah antarmuka untuk fungsi yang telah ditetapkan yang dapat digunakan oleh pemrogram untuk melakukan berbagai tugas (Robillard et al., 2009). Ini memungkinkan kode dapat digunakan kembali dan menawarkan abstraksi tingkat tinggi untuk mempermudah tugas pemrograman. Dengan berperan sebagai penghubung antara berbagai program perangkat lunak, API memfasilitasi komunikasi yang efektif dan efisien antara mereka yang mempercepat proses integrasi dan meningkatkan arsitektur perangkat lunak secara keseluruhan.

Dalam konteks sistem EEWS, API berperan penting dalam mengintegrasikan komponen sistem yang berbeda-beda seperti sensor seismik, server analitik, dan sistem visualisasi. API menyediakan serangkaian operasi yang sudah ditentukan sehingga para pengembang dapat dengan mudah meminta data seismik, memprosesnya, dan akhirnya menyajikan hasil analisis kepada pengguna atau sistem lain yang membutuhkan informasi tersebut.

Selain itu, API juga memungkinkan modularitas dalam sistem EEWS. Hal ini memungkinkan pemisahan yang jelas antara pengumpulan data, pemrosesan, dan presentasi, yang memudahkan pemeliharaan dan peningkatan sistem. Melalui API, perubahan pada satu bagian sistem, seperti peningkatan algoritma deteksi gempa dapat dilakukan tanpa mengganggu komponen lain. Ini membantu dalam memastikan bahwa sistem EEWS dapat berkembang sesuai dengan kemajuan teknologi tanpa memerlukan rekonfigurasi yang luas.

2.4.4 REST API

Dengan menggunakan HTTP dan JSON, REST (Representational State Transfer) adalah cara membuat API web yang membuat sistem bisa saling terhubung meski tidak erat (Ed-Doubi et al., 2018). Seperti yang divisualisasikan pada Gambar 2.5, REST ini unik karena tidak menyimpan data (*stateless*) dan menggunakan cara-cara umum komunikasi di HTTP seperti GET, POST, PUT, dan DELETE.



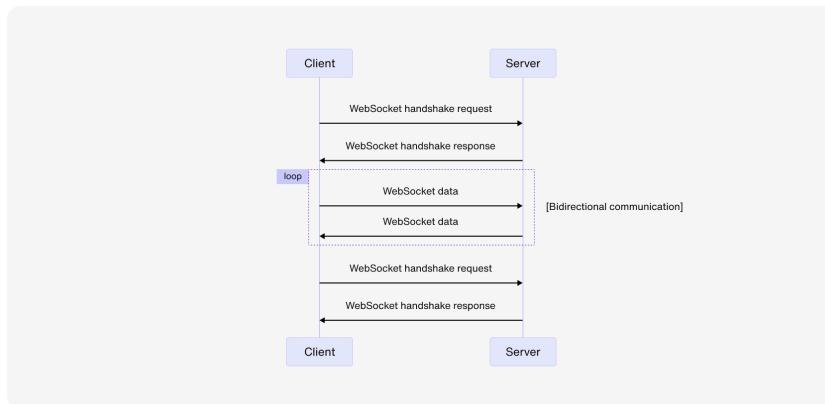
Gambar 2.5: Cara Kerja REST API

Sumber: (Olahan penulis, 2024)

REST API ini mudah untuk ditingkatkan ukurannya dan mudah digabungkan dengan berbagai layanan web karena sederhana dan fleksibel. RESTful API juga lebih mudah dipakai oleh para pengembang dan membuat pertukaran data antara pengguna dan server jadi lebih lancar karena biasanya menggunakan alamat web (URL) dan kode respon yang sudah umum. Karena cara ini efektif dan mudah digunakan, banyak layanan web yang dibuat dengan menggunakan REST API.

2.4.5 WebSocket

WebSocket adalah cara komunikasi standar yang dibangun di atas HTTP(S) dan memungkinkan komunikasi dua arah secara langsung antara server web dan klien, umumnya aplikasi peramban web biasa (Imre et al., 2006). Berbeda dari HTTP biasa yang tidak bisa melakukan koneksi terus-menerus dan interaktif, *websocket* bisa mengirimkan data bolak-balik tanpa henti antara permintaan (*request*) dan respons seperti pada Gambar 2.6. Teknologi ini sangat berguna untuk aplikasi yang perlu memperbarui data secara langsung dan cepat, seperti permainan *online*, sistem *chat* langsung, dan sistem pendekripsi suatu kejadian.



Gambar 2.6: Ilustrasi Cara Kerja WebSocket

Sumber: <https://sendbird.com/developer/tutorials/websocket-vs-http-communication-protocols>

2.4.6 Database

Seperti lemari arsip elektronik, *database* (basis data) yang terdiri dari perangkat keras dan lunak ini memungkinkan pengguna untuk menyimpan dan mengatur data mereka secara digital (Boucher and Yalcin, 2006). *Database SQL* yang biasa, seperti Microsoft SQL Server dan Oracle, punya aturan struktur yang ketat dan butuh bahasa khusus untuk mengubah data. Akan tetapi, ada juga *database NoSQL*, seperti Couchbase dan MongoDB yang lebih bebas dalam mengatur data. Ini cocok untuk data yang besar dan tersebar, serta beragam jenis data. Perbedaan ini menunjukkan bahwa database punya banyak cara penggunaan dan kebutuhan di dunia komputer sekarang ini.

2.4.6.1 MongoDB

Dikenal karena skalabilitas dan fleksibilitasnya, MongoDB adalah *database NoSQL* yang terkenal dapat digunakan untuk berbagai keperluan, mulai dari yang sederhana hingga yang rumit (Basren, 2019). Tidak seperti basis data relasional konvensional yang menyimpan data dalam dokumen mirip JSON (BSON), *document-oriented database* menawarkan cara yang lebih kuat dan alami untuk memodelkan data. Struktur ini memudahkan untuk mengubah skema data tanpa menyebabkan *downtime* dan mendukung berbagai *nested data*.

Salah satu fitur terpentingnya adalah skalabilitasnya. MongoDB dapat mendukung skalabilitas secara horizontal dengan membagi data di antara beberapa mesin melalui mekanisme *sharding*. MongoDB memiliki ketersediaan yang tinggi (*high*

availability) karena arsitektur set replikanya yang memelihara beberapa salinan data di berbagai server *database* untuk mendorong redundansi data dan toleransi kesalahan.

MongoDB cocok untuk berbagai aplikasi, seperti IoT, aplikasi seluler, dan *real-time analytics* karena kemampuan bahasa query dan pengindeksannya yang sangat baik. Karena banyak fiturnya, termasuk *ad-hoc queries*, agregasi, dan replikasi, *database* ini menjadi semakin populer di kalangan pengembang untuk berbagai aplikasi.

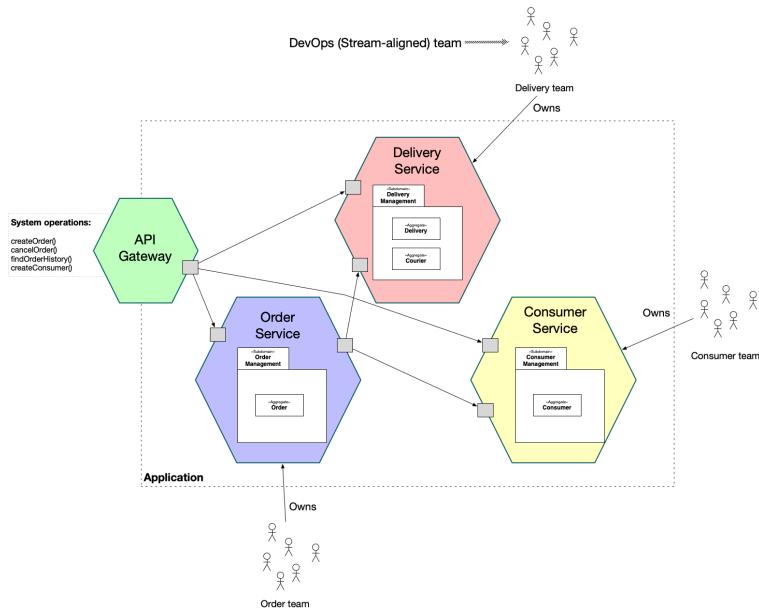
2.4.6.2 Redis

Redis adalah penyimpanan struktur data *in-memory* yang sangat efektif yang kebanyakan digunakan sebagai *message broker*, *database*, dan *cache* (Zhang et al., 2019). Karena didesain di dalam memori, Redis dapat melakukan operasi baca dan tulis dengan sangat cepat yang penting untuk aplikasi seperti *caching* yang perlu mengakses data dengan cepat.

Redis juga sangat baik dalam mengoptimalkan kinerja. Karena *throughput* yang lebih tinggi dan latensi respons yang lebih rendah, Redis sangat cocok untuk aplikasi yang membutuhkan kecepatan tinggi, seperti *game*, layanan keuangan, dan *real-time analytics*. Redis juga berkontribusi terhadap toleransi kesalahan dan daya tahan data dengan mendukung replikasi *master-slave* dan menyediakan berbagai tingkat persistensi di disk.

2.4.7 Microservices

Microservices adalah sebuah metode desain perangkat lunak yang berorientasi pada layanan (*service*) (Esposito et al., 2016). Seperti yang diilustrasikan pada Gambar 2.7, metode ini membagi kompleksitas sebuah aplikasi ke dalam unit-unit kecil yang dapat dikerjakan secara mandiri dan sangat fokus pada satu fungsi tertentu. Dengan menggunakan arsitektur *microservices*, aplikasi perangkat lunak dibagi menjadi unit-unit yang lebih kecil dan lebih mudah untuk dikelola. Setiap unit hanya bertanggung jawab pada satu fungsi atau layanan. Pendekatan ini memungkinkan lebih banyak modularitas, membuat setiap microservice bisa diperbesar skala dan diterapkan secara independen. Ini berlawanan dengan arsitektur monolitik tradisional, yang sering kali menyebabkan kesulitan dalam penerapan dan pemeliharaan karena semua komponen aplikasi terikat erat dan perlu diperbesar skala secara bersamaan.

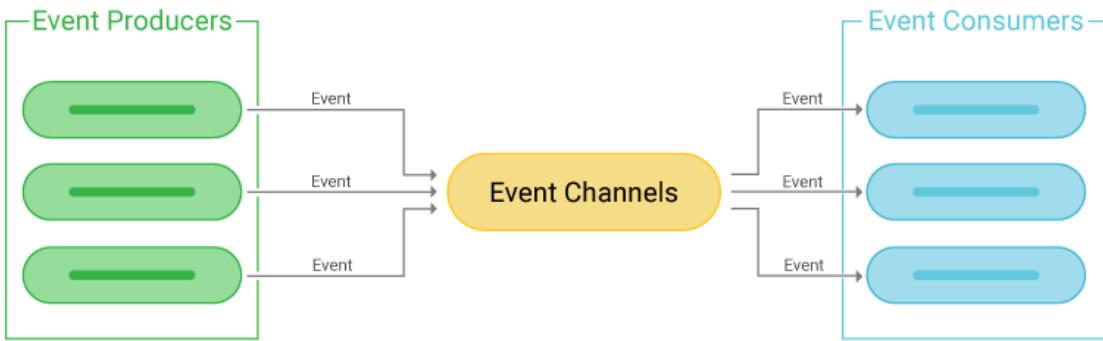


Gambar 2.7: Ilustrasi Arsitektur Microservices

Sumber: <https://microservices.io/patterns/microservices.html>

2.4.8 Arsitektur Event-Driven

Arsitektur *Event-Driven* (Event-driven architecture, EDA) adalah gaya arsitektur unik yang menggunakan *events* untuk memicu pelaksanaan proses bisnis tertentu (Wieland et al., 2009). Seperti pada Gambar 2.8, dalam arsitektur ini, *events* adalah transisi keadaan yang penting, yang setelah terdeteksi, memicu berbagai alur kerja atau proses bisnis. Karena komponen dalam arsitektur ini berkomunikasi secara asinkronus dan dapat cepat beradaptasi dengan perubahan atau informasi baru, ketangkasan dan skalabilitas sistem ditingkatkan, menghasilkan sistem yang sangat responsif. EDA juga memungkinkan antar komponen saling independen yang meningkatkan fleksibilitas sistem dan memudahkan pemeliharaan serta pembaruan.

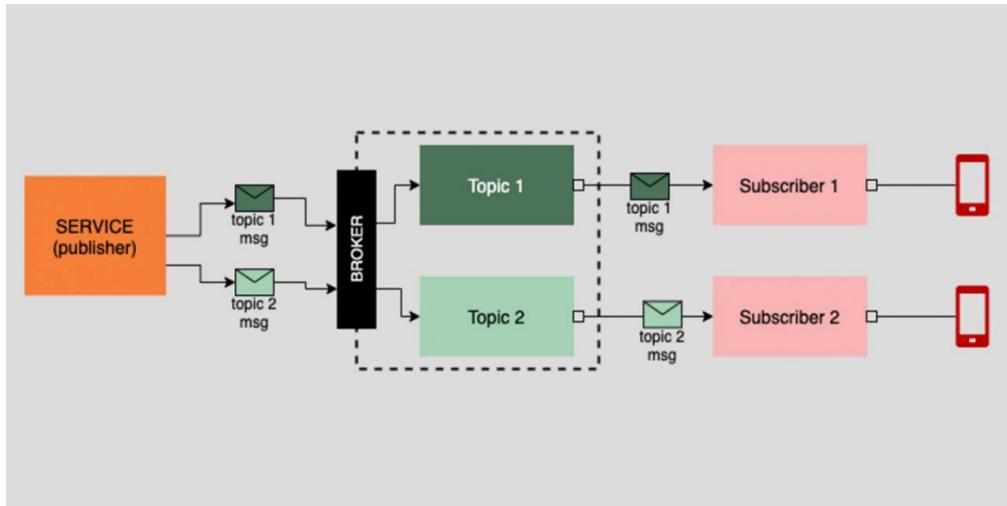


Gambar 2.8: Ilustrasi Arsitektur Event-Driven

Sumber: <https://www.scylladb.com/glossary/event-driven-architecture>

2.4.9 Message Broker

Message broker adalah komponen dasar di lapisan transportasi dalam layanan yang mengatasi masalah seperti ketidakandalan jaringan, keterkaitan kuat antara produsen dan konsumen, serta perbedaan jenis aplikasi (Magoni, 2015). *Message broker* bertindak sebagai penghubung dalam jaringan, mengatur pengiriman dan penerimaan pesan, sehingga memudahkan komunikasi antara berbagai layanan dan klien seperti visualisasi pada Gambar 2.9. Sistem ini sering menggunakan model *publish* dan *subscribe*. Penerbit (publisher) mengirim pesan tanpa perlu tahu siapa penerima, sedangkan pelanggan (subscriber) menerima pesan sesuai dengan yang mereka daftarkan. Dengan cara ini, memasukkan layanan yang berbeda-beda ke dalam jaringan besar menjadi lebih mudah dan komunikasi antara mereka bisa dilakukan tanpa harus saling terikat langsung.

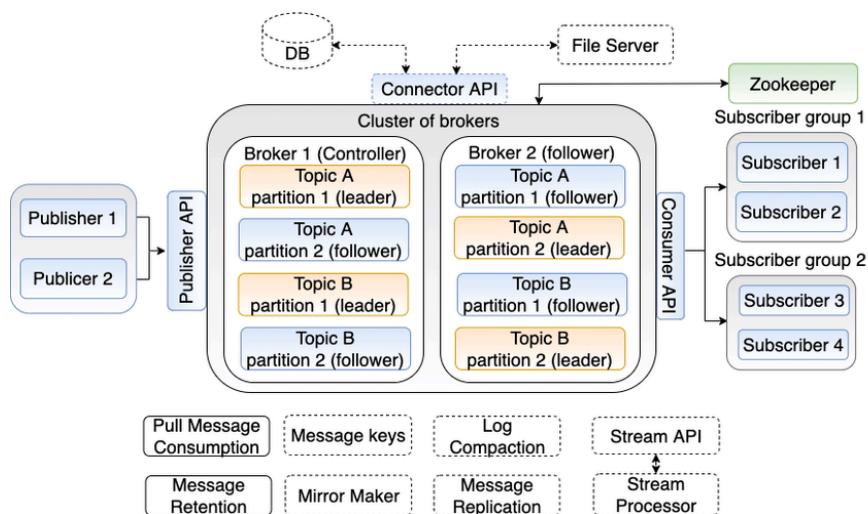


Gambar 2.9: Ilustrasi Cara Kerja Message Broker

Sumber: <https://www.vmware.com/topics/glossary/content/message-brokers.html>

2.4.9.1 Apache Kafka

Apache Kafka adalah sebuah sistem terdistribusi yang sangat *powerful* yang dirancang untuk tingkat *throughput* dan skalabilitas yang tinggi (Kim et al., 2019). Kafka mampu memproses 2 juta data dalam 1 detik yang menjadikannya sebagai *message broker* paling banyak digunakan untuk sistem yang membutuhkan pemrosesan data yang sangat cepat (Kreps, 2014).



Gambar 2.10: Arsitektur Apache Kafka

Sumber: <https://www.researchgate.net/>

Penanganannya yang efisien terhadap volume data yang besar membuatnya populer dalam *real-time analytics* dan aplikasi big data. Integritas data dan toleransi kesalahan dijamin oleh operasi *commit log* yang terdistribusi. Ini membuatnya menjadi pilihan yang sempurna untuk aplikasi yang memerlukan streaming data yang konsisten, seperti sistem pemantauan *real-time* atau pemrosesan transaksi keuangan. Karena arsitekturnya yang terdistribusi, Kafka mudah untuk diskalakan secara horizontal, menampung beban data yang lebih besar tanpa mengalami penurunan kinerja yang mencolok. Seperti pada Gambar 2.10, Apache Kafka terdiri dari beberapa komponen utama berikut.

1. Producers

Producers adalah komponen yang berfungsi untuk mengirim data kepada Kafka, lebih tepatnya mengirim data ke Kafka Brokers yang kemudian data tersebut akan disimpan. Producers dapat memilih kepada topik dan partisi yang mana datanya akan dikirim

2. Brokers

Kluster Kafka terdiri dari satu atau lebih server yang kemudian dikenal sebagai broker. Broker-broker ini memiliki tugas untuk menjaga data yang dikirimkan oleh producer. Setiap broker bisa saja memiliki nol atau lebih partisi untuk setiap topik dan setiap partisi adalah urutan catatan yang teratur dan tidak dapat diubah.

3. Consumers

Consumers membaca data dari brokers dengan cara melakukan subscribe pada satu atau lebih topik. Komponen ini melacak data mana saja yang sudah dikonsumsi dengan menggunakan offset. Offset adalah nomor urutan yang diberikan kepada data-data pada suatu partisi.

4. Topik dan Partisi

Topik adalah suatu nama tempat suatu data dipublikasikan. Topik dibuat agar bisa membedakan data-data yang ada di dalamnya. Di dalam topik, terdapat partisi yang berfungsi untuk meningkatkan skalabilitas dan pemrosesan paralel. Partisi juga berguna untuk menyebarkan data di beberapa *broker* sehingga dapat mentoleransi kesalahan (*fault tolerance*).

5. Zookeeper

Komponen ini merupakan komponen yang sangat krusial karena mengatur bagaimana kluster kafka berfungsi. Kafka menggunakan Zookeeper untuk mengelola dan mengkoordinasikan *broker-broker* pada Kafka. Zookeeper bertindak sebagai layanan terpusat untuk memelihara informasi konfigurasi, penamaan, menyediakan sinkronisasi terdistribusi, dan menyediakan layanan kelompok.

2.4.10 Docker

Docker adalah teknologi kontainerisasi yang digunakan dalam alur kerja devops untuk mengirimkan aplikasi dalam bentuk *image*, bersama dengan dependensi dan lingkungan eksekusinya (Rosa et al., 2022). Docker dapat mempercepat fase pengembangan, pengujian, dan penyebaran perangkat lunak pada suatu sistem yang menerapkan arsitektur *microservices*.

Teknologi kontainerisasinya mengenkapsulasi sebuah aplikasi, dependensi, dan lingkungan eksekusinya ke dalam satu *image* kontainer. Karena *image* Docker yang sama dapat digunakan dalam lingkungan pengembangan, pengujian, dan produksi, teknik ini menjaga konsistensi di seluruh siklus pengembangan. Docker mempermudah proses migrasi aplikasi dari satu lingkungan komputer ke lingkungan lain dengan mengemas segala yang dibutuhkan untuk menjalankan sebuah aplikasi ke dalam satu kontainer.

2.4.10.1 Docker Image

Docker Image adalah sebuah *template* yang bersifat *read-only* dengan instruksi untuk membuat Docker Container (Docker, n.d.). Docker Image pada dasarnya adalah potret suatu aplikasi dan lingkungannya. Gambar ini bersifat tidak dapat diubah (*read-only*), yang berarti setelah dibuat, tidak mengalami perubahan.

Ketika Docker Container dijalankan, hal tersebut adalah instansi (*instance*) dari Docker Image yang menyediakan lingkungan runtime yang portabel dan konsisten di berbagai lingkungan pengembangan, seperti *staging* dan *production*. Docker Image

biasanya dibuat melalui proses yang didefinisikan dalam Dockerfile, sebuah file teks biasa dengan serangkaian perintah untuk merakit *image* tersebut.

2.4.10.2 Docker Container

Sebuah *instance* yang dapat dieksekusi dari Docker Image disebut dengan Docker Container (Docker, n.d.). API dan CLI Docker memungkinkan untuk membuat, menghentikan, memindahkan, dan menghapus kontainer. Sebuah kontainer dapat memiliki penyimpanan yang terpasang, terhubung ke beberapa jaringan, atau bahkan dapat membuat image baru berdasarkan keadaan saat ini.

Satu kontainer dengan kontainer yang lainnya bersifat saling terisolasi. Akan tetapi, setiap kontainer dapat saling berkomunikasi dengan menggunakan jaringan yang sama sehingga setiap kontainer dapat saling bertukar pesan melalui jaringan tersebut. Sifat ini sangat menguntungkan karena memungkinkan setiap layanan yang berjalan dalam kontainer untuk tetap terjaga keamanannya dan stabil, sementara masih memfasilitasi interaksi yang diperlukan antar layanan. Isolasi ini memastikan bahwa konfigurasi atau keadaan dari satu kontainer tidak akan secara tidak sengaja mempengaruhi kontainer lainnya, yang menambah lapisan keamanan dan keandalan dalam pengembangan aplikasi. Selain itu, komunikasi antar kontainer yang efisien sangat esensial dalam arsitektur berbasis mikroservis, di mana aplikasi dibagi menjadi layanan-layanan kecil yang saling berinteraksi

2.5 Data Seismik dan Sumber Data

Seismologi sangat bergantung pada kerjasama, baik dalam skala nasional maupun internasional. Hanya dengan mengumpulkan data masif dalam format standar dari banyak stasiun dan jaringan di seluruh dunia selama periode waktu yang panjang dapat memberikan hasil yang *reliable* untuk menentukan *event localization*, tingkat seismisitas, penilaian bahaya, penyelidikan struktur dan rheologi interior Bumi, serta hal penting lainnya dalam penelitian seismologi (Dost, Bernard et al. 2012).

Saat ini, format paling populer yang digunakan untuk pertukaran data seismik adalah SEED (Standard for the Exchange of Earthquake Data). SEED dikembangkan oleh IRIS (Incorporated Research Institutions for Seismology) bersama dengan FDSN (Federation of Digital Seismographic Networks) yang didesain sebagai format standar

untuk pertukaran data seismik digital. Dalam praktisnya saat ini, format SEED terbagi menjadi dua bentuk, yaitu miniSEED yang digunakan untuk menyimpan data *waveform*, dan dataless SEED yang digunakan untuk menyimpan metadata stasiun. (IRIS, 2012)

2.5.1 FDSN Web Service

Dikutip dari jurnal FDSN (2019), FDSN Web Services merupakan spesifikasi RESTful Web Service yang ditetapkan oleh International Federation of Digital Seismograph Networks (FDSN) untuk pertukaran data *time series*, metadata stasiun seismograf, parameter *event* gempa, dan data lainnya yang berada dalam konteks FDSN. Web service menyediakan layanan data yang dapat diakses mesin yang biasanya diakses melalui protokol HTTP. Umumnya, layanan ini diekspos secara publik sehingga pengguna dapat mengaksesnya secara langsung melalui aplikasi mereka sendiri (Dost, Bernard et al. 2012).

Tujuan dari spesifikasi ini adalah memberi *client software* kemudahan untuk menggunakan data dari pusat data FDSN. Data center yang mengadopsi standar ini seperti IRIS, Geonet, dan Geofon dapat bertukar dan mempublish data dengan mudah dan konsisten.

2.5.2 SeedLink

Seedlink merupakan *application-level protocol* yang digunakan untuk mentransfer *time-series* data untuk aplikasi seismologi secara *near real-time*. Seedlink pertama dikembangkan sebagai *transport layer* untuk SeisComP. Seedlink menggunakan TCP/IP connection dengan default port 18000 atau 18500 dengan menggunakan TLS. Secara umum, protokol ini terdiri atas dua fase, yaitu *handshake* dan transfer data. Selama fase *handshake*, klien dapat *subscribe* ke stasiun tertentu dan menerima data dengan memberikan *command*. Ketika fase *handshake* selesai, data dikirim oleh server SeedLink berupa paket miniSEED berukuran 512 byte yang dikeluarkan dengan urutan First In First Out (FIFO) (GFZ et GEMPA GmbH, 2008).

Tidak ada protokol standard yang diakui secara internasional untuk pertukaran data seismik secara *real-time* saat ini. Akan tetapi, hanya terdapat sejumlah protokol dan

yang paling sering dipakai adalah SeedLink, SCREAM, NAQS, CD1.x, Antelope, EarthWorm, NRTS, dan LISS.

Protocol SeedLink dapat berjalan menggunakan dua mode transmisi data, yaitu uni-station dan multi-station. Mode uni-station beroperasi dengan mentransmisikan satu data *stream* melalui satu channel TCP. Sementara itu, mode multi-station dapat menerima data dari berbagai stasiun dengan memanfaatkan multiplex dalam satu channel TCP (SAGE, n.d.).

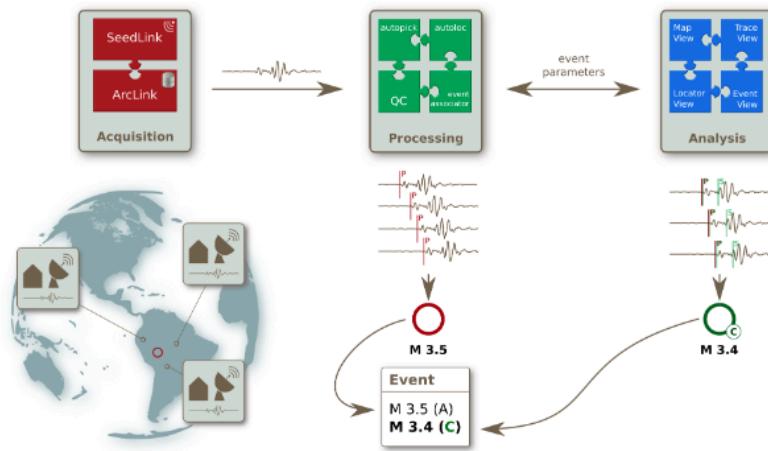
2.5.3 ObsPy

ObsPy merupakan *library* dalam bahasa python yang ditujukan untuk mempermudah pengembangan *software* dan *workflow* dalam bidang seismologi yang bisa menghubungkan disiplin ilmu ini dengan komunitas ilmiah python yang besar (Beyreuther, et al. 2010).

Dikutip dari (Krischer, Lion, et al. 2014), Library ObsPy memiliki kemampuan untuk melakukan operasi I/O untuk semua jenis format yang umumnya muncul dalam komunitas seismologi. Kemudian, terdapat berbagai fungsi signal processing yang penting dalam analisa dan pengolahan data. Selain itu, library ObsPy memberi akses untuk melakukan integrasi data dari berbagai data source seperti SeedLink, ArcLink, FDSN Web Service, SDS, dan lain lain menggunakan interface client yang sederhana.

2.6 SeisComP

SeiscomP merupakan paket software yang digunakan dalam dunia seismologi untuk *data acquisition, processing, distribution* dan *interactive analysis*. Pengembangan SeisComP dimulai pada 2006 hingga 2008 dalam proyek GITEWS (German Indonesian Tsunami Early Warning System) oleh tim GFZ (German Research Centre for Geosciences). Seperti ilustrasi pada Gambar 2.11, sistem SeiscomP berawal dari pengembangan sistem akuisisi data sampai sistem pemantauan gempa bumi secara *real time*. Sejak tahun 2008, pengembangan SeisComP dilakukan oleh tim GFZ beserta gempa GmbH, perusahaan yang dibentuk oleh para pengembang awal software yang sekarang menawarkan layanan, produk, dan solusi terkait SeisComP.

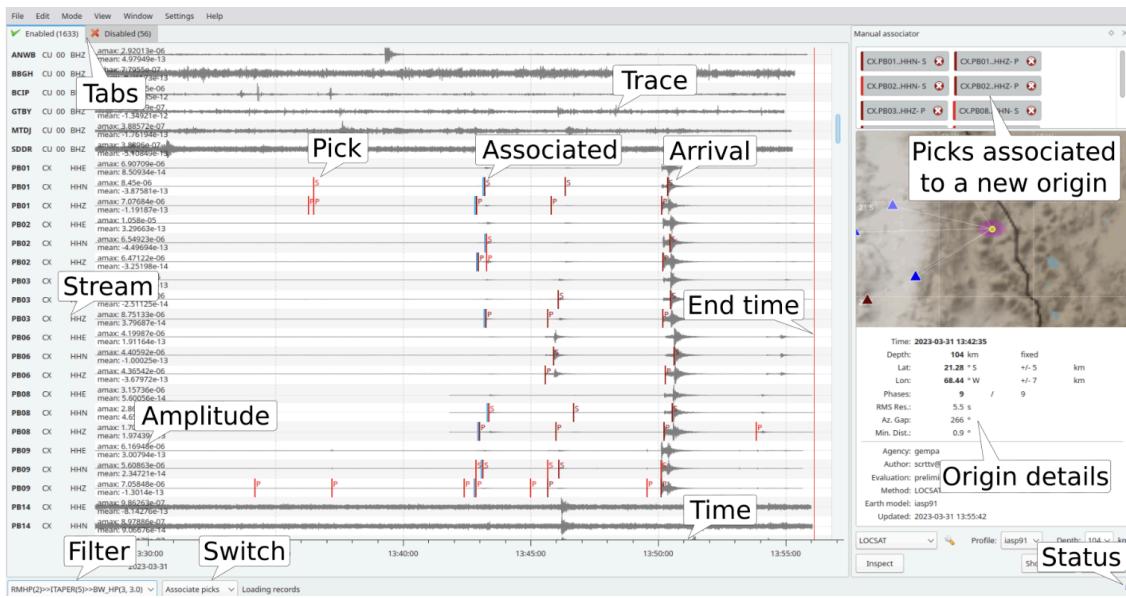


Gambar 2.11: Akuisisi Data Real-Time, Pemrosesan Data, dan Analisa Data dalam SeisComP

Sumber: <https://www.seiscomp.de/doc/base/overview.html>

Sistem SeisComP tidak berdiri secara monolith, melainkan terdiri atas beberapa modul yang berjalan secara terpisah untuk memproses data dan menganalisa aktivitas seismik. Sebuah modul *trunk* merupakan bagian yang menggunakan dan berada di dalam framework SeisComP. Semua *trunk* memiliki konfigurasi yang umum dan menggunakan database yang menyimpan Inventory, EventParameters, Configuration, Routing dan QC schemas bersama. Contoh modul ini adalah scautoloc, scautopick, dan GUI seperti scrttv dan scolv. Sementara itu, modul *standalone* merupakan modul yang tidak terhubung dengan framework. Contoh modul ini adalah seedlink, slarchieve, dan slmon (GFZ & GEMPA GmbH, 2008)

Setiap modul memiliki tugas tertentu seperti akuisisi data, pengarsipan data, pemrosesan data, pengidentifikasi *phase*, pengidentifikasi *event* gempa dan lokasi, dan Graphical user Interface (GUI). Komunikasi antar modul dicapai dengan menggunakan *messaging system* yang memungkinkan komputasi terdistribusi dan *remote review*. Untuk pertukaran metadata seismologis, SeisComP memiliki alat ekspor/import ke/dari QuakeML dan FDSN StationXML (GFZ & GEMPA GmbH, 2008).



Gambar 2.12: Modul SCRTTV

Sumber: (GFZ & GEMPA, GmbH, 2008)

SCRTTV (SeisComP Real-Time Trace View) merupakan modul GUI yang memvisualisasikan *waveform* data secara *real-time* atau melakukan *playback* melalui modul arsip seperti yang dapat dilihat pada Gambar 2.12. Modul ini dapat juga menunjukkan *phase pick* dari *p wave* dan *origin* gempa yang meliputi lokasi (*hypocenter*), waktu, dan kekuatan gempa berdasarkan fase seismik dan amplitudo.

2.7 Web-Based Application

Web-based applications merupakan *software* yang disimpan di suatu server dan menggunakan teknologi web (seperti Flash, Javascript, HTML, CSS) dan menggunakan web browser untuk menjalankan fungsinya (Julie Craig, 2017). Secara umum, web-based applications berbeda dari aplikasi client-server tradisional. Layer presentation masih berjalan di *workstation* pengguna, tetapi menggunakan web browser (Chrome, Firefox, Opera, etc.) dan bukan program GUI seperti Qt dan GTK+, yang memiliki masalah portabilitas karena hanya berjalan di *workstation* dengan OS tertentu saja.

Keuntungan web-based applications adalah dapat diakses dari mana saja melalui *web browser*, sehingga menghilangkan kebutuhan untuk menginstal *software* pada klien, menghilangkan kebutuhan OS spesifik, dan mempermudah update *software*

secara rutin (Julie Craig, 2017). Tidak hanya itu, web-based applications membantu menekan *cost* pengembangan karena pengembang hanya butuh membuat *software* di satu platform saja .

2.7.1 User Interface and Experience

User Interface (UI) adalah interaksi antara pengguna dan sistem yang menggunakan perintah, grafik, tampilan, konten, dan lain-lain. User interface (UI) adalah salah satu bagian dari pengalaman pengguna (*user experience*) (Jones, 2022). Sementara itu, User experience (UX) adalah proses keseluruhan dan pengalaman antara pengguna dengan aplikasi. Proses ini melibatkan branding, desain, penggunaan produk, dan fungsi. UX adalah bagian dari desain yang mempertimbangkan serangkaian pengalaman yang terintegrasi mulai dari tahap refleksi hingga saat produk atau layanan mencapai pengguna. (Stull, Edward 2018). Untuk mencapai suatu UX yang bagus, diperlukan sebuah pengertian dalam terhadap pengguna dan kepentingan pengguna, yang pada akhirnya berbicara kepada pengalaman yang akan dimiliki pengguna saat menggunakan produk (Pillay & Wing, 2019).

