Vrije Universiteit Amsterdam

Universitas Amsterdam

Tesis Magister

Mengoptimalkan Sistem Pembayaran dengan Arsitektur Mikroservis dan Berbasis Peristiwa: Studi Kasus Mollie Platform

---

Penulis: Yufei Wang

VU: 2756993

UvA: 14170884

Pengawas pertama: Adam SZ Belloum

Pengawas harian: John Zhao (Mollie BV)

Pembaca kedua: Shashikant Ilager

Tesis yang diajukan sebagai pemenuhan persyaratan untuk

Program Magister Sains gabungan UvA-VU di bidang Ilmu Komputer

4 Desember 2024

“Akulah penguasa takdirku, akulah kapten jiwaku”

dari Invictus, karya William Ernest Henley

Abstrak

Pertumbuhan eksponensial dalam transaksi digital telah menciptakan permintaan yang mendesak.
untuk platform pembayaran yang terukur, andal, dan real-time. Tesis ini mengeksplorasi
transisi dari arsitektur monolitik ke arsitektur berbasis layanan mikro dan berbasis peristiwa

Penelitian ini mengusulkan pendekatan modular yang menggunakan arsitektur berbasis Mollie, sebuah platform pembayaran
frekuensi tinggi. Dengan menganalisis tantangan skalabilitas, toleransi kesalahan, dan kinerja dalam sistem monolitik tradisional,
penelitian ini menerapkan arsitektur berbasis arsitektur modular yang menggunakan Mollie, sebuah platform pembayaran frekuensi tinggi.
Layanan mikro dan komunikasi asinkron dengan arsitektur berbasis peristiwa.
Arsitektur ini dirancang untuk meningkatkan efisiensi operasional, memastikan waktu nyata.
memproses, dan beradaptasi dengan meningkatnya permintaan akan beragam metode pembayaran.
Alat seperti DataDog dan GCP digunakan untuk pemantauan dan penerapan secara real-time,
Menunjukkan peningkatan keandalan dan kemudahan perawatan. Terlepas dari kemajuan yang ada,
tantangan seperti duplikasi acara, ketergantungan API eksternal, dan sumber daya
Biaya overhead juga dibahas. Temuan ini memberikan wawasan berharga tentang arsitektur yang
terukur dan mudah beradaptasi untuk platform pembayaran digital.

Kata kunci— Platform Pembayaran, Arsitektur Layanan Mikro, Arsitektur Berbasis
Peristiwa (EDA), Pemantauan Waktu Nyata, Skalabilitas, Toleransi Kesalahan

Isi

Daftar Gambar

DAFTAR GAMBAR

Daftar tabel

DAFTAR TABEL

1

Perkenalan

Seiring perkembangan digital yang membentuk kembali lanskap keuangan, hal ini telah berubah secara signifikan.
cara pembayaran diproses, yang menyebabkan peningkatan eksponensial dalam variasi dan volumenya.
berbagai pilihan pembayaran di seluruh dunia. Konsumen saat ini menuntut metode pembayaran yang cepat, aman, dan fleksibel, sementara pedagang mencari solusi yang andal dan mampu beradaptasi dengan dinamika pasar yang terus berkembang. Penyedia Layanan Pembayaran (PSP), seperti Mollie(1), memainkan peran penting dalam menjembatani kesenjangan ini, memastikan pengalaman transaksi yang lancar bagi pedagang dan konsumen. Namun, mengintegrasikan beragam metode pembayaran ke dalam satu sistem terpadu
Platform ini menghadirkan tantangan kritis dalam hal skalabilitas, efisiensi, dan kinerja waktu nyata (2). Sebagai perusahaan pembayaran multi-pasar, Mollie berupaya untuk memenuhi tujuan ini. Melayani pedagang di setiap industri dan negara di Eropa, Mollie menyederhanakan pemrosesan pembayaran dengan memungkinkan bisnis untuk menerima berbagai metode pembayaran, termasuk kartu kredit, debit langsung, PayPal, dan opsi lokal yang disesuaikan dengan pasar tertentu. Namun, tantangan ada di samping peluang. Seiring platform berkembang untuk mengintegrasikan metode pembayaran baru dan
Untuk menangani peningkatan volume transaksi, sistem tersebut harus mengatasi hambatan yang terkait dengan kinerja, keandalan, dan fleksibilitas.

