ZONAsi Jurnal Sistem Informasi
ZONAsi VOL. 8 NO. 2 Mei 2026
Page: 605 - 615 Akreditasi Sinta 4
ISSN: 2656-7407 (Online)
2656-7393 (Print)

PENGEMBANGAN ANTARMUKA PEMROGRAMAN APLIKASI (API) UNTUK SISTEM PENJUALAN TIKET BERBASIS KONTRAK PINTAR

Muhammad Rafi Juliansyah[1], Hustinawaty[2]
[1,2]Universitas Gunadarma
([1,2]Magister Manajemen Sistem Informasi)
(Jl. Margonda Raya 100, Kemiri Muka, Beji, Depok, Jawa Barat, telp. 021-788112)
e-mail: [1]**mrafijuliansyah14@gmail.com**, [2]hustina@staff.gunadarma.ac.id

Abstrak

Sistem penjualan tiket konser daring di Indonesia masih rentan terhadap pemalsuan, penipuan, dan praktik percaloan. Penelitian ini mengembangkan RESTful API yang terintegrasi dengan kontrak pintar pada Hyperledger Fabric untuk meningkatkan keamanan dan transparansi sistem tiket. Menggunakan pendekatan prototyping, sistem dibangun dengan Node.js, Express.js, dan CouchDB, yang dideploy pada tiga peer node (Ticket Operator, Ticket Buyer, Auditor) yang terhubung melalui satu channel. Enam endpoint API berbasis CRUD diimplementasikan sesuai dengan fungsi-fungsi kontrak pintar CreateTicket, ReadTicket, GetAllTickets, UpdateTicket, TransferTicket, dan DeleteTicket. Pengujian fungsional mengkonfirmasi bahwa seluruh endpoint berfungsi dengan benar. Pengujian performa menggunakan Hyperledger Caliper menunjukkan throughput puncak sebesar 298 TPS dengan rata-rata latency 0,23–0,95 detik pada beban sedang. Pada beban tinggi (1000–2000 TPS), throughput turun menjadi 135,8 TPS dan rata-rata latency mencapai 9,52 detik. Sistem ini layak diterapkan untuk distribusi tiket digital terdesentralisasi skala menengah.

Kata kunci*:* *Antarmuka Pemrograman Aplikasi, Kontrak Pintar, Rantai Blok, Sistem Penjualan Tiket, Sistem Terdistribusi.*

Abstract

Online concert ticketing systems in Indonesia remain vulnerable to counterfeiting, fraud, and ticket scalping practices. This study develops a RESTful API integrated with smart contracts on Hyperledger Fabric to enhance the security and transparency of the ticketing system. Using a prototyping approach, the system was built with Node.js, Express.js, and CouchDB, and deployed across three peer nodes (Ticket Operator, Ticket Buyer, and Auditor) connected through a single blockchain channel. Six CRUD-based API endpoints were implemented in accordance with the smart contract functions: CreateTicket, ReadTicket, GetAllTickets, UpdateTicket, TransferTicket, and DeleteTicket. Functional testing confirmed that all endpoints operated correctly. Performance evaluation using Hyperledger Caliper demonstrated a peak throughput of 298 TPS with an average latency of 0.23–0.95 seconds under moderate load. Under high load (1000–2000 TPS), throughput decreased to 135.8 TPS with an average latency of 9.52 seconds. The system is suitable for medium-scale decentralized digital ticket distribution.

Keywords: *application programming interface, smart contract, blockchain, ticketing system, distributed system.*

1. PENDAHULUAN

Perkembangan teknologi informasi telah mengubah berbagai aspek kehidupan, termasuk dalam bidang penjualan tiket konser yang kini beralih dari sistem konvensional berbasis tiket fisik menuju sistem digital berbasis internet. Kemajuan ini memberikan kemudahan akses bagi konsumen, namun sekaligus menimbulkan tantangan serius berupa maraknya praktik pemalsuan, penipuan, dan *scalping* tiket yang merugikan konsumen maupun penyelenggara acara[1].

Kasus konser Coldplay di Indonesia menjadi sorotan nasional sebagai contoh nyata dampak dari kelemahan sistem penjualan tiket digital yang ada saat ini. Antusiasme publik yang tinggi dimanfaatkan oleh pihak tidak bertanggung jawab untuk melakukan penipuan tiket secara masif, mulai dari pemalsuan *barcode* hingga penjualan satu tiket kepada beberapa pembeli sekaligus[2]. Kondisi ini menunjukkan bahwa sistem penjualan tiket konvensional, baik dalam format PDF, kode QR, maupun *barcode*, masih rentan terhadap duplikasi dan pemalsuan digital[3].

Distributed Ledger Technology (DLT) muncul sebagai solusi potensial untuk mengatasi permasalahan tersebut. DLT memungkinkan pencatatan transaksi secara terdesentralisasi, transparan, dan tidak dapat dimanipulasi sehingga setiap tiket yang diterbitkan dapat dilacak dan diverifikasi secara independen tanpa memerlukan perantara terpusat[4]. Hyperledger Fabric, sebagai salah satu platform DLT, menawarkan arsitektur modular dengan alur *execute-order-commit* yang dapat menjamin efisiensi [5].

Berbagai penelitian telah menunjukkan potensi DLT dalam sistem penjualan tiket. Platform DeTi yang dikembangkan oleh [6] membuktikan distribusi tiket terdesentralisasi melalui kontrak pintar. [7] mengintegrasikan *self-sovereign identity* (SSI) untuk memperkuat kontrol kepemilikan di pasar sekunder. [8] membangun sistem *mobile end-to-end* untuk penerbitan dan pelacakan tiket berbasis *blockchain*, yang merupakan salah satu bentuk implementasi DLT. Namun, sebagian besar penelitian tersebut belum membahas secara mendalam pengembangan API sebagai lapisan integrasi yang memungkinkan interoperabilitas sistem penjualan tiket dengan aplikasi eksternal.