Dalam artian sederhana User Interface (UI) adalah bagaimana pengalaman estetik dari produk atau servis, sementara User Experience (UX) adalah bagaimana mendesain pengalaman secara utuh dari suatu produk atau servis dari tahap konseptualisasi hingga akhir (Jones, 2022).

BAB 3

METODOLOGI

Bab ini menjelaskan metode-metode yang digunakan untuk menyelesaikan masalah yang diidentifikasi dalam penelitian ini sesuai dengan yang dijelaskan pada bab 1. Bab ini membahas metodologi penelitian yang terdiri atas pembagian ruang lingkup penelitian, tahapan penelitian, deskripsi data, dan metode evaluasi penelitian.

3.1 Pembagian Ruang Lingkup Penelitian

Sesuai dengan latar belakang dan pertanyaan penelitian yang sudah dijelaskan pada bab 1, penelitian ini dapat dibagi menjadi tiga ruang lingkup masalah seperti pada Tabel 3.1. Ruang lingkup masalah yang dibahas dalam penelitian ini adalah pengembangan model *machine learning*, pengembangan *backend* dengan arsitektur *event-driven*, dan pengembangan aplikasi web. Setiap ruang lingkup memiliki penjelasan metode penelitian, implementasi, dan evaluasi yang berbeda-beda dan dibahas dalam babnya masing-masing.

Tabel 3.1: Pembagian Identifikasi Masalah

| Identifikasi masalah | Sub Masalah | Ruang lingkup masalah | Bab Pembahasan |
|------------------------|--|-------------------------------|----------------|
| Model Machine Learning | 1. Pembuatan model deteksi kedatangan gelombang P 2. Pembuatan model deteksi kedatangan geombang S 3. Pembuatan model aproksimasi magnitudo dan lokasi gempa | Implementasi Machine Learning | Bab 4 |

Tabel 3.1: Pembagian Identifikasi Masalah (sambungan)

| Identifikasi masalah | Sub Masalah | Ruang lingkup masalah | Bab Pembahasan |
|-----------------------|---|---|----------------|
| Sistem <i>Backend</i> | <ol style="list-style-type: none"> 1. Membangun <i>loosely coupled microservices</i> 2. Membangun arsitektur <i>event-driven</i> 3. Membangun sistem dengan <i>throughput</i> tinggi | Implementasi <i>Backend Engineering</i> | Bab 4 |
| Aplikasi Berbasis Web | <ol style="list-style-type: none"> 1. Membuat User interface 2. Membangun visualisasi data | Implementasi Frontend Engineering | Bab 4 |

3.2 Metodologi Umum

Pendekatan yang dilakukan berupa pendekatan kuantitatif dan kualitatif dalam proses evaluasinya. Namun, masing-masing ruang lingkup masalah memiliki skenario evaluasi yang berbeda. Hasil evaluasi lalu diukur dengan matriks evaluasi yang telah ditentukan dan dapat dilakukan analisis serta penarikan kesimpulan penelitian.

3.2.1 Tahapan Penelitian

Proses penelitian ini secara umum dibagi menjadi lima tahapan penelitian yaitu perumusan masalah, studi literatur, perancangan dan implementasi sistem, evaluasi sistem, dan penarikan kesimpulan. Seperti yang dapat dilihat pada Tabel 3.2, masing-masing tahapan memiliki masukan, metode, dan keluaran yang berbeda-beda. Untuk tahapan pertama, setiap ruang lingkup memiliki masukan dan metode tersendiri yang dijelaskan lebih lanjut pada masing-masing bab. Hasil dari tahapan pertama adalah pertanyaan penelitian yang menjadi masukan untuk tahapan berikutnya

Tabel 3.2: Tahapan Penelitian Secara Umum

| No | Tahap | Input | Proses | Output |
|----|-----------------------|---|--|----------------------------------|
| 1 | Pengumpulan Kebutuhan | Latar Belakang Penelitian, interview user bmkg | Pencarian <i>gap</i> antara sistem yang ada dengan kebutuhan | Rumusan Masalah |
| 2 | Studi Literatur | Penelitian terkait, penelitian terdahulu | Pencarian konsep, ide, alat/tools yang mendukung penelitian | Landasan teori, tinjauan pustaka |
| 3 | Perancangan | Landasan teori, hasil literatur, ide penelitian | Perencanaan bagaimana mengintegrasikan konsep, ide, dan alat untuk membuat satu sistem yang utuh | Sprint Planning |
| 4 | Implementasi/S print | Sprint Planning | Daily standup, sprint review | Kode Aplikasi |

Tabel 3.2: Tahapan Penelitian Secara Umum (sambungan)

| No | Tahap | Input | Proses | Output |
|----|----------------------|---------------------------------------|---|--------------------------|
| 5 | Evaluasi | Kode Aplikasi | Pembuatan unit test, end-to-end test, dan load test | Sistem EEWs yang bekerja |
| 6 | Penarikan Kesimpulan | Hasil evaluasi, pertanyaan penelitian | Analisis evaluasi | Kesimpulan dan saran |

Berikut adalah penjelasan singkat mengenai tahap-tahap penelitian tersebut:

1. Pengumpulan Kebutuhan

Pada tahap ini, penulis mengidentifikasi dan menganalisis kebutuhan fungsional serta non-fungsional sistem EEWs. Kebutuhan fungsional mencakup fitur-fitur inti yang harus disediakan sistem, seperti pengumpulan data seismik, pemrosesan dan analisis data, serta mekanisme peringatan. Sementara itu, kebutuhan non-fungsional berkaitan dengan aspek kinerja sistem, termasuk skalabilitas, kehandalan, dan waktu respons.

Penulis pertama melakukan kunjungan ke kantor BMKG pusat yang berada di Jakarta. Dalam kunjungan ini, penulis hadir bersama Bapak Ari Wibisono, tim dari Universitas Diponegoro, dan Institut Teknologi Bandung. Pada kunjungan ini, pihak BMKG mendemonstrasikan sistem Seiscomp3 yang digunakan untuk mendeteksi gempa. Sistem seiscomp terbagi menjadi tiga subsistem, yaitu *acquisition*, *processing*, dan *dissemination*. Sistem *acquisition* merupakan modul Seiscomp3 bernama seedlink server. Seedlink server melakukan *pulling* data dari ratusan sensor seismograf di Indonesia. Pengaturan sumber data dan koneksi yang bisa menggunakan VSAT atau GSM dapat diatur di file konfigurasi Seiscomp3. Kemudian, sistem berikutnya merupakan

processing atau *scproc*. Scproc memproses semua data dari *acquisition* dan melakukan *autopick* untuk mendapat data *p wave*. Begitu *p wave* ditemukan, maka modul *seiscomp3* lain seperti *Scesv* dapat menerima *event* gempa bumi. Lalu, operator dapat memeriksa data dan menentukan apakah gempa tersebut adalah *false positive* atau bukan. Apabila operator memutuskan bahwa prediksi itu valid, maka *Seiscomp3* akan memproses *event* untuk mendapat parameter seperti magnitudo, lokasi, dan kedalaman gempa. Lalu, modul diseminasi akan diaktifkan untuk memberi informasi gempa melalui SMS ke masyarakat.

Sistem di BMKG mengharuskan dapat memproses 530 titik stasiun seismograf di Indonesia dan luar negeri. Saat ini, hanya terdapat 20 stasiun yang bersifat publik, dan sisanya hanya dapat diakses dari jaringan internal BMKG. Untuk keperluan pengujian, akan dibuat mesin proxy dari jaringan BMKG untuk mengakses *data stream* stasiun seismograf. Kemudian, saat ini terdapat kekurangan dari sistem yang sudah ada, yaitu algoritma *p wave* saat ini masih terdapat kekurangan akurasi sehingga perlu diperiksa manual oleh operator. Kemudian, hasil kedalaman gempa dan magnitudo memiliki presisi yang rendah. Saat ini, tingkat error magnitudo yang diterima adalah 0.3 skala richter, tetapi pada kenyataanya bisa sampai 0.5 skala richter..

2. Studi Literatur

Pada tahapan penelitian ini, penulis melakukan studi literatur secara mendalam berdasarkan pertanyaan penelitian yang ada. Sumber-sumber yang didapatkan baik dari penelitian terdahulu dan *software* terkait (*Seiscomp3*) diolah untuk mendapatkan landasan teori yang mendukung penelitian.

3. Perancangan

Pada tahap perancangan, penulis mengadopsi arsitektur *event-driven* yang telah menjadi pilihan utama dalam pengembangan sistem ini, dengan memanfaatkan konsep microservices untuk menciptakan sistem yang *loosely coupled*. Hal ini memungkinkan sistem untuk menjadi modular, di mana setiap layanan dapat dikembangkan, diuji, dan dikelola secara terpisah. Dengan memisahkan layanan menjadi unit-unit kecil yang berkomunikasi melalui event, penulis memastikan

bahwa sistem memiliki keandalan dan fleksibilitas tinggi, serta memudahkan dalam menangani perubahan atau penambahan fitur baru.

Untuk menunjang arsitektur ini, Kafka dipilih sebagai *message broker* karena kemampuannya yang terbukti dalam menangani *streaming data* dengan kecepatan tinggi dan skalabilitas yang sangat baik. Kafka berperan sebagai tulang punggung dalam menyalurkan *event* antara layanan, memastikan bahwa pesan diproses dan diteruskan dengan efisien. Ini sangat penting dalam sistem yang mengandalkan kecepatan pemrosesan data *real-time*, seperti EEWS yang setiap detik dalam deteksi dan peringatan dini gempa bumi sangat berharga.

Untuk penyimpanan hasil prediksi, MongoDB menjadi pilihan karena sifatnya yang mudah di-*scaling* serta memiliki performa yang cepat dalam membaca dan menulis data besar. MongoDB, sebagai basis data NoSQL, menawarkan fleksibilitas dalam menangani berbagai format data yang diterima dari layanan prediksi, memungkinkan pengumpulan dan analisis data secara efektif. Kinerjanya yang optimal untuk operasi dengan skala besar menjadikannya sangat cocok untuk sistem yang memproses volume data seismik yang signifikan.

Selain itu, Redis digunakan untuk *caching*, yang sangat membantu dalam meningkatkan kinerja sistem. Dengan menyimpan data sementara yang sering diakses dalam *cache Redis*, sistem dapat mengurangi waktu akses data yang berulang. Ini mempercepat respons sistem terhadap event baru, yang merupakan aspek kritikal dalam pemberian peringatan dini pada EEWS. Kombinasi penggunaan Kafka, MongoDB, dan Redis ini dirancang untuk memaksimalkan kecepatan dan efisiensi pemrosesan data seismik, sehingga dapat memberikan peringatan secepat mungkin dengan data yang akurat.

4. Implementasi/*Sprint*

Implementasi sistem EEWS dilakukan menggunakan metodologi Agile, khususnya pendekatan Scrum. Pemilihan metodologi Agile berbasis Scrum ini didasarkan pada beberapa pertimbangan kunci. Pertama, fleksibilitas yang ditawarkan oleh Agile memungkinkan penulis dan tim untuk menyesuaikan dengan perubahan kebutuhan dan prioritas secara dinamis, yang sering terjadi

dalam proyek pengembangan teknologi. Kedua, Scrum mempromosikan kolaborasi yang erat dan komunikasi terbuka antara anggota tim, yang sangat penting dalam proyek yang melibatkan berbagai komponen teknis yang kompleks. Ketiga, pendekatan iteratif Scrum memungkinkan tim untuk secara berkala mengevaluasi dan memperbaiki sistem, sehingga meningkatkan kualitas dan efektivitas produk akhir.

Sprint dalam metodologi Scrum akan dijalankan selama periode pengembangan. Setiap sprint dimulai dengan perencanaan sprint. Setiap anggota menentukan fitur atau komponen sistem yang akan dikembangkan atau ditingkatkan. Kegiatan ini juga melibatkan penetapan tujuan dan *deliverables* yang jelas untuk sprint tersebut.

Durasi setiap sprint ditetapkan selama dua minggu. Durasi ini dipilih karena memberikan keseimbangan yang baik antara memiliki cukup waktu untuk membuat kemajuan signifikan dalam pengembangan, dan tetap cukup pendek untuk memastikan tanggapan cepat terhadap feedback dan perubahan. Di akhir setiap sprint, tim akan melakukan review sprint untuk mengevaluasi apa yang telah diselesaikan, dan retrospektif sprint untuk mengidentifikasi area yang bisa ditingkatkan pada sprint berikutnya. Pendekatan ini memastikan bahwa proses pengembangan berlangsung secara efisien dan produk yang dihasilkan selaras dengan kebutuhan proyek.

5. Evaluasi

Evaluasi merupakan tahap penting dalam penelitian ini, di mana penulis akan menilai kinerja keseluruhan sistem Earthquake Early Warning System (EEWS) yang telah dikembangkan. Tahapan evaluasi ini meliputi beberapa aspek kunci, yaitu pengujian akurasi model deteksi gelombang P (P wave), pengujian kecepatan model deteksi gelombang P dan prediksi parameter gempa, pengujian sistem secara end-to-end, dan load testing menggunakan data seismograf dari BMKG.

6. Penarikan Kesimpulan

Pada tahap ini, penulis akan menganalisis hasil dari pengujian yang telah dilakukan pada tahap evaluasi untuk menarik kesimpulan yang relevan. Analisis

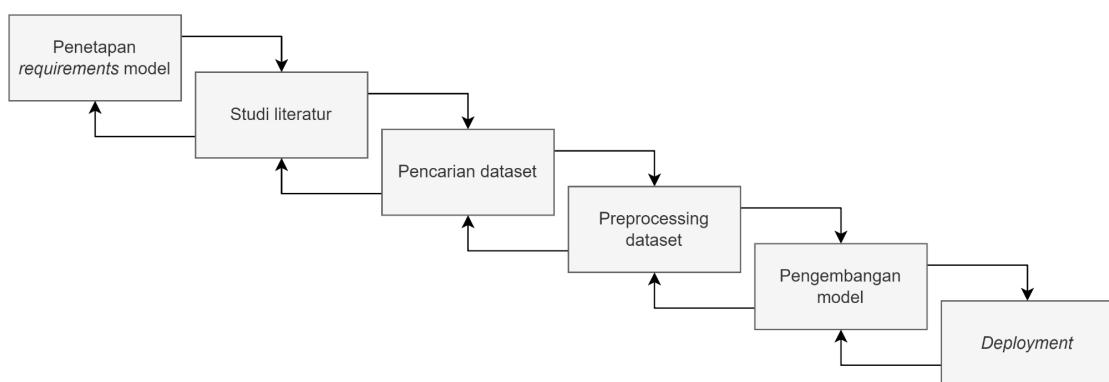
ini akan difokuskan untuk menjawab rumusan masalah yang telah ditetapkan pada awal penelitian. Melalui tahap ini, penulis akan mengeksplorasi sejauh mana sistem EEWs yang dikembangkan memenuhi kriteria dan tujuan yang telah ditetapkan, serta dampaknya terhadap bidang seismologi dan teknologi peringatan dini gempa.

BAB 4

IMPLEMENTASI

4.1 Implementasi Machine Learning

Dalam sub bagian ini, akan dijabarkan langkah-langkah dalam proses implementasi model machine learning yang dikerjakan oleh penulis. Pengembangan model deep learning pada penelitian ini mengikuti pendekatan waterfall yang terdiri dari lima tahapan, yakni penetapan *requirements* model, studi literatur, pencarian dataset, preprocessing dataset, pengembangan model, dan deployment. Gambar 4.1 adalah gambaran proses implementasi model machine learning yang dilakukan oleh penulis.



Gambar 4.1: Tahapan Implementasi model Machine Learning

Sumber: (Woollam, et al., 2022)

Studi literatur telah diuraikan pada Bab 2 dan 3 sebelumnya. Oleh karena itu, dalam subbab ini, penjelasan akan difokuskan pada dataset yang digunakan, proses preprocessing, arsitektur model yang dikembangkan, serta proses deployment sistem dan juga tampilan sistem Machine Learning secara *end-to-end*. Selain itu, penulis juga menuliskan sebuah pendapat pribadi di akhir sub-bab ini.

Pada akhir dari penelitian ini, model deteksi gelombang yang penulis buat nantinya akan dibandingkan dengan sebuah model yang bernama PhaseNet yang pada dasarnya memiliki arsitektur dasar yang sama dengan model yang dibuat oleh penulis. Akan tetapi, model PhaseNet membutuhkan panjang data input yang besar mencapai 3000 *data points* dan data seismogram yang dibutuhkan harus memiliki *sampling rate* sebesar 100Hz. Sementara itu, seismogram yang digunakan di Indonesia memiliki

sampling rate yang hanya sebesar 20 Hz. Oleh karena itu, target penulis adalah untuk membuat sebuah model yang lebih kecil dari PhaseNet yang membutuhkan panjang data input yang jauh lebih kecil dan bisa digunakan untuk deteksi gelombang dengan seismogram yang sesuai dengan spesifikasi Indonesia.

4.1.1 Requirements Model

Pada penelitian ini, penulis mengembangkan dua jenis model, yaitu model deteksi kedatangan gelombang dan model aproksimasi parameter gempa. Model deteksi kedatangan gelombang adalah model yang memiliki kemampuan untuk mendeteksi kedatangan gelombang p/s dan melakukan picking (memberikan tanda waktu kejadian) ketika terdeteksi kedatangan gelombang. Model ini diharapkan dapat menerima sebuah data gelombang seismik yang terdiri atas 3 channel dan mengeluarkan sebuah vektor yang memiliki panjang yang sama dengan inputnya, yang mana vektor tersebut menandakan *likelihood* bahwa gelombang p/s terjadi di sini. Sementara itu, model aproksimasi parameter gempa adalah model yang dapat memperkirakan nilai-nilai dari parameter sebuah peristiwa gempa, yang mencakup magnitudo gempa, lokasi episentrum, dan kedalaman gempa. Model ini diharapkan dapat menerima sebuah data input kejadian gempa dan mengeluarkan parameter-parameter dari gempa tersebut.

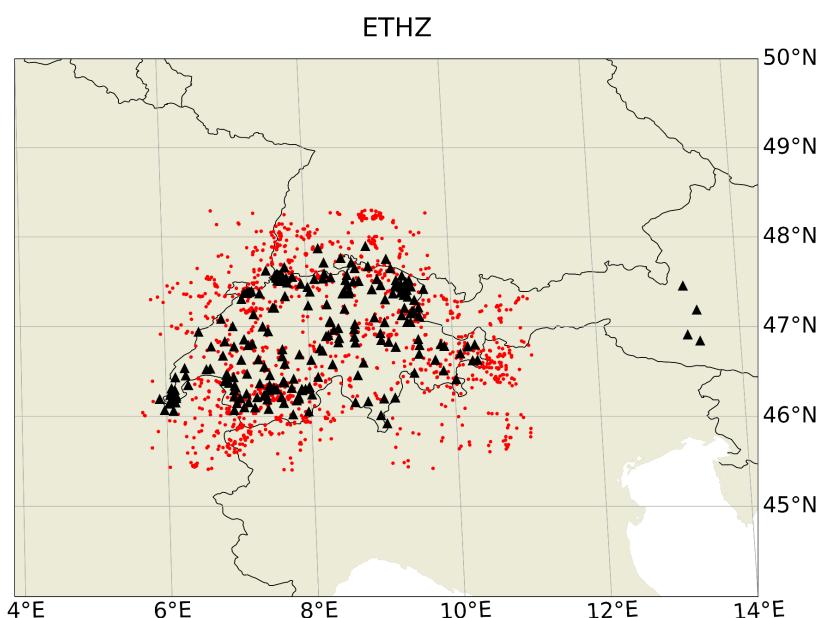
Untuk kedepannya, penulis akan menyebut model deteksi kedatangan gelombang p/s sebagai *model gelombang*, dan model untuk aproksimasi parameter gempa sebagai *model parameter*.

Tabel 4.1: Spesifikasi Input dan Output Model

| Jenis Model | Input | Output |
|-----------------|--|---|
| Model Gelombang | Data <i>stream waveform</i> seismograf | Vektor yang menunjukkan probabilitas adanya kedatangan gelombang p/s di setiap waktunya |
| Model Parameter | Data rekaman <i>waveform</i> dari suatu <i>event</i> gempa | Nilai parameter gempa (magnitudo, jarak, atau kedalaman) |

4.1.2 Deskripsi Dataset

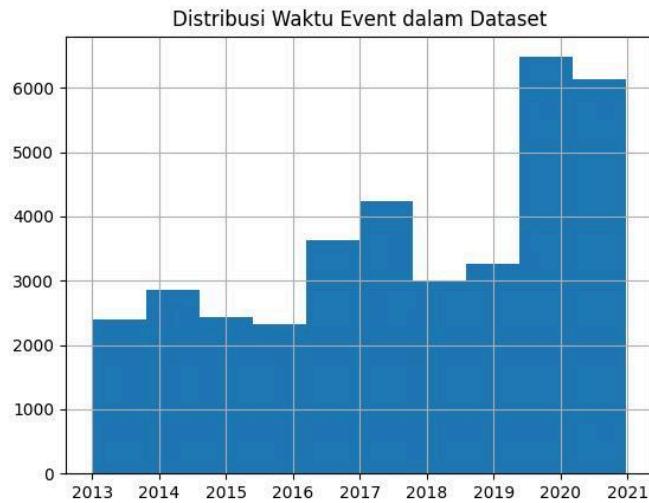
Dataset yang digunakan untuk pengembangan semua model machine learning dalam lingkup penelitian ini adalah dataset data gempa yang telah dikumpulkan oleh ETH Zurich. Dataset ini mencakup ribuan peristiwa gempa yang terekam oleh sejumlah stasiun gempa yang tersebar di seluruh wilayah Swiss dan sekitarnya. Dataset ini terstruktur dalam dua berkas utama, yakni file waveforms.hdf5 yang memuat rekaman kontinu dari seismograf yang beroperasi di wilayah Swiss, dan metadata.csv yang berisikan informasi penting terkait setiap peristiwa gempa.



Gambar 4.2: Distribusi Gempa Dataset ETHZ

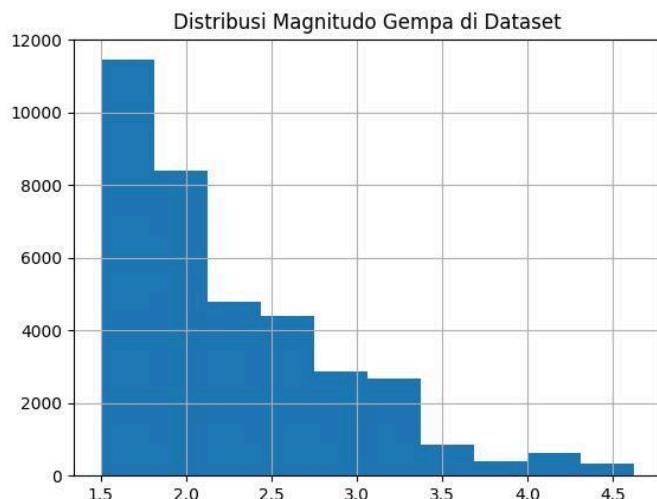
Sumber: (Woollam, et al., 2022)

Dataset ini terdiri dari 36,743 rekaman *event* gempa dan mencakup periode waktu dari tahun 2013 hingga 2021. Rentang magnitudo gempa dalam dataset bervariasi antara 1.5 hingga 4.5, dengan distribusi magnitude yang memiliki *skewness* ke kanan yang berarti, sebagian besar gempa dalam dataset cenderung memiliki magnitudo yang lebih rendah. Rata-rata magnitudo dalam dataset ini adalah sekitar 2.25 SR.



Gambar 4.3: Distribusi Kejadian Gempa dalam Dataset ETHZ

Sumber: (Olahan Penulis, 2023)



Gambar 4.4: Distribusi Magnitudo Gempa dalam Dataset ETHZ

Sumber: (Olahan Penulis, 2023)

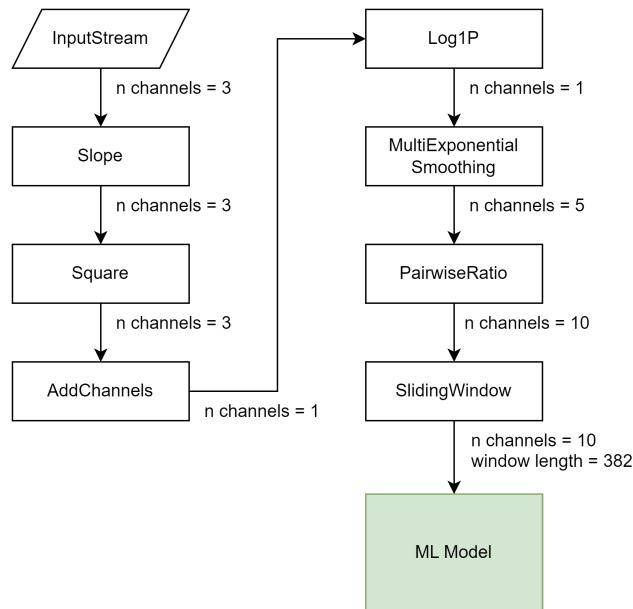
Selain itu, *waveform* rekaman gempa yang ada di dalam dataset ini memiliki *sampling rate* sebesar 100 Hz. Mengingat bahwadata yang dikeluarkan oleh geofon di Indonesia memiliki *sampling rate* sebesar 20 Hz, maka penulis melakukan *downsampling* terhadap data waveform di dalam dataset ini dengan hanya mengambil datapoint setiap kelipatan lime dari waveformnya. Dengan proses downsampling ini,

maka diharapkan dataset yang digunakan dapat digunakan untuk mendeteksi gempa dengan data yang dihasilkan oleh geofon karena memiliki *sampling rate* yang sama.

4.1.3 Preprocessing Data

Proses *preprocessing* untuk *training* model gelombang dibedakan dengan *preprocessing* untuk model parameter. Model deteksi gelombang diarahkan untuk mendukung inferensi secara *real-time*, oleh karena itu, fokus utama penulis adalah mengoptimalkan ukuran model agar seminimal mungkin dengan cara membuat proses pengolahan data sedemikian hingga menonjolkan daerah kedatangan gelombang p/s. Dengan itu, diharapkan bahwa model gelombang dapat dengan mudah mengidentifikasi kedatangan gelombang p/s. Sementara itu, untuk model parameter, penulis tidak melakukan *preprocessing* apapun. Karena model ini tidak diharapkan untuk bekerja secara *real-time*, maka penulis bisa membuat model dengan arsitektur yang besar tanpa proses *preprocessing* dan menyerahkan identifikasi fitur penting ke model parameter saja.

Pertama, penulis akan menjelaskan tentang proses *preprocessing* data untuk model gelombang. Berikut adalah diagram yang menggambarkan tahapan *preprocessing* yang dilalui oleh data gelombang sebelum masuk ke model *machine learning* untuk di-*train* / diinferensikan.



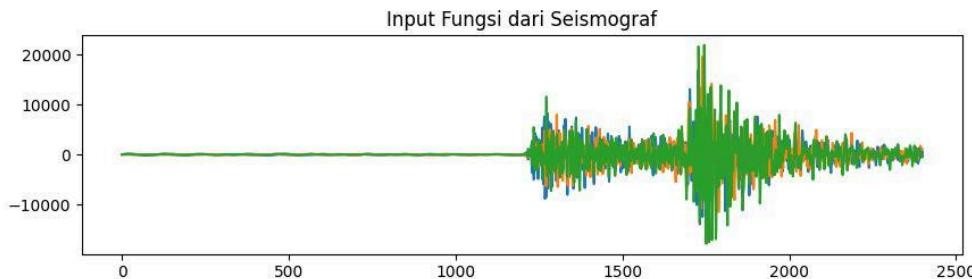
Gambar 4.5: Pipeline Preprocessing Data

Sumber: (Olahan Penulis, 2023)

Proses ini dilakukan melalui penggunaan *pipeline* khusus yang telah disusun oleh penulis, yang kemudian akan menerima aliran data *stream waveform* gempa. Berikut ini adalah penjelasan lebih detail mengenai rangkaian fungsi yang diimplementasikan dalam *preprocessing pipeline* tersebut (beserta penjelasan mengenai input stream pada poin pertama):

1. Input Stream

Gambar 4.6 di bawah adalah tampilan data input yang dimasukkan ke dalam *pipeline*. Input terdiri dari 3 channel yaitu BHE, BHN, BHZ, yang secara berturut-turut menggambarkan data pada seismograf yang menunjuk ke timur, utara, dan azimuth (atas).

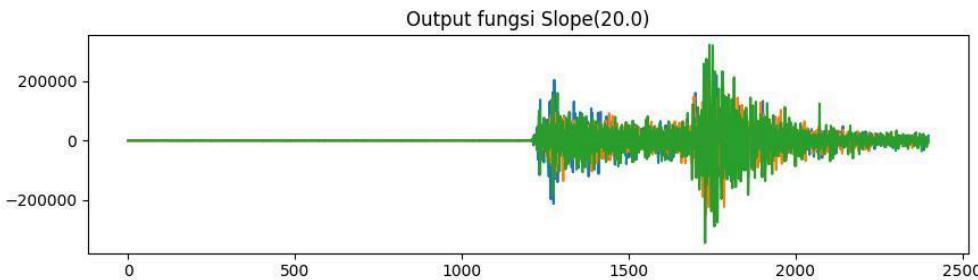


Gambar 4.6: Data yang Diinputkan ke dalam *Pipeline*

Sumber: (Olahan Penulis, 2023)

2. Slope

Fungsi Slope dirancang untuk mengubah seismograf, yang awalnya berisi nilai simpangan (*displacement*) dari jarum jam seismograf, menjadi turunannya terhadap waktu, yaitu kecepatan gerak jarum seismograf. Pendekatan ini dilakukan dengan pertimbangan bahwa osilasi simpangan jarum jam seismograf tidak selalu berada pada nilai nol dan terkadang memiliki *offset* tertentu. Oleh karena itu, konversi ke kecepatan diimplementasikan untuk memastikan bahwa titik tengah osilasi seismograf selalu berada pada titik nol.



Gambar 4.7: Output fungsi Slope dalam *pipeline*

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini, dengan hasil pada gambar 4.7:

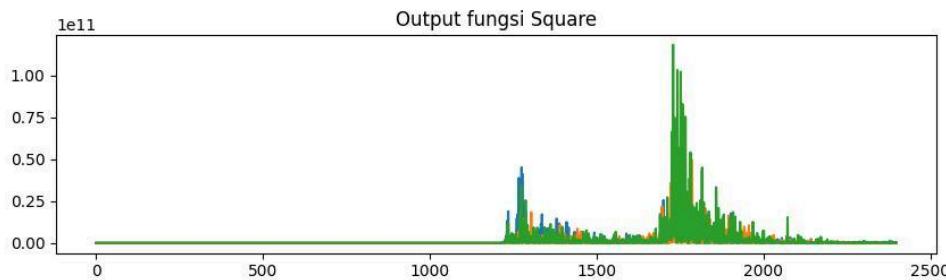
- Deskripsi Output: Turunan input terhadap waktu
- Parameter fungsi: sample_rate = frekuensi seismograf dalam Hz (float)
- Implementasi:

```
def convert_to_velocity(waveform, frequency):
    vel = (waveform[1:] - waveform[:-1]) * frequency
    return vel
```

3. Square

Fungsi Square dirancang untuk mengkuadratkan nilai dari data input. Pemberian fungsi ini pada *pipeline* berdasarkan asumsi penulis bahwa data vibrasi termasuk sebagai data yang susah untuk dipelajari oleh model machine learning karena nilainya yang berosilasi dengan cepat. Penulis ingin melakukan proses *smoothing* pada data input agar bisa lebih mudah untuk dipelajari oleh model *machine learning*.

Dengan transformasi ini, ada dua tujuan yang ingin dicapai oleh penulis. Pertama, untuk mempositifkan semua nilai pada data sebagai tahapan awal proses *smoothing*. Kedua, untuk mengubah representasi kecepatan menjadi representasi energi (mengingat bahwa kuadrat dari kecepatan berbanding lurus dengan energi).



Gambar 4.8: Output fungsi Square dalam pipeline

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini, dengan output ditunjukkan oleh gambar 4.8:

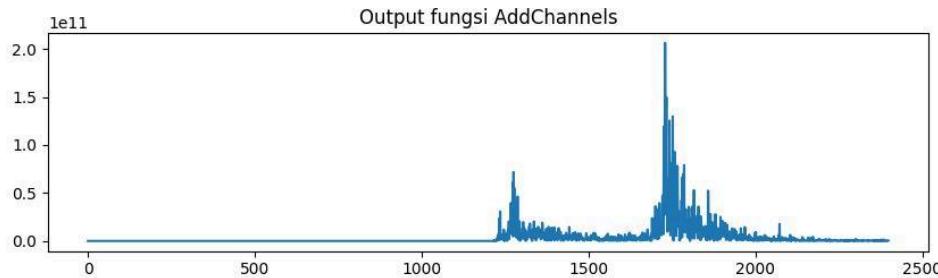
- Deskripsi Output: Kuadrat input
- Parameter fungsi: sample_rate = -
- Implementasi:

```
def square(waveform):
    waveform = waveform**2
    return waveform
```

4. AddChannels

Fungsi *AddChannels* dirancang untuk menggabungkan semua channel yang terdapat pada input menjadi satu channel tunggal dengan menambahkannya. Tujuan dari penerapan transformasi ini adalah karena penulis ingin menyatukan hasil transformasi energi dari fungsi Square sebelumnya menjadi satu channel tunggal yang merepresentasikan total energi.

Penulis berasumsi bahwa pemisahan energi menjadi tiga channel tidak memiliki makna apa-apa dan lebih baik jika data direpresentasikan sebagai total energi saja. Oleh karena itulah ketiga channel dari fungsi sebelumnya digabungkan saja dalam fungsi pipeline ini.



Gambar 4.9: Output fungsi AddChannels dalam pipeline

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini, dengan output ditunjukkan oleh gambar 4.9:

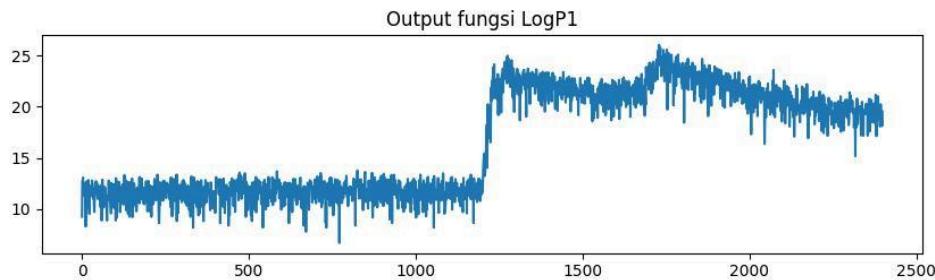
- Deskripsi Output: Penjumlahan dari setiap channel pada input
- Parameter fungsi: -
- Implementasi:

```
def add_channels(waveform, label, state):
    waveform = np.sum(waveform, axis=-1, keepdims=True)
    return waveform
```

5. Log1P

Fungsi Log1P dirancang untuk menurunkan skala data input dengan cara menerapkan operasi logaritma pada data input yang telah ditambahkan dengan 1. Data input pada proses sebelumnya memiliki rentang nilai yang sangat besar yang bahkan mencapai orde 10^{11} .