Awalnya berbasis pada arsitektur monolitik, platform Mollie menunjukkan beberapa keterbatasan, seperti komponen yang saling terkait erat, hambatan skalabilitas, dan peningkatan risiko kegagalan.
Sistem monolitik mengalami masalah kinerja seiring pertumbuhannya, kurangnya modularitas, dan membuat pembaruan kecil sekalipun menjadi rumit.
Untuk bidang yang berkembang pesat seperti teknologi keuangan, di mana kelincahan dan kemampuan beradaptasi sangat penting, keterbatasan ini tidak berkelanjutan. Setiap fitur baru atau integrasi pembayaran membutuhkan pengujian dan sumber daya pengembangan yang ekstensif, bahkan berisiko menyebabkan gangguan. Selain itu, arsitektur monolitik Sistem menghadapi tantangan dalam menjaga kepatuhan terhadap peraturan keuangan yang dinamis dan memastikan manajemen data yang aman seiring dengan peningkatan skala transaksi secara global.

Untuk mengatasi tantangan ini, kami memutuskan untuk mengadopsi arsitektur berbasis microservices.
Memisahkan fungsi, meningkatkan toleransi kesalahan, dan memungkinkan praktik pengembangan yang lebih lincah.
Praktik-praktik tersebut. Microservices membagi platform menjadi unit-unit yang lebih kecil dan otonom, yang masing-masing bertanggung jawab.
untuk fungsi tertentu, memungkinkan siklus pengembangan yang lebih cepat, mengurangi waktu henti, dan kesalahan.
isolasi. Selain itu, arsitektur berbasis peristiwa (event-driven architecture/EDA) telah diperkenalkan untuk mengoptimalkan
pemrosesan data waktu nyata dan memastikan responsivitas di seluruh sistem terdistribusinya.
Secara bersama-sama, perubahan arsitektur ini bertujuan untuk meningkatkan pertumbuhan Mollie di masa depan, memungkinkan
untuk menangani peningkatan volume transaksi sekaligus mengintegrasikan berbagai metode pembayaran.
lebih cepat dan lancar.

Tesis ini berfokus pada pertanyaan-pertanyaan penelitian berikut:

- Bagaimana arsitektur microservices dapat diterapkan secara efektif untuk meningkatkan skalabilitas?
  dan toleransi kesalahan pada platform pembayaran?
- Apa peran arsitektur berbasis peristiwa dalam meningkatkan pemrosesan waktu nyata dan
  Pemisahan (decoupling) dalam sistem pembayaran?
- Bagaimana pipeline pemantauan dan penerapan dapat diimplementasikan untuk memastikan sistem
  keandalan?

Pertanyaan-pertanyaan ini sangat penting untuk mengatasi tantangan-tantangan yang dihadapi oleh platform pembayaran
Mollie, seperti kebutuhan akan skalabilitas untuk menangani volume transaksi yang terus meningkat.
terhadap kesalahan untuk menjaga layanan tetap berjalan tanpa gangguan, dan manajemen kesalahan yang efisien untuk
memastikan pengalaman pengguna yang lancar. Selain itu, masalah-masalah ini tidak hanya terjadi pada Mollie tetapi juga
Hal ini mewakili tantangan yang lebih luas yang dihadapi oleh PSP dan transaksi frekuensi tinggi lainnya.
sistem di berbagai industri. Dengan menggali pertanyaan-pertanyaan ini, penelitian ini tidak hanya memberikan solusi yang
disesuaikan untuk Mollie tetapi juga mengembangkan metodologi generik yang dapat diterapkan.
untuk masalah skala besar yang melibatkan platform pemrosesan transaksi yang terukur dan real-time.
Mengatasi tantangan-tantangan ini akan memperluas cakupan pertumbuhan Mollie secara lebih luas, serta menawarkan wawasan baru.
dalam merancang sistem yang terukur, tahan terhadap kesalahan, dan efisien di lingkungan yang dinamis.