Dalam penanganan masalah pemalsuan dan penipuan tiket, terdapat beberapa pendekatan teknologi yang telah digunakan. Pendekatan konvensional berbasis basis data terpusat, seperti sistem manajemen tiket pada umumnya, bergantung pada satu entitas pusat sebagai otoritas tunggal sehingga rentan terhadap manipulasi data dan titik kegagalan tunggal. Pendekatan berbasis NFT (*Non-Fungible Token*) pada *blockchain* publik seperti Ethereum menawarkan kepemilikan tiket yang terverifikasi secara kriptografi, namun memiliki kelemahan berupa throughput rendah akibat mekanisme konsensus *Proof of Work* (PoW) serta biaya transaksi yang tinggi[3]. *DLT* dengan jaringan privat hadir sebagai alternatif yang lebih seimbang: pencatatan transaksi bersifat terdesentralisasi dan tidak dapat dimanipulasi, akses jaringan dikontrol hanya untuk pihak yang berwenang, serta *throughput* jauh lebih tinggi dibandingkan *blockchain* publik[4][10]. Karakteristik ini menjadikan DLT privat sebagai pilihan yang lebih tepat untuk konteks *enterprise* seperti sistem distribusi tiket konser yang membutuhkan keamanan, skalabilitas, dan efisiensi secara bersamaan.

Penelitian ini bertujuan untuk mengisi celah tersebut dengan mengembangkan API berbasis kontrak pintar pada Hyperledger Fabric yang dapat menjadi jembatan antara sistem DLT dan aplikasi pengguna. Tujuan utama penelitian ini adalah: (1) merancang dan mengimplementasikan kontrak pintar untuk mengelola transaksi tiket konser; (2) mengembangkan API menggunakan Node.js dan Express.js sebagai penghubung antara aplikasi klien dan jaringan Hyperledger Fabric; dan (3) mengevaluasi performa sistem berdasarkan parameter *throughput* dan *latency* menggunakan Hyperledger Caliper. Untuk mendukung pemahaman terhadap sistem yang dikembangkan, berikut diuraikan landasan teori dari teknologi-teknologi utama yang digunakan.

Distributed Ledger Technology (DLT) adalah sistem basis data terdistribusi yang mencatat, membagi, dan menyinkronkan transaksi secara terdesentralisasi tanpa memerlukan otoritas pusat. Setiap transaksi dicatat pada semua node yang berpartisipasi, menciptakan catatan yang transparan, aman, dan sulit dimanipulasi[4]. DLT memiliki karakteristik utama berupa desentralisasi, imutabilitas, auditabilitas, *timestamping*, dan non-repudiasi[9]. DLT dapat diklasifikasikan berdasarkan tingkat akses menjadi publik (seperti Bitcoin) dan privat (seperti Hyperledger Fabric). Pada implementasi privat, hanya pihak yang memiliki otorisasi yang dapat berpartisipasi dalam validasi transaksi dan pengelolaan sistem[10]. Klasifikasi ini menjadikan DLT privat sangat relevan untuk aplikasi *enterprise* yang membutuhkan kontrol akses ketat seperti sistem penjualan tiket.

Kontrak pintar adalah program digital yang dijalankan secara otomatis di dalam DLT ketika kondisi yang telah ditentukan terpenuhi, tanpa memerlukan perantara[11]. Dalam Hyperledger Fabric, kontrak pintar diimplementasikan sebagai *chaincode* yang bertanggung jawab mengeksekusi logika bisnis, seperti validasi transaksi dan pengelolaan aset digital[12]. Hyperledger Fabric menggunakan alur transaksi *execute-order-commit* yang membedakannya dari platform DLT publik. Transaksi pertama dieksekusi dalam lingkungan terisolasi untuk menghasilkan *read-write set*, kemudian

diurutkan oleh *ordering service*, dan akhirnya divalidasi serta dicatat secara permanen ke dalam *ledger*[5]. Arsitektur modular ini memungkinkan konfigurasi jaringan yang fleksibel dan mendukung determinisme kontrak pintar[13].

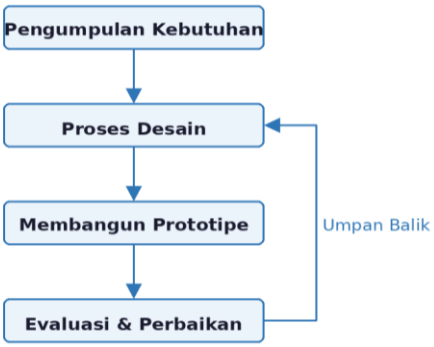
Antarmuka Pemrograman Aplikasi (API) merupakan antarmuka yang memungkinkan satu program berinteraksi dengan aplikasi atau layanan lainnya. Dalam sistem berbasis web, pemanggilan fungsi API dilakukan melalui protokol HTTP dan memberikan respons berupa data dalam format JSON atau XML [14]. Arsitektur API menghubungkan aplikasi *client-side*, *web service* API, dan *database* dalam pola yang memisahkan antarmuka pengguna dari logika *server* untuk meningkatkan efisiensi dan kemudahan integrasi [15]. Node.js sebagai lingkungan *runtime* JavaScript dengan arsitektur *event-driven* dan *non-blocking* I/O menjadi pilihan ideal untuk pengembangan API yang skalabel[16]. Express.js sebagai *framework* yang dibangun di atas Node.js menyederhanakan pengelolaan rute, *middleware*, dan pemrosesan permintaan HTTP, sehingga mempercepat pengembangan layanan API[17]. CouchDB sebagai basis data berbasis dokumen yang mendukung format JSON dan *multi-version concurrency control* (MVCC) menjadi pelengkap ideal untuk menyimpan status data tiket dalam jaringan Hyperledger Fabric[18].