Tujuan utama dari penerapan fungsi ini adalah sebagai langkah selanjutnya dari upaya untuk melakukan *smoothing* pada data. Dapat diperhatikan bahwa data *waveform* memang secara subjektif menjadi lebih *smooth* setelah dimasukkan ke dalam fungsi ini.



Gambar 4.10: Output fungsi Log1P dalam pipeline

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini, dengan output ditunjukkan oleh gambar 4.10:

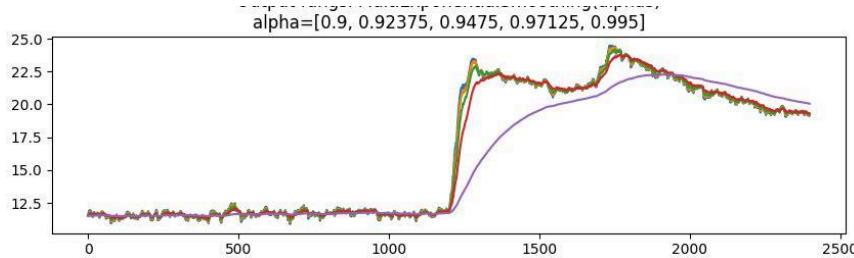
- Deskripsi Output: Nilai log dari input yang ditambah 1
- Parameter fungsi: -
- Implementasi:

```
def log_p1(waveform, label, state):
    waveform = np.log(waveform + 1)
    return waveform
```

6. MultiExponentialSmoothing

Fungsi MultiExponentialSmoothing dirancang untuk menghasilkan nilai dari operasi *exponential moving average* (EMA) pada input. Tujuan utama dari penerapan fungsi ini adalah sebagai langkah akhir dari upaya *smoothing* pada *waveform* input. Operasi EMA memang digunakan untuk melakukan *smoothing* pada sebuah data *timeseries*. Operasi tersebut memiliki satu parameter alpha yang mengatur tingkat kehalusan dari nilai outputnya. Semakin besar nilai alpha, maka semakin halus output yang dihasilkan.

Untuk mencegah terlalu banyak informasi yang hilang, maka penerapan operasi EMA pada proses ini dilakukan dengan variasi nilai alpha yang berbeda-beda, mulai dari nilai yang kecil hingga ke yang besar. Oleh karena itu, luaran dari fungsi ini adalah hasil *smoothing* dari data input dengan nilai alpha yang berbeda-beda.



Gambar 4.11: Output fungsi MultiExponentialSmoothing dalam pipeline

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini, dengan output ditunjukan oleh gambar 4.11:

- Deskripsi Output: Penerapan operasi exponential smoothing pada input dengan nilai alpha yang berbeda-beda
- Parameter fungsi: alphas = list nilai alpha yang digunakan (List[float])
- Implementasi:

```
def _multi_exponential_smoothing(waveform, alphas):
    # Cache 1 - alpha
    alphas_ = 1-alphas

    # EMA initialization
    ema = 0
    if avg_until != -1:
        ema = np.mean(waveform[:avg_until, :], axis=0)

    # Compute ema record for every alpha
    emas = []
    for alpha, alpha_ in zip(alphas, alphas_):
        ema_arr = np.zeros_like(waveform)
        ema_arr[0,:] = ema
        for i in range(1, waveform.shape[0]):
            ema_arr[i,:] = (waveform[i,:] * alpha_ +
                            ema_arr[i-1,:] * alpha)
        emas.append(ema_arr)

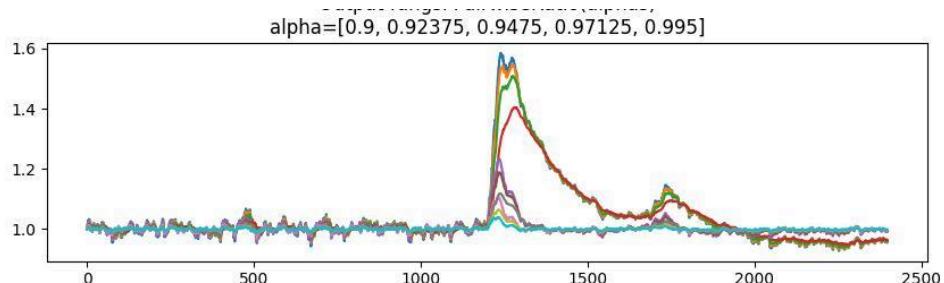
    waveform = np.concatenate(emas, axis=-1)
    return waveform
```

7. PairwiseRatio

Fungsi PairwiseRatio dirancang untuk menghitung rasio dari setiap pasang channel pada input yang telah ditambahkan nilai 1. Fungsi ini bertujuan untuk mempertegas sinyal di daerah kedatangan gelombang p dan s. Dapat

diperhatikan pada output dari fungsi *pipeline* sebelumnya (yaitu fungsi MultiExponentialSmoothing), nilai dari setiap channel akan memiliki nilai yang identik. Akan tetapi, pada saat kedatangan impuls gelombang p/s, channel dengan nilai alpha yang kecil akan mengalami kenaikan nilai energi yang lebih besar dibandingkan channel dengan nilai alpha yang besar. Oleh karena itu, jika kita menghitung rasio antara channel bernilai alpha kecil dengan channel bernilai alpha besar ketika terjadi kedatangan gelombang p/s, maka nilai rasionalya akan lebih besar dari 1, dan juga akan bernilai mendekati 1 ketika dalam keadaan tidak ada kedatangan gelombang. Hal tersebut memang terlihat dengan jelas pada gambar di atas dan secara subjektif sinyal di daerah kedatangan p/s terlihat menjadi lancip setelah penerapan fungsi ini.

Berdasarkan deskripsi sebelumnya, terdapat sebuah syarat khusus terkait pasangan channel yang dapat dirasionalkan, yaitu channel yang berada di bagian pembilang harus memiliki nilai alpha yang lebih kecil dibandingkan dengan channel yang berada di bagian penyebut.



Gambar 4.12: Output fungsi PairwiseRatio dalam *pipeline*

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini, dengan output ditunjukan oleh gambar 4.12:

- Deskripsi Output Fungsi: Rasio dari setiap pasang channel pada input yang ditambah dengan 1
- Parameter fungsi: alphas = list nilai alpha yang digunakan (List[float])
- Implementasi:

```
def _pairwise_ratio(waveform, alpha):
    n = waveform.shape[-1]
```

```

ratios = []
for i, j in generate_index(alpha):
    ratios.append((1+waveform[:,i])/(1+waveform[:,j]))
ratios = np.stack(ratios, axis=-1)
return ratios, label

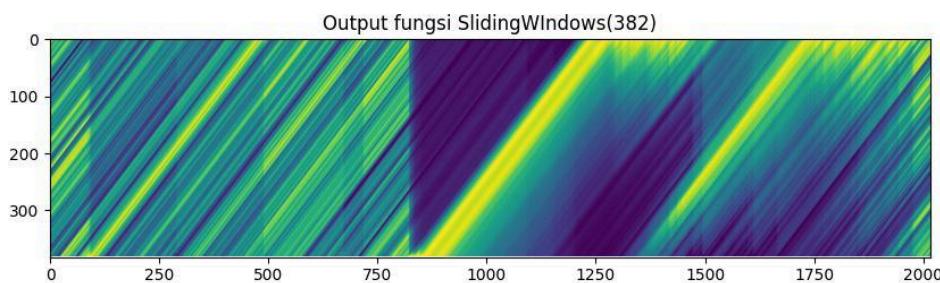
# KET: Generate index menghasilkan pasangan alpha yang memenuhi
#      deskripsi di atas

```

8. SlidingWindow

Fungsi SlidingWindow ini adalah fungsi terakhir pada rangkaian *pipeline*. Fungsi ini dirancang untuk memotong data input dengan menggunakan *sliding windows* yang memiliki ukuran tertentu dan menjalankan proses normalisasi min-max terhadap setiap jendela tersebut. Proses ini dilakukan karena model *machine learning* yang akan menerima data ini hanya menerima data dengan panjang tertentu saja.

Akan tetapi, hal yang paling krusial dari fungsi ini adalah proses *min-max normalization* yang membuat setiap nilai di dalam sebuah *windows* berada dalam rentang interval [0,1]. Penerapan normalisasi akan mempengaruhi sesuatu yang penulis sebut sebagai “bentuk karakteristik” dari setiap *windows* yang menjadi input. Secara ringkas, “bentuk karakteristik” adalah bentuk umum dari sliding windows ketika berada pada fase tertentu. Penjelasan lebih rinci mengenai “bentuk karakteristik” akan dijelaskan pada akhir subbab ini.



Gambar 4.13: Output salah satu channel dari fungsi SlidingWindow dalam *pipeline*

Sumber: (Olahan Penulis, 2023)

Berikut adalah deskripsi lebih detail mengenai fungsi ini:

Deskripsi Fungsi:

- Deskripsi Output: *Sliding windows* dari data input yang disusun secara sekuensial yang setiap windowsnya sudah diterapkan min-max normalization
- Parameter fungsi: `window_size` = panjang *sliding windows* yang dipakai (int)
- Implementasi:

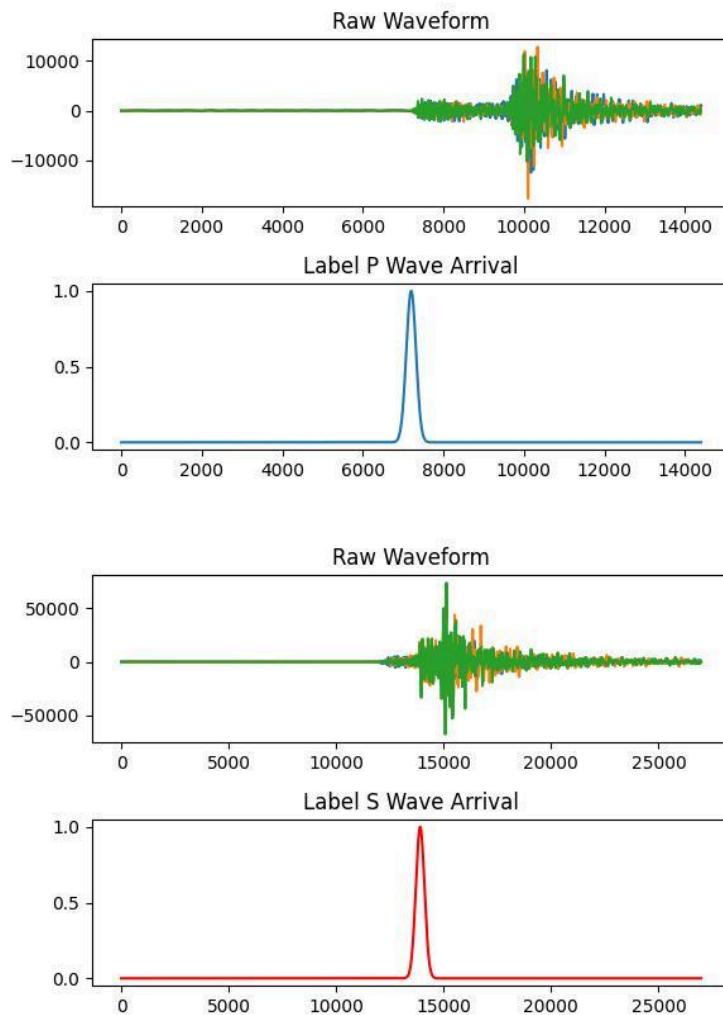
```
def slidingwindows(waveform, window_size):
    windows = []
    for i in range(len(waveform)-window_size):
        windows.append(
            min_max_normalize(waveform[i:window_size]))
    return windows
```

4.1.4 Labelling Data

Labelling data adalah sebuah tahapan yang harus dilakukan sebelum proses *training*. Model parameter sudah memiliki label yang terletak dalam file `metadata.csv` sehingga tidak memerlukan proses pelabelan yang lebih jauh. Akan tetapi, model gelombang belum memiliki label yang menandai kedatangan gelombang p/s secara langsung pada data. Oleh karena itu, penulis di sini melakukan pelabelan untuk dataset secara manual menggunakan sebuah *script* yang membaca `metadata` gempa.

Model gelombang membutuhkan sebuah mask yang dapat mengidentifikasi bagian mana dari *waveform* yang merupakan *p/s-arrival* dan bagian mana yang bukan. *Mask* yang dihasilkan akan berupa sebuah vektor dengan panjang yang setara dengan panjang *event* gempa dan di dalamnya terdapat nilai-nilai probabilitas kedatangan gelombang p/s pada saat tertentu.

Dalam proses ini, untuk pelabelan waktu kedatangan gelombang p/s, penulis menggunakan kurva seperti distribusi normal yang diterapkan pada mask. Kurva ini memiliki standar deviasi sebesar dua dan nilai *mean* (puncak dari kurva distribusi normal) yang berada pada lokasi *p/s-arrival*. Berikut adalah gambar yang mengilustrasikan hasil pelabelan kedatangan gelombang p/s terhadap sebuah data kejadian gempa.



Gambar 4.14: Pelabelan Mask P & S Wave Arrival

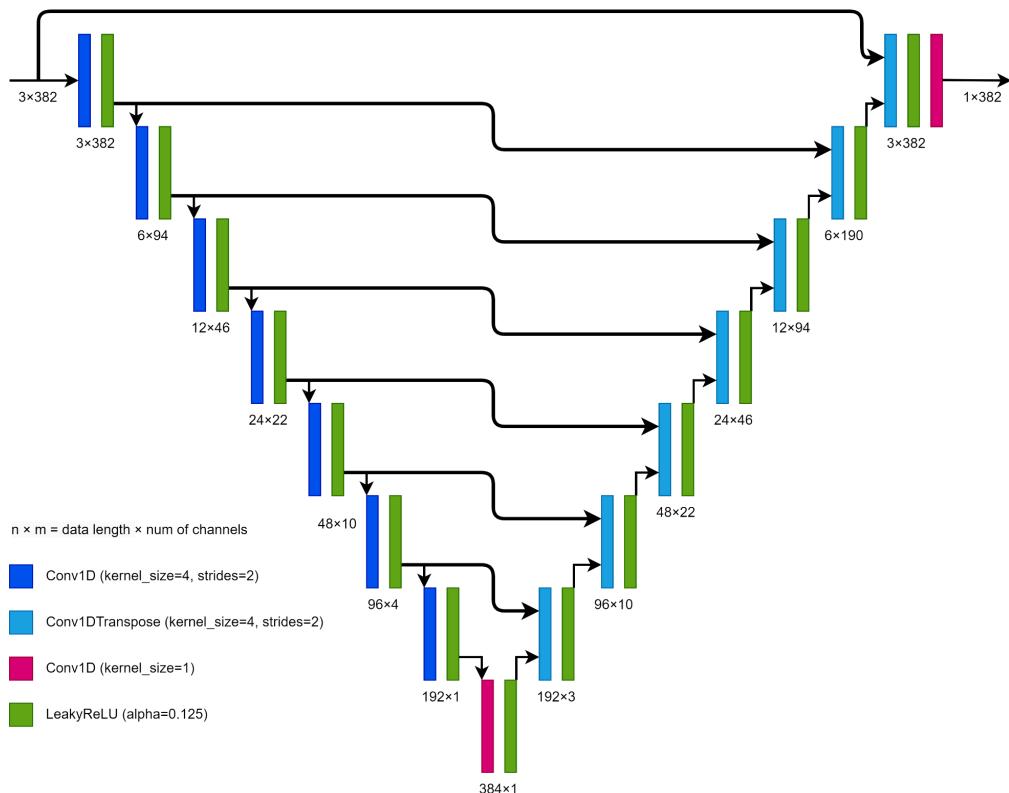
Sumber: (Olahan Penulis, 2023)

4.1.5 Arsitektur Model Deep Learning

Sebagaimana telah disebutkan sebelumnya, dua macam model, yakni model gelombang, dan model parameter. Model gelombang terdiri atas dua model, yaitu model deteksi kedatangan gelombang p, dan juga model deteksi kedatangan gelombang s. Pada bagian berikutnya, penulis akan memaparkan mengenai setiap arsitektur dari setiap jenis model. Model-model dari jenis yang sama nantinya akan memiliki arsitektur yang sama pula.

4.1.5.1. Arsitektur Model Gelombang

Untuk implementasi model ini, penulis memilih menggunakan arsitektur U-Net. Model U-Net diketahui dapat melakukan *task* segmentasi dengan sangat baik. *Task* yang dilakukan oleh model gelombang dapat dipandang sebagai *task* segmentasi karena *task*-nya adalah mengidentifikasi bagian mana dari data *waveform* yang merupakan kedatangan gelombang p/s, dan bagian mana yang bukan. Model gelombang akan dikembangkan dengan menggunakan *library* TensorFlow. Adapun arsitektur model ini dapat digambarkan dalam sebuah diagram untuk memberikan gambaran visual yang lebih jelas pada gambar 4.15.

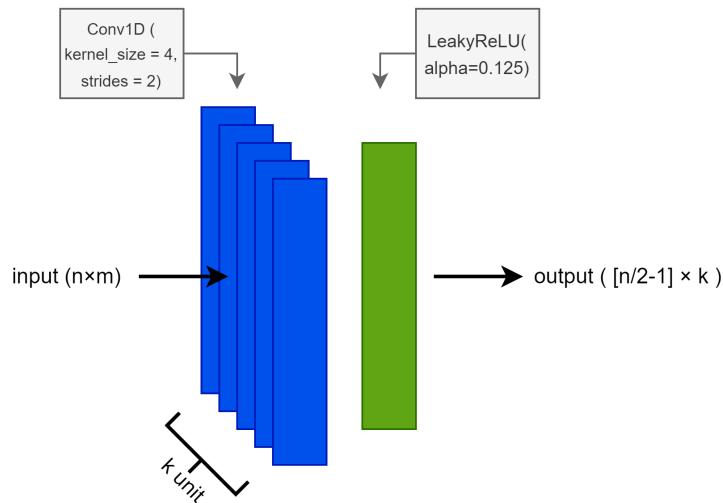


Gambar 4.15: Arsitektur Model Pendeksi Kedatangan Gelombang P & S

Sumber: (Olahan Penulis, 2023)

Model ini terdiri dari tiga bagian, yaitu bagian *encoder*, bagian tengah, dan bagian *decoder*. Bagian encoder terdiri atas unit-unit encoder yang tersusun atas sebuah unit konvolusi Conv1D dan unit fungsi aktivasi LeakyReLU. Unit Conv1D memiliki

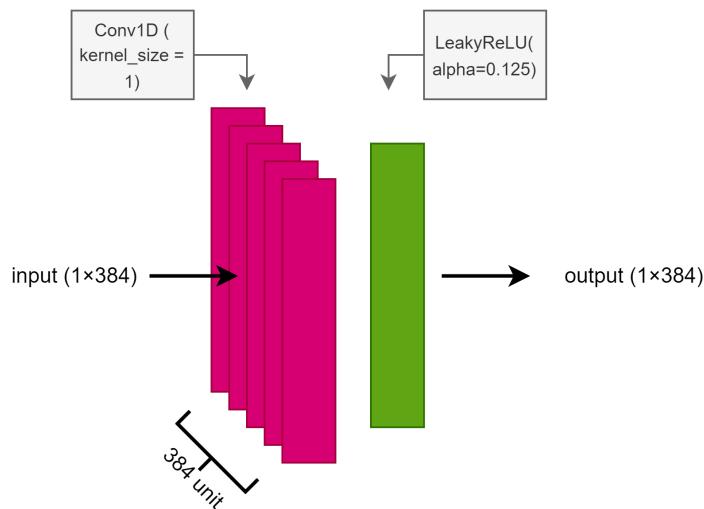
parameter `kernel_size` sebesar 4 dan juga `strides` sebesar 2. Penggunaan `strides` sebesar dua dilakukan untuk memangkas dari tensor input menjadi setengahnya. Unit LeakyReLU digunakan untuk menambahkan nonlinearitas pada tensor input. LeakyReLU digunakan karena unit ini terbukti memiliki performa yang lebih baik dibandingkan fungsi aktivasi ReLU biasa. Berikut adalah detail mengenai sebuah unit *encoder* beserta keterangan mengenai ukuran input dan outputnya.



Gambar 4.16: Unit Encoder

Sumber: (Olahan Penulis, 2023)

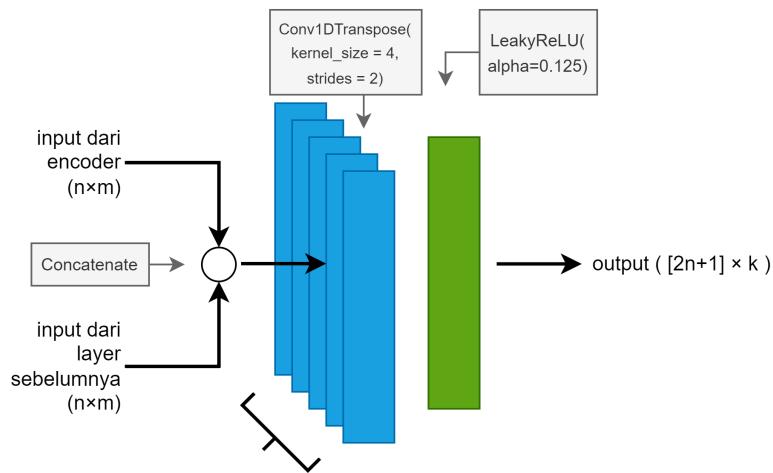
Setelah melalui tujuh lapis encoder, data yang awalnya memiliki bentuk 382×3 , sekarang memiliki bentuk 1×384 . Gelombang input sekarang sudah ter-*encode* menjadi sebuah *latent vector* berukuran 384. Oleh karena itu, sebelum dimasukkan kembali ke decoder, *latent vector* ini dimasukkan ke dalam sebuah *dense layer* berupa unit Conv1D yang memiliki `kernel_size` 1 dan unit aktivasi LeakyReLU. Penambahan bagian tengah ini diharapkan dapat mentransformasi *latent vector* menjadi bentuk lain sebelum dimasukkan ke dalam decoder. Berikut adalah detail mengenai bagian tengah beserta dengan ukuran input dan outputnya.



Gambar 4.17: Bagian Tengah

Sumber: (Olahan Penulis, 2023)

Pada bagian selanjutnya, barulah data memasuki bagian decoder. Bagian ini terdiri atas unit-unit decoder yang terdiri atas unit dekonvolusi Conv1DTranspose dan juga unit fungsi aktivasi LeakyReLU. Unit Conv1DTranspose memiliki parameter yang sama dengan unit Conv1D pada bagian encoder, begitu pula dengan unit LeakyReLU-nya. Akan tetapi, unit ini tidak hanya menerima satu input saja, akan tetapi dua input yang terdiri atas tensor dari unit decoder pada lapisan sebelumnya dan dari unit encoder yang memiliki “level yang sama” dengan unit decoder yang berkaitan, sehingga kedua input ini harus dikonkatentasi terlebih dahulu dengan menggunakan layer Concatenate. Berikut adalah detail dari unit decoder beserta detail mengenai input dan outputnya.



Gambar 4.18: Unit Decoder

Sumber: (Olahan Penulis, 2023)

Setelah keluar dari Unit Decoder ini, barulah tensor memiliki ukuran panjang yang sama dengan panjang tensor input. Pada bagian akhir model, penulis menambahkan sebuah unit Conv 1D dengan parameter kernel_size 1, banyak 1 unit, dan fungsi aktivasi sigmoid. Pemberian unit ini bertujuan untuk mentransformasi tensor menjadi satu dimensi dan nilainya berada di antara [0,1] sehingga bersesuaian dengan spesifikasi model yang mengharapkan luaran berupa karenanya model diharapkan untuk mengeluarkan sebuah vektor yang memiliki satu channel yang berisi nilai probabilitas keberadaan kedatangan gelombang p/s di suatu lokasi.

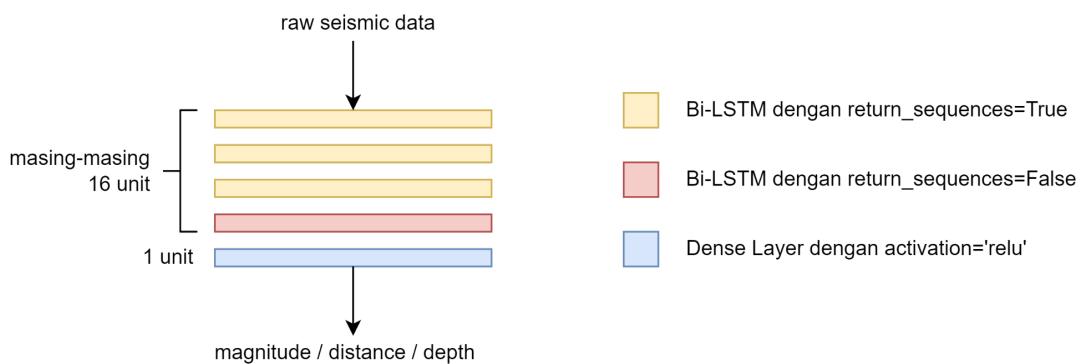
Sekian penjelasan penulis terhadap arsitektur model gelombang. Dengan ini, penulis berharap bahwa pembaca dapat memahami dengan detail tentang bagaimana arsitektur dari model ini bekerja.

4.1.5.2. Arsitektur Model Parameter

Dalam pengembangan model parameter, penulis menggunakan unit Bi-LSTM. Model ini terdiri dari empat lapisan layer yang terdiri atas unit-unit Bi-LSTM seperti diilustrasikan oleh gambar 4.19, dimulai dari layer input dengan bentuk (None, 3), yang kemudian diikuti dengan empat layer Bi-LSTM. Setiap layer LSTM memiliki 16 unit dan memiliki parameter `return_sequences=True` (kecuali untuk Bi-LSTM terakhir). Parameter `return_sequences` memungkinkan pengembalian output Bi-LSTM pada setiap

waktunya sehingga mengeluarkan sebuah sekuens baru yang pada setiap titik waktunya dalam bentuk vektor laten yang lebih *high-level*.

Pada lapisan terakhir, hasil output dari keempat layer LSTM dihubungkan dengan lapisan Dense yang memiliki satu unit, serta menggunakan fungsi aktivasi ReLU. Tujuan dari lapisan ini adalah untuk menghasilkan satu nilai skalar sebagai prediksi dari parameter gempa (magnitudo, jarak, atau kedalaman).

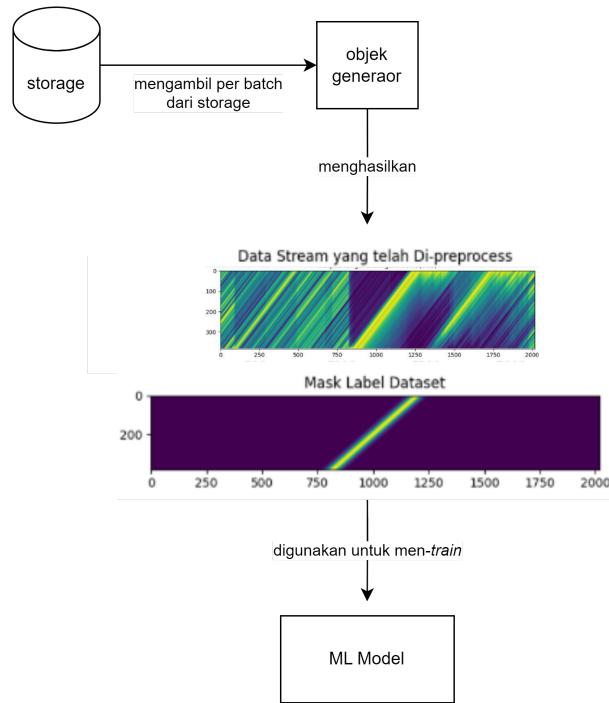


Gambar 4.19: Arsitektur Model Aproksimasi Parameter Gempa

Sumber: (Olahan Penulis, 2023)

4.1.6 Proses *Training*

Dataset ETHZ ini terdiri atas 22 GB data rekaman waveform event gempa. Dengan ukuran data sebesar ini, semua dataset waveform gempa tidak mungkin dibawa semuanya ke dalam memori untuk dilakukan proses training. Oleh karena itu, untuk menangani masalah ini, penulis membuat sebuah objek generator dari TensorFlow. Generator ini nantinya akan membaca file waveform.hdf5 beserta metadata.csv-nya dan menghasilkan sebuah objek Dataset yang bisa digunakan oleh model gelombang. Untuk dataset training model gelombang, objek generator ini juga akan meng-generate mask label untuk sebuah waveform yang nantinya akan digunakan untuk men-train model gelombang. Sementara itu, untuk dataset training model parameter, objek generator ini akan mengeluarkan paramater-parameter gempa yang juga akan digunakan untuk men-train model parameter. Berikut adalah diagram yang menggambarkan proses training dari setiap model. Proses training ini diilustrasikan oleh gambar 4.20.



Gambar 4.20: Proses Training

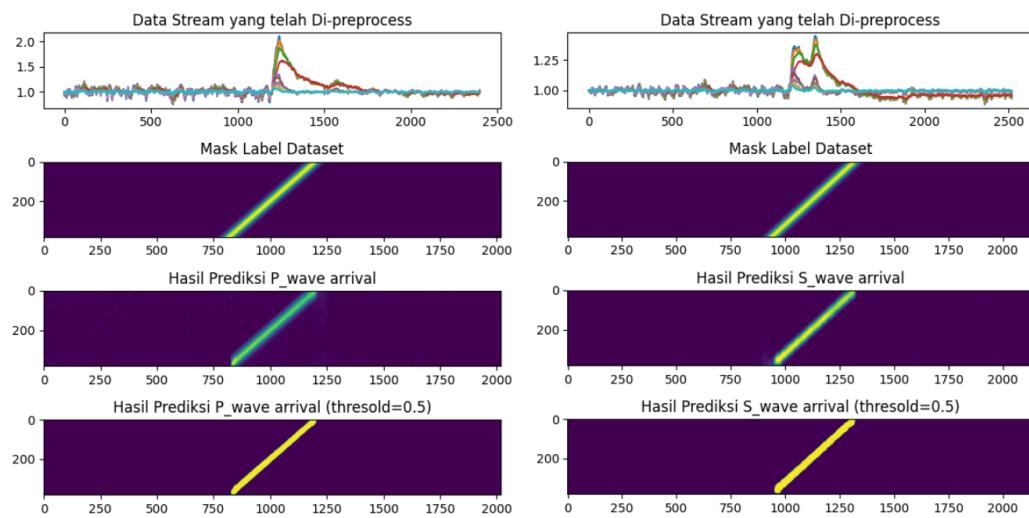
Sumber: (Olahan Penulis, 2023)

Untuk keperluan analisis, dataset ini dibagi menjadi dua subset utama: 80% untuk pelatihan (*train*) dan 20% untuk pengujian (*test*). Pembagian ini memungkinkan peneliti untuk melatih model pada sebagian besar data dan menguji kinerja model pada data yang belum pernah dilihat sebelumnya. Pendekatan ini diharapkan dapat memberikan pemahaman yang lebih baik tentang kemampuan model dalam mengenali dan meramalkan gempa berdasarkan karakteristik yang dipelajari dari data pelatihan. Dengan demikian, skripsi ini diarahkan pada pengembangan dan evaluasi model prediktif berdasarkan dataset gempa ETHZ yang telah dijelaskan secara rinci.

Proses training model dilakukan menggunakan *optimizer* Adam dengan *learning rate* sebesar 0.001, dan fungsi *loss* yang diterapkan adalah *Mean Squared Error*. Pelatihan dilaksanakan selama 15 *epoch*, di mana setiap *epoch* mengandung 1024 *steps*. Untuk memonitor potensi *overfitting*, proses *training* dilengkapi dengan evaluasi dengan menggunakan dataset test.

4.1.7 Output Hasil Inferensi

Setelah melalui tahapan *preprocessing* data, data stream dimasukkan ke dalam model *machine learning* untuk melakukan deteksi kedatangan gelombang p/s. Untuk model gelombang, gambar 4.21 adalah hasil luaran dari modelnya. Pada gambar ini, output dari inferensi model *machine learning*, berupa vektor sepanjang 382 yang disusun secara vertikal. Vektor-vektor ini juga diurutkan secara sekuenzial berdasarkan waktu, menciptakan gambar seperti yang terlihat di bawah ini.



Gambar 4.21: Hasil Inferensi Model Deteksi Kedatangan Gelombang P dan S

Sumber: (Olahan Penulis, 2023)

Sementara itu, untuk model parameter, model mengeluarkan sebuah nilai berupa nilai skalar yang menyatakan besaran aproksimasi dari parameter tersebut.

4.1.8. Postprocessing Hasil

Pada beberapa model, output yang dihasilkan tidak selalu sesuai dengan kebutuhan sistem Earthquake Early Warning System (EEWS). Pertama, EEWS memerlukan nilai boolean untuk mengetahui kedatangan gelombang p, sementara model yang relevan tidak menghasilkan nilai boolean. Kedua, salah satu parameter gempa yang diperlukan adalah koordinat gempa (lintang dan bujur), namun hanya terdapat model yang mengeluarkan jarak gempa dari sebuah stasiun. Ketiga, harapan adalah untuk menemukan satu set parameter tunggal untuk sebuah event gempa, namun kenyataannya terdapat banyak data parameter dengan nilai yang berbeda untuk satu

event gempa. Oleh karena itu, diperlukan tahap post-processing terhadap output setiap model agar dapat memperoleh nilai-nilai yang tidak dihasilkan secara langsung oleh model yang ada.

Pada bagian selanjutnya, penulis akan menjelaskan post-processing yang dilakukan, yang meliputi (1) mendapatkan sinyal kedatangan gelombang, (2) mendapatkan waktu picking, (3) mendapatkan koordinat gempa, dan (4) menggabungkan hasil perhitungan magnitudo dan lokasi beberapa event gempa.

4.1.8.1. Mendapatkan Sinyal Kedatangan Gelombang

Model deteksi kedatangan gelombang p/s mengeluarkan output berupa vektor yang berukuran (382, 1) yang berisi likelihood di mana p/s arrival berada. Akan tetapi, yang dibutuhkan oleh sistem Earthquake Early Warning System (EEWS) adalah satu nilai boolean yang menjadi sinyal untuk menandakan keberadaan gelombang p. Oleh karena itu, perlu menemukan cara untuk mengubah vektor output yang berukuran (382, 1) menjadi satu nilai boolean penanda eksistensi p/s arrival di dalam windows tersebut.