Tesis ini mengeksplorasi transisi dari arsitektur monolitik ke arsitektur layanan mikro dengan
mekanisme berbasis peristiwa, menggunakan platform pembayaran Mollie sebagai studi kasus. Penelitian ini
Menganalisis bagaimana perubahan arsitektur dapat meningkatkan skalabilitas, toleransi kesalahan, dan kinerja dalam lingkungan
transaksi frekuensi tinggi. Dengan menerapkan alat seperti Apache Kafka.
untuk event streaming dan DataDog (3) untuk pemantauan waktu nyata, penelitian ini bertujuan untuk mendemonstrasikan
metodologi praktis untuk mencapai platform yang andal, efisien, dan adaptif.

Namun, mendesain dan mengembangkan sistem yang skalabel seperti itu menimbulkan tantangan yang signifikan, termasuk memastikan toleransi kesalahan, menjaga integritas data, dan menangani transaksi secara real-time. pemrosesan dalam lingkungan terdistribusi. Tesis ini pertama-tama membahas tantangan-tantangan ini dan teknologi-teknologi baru yang dapat dimanfaatkan untuk mengatasi tantangan-tantangan tersebut, dan kemudian memberikan solusi dan peningkatan yang lebih baik untuk platform pembayaran Mollie yang sudah ada. Yang utama
Tujuan dari tesis ini adalah untuk mencapai hal-hal berikut:

- Skalabilitas: Tesis ini bertujuan untuk menggunakan kerangka kerja layanan mikro yang memisahkan layanan pembayaran inti, memungkinkan layanan individual—seperti pemrosesan transaksi dan pemberitahuan—untuk berkembang secara independen. Penyempurnaan ini meningkatkan kemampuan sistem.
  untuk memenuhi tuntutan transaksi frekuensi tinggi dan memberikan fleksibilitas untuk pertumbuhan di masa depan.

- Penerapan (Deployment): Penerapan selalu menjadi masalah yang rumit bagi perusahaan besar dengan basis kode yang besar. Pipeline ini memakan banyak sumber daya dan dapat berisiko, terutama di lingkungan dengan pembaruan yang sering. Tesis ini membahas tantangan penerapan dengan
  menerapkan orkestrasi, memastikan setiap microservice dapat di-deploy dan diperbarui secara independen. Dengan memanfaatkan alat seperti Docker dan Kubernetes, kerangka kerja ini mendukung integrasi dan deployment berkelanjutan (CI/CD), mengurangi downtime dan memungkinkan iterasi cepat (4).

- Keamanan: Seiring pertumbuhan data yang eksponensial, memastikan keamanan data sangat penting dalam sektor keuangan. teknologi. Tesis ini mengintegrasikan protokol keamanan berlapis untuk melindungi data pengguna, dengan fokus pada pemenuhan standar seperti GDPR(5), dan persyaratan khusus lainnya dari mitra metode pembayaran dari berbagai negara. Dengan membangun kepatuhan dan
  Dengan menerapkan privasi data ke dalam arsitektur terdesentralisasi, platform ini lebih baik dalam melindungi data. data yang diperoleh oleh pihak-pihak yang berniat jahat.