Hyperledger Caliper merupakan alat *benchmarking* resmi yang dikembangkan oleh konsorsium Hyperledger untuk mengukur kinerja platform DLT. Caliper menghasilkan laporan yang mencakup *throughput*, *latency*, dan *send rate* sebagai indikator utama performa sistem[19]. *Throughput* mengukur jumlah transaksi yang berhasil diproses per satuan waktu, sedangkan latency mencerminkan waktu yang dibutuhkan transaksi dari pengiriman hingga konfirmasi. [20]membuktikan efektivitas Caliper dalam mengevaluasi ketahanan sistem DLT terhadap berbagai kondisi beban, termasuk serangan DDoS, dengan merekam variasi *throughput* dan *latency* secara *real-time*.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan *System Development Life Cycle* (SDLC) dengan model *prototyping*. Pendekatan ini dipilih karena pengembangan sistem berbasis DLT dan kontrak pintar memerlukan proses validasi bertahap melalui pengujian berulang dan penyempurnaan sistem secara iteratif. Model *prototyping* memungkinkan pengembangan versi awal sistem yang kemudian dievaluasi untuk memperoleh umpan balik sebelum dilakukan perbaikan lanjutan[21].

Tahapan penelitian meliputi pengumpulan kebutuhan, proses desain, membangun prototipe, serta evaluasi dan pengujian performa sistem. Tahapan tersebut dapat dilihat pada Gambar 1.



Gambar 1. Metode prototyping pengembangan sistem [22]

2.1. Pengumpulan Kebutuhan API Sistem Penjualan Tiket

Tahap pengumpulan kebutuhan dilakukan melalui studi literatur dan observasi terhadap praktik penjualan tiket konser. Analisis difokuskan pada permasalahan umum seperti pemalsuan tiket, duplikasi data, serta kurangnya transparansi dalam distribusi dan perpindahan kepemilikan. Permasalahan tersebut menjadi dasar perancangan sistem berbasis DLT yang menjamin integritas, keterlacakan, dan keabsahan setiap transaksi.

Kebutuhan fungsional sistem mencakup pembuatan tiket oleh penyelenggara, pembacaan informasi tiket secara *real-time*, transfer kepemilikan yang terdokumentasi, pembaruan data, serta

penghapusan tiket sesuai aturan sistem. Seluruh fungsi tersebut harus terintegrasi dengan logika bisnis pada kontrak pintar agar setiap transaksi tercatat secara permanen dalam *ledger* terdistribusi.

Kebutuhan non-fungsional meliputi aspek keamanan, skalabilitas, dan ketersediaan layanan. Sistem dirancang untuk mampu menangani lonjakan transaksi dalam waktu singkat, khususnya saat pembukaan penjualan tiket. Selain itu, sistem harus menjamin autentikasi identitas dan enkripsi komunikasi antar *node* melalui mekanisme kriptografi yang sesuai dengan karakteristik jaringan *permissioned blockchain*.

2.2. Proses Desain API Sistem Penjualan Tiket

Perancangan API dilakukan sebagai lapisan abstraksi antara aplikasi klien dan jaringan DLT berbasis Hyperledger Fabric. API berfungsi menerjemahkan permintaan HTTP menjadi pemanggilan fungsi pada kontrak pintar yang berjalan di *peer node*.

Setiap *endpoint* dirancang selaras dengan fungsi dalam *chaincode*, yaitu CreateTicket, ReadTicket, GetAllTickets, TransferTicket, UpdateTicket, dan DeleteTicket. Pendekatan ini memastikan konsistensi antara layanan yang tersedia di tingkat aplikasi dan logika bisnis pada *blockchain*.

State database menggunakan CouchDB dengan struktur dokumen berbentuk JSON yang memuat atribut utama seperti _id, event, location, datetime, seat, owner, price, dan docType. Struktur ini memungkinkan fleksibilitas pengelolaan data sekaligus menjaga konsistensi format antar entitas tiket. Dengan desain tersebut, API tidak hanya berperan sebagai antarmuka komunikasi, tetapi juga sebagai mekanisme kontrol transaksi yang memastikan kepatuhan terhadap aturan sistem terdesentralisasi.

2.3. Pembangunan Prototipe API Sistem Penjualan Tiket

Pembangunan prototipe diawali dengan konfigurasi identitas kriptografi menggunakan tool cryptogen untuk menghasilkan sertifikat digital dan kunci enkripsi bagi setiap organisasi dan node. Sertifikat ini digunakan untuk autentikasi serta pengamanan komunikasi berbasis TLS.

Jaringan kemudian dijalankan menggunakan Docker melalui konfigurasi docker-compose yang mendefinisikan *peer*, *orderer*, *Certificate Authority*, dan CLI. Setelah jaringan aktif, pembentukan *channel* dilakukan dengan menghasilkan *genesis block* menggunakan configtxgen. *Orderer* dan *peer* selanjutnya bergabung ke channel sehingga jaringan siap memproses transaksi.

Kontrak pintar dikemas, diinstal, disetujui, dan dikomitkan sesuai mekanisme *lifecycle chaincode* pada Hyperledger Fabric. Setelah proses tersebut selesai, *ledger* diinisialisasi dengan data tiket awal. API dikembangkan menggunakan Node.js dan Express.js yang terintegrasi dengan Fabric SDK for Node.js untuk memungkinkan komunikasi langsung dengan jaringan *blockchain*.