Nilai boolean tersebut dinamakan sebagai `p_arr`. Proses ini dilakukan dengan membandingkan setiap nilai dalam vektor dengan sebuah threshold `t`. Jika ada nilai pada windows yang melebihi threshold tersebut, maka nilai `p_arr` akan diatur menjadi true.

Untuk implementasi lengkap dari algoritma yang dimaksud, penulis dapat membaca Lampiran 1.1 yang terdapat pada lembar akhir dokumen ini.

```
def detect_wave_arrival(prediction, threshold):
    arrival_idx = np.where((prediction > threshold).any(axis=1))[0]
    return len(arrival_idx) > 0
```

4.1.8.2. Mendapatkan Picking Time

Picking time adalah lokasi di dalam windows di mana kedatangan gelombang p/s terdeteksi. Picking time dihitung ketika kedatangan gelombang p/s terdeteksi. Dari vektor output yang dihasilkan, picking time di dalam windows tersebut diambil sebagai puncak dari kurva distribusi normal yang terdapat di dalam vektor output (`argmax` vektor output).

Karena pada satu kali inferensi terdapat banyak vektor output (karena model menerima data stream, sehingga satu deteksi gelombang p/s bisa memiliki banyak

windows), maka windows-windows yang mengandung kedatangan gelombang p/s itu digabungkan dan dihitung rata-ratanya untuk menghasilkan hasil picking yang lebih presisi. Untuk implementasi lengkap dari algoritma yang dimaksud, penulis dapat membaca Lampiran 1.2 yang terdapat pada lembar akhir dokumen ini.

4.1.8.3. Mendapatkan Prediksi Koordinat

Model prediksi jarak, sesuai dengan namanya, menghasilkan hasil prediksi jarak antara episentrum gempa dengan stasiun perekam gempa. Untuk mendapatkan aproksimasi koordinat dari episentrum gempa, maka dibutuhkan paling sedikit tiga buah inferensi jarak dari stasiun-stasiun yang berbeda. Secara geometris, jika jarak antara setiap stasiun dapat diketahui dengan tepat, maka jika kita membuat sebuah lingkaran dengan setiap stasiun sebagai pusatnya dan jarak antara setiap stasiun ke episentrum sebagai jari-jarinya, maka ketiga lingkaran tersebut akan berpotongan di satu titik dan titik tersebut adalah episentrum gempanya. Proses ini diilustrasikan oleh gambar 4.22.



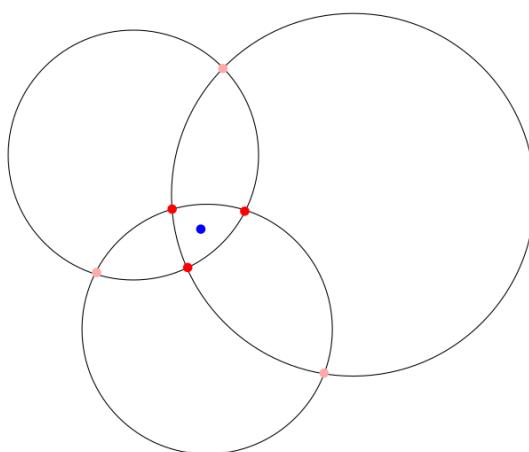
Gambar 4.22: Ilustrasi Penentuan Lokasi Episentrum Gempa dari Tiga Stasiun

Sumber: (<https://flexbooks.ck12.org/>, 2023)

Namun, pada kenyataannya, tidak mungkin mengetahui jarak eksak dari setiap stasiun ke episentrum. Selalu akan ada error yang menyebabkan ketiga lingkaran tidak berpotongan pada satu titik. Oleh karena itu, penulis mengembangkan algoritma untuk memperkirakan lokasi perpotongan ketiga lingkaran tersebut. Secara singkat, algoritma tersebut mencari tripel titik perpotongan antara dua lingkaran (untuk setiap pasang

lingkaran) dengan varians paling kecil di antara tripel-tripel titik perpotongan lainnya. Untuk implementasi lengkap dari algoritma yang dimaksud, penulis dapat membaca Lampiran 1.3 yang terdapat pada lembar akhir dokumen ini.

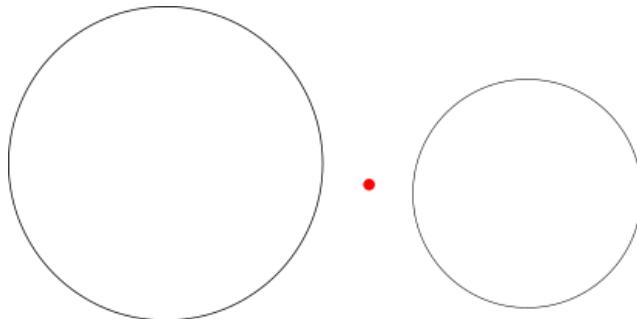
Gambar 4.23 adalah gambaran triplet titik yang dicari di dalam algoritma tersebut. Titik-titik berwarna merah merupakan titik yang terpilih sebagai triplet titik perpotongan yang memiliki varians paling rendah. Setelah titik-titik tersebut berhasil diidentifikasi, maka ketiganya dirata-ratakan untuk mendapatkan aproksimasi episentrum yang pada gambar diperlihatkan sebagai titik berwarna biru.



Gambar 4.23: Ilustrasi Algoritma Pencari Episentrum

Sumber: (Olahan Penulis, 2023)

Perhatikan bahwa terdapat kemungkinan ada sepasang lingkaran yang tidak memiliki titik potong seperti di gambar 4.24. Walaupun begitu, kedua lingkaran tersebut masih berpotongan di luar bidang euclidian riil dan memiliki koordinat perpotongan yang nilainya kompleks. Oleh karena itu, jika kasus ini terjadi, maka penulis akan mengambil hasil proyeksi koordinat perpotongan kompleks ke sumbu riilnya sebagai perwakilan dari “titik potong” untuk pasangan lingkaran tersebut, dan kedua titik perpotongan lingkaran ini nantinya akan saling berhimpit karena pada dasarnya titik perpotongan kedua lingkaran ini adalah saling *complex conjugate*, sehingga memiliki komponen riil yang sama.



Gambar 4.24: Penentuan Perwakilan dari Lingkaran yang Tidak Berpotongan

Sumber: (Olahan Penulis, 2023)

4.1.4.4 *Post-processing Statistik Parameter Gempa*

Pada sistem EEWS ini, setiap stasiun memiliki hasil inferensi masing-masing terkait parameter-parameter suatu event gempa. Penting untuk menggabungkan kumpulan hasil inferensi ini agar hanya mendapatkan satu set parameter untuk satu event gempa. Jika dalam rentang 10 menit terakhir terdapat stasiun-stasiun yang berdekatan (dalam threshold <500 km) yang mendeteksi kedatangan gelombang p, maka hasil deteksi dari beberapa stasiun tersebut dapat dianggap sebagai rekaman dari satu kejadian gempa yang sama.

Untuk setiap stasiun berdekatan yang mendeteksi satu event gempa yang sama, dilakukan pengambilan beberapa triplet yang mungkin dari kumpulan stasiun tersebut dan menghitung lokasi dari episentrum gempa. Selanjutnya, setiap perhitungan lokasi tersebut dirata-ratakan untuk mendapatkan satu nilai sebagai representasi lokasi dari gempa. Di samping itu, hasil perhitungan magnitudo dari setiap stasiun juga dirata-ratakan untuk mendapatkan representasi magnitudo dari event gempa tersebut.

4.1.9. *Deployment Model ML*

Proses deployment dilakukan menggunakan BentoML, sebuah platform yang memfasilitasi deployment model machine learning. Pengguna hanya perlu memberikan modelnya kepada Bentoml dan membuat service untuk inferensinya. Saat service inference dijalankan, BentoML secara otomatis membuat beberapa instance service untuk inferensi dan melakukan optimasi lainnya untuk memaksimalkan penggunaan sumber daya.

Machine Learning service yang di-deploy mencakup semua tahapan dari preprocessing data, inferensi, hingga postprocessing data inferensi. Oleh karena itu, service lain pada sistem backend di EEWs hanya perlu mengirimkan data seismik mentah ke service machine learning ini. Service ini akan menghasilkan hasil yang sudah diolah, siap digunakan oleh service lain pada backend tanpa perlu memikirkan postprocessing data hasil inferensi.

Service ini memiliki dua buah *endpoint* yang dapat digunakan, yaitu *endpoint* “/predict” untuk mendapatkan hasil inferensi deteksi kedatangan gelombang p/s, *endpoint* “/approx_earthquake_statistics” untuk mendapatkan hasil hitungan untuk parameter-parameter gempa (magnitudo, jarak, dan kedalaman), dan *endpoint* “/recalculate” untuk menghitung koordinat episentrum gempa dan rata-rata dari magnitudo dan kedalaman menggunakan hasil inferensi dari tiga stasiun. Berikut adalah luaran yang dihasilkan oleh masing-masing *endpoint* (sebuah JSON) beserta keterangannya dari setiap nilainya.

- /predict

Tabel 4.2: Skema Data JSON *Endpoint* /predict

| Nama & Tipe Data | Keterangan |
|-------------------|---|
| station_code: str | Kode stasiun yang memanggil endpoint ini |
| init_end: bool | Flag penanda apakah tahapan inisialisasi selesai |
| p_arr: bool | Flag penanda kedatangan apakah terdeteksi p arrival |
| p_arr_time: str | Pick time kedatangan gelombang p |
| p_arr_id: int | ID p arrival |
| new_p_event: bool | Flag penanda apakah deteksi bersifat baru/lanjutan |
| s_arr: bool | Flag penanda kedatangan apakah terdeteksi s arrival |
| s_arr_time: str | Pick time kedatangan gelombang s |
| s_arr_id: int | ID s arrival |
| new_s_event: bool | Flag penanda apakah deteksi bersifat baru/lanjutan |

- /approx_earthquake_statistics

Tabel 4.3: Skema Data JSON *Endpoint* /approx_earthquake_statistics

| Nama & Tipe Data | Keterangan |
|-------------------|--|
| station_code: str | Kode stasiun yang memanggil endpoint ini |
| magnitude: float | Prediksi magnitudo gempa |
| distance: float | Prediksi jarak gempa |
| depth: float | Prediksi kedalaman gempa |

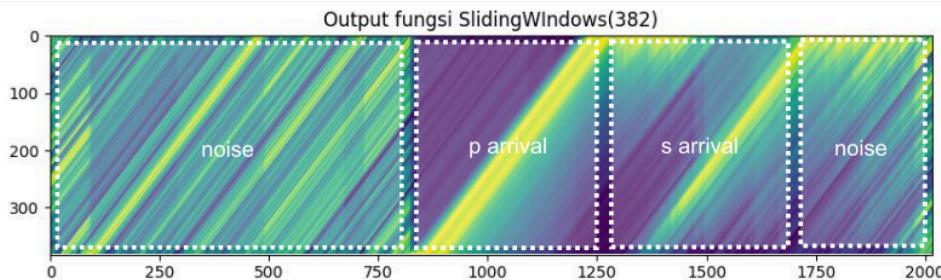
- /recalculate

Tabel 4.4: Skema Data JSON *Endpoint* /recalculate

| Nama & Tipe Data | Keterangan |
|--------------------------|---|
| station_codes: List[str] | List tiga kode stasiun yang digunakan untuk kalkulasi koordinat gempa |
| magnitude: float | Prediksi magnitudo gempa |
| distance: float | Prediksi jarak gempa |
| depth: float | Prediksi kedalaman gempa |

4.1.10 Hipotesis Penulis tentang Karakteristik Data yang telah di-*Preprocess*

Seperti yang telah dijelaskan sebelumnya, proses preprocessing data dilakukan untuk menonjolkan fitur-fitur yang berhubungan dengan kedatangan gelombang p/s, sehingga memungkinkan pembuatan model dengan parameter yang minimal. Pada bagian ini, penulis akan menyampaikan pendapat mengenai mengapa tahap preprocessing ini dapat membantu menonjolkan fitur-fitur pada data.



Gambar 4.25: Karakteristik Distribusi Nilai di Sliding Windows untuk Setiap Fasa

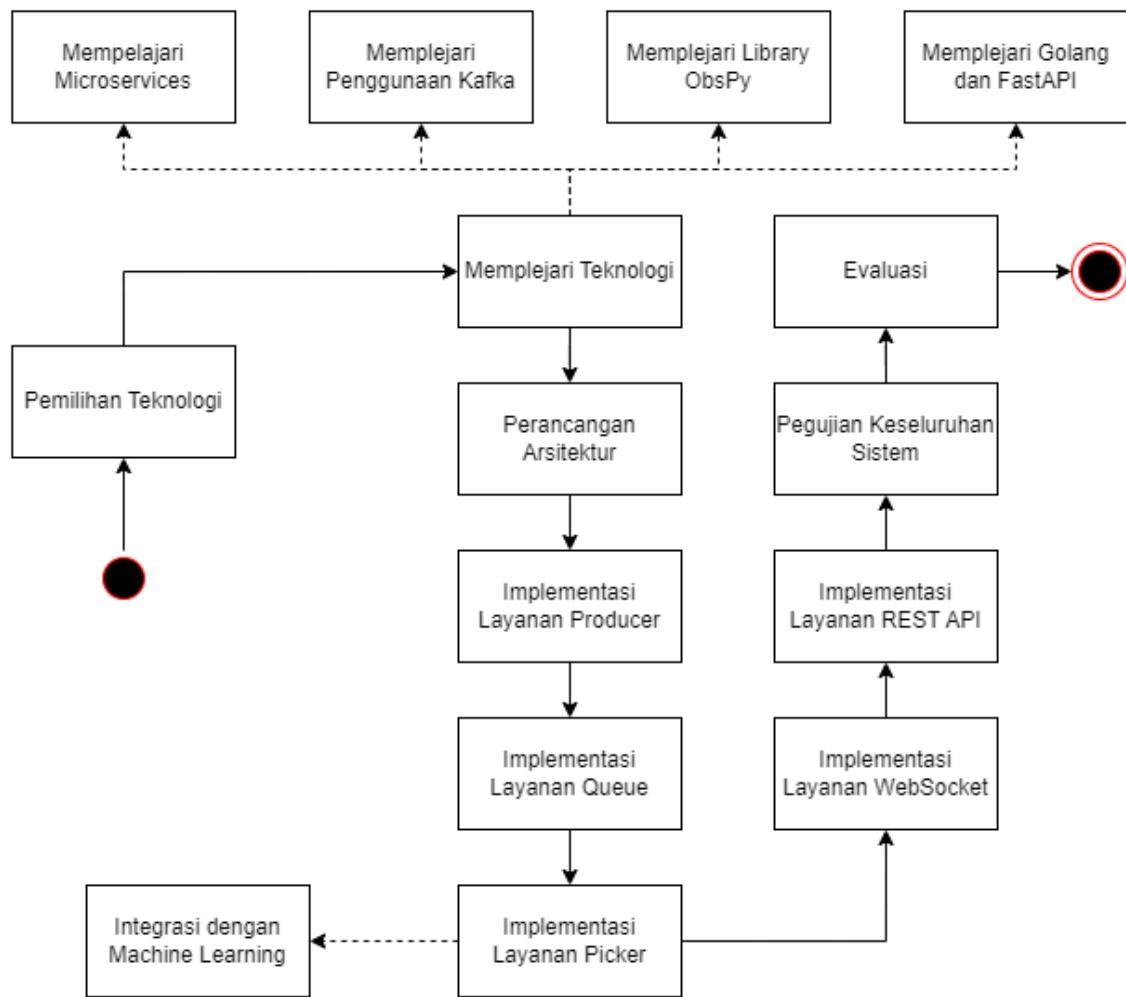
Sumber: (Olahan Penulis, 2023)

Gambar 4.25 menunjukkan data gambar dari salah satu channel hasil tahapan preprocessing sebelumnya. Di sini, penulis membagi gambar menjadi tiga bagian, yaitu bagian noise, p-arrival, dan s-arrival. Setiap bagian memiliki nilai *sliding windows* dengan karakteristik yang berbeda.

Bagian *noise* memiliki bentuk yang acak. Bagian *p-arrival* memiliki bentuk yang sangat terpolarisasi, dengan nilai yang berbeda secara signifikan antara bagian bawah dan bagian atas sliding windows, membentuk pola segitiga seperti yang terlihat pada gambar. Bagian *s-arrival* memiliki bentuk yang mirip dengan bagian *p-arrival*, tetapi sedikit lebih ber-noise, dan pada bagian atasnya seringkali terdapat nilai yang tinggi. Karena bentuk karakteristik sliding windows yang berbeda-beda pada setiap bagian, penulis berpendapat bahwa model menjadi lebih mudah dalam membedakan antara noise dan bagian gelombang datang, serta mengidentifikasi lokasi kedatangan gelombang pada sliding windows.

4.2 Implementasi *Backend*

Pada subbab ini akan diuraikan tentang proses pengimplementasian sistem *Earthquake Early Warning System* (EEWS) pada sisi *backend*. *Backend* sistem ini merupakan inti dari pemrosesan data seismik dan merupakan fondasi yang memungkinkan EEWS beroperasi secara efektif. Implementasi *backend* mencakup beberapa komponen utama yang akan dijelaskan pada bab ini. Pentingnya pengimplementasian ini tidak bisa diabaikan karena melalui serangkaian proses yang terjadi di sisi server, sistem EEWS mampu mengolah dan mengirimkan data dengan kecepatan yang diperlukan.



Gambar 4.26: Proses Pembuatan Sistem *Backend*

Sumber: (Olahan Penulis, 2023)

Proses pengimplementasian sistem *backend* melibatkan beberapa tahapan penting sampai mendapatkan sistem yang sesuai dengan kebutuhan penelitian ini seperti pada gambar 4.26. Proses diawali dengan memilih teknologi yang sesuai dengan kebutuhan untuk menciptakan sistem EEWS yang mumpuni. Penulis kemudian melakukan pembelajaran teknologi-teknologi yang sudah dipilih tersebut. Ketika penulis sudah memahami semua teknologi yang akan dipakai, penulis merancang arsitektur yang akan dibangun. Setelah rancangan arsitektur dibuat, penulis melakukan implementasi setiap komponen *backend*, seperti Producer, Queue, Picker, WebSocket, dan REST API. Ketika semua komponen sudah dibuat dan diintegrasikan dengan baik, penulis melakukan pengujian terhadap sistem yang sudah dibangun. Pengujian akan menghasilkan data-data yang nantinya akan penulis evaluasi.

4.2.1 Pemilihan Teknologi

Pemilihan teknologi dalam pengembangan sistem Earthquake Early Warning System (EEWS) yang penulis rancang sangat dipengaruhi oleh kebutuhan sistem untuk mengolah dan memproses data seismik dengan kecepatan tinggi. Hal ini sangat krusial untuk memastikan bahwa deteksi gempa yang akurat dapat segera dikomunikasikan kepada pengguna akhir tanpa penundaan yang berarti. Dalam menentukan teknologi yang sesuai, penulis tidak hanya mempertimbangkan performa kecepatan tetapi juga spesifikasi teknologi dalam menangani tugas-tugas spesifik.

Teknologi harus dapat mengumpulkan data seismik dari sumber dengan cepat, mengekstrak informasi yang relevan, dan memprosesnya untuk analisis lebih lanjut. Dengan demikian, pemilihan teknologi juga berfokus pada aspek skalabilitas, ketahanan terhadap kegagalan (*fault tolerance*), dan kemampuan untuk memproses aliran data yang besar secara *real-time*. Ini mencakup kemampuan untuk mengelola beban kerja yang bervariasi, dari operasi penyimpanan dan pengambilan data yang efisien hingga penyediaan antarmuka pengguna yang responsif. Setiap teknologi yang dipilih harus memenuhi persyaratan ini untuk memastikan bahwa sistem EEWS yang penulis bangun dapat beroperasi dengan standar yang ditetapkan, memberikan layanan yang andal dalam situasi kritis.

4.2.1.1 Apache Kafka

Apache Kafka adalah platform distribusi streaming yang memainkan peran krusial dalam arsitektur EEWS. Kafka dipilih karena keunggulan dalam memproses aliran data besar secara real-time, yang sangat penting dalam konteks pendekripsi gempa bumi. Kafka menawarkan kemampuan *fault-tolerant*, skalabilitas horizontal, dan *throughput* tinggi, yang memungkinkan sistem untuk menangani volume data yang besar tanpa kehilangan performa. Selain itu, Kafka memiliki fitur replikasi dan partisi yang memastikan bahwa data dapat diproses secara paralel dan tetap tersedia meskipun terjadi kegagalan pada bagian dari sistem. Kemampuan ini menjadikan Kafka pilihan yang handal untuk memastikan distribusi data seismik yang efisien dan aman dalam sistem EEWS.

Kafka memfasilitasi komunikasi antara layanan dengan cara yang efisien dan andal. Penulis mengimplementasikan Kafka sebagai pusat pengaturan event,

memanfaatkan model publish dan subscribe untuk memungkinkan layanan saling berkomunikasi. Setiap layanan dapat mempublikasikan event ke topik-topik yang telah ditentukan di dalam Kafka, sementara layanan lain dapat melakukan subscribe topik-topik tersebut untuk menerima event. Kafka juga memberikan fitur-fitur seperti durabilitas, skalabilitas, dan kemampuan untuk memproses event dalam urutan yang tepat. Fitur-fitur tersebut sangat penting untuk integritas sistem peringatan dini gempa.

4.2.1.2 MongoDB

MongoDB, sistem database NoSQL yang berorientasi dokumen, diintegrasikan ke dalam *backend* EEWS karena fleksibilitasnya yang besar dalam menangani struktur data yang bervariasi. MongoDB juga menyediakan fitur indeksasi yang efisien dan kemampuan *query* yang cepat, keduanya sangat penting untuk pengambilan data historis selama analisis dan audit.

Dalam konteks seismologi, data yang dikumpulkan dapat sangat heterogen dan MongoDB mampu menyimpan data tersebut dengan skema yang dinamis. Hal ini memungkinkan penyimpanan dan pengambilan data secara cepat tanpa perlu melakukan banyak modifikasi skema *database*.

4.2.1.3 Redis

Redis digunakan sebagai sistem penyimpanan berbasis kunci-nilai yang disimpan langsung di dalam memori yang berperan sebagai cache dalam sistem EEWS. Redis dipilih karena latensi yang sangat rendah dan kemampuan untuk menangani ratusan ribu operasi per detik, yang membuatnya ideal untuk caching hasil prediksi gelombang P dan S yang harus diakses dengan cepat oleh komponen lain dalam sistem. Redis juga mendukung struktur data yang kaya, seperti strings, hashes, lists, sets, dan sorted sets dengan range queries, yang memberikan fleksibilitas dalam pengelolaan data yang di-cache.

4.2.1.4 FastAPI

Dalam pemilihan teknologi *backend* untuk sistem Earthquake Early Warning System (EEWS), FastAPI yang merupakan *library* dari Python muncul sebagai pilihan yang sangat sesuai, tidak hanya karena kemampuannya untuk menangani permintaan secara asinkron dengan cepat, tetapi juga karena fitur *background tasks* yang memungkinkan

eksekusi tugas-tugas di latar belakang. Fitur ini sangat penting untuk sistem EEWS yang membutuhkan pemrosesan dan pengolahan data seismik secara *real-time*.

Dengan fitur *background task*, FastAPI dapat menjalankan proses pengambilan dan analisis data seismik secara paralel sementara server tetap responsif terhadap permintaan pengguna lainnya. Selain itu, kompatibilitas dengan *library* ObsPy juga penting, karena *library* ini secara khusus digunakan untuk mengambil data seismik dari jaringan GEOFON.

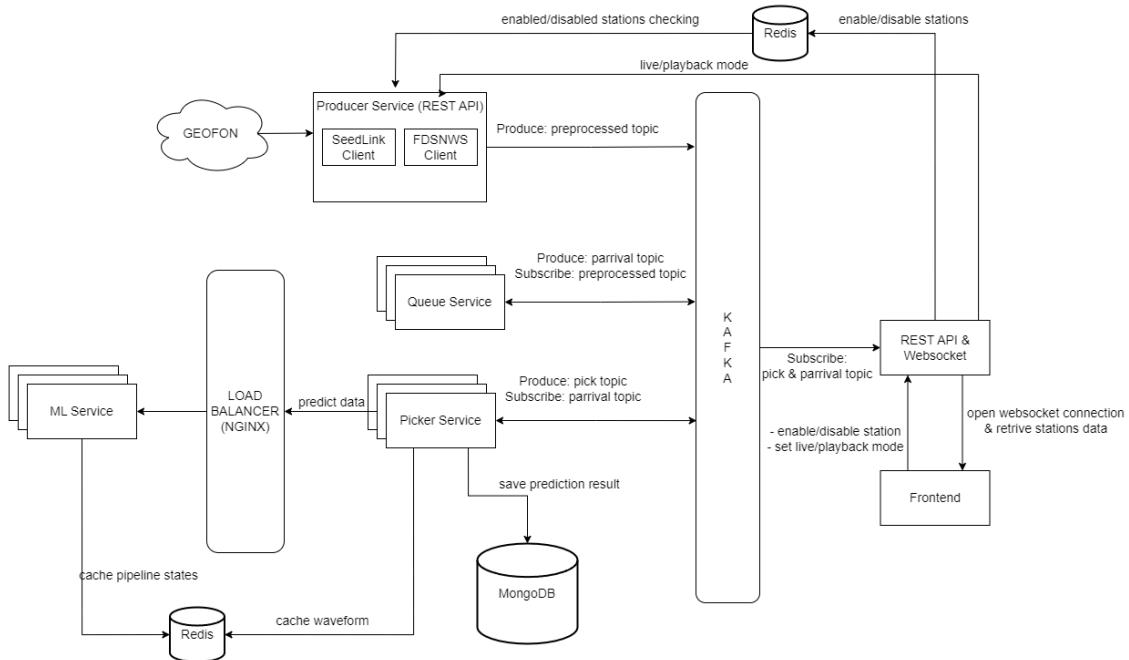
4.2.1.5 Golang

Golang atau Go adalah bahasa pemrograman yang dikembangkan oleh Google yang terkenal dengan performa konkurensinya yang tinggi dan penggunaan memori yang efisien. Golang menjadi pilihan untuk pengembangan REST API dan layanan WebSocket karena kemampuannya dalam menangani ribuan hingga jutaan koneksi simultan tanpa penurunan kinerja yang signifikan.

Hal ini berkat model goroutines yang ringan dan efisien, yang memungkinkan operasi *non-blocking I/O* dan pengelolaan *state* secara bersamaan. Kecepatan kompilasi dan eksekusi yang tinggi, serta alat manajemen memori yang otomatis seperti *garbage collection*, menjadikan Golang pilihan yang tepat untuk memastikan bahwa komunikasi antara *backend* dan *frontend* berjalan dengan lancar dan tanpa hambatan.

4.2.1 Arsitektur Sistem *Backend*

Penulis memilih pendekatan arsitektur *microservices* untuk membangun sistem ini. Hal ini didasari pada kebutuhan untuk memiliki sistem yang dapat dikembangkan dan dipelihara dengan lebih mudah. Dengan memisahkan sistem menjadi serangkaian layanan yang lebih kecil dan independen, penulis dapat mengisolasi tanggung jawab fungsionalitas yang memungkinkan penyesuaian, pengujian, dan penyebaran yang lebih cepat dan terkontrol. Setiap layanan dirancang untuk menjalankan satu set tugas spesifik, beroperasi dalam kontainerisasi yang memungkinkan mereka untuk dikerahkan secara individual tanpa mengganggu komponen lain dari sistem. Gambaran arsitektur sistem dapat dilihat pada Gambar 4.27.

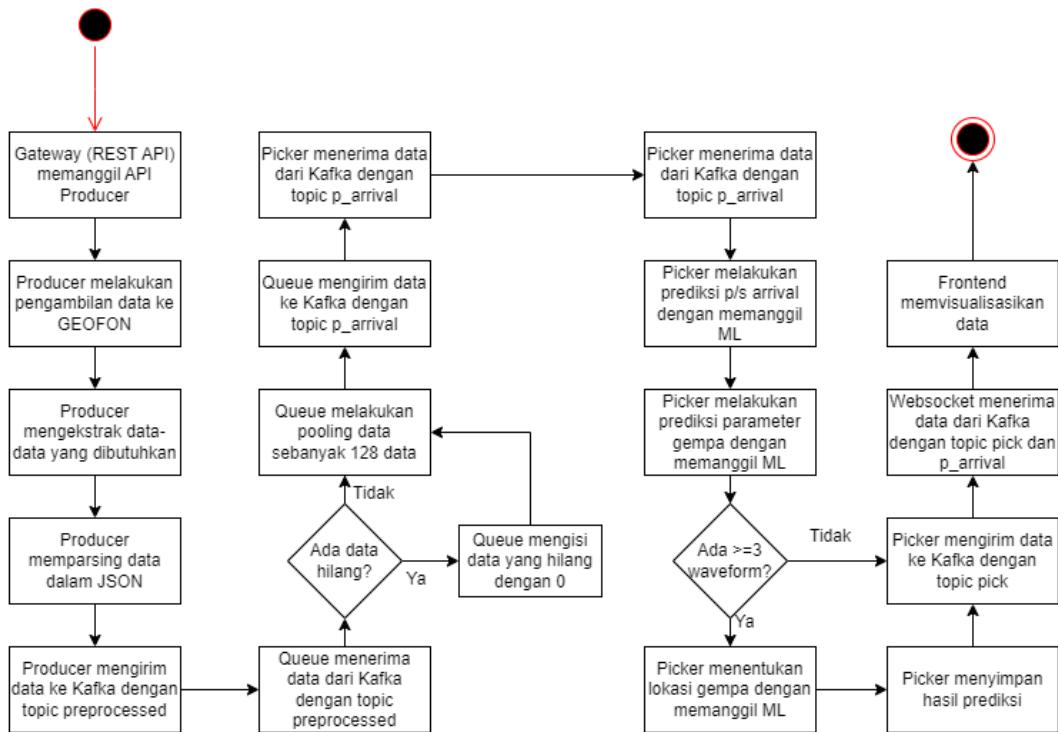


Gambar 4.27: Rancangan Arsitektur EEWS

Sumber: (Olahan Penulis, 2023)

Penggunaan arsitektur yang proses eksekusi bisnisnya didorong oleh event (*event-driven architecture*) dipilih untuk memastikan bahwa sistem dapat merespons secara dinamis terhadap peristiwa atau data yang masuk. Dalam konteks sistem peringatan dini gempa, responsivitas ini kritikal. Penulis mendesain agar setiap komponen layanan dapat bereaksi terhadap peristiwa yang dikomunikasikan melalui *event*. Ini membantu dalam memastikan bahwa tidak ada latensi yang tidak perlu antara deteksi sinyal gempa dan tindakan yang diambil oleh sistem.

Pada sistem EEWS yang penulis buat, terdapat 5 layanan utama yang saling bekerja sama untuk membangun sistem pendekripsi dini gempa. Setiap layanan memiliki keterikatan yang *loosely-coupled* yang memungkinkan modifikasi atau peningkatan pada salah satu komponen tanpa mengganggu komponen lain. Ini menjamin fleksibilitas dan kemudahan dalam pemeliharaan serta skalabilitas sistem.



Gambar 4.28: Alur Kerja Sistem

Sumber: (Olahan Penulis, 2023)

Seperti pada gambar 4.28, dalam sistem Earthquake Early Warning System (EEWS), proses pendekripsi gempa dimulai dengan pengambilan data seismik oleh Producer dari GEOFON, yang selanjutnya diproses dan dikirimkan ke Kafka. Data ini diolah oleh Queue Service yang mengumpulkan data sebelum dikirim ke topik *p_arrival*. Picker kemudian menggunakan data ini untuk melakukan prediksi awal gelombang P dan S dengan model ML, setelah mendekripsi gelombang yang cukup, menghitung parameter gempa. Hasilnya dikirim kembali ke Kafka dan di-cache di Redis, sedangkan Frontend menerima dan memvisualisasikan data tersebut melalui WebSocket, memungkinkan pengguna untuk melihat hasil analisis secara *real-time*.

Layanan pertama adalah Producer, yang memiliki peran kritis dalam mengumpulkan data dari sensor GEOFON. Tugas utamanya adalah memproses data mentah tersebut dan mengkonversinya ke format JSON yang standar. Setelah proses konversi, data yang telah diproses tersebut kemudian dipublikasikan ke topik *preprocessed* pada Kafka, memastikan bahwa data siap diproses oleh komponen selanjutnya dalam sistem.

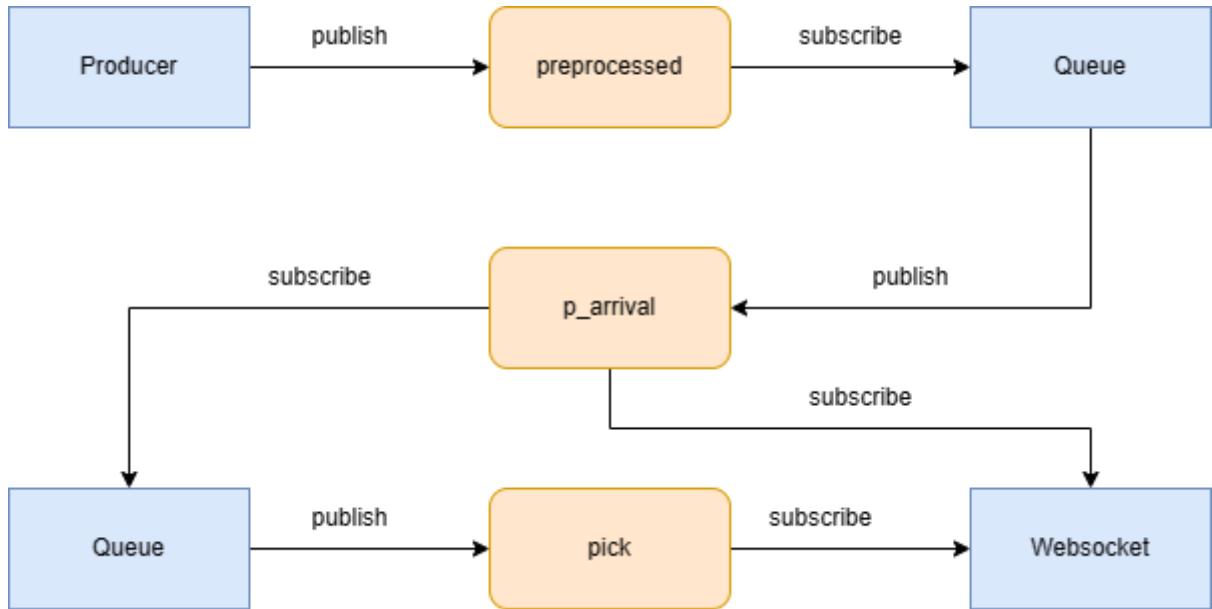
Layanan kedua adalah Queue, bertindak sebagai manajer antrian yang mengambil data dari topik *preprocessed*. Dalam perannya, Queue mengatur data ini dalam batch-batch yang terorganisir, yang kemudian siap untuk diproses lebih lanjut. Setiap *batch* data tersebut dengan cermat dipublikasikan ke topik *p_arrival*, yang menandai langkah selanjutnya dalam alur kerja pendeksi gempa.

