Laporan Praktek Topik Khusus 3

Message Queue

SEMESTER VI

DISUSUN OLEH :

KURNIAWAN ALEXANDER

2211083030

DOSEN PENGAMPU :

YULHERNIWATI, S.Kom.,MT

YUNUS SUPRIADI WIJAYA

PROGRAM STUDI TEKNOLOGI REKAYASA PERANGKAT LUNAK

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI PADANG

2025

A. Landasan Teori

1. **Golang (Go Language)**
   Golang, atau Go, adalah bahasa pemrograman open-source yang dikembangkan oleh Google dengan tujuan utama menyediakan efisiensi, kesederhanaan, dan keandalan dalam pengembangan perangkat lunak. Bahasa ini memiliki dukungan bawaan untuk pemrograman konkuren melalui fitur goroutine dan channel, menjadikannya sangat ideal untuk membangun aplikasi berbasis jaringan, sistem terdistribusi, dan layanan web. Golang juga memiliki garbage collector otomatis yang membantu pengelolaan memori tanpa perlu intervensi manual. Karena bersifat statis dan dikompilasi, kesalahan dapat terdeteksi lebih awal pada saat kompilasi. Dengan performa yang tinggi dan framework populer seperti Gin dan Echo, Golang sering dipilih dalam pengembangan REST API maupun aplikasi web modern.

2. **API (Application Programming Interface)**
   API merupakan antarmuka yang memungkinkan satu aplikasi berinteraksi dengan aplikasi lain. Dalam konteks pengembangan aplikasi web, API memainkan peran penting dalam pertukaran data antar sistem. Salah satu bentuk API yang umum digunakan adalah RESTful API, yang memanfaatkan protokol HTTP untuk memfasilitasi operasi dasar terhadap data, seperti **Create (membuat data baru), Read (mengambil data), Update (memperbarui data),** dan **Delete (menghapus data)** — yang dikenal dengan istilah CRUD. Pendekatan ini membuat pengelolaan dan integrasi data menjadi lebih mudah dan terstruktur.

3. **Erlang**
   Erlang adalah bahasa pemrograman berparadigma fungsional yang dikembangkan untuk menciptakan sistem yang mampu menangani banyak proses secara bersamaan (high concurrency), tahan terhadap kesalahan (fault-tolerant), dan mudah diskalakan (scalable). Dengan pendekatan proses ringan yang saling berkomunikasi melalui pengiriman pesan, Erlang dapat menjalankan banyak tugas secara paralel secara efisien. Filosofi “let it crash” atau “biarkan gagal” memungkinkan sistem untuk menangani kegagalan tanpa menghentikan keseluruhan layanan. Selain itu, Erlang dilengkapi dengan OTP (Open Telecom Platform), sebuah kerangka kerja yang menyediakan kumpulan pustaka dan prinsip arsitektural untuk membangun sistem yang andal dan tahan banting.

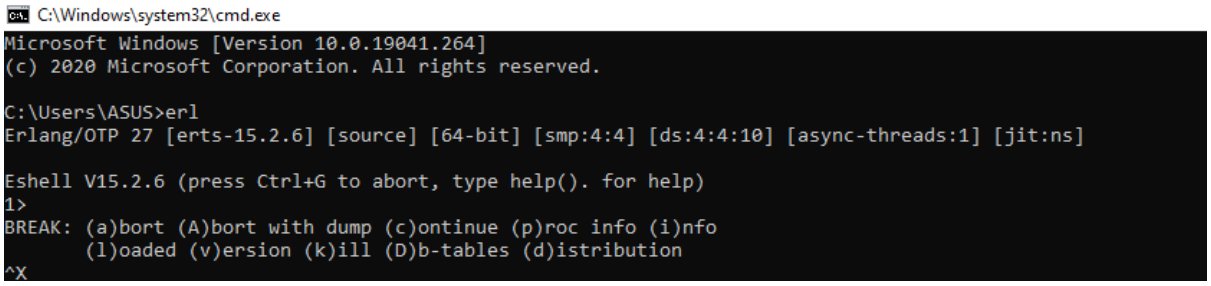
4. **RabbitMQ**
   RabbitMQ adalah perangkat lunak open-source yang berfungsi sebagai message broker dan mendukung protokol AMQP (Advanced Message Queuing Protocol). Peran utamanya adalah menerima, menyimpan sementara, dan mengirimkan pesan antar aplikasi. RabbitMQ memungkinkan komunikasi antar sistem berjalan dengan efisien, terutama dalam lingkungan yang tersebar secara geografis atau sistem mikroservis. Dengan mendukung berbagai pola komunikasi seperti point-to-point (satu pengirim ke satu penerima) dan publish/subscribe (satu pengirim ke banyak penerima), RabbitMQ menjadi solusi fleksibel untuk kebutuhan distribusi pesan dalam skala besar.

B. Tools

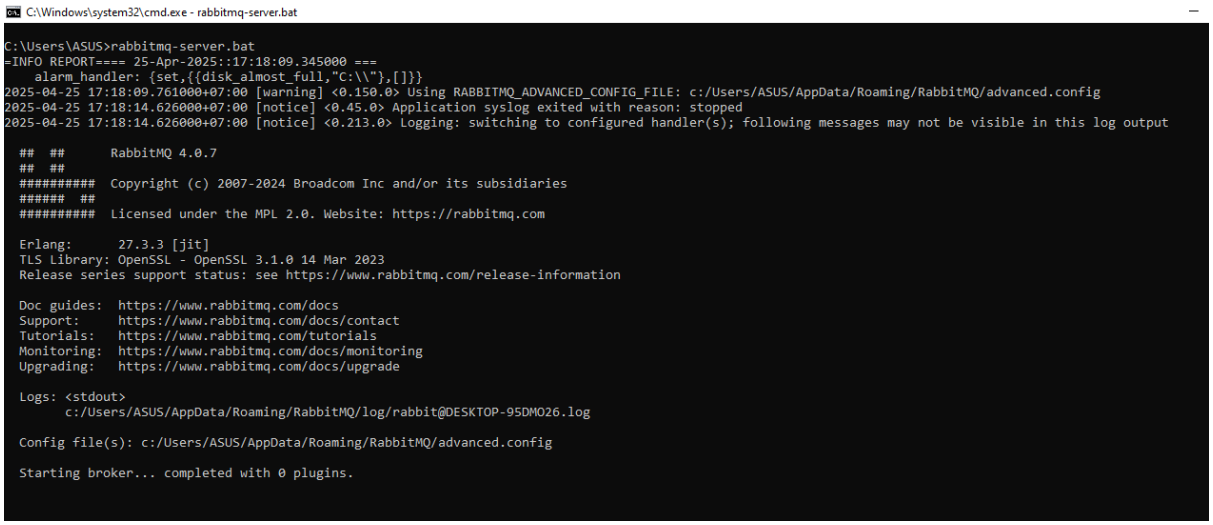
- Erlang
- Golang
- Vs Code
- RabbitMQ

C. Langkah Kerja

1. Install Erlang dan RabbitMQ
    - Download dan Install Erlang (RabbitMQ perlu erlang)



    - Download dan Install RabbitMQ
    - Menambahkan RabbitMQ ke Path
    - Menjalankan RabbitMQ



2. Clone Repository Notification Publisher
    - Lakukan clone repository yang akan dijadikan bahan praktek
    - Membuat file go.mod di direktori Anda, yang akan melacak dependensi proyek Anda.
    - Menambahkan dependensi yang diperlukan yang ditemukan dalam kode Anda ke file go.mod.
    - Menyalin semua dependensi proyek ke direktori vendor di dalam direktori proyek Anda.

```
PS D:\Tugas Topik Khusus\Message_Queue> git clone https://github.com/kusnadi8605/notification_publisher
Cloning into 'notification_publisher'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 30 (delta 4), reused 28 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (30/30), 5.61 KiB | 410.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
PS D:\Tugas Topik Khusus\Message_Queue> cd notification_publisher
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> go mod init notification_publisher
go: creating new go.mod: module notification_publisher
go: to add module requirements and sums:
        go mod tidy
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> go mod tidy
go: finding module for package github.com/labstack/echo/v4
go: finding module for package github.com/rabbitmq/amqp091-go
go: finding module for package github.com/kelseyhightower/envconfig
go: downloading github.com/kelseyhightower/envconfig v1.4.0
go: downloading github.com/rabbitmq/amqp091-go v1.10.0
go: found github.com/labstack/echo/v4 in github.com/labstack/echo/v4 v4.13.3
go: found github.com/kelseyhightower/envconfig in github.com/kelseyhightower/envconfig v1.4.0
go: found github.com/rabbitmq/amqp091-go in github.com/rabbitmq/amqp091-go v1.10.0
go: downloading go.uber.org/goleak v1.3.0
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> go mod vendor
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher>
```

3. Jalankan Aplikasi
    - Run file main.go untuk menjalankan aplikasi

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> ^C
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> $env:PORT = "8081"
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> go run main.go
2025/04/25 17:40:50 Using Exchange: notifications


   ____    __
  / __/___/ /  ___
 / _// __/ _ \/ _ \
/___/\__/_//_/\___/ v4.13.3
High performance, minimalist Go web framework
https://echo.labstack.com
____________________________________O/_______
                                    O\
⇨ http server started on [::]:8081
```

4. Menguji Aplikasi
    - Mengirim pesan ke server dan hasil outputnya

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> $body = @{
>>     order_id  = "12345"
>>     user_id   = "67890"
>>     content   = "New order received"
>>     timestamp = "2025-03-11T10:00:00Z"
>> } | ConvertTo-Json
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher>
PS D:\Tugas Topik Khusus\Message_Queue\notification_publisher> Invoke-RestMethod -Uri "http://localhost:8081/publish" `
>>                 -Method POST `
>>                 -Body $body `
>>                 -ContentType "application/json"

code message
---- -------
 200 Message published successfully
```

5. Clone Aplikasi Consumer
    - Lakukan clone repository yang akan dijadikan bahan praktek
    - Membuat file go.mod di direktori Anda, yang akan melacak dependensi proyek Anda.
    - Menambahkan dependensi yang diperlukan yang ditemukan dalam kode Anda ke file go.mod.
    - Menyalin semua dependensi proyek ke direktori vendor di dalam direktori proyek Anda.

```
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\Tugas Topik Khusus\Message_Queue>git clone https://github.com/kusnadi8605/notification_consumer
Cloning into 'notification_consumer'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 32 (delta 6), reused 32 (delta 6), pack-reused 0 (from 0)
Receiving objects: 100% (32/32), 4.22 KiB | 287.00 KiB/s, done.
Resolving deltas: 100% (6/6), done.

D:\Tugas Topik Khusus\Message_Queue>cd notification_consumer

D:\Tugas Topik Khusus\Message_Queue\notification_consumer>go mod init notification_consumer
go: creating new go.mod: module notification_consumer
go: to add module requirements and sums:
        go mod tidy

D:\Tugas Topik Khusus\Message_Queue\notification_consumer>go mod tidy
go: finding module for package github.com/rabbitmq/amqp091-go
go: finding module for package github.com/kelseyhightower/envconfig
go: found github.com/kelseyhightower/envconfig in github.com/kelseyhightower/envconfig v1.4.0
go: found github.com/rabbitmq/amqp091-go in github.com/rabbitmq/amqp091-go v1.10.0

D:\Tugas Topik Khusus\Message_Queue\notification_consumer>go mod vendor

D:\Tugas Topik Khusus\Message_Queue\notification_consumer>
```

6. Menjalankan Aplikasi Consumer
    - Run file main.go pada folder email

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_consumer> go run cmd/email/main.go
2025/04/25 17:58:24 Using Exchange: notifications
2025/04/25 17:58:24 [EMAIL] Listening for messages...
```

    - Run file main.go pada folder sms

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_consumer> go run cmd/sms/main.go
2025/04/25 17:58:52 Using Exchange: notifications
2025/04/25 17:58:52 [SMS] Listening for messages...
```

    - Run file main.go pada folder fcm

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_consumer> go run cmd/fcm/main.go
2025/04/25 17:59:15 Using Exchange: notifications
2025/04/25 17:59:15 [FCM] Listening for messages...
```

7. Outputnya
    - Output Email

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_consumer> go run cmd/email/main.go
2025/04/25 17:58:24 Using Exchange: notifications
2025/04/25 17:58:24 [EMAIL] Listening for messages...
2025/04/25 18:01:05 [EMAIL] Received message: {OrderID:12345 UserID:67890 Content:New order received Tim
estamp:2025-03-11T10:00:00Z}
sending email ... New order received
```

    - Output SMS

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_consumer> go run cmd/sms/main.go
2025/04/25 17:58:52 Using Exchange: notifications
2025/04/25 17:58:52 [SMS] Listening for messages...
2025/04/25 18:01:05 [SMS] Received message: {OrderID:12345 UserID:67890 Content:New order received Times
tamp:2025-03-11T10:00:00Z}
sending sms ... New order received
```

    - Output FCM

```
PS D:\Tugas Topik Khusus\Message_Queue\notification_consumer> go run cmd/fcm/main.go
2025/04/25 17:59:15 Using Exchange: notifications
2025/04/25 17:59:15 [FCM] Listening for messages...
2025/04/25 18:01:05 [FCM] Received message: {OrderID:12345 UserID:67890 Content:New order received Times
tamp:2025-03-11T10:00:00Z}
sending fcm ...  New order received
```

D. Kesimpulan

Penggunaan RabbitMQ dengan Go sebagai Message Queue

Mengintegrasikan RabbitMQ sebagai sistem antrian pesan (message queue) dalam pengembangan aplikasi menggunakan Go memberikan solusi yang andal dan mudah diskalakan, terutama untuk kebutuhan komunikasi asinkron. RabbitMQ memungkinkan pemisahan logika aplikasi menjadi dua peran utama, yaitu produsen (producer) dan konsumen (consumer), sehingga proses pertukaran data dapat dilakukan secara efisien dan tidak saling bergantung secara langsung.

Bahasa Go, melalui pustaka pendukung RabbitMQ, memudahkan penerapan berbagai pola komunikasi pesan serta menyediakan mekanisme penanganan kesalahan yang tangguh. Hal

ini memungkinkan aplikasi untuk terus berjalan meskipun terjadi gangguan atau kegagalan pada salah satu bagian sistem. Kombinasi kekuatan RabbitMQ dalam pengelolaan pesan dan performa tinggi yang dimiliki Go menjadikan keduanya sangat ideal untuk membangun aplikasi dengan kebutuhan throughput tinggi, keandalan sistem yang kuat, serta arsitektur yang fleksibel dan mudah diperluas.