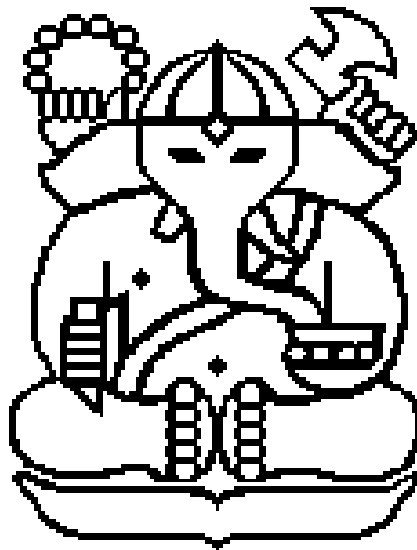


# **Pengembangan Perangkat Lunak Terdistribusi pada Domain Ticketing dan Reservation IF4031 Pengembangan Aplikasi Terdistribusi**

Spesifikasi Tugas Besar IF4031 Pengembangan Aplikasi Terdistribusi 2023/2024



PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
<2023/2024>

Context :

Sejak pandemi COVID-19 usai, dunia entertainment menjadi naik daun lagi. Banyak artis ataupun *public figure* yang berkunjung ke banyak negara dan melaksanakan banyak event. Peristiwa ini membuat kantor tempat kamu bekerja ingin mengembangkan sebuah perangkat lunak yang memungkinkan pengguna untuk memesan tiket pada acara tertentu. Perangkat lunak ini hadir sebagai penengah customer/client/app lain untuk booking acara tertentu.

Namun kamu mendapatkan project manager yang menyebalkan pada proyek ini. Sebut saja namanya Philip. Pak Philip, sebagai project manager yang *baik* menginginkan perangkat lunak yang dibuat dapat mengakomodasi *load* yang besar, dan *concurrency* yang tinggi. Oleh karena itu, Pak Philip memberikan gambaran bahwa sistem ticketing yang dibangun harus dibangun dengan arsitektur *microservice* dengan setidaknya tiga buah *service* terdefinisi, dimana 1 *service* merupakan *service* yang akan berinteraksi langsung dengan end user, sebut saja *client app*, dan 1 *service* lagi merupakan *service* yang akan menyimpan data kursi di berbagai *event* beserta dengan ketersediaannya, sebut saja *ticket app*, selain itu, aplikasi yang dibangun juga akan memanfaatkan *payment gateway* yang akan disimulasikan menjadi *service* tersendiri yang akan disebut sebagai *payment service*.

Karena Pak Philip merupakan *project manager* yang tidak terlalu mengerti pemrograman, maka implementasi diserahkan kepada anda, namun pak Philip sambil tetap berusaha menjadi project manager yang baik, memberikan anda spek masing-masing app sebagai berikut:

Client App (Service 1) :

- Client app harus memiliki CRUD functionality untuk manajemen data pengguna
- Client app harus menyimpan data pengguna secara persisten
- Client app memungkinkan pengguna tertentu untuk melakukan booking terhadap kursi dari *event* tertentu yang terdaftar pada *ticket app*.
- Client app harus menyimpan data historis booking ticket masing-masing pengguna termasuk status bookingnya
- Client app harus menyimpan data booking ticket yang berhasil maupun gagal

Ticket App (Service 2) :

- *Ticket app* harus memiliki CRUD functionality untuk manajemen data event dan kursi yang tersedia
- *Ticket app* harus menyimpan data kursi yang tersedia yang setidaknya memuat ID Kursi, status (OPEN | ON GOING | BOOKED)
- Karena *Ticket App* tidak menyelenggarakan satupun dari *event* yang ada, melainkan hanya sekedar menjadi penengah antara *customer* dan penyelenggara *event* tertentu, maka dalam proses pemesanan *Ticket app* akan melakukan pemanggilan eksternal ke sistem yang berkaitan (sistem event X, Y, dst). Pemanggilan eksternal ini terjadi secara *synchronous* dan hanya berperan untuk *hold* kursi dari *event* tertentu. Pemanggilan eksternal ini memiliki tingkat kegagalan 20% (simulasikan dengan random saja)