- Toleransi Kesalahan: Toleransi kesalahan sangat penting untuk menjaga layanan yang tidak terputus dalam sistem terdistribusi. Insiden yang tidak terduga dapat secara dramatis merusak pengalaman pengguna dan kepercayaan pelanggan. Tesis ini memanfaatkan prinsip-prinsip arsitektur berbasis peristiwa (EDA) (6) dan Apache Kafka untuk komunikasi asinkron dan redistribusi beban. Dengan menerapkan pemantauan waktu nyata, sistem dapat mendeteksi kegagalan halus sebelum pengguna menyadarinya. Selain itu, tujuannya adalah untuk mengelola kegagalan layanan tanpa

  mengganggu operasi inti, memastikan bahwa pemrosesan pembayaran tetap tangguh dalam kondisi tersebut. gangguan beban atau layanan sebagian.

• Kinerja: Pemrosesan transaksi secara real-time memerlukan kinerja yang dioptimalkan.
di semua layanan. Meskipun sumber daya komputasi dapat menjadi faktor penentu,
Efisiensi perangkat lunak juga memainkan peran penting di sini. Tesis ini berfokus pada pengurangan...
Mengurangi latensi dan meningkatkan throughput melalui streaming data yang efisien dan pemantauan aktif sesuai dengan arsitektur microservice. Optimasi tersebut memungkinkan pengalaman pengguna yang responsif, bahkan di bawah volume transaksi puncak, sambil memastikan data diproses dengan cepat dan efisien.

• Kemudahan Pemeliharaan & Perluasan: Seiring dengan terus berkembangnya ekosistem pembayaran,
Semakin banyak penyedia layanan pembayaran dan pelanggan yang menimbulkan tuntutan yang lebih tinggi. Dengan demikian,
Platform harus mampu beradaptasi dengan teknologi baru dan kebutuhan bisnis. Dalam hal ini,
Dalam makalah ini, kami menekankan kemudahan pemeliharaan melalui desain modular dan dokumentasi yang jelas.
dan pengujian otomatis. Selain itu, dengan membuat setiap layanan bersifat modular, sistem ini dapat dengan mudah mengintegrasikan gerbang pembayaran tambahan, alat analitik, atau fitur kepatuhan, sehingga mendukung inovasi dan evolusi yang berkelanjutan.

Cakupan penelitian ini meliputi analisis komprehensif mengenai pergeseran arsitektur,
Tesis ini berfokus pada aspek-aspek kunci seperti desain microservice modular, strategi komunikasi berbasis peristiwa, dan integrasi pemantauan dan pipeline deployment. Tesis ini juga membahas tantangan seperti pengelolaan sistem terdistribusi dan menjaga konsistensi data di seluruh layanan. Tesis ini memberikan wawasan berharga bagi bidang teknologi keuangan dan rekayasa perangkat lunak dengan menyediakan proses desain dan implementasi transisi. Tesis ini menyoroti keunggulan microservice dan arsitektur berbasis peristiwa serta menawarkan potensi tantangan dan arah masa depan proyek. Selain itu, penelitian ini juga membahas...

menggarisbawahi pentingnya pemantauan waktu nyata dan manajemen kesalahan proaktif dalam
Mempertahankan sistem berkinerja tinggi. Temuan ini diharapkan dapat memandu praktik industri.
Para peneliti dan konsultan dalam mengatasi tantangan serupa di ekosistem pembayaran digital,
mendukung inovasi dan pertumbuhan dalam lanskap keuangan yang semakin kompleks(7).

Melalui eksplorasi ini, tesis ini tidak hanya meningkatkan kemampuan Mollie untuk memenuhi
memenuhi tuntutan saat ini dan masa depan, tetapi juga menyediakan model arsitektur yang terukur dan mudah beradaptasi yang dapat diterapkan pada semua platform skala besar yang menghadapi lingkungan yang dinamis dan kompetitif.
komentar.