2.4. Evaluasi dan Pengujian Sistem API Sistem Penjualan Tiket

Tahap evaluasi dan pengujian dilakukan melalui dua tahapan. Pertama, pengujian fungsional terhadap enam endpoint API yang merepresentasikan seluruh siklus transaksi dalam sistem penjualan tiket: POST /api/tickets (CreateTicket), GET /api/tickets (GetAllTickets), GET /api/tickets/:id (ReadTicket), PUT /api/tickets/:id (UpdateTicket), PUT /api/tickets/:id/transfer (TransferTicket), dan DELETE /api/tickets/:id (DeleteTicket). Setiap endpoint diuji untuk memverifikasi bahwa respons JSON yang diberikan sistem sesuai dengan data yang dikirimkan atau diminta. Kedua, pengujian performa menggunakan Hyperledger Caliper dengan enam skenario beban transaksi yang meningkat secara bertahap: 100, 250, 500, 1000, 1500, dan 2000 TPS. Indikator yang diukur mencakup *throughput*, *send rate*, serta max, min, dan average *latency* untuk menilai kelayakan sistem dalam skenario penggunaan nyata.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Analisis Kebutuhan API Sistem Penjualan Tiket

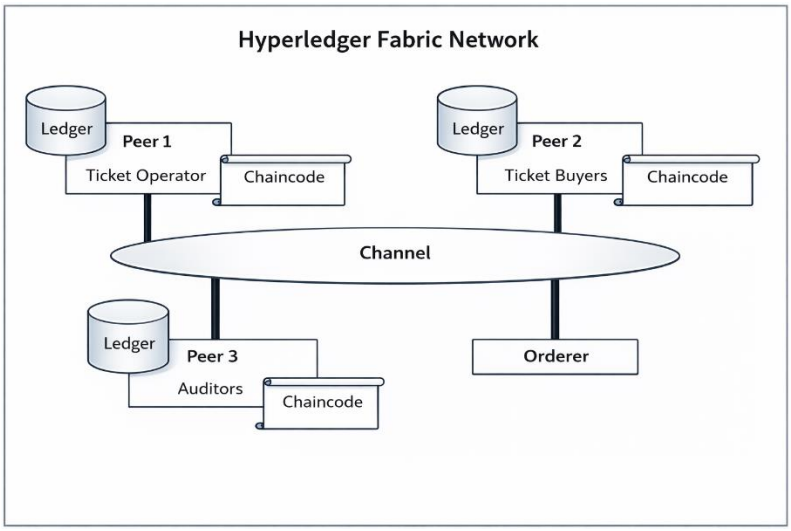
Hasil analisis kebutuhan menghasilkan dua kelompok spesifikasi sistem. Dari sisi fungsional, sistem dirancang untuk menyediakan empat layanan inti: pembuatan tiket oleh pihak penyelenggara konser dengan jaminan keaslian data, pengambilan informasi tiket secara *real-time*, transfer kepemilikan tiket yang aman dan transparan antar pengguna, serta kemampuan menangani lonjakan

beban transaksi dalam waktu singkat saat penjualan tiket dibuka secara serentak. Seluruh fungsi ini dirancang untuk mengatasi permasalahan pemalsuan yang kerap terjadi dalam industri hiburan dengan memastikan setiap transaksi dapat dilacak dan divalidasi melalui jaringan terdesentralisasi.

Dari sisi non-fungsional, sistem harus memenuhi aspek keamanan data melalui mekanisme kriptografi jaringan Fabric, skalabilitas jaringan untuk menangani berbagai tingkat beban transaksi, ketersediaan layanan yang konsisten, serta kompatibilitas dengan spesifikasi perangkat pengembangan yang terdiri dari sistem operasi Ubuntu 24.04 LTS (WSL2), laptop HP ProBook 430 G6, RAM 8192 MB, dan prosesor Intel Core i7-8565U pada 1,80 GHz dengan 8 CPU.

3.2 Hasil Desain API Sistem Penjualan Tiket

Hasil desain menghasilkan arsitektur jaringan Hyperledger Fabric yang terdiri dari tiga *peer node* dengan peran berbeda, satu komponen *orderer*, dan satu *channel* yang menghubungkan semua entitas seperti yang ditunjukkan pada Gambar 2.



Gambar 2. Hasil desain jaringan Fabric sistem penjualan tiket konser

Peer 1 berperan sebagai *Ticket Operator* yang memiliki otorisasi untuk membuat, menerbitkan, dan mengelola tiket melalui kontrak pintar yang terinstal. *Peer* 2 berfungsi sebagai *Ticket Buyer* yang mewakili pengguna akhir dalam melakukan pembelian tiket, di mana setiap transaksi dicatat di *ledger* sebagai bukti pembelian yang tidak dapat diubah. *Peer* 3 menjalankan peran sebagai Auditor yang memverifikasi setiap transaksi dan memantau kepatuhan terhadap aturan sistem. Komponen *Orderer* mengumpulkan transaksi dari *peer*, menyusunnya menjadi blok secara kronologis, lalu mendistribusikannya ke seluruh peer untuk divalidasi sehingga *ledger* setiap peer tetap sinkron dan bersifat *immutable*.

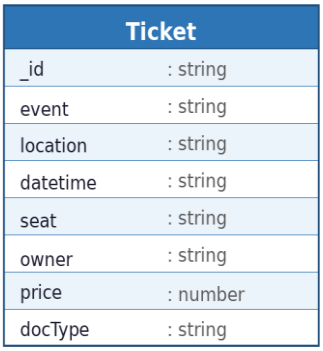
Gambar 3 menunjukkan struktur basis data menggunakan struktur dokumen CouchDB dengan atribut-atribut utama yang merepresentasikan entitas tiket dalam format JSON. Atribut _id berfungsi sebagai identifier unik setiap tiket, sedangkan *docType* digunakan untuk menandai jenis dokumen sebagai "*ticket*" guna memudahkan proses pencarian dan penyaringan data melalui indeks CouchDB. Struktur ini memungkinkan pengelolaan data yang fleksibel karena sifat CouchDB yang *schemaless*, namun tetap menjaga konsistensi data melalui struktur atribut yang sseragam.

Data tiket disimpan di CouchDB dalam format JSON dengan struktur yang telah ditentukan. Setiap dokumen tiket memiliki atribut seperti _id (unik), Event, Location, Datetime, Seat, Owner, dan Price, serta docType yang ditandai sebagai "*ticket*". Struktur ini memudahkan pengindeksan dan pencarian data.

Kontrak pintar (*chaincode*) dirancang dengan fungsi-fungsi utama:

- CreateTicket: Menerbitkan tiket baru ke dalam *ledger*.
- ReadTicket: Mengambil data tiket berdasarkan ID.
- GetAllTickets: Mengambil seluruh data tiket.

- UpdateTicket: Memperbarui seluruh data tiket.
- TransferTicket: Mengubah nilai atribut *Owner* untuk mentransfer kepemilikan tiket.
- DeleteTicket: Menghapus tiket dari *ledger*.



Gambar 3. Struktur *database*

3.3 Hasil Pembangunan Prototipe API Sistem Penjualan Tiket

Tahap pembangunan prototipe merupakan inti dari penelitian ini, karena pada tahap inilah seluruh rancangan sistem yang telah disusun pada fase sebelumnya diimplementasikan menjadi sebuah sistem yang dapat berfungsi secara nyata. Proses dilakukan secara bertahap dan sistematis dengan mengacu pada praktik terbaik pengembangan solusi berbasis Hyperledger Fabric. Tujuan utamanya adalah membangun jaringan DLT yang operasional, mengimplementasikan kontrak pintar sebagai representasi logika bisnis, serta menyediakan antarmuka pemrograman aplikasi (API) untuk menjembatani interaksi antara pengguna dan jaringan. Lingkungan implementasi menggunakan sistem operasi Ubuntu 24.04 LTS sesuai dengan spesifikasi perangkat yang telah ditetapkan pada tahap analisis kebutuhan.

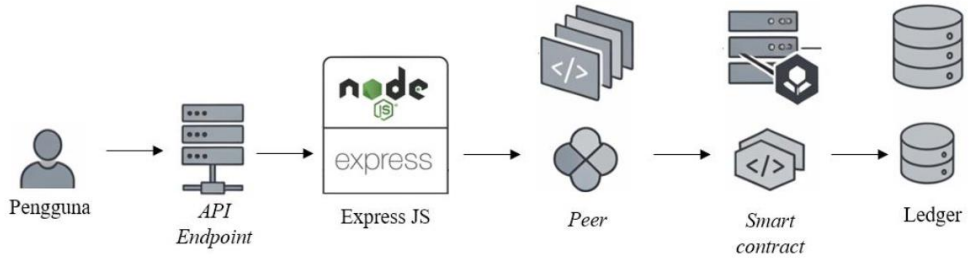
Tahap awal difokuskan pada penyediaan infrastruktur kriptografi sebagai fondasi keamanan jaringan. Setiap entitas dalam jaringan, termasuk organisasi, *peer*, dan *orderer*, dibekali dengan sertifikat digital dan pasangan kunci kriptografi yang berfungsi untuk menjamin autentikasi, integritas, dan kerahasiaan komunikasi. Infrastruktur ini memastikan bahwa hanya pihak yang memiliki kredensial sah yang dapat berpartisipasi dalam jaringan, sehingga membentuk lingkungan yang terkontrol dan terpercaya.

Setelah material kriptografi tersedia, jaringan DLT diinisialisasi menggunakan pendekatan berbasis kontainer. Seluruh komponen utama—*peer node* yang merepresentasikan masing-masing organisasi, *orderer* sebagai pengelola konsensus, *Certificate Authority* (CA), serta antarmuka baris perintah—dijalankan secara terisolasi namun saling terhubung dalam satu lingkungan jaringan. Arsitektur ini memungkinkan simulasi ekosistem DLT yang menyerupai kondisi operasional nyata, dengan pemisahan peran dan tanggung jawab antar entitas.

Tahap berikutnya adalah pembentukan *channel* sebagai mekanisme komunikasi privat antaranggota jaringan. *Channel* berfungsi sebagai ruang transaksi terisolasi yang hanya dapat diakses oleh organisasi yang tergabung di dalamnya. Proses ini diawali dengan pembentukan blok genesis (*genesis block*) yang memuat konfigurasi dasar *channel*, termasuk daftar anggota dan kebijakan konsensus. Setelah *channel* terbentuk, seluruh *peer* dan *orderer* yang berwenang dihubungkan ke dalamnya sehingga dapat memproses dan memvalidasi transaksi secara kolektif.

Implementasi logika bisnis dilakukan melalui instalasi dan aktivasi kontrak pintar (*chaincode*). Kontrak pintar dikembangkan menggunakan bahasa pemrograman yang didukung oleh platform dan memuat fungsi-fungsi utama seperti pembuatan tiket, pembacaan data tiket, serta transfer kepemilikan. Proses aktivasi dilakukan melalui mekanisme persetujuan antarorganisasi sesuai dengan kebijakan *endorsement* yang ditetapkan. Pendekatan ini mencerminkan prinsip tata kelola terdistribusi, di mana perubahan pada logika bisnis harus memperoleh persetujuan dari pihak-pihak yang berkepentingan sebelum dapat dijalankan dalam jaringan. Setelah dikomit ke dalam *channel*, kontrak pintar resmi aktif dan siap menangani transaksi. Untuk keperluan pengujian awal, *ledger* diinisialisasi dengan sejumlah data sampel.

Tahap akhir adalah pembangunan lapisan aplikasi berupa Antarmuka Pemrograman Aplikasi (API) yang dikembangkan menggunakan lingkungan Node.js dengan kerangka kerja Express.js. API ini berfungsi sebagai abstraksi terhadap kompleksitas jaringan *blockchain* dan menyediakan endpoint RESTful untuk operasi utama sistem, seperti pembuatan tiket, pengambilan data, transfer kepemilikan, dan penghapusan tiket.



Gambar 4. Integrasi aplikasi klien

Setiap permintaan yang diterima oleh *server* API diproses melalui Fabric SDK untuk Node JS yang bertanggung jawab membangun koneksi ke jaringan, melakukan autentikasi berbasis sertifikat, serta mengirimkan proposal transaksi ke *peer* yang relevan seperti yang ditampilkan pada Gambar 4. Setelah transaksi divalidasi dan dikomit oleh jaringan, hasilnya dikembalikan dalam format JSON sebagai respons HTTP kepada klien. Dengan arsitektur ini, interaksi pengguna terhadap sistem dapat dilakukan secara sederhana melalui protokol web standar, tanpa memerlukan pemahaman terhadap mekanisme internal DLT seperti konsensus, *endorsement*, maupun pengelolaan node.

Secara keseluruhan, tahap implementasi ini menghasilkan prototipe sistem penjualan tiket berbasis DLT yang aman, transparan, dan terkelola, sekaligus merepresentasikan integrasi antara lapisan infrastruktur *blockchain* dan lapisan aplikasi secara menyeluruh.

3.4 Hasil Evaluasi dan Pengujian Sistem API Sistem Penjualan Tiket

Pengujian fungsional terhadap seluruh *endpoint* API pada Tabel 1 menunjukkan bahwa semua operasi berjalan sesuai dengan logika bisnis yang diimplementasikan dalam kontrak pintar. *Endpoint* POST /api/tickets berhasil membuat tiket baru dan menyimpannya ke dalam *ledger* melalui fungsi CreateTicket menggunakan putState(), dengan respon JSON yang memuat seluruh atribut tiket yang dikirimkan. *Endpoint* GET /api/tickets berhasil mengambil seluruh daftar tiket menggunakan getStateByRange() pada *chaincode* dan mengembalikan *array* JSON yang memuat semua tiket tersimpan. *Endpoint* GET /api/tickets/:id berhasil membaca data tiket berdasarkan ID tertentu menggunakan getState(), menampilkan detail lengkap tiket yang diminta. *Endpoint* PUT /api/tickets/:id berhasil memperbarui seluruh data tiket menggunakan putState() untuk mengganti data lama dengan data baru. *Endpoint* PUT /api/tickets/:id/transfer berhasil mengalihkan kepemilikan tiket dengan merespons informasi pemilik lama dan pemilik baru melalui fungsi TransferTicket. *Endpoint* DELETE /api/tickets/:id berhasil menghapus tiket dari *ledger* menggunakan deleteState() dengan respon berupa objek *deleted* yang merepresentasikan tiket yang telah dihapus.

Tabel 1. Tabel Hasil Pengujian *Endpoint*

Endpoint	Fungsi Kontrak Pintar	Metode HTTP	Kode Respons	Status
POST /api/tickets	CreateTicket	POST	201 Created	Berhasil
GET /api/tickets	GetAllTickets	GET	200 OK	Berhasil
GET /api/tickets/:id	ReadTicket	GET	200 OK	Berhasil
PUT /api/tickets/:id	UpdateTicket	PUT	200 OK	Berhasil
PUT /api/tickets/:id/transfer	TransferTicket	PUT	200 OK	Berhasil

Endpoint	Fungsi Kontrak Pintar	Metode HTTP	Kode Respons	Status
DELETE /api/tickets/:id	DeleteTicket	DELETE	200 OK	Berhasil

Pengujian performa dilakukan pada enam skenario beban transaksi yang meningkat secara bertahap dari 100 hingga 2000 TPS. Tabel 2 menyajikan hasil pengukuran *throughput*, *send rate*, serta max, min, dan rata-rata *latency* pada setiap skenario.

Tabel 2. Tabel Hasil Pengujian Performa

Transaction Load	*Throughput (TPS)*	*Send Rate (TPS)*	*Max Latency (s)*	*Min Latency (s)*	*Average Latency (s)*
100	292.4	292.5	0.47	0.01	0.23
250	298.1	298.3	0.91	0.01	0.53
500	288.9	289.0	0.92	0.01	0.95
1000	249.4	249.4	3.92	0.01	1.46
1500	272.8	272.9	8.06	0.01	4.73
2000	135.8	135.7	15.64	0.01	9.52

Pada beban rendah hingga sedang (100–500 TPS), sistem mencapai *throughput* di atas 288 TPS dengan *latency* rata-rata di bawah 1 detik, menunjukkan efisiensi pemrosesan yang tinggi. *Throughput* mencapai puncaknya pada beban 250 TPS dengan nilai 298,1 TPS. Penurunan performa mulai terlihat signifikan pada beban 1000 TPS ke atas, di mana *latency* rata-rata melonjak dari 1,46 detik hingga 9,52 detik dan *latency* maksimum mencapai 15,64 detik pada beban 2000 TPS. Nilai *latency* minimum yang konsisten sebesar 0,01 detik di seluruh skenario menunjukkan bahwa jaringan masih memiliki jalur pemrosesan yang sangat efisien meskipun sistem berada di bawah tekanan tinggi. Secara keseluruhan, hasil pengujian mengkonfirmasi bahwa sistem layak diimplementasikan pada skala menengah dengan kapasitas hingga 500 TPS, dan memerlukan optimasi infrastruktur lebih lanjut untuk mendukung skenario beban sangat tinggi.

Berdasarkan hasil pengujian keseluruhan, sistem yang dikembangkan menunjukkan kelayakan implementasi untuk skala menengah. Pada rentang beban 100–500 TPS, sistem beroperasi dengan *throughput* di atas 288 TPS dan *latency* rata-rata di bawah 1 detik, yang merupakan performa yang sangat baik untuk skenario distribusi tiket konser digital. Pada skenario penjualan tiket konser menengah dengan jumlah kursi 5.000–10.000, sistem ini mampu menangani antrian pembelian secara efisien.