Layanan ketiga, Picker, memainkan peran yang sangat penting dalam sistem ini. Picker mengkonsumsi data dari topik *p_arrival*. Layanan ini melakukan prediksi awal terkait gelombang P dan S. Ini dilakukan dengan bantuan layanan ML yang telah dioptimalkan untuk tugas ini. Setelah berhasil mendapatkan prediksi, hasilnya kemudian disimpan ke dalam Redis. Ketika sudah ada tiga buah gelombang P atau S dalam waktu dan jarak yang berdekatan, Picker akan melakukan prediksi parameter gempa dan menyimpan hasilnya ke MongoDB. Selain itu, Picker juga bertanggung jawab untuk mempublikasikan informasi ini ke topik *pick*, yang menjadikan data tersedia untuk layanan lain.

Layanan keempat, Machine Learning, merupakan jantung dari kapabilitas prediktif sistem. Di dalamnya, model-model *machine learning* canggih diterapkan dan siap digunakan untuk memprediksi karakteristik gelombang seismik dan parameter gempa penting lainnya. Penyediaan prediksi yang akurat adalah kunci dari keberhasilan sistem ini dalam menyediakan peringatan dini yang dapat diandalkan.

Terakhir, REST API & WebSocket berfungsi sebagai penghubung vital antara *backend* dan *frontend*. Layanan ini mengelola koneksi *websocket* yang memungkinkan transmisi data secara *real-time* kepada pengguna, memastikan bahwa informasi kritis disampaikan tanpa penundaan. Selain itu, REST API memungkinkan akses terprogram ke data stasiun gempa sehingga memperluas kemampuan sistem untuk berintegrasi dengan aplikasi dan layanan lain.

Penulis merancang interaksi antar layanan dengan memanfaatkan model *publish* dan *subscribe* yang disediakan oleh Kafka sehingga setiap layanan dapat berkomunikasi secara efisien dan independen. Gambaran ilustrasi interaksi antar sistem dapat dilihat pada gambar di bawah.



Gambar 4.29: Interaksi antar Layanan Melalui Topik Kafka

Sumber: (Olahan Penulis, 2023)

Untuk manajemen *state* yang efektif, Redis dipilih sebagai solusi. Redis memungkinkan penyimpanan dan pengelolaan *state* sementara yang diperlukan selama proses pemilihan gelombang P dan S. Selain itu, MongoDB juga akan digunakan untuk menyimpan hasil prediksi dari *machine learning* yang dilakukan oleh Picker.

4.2.2 Desain Data JSON

Data yang diperoleh dari GEOFON dikonversi menjadi format JSON untuk memudahkan serialisasi dan deserialisasi data yang ditransmisikan melalui Kafka. JSON dipilih karena kemudahan pembacaannya oleh manusia dan mesin, serta dukungannya yang luas dalam berbagai bahasa pemrograman.

Penulis menyediakan skema JSON (Tabel 4.5) yang digunakan untuk memformat data seismik. Skema ini meliputi atribut-atribut penting seperti *starttime*, *endtime*, kumpulan nilai amplitudo, *sampling rate*, dan metadata sensor.

Tabel 4.5: Skema Data JSON

| Nama Atribut | Tipe Data | Deskripsi |
|--------------|-----------|------------------------------|
| network | string | Nama jaringan yang digunakan |

Tabel 4.5: Skema Data JSON (sambungan)

| Nama Atribut | Tipe Data | Deskripsi |
|---------------|-----------|--|
| station | string | Nama stasiun gempa |
| channel | string | Nama <i>channel</i> |
| starttime | timestamp | Waktu mulai pengambilan data |
| endtime | timestamp | Waktu akhir pengambilan data |
| sampling_rate | integer | Laju pengambilan data |
| npts | integer | Jumlah titik data |
| data | array | Kumpulan titik data yang merepresentasikan amplitudo |

4.2.3 Desain Topik Kafka

Penulis menggunakan topik Kafka sebagai kanal komunikasi antara berbagai layanan dalam arsitektur *microservices*. Setiap topik dirancang untuk menangani jenis data atau *event* tertentu dalam proses EEWS (dapat dilihat pada Tabel 4.6).

Tabel 4.6: Deskripsi Topik Kakfa

| Nama Topik | Fungsi | Producer | Subscriber |
|--------------|--|----------|-------------------|
| preprocessed | Digunakan untuk menyalurkan data seismik yang telah diproses oleh layanan Producer | Producer | Queue |
| p_arrival | Menerima batch data dari Queue Service yang siap untuk deteksi gelombang P dan S oleh layanan Picker | Queue | Picker, Websocket |
| pick | Menampung hasil prediksi gelombang P dan S serta parameter gempa yang dikirim oleh layanan Picker | Picker | Websocket |

Kafka menyediakan kemampuan untuk membagi topik menjadi beberapa partisi, yang memungkinkan data untuk diproses secara paralel, meningkatkan throughput, dan menurunkan latensi. Replikasi partisi menambah reliabilitas dan toleransi terhadap kegagalan. Konfigurasi partisi dan replikasi untuk setiap topik dapat dilihat pada Tabel 4.7.

Tabel 4.7: Topik, Partisi, dan Replika

| Nama Topik | Jumlah Partisi | Jumlah Replika |
|--------------|----------------|----------------|
| preprocessed | 3 | 3 |
| p_arrival | 3 | 3 |
| pick | 3 | 3 |

Desain data dan konfigurasi topik Kafka yang telah penulis kembangkan adalah inti dari *backend* sistem EEW, memastikan bahwa data dapat diproses dengan cepat dan andal. Struktur data yang standar dan pengelolaan topik yang terorganisir memungkinkan layanan yang berbeda untuk berkomunikasi dengan efisien, sementara konfigurasi partisi dan replikasi menjamin bahwa sistem dapat diskalakan dan tetap berfungsi bahkan jika terjadi kegagalan komponen.

4.2.4 Implementasi Layanan Producer

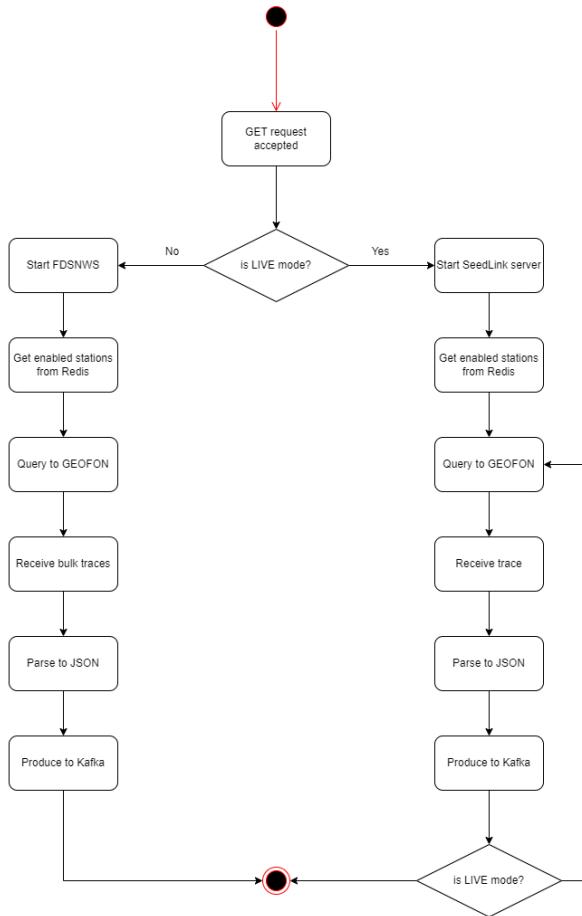
Layanan ini berbasis web yang menyediakan dua *endpoint* untuk melakukan inisiasi pengambilan data. Kedua *endpoint* ini nantinya hanya akan digunakan oleh layanan REST API yang berperan sebagai *gateway* dan tidak akan diekspos ke luar lingkungan *microservices*. *Endpoint-endpoint* yang ada pada layanan Producer dapat dilihat lebih detail pada Tabel 4.8.

Tabel 4.8: Endpoint Producer

| Endpoint | Method | Query Params |
|-----------|--------|--|
| /live | GET | - |
| /playback | GET | starttime: waktu mulai pengambilan data endtime: waktu akhir pengambilan data |
| /stop | POST | - |

Pembuatan dua endpoint tersebut didasari karena adanya dua mode pengambilan data yang berbeda sehingga pengguna akhir dapat memilih untuk memilih mode yang mereka inginkan. Kedua mode tersebut adalah LIVE dan PLAYBACK. Perbedaan utama kedua mode ini adalah dalam penggunaan sumber data. Mode LIVE menggunakan seedlink sebagai sumber datanya, sedangkan mode PLAYBACK menggunakan fdsnws. Oleh karena itu, dibutuhkan dua endpoint berikut.

Producer memiliki tugas kritis untuk mengakuisisi data seismik dan menyediakannya ke dalam sistem. Gambaran detail dari tugas Producer dapat dilihat pada Gambar 4.30. Prosedur ini diinisiasi dengan penerimaan permintaan GET. Tahap awal proses ini melibatkan penentuan status operasional sistem, khususnya apakah sistem sedang beroperasi dalam mode LIVE. Jika sistem dalam mode LIVE, maka server SeedLink diaktifkan. Sebaliknya, jika sistem tidak dalam mode LIVE, maka inisialisasi FDSNWS akan dilakukan. Kedua mode ini ditujukan untuk mendapatkan akses ke data seismik, tetapi dari sumber yang berbeda sesuai dengan kebutuhan operasional.



Gambar 4.30: Alur Kerja Layanan Producer

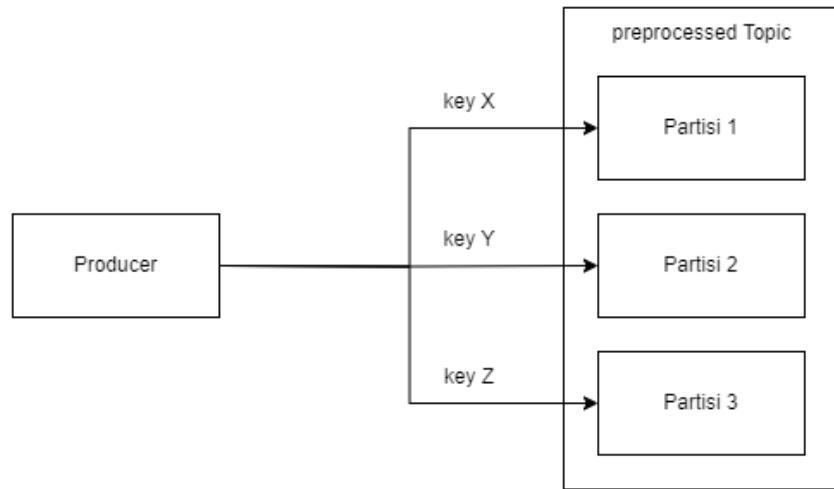
Sumber: (Olahan Penulis, 2023)

Setelah aktivasi server yang sesuai, Producer mengambil informasi mengenai stasiun yang aktif dari Redis. Ini merupakan langkah kunci dalam proses ini, mengingat Redis diimplementasikan sebagai *database* penyimpanan data dengan akses cepat, yang berfungsi untuk menyimpan konfigurasi sistem yang dapat diakses dengan efisiensi tinggi.

Dalam konteks FDSNWS, Producer mengeksekusi kueri ke GEOFON untuk mendapatkan akses data secara massal yang merupakan agregasi data dari stasiun-stasiun terpilih. Untuk SeedLink, yang lebih berorientasi pada data *real-time*, layanan mengolah data secara individual dari setiap stasiun.

Setelah data seismik berhasil diperoleh, langkah selanjutnya adalah mengkonversinya ke format JSON. Konversi ini esensial karena JSON adalah format

yang kompatibel secara luas dan memfasilitasi manipulasi data serta interoperabilitas dalam ekosistem EEWS.



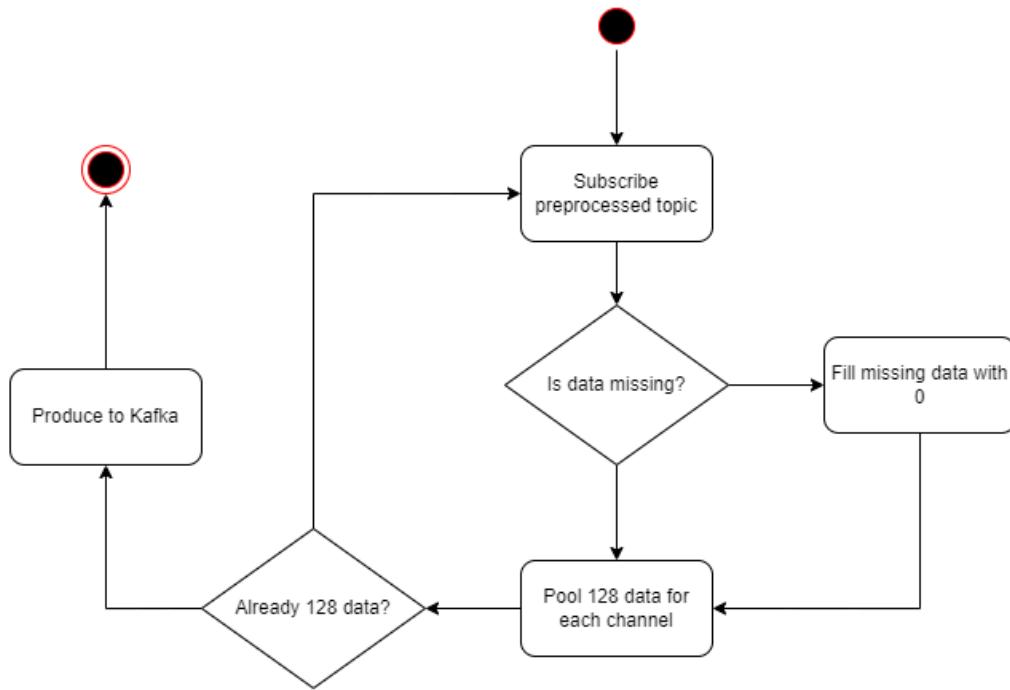
Gambar 4.31: Pendistribusian Data oleh Producer

Sumber: (Olahan Penulis, 2023)

Data dalam format JSON ini selanjutnya diproses dan dipublikasikan ke Kafka. Kafka bertindak sebagai pusat distribusi data, mengelola dan mengarahkan aliran data ke topik *preprocessed*. Seperti pada Gambar 4.31, Kafka mengorganisir data ini ke dalam partisi yang beragam untuk mendukung peningkatan *throughput* dan efisiensi. Partisi ini didistribusikan berdasarkan kunci unik yang ditetapkan, yang mempromosikan pembagian beban yang seimbang dan pemrosesan paralel, sehingga secara efektif mengurangi latensi sistem.

4.2.5 Implementasi Layanan Queue

Layanan ini bertugas mengelola data *streaming* yang diterima dari sumber seperti sensor atau sistem eksternal. Fungsi utama dari layanan ini adalah untuk menerima, menyimpan sementara, dan mengatur data dalam antrean sebelum diproses lebih lanjut. Implementasi layanan ini menggunakan Kafka, sebuah platform distribusi *streaming* data yang memungkinkan penyimpanan data dalam jumlah besar dengan latensi rendah.

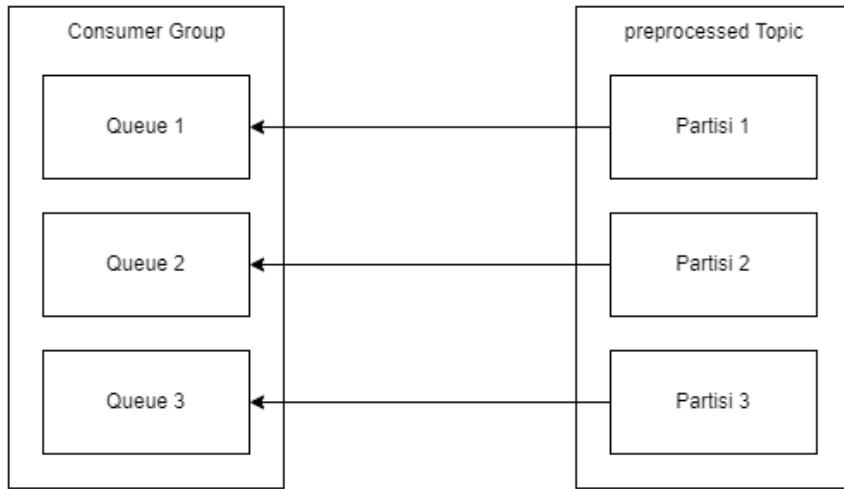


Gambar 4.32: Alur Kerja Layanan Queue

Sumber: (Olahan Penulis, 2023)

Seperti pada Gambar 4.32, proses dimulai ketika layanan melakukan *subscribe* ke topik *preprocessed*. Setelah berlangganan, layanan memeriksa keberadaan data. Jika terdapat data yang hilang, mekanisme otomatis akan mengisi kekosongan tersebut dengan nilai nol untuk menjaga integritas dan kesinambungan set data.

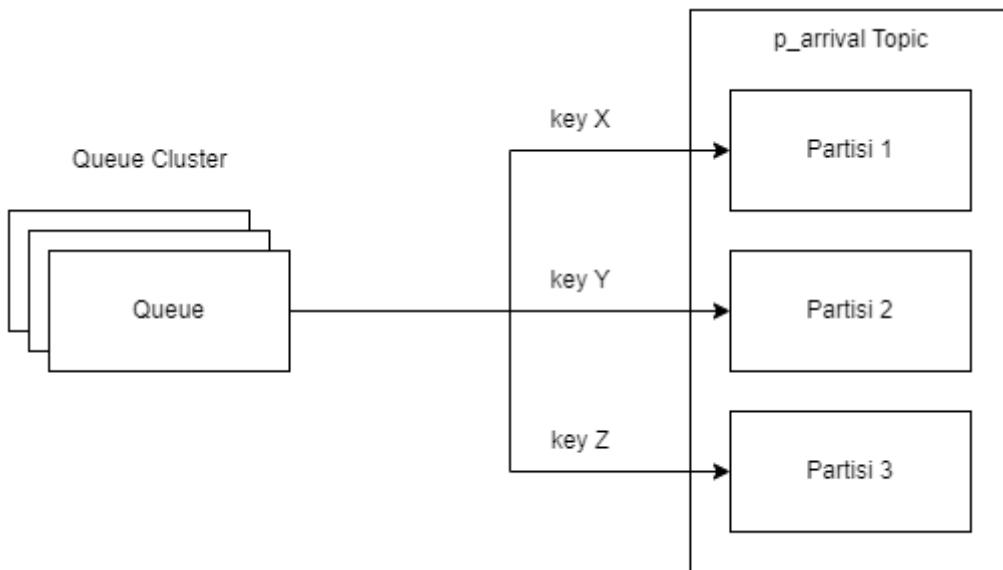
Selanjutnya, layanan ini akan mengumpulkan data sebanyak 128 poin data untuk setiap *channel*. Pengumpulan data ini dilakukan secara efisien untuk memastikan bahwa tidak ada penundaan atau latensi yang signifikan yang bisa mempengaruhi *throughput* sistem. Proses pengumpulan ini juga bertujuan untuk membentuk *batch* data yang siap untuk dikirimkan ke topik *p_arrival*.



Gambar 4.33: Proses Konsumsi Data oleh Queue

Sumber: (Olahan Penulis, 2023)

Untuk memaksimalkan performa dan *throughput*, Queue memanfaatkan konsep *consumer groups* dan partisi Kafka. Seperti ditunjukkan pada Gambar 4.33, setiap Queue dalam grup konsumen terhubung ke partisi yang sesuai pada topik *preprocessed*. Hal ini memungkinkan pembacaan dan pengolahan data secara paralel, mengurangi waktu tunggu, dan meningkatkan kapasitas pemrosesan data secara keseluruhan.



Gambar 4.34: Proses Distribusi Data oleh Queue

Sumber: (Olahan Penulis, 2023)

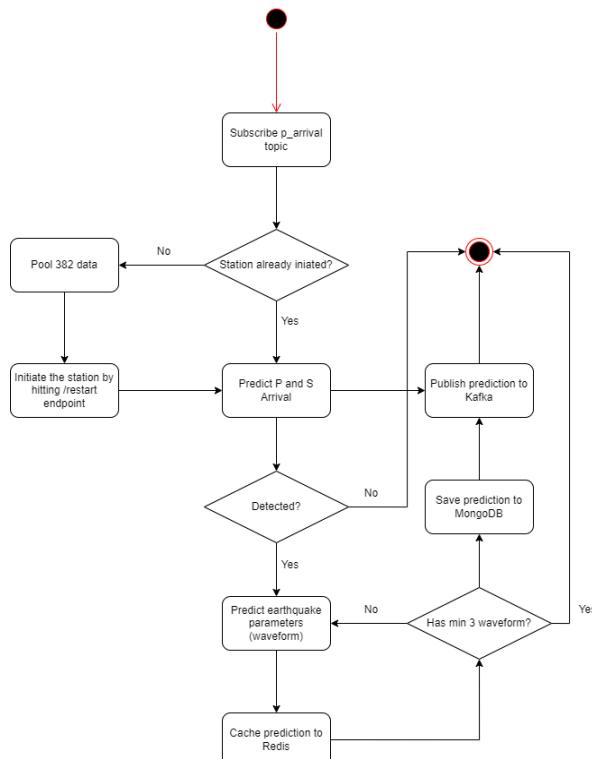
Di sisi publikasi data, seperti yang digambarkan pada Gambar 4.34, Queue mempublikasi data yang telah dikumpulkan ke topik p arrival. Setiap Queue dalam

cluster mempublikasikan data ke partisi tertentu di topik *p_arrival*, memastikan distribusi beban kerja yang merata dan meminimalkan kemungkinan *bottleneck*.

Integrasi antara konsumsi dan produksi data ini mengoptimalkan aliran data seismik melalui sistem, memastikan bahwa data yang diperlukan untuk pendekripsi gempa dapat diproses dan dianalisis dengan kecepatan tertinggi. Dengan demikian, EEWs dapat menjamin waktu respons yang cepat. Hal ini sangat kritikal untuk memberikan peringatan dini gempa bumi kepada pengguna.

4.2.6 Implementasi Layanan Picker

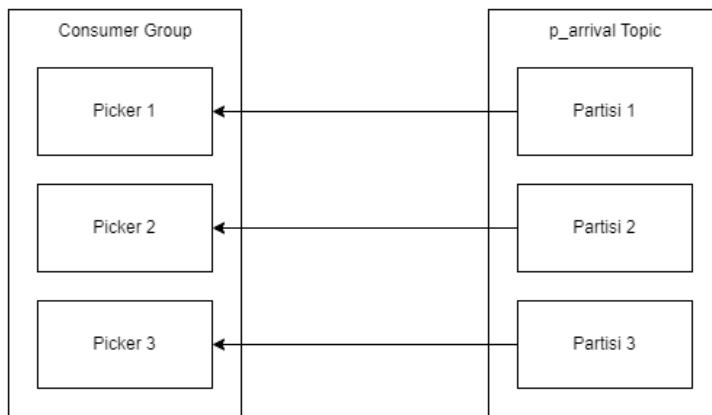
Picker dirancang untuk men-*subscribe* dan mengonsumsi data dari topik *p_arrival* di Kafka seperti yang ditunjukkan pada Gambar 4.35. Data yang diterima ini merupakan *batch data point* seismik yang telah disiapkan oleh layanan Queue. Penulis mengimplementasikan mekanisme pembacaan data yang memastikan bahwa tidak ada latensi yang signifikan antara penerimaan *batch* data dan proses analisis yang dilakukan oleh layanan ini.



Gambar 4.35: Alur Kerja Layanan Picker

Sumber: (Olahan Penulis, 2023)

Layanan Picker memanfaatkan kemampuan Kafka dalam *consumer groups* untuk paralelisasi pengolahan data. Dengan membagi data ke dalam beberapa partisi, Kafka memungkinkan Picker untuk secara simultan mengonsumsi dari berbagai partisi menggunakan *multiple instances*. Ini menjamin bahwa setiap *batch* data dapat diproses segera setelah tiba, tanpa menunggu batch sebelumnya selesai. Proses distribusi data pada layanan ini dapat dilihat pada Gambar 4.36.



Gambar 4.36: Proses Konsumsi Data oleh Picker

Sumber: (Olahan Penulis, 2023)

Setelah mendapatkan data dari hasil *subscription*, layanan memeriksa apakah stasiun telah diinisiasi. Jika belum, layanan akan menginisiasi stasiun dengan mengirim permintaan ke *endpoint /restart*. Ini merupakan langkah penting untuk memastikan bahwa stasiun siap mengirim data seismik yang akan diproses. Apabila stasiun telah diinisiasi, Picker melanjutkan dengan memprediksi waktu kedatangan gelombang P dan S, yang merupakan dua komponen utama dari gelombang seismik. Deteksi ini penting untuk mengidentifikasi keberadaan gempa.

Setelah prediksi kedatangan gelombang P dan S berhasil dilakukan, sistem kemudian memeriksa apakah gelombang tersebut terdeteksi. Jika ya, Picker melanjutkan dengan memprediksi parameter gempa, seperti magnitudo dan lokasi episenter, berdasarkan analisis waveform yang diperoleh. Prediksi yang telah dilakukan kemudian di-*cache* ke dalam Redis. Ini memungkinkan penyimpanan sementara hasil prediksi yang membutuhkan akses cepat dan efisien untuk proses selanjutnya.

Selanjutnya, sistem mengevaluasi apakah sudah terdapat minimal tiga *event* gempa. Jika belum, proses akan berlanjut kembali untuk mengumpulkan lebih banyak data. Jika sudah ada tiga atau lebih *event* gempa, hasil prediksi parameter gempa disimpan ke dalam MongoDB, *database* yang digunakan untuk penyimpanan jangka panjang.

Terakhir, prediksi yang telah dilakukan dipublikasikan kembali ke Kafka. Ini memungkinkan integrasi dengan komponen sistem lainnya, memastikan bahwa informasi yang relevan dapat disebarluaskan dan digunakan untuk langkah-langkah selanjutnya, termasuk pemberian peringatan dini kepada pengguna sistem. Proses ini berlangsung dalam siklus yang berkesinambungan, memungkinkan sistem EEWS untuk secara konstan memonitor dan merespons sinyal seismik untuk deteksi dini gempa bumi.

4.2.6 Implementasi Layanan REST API dan Websocket

Penulis mengembangkan REST API sebagai antarmuka utama untuk interaksi dengan sistem backend. API ini dirancang untuk memudahkan pengambilan data stasiun, melakukan *enable* dan *disable* stasiun. Menggunakan prinsip RESTful, penulis memastikan bahwa API ini bersifat stateless, dengan setiap permintaan HTTP berisi semua informasi yang diperlukan untuk diproses. Hal ini meningkatkan skalabilitas dan keandalan sistem. *API endpoints* disusun dengan logika yang jelas, memudahkan integrasi dengan aplikasi *frontend*. Tabel 4.9 menunjukkan daftar endpoint-endpoint yang ada pada REST API ini.

Tabel 4.9: Endpoint REST API

| Endpoint | Method | Deskripsi | Input | Contoh Respons |
|-------------|--------|--|-------------------------------------|----------------|
| /api/toggle | POST | Untuk melakukan enable dan disable stasiun | Body {station_code: <string>} | - |
| /api/live | GET | Untuk menginisiasi pengambilan data dengan mode LIVE | - | - |
| /api/stop | GET | Untuk menghentikan mode LIVE | - | - |

Tabel 4.9: Endpoint REST API (sambungan)

| Endpoint | Method | Deskripsi | Input | Contoh Respons |
|---------------|--------|--|-----------------------------------|--|
| /api/playback | GET | Untuk menginisiasi pengambilan data dengan mode PLAYBACK | Query Params start_time, end_time | - |
| /api/stations | GET | Untuk mendapatkan data stasiun dan channelnya | - | [{ "code": "TNTI", "lat": "0.7718", "long": "127.3667", "elevation": "43", "name": "GEOFON Station GEOFON Station Ternate, Indonesia", "channels": [{ "code": "BHN", "depth": 3.0, "azimuth": 0.0, "dip": 0.0, "sample_rate": 20.0 }, { "code": "BHE", "depth": 3.0, "azimuth": 90.0, "dip": 0.0, "sample_rate": 20.0 }, { "code": "BHZ", "depth": 3.0, "azimuth": 0.0, "dip": -90.0, "sample_rate": 20.0 }] }] |

WebSocket digunakan untuk mengirimkan data *real-time* dari *backend* ke *frontend*. Ini sangat penting dalam konteks sistem EEW karena kecepatan dan keaktualan informasi adalah kunci dari sistem ini. Penulis mengimplementasikan WebSocket server yang terintegrasi dengan sistem Kafka, memungkinkan transmisi data gempa secara langsung ke pengguna akhir segera setelah data diproses dan hasil prediksi tersedia. WebSocket server ini dirancang untuk menangani banyak koneksi secara simultan, memastikan distribusi informasi yang luas dan cepat.

Sistem *backend* akan mengirim kumpulan data yang didapat dari topik *p_arrival* dan topik *pick* secara periodik ke aplikasi web. Penulis memutuskan untuk mengirim kumpulan data secara periodik untuk membatasi pemanggilan *callback* karena *callback* akan mengubah *state* dan melakukan *render* terhadap komponen React. Tabel 4.10 menunjukkan bentuk data yang akan diterima oleh WebSocket.

Tabel 4.10: WebSocket Response

| Topik | Tipe | Keterangan | Respons |
|-----------|-------|--|--|
| p_arrival | start | Data untuk menandakan bahwa mode baru dimulai | {"topic": "p_arrival", "value": {"type": "start"}} |
| | trace | Data seismik beserta metadata sensor | {"topic": "p_arrival", "value": {"station": "BKB-940", "channel": "BHZ", "starttime": "2019-12-31T11:21:01.769538+00:00", "endtime": "2019-12-31T11:21:08.169538+00:00", "data": [872, 725, 669], "len": 3, "eews_producer_time": ["2023-12-08T11:34:00.111852", "2023-12-08T11:34:14.242107"], "eews_queue_time": ["2023-12-08T11:34:58.371948", "2023-12-08T11:34:58.373876"], "type": "trace"}} |
| | stop | Data untuk menandakan bahwa mode saat ini dihentikan | {"topic": "p_arrival", "value": {"type": "stop"}} |

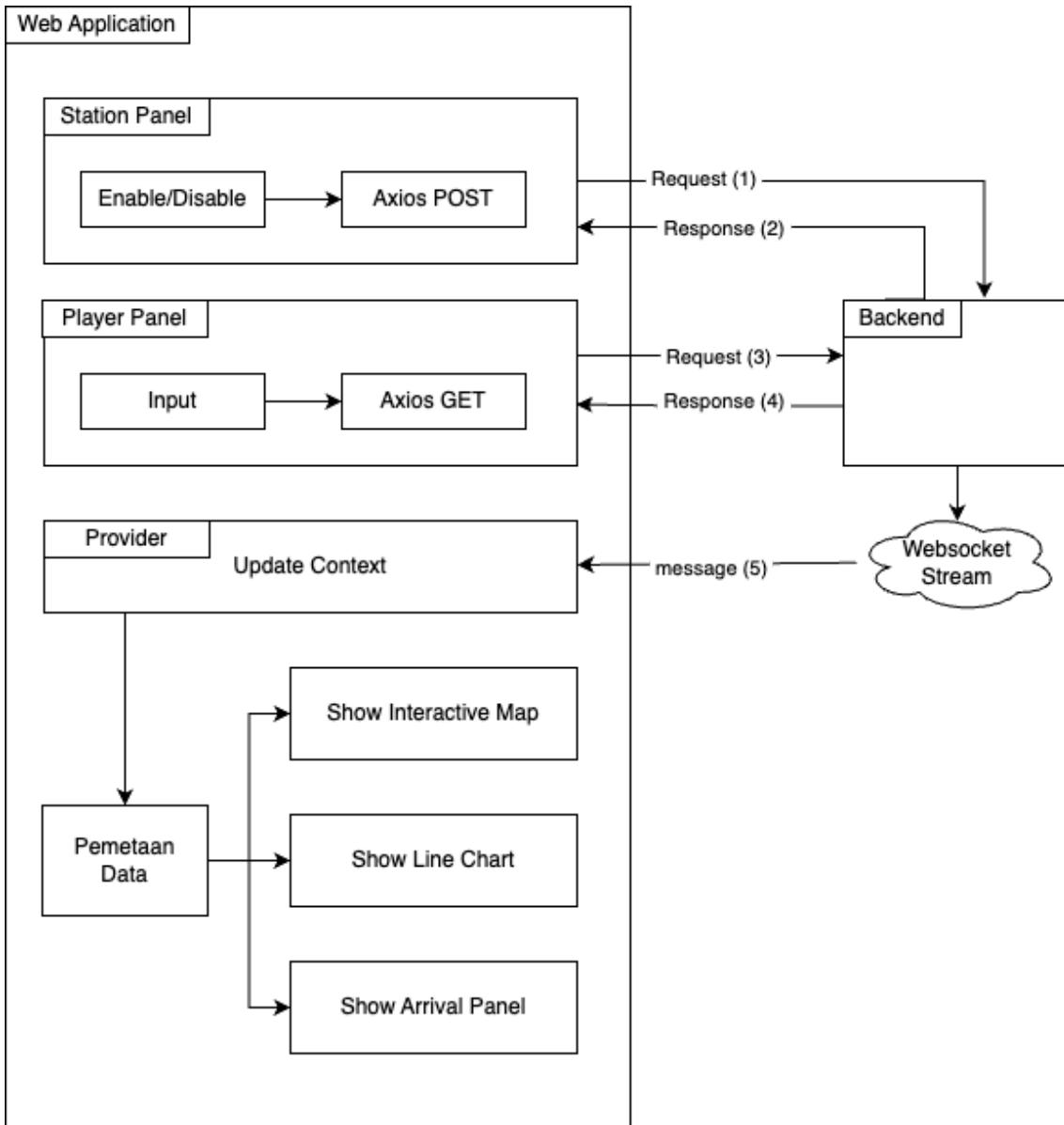
Tabel 4.10: WebSocket Response (sambungan)

| Topik | Tipe | Keterangan | Respons |
|-------|--------|--------------------------------------|--|
| pick | ps | Data hasil prediksi P atau S arrival | {"topic": "pick", value: {"station_code": "JAGI", "init_end": true, "p_arr": false, "p_arr_time": "2023-11-20 20:02:56.357314", "p_arr_id": "JAGI~0", "new_p_event": false, "s_arr": true, "s_arr_time": "2023-11-20 20:02:48.907314", "s_arr_id": "JAGI~2", "new_s_event": true, "type": "ps"}} |
| | params | Data hasil prediksi parameter gempa | {"topic": "pick", value: {"station_codes": ["TNTI", "TOLI", "GANI"], "magnitude": 5.966666666666666, "latitude": -0.23323952696581063, "longitude": 129.44385843930837, "depth": 0.0, "type": "params", "time": "2023-11-20 20:02:48.907314"}}} |

4.3 Implementasi Frontend

Pada sub bab ini, akan diuraikan tentang proses pengimplementasian sistem Earthquake Early Warning System (EEWS) pada sisi klien dan penyajian data melalui aplikasi berbasis web. Tujuan aplikasi web adalah sebagai GUI (Graphical User Interface) bagi pengguna untuk melakukan analisis prediksi p-wave dan parameter gempa berdasarkan p-wave menggunakan data trace. Secara umum, fungsi ini terdapat di salah satu modul SeiscomP, yaitu modul SCRTTV (Seiscomp Real-Time Trace View).

Dalam pengembangan aplikasi web, penulis menyederhanakan tampilan UI daripada SCRTTV agar lebih mudah digunakan dan dikembangkan, tetapi tetap menjaga fungsionalitas aplikasi. Implementasi aplikasi web akan membahas bagaimana memvisualisasikan data stasiun, waveform, p-wave, dan event gempa, dan bagaimana melakukannya dengan efisien. Setelah penjelasan mengenai implementasi, akan disertakan hasil evaluasi dari fungsionalitas aplikasi web. Alur kerja dari aplikasi web yang penulis buat dapat dilihat pada Gambar 4.37.



Gambar 4.37: Alur Teknis Pemrograman Aplikasi Web

Sumber: (Olahan Penulis, 2023)

4.3.1 Pemilihan Teknologi

Pemrograman frontend dilakukan menggunakan bahasa pemrograman Typescript dengan library React dan framework Next. Penulis menggunakan teknologi tersebut sebagai upaya pengembangan lebih lanjut dari penelitian sebelumnya yang menggunakan bahasa pemrograman Javascript dan library React.

Typescript bertujuan untuk membuat pengembangan JavaScript lebih scalable dan mudah dipelihara dengan berbagai fitur-fitur seperti interfaces, type aliases, abstract

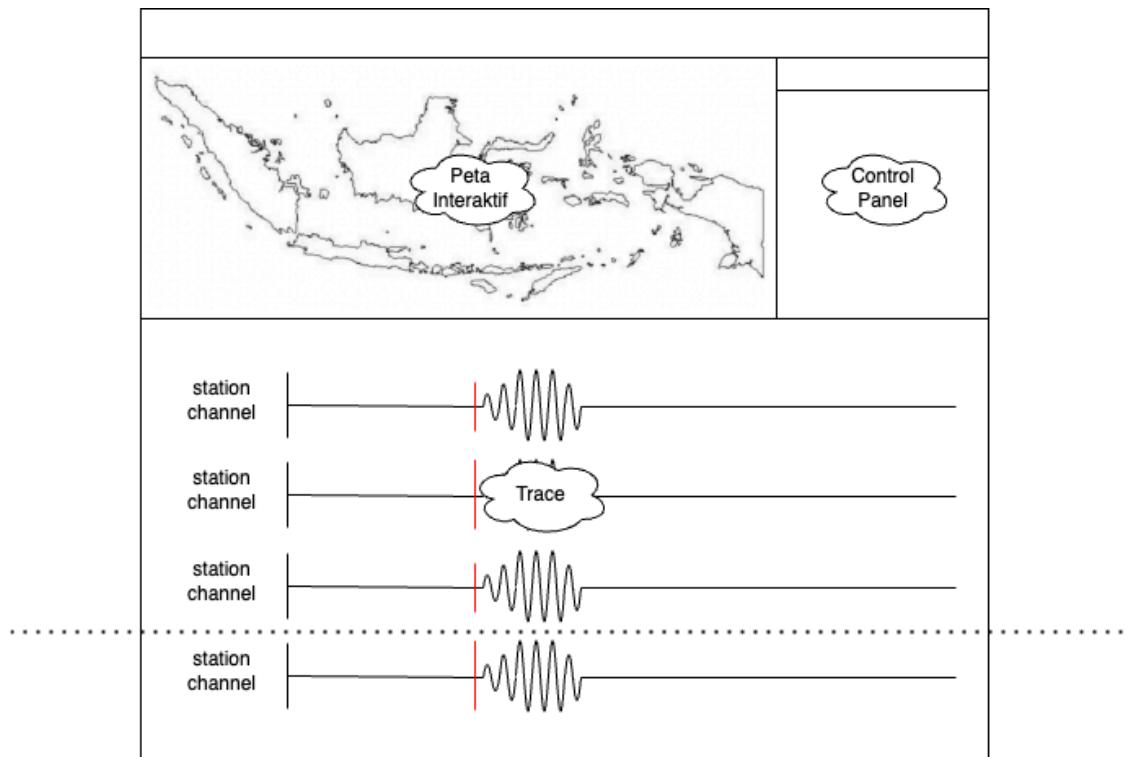
classes, function overloading, tuple, dan generic. Kode TypeScript akan diubah menjadi Javascript, memungkinkan penulis memanfaatkan standar Javascript sambil mendapatkan keuntungan dari static-typing. Penggunaan TypeScript mempermudah pengaturan kode, mendeteksi kesalahan lebih awal, dan meningkatkan dokumentasi kode, sehingga meningkatkan keterbacaan kode untuk penelitian lain di masa depan.

Library React digunakan karena memiliki dukungan komunitas yang besar sehingga terdapat banyak fitur dan library yang bisa penulis gunakan. Selain itu, pengembangan dengan React lebih cepat dibanding Javascript atau jQuery untuk membangun user interfaces, mengatur data, dan lebih efisien bagi pengembang. Untuk melakukan styling, penulis menggunakan library Tailwind CSS dengan alasan kenyamanan dan kecepatan pengembangan. Kemudian, framework Next bertujuan untuk memberi struktur tambahan bagi library React. Next melakukan abstraksi dan mengkonfigurasi berbagai tools yang diperlukan oleh React, seperti bundling, kompilasi, dan test.

4.3.2 Arsitektur Aplikasi Web

Tampilan halaman web memiliki tujuan agar pengguna bisa mengawasi stasiun gempa dan visualisasi trace, p-wave, dan origin secara near real-time. Data yang diterima didapatkan dari producer service yang melakukan sambungan dengan server Seedlink atau server FDSN milik GEONET. Terdapat tiga komponen utama dalam halaman web, yaitu peta interaktif, control panel, dan trace.

Peta interaktif merupakan bagian di mana pengguna bisa melihat posisi stasiun seismograf di peta, melihat informasi mengenai stasiun, melihat posisi gempa apabila terdeteksi, dan melihat informasi mengenai gempa itu. Untuk menghemat layar, informasi mengenai stasiun dan parameter gempa akan muncul apabila pengguna mengklik stasiun atau gempa layaknya tooltip. Penulis menetapkan posisi *default* peta berada di tengah Indonesia (2,28 LS dan 117,59 BT) dan skala *default* 4891.968 meters/pixel agar keseluruhan wilayah Indonesia bisa tampak. Peta tersebut dapat digeser untuk mengubah posisi, dan bisa untuk diperbesar untuk mengubah skala yang dipakai. Tampilan umum dari aplikasi dapat dilihat pada Gambar 4.38.



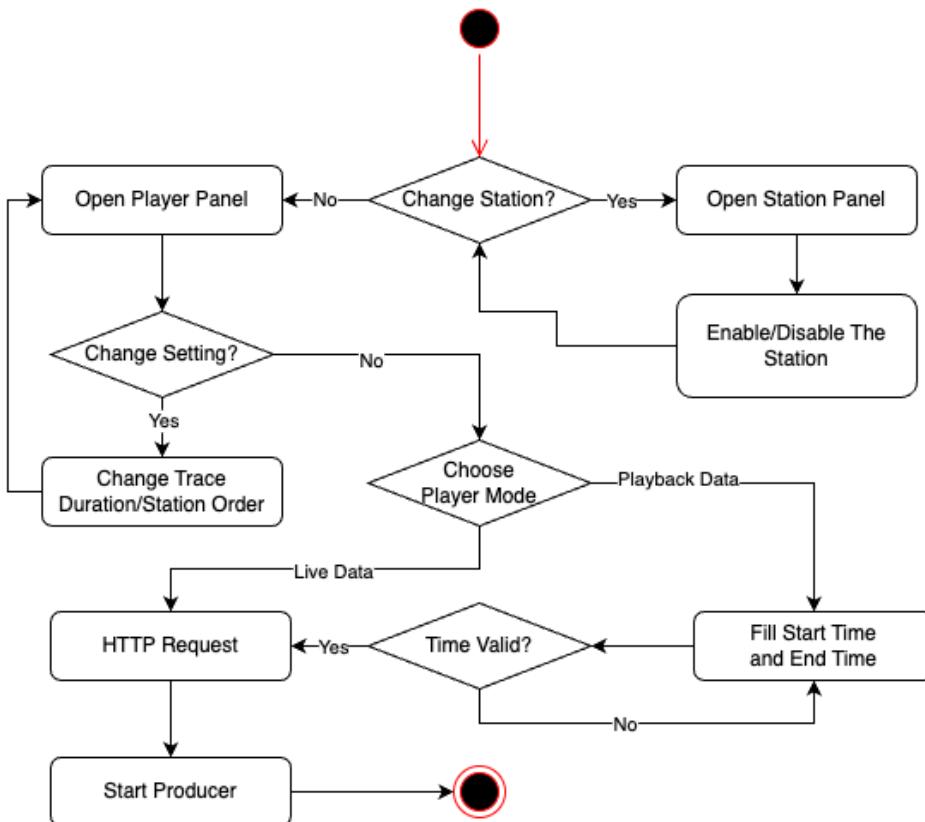
Gambar 4.38: Tampilan Low Fidelity Aplikasi Web

Sumber: (Olahan Penulis, 2023)

Control Panel terbagi menjadi beberapa panel, yaitu player, station, arrival, dan statistics. Panel player memiliki fungsi untuk mengubah setting dari komponen trace seperti panjang durasi trace dan urutan stasiun untuk ditunjukkan, lalu untuk membuat request data trace ke backend baik data *real-time* atau data playback. Kemudian, panel station memiliki fungsi untuk memantau status setiap channel dan stasiun, serta mengubah status stasiun. Terdapat tiga status stasiun, yaitu disabled untuk stasiun yang di-exclude dari data trace, enabled untuk stasiun yang diminta di data trace, dan active untuk stasiun yang enabled dan data tracennya sudah masuk dalam sistem. Selain itu, terdapat fitur navigasi apabila pengguna menekan nama stasiun, maka layar pengguna akan dibawa ke *line chart* yang sesuai. Lalu, panel arrival berfungsi untuk menunjukkan hasil p-wave dengan menunjukkan stasiun, channel, waktu pick, dan waktu terdeteksi dalam bentuk tabel. Terdapat pula informasi mengenai parameter gempa: waktu origin, waktu terdeteksi, lintang, bujur, dan magnitude. Terakhir adalah panel statistics yang berisi informasi mengenai paket trace yang diterima setiap channel. Informasi yang tersedia adalah waktu paket diterima klien, waktu asli, waktu di servis producer, dan

waktu di servis queue. Informasi ini berguna untuk mengetahui latensi data dalam sistem backend.

Komponen trace merupakan kumpulan *line chart* yang memvisualisasikan data waveform setiap channel yang ada. Tampilan data waveform didasarkan pada modul SCRTV SeisComP yang juga menggunakan *line chart* dengan sumbu-y untuk amplitudo dan sumbu-x untuk waktu. Selain itu, hasil p-wave yang terdeteksi divisualisasikan dengan garis vertikal berwarna merah pada waktu p-wave datang (arrival). Di sisi kiri *line chart* terdapat nama stasiun dan nama channel serta statistik data waveform seperti nilai amplitudo/absolute max, nilai mean, dan waktu arrival. Durasi waktu yang ditunjukkan di *line chart* adalah 5 menit, tetapi dapat diubah melalui *player panel*. Terakhir, untuk menghemat layar, stasiun dan channel yang tidak atau belum menerima data *waveform* tidak akan ditunjukkan di layar.



Gambar 4.39: Ilustrasi *User Flow* Aplikasi Web

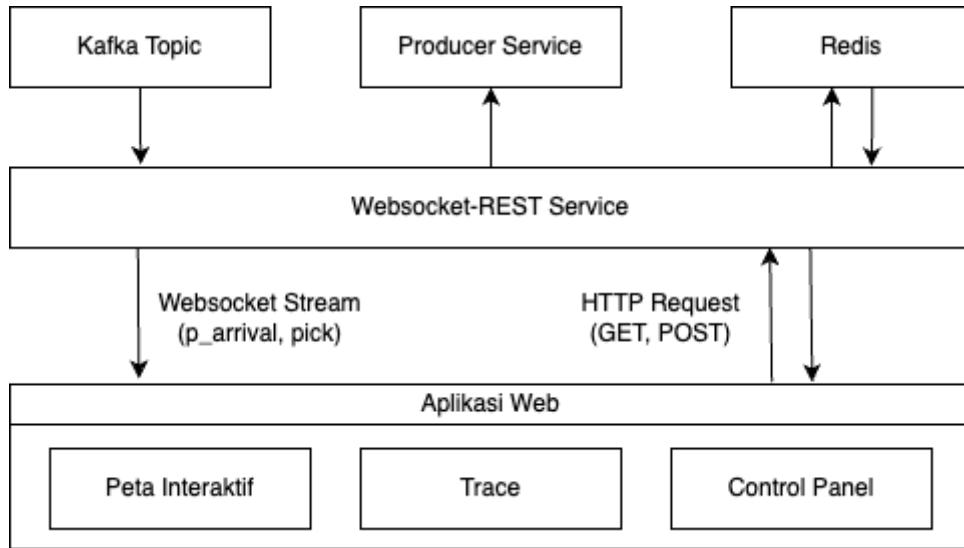
Sumber: (Olahan Penulis, 2023)

Gambar 4.39 menjelaskan mengenai alur *user flow*. Aplikasi web dibuat berdasarkan kebutuhan untuk memantau kejadian gempa dari puluhan stasiun di network GEOFON saat ini, dan berpotensi bisa sampai ratusan stasiun apabila dibutuhkan. Untuk itu, fitur memilih stasiun yang akan digunakan merupakan elemen yang penting. Setelah membuka halaman web, pengguna dapat menentukan stasiun yang ingin digunakan di panel stasiun. Kemudian, menentukan pengaturan tampilan komponen trace seperti durasi data yang ditampilkan di *line chart* dan urutan stasiun ditampilkan. Kedua langkah tersebut sepenuhnya opsional. Lalu, pengguna memilih mode yang ingin digunakan, yaitu *playback* yang memutar data terdahulu dari server FDSN GEOFON atau *live* yang memutar data *real-time* dari server SEEDLINK GEOFON. Apabila *playback*, pengguna perlu memasukan waktu mulai dan waktu selesai. Terdapat *validity check* untuk memastikan waktu selesai setelah waktu mulai, dan selisih waktu tidak lebih dari 15 menit. Penetapan selisih 15 menit didasarkan dari pengamatan penulis bahwa data waveform memakan waktu lama sehingga servis produser terjadi timeout.

4.3.3 Komunikasi dengan Backend

Aplikasi web mengirim dan menerima data dari sistem backend menggunakan HTTP request dan websocket. Aplikasi web menerima data stasiun, melakukan perubahan data stasiun, dan meminta data trace melalui HTTP request, sementara itu menerima data trace dilakukan melalui websocket. Rincian mengenai endpoint HTTP dan websocket terdapat pada tabel 4.9 dan 4.10

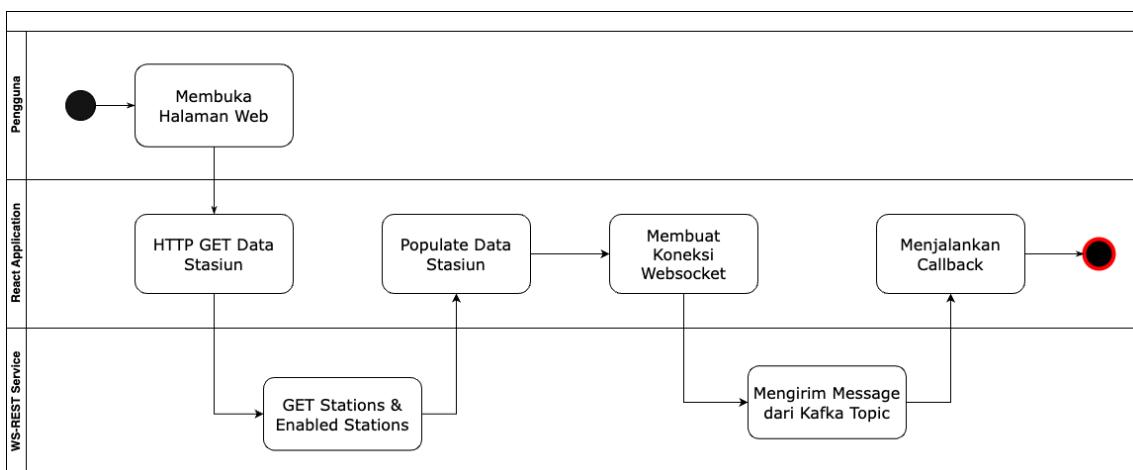
Untuk melakukan HTTP request, penulis menggunakan library Axios untuk membuat singleton yang akan digunakan di semua request. Keuntungan singleton ini adalah membuat kode lebih sederhana dan efisien. Kemudian, untuk websocket penulis tidak memanfaatkan library, melainkan menggunakan Websocket API. Pertimbangan penulis adalah use-case yang dipakai di aplikasi cukup sederhana, yaitu membuat satu koneksi dan menerima *messages* dari backend dari koneksi itu. Komunikasi dengan backend diilustrasikan oleh gambar 4.40.



Gambar 4.40: Ilustrasi Komunikasi Data Aplikasi Web

Sumber: (Olahan Penulis, 2023)

Pengambilan data melalui websocket dimulai dari pengguna mengunjungi halaman web. Aplikasi akan melakukan request data stasiun melalui HTTP. Setelah data stasiun didapatkan, akan dibuat koneksi websocket dengan backend untuk menerima data trace, p-wave, dan parameter gempa. Websocket akan menerima data secara periodik dan menjalankan *callback* untuk mengupdate global state di komponen context provider.



Gambar 4.41: Ilustrasi Komunikasi Data Websocket Aplikasi Web

Sumber: (Olahan Penulis, 2023)

4.3.4 Manajemen State atau Data

Manajemen state adalah proses menjaga informasi mengenai input aplikasi dari berbagai *data flow* untuk memahami state aplikasi pada suatu waktu. Terdapat beberapa teknik *state management* dalam React seperti React Context dan Redux. Akan tetapi, penulis memutuskan tidak menggunakan redux karena scope aplikasi belum membutuhkannya, menambah dependensi, dan React Context sudah memberikan fungsionalitas yang sama tanpa boilerplate. Dengan menggunakan Context, penulis dapat mengurangi kompleksitas dalam pengiriman data antar komponen dan menghilangkan prop tambahan. Hal ini dapat membuat kode lebih ringkas dan lebih mudah untuk dikelola.

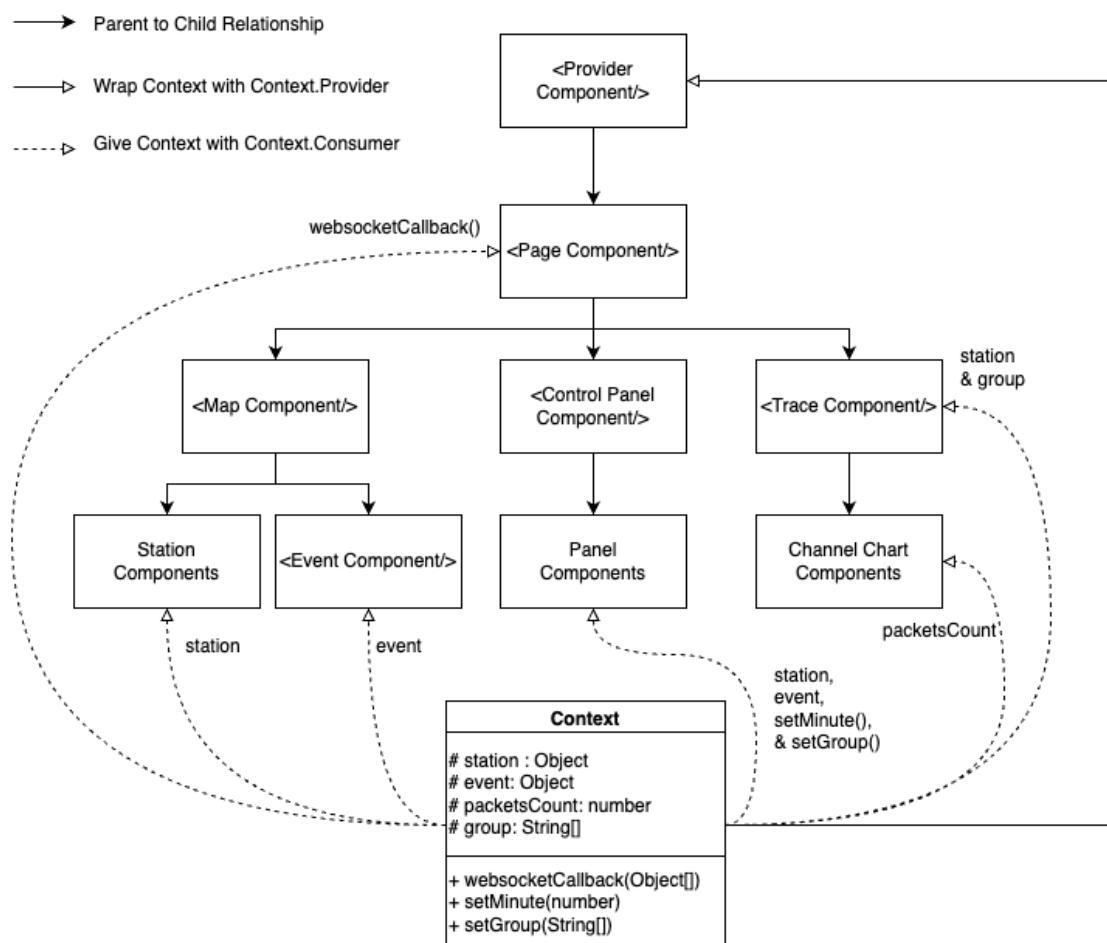
Secara umum, terdapat dua konsep utama dalam React Context API, yaitu Context Provider, dan Context Consumer. Context Provider berfungsi sebagai pembungkus yang menyediakan data atau state global untuk seluruh child component. Data ini dapat diakses oleh komponen-komponen di dalam subtree tersebut tanpa perlu melewatkkan prop secara manual. Sementara itu, Context Consumer digunakan untuk mengambil data konteks tersebut di dalam komponen.

Tabel 4.11: Penjelasan State Global

| Properti | Tipe | Keterangan |
|-------------------|------------------------------|---|
| stations | Object | Data stasiun (channel, waveform, p-wave) |
| event | Object Null | Data parameter gempa |
| packetsCount | number | Jumlah paket dalam <i>line chart</i> |
| group | String[] | Urutan stasiun dalam komponen trace |
| websocketCallback | (messages: Object[]) => void | Mengatur data waveform, p-wave, dan parameter gempa |
| setMinutes | (min: number) => void | Mengatur durasi waktu dalam <i>line chart</i> |
| setGroup | (group: String[]) => void | Mengatur urutan stasiun |

Tabel 4.11 merupakan penjelasan mengenai data state yang ada pada komponen provider. Komponen provider berada hirarki paling atas untuk menyediakan data yang

digunakan komponen lainnya. Page Component merupakan tempat koneksi websocket diciptakan dan disimpan, sehingga membutuhkan `websocketCallback()` yang dijalankan setiap menerima *message* baru. Kemudian, Trace Component merender hasil waveform dan p-wave setiap stasiun dan membutuhkan data stations dan group. Sementara itu, anak komponennya akan merender satu channel dan membutuhkan data packetsCount untuk mengatur durasi trace yang ditunjukan. Panel components merupakan tempat mengatur durasi trace, urutan stasiun, dan menunjukan data stasiun serta parameter gempa, sehingga menggunakan data station, event, `setMinute()`, dan `setGroup()`. Station Components akan merender posisi stasiun di peta sehingga membutuhkan data station. Terakhir, Event Component akan merender posisi origin apabila terjadi gempa, sehingga membutuhkan data event.



Gambar 4.42: Ilustrasi Manajemen State pada *React Component Tree*

Sumber: (Olahan Penulis, 2023)

4.3.5 Visualisasi Data

Pada aplikasi web, penulis menggunakan dua library untuk membuat user visualisasi data, library MapBox untuk membuat komponen peta interaktif yang menunjukkan posisi stasiun seismograf dan origin gempa, dan library recharts.js untuk membuat komponen trace yang memvisualisasikan data waveform dan p-wave di suatu channel. Data yang ditampilkan berasal dari hook useEEWS(). Gambar 4.43 memperlihatkan peta interaktif yang digunakan.



Gambar 4.43: Peta Interaktif Tanpa Data

Sumber: (Olahan Penulis, 2023)

Ketika pengguna mengunjungi halaman, peta akan menunjukkan wilayah Indonesia dengan pin yang melambangkan stasiun hasil dari HTTP request. Pengguna dapat merubah posisi map dengan mendrag peta. Kemudian, untuk melakukan zoom-in atau zoom-out pengguna dapat menekan tombol di sisi kanan atas. Pin stasiun memiliki tiga warna, masing masing untuk setiap status stasiun, yaitu disabled, enabled, dan active. Status disabled diberikan pada stasiun yang tersedia di stasiun tapi tidak ingin direquest datanya. Status enabled adalah stasiun yang datanya diinginkan. Status active adalah stasiun yang enabled dan data waveform sudah diterima oleh aplikasi web. Pengguna dapat melihat informasi terkait stasiun tersebut dengan menekan pin. Popup akan muncul dengan informasi status, kode stasiun, nama stasiun, latitude, longitude, ketinggian, dan channel yang tersedia.

Apabila sistem mendeteksi bahwa terjadi gempa, peta akan memunculkan lingkaran merah pada titik origin gempa. Pengguna dapat melihat informasi terkait

gempa tersebut dengan menekan lingkaran tersebut. Popup akan muncul dengan informasi tanggal dan waktu origin (UTC+07:00), magnitudo (skala richter), latitude, dan longitude.



Gambar 4.44: Line Chart

Sumber: (Olahan Penulis, 2023)

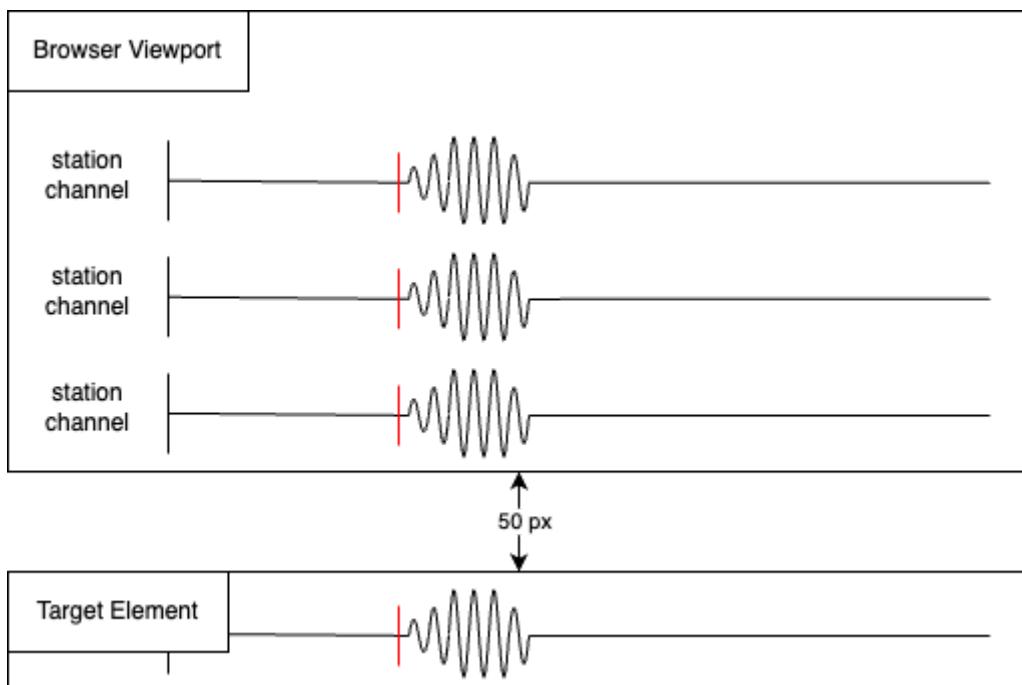
Waveform seluruh stasiun dan channel akan ditunjukkan pada Trace Component yang didapat dari konteks melalui `useEWS()`. Komponen ini akan mengurutkan stasiun berdasarkan prioritas sesuai dari urutan kode pada nilai group. Apabila tidak disebutkan pada group, maka akan menggunakan urutan abjad dari kode stasiun. Pada line chart sendiri, terdapat informasi mengenai stasiun dan channel di sisi paling kiri, nilai absolute maximum/amplitudo, rata-rata/mean, dan waktu arrival dari p-wave. Gambar 4.44 menunjukkan *line chart* stasiun BNDI di channel BHZ memiliki *p-arrival* pada 13.02.27.

Penggambaran line chart dilakukan menggunakan komponen `LineChart` dari library `rechart`. Sumbu-y melambangkan nilai poin waveform dengan nilai 0 di tengah. Kemudian, sumbu-y melambangkan waktu dalam UTC+07:00 dan terdapat 10 label waktu selama panjang sumbu. Lama data yang ditunjukkan bergantung pada nilai `packetsCount`.

Library `rechart` akan melakukan render ulang pada line chart setiap kali data stations diperbarui. Hal ini berarti satu channel akan diperbarui setiap 120 data points, atau 6 detik sekali bila menganggap frekuensi stasiun 20 Hz. Untuk jumlah stasiun yang rendah hal ini tidak bermasalah, tetapi apabila sudah mencapai angka yang tinggi hal ini akan berakibat buruk pada performa aplikasi karena merupakan task yang memakan memori. Untuk mengatasinya, penulis perlu mencari mekanisme yang bisa meningkatkan memori atau mencegah render yang tidak diperlukan.

Penulis memutuskan untuk menggunakan Intersection Observer API. Ini merupakan API JavaScript yang memberikan cara efisien dan performan untuk memonitor kapan suatu elemen HTML masuk atau keluar dari viewport (bagian yang

terlihat oleh pengguna pada halaman web). API ini berguna untuk mengimplementasikan event-based function terkait dengan elemen yang muncul atau menghilang dari layar, yang dapat berguna dalam skenario seperti pengunduhan data saat elemen muncul atau memicu animasi pada saat elemen masuk tampilan. Ketika *line chart* berada dalam 100px dari viewport, maka nilai intersection menjadi true dan baru akan merender *line chart*. Bila *line chart* di luar itu, maka nilai intersection menjadi false dan akan merender div kosong.



Gambar 4.45: Ilustrasi *Intersection Observer*

Sumber: (Olahan Penulis. 2023)

4.3.6 Hasil Implementasi Aplikasi Web

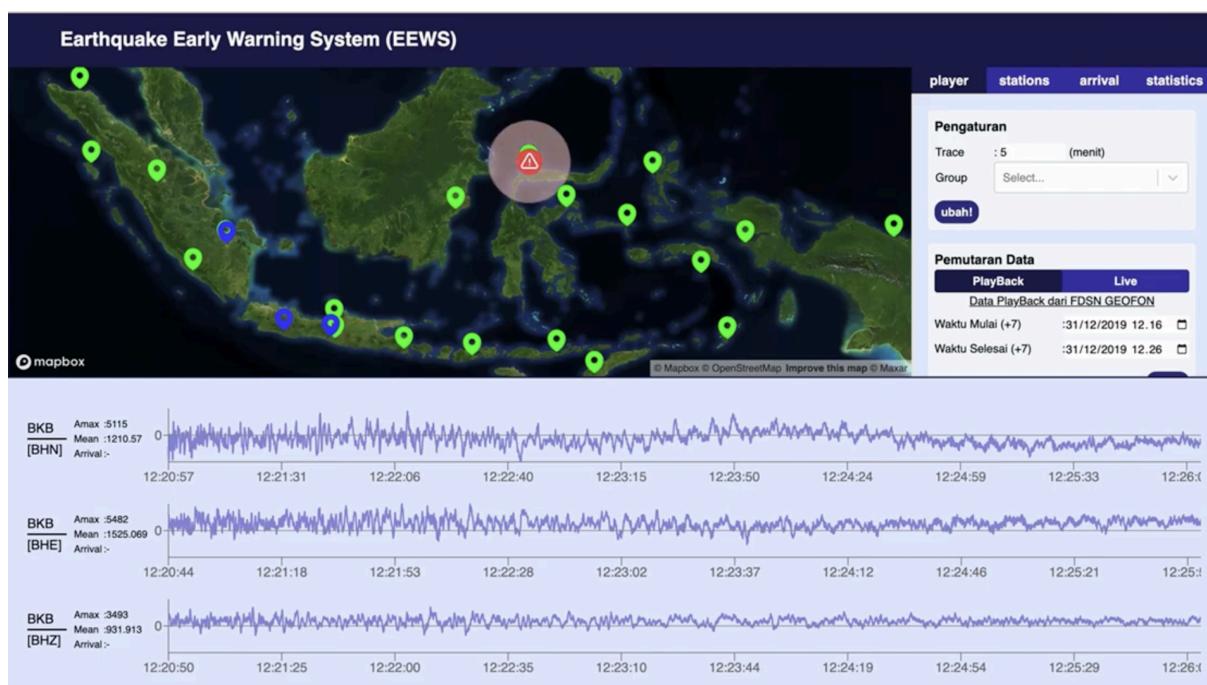
Ketika pengguna membuka halaman web dan aplikasi berjalan, pengguna akan melihat tampilan seperti di gambar 4.46. Saat ini, aplikasi hanya mengambil informasi stasiun seismograf. Setelah pengguna mengambil data, warna pin stasiun di peta akan mulai berubah dan line chart akan muncul di bagian trace. Line chart hanya akan ditunjukkan pada channel yang memiliki data saja. Data yang divisualisasikan pada gambar 4.47 adalah data playback pada tanggal 31 Desember 2019 12.16 UTC+07:00, sampai 31

Desember 2019 12.26 UTC+07:00. Data tersebut diambil menggunakan mode playback dari server FDSN Web Service milik GEOFON.



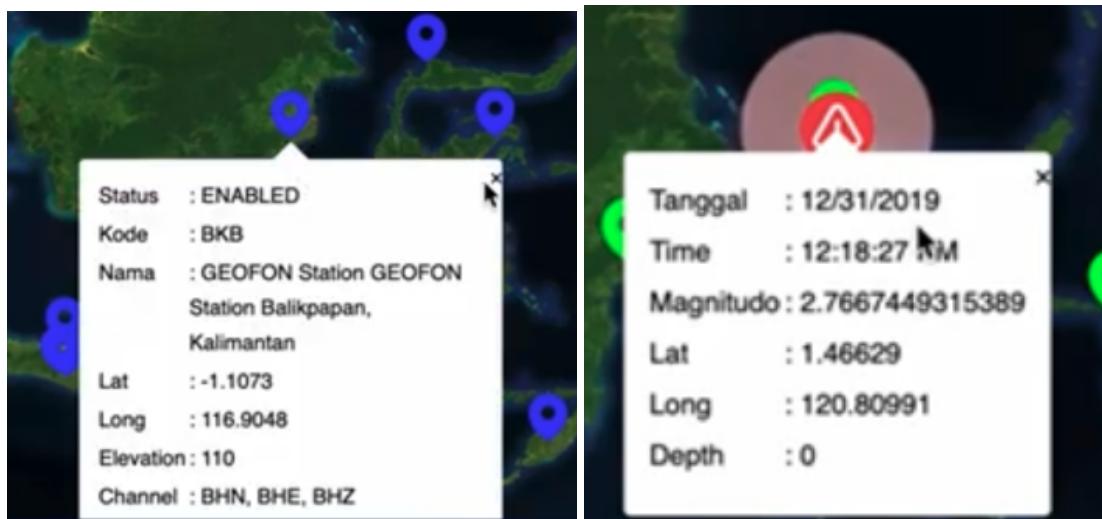
Gambar 4.46: Halaman Web Tanpa Data Waveform

Sumber: (Olahan Penulis., 2023)



Gambar 4.47: Halaman Web dengan Data Waveform

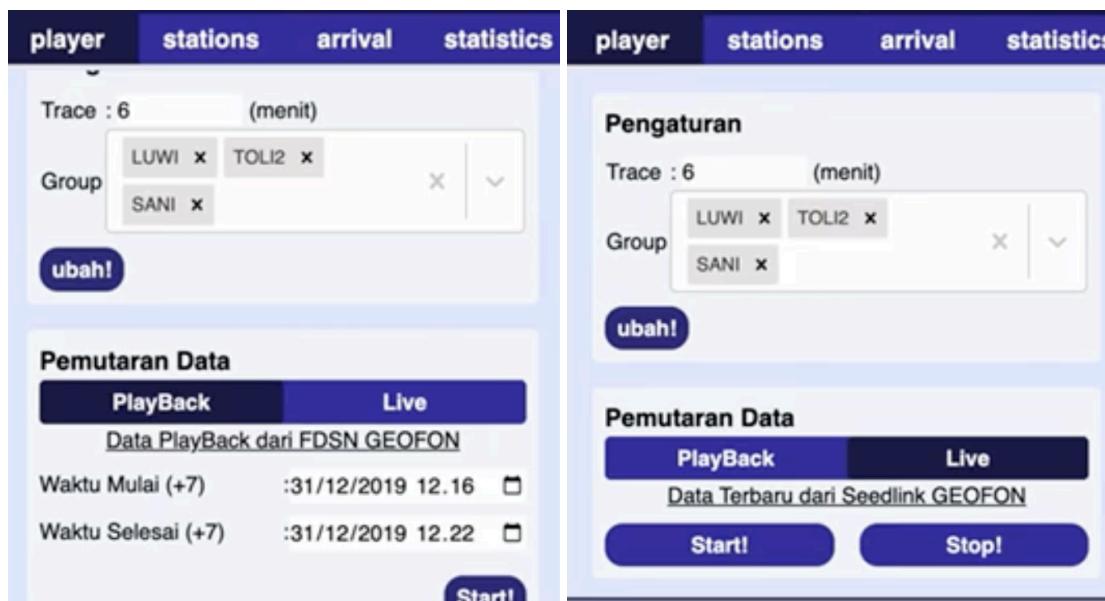
Sumber: (Olahan Penulis., 2023)



Gambar 4.48: *Popup* Informasi Stasiun (kiri) dan Parameter Gempa (kanan)

Sumber: (Olahan Penulis., 2023)

Peta interaktif memiliki fitur *popup* yang akan memunculkan informasi mengenai stasiun seismograf, atau parameter gempa apabila menekan simbol. Gambar 4.48 menunjukkan *popup* stasiun dengan kode BKB dan *popup* parameter gempa yang terjadi di Sulawesi pada 31 Desember 2019, 12.18.17 WIB (UTC+07:00).



Gambar 4.49: Panel *Player Playback* (kiri) dan Panel *Player Live* (kanan)

Sumber: (Olahan Penulis., 2023)

Control Panel memiliki posisi di kanan atas halaman web. Control panel terdiri atas panel player, panel stations, panel arrival, dan panel statistics. Gambar 4.49 merupakan tampilan panel player. Pada bagian atas terdapat input nilai angka untuk panjang trace yang ditunjukan di *line chart* dan input *multi-value* yang merupakan dropdown di mana pengguna bisa mengatur urutan stasiun muncul di layar. Stasiun yang dipilih akan ditunjukan sesuai urutannya, dan yang tidak dipilih akan ditunjukan sesuai urutan abjad kode stasiun. Kemudian, pengguna perlu menekan tombol ubah agar state tersimpan.

Kemudian, pada bagian bawah adalah mode pemutaran data. Terdapat dua mode, yaitu playback dan live. Pada playback, pengguna perlu memasukan waktu mulai dan waktu selesai dalam WIB (UTC+07:00). Terdapat mekanisme pengecekan agar waktu selesai tidak lebih kecil daripada waktu mulai. Pada mode live, pengguna cukup menekan tombol start untuk mulai menerima data *real-time*. Untuk menghentikannya, dapat menekan stop, atau dengan menjalankan playback.

The image shows two side-by-side panels from a software application. Both panels have a header with four tabs: 'player', 'stations', 'arrival', and 'statistics'. The 'arrival' tab is selected.

Panel Arrival Pick (Left):

| Station | Channels | Pick | Detected |
|---------|----------|-------------------------|------------------------|
| SANI | BHN | 12/31/2019, 12:18:48 PM | 12/7/2023, 12:46:14 AM |
| SANI | BHE | 12/31/2019, 12:18:48 PM | 12/7/2023, 12:46:14 AM |
| SANI | BHZ | 12/31/2019, 12:18:48 PM | 12/7/2023, 12:46:14 AM |
| TOLI2 | BHN | 12/31/2019, 12:18:47 PM | 12/7/2023, 12:46:44 AM |
| TOLI2 | BHE | 12/31/2019, 12:18:47 PM | 12/7/2023, 12:46:44 AM |
| TOLI2 | BHZ | 12/31/2019, 12:18:47 PM | 12/7/2023, 12:46:44 AM |

Panel Arrival Event (Right):

| Station | Channels | Pick | Detected |
|---------|----------|-------------------------|------------------------|
| LUWI | BHZ | 12/31/2019, 12:19:07 PM | 12/7/2023, 12:46:33 AM |

Event (Earthquake) Data:

- Tanggal : 12/31/2019
- Time : 12:18:27 PM
- Magnitudo : 2.7667449315389
- Lat : 1.466629
- Long : 120.80991
- Depth : 0
- Detected At : 12/7/2023, 12:47:01 AM

Gambar 4.50: Panel *Arrival Pick* (kiri) dan Panel *Arrival Event* (kanan)

Sumber: (Olahan Penulis., 2023)

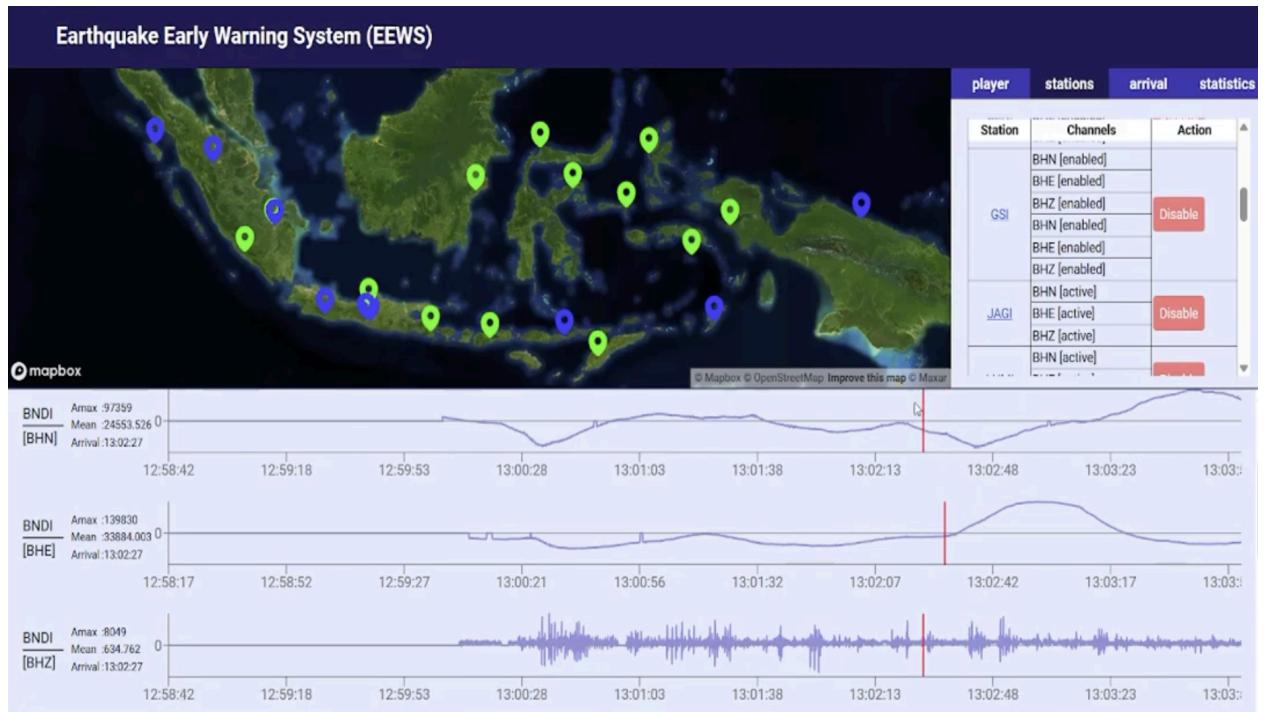
Gambar 4.50 merupakan panel arrival. Panel ini berfungsi menunjukkan informasi mengenai hasil prediksi p-wave dalam bentuk tabel, seperti stasiun, channel, waktu p-wave, dan waktu p-wave terdeteksi oleh sistem. Kemudian, di bagian bawah terdapat rincian hasil prediksi parameter gempa bumi. Informasi yang ditampilkan sama dengan yang ada pada popup origin gempa di gambar 4.48.

| player | stations | arrival | statistics | player | stations | arrival | statistics | | |
|----------------------|----------------|--------------------------|---------------|----------------------|--------------------|--------------|------------|--------------|--------------|
| | Station | Channels | Action | | Last Packet | | | | |
| GSI | BHE [enabled] | <button>Disable</button> | | BKB | BHN | 00:45:28.475 | Waveform | 12:21:55.669 | 12:22:01.119 |
| | BHZ [enabled] | <button>Disable</button> | | | | | | Producer | 00:15:41.955 |
| JAGI | BHN [enabled] | <button>Disable</button> | | | | | | Queue | 00:15:41.955 |
| | BHE [enabled] | <button>Disable</button> | | BHE | 00:45:28.475 | | Waveform | 12:22:04.369 | 12:22:07.519 |
| | BHZ [enabled] | <button>Disable</button> | | | | | | Producer | 00:15:41.955 |
| LHMI | BHN [disabled] | <button>Enable</button> | | | | | | Queue | 00:15:41.955 |
| | BHE [disabled] | <button>Enable</button> | | BHZ | 00:45:28.475 | | Waveform | 12:22:02.119 | 12:22:07.519 |
| | BHZ [disabled] | <button>Enable</button> | | | | | | Producer | 00:15:41.955 |
| LUWI | BHN [enabled] | <button>Disable</button> | | | | | | Queue | 00:15:41.955 |
| | BHE [enabled] | <button>Disable</button> | | BKNI | 00:45:28.475 | | Waveform | 12:22:02.819 | 12:22:07.469 |
| | BHZ [enabled] | <button>Disable</button> | | | | | | Producer | 00:15:41.955 |
| BKB | BHN [enabled] | <button>Disable</button> | | | | | | Queue | 00:15:41.955 |

Gambar 4.51: Panel *Stations* (kiri) dan Panel *Statistics* (kanan)

Sumber: (Olahan Penulis., 2023)

Gambar 4.51 merupakan panel stations dan panel statistics. Panel stations berfungsi memberi tahu status setiap stasiun dan channel di dalamnya. Status ini sebagaimana yang dirinci pada sub-bab 4.3.5 Visualisasi Data. Kemudian, terdapat tombol action untuk mengubah status stasiun menjadi aktif atau non-aktif. Panel statistics digunakan untuk mengetahui performa sistem backend. Terdapat informasi kapan paket data waveform masuk dan meninggalkan produser service, masuk dan meninggalkan queue service, dan kapan diterima oleh aplikasi web.



Gambar 4.52: *Line Chart* dengan Prediksi *P-Wave*

Sumber: (Olahan Penulis., 2023)

Pengguna dapat melihat visualisasi data waveform dari komponen *line chart* di bagian bawah halaman web. Di bagian ini, ditunjukkan trace berdasarkan urutan grup atau abjad stasiun. Setiap *line chart* hanya dirender apabila *line chart* memasuki visible offset sebesar 50px, dengan artian *line chart* baru di render ketika ada 50px di bawah atau di atas viewport. Nilai *p-wave* akan ditandai dengan garis merah pada trace seperti pada gambar 4.44.

BAB 5

EVALUASI SISTEM

Evaluasi umum dilakukan menggunakan pengujian end-to-end untuk memastikan sistem telah berjalan dengan semestinya. Pengujian ini dilakukan dari sudut pandang pengguna yang akan menggunakan sistem. Pengujian ini akan secara tidak langsung menguji servis-servis di dalam sistem. Pengujian end-to-end dilakukan dengan menggunakan metode pengujian fungsionalitas dengan 7 skenario yang dijalankan. Jika skenario berjalan dengan sukses dan menghasilkan hasil yang sesuai, penulis memberikan hasil tes dengan label sukses. Akan tetapi, apabila skenario gagal dan tidak memberikan hasil yang sesuai, penulis menandainya sebagai gagal. Skenario tersebut ditunjukkan oleh tabel 5.1

Tabel 5.1: Hasil Evaluasi End-to-end

| No | Skenario | Hasil Pengujian |
|----|---|-----------------|
| 1 | Sistem menunjukan data waveform dari keseluruhan stasiun secara real-time | sukses |
| 2 | Sistem menunjukan data waveform dari keseluruhan stasiun pada rentang waktu | sukses |
| 3 | Sistem dapat mengaktifkan dan mengnonaktifkan stasiun tertentu | sukses |
| 4 | Sistem dapat menambah dan mengurangi stasiun yang digunakan | sukses |
| 5 | Sistem dapat menangkap <i>p wave</i> dari data <i>waveform</i> stasiun | sukses |
| 6 | Sistem dapat menangkap <i>event</i> dan parameter dari data <i>waveform</i> | sukses |
| 7 | Sistem dapat memberikan hasil prediksi lokasi, magnitudo, dan waktu gempa | sukses |

5.1 Evaluasi Machine Learning

Dari kelima model yang telah dibuat, evaluasi diperlukan untuk menentukan sejauh mana model-model tersebut memenuhi ekspektasi dasar. Evaluasi mencakup performa kecepatan inferensi, ketepatan inferensi, dan ukuran model. Khusus untuk model deteksi

gelombang, hasil evaluasi akan dibandingkan dengan model-model lain terhadap dataset pengujian sebagai benchmark untuk memberikan gambaran lebih baik tentang kualitas model yang dikembangkan.

Dalam evaluasi kedua model ini, penulis menetapkan tiga metrik yang perlu dievaluasi, yaitu MAE dari residual, F1 score hasil prediksi, dan ukuran model. Ketiga metrik ini penting untuk diketahui. Metrik pertama (MAE residual) mengindikasikan seberapa cepat model dapat mendeteksi kedatangan gelombang p/s, memungkinkan penyampaian early warning gempa secepat mungkin. Metrik kedua menunjukkan seberapa sering model menghasilkan prediksi false negative atau false positive, yang dapat berpotensi menimbulkan kepanikan di masyarakat. Metrik ketiga memberikan gambaran tidak langsung tentang kecepatan inferensi model, karena waktu inferensi berkorelasi dengan jumlah parameter. Untuk model prediksi parameter gempa, penulis akan menampilkan hasil evaluasi berupa nilai RMSE (Root Mean Squared Error) dari setiap model prediksi parameter gempa yang dibuat.

5.1.1 Metodologi Evaluasi

Evaluasi dilakukan dengan menggunakan sampel data gempa yang diambil dari dalam test set, dengan ukuran sampel sebesar 200 sampel. Metode evaluasi pertama menggunakan mean absolute error (MAE) dari offset picking model. Offset picking didefinisikan sebagai selisih waktu picking yang asli dibandingkan dengan waktu picking hasil prediksi.

Pada metode evaluasi kedua, menggunakan F1 Score. Pertama, dihitung jumlah event yang diprediksi sebagai true positive, true negative, false positive, dan false negative. Dari hasil perhitungan tersebut, dihitung F1 Score-nya. Sebuah event gempa disebut sebagai true positive jika gempa benar-benar terdeteksi ketika event kedatangan gelombang p/s benar-benar terdapat di dalam sliding window model yang berukuran 382. Oleh karena itu, setiap deteksi gelombang yang jaraknya paling jauh 382 datapoints diklasifikasikan sebagai true positive.

Untuk evaluasi ukuran model, penulis dapat menghitung parameter count pada setiap model. Untuk model-parameter gempa, hasil evaluasinya berasal dari perhitungan root mean squared error (RMSE) dari dataset pengujian terhadap error hasil prediksi dengan hasil sebenarnya.

5.1.2 Hasil Evaluasi Offset Picking

Berikut adalah hasil evaluasi model untuk metrik ini dengan model PhaseNet sebagai bandingannya

Tabel 5.2: Hasil Evaluasi Offset Picking

| Model | MAE Offset Picking | | r^2 |
|---------------------|--------------------|------------|-------|
| | μ | σ | |
| Model Gelombang (P) | 0.37 detik | 0.24 detik | 0.99 |
| Model Gelombang (S) | 0.45 detik | 0.55 detik | 0.99 |
| PhaseNet (P) | 0.08 detik | 0.05 detik | 0.99 |
| PhaseNet (S) | 0.13 detik | 0.08 detik | 0.99 |

Dari hasil evaluasi offset picking pada tabel 5.2, terlihat bahwa model gelombang (P) memiliki MAE sebesar 0.37 detik dengan deviasi standar (σ) sekitar 0.24 detik. Sementara itu, model gelombang (S) memiliki MAE sebesar 0.45 detik dengan deviasi standar σ sekitar 0.55 detik. Sebagai perbandingan, model PhaseNet untuk deteksi gelombang P memiliki MAE sekitar 0.08 detik dengan deviasi standar σ sekitar 0.05 detik, dan untuk deteksi gelombang S memiliki MAE sekitar 0.13 detik dengan deviasi standar σ sekitar 0.08 detik. Dengan demikian, model gelombang (P) dan (S) menunjukkan kinerja yang lebih rendah dibandingkan dengan model PhaseNet dalam hal ketepatan offset picking. Untuk setiap model, semuanya memiliki koefisien determinasi yang sangat baik, yaitu 0.99.

PhaseNet memiliki ketepatan yang lebih baik dalam melakukan *picking* karena model ini menerima input berupa data seismogram dengan *sampling rate* (resolusi) sebesar 100 Hz dan lebar *window* sebesar 3000 datapoints. Sementara itu, model yang dibuat oleh penulis menerima input berupa data seismogram dengan *sampling rate* (resolusi) sebesar 20 Hz dan lebar *window* sebesar 387 datapoints. Resolusi data input yang lebih tinggi dari PhaseNet mengakibatkan outputnya juga memiliki resolusi yang tinggi juga. Tingginya resolusi input dan outputnya ini berkontribusi terhadap

keakuratan model PhaseNet ini dalam melakukan picking. Untuk kedepannya, model yang kami buat dapat ditingkatkan performansinya dengan meningkatkan *sampling rate* dari input dan bisa juga dilakukan pemanjangan window.

5.1.3 Hasil Evaluasi F1 Score

Dari hasil eksperimen tersebut, dapat dilihat bahwa model penulis memiliki F1 score yang lebih baik dibandingkan dengan PhaseNet. Model yang dibuat penulis terhitung memiliki F1 score sebesar 0.891 untuk deteksi kedatangan gelombang P, sedangkan model PhaseNet memiliki F1 score sebesar 0.896.

Sementara itu, untuk deteksi kedatangan gelombang S, model penulis memiliki F1 score sebesar 0.985, sementara model PhaseNet memiliki F1 score sebesar 0.801. Hal tersebut menunjukkan bahwa model penulis lebih unggul dalam melakukan deteksi kedatangan gelombang S.

Tabel 5.3: Hasil Evaluasi F1 Score

| Model | F1 Score |
|--------------------------------------|----------|
| Model Deteksi Kedatangan Gelombang P | 0.891 |
| Model Deteksi Kedatangan Gelombang S | 0.985 |
| PhaseNet (untuk deteksi P) | 0.896 |
| PhaseNet (untuk deteksi S) | 0.801 |

5.1.4 Hasil Evaluasi Jumlah Parameter

Berikut adalah hasil evaluasi jumlah parameter untuk model deteksi kedatangan gelombang P dan S, dengan model PhaseNet sebagai perbandingan:

Tabel 5.4: Hasil Jumlah Parameter

| Model | Jumlah Parameter |
|--------------------------------------|--------------------|
| Model Deteksi Kedatangan Gelombang P | 616 ribu parameter |
| Model Deteksi Kedatangan Gelombang S | 616 ribu parameter |
| PhaseNet | 623 ribu parameter |

Dari hasil evaluasi jumlah parameter pada tabel 5.4, terlihat bahwa model deteksi kedatangan gelombang P dan S memiliki jumlah parameter sebanyak 616 ribu parameter untuk masing-masing model. Sebagai perbandingan, model PhaseNet memiliki jumlah parameter sebanyak 623 ribu. Dengan demikian, kedua model deteksi kedatangan gelombang P dan S memiliki jumlah parameter yang serupa dengan model PhaseNet.

5.1.5 Hasil Evaluasi Model Parameter Gempa

Dari hasil evaluasi model parameter gempa pada tabel 5.5, terlihat bahwa model prediksi magnitudo memiliki Mean Absolute Error (MAE) sebesar 0.36 SR dengan deviasi standar σ sebesar 0.26 SR. Sementara itu, model prediksi jarak menunjukkan MAE sebesar 38.2 km dengan deviasi standar σ sebesar 27.3 km, menunjukkan bahwa model ini memiliki error yang cukup signifikan.

Untuk model prediksi kedalaman, MAE yang dihasilkan adalah sebesar 2.08 km dengan deviasi standar σ sebesar 1.56 km. Dengan demikian, model-model parameter gempa yang dikembangkan cenderung memberikan prediksi yang cukup akurat, namun perlu perhatian khusus pada prediksi jarak yang menunjukkan error yang lebih besar. Bahkan, pada hasil evaluasi nilai R Squared, model prediksi jarak memiliki nilai yang mendekati nol yang berarti bahwa nilai error nya hampir sama dengan nilai rata-ratanya, berbeda dengan model magnitudo dan kedalaman yang masih memiliki R Squared yang positif.

Tabel 5.5: Hasil Evaluasi RMSE Setiap Model Parameter

| Model | MAE Parameter Gempa | | r^2 |
|--------------------|---------------------|----------|-------|
| | μ | σ | |
| Prediksi Magnitudo | 0.36 SR | 0.26 SR | 0.73 |
| Prediksi Jarak | 38.2 km | 27.3 km | 0.00 |
| Prediksi Kedalaman | 2.08 km | 1.56 km | 0.25 |

5.2 Evaluasi Sistem *Backend*

Dalam pembangunan sistem *Earthquake Early Warning* (EEWS) yang menggunakan arsitektur *event-driven*, evaluasi performa *backend* menjadi krusial. *Backend* merupakan komponen penting dari EEWS karena semua proses pengolahan data seismik berlangsung di komponen ini. Kinerja *backend* secara langsung mempengaruhi kecepatan dan keakuratan dalam mendeteksi serta merespons peristiwa seismik yang terjadi. Oleh karena itu, subbab ini akan memfokuskan pada evaluasi kinerja *backend*, yang mencakup Producer, Queue, Picker, dan REST API & WebSocket dalam konteks EEWS.

Tujuan dari evaluasi ini adalah untuk memastikan bahwa semua komponen dalam arsitektur backend beroperasi dengan efisiensi yang tinggi dan sesuai dengan ekspektasi desain. Evaluasi ini akan mengukur kemampuan sistem untuk menangani beban kerja dengan jumlah stasiun yang berbeda-beda, yang akan memberikan gambaran tentang *scalability* dan *reliability* sistem. Dengan menguji sistem menggunakan berbagai skenario beban kerja, penulis dapat memahami batasan dari sistem saat ini dan mengidentifikasi area yang memerlukan peningkatan.

Evaluasi dilakukan dengan menggunakan Prometheus, sebuah sistem monitoring yang sangat kuat yang memungkinkan kita untuk mengumpulkan metrik waktu-nyata dari berbagai komponen sistem penulis. Prometheus akan mengumpulkan data seperti throughput, latency, penggunaan sumber daya, error rate, dan metrik lainnya yang relevan dengan performa sistem.

Pentingnya evaluasi ini tidak hanya terletak pada pengukuran performa saat ini, tetapi juga pada pemahaman tentang bagaimana sistem berskala dan bereaksi terhadap perubahan beban kerja. Hal ini penting tidak hanya untuk memastikan sistem yang tangguh selama kondisi normal, tetapi juga untuk menjamin bahwa sistem dapat terus beroperasi secara efektif selama peristiwa seismik yang memerlukan respon cepat dan akurat.

Dengan demikian, subbab ini akan menjelaskan secara rinci bagaimana evaluasi performa *backend* dilakukan, metrik apa saja yang akan diukur, dan bagaimana hasil pengukuran tersebut dapat digunakan untuk meningkatkan sistem EEWS yang penulis kembangkan. Evaluasi ini akan menjadi dasar untuk pengambilan keputusan terkait

dengan peningkatan infrastruktur dan optimisasi kode yang mungkin diperlukan untuk mendukung kebutuhan operasional dari sistem EEWs.

5.2.1 Metodologi Evaluasi

Sebelum memulai evaluasi, penulis menetapkan lingkungan pengujian yang mencerminkan kondisi operasional yang sebenarnya. Ini melibatkan konfigurasi stasiun seismik virtual dalam berbagai skala, mulai dari 3 hingga 1000 stasiun. Setiap stasiun diatur untuk mengirim data seismik secara periodik yang kemudian diproses oleh sistem. Penulis juga akan menggunakan server dengan spesifikasi sesuai pada Tabel 5.6.

Tabel 5.6: Spesifikasi Server Penguji

| | |
|----------------|------------------|
| Sistem Operasi | Ubuntu 22.04 lts |
| CPU | 10 VCPU |
| RAM | 16 GB |
| Boot Disk | 80 GB |

Pengujian dilakukan dengan menjalankan serangkaian simulasi di mana data seismik dikirim ke sistem secara berurutan dari jumlah stasiun yang telah ditentukan. Penulis merekam respons sistem terhadap setiap simulasi dan mengumpulkan data untuk setiap metrik yang telah ditetapkan. Penulis mendefinisikan serangkaian metrik yang akan digunakan untuk mengevaluasi kinerja sistem:

- Throughput: Jumlah data yang berhasil diproses per detik oleh setiap layanan.
- Latency: Waktu yang diperlukan untuk sebuah permintaan dari pengiriman hingga penerimaan respons.
- Penggunaan CPU: Persentase penggunaan CPU oleh setiap layanan selama proses pengujian.
- Penggunaan Memori: Persentase penggunaan memori yang diobservasi selama pengujian.
- Tingkat Keberhasilan: Persentase permintaan yang berhasil diproses tanpa error.

Setelah pengumpulan data selesai, penulis melakukan analisis terhadap data tersebut untuk mengidentifikasi pola, mengevaluasi kinerja, dan mengidentifikasi potensi bottleneck dalam sistem.

5.2.2 Hasil Evaluasi Layanan Producer

Hasil evaluasi dari layanan Producer dalam sistem Earthquake Early Warning System (EEWS) yang direfleksikan pada Tabel 5.7 mengungkapkan sejumlah temuan penting mengenai performa sistem ketika jumlah stasiun bervariasi. Pertama dan paling menonjol adalah peningkatan throughput yang konsisten dengan penambahan jumlah stasiun. Ini dimulai dari 213141 data per detik saat hanya tiga stasiun yang aktif, meningkat secara progresif ke 410998 data per detik dengan seribu stasiun yang terlibat. Penambahan *throughput* ini menunjukkan bahwa layanan Producer memiliki kemampuan untuk menyesuaikan diri dengan skala yang lebih besar, mengelola peningkatan volume data tanpa mengalami kemunduran yang signifikan dalam kinerja.

Tabel 5.7: Hasil Evaluasi Producer

| Jumlah Stasiun | Throughput | Latency | CPU Usage | Memory Usage | Success Rate |
|----------------|-----------------|---------|-----------|--------------|--------------|
| 3 | 213141 data/sec | 0.014s | 3.91% | 0.48% | 100% |
| 27 | 284110 data/sec | 0.018s | 3.81% | 0.45% | 100% |
| 243 | 291713 data/sec | 0.017s | 3.92% | 0.44% | 100% |
| 520 | 361762 data/sec | 0.016s | 3.90% | 0.44% | 100% |
| 1000 | 410998 data/sec | 0.015s | 3.92% | 0.44% | 100% |

Secara lebih terinci, latensi, yang mengukur waktu yang dibutuhkan untuk data diproses dan dikirimkan, hanya mengalami fluktuasi kecil sepanjang evaluasi, dari 0.014 detik hingga 0.018 detik, meskipun dihadapkan pada beban kerja yang meningkat. Stabilitas latensi pada tingkat yang rendah ini menunjukkan bahwa sistem mampu mempertahankan responsivitasnya bahkan saat kompleksitas operasional meningkat. Hal ini penting dalam konteks EEWS, di mana setiap milidetik dalam deteksi dan peringatan gempa bisa memiliki dampak signifikan.

Sementara itu, penggunaan CPU dan memori, yang merefleksikan seberapa banyak sumber daya yang dikonsumsi oleh layanan Producer selama operasi, menunjukkan konsistensi yang mengesankan. Tingkat penggunaan CPU hanya berkisar antara 3.81% hingga 3.92%, dan penggunaan memori stabil antara 0.44% hingga 0.48% di seluruh skenario pengujian. Konsistensi ini, bahkan saat dihadapkan pada peningkatan jumlah stasiun, menandakan bahwa layanan Producer dirancang dengan efisiensi sumber daya yang tinggi. Tak kalah penting, tingkat keberhasilan 100% mencerminkan keandalan layanan Producer, dengan tidak adanya kesalahan yang tercatat selama proses pengujian. Keberhasilan ini menegaskan bahwa layanan dapat diandalkan dalam mengelola dan memproses data seismik untuk sistem peringatan dini gempa.

5.2.3 Hasil Evaluasi Layanan Queue

Hasil pengujian seperti pada Tabel 5.8 menunjukkan bahwa layanan Queue menangani *throughput* dengan sangat baik, dimulai dari 50458 data/detik dengan 3 stasiun hingga 48896 data/detik dengan 1000 stasiun. Kemampuan untuk mempertahankan *throughput* yang hampir konstan menandakan bahwa layanan ini dirancang dengan skalabilitas yang baik dan mampu memproses volume data yang besar tanpa penurunan performa yang signifikan. *Latency* layanan Queue konsisten berada di bawah 0.003 detik, sebuah pencapaian yang mengesankan yang menegaskan bahwa layanan ini dapat merespons dengan cepat terhadap permintaan data. Stabilitas nilai *latency* ini, bahkan saat skala stasiun meningkat, menunjukkan bahwa layanan Queue memiliki kemampuan *real-time processing* yang sangat baik.

Tabel 5.8: Hasil Evaluasi Queue

| Jumlah Stasiun | Throughput | Latency | CPU Usage | Memory Usage | Success Rate |
|----------------|----------------|---------|-----------|--------------|--------------|
| 3 | 50458 data/sec | 0.0025s | 3.15% | 0.13% | 100% |
| 27 | 51673 data/sec | 0.0024s | 3.40% | 0.14% | 100% |
| 243 | 50467 data/sec | 0.0028s | 3.48% | 0.21% | 100% |
| 520 | 47500 data/sec | 0.0026s | 2.99% | 0.33% | 100% |
| 1000 | 48896 data/sec | 0.0026s | 2.73% | 0.51% | 100% |

Penggunaan CPU menunjukkan sedikit peningkatan dari 3.15% ke 3.48% ketika jumlah stasiun meningkat dari 3 menjadi 243, dan kemudian mengalami penurunan kembali ketika jumlah stasiun lebih dari itu. Hal ini dapat diindikasikan sebagai optimalisasi *resource allocation* dalam sistem yang dinamis. Sementara itu, penggunaan memori bertambah secara linear dari 0.13% hingga 0.51% sejalan dengan peningkatan jumlah stasiun. Ini mencerminkan penggunaan memori yang efisien dan tidak adanya kebocoran memori dalam implementasi. Tingkat keberhasilan 100% pada semua skenario pengujian menunjukkan bahwa layanan Queue sangat reliable dan dapat diandalkan. Ini memastikan bahwa layanan ini bisa menjamin integritas data dan tidak ada data yang hilang atau tidak terproses selama proses queueing.

Secara keseluruhan, dari hasil evaluasi yang dilakukan, penulis dapat menyimpulkan bahwa layanan Queue memiliki performa yang sangat handal dengan kemampuan untuk menangani jumlah stasiun yang besar tanpa mempengaruhi *throughput* atau *latency* secara signifikan. Keberhasilan ini merupakan hasil dari desain dan implementasi yang baik dalam sistem EEWS, yang memastikan data diproses dengan cepat dan efisien.

5.2.4 Hasil Evaluasi Layanan Picker

Sesuai dengan hasil pengujian pada Tabel 5.9, penulis mengamati bahwa *throughput* layanan Picker berubah secara signifikan seiring dengan peningkatan jumlah stasiun. Dari awal 1250 data/detik pada 3 stasiun, *throughput* meningkat drastis menjadi 12547 data/detik pada 1000 stasiun. Peningkatan ini menunjukkan bahwa layanan ini mempunyai kemampuan scaling yang baik dan mampu menyesuaikan prosesnya untuk menangani volume data yang lebih besar dengan efisiensi yang meningkat.

Tabel 5.9: Hasil Evaluasi Picker

| Jumlah Stasiun | Throughput | Latency | CPU Usage | Memory Usage | Success Rate |
|----------------|----------------|---------|-----------|--------------|--------------|
| 3 | 1250 data/sec | 0.51s | 3.73% | 0.40% | 100% |
| 27 | 1263 data/sec | 0.48s | 3.73% | 0.42% | 100% |
| 243 | 4952 data/sec | 0.51s | 8.54% | 0.60% | 100% |
| 520 | 10688 data/sec | 0.52s | 6.97% | 0.90% | 100% |
| 1000 | 12547 data/sec | 0.51s | 9.29% | 1.40% | 100% |

Latency tercatat stabil pada 0.51 detik untuk sebagian besar skenario pengujian, dengan penurunan kecil menjadi 0.48 detik pada 27 stasiun dan sedikit peningkatan menjadi 0.52 detik pada 520 stasiun. Stabilitas *latency*, terutama pada jumlah stasiun yang lebih tinggi, menunjukkan bahwa layanan Picker telah dirancang dengan algoritma yang efektif, memungkinkan pemrosesan data seismik dengan waktu respons yang konsisten.

Penggunaan CPU yang tercatat mengalami peningkatan dari 3.73% pada pengujian dengan 3 dan 27 stasiun, menjadi 8.54% pada 243 stasiun, dan stabil di kisaran 6.97% sampai 9.29% untuk 520 dan 1000 stasiun. Penyesuaian ini mencerminkan manajemen resource yang dinamis karena layanan Picker mengalokasikan lebih banyak sumber daya ketika dibutuhkan untuk mempertahankan *throughput* dan *latency*.

Penggunaan memori menunjukkan peningkatan yang lebih linear, dimulai dari 0.40% untuk 3 stasiun dan meningkat menjadi 1.40% untuk 1000 stasiun. Peningkatan ini konsisten dengan peningkatan beban kerja dan menunjukkan bahwa layanan Picker memerlukan lebih banyak memori untuk memproses jumlah data yang lebih besar. Tingkat keberhasilan 100% di semua skenario mengindikasikan keandalan layanan Picker dalam memproses dan memilih data seismik tanpa kesalahan. Ini adalah hasil yang sangat positif, menandakan bahwa layanan Picker dapat diandalkan untuk operasi sistem EEWS.

Dari evaluasi yang penulis lakukan, dapat disimpulkan bahwa layanan Picker memenuhi kriteria kinerja yang telah ditetapkan, dengan kemampuan untuk menangani peningkatan jumlah data seismik tanpa mengorbankan keakuratan dan waktu respons. Kemampuan untuk mempertahankan throughput yang tinggi dan latency yang relatif stabil, sambil mengelola sumber daya CPU dan memori dengan efektif, menunjukkan bahwa layanan ini adalah komponen yang kuat dan vital dalam arsitektur EEWS.

Namun, penggunaan memori yang meningkat sebanding dengan peningkatan jumlah stasiun menunjukkan bahwa ada kebutuhan untuk terus memantau dan mungkin mengoptimalkan penggunaan memori untuk memastikan bahwa sistem dapat beroperasi secara efisien pada skala yang lebih besar. Penyesuaian ini akan penting untuk mempertahankan kinerja sistem yang optimal saat skala operasi EEWS terus berkembang.

5.2.5 Hasil Evaluasi Layanan WebSocket dan REST API

Hasil pengujian pada Tabel 5.10 menunjukkan bahwa *throughput* WebSocket meningkat secara eksponensial dari 37267 data/detik pada 3 stasiun menjadi 3714039 data/detik pada 1000 stasiun. Peningkatan *throughput* yang signifikan ini menunjukkan bahwa layanan WebSocket mampu mengelola aliran data yang besar dan meningkatkan kapasitasnya sesuai dengan peningkatan jumlah stasiun yang terhubung.

Tabel 5.10: Hasil Evaluasi WebSocket dan REST API

| Jumlah Stasiun | Throughput (Websocket) | Latency (REST API) | CPU Usage | Memory Usage | Success Rate |
|----------------|------------------------|--------------------|-----------|--------------|--------------|
| 3 | 37267 data/sec | 0.2s | 2.98% | 0.07% | 100% |
| 27 | 521702 data/sec | 0.4s | 3.07% | 0.12% | 100% |
| 243 | 1635849 data/sec | 1.5s | 5.39% | 0.67% | 100% |
| 520 | 3115998 data/sec | 2.1s | 24.98% | 1.15% | 100% |
| 1000 | 3714039 data/sec | 3.3s | 39.92% | 2.40% | 100% |

Latency untuk REST API menunjukkan peningkatan yang terukur seiring dengan pertambahan jumlah stasiun. Mulai dari 0.2 detik pada 3 stasiun hingga 3.3 detik pada 1000 stasiun, peningkatan ini sesuai dengan ekspektasi mengingat beban

kerja yang juga bertambah. Namun, penting untuk dicatat bahwa *latency* ini tetap berada di bawah ambang batas yang dapat diterima untuk operasional sistem dalam kondisi darurat.

Penggunaan CPU untuk layanan ini meningkat dari 2.98% untuk 3 stasiun menjadi 39.92% untuk 1000 stasiun. Peningkatan ini menunjukkan beban kerja yang lebih berat pada server seiring dengan peningkatan jumlah stasiun. Penggunaan memori juga mengalami peningkatan yang sesuai, dari 0.07% menjadi 2.40%, yang menunjukkan skala alokasi memori yang proporsional dengan beban kerja yang diproses oleh layanan. Tingkat keberhasilan yang tetap pada 100% untuk semua skenario pengujian memberikan bukti yang kuat akan keandalan layanan REST API dan WebSocket. Ini menunjukkan bahwa kedua layanan ini mampu menangani permintaan yang masuk tanpa kehilangan data, yang merupakan indikator kinerja yang sangat baik.

Berdasarkan evaluasi yang dilakukan, dapat menyimpulkan bahwa layanan REST API dan WebSocket memiliki kinerja yang sangat baik dalam mengelola dan meneruskan data seismik yang besar dan variatif dari jaringan stasiun yang luas. Meskipun ada peningkatan *latency* yang terukur seiring dengan penambahan jumlah stasiun, kedua layanan ini menunjukkan skalabilitas yang sangat baik dan kemampuan untuk mengelola sumber daya secara efektif.

Namun, pengamatan penulis juga menunjukkan bahwa ada ruang untuk peningkatan, terutama dalam mengoptimalkan *latency* untuk REST API pada jumlah stasiun yang lebih tinggi. Peningkatan ini dapat melibatkan penyesuaian pada konfigurasi server, optimasi pada kode, atau mungkin penggunaan teknologi penyimpanan *cache* yang lebih canggih untuk memastikan bahwa latensi yang lebih rendah dapat dicapai tanpa mengorbankan keakuratan atau keandalan data yang diterima.

5.2.6 Hasil Evaluasi Keseluruhan

Dalam subbab ini, penulis merangkum evaluasi komprehensif dari sistem EEWS, meliputi aspek *throughput*, *latency*, penggunaan CPU, dan memori, sebagaimana terurai dalam subbab-subbab sebelumnya (5.2.2 hingga 5.2.5). Evaluasi menunjukkan bahwa sistem memiliki kecenderungan performa yang baik dalam kondisi tertentu. Namun,

terdapat ruang untuk peningkatan, khususnya dalam aspek pengelolaan antrian pesan dan skalabilitas sistem.

Sebagai perbandingan dan untuk memperkaya analisis kinerja sistem *backend* EEWS, penelitian terkait performa antara Kafka dan RabbitMQ menunjukkan hasil yang signifikan. Kafka dengan pengaturan optimal menunjukkan kinerja lebih baik dalam hal *throughput* dan latensi dibandingkan RabbitMQ. Hal ini relevan karena sistem EEWS kami menggunakan arsitektur *event-driven* karena efisiensi dan kecepatan antrian pesan sangat kritis.

Rabiee (2018) membahas pengujian Kafka dan RabbitMQ, hasilnya menunjukkan perbedaan kinerja yang signifikan antara kedua sistem tersebut. Kafka terutama dioptimalkan untuk data berbasis aliran dan mengungguli RabbitMQ dalam hal *throughput* dan *latency*. Kafka yang menggunakan mekanisme penyimpanan data ke *disk* secara sekuensial, menunjukkan kinerja yang lebih baik, dengan jumlah pesan yang dikirim lebih dari lima kali lipat dari RabbitMQ, mencapai lebih dari 250.000 pesan per detik dibandingkan dengan hanya 12.000 pesan per detik untuk RabbitMQ. Kafka juga mengungguli RabbitMQ dalam aspek latensi.

Di sisi lain, RabbitMQ cenderung menyimpan semua pesan dalam memori, yang membatasi skalabilitasnya karena adanya batas *prefetch* yang dapat menghentikan konsumen jika mencapai ambang batas, mengakibatkan penumpukan pesan. RabbitMQ juga lebih efisien dalam hal jumlah data yang ditulis ke disk, tetapi lebih berat dalam penggunaan CPU dibandingkan dengan Kafka.

Secara keseluruhan, eksperimen tersebut menunjukkan bahwa Kafka lebih unggul dalam hal *throughput* dan latensi, terutama ketika dikonfigurasi dengan kompresi yang mengurangi penggunaan disk dari 7Gb hingga 4Gb tergantung pada ukuran *batch*. Eksperimen ini memberikan wawasan yang berguna tentang bagaimana kedua teknologi ini dapat dioptimalkan untuk skenario penggunaan yang berbeda, dengan Kafka menonjol dalam skenario yang memerlukan *throughput* tinggi dan latensi rendah. Pengetahuan ini dapat memberikan wawasan berharga dalam peningkatan sistem EEWS, khususnya dalam pemilihan teknologi antrian pesan.

5.3 Evaluasi Frontend

Tujuan dari evaluasi ini adalah untuk memastikan bahwa semua komponen dalam aplikasi web bekerja sesuai fungsionalitas yang dimaksud. Dengan mengevaluasi sistem ini, penulis dapat mengetahui keberhasilan fitur-fitur yang ada, aspek-aspek desain user interface dan user experience yang sudah baik atau bisa ditingkatkan, dan terakhir mengetahui tingkat kebergunaan aplikasi web ini bagi pihak BMKG melalui System Usability Scale. Evaluasi ini akan menjadi dasar untuk meningkatkan aplikasi web yang menjadi interface terhadap sistem EEWS yang penulis kembangkan

Evaluasi aplikasi web dimulai sejak penulis melakukan presentasi kepada tim BMKG pada Jumat, 1 Desember 2023 melalui Zoom. Penulis berhasil mempresentasikan hasil pekerjaan dari sisi model machine learning dan sistem event-driven di backend. Akan tetapi, karena kendala waktu dihentikan tim BMKG sebelum penulis belum sempat mempresentasikan secara keseluruhan aplikasi web.

Pada tanggal 7 Desember 2023, penulis mengirimkan form penilaian dan demo aplikasi web yang diupload di platform Youtube kepada tim BMKG melalui Pembimbing penulis, Bapak Ari Wibisono, S.Kom., M.Kom. Respon penilaian kemudian diberikan pada 8 Desember 2023, pukul 17.05 PM

5.3.1 Hasil Evaluasi Uji Skenario

Pengujian fungsionalitas dilakukan dengan menggunakan menguji 14 skenario yang tertera pada tabel 5.11. Jika skenario berjalan dengan sukses dan menghasilkan hasil yang sesuai, penulis memberikan hasil tes dengan label sukses. Akan tetapi, apabila skenario gagal dan tidak memberikan hasil yang sesuai, penulis menandainya sebagai gagal.

Tabel 5.11: Hasil Uji Skenario

| No | Skenario | Hasil |
|----|--|--------|
| 1 | sistem menunjukan peta indonesia beserta penanda stasiun | sukses |
| 2 | penanda stasiun dapat menunjukan informasi stasiun | sukses |
| 3 | sistem menunjukan lokasi gempa yang terdeteksi pada peta | sukses |
| 4 | sistem dapat memilih mode real-time untuk mendapat data <i>live</i> dari server Seedlink | sukses |
| 5 | sistem dapat memilih mode playback dengan mengisi rentang waktu mulai dan waktu akhir untuk mendapat data dari server FDSNWS | sukses |
| 6 | sistem dapat meminta menghentikan siaran data <i>live</i> dari server Seedlink | sukses |
| 7 | sistem dapat menunjukkan <i>line chart</i> seluruh <i>channel</i> | sukses |
| 8 | sistem dapat mengubah durasi trace yang ditunjukan dalam <i>line chart</i> | sukses |
| 9 | <i>line chart</i> memiliki label stasiun dan channel dari data <i>trace</i> | sukses |
| 10 | <i>line chart</i> menunjukkan nilai amplitudo, mean, dan waktu <i>p-wave</i> dari data <i>trace</i> | sukses |
| 11 | <i>line chart</i> dikosongkan setiap kali mode diubah | sukses |
| 12 | <i>line chart</i> menunjukan penanda waktu data <i>trace</i> | sukses |
| 13 | <i>line chart</i> dapat menunjukkan <i>p wave</i> dengan garis vertikal | sukses |
| 14 | sistem me-render <i>line chart</i> yang berada di <i>viewport</i> dan sekitarnya | sukses |

5.3.2 Hasil Evaluasi Penilaian BMKG

Selain uji fungsionalitas, diperlukan pengujian dari BMKG sebagai pengguna dari sistem. dengan memberikan form penilaian kepada pihak BMKG. Penilaian tersebut dibagi menjadi pertanyaan open-ended, penilaian user interface dan user experience (UI/UX), dan penilaian system usability scale (SUS). Penilaian open-ended akan menggunakan isian, sementara itu penilaian UI/UX dan SUS akan menggunakan skala

likert untuk menilai kesesuaian dengan nilai 1 (terendah) menyatakan sangat tidak setuju, sampai 5 (tertinggi) menyatakan sangat setuju.

Berdasarkan pertanyaan open-ended dalam penilaian, tim BMKG mengemukakan bahwa secara umum penggunaan antarmuka pengguna cukup mudah dan visualisasi peta sudah cukup bagus. Visualisasi data *trace*, indikasi gempa dan p arrival menjadi fitur paling menarik dalam sistem ini. Saran yang diberikan adalah Visualisasi map dapat diperjelas sehingga dapat menampilkan nama daerah, dan meningkatkan performa visualisasi data *trace*.

Kemudian, didapatkan bahwa penilaian user interface dan user experience memiliki nilai rata-rata 4.6875. Terdapat 16 pertanyaan UI/UX mengenai tiga komponen, yaitu komponen peta interaktif, komponen control panel, dan komponen *trace*. Komponen peta interaktif memiliki 4 pertanyaan dengan nilai rata-rata 4,75. Komponen control panel memiliki 8 pertanyaan dengan nilai rata-rata 4.625. Terakhir, komponen trace memiliki 4 pertanyaan dengan nilai rata-rata 4.75. Penilaian SUS menunjukkan skor 70 dari 100, yang dapat diinterpretasikan sebagai diatas rata-rata. Berdasarkan hasil diatas, dapat dikatakan bahwa sistem frontend sudah cukup baik dan mencapai tujuannya sebagai antarmuka pengguna dengan sistem backend.

BAB 6

KESIMPULAN DAN SARAN

Setelah penjelasan pada bab 4 tentang implementasi dan bab 5 tentang evaluasi, bab ini memaparkan kesimpulan yang bisa diambil mengenai pertanyaan masalah. Kemudian, penulis menjelaskan saran-saran yang dapat digunakan untuk penelitian berikutnya.

6.1 Kesimpulan

Dari eksperimen dan penerapan yang telah dijalankan, terdapat beberapa simpulan yang bisa diambil berdasarkan data yang terkumpul. Berikut adalah beberapa kesimpulan yang ditarik dari penelitian ini:

1. Model pendekripsi kedatangan gelombang p/s mampu untuk mendekripsi kedatangan gelombang dengan baik dan sedikit melebihi model PhaseNet. Model prediksi magnitudo dan kedalaman juga memiliki performansi yang baik, akan tetapi model jarak masih memiliki error yang cukup signifikan.
2. Arsitektur *event-driven* memberikan skalabilitas dan fleksibilitas yang memadai, memungkinkan sistem untuk menyesuaikan diri dengan volume data yang dinamis.
3. Pemanfaatan Kafka sebagai *message broker* mampu mengurangi *latency* dan memastikan integritas data dalam situasi jaringan yang tidak stabil.
4. Sistem berhasil memproses data seismik secara *real-time*, dengan latency yang sangat rendah, dan *throughput* yang tinggi, menjadikannya sangat efektif untuk peringatan dini.
5. Aplikasi web dapat digunakan untuk melakukan visualisasi data trace dan juga melihat hasil prediksi *p-wave* dan parameter gempa melalui peta interaktif dan *line chart*. Berdasarkan evaluasi, aplikasi web mudah dimengerti oleh calon pengguna. Untuk efisiensi, aplikasi akan menahan proses render visualisasi *line chart* hingga berada di *viewport*.

6.2 Saran

Penulis menyadari adanya beberapa aspek yang belum sempurna dalam penelitian ini. Untuk itu, penulis menawarkan beberapa rekomendasi untuk penelitian yang akan datang, yaitu:

1. Peningkatan terhadap performansi model prediksi parameter, terutama untuk model deteksi jarak
2. Pengembangan metode untuk mengklasifikasikan apakah suatu *event* yang dideteksi dari dua stasiun yang berbeda adalah event yang sama menggunakan pendekatan yang lebih kompleks karena pendekatan saat ini masih sangat naif.
3. Pengembangan mekanisme *backpressure* yang lebih canggih untuk memanajemen data secara efisien saat terjadi lonjakan data secara tiba-tiba.
4. Optimalisasi konfigurasi Kafka untuk mengurangi *overhead* dan memaksimalkan *throughput* data dalam skenario beban kerja tinggi.
5. Integrasi sistem monitoring yang lebih *robust* untuk mengidentifikasi dan menyelesaikan *bottleneck* secara *real-time*, memastikan sistem selalu beroperasi pada performa optimal.
6. Melakukan isolasi sehingga bisa melayani analisis data lebih dari satu pengguna dalam waktu yang bersamaan.

DAFTAR REFERENSI

- Alaasam, A., Radchenko, G., & Tchernykh, A. (2019). Stateful Stream Processing for Digital Twins: Microservice-Based Kafka Stream DSL. 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), 0804-0809. <https://doi.org/10.1109/SIBIRCON48586.2019.8958367>.
- Allen, R., Gasparini, P., Kamigaichi, O., & Böse, M. (2009). The Status of Earthquake Early Warning around the World: An Introductory Overview. *Seismological Research Letters*, 80, 682-693. <https://doi.org/10.1785/GSSRL.80.5.682>.
- Basren, B. (2019). Forensik Investigation Framework For MongoDB Deployment Type Replica Set. , 13. <https://doi.org/10.26555/jifo.v13i1.a10301>.
- Berners-Lee, T., Fielding, R., & Nielsen, H. (1996). Hypertext Transfer Protocol - HTTP/1.0. RFC, 2068, 1-162. <https://doi.org/10.17487/RFC1945>.
- Beyreuther, Moritz & Barsch, Robert & Krischer, Lion & Megies, Tobias & Behr, Yannik & Wassermann, Joachim. (2010). ObsPy: A Python Toolbox for Seismology. *Seismological Research Letters*. 81. 530-533. doi: <https://doi.org/10.1785/gssrl.81.3.530>.
- Boucher, T., & Yalcin, A. (2006). The Relational Database Model. , 27-69. <https://doi.org/10.1016/B978-012370492-4/50002-7>.
- Crockford, D. (2006). The application/json Media Type for JavaScript Object Notation (JSON). RFC, 4627, 1-10. <https://doi.org/10.17487/RFC4627>.
- B. Dost, J. Zednik, J. Havskov, R. J. Willemann and P. Bormann. (2011). Seismic Data Formats, Archival and Exchange. doi: https://doi.org/10.2312/GFZ.NMSOP-2_CH10 .
- D'Addona, D. M. (2014). Neural network. CIRP Encyclopedia of Production Engineering, 911–918. https://doi.org/10.1007/978-3-642-20617-7_6563.
- Ed-Douibi, H., Izquierdo, J., & Cabot, J. (2018). Automatic Generation of Test Cases for REST APIs: A Specification-Based Approach. 2018 IEEE 22nd International

- Enterprise Distributed Object Computing Conference (EDOC), 181-190. <https://doi.org/10.1109/EDOC.2018.00031>.
- Esposito, C., Castiglione, A., & Choo, K. (2016). Challenges in Delivering Software in the Cloud as Microservices. IEEE Cloud Computing, 3, 10-14. <https://doi.org/10.1109/MCC.2016.105>.
- Fette, I., & Melnikov, A. (2011). The WebSocket Protocol [White paper]. RFC Editor. doi: <https://doi.org/10.17487/RFC6455>.
- GFZ German Research Centre for Geosciences and GEMPA GmbH (2008). The SeisComP seismological software package. GFZ Data Services. Doi: <https://doi.org/10.5880/GFZ.2.4.2020.003>
- Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222–2232. <https://doi.org/10.1109/tnnls.2016.2582924>
- Herbert, David. June 2021. A Beginner's Guide to JavaScript's The Intersection Observer API. <https://hackernoon.com/a-beginners-guide-to-javascripts-the-intersection-observer-api-j8s32rb>
- Imre, G., Mezei, G., & Sarosi, R. (2016). Introduction to a WebSocket benchmarking infrastructure. 2016 Zooming Innovation in Consumer Electronics International Conference (ZINC), 84-87. <https://doi.org/10.1109/ZINC.2016.7513661>.
- Incorporated Research Institutions for Seismology. (2012) SEED Reference Manual-Standard for the Exchange of Earthquake Data.
- Jones, Carl. (2022). A Comprehensive UI & UX Guide to Master Web Design and Mobile App Sketches for Beginners and Pros. ISBN 979-8789439784
- Kim, H., Bang, J., Son, S., Joo, N., Choi, M., & Moon, Y. (2019). Message Latency-Based Load Shedding Mechanism in Apache Kafka. , 731-736. https://doi.org/10.1007/978-3-030-48340-1_58.
- Krischer, Lion., et al. (2015). ObsPy: a bridge for seismology into the scientific Python ecosystem. doi: <https://doi.org/10.1088/1749-4699/8/1/014003>.

- Loreto, Salvatore., Saint-Andre, Peter., Salsano, Stefano., and Wilkins, Greg (2011). Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP [White paper]. RFC Editor.
- L. Deng and D. Yu, Deep Learning: Methods and Applications, Foundations and Trends in Signal Processing, vol. 7, no. 3–4, pp. 197-387, 2014.
- Magnoni, L. (2015). Modern Messaging for Distributed Systems. Journal of Physics: Conference Series, 608. <https://doi.org/10.1088/1742-6596/608/1/012038>.
- N. Pillay and J. Wing, "Agile UX: Integrating good UX development practices in Agile," 2019 Conference on Information Communications Technology and Society (ICTAS), Durban, South Africa, 2019, pp. 1-6, doi: <https://ieeexplore.ieee.org/document/8703607>.
- Oh, F., Kim, S., Eom, H., Yeom, H., Park, J., & Lee, Y. (2011). A scalable and adaptive cloud-based message brokering service. 13th International Conference on Advanced Communication Technology (ICACT2011), 498-501.
- O’Shea, Nash Ryan, An Introduction to Convolutional Neural Networks, Aberystwyth University, <https://doi.org/10.48550/arXiv.1511.08458>
- Porccello, Eve., Banks, Alex. (2020). Learning React, 2nd Edition. O'Reilly Media, Inc..
- Rabiee, A. (2018). Analyzing Parameter Sets For RabbitMQ and Apache Kafka On A Cloud Platform.
- Robillard, M. (2009). What Makes APIs Hard to Learn? Answers from Developers. IEEE Software, 26. <https://doi.org/10.1109/MS.2009.193>.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Lecture Notes in Computer Science, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28
- Rosa, G., Scalabrino, S., & Oliveto, R. (2022). Assessing and Improving the Quality of Docker Artifacts. 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), 592-596. <https://doi.org/10.1109/ICSME55016.2022.00081>.

- Seismological Facility for the Advancement of Geoscience. (n.d.) Seedlink. <https://ds.iris.edu/ds/nodes/dmc/services/seedlink/>.
- Saldamli, G., Doshatti, A., Kapadia, D., Nyati, D., Bodiwala, M., & Ertaul, L. (2021). Enterprise Backend as a Service (EBaaS), 1077-1099. https://doi.org/10.1007/978-3-030-69984-0_78.
- Stull, Edward. (2018). UX Fundamentals for Non-UX Professionals. Apress Berkeley, CA. <https://doi.org/10.1007/978-1-4842-3811-0>.
- T. Bray (2017). The JavaScript Object Notation (JSON) Data Interchange Format [White paper]. RFC Editor. doi: <https://www.rfc-editor.org/rfc/rfc8259>.
- Wieber, N. (2020). Automated Generation of Client-Specific Backends Utilizing Existing Microservices and Architectural Knowledge. 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 1158-1160. <https://doi.org/10.1145/3324884.3415283>.
- Wieland, M., Martin, D., Kopp, O., & Leymann, F. (2009). SOEDA: A Method for Specification and Implementation of Applications on a Service-Oriented Event-Driven Architecture, 193-204. https://doi.org/10.1007/978-3-642-01190-0_17.
- Woollam, J., Münchmeyer, J., Tilmann, F., Rietbrock, A., Lange, D., Bornstein, T., Diehl, T., Giuchi, C., Haslinger, F., Jozinović, D., Michelini, A., Saul, J., & Soto, H. (2022). SeisBench - A Toolbox for Machine Learning in Seismology. in Seismological Research Letters <https://doi.org/10.1785/0220210324>.
- Zhang, P., Xing, L., Yang, N., Tan, G., Liu, Q., & Zhang, C. (2018). Redis++: A High Performance In-Memory Database Based on Segmented Memory Management and Two-Level Hash Index. 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), 840-847. <https://doi.org/10.1109/BDCLOUD.2018.00125>.

LAMPIRAN

Lampiran 1: Implementasi Kode *Postprocessing* Machine Learning

Lampiran ini berisi implementasi kode yang terdapat pada langkah *postprocessing* pada bagian implementasi *machine learning*.

Lampiran 1.1: Kode Deteksi Kedatangan Gempa

```
def detect_wave_arrival(prediction, threshold):
    arrival_idx = np.where((prediction > threshold).any(axis=1))[0]
    return len(arrival_idx) > 0
```

Lampiran 1.2: Kode Ekstraksi Informasi Picking

```
def pick_arrival(prediction: np.ndarray, threshold=0.5,
                 window_size=settings.WINDOW_SIZE) -> Tuple[bool, float, int]:
    # Detect p wave occurrence
    detected_indices = np.where((prediction > threshold).any(axis=1))[0]

    # Case if p wave is detected
    if detected_indices.any():
        first_detection_index = detected_indices[0]
        ideal_deviation = np.array(
            detected_indices) - first_detection_index

        # For all triggered windows, find its argmax
        argmax = np.array(prediction[detected_indices].argmax(axis=1))
        deviation = argmax + ideal_deviation # predicted deviation

        # Find mean while excluding outliers
        mean_approx = first_detection_index - (window_size
                                                round(np.mean(deviation)))

    return True, mean_approx, len(detected_indices)

    # Case if no p wave detected
    return False, 0.0, 0
```

Lampiran 1.3: Kode untuk Mendapatkan Koordinat Episentrum Gempa dari Hasil Inferensi 3 Stasiun

```
def calculate_loc(
    station_latitudes,
    station_longitudes,
    distances # array prediksi jarak antara tiap stasiun ke episenter
):
```

```

# Convert to radian
station_latitudes_rad = np.radians(station_latitudes)
station_longitudes_rad = np.radians(station_longitudes)

for i in range(len(station_latitudes_rad)-1):
    for j in range(i+1, len(station_latitudes)):

        # distance between two stations
        R = haversine(
            station_latitudes[i],
            station_longitudes[i],
            station_latitudes[j],
            station_longitudes[j]
        )

        # Radians coordinate of two stations
        xi = station_latitudes_rad[i]
        yi = station_longitudes_rad[i]
        xj = station_latitudes_rad[j]
        yj = station_longitudes_rad[j]
        ri = distances[i]
        rj = distances[j]

        # Auxiliary variable to calculate coordinate
        x_delta = 0.5 * np.sqrt(
            2 * (ri**2+rj**2)/R**2 - (ri**2-rj**2)**2/R**4 - 1
        ) * (yj-yi)

        y_delta = 0.5 * np.sqrt(
            2 * (ri**2+rj**2)/R**2 - (ri**2-rj**2)**2/R**4 - 1
        ) * (xi-xj)

        x_base = 0.5*(xi+xj) + (ri**2-rj**2)/(2*R**2) * (xj-xi)
        y_base = 0.5*(yi+yj) + (ri**2-rj**2)/(2*R**2) * (yj-yi)

        # Coordinates intersection points
        x_1 = x_base + x_delta
        x_2 = x_base - x_delta
        y_1 = y_base + y_delta
        y_2 = y_base - y_delta

        # 1 circle intersects at two points, compose the previously
        # calculated result into this structure
        points.append(np.array([[x_1, y_1], [x_2, y_2]]))

# From all intersection points generated from previous computation,
# Find points with the least variance
triplets = []
variances = []
for i in range(2):
    for j in range(2):
        for k in range(2):
            # Generate triplets
            triplet = np.ndarray = np.array(
                [points[0][i], points[1][j], points[2][k]])
            triplets.append(triplet)

            # Calculate variance
            variances.append(triplet.var(axis=0).sum())

```

```
# Select the triplets with the least variance value
variances = np.array(variances)
argmin = variances.argmin()

# Retrieve argmin-th triplet
triplet: np.ndarray = triplets[argmin]

# Project triplet into real number
triplet = triplet.real

# Take the average
ans = triplet.mean(axis=0)

# Convert result back to degree, 6371.0 is earth radius
ans *= 180 / np.pi / 6371.0
```

Lampiran 2: Penilaian BMKG

| No | Pertanyaan Open-Ended | Jawaban |
|----|--|---|
| 1 | Silakan berikan komentar umum tentang sistem yang kami presentasikan. | Visualisasi peta dan grafiknya sudah bagus |
| 2 | Apa fitur paling berguna atau menarik dari sistem ini? | Visualisasi grafik, indikasi gempa dan p arrival |
| 3 | Apakah ada aspek dari sistem yang perlu diperbaiki atau dikembangkan? | Performa visualisasi grafik |
| 4 | Bagaimana pendapatnya tentang desain dan kemudahan penggunaan antarmuka pengguna? | Cukup mudah |
| 5 | Apakah Bapak/Ibu menemukan kesulitan dalam navigasi atau menggunakan fitur tertentu? | Tidak |
| 6 | Bagaimana pendapat Bapak/Ibu tentang akurasi dan efektivitas model ML dalam mendeteksi gempa bumi? | Lumayan bagus |
| 7 | Apakah ada masukan tentang cara sistem kami memproses dan menganalisis data? | - |
| 8 | Berdasarkan presentasi kami, apakah Bapak/Ibu akan tertarik menggunakan sistem ini di masa depan? | Mungkin |
| 9 | Silakan tambahkan komentar atau saran tambahan yang Bapak/Ibu miliki untuk kami. | Visualisasi map dapat diperjelas sehingga dapat menampilkan nama daerah |

| No | Penilaian User Interface dan User Experience | Nilai (1-5) |
|----|--|-------------|
| 1 | peta interaktif memiliki ukuran, warna, dan bentuk yang nyaman untuk dilihat | 4 |
| 2 | peta interaktif yang dioperasikan dengan digeser untuk mengubah posisi, dan dizoom-in dan dizoom-out untuk mengubah skala, akan nyaman untuk digunakan | 5 |
| 3 | peta interaktif menunjukkan posisi dan informasi stasiun seismograf di indonesia nyaman untuk dilihat dan sudah memiliki informasi yang cukup bagi pengguna. | 5 |

| | | |
|----|--|---|
| 4 | peta interaktif menunjukan posisi dan informasi dari origin gempa di indonesia nyaman untuk dilihat dan sudah memiliki informasi yang cukup bagi pengguna. | 5 |
| 5 | pengguna dapat dengan mudah dan intuitif mengubah status stasiun seismograf menjadi enabled atau disabled. | 5 |
| 6 | pengguna dapat dengan mudah dan intuitif mengatur panjang durasi trace yang ditunjukan | 5 |
| 7 | pengguna dapat dengan mudah dan intuitif mengatur stasiun apa yang urutannya diprioritaskan di paling atas | 5 |
| 8 | pengguna dapat dengan mudah dan intuitif melakukan pemutaran dari data waveform pada waktu terdahulu melalui mode playback | 5 |
| 9 | pengguna dapat dengan mudah dan intuitif melakukan pemutaran dan penghentian dari data waveform pada waktu saat ini (real-time) melalui mode live | 4 |
| 10 | panel menunjukan dengan jelas daftar stasiun yang ada di sistem beserta status dari channelnya. | 4 |
| 11 | panel menunjukan dengan jelas daftar p-wave yang terdeteksi dengan informasi stasiun, channel, waktu p-wave, dan waktu prediksi terjadi. | 4 |
| 12 | panel menunjukan dengan jelas informasi mengenai gempa yang terjadi | 5 |
| 13 | sistem menunjukan amplitudo, mean, dan p-wave suatu channel berdasarkan data trace | 5 |
| 14 | data amplitudo, mean, dan p-wave sudah cukup bagi pengguna sebagai informasi tambahan untuk data trace suatu channel | 5 |
| 15 | grafik data trace berhasil divisualisasikan dengan baik sehingga nyaman dilihat dan mudah dimengerti | 4 |
| 16 | grafik data trace memberikan garis vertikal untuk menandakan p-wave sehingga nyaman dilihat dan mudah dimengerti | 5 |

| No | Penilaian System Usability Scale | Nilai (1-5) |
|----|----------------------------------|-------------|
|----|----------------------------------|-------------|

| | | |
|----|--|---|
| 1 | I think that I would like to use this system frequently | 3 |
| 2 | I found the system unnecessarily complex | 4 |
| 3 | I thought the system was easy to use | 5 |
| 4 | I think that I would need the support of a technical person to be able to use this system. | 3 |
| 5 | I found the various functions in this system were well integrated. | 5 |
| 6 | I thought there was too much inconsistency in this system. | 1 |
| 7 | I would imagine that most people would learn to use this system very quickly. | 3 |
| 8 | I found the system very cumbersome to use. | 1 |
| 9 | I felt very confident using the system. | 4 |
| 10 | I needed to learn a lot of things before I could get going with this system. | 3 |