2

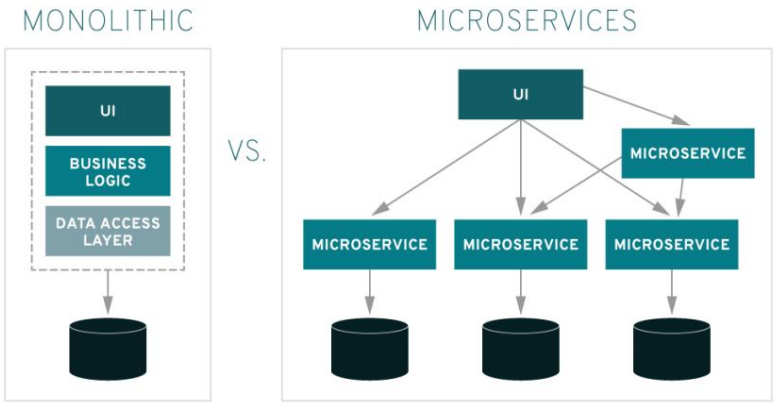
Latar belakang

Digitalisasi layanan keuangan yang pesat telah menyebabkan ledakan pilihan pembayaran,
masing-masing melayani pasar dan kebutuhan pelanggan yang berbeda. Penyedia layanan pembayaran (PSP),
Seperti Mollie, mereka harus mengintegrasikan berbagai metode pembayaran ke dalam platform mereka untuk menawarkan layanan yang komprehensif.
pengalaman yang lancar bagi pedagang dan konsumen. Seiring munculnya metode pembayaran baru di berbagai bidang
di berbagai wilayah, kemampuan untuk mengintegrasikan metode-metode ini menjadi satu kesatuan yang terukur dan efisien.
Sistem ini telah menjadi tantangan utama bagi platform pembayaran.

2.1 Arsitektur Layanan Mikro

Secara tradisional, platform pembayaran dirancang menggunakan arsitektur monolitik, di mana semua
fungsionalitas—seperti pemrosesan pembayaran, keamanan, manajemen pengguna, dan integrasi API—
tions—terkait erat dalam satu sistem. Meskipun desain ini menawarkan kesederhanaan dan
Dengan sentralisasi, sistem ini kesulitan untuk berkembang secara efektif seiring dengan munculnya metode dan fitur pembayaran baru.
Sistem monolitik biasanya menghadapi hambatan kinerja, sulit untuk dipelihara.
tetap, dan menjadi semakin kompleks dari waktu ke waktu, sehingga menyulitkan untuk mengintegrasikan hal-hal baru.
metode pembayaran tanpa mengganggu layanan yang sudah ada.

Untuk mengatasi keterbatasan ini, banyak organisasi telah beralih ke arsitektur microservices.
Arsitektur di mana fungsionalitas dibagi menjadi layanan yang lebih kecil dan dapat diimplementasikan secara independen.
Microservices memungkinkan skalabilitas yang lebih baik, isolasi kesalahan, dan pemeliharaan yang lebih mudah,
memungkinkan organisasi untuk mengintegrasikan fitur atau layanan baru (seperti metode pembayaran) tanpa
yang memengaruhi seluruh sistem. Seperti yang ditunjukkan pada gambar , dalam konteks sistem pembayaran,
2.1, microservices dapat sangat bermanfaat dengan memungkinkan penambahan API pembayaran baru atau
metode, seperti GoCardless, tanpa mengubah seluruh arsitektur platform (8). Tidak seperti
arsitektur monolitik, di mana semua komponen saling terhubung dan harus diimplementasikan.



Gambar 2.1: Monolitik vs. Mikroservis

Sebagai satu kesatuan, microservices adalah layanan independen yang dapat dikembangkan, diimplementasikan, dan diskalakan secara terpisah. Gaya arsitektur ini telah mendapatkan popularitas dalam beberapa tahun terakhir, khususnya untuk platform berskala besar, karena fleksibilitas, skalabilitas, dan sifat toleransi kesalahannya. Berikut adalah beberapa karakteristik utama dari Microservices:

- Modularitas: Setiap microservice mencakup fungsionalitas bisnis spesifik, seperti pemrosesan pembayaran, manajemen pengguna, atau penanganan notifikasi. Modularitas ini memastikan bahwa perubahan atau peningkatan pada satu layanan dapat dilakukan secara independen.
  tanpa memengaruhi seluruh sistem.