Keterbatasan sistem mulai terlihat pada beban di atas 1000 TPS, di mana penurunan *throughput* dan lonjakan *latency* yang signifikan dapat mengganggu pengalaman pengguna. Hal ini disebabkan oleh keterbatasan sumber daya komputasi *hardware* yang digunakan (RAM 8 GB, Intel Core i7-8565U) untuk menjalankan tiga *peer node* dan satu orderer secara bersamaan. Implementasi pada infrastruktur server produksi dengan spesifikasi lebih tinggi dan konfigurasi *peer* tambahan diperkirakan akan meningkatkan kapasitas sistem secara signifikan.

Dibandingkan dengan penelitian sebelumnya, platform DeTi [6] menggunakan Ethereum yang memiliki throughput jauh lebih rendah akibat mekanisme konsensus *Proof of Work* (PoW). Sistem Hyperledger Fabric yang dikembangkan dalam penelitian ini menawarkan throughput yang lebih tinggi dan konsumsi energi yang lebih efisien berkat konsensus Raft, menjadikannya pilihan yang lebih praktis untuk implementasi enterprise sistem penjualan tiket.

3.5. Pembahasan

Hasil pengujian menunjukkan bahwa arsitektur RESTful API berbasis Hyperledger Fabric mampu memenuhi kebutuhan sistem distribusi tiket digital pada skala menengah. Penggunaan CouchDB sebagai *state database* terbukti efektif dalam mendukung operasi kueri kompleks seperti GetAllTickets, yang membutuhkan pencarian lintas dokumen secara efisien. Sementara itu, mekanisme *endorsement* pada Hyperledger Fabric memastikan setiap transaksi divalidasi oleh minimal dua *peer* sebelum dikomit ke *ledger*, sehingga integritas data tiket terjaga secara konsisten. Temuan ini

memperkuat relevansi pendekatan DLT privat dibandingkan DLT publik untuk konteks *enterprise* yang membutuhkan kontrol akses terstruktur.

Penurunan performa pada beban tinggi (di atas 1000 TPS) tidak semata-mata disebabkan oleh keterbatasan perangkat keras, tetapi juga berkaitan dengan mekanisme konsensus Raft yang mengharuskan *orderer* memproses setiap transaksi secara sekuensial sebelum mendistribusikannya ke *peer*. Studi [13] mengonfirmasi bahwa konfigurasi ukuran blok dan interval waktu pengiriman blok pada Hyperledger Fabric berdampak signifikan terhadap *throughput* dan *latency*. Optimasi parameter-parameter tersebut, dikombinasikan dengan peningkatan kapasitas infrastruktur, merupakan langkah yang perlu dilakukan untuk mendukung skenario beban sangat tinggi di masa mendatang.

Dibandingkan dengan penelitian sebelumnya, platform DeTi [6] menggunakan Ethereum yang memiliki throughput jauh lebih rendah akibat mekanisme konsensus *Proof of Work*. Sistem Hyperledger Fabric yang dikembangkan dalam penelitian ini menawarkan throughput yang lebih tinggi dan konsumsi energi yang lebih efisien berkat konsensus Raft, menjadikannya pilihan yang lebih praktis untuk implementasi enterprise sistem penjualan tiket.

4. Kesimpulan

Penelitian ini berhasil mengembangkan sistem penjualan tiket konser berbasis DLT dengan tiga kontribusi utama. Pertama, kontrak pintar pada Hyperledger Fabric berhasil diimplementasikan dengan tujuh fungsi operasional (InitLedger, CreateTicket, ReadTicket, UpdateTicket, DeleteTicket, TransferTicket, GetAllTickets) yang mampu mengelola seluruh siklus transaksi tiket secara transparan, aman, dan terdesentralisasi.

Kedua, API berbasis Node.js dan Express.js berhasil dikembangkan sebagai jembatan integrasi antara aplikasi klien dan jaringan Hyperledger Fabric, dengan enam endpoint RESTful yang telah diuji secara fungsional dan terbukti beroperasi sesuai spesifikasi. Antarmuka ini memungkinkan pengguna berinteraksi dengan sistem DLT tanpa perlu memahami kompleksitas internal Fabric.

Ketiga, pengujian performa menggunakan Hyperledger Caliper membuktikan bahwa sistem mampu mencapai *throughput* optimal 298,1 TPS dengan rata-rata *latency* 0,53 detik pada beban sedang, menjadikannya layak untuk implementasi skala menengah dalam distribusi tiket konser digital. Keterbatasan performa pada beban tinggi (di atas 1000 TPS) mengindikasikan perlunya optimasi infrastruktur untuk mendukung implementasi skala besar di masa depan.

Untuk pengembangan lebih lanjut, disarankan untuk mengoptimalkan konfigurasi Hyperledger Fabric (parameter ukuran blok, jumlah peer, dan sumber daya komputasi), mengintegrasikan teknologi *Self-Sovereign Identity* (SSI) untuk memperkuat autentikasi pengguna, mengembangkan fitur pasar sekunder tiket berbasis kontrak pintar, serta melakukan uji coba dalam lingkungan produksi nyata yang terintegrasi dengan sistem pembayaran digital.

Daftar Pustaka

[1] G. S. Nabila and R. B. Setianingrum, "Studi Komparatif Perlindungan Hukum dan Pengaturan Scalping Tiket Konser K-Pop di Indonesia dan Korea Selatan," *Indones. J. Crim. Law Criminol.*, vol. 6, no. 3, pp. 131–138, 2025, doi: 10.18196/ijclc.v6i3.27330.

[2] H. Sinaga, "Kasus Jasa Titip (Jastip) Tiket Coldplay Secara Online Dalam Perspektif Hukum Pidana," *Innov. J. Soc. Sci. Res.*, vol. 3, no. 1, pp. 755–768, 2023.

[3] A. Ahmad, F. Ilmawa, and A. Amrulloh, "Implementasi Penjualan Tiket Event Musik Berbasis NFT Menggunakan Blockchain," *J. Smart Syst.*, vol. 3, no. 2, 2024.

[4] M. Gorbunova, P. Masek, M. Komarov, and A. Ometov, "Distributed Ledger Technology: State-of-the-Art and Current Challenges," *Comput. Sci. Inf. Syst.*, vol. 19, no. 1, pp. 65–85, Jan. 2022, doi: 10.2298/CSIS210215037G.

[5] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling hyperledger fabric to 20 000 transactions per second," *Int. J. Netw. Manag.*, vol. 30, no. 5, Sep. 2020, doi: 10.1002/nem.2099.

[6] S. Rafati Niya, S. Bachmann, C. Brasser, M. Bucher, N. Spielmann, and B. Stiller, "DeTi: A Decentralized Ticketing Management Platform," *J. Netw. Syst. Manag.*, vol. 30, no. 4, Oct. 2022, doi: 10.1007/s10922-022-09675-3.

[7] S. Feulner, J. Sedlmeir, V. Schlatt, and N. Urbach, "Exploring the use of self-sovereign identity for event ticketing systems," *Electron. Mark.*, vol. 32, no. 3, pp. 1759–1777, Sep. 2022, doi:

10.1007/s12525-022-00573-9.
[8] H. Q. Nguyen, H. T. Nguyen, and T. T. Pham, "Applying Smart Contract in Blockchain Technology to Manage the Ticketing Issuance and Ticketing Traceability," in *2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 160–164. doi: 10.1109/JEEIT58638.2023.10185771.
[9] M. Sachy, R. Spigolon, and S. Nonneman, "Three properties of distributed ledger technology systems applied in the nuclear sector adding value to safeguards – immutability, timestamping and auditability," *ESARDA Bull.*, vol. 2021, no. 62, pp. 60–79, Jun. 2021, doi: 10.3011/ESARDA.IJNSNP.2021.6.
[10] V. Garcia-Font, "Conceptual technological framework for smart cities to move towards decentralized and user-centric architectures using DLT," *Smart Cities*, vol. 4, no. 2, pp. 728–745, Jun. 2021, doi: 10.3390/smartcities4020037.
[11] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart Contract: Attacks and Protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020, doi: 10.1109/ACCESS.2020.2970495.
[12] S. Mohan M and L. Sujihelen, "An efficient chain code for access control in hyper ledger fabric healthcare system," *e-Prime - Adv. Electr. Eng. Electron. Energy*, vol. 5, Sep. 2023, doi: 10.1016/j.prime.2023.100204.
[13] T. Guggenberger, J. Sedlmeir, G. Fridgen, and A. Luckow, "An in-depth investigation of the performance characteristics of Hyperledger Fabric," *Comput. Ind. Eng.*, vol. 173, Nov. 2022, doi: 10.1016/j.cie.2022.108716.
[14] Hasanuddin, H. Asgar, and B. Hartono, "RANCANG BANGUNG REST API APLIKASI WESHARE SEBAGAI UPAYA MEMPERMUDAH PELAYANAN DONASI KEMANUSIAAN," *JINTEKS (Jurnal Inform. dan Teknol. dan Sains)*, vol. 4, pp. 8–14, Feb. 2022.
[15] Rangga Gelar Guntara and V. Azkarin, "Implementasi dan Pengujian REST API Sistem Reservasi Ruang Rapat dengan Metode Black Box Testing," *J. Minfo Polgan*, vol. 12, no. 1, pp. 1229–1238, Jul. 2023, doi: 10.33395/jmp.v12i1.12691.
[16] A. Ramadhani, N. Iriadi, and R. Hidayat, "Implementasi Teknologi Rest API Dengan Node Js Untuk Aplikasi Rekomendasi Destinasi Wisata," *J. Comput. Sci.*, vol. 4, no. 1, 2025.
[17] D. H. Kostrzewa and M. Miłosz, "Comparative analysis of the Express.js and ElysiaJS frameworks in the context of web application development," *J. Comput. Sci. Inst.*, vol. 32, pp. 246–250, 2024.
[18] I. Carvalho, F. Sá, and J. Bernardino, "NoSQL Document Databases Assessment: Couchbase, CouchDB, and MongoDB," in *Proceedings ofthe 11th International Conference on Data Science, Technology and Applications (DATA 2022)*, Scitepress, Jul. 2022, pp. 557–564. doi: 10.5220/0011352700003269.
[19] C. Melo *et al.*, "Performance and availability evaluation of the blockchain platform hyperledger fabric," *J. Supercomput.*, vol. 78, no. 10, pp. 12505–12527, Jul. 2022, doi: 10.1007/s11227-022-04361-2.
[20] V. Jayadev, N. Moradpoor, and A. Petrovski, "Assessing the Performance of Ethereum and Hyperledger Fabric Under DDoS Attacks for Cyber-Physical Systems," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jul. 2024. doi: 10.1145/3664476.3670927.
[21] E. Putra Pane and F. Juliani, "MANAJEMEN SISTEM DAN MONITORING PENJUALAN UMKM PADA OUTLET SOMAY BATAGOR CAKRA BUANA," *Zo. J. Sist. Inf.*, vol. 7, no. 3, pp. 946–955, Sep. 2025.
[22] N. Andini, R. Taufiq, D. Y. Priyanggodo, and Y. Sugiyani, "PENGGUNAAN METODE PROTOTYPE PADA PENGEMBANGAN SISTEM INFORMASI IMUNISASI POSYANDU," *JIKA (Jurnal Inform.*, vol. 7, no. 4, p. 431, Nov. 2023, doi: 10.31000/jika.v7i4.9329.