- Setelah *Ticket App* berhasil melakukan *hold* dari kursi dari *event* tertentu, *ticket app* akan membuat *invoice* ke *Payment App*. *Payment App* kemudian akan mengembalikan *invoice* number dan url untuk membayarnya ke *Ticket App* yang kemudian akan diteruskan pada *Client App*.
- Proses "*hold seat*" yang dilakukan dari client ke *ticket app* terjadi secara synchronous
- *Ticket app* harus tetap dapat menerima booking meskipun sedang ada booking yang diproses (belum selesai diproses)
- *Ticket app* akan memberitahu *client app* apabila ada booking yang berhasil/gagal. Apabila booking berhasil, *ticket app* akan meng-generate PDF hasil booking dan mengirimkannya juga ke client (proses/prosedur *storing & delivery* dibebaskan)
- *Ticket app* akan memiliki endpoint untuk *client app* dapat secara manual mengecek status dari booking tertentu.

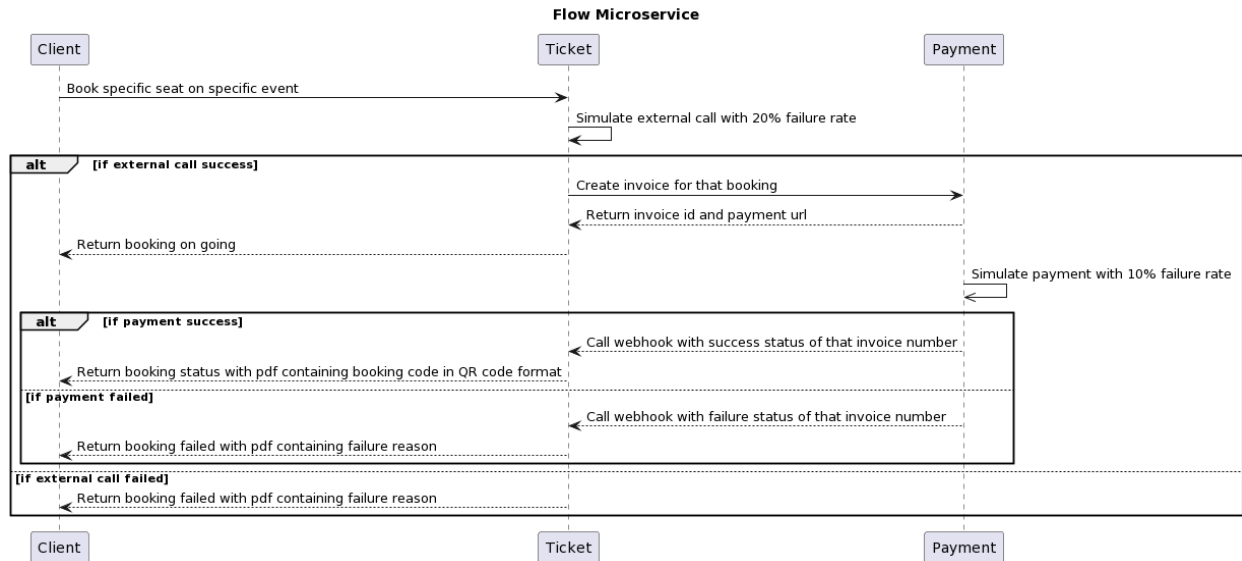
#### Payment App (Service 3):

- *Payment App* harus menyimpan data *invoice* baik yang berhasil maupun tidak secara persisten
- *Payment App* akan memproses pembayaran dengan *rate* kegagalan 10% (simulasikan dengan random)
- Pemrosesan pembayaran ini terjadi secara *asynchronous*, dan apabila prosesnya sudah selesai (berhasil maupun gagal), *Payment App* akan melakukan pemanggilan webhook (*push message*) ke *WEBHOOK\_URL* yang berisikan status keberhasilan dari pembayaran tersebut.
- *Payment App* menyimpan konfigurasi berupa *WEBHOOK\_URL* yang merupakan URL/Endpoint milik *Ticket App*.
- *WEBHOOK\_URL* akan dipanggil setiap proses payment selesai dilaksanakan (baik sukses maupun gagal).

#### Spesifikasi Minimum Perangkat Lunak :

- Ketiga service harus berjalan di dalam docker environment.
- Ketiga service harus memiliki databasenya masing-masing.

Karena Pak Philip berusaha semaksimal mungkin untuk memberikan gambaran teknis kepada anda selagi berusaha menjadi *project manager* yang *baik*. Pak Philip membuatkan sebuah sequence diagram yang menggambarkan interaksi antar *service* sebagai berikut.



### Bonus :

- Anda akan mendapatkan bonus apabila membangun microservice dengan setidaknya 2 bahasa pemrograman yang berbeda.
- Anda akan mendapatkan bonus sangat kecil (maksimum 5 poin) apabila anda membuat tampilan depan.

Karena Pak Philip memiliki pengalaman dalam membangun ticketing system, Pak Philip memberikan konsiderasinya terhadap kemungkinan ticket dibatalkan. Konsiderasi tersebut juga membuka sebuah peluang bagi fitur baru, yaitu reservasi tiket dimana pengguna dapat “mengantri” kursi tertentu yang sudah terisi dengan harapan apabila ada yang membatalkan ticket atas kursi tersebut, maka pengguna yang mengantri terlebih dahulu dapat memperoleh kursi tersebut. Pak Philip memberikan kebebasan pada anda apakah anda hendak menanggulangi konsiderasi Pak Philip, namun Pak Philip memberikan iming-iming promosi dan bonus yang besar apabila anda menanggulangi konsiderasi Pak Philip.



### Deliverables (Deadline: Rabu 29 November 2023 pukul 23.59 WIB) :

1. **Laporan** yang berisi:
  - a. Analisis masalah
  - b. Deskripsi solusi
  - c. Analisis pemilihan arsitektur
  - d. Analisis pemilihan stack teknologi
2. **Repository** github yang berisi:
  - a. Readme
  - b. Postman
  - c. Dockerized file
  - d. Seeds & migration (pastikan script migration schema dan seed DB ada dalam repository termasuk bagaimana menjalankannya)
3. **Video** demo yang berisi (max 10 menit, upload ke youtube):

- Cara menjalankan ketiga service dan dependensinya jika ada
- Penjelasan singkat arsitektur ketiga aplikasi (Stack, DB, dkk)
- Penjelasan singkat fungsionalitas ketiga service & API yang tersedia
- Demo melakukan booking (usahakan ada yang sukses dan gagal)
- Demo melakukan reservasi (jika mengerjakan bonus; akan mendapat toleransi durasi video)

**Karena UI opsional (jangan diutamakan ya), maka demo aplikasi dapat dilakukan dengan postman saja.**

Untuk mempermudah hidup, kita sudah siapkan template laporan dan project yang dapat diakses pada segmen links dibawah ini. Untuk template github dan laporan tidak akan mempengaruhi penilaian selama komponen2 yang intended dalam deliverables dapat ditemukan dengan baik.

Links	
QnA	 QnA
Pengumpulan Tugas	<a href="#">Form</a>
Template Laporan	 IF4031 Template Laporan
Template Github	<a href="#">Github</a>

References	
Knowledge	<a href="#">Push/Pull Message</a> <a href="#">Webhook best practice</a>
Docker	<a href="#">Docker with Node</a> <a href="#">Docker with express</a> <a href="#">Golang with docker github</a> <a href="#">Golang with docker medium</a> <a href="#">Docker volume</a>
Docker Compose	<a href="#">Express with mysql</a> <a href="#">Express with psql</a> <a href="#">Golang with psql</a>
Postman	<a href="#">Get started</a>
System Architecture Design	<a href="#">C4</a>

\*Tidak usah terlalu overthink di system architecture design ya, yang penting ada teknologi2 yang dipakai dan interaksinya bagaimana (kayak diagram bulat2 di template itu cukup)