- Penerapan Independen: Layanan mikro dapat diterapkan secara independen, memungkinkan-
  Memungkinkan pengembang untuk memperbarui atau mengembalikan layanan tertentu tanpa perlu melakukan penyebaran ulang seluruh sistem. Fitur ini sangat berguna dan ampuh, terutama untuk platform pembayaran besar seperti Mollie, di mana masalah dapat diselesaikan atau metode pembayaran dapat diintegrasikan tanpa mengganggu fungsi lain dan pengalaman pengguna.

- Skalabilitas: Salah satu keunggulan utama dari microservices adalah skalabilitas horizontal.
  Layanan individual dapat diskalakan secara independen berdasarkan beban yang ditunjukkan pada monitor. aplikasi toring. Misalnya, layanan pembayaran yang memproses transaksi dapat diskalakan.
  meningkat selama periode permintaan tinggi (misalnya, Black Friday) tanpa memengaruhi layanan lain. seperti pelaporan atau manajemen pelanggan.

- Pengembangan Poliglot: Arsitektur microservices memungkinkan penggunaan berbagai bahasa pemrograman dan basis data yang disesuaikan dengan kebutuhan spesifik setiap layanan.
  Fleksibilitas ini sangat penting dalam sistem heterogen di mana layanan dapat dirancang.

berdasarkan kebutuhan mereka akan kinerja, skalabilitas, atau keamanan, tanpa ketergantungan yang signifikan.

- Isolasi Kesalahan: Karena layanan mikro beroperasi secara independen, kegagalan pada satu layanan tidak selalu menyebabkan seluruh sistem gagal. Isolasi kesalahan ini memastikan ketahanan sistem yang lebih baik, yang sangat penting dalam platform pembayaran di mana pemrosesan transaksi harus selalu andal dan berkelanjutan.

Integrasi berbagai metode pembayaran ke dalam platform terpadu seperti Mollie membutuhkan Arsitektur yang dapat mendukung fleksibilitas dan skalabilitas. Sistem monolitik tradisional- Sistem berbasis monolitik, di mana semua fitur terhubung erat ke dalam satu basis kode yang besar, seringkali kesulitan memenuhi tuntutan ini karena keterbatasan bawaannya. Pertama, sulit untuk diskalakan. Dalam arsitektur monolitik, penskalaan membutuhkan duplikasi seluruh sistem, bahkan jika hanya satu komponen yang membutuhkan sumber daya tambahan. Hal ini menyebabkan inefisiensi dan peningkatan biaya operasional yang dapat secara signifikan memengaruhi kinerja seiring bertambahnya ukuran sistem. Selain itu, memodifikasi atau menambahkan fungsionalitas baru dalam sistem monolitik dapat memerlukan perubahan ekstensif pada seluruh basis kode, meningkatkan risiko munculnya bug atau waktu henti selama penerapan. Hal ini dapat sangat merepotkan di domain fintech yang dinamis. Terdapat volume data yang besar dan layanan pemrosesan backend terlibat untuk setiap pelanggan. Untuk penerapan berkelanjutan, hal ini lebih menantang dalam arsitektur monolitik. Seluruh sistem harus diuji dan diterapkan sebagai satu unit, bahkan untuk perubahan kode kecil. Microservices memungkinkan siklus pengembangan yang lebih cepat, memungkinkan adaptasi yang lebih cepat dan membutuhkan lebih sedikit sumber daya komputasi untuk perubahan pasar dan peraturan.

Di sisi lain, arsitektur berbasis microservices mendukung modularitas dan independensi. dengan cara yang sempurna, yang secara efektif menyelesaikan masalah yang dihadapi oleh sistem skala besar tradisional. Dengan Dengan memecah berbagai fungsi menjadi layanan yang lebih kecil dan mandiri, setiap layanan dapat dikembangkan, diterapkan, dan diskalakan secara independen. Hal ini meningkatkan kelincahan dan mengurangi kompleksitas. Pendekatan ini mengurangi ketergantungan antar tim dan memungkinkan mereka untuk memilih alat dan teknologi terbaik untuk setiap layanan, sehingga mengoptimalkan kinerja sistem.
Selain itu, microservices memungkinkan pembaruan yang lebih cepat dan lebih sering melalui deployment independen. Organisasi mampu beradaptasi dengan persyaratan baru dan membuat kerja tim lebih efektif. efisien.

Meskipun microservices menawarkan banyak keuntungan, masih ada beberapa tantangan tertentu yang harus dipertimbangkan dan diatasi, mulai dari desain, pemantauan, hingga pengujian (9). Mengelola banyak layanan independen membutuhkan alat kolaborasi dan pemantauan yang canggih untuk memastikan kelancaran operasi. Platform pembayaran harus menerapkan infrastruktur yang kuat untuk

Mengelola komunikasi dan penyeimbangan beban di seluruh layanan mikro. Selain itu, dalam layanan mikro-
Dalam arsitektur layanan, layanan seringkali perlu berkomunikasi satu sama lain. Hal ini memunculkan
kompleksitas. Selama pengembangan, pengembang harus memulai berbagai microservice untuk memastikan perubahan
berfungsi pada komunikasi antar microservice tersebut. Ini juga berarti memastikan
Komunikasi yang andal, aman, dan berlatensi rendah sangat penting. Komunikasi berbasis peristiwa, yang akan dibahas di
bagian selanjutnya, adalah salah satu solusi untuk tantangan ini. Solusi lainnya...
Tantangannya adalah konsistensi data. Mencapai konsistensi data di seluruh layanan terdistribusi bisa jadi kompleks, terutama
jika ada beberapa data yang dibagikan melalui platform yang berbeda. Pembayaran
Platform sering kali menangani data secara real-time, dan memastikan konsistensi antara berbagai layanan pembayaran
memerlukan desain yang cermat, terutama dalam hal mematuhi standar regulasi.

2.2 Sistem Berbasis Peristiwa

Sistem berbasis peristiwa adalah pola arsitektur di mana komponen sistem saling berkomunikasi.
dengan menghasilkan dan mengonsumsi peristiwa atau masukan yang memicu sistem untuk merespons. Dalam sistem jenis
ini, berbagai komponen berkomunikasi melalui peristiwa, dan sistem bereaksi.
Dalam sistem semacam ini, layanan merespons peristiwa saat peristiwa tersebut terjadi. Dalam sistem semacam ini, layanan
merespons peristiwa, yang merupakan sinyal yang menunjukkan bahwa telah terjadi perubahan, seperti selesainya transaksi
pembayaran, tindakan pengguna, atau webhook. Arsitektur berbasis peristiwa (Event-driven architectures/EDA) menawarkan
keuntungan signifikan dalam sistem yang membutuhkan pemrosesan waktu nyata dan skalabilitas tinggi, sehingga menjadikannya
sangat cocok untuk platform pembayaran yang menangani banyak transaksi.

2.2.1 Fitur Utama

Sesuai namanya, sistem berbasis peristiwa (event-driven systems) didasarkan pada peristiwa. Suatu peristiwa mewakili...
Perubahan status sistem. Misalnya, otorisasi pembayaran yang berhasil, dari mitra seperti Ideal, akan menghasilkan suatu
peristiwa yang dapat memicu proses selanjutnya.
atau layanan. Entitas yang menghasilkan atau menerbitkan peristiwa disebut produsen peristiwa. Untuk
Sebagai contoh, dalam konteks pengintegrasian metode pembayaran, penyelenggara acara dapat berupa keduanya.
penyedia solusi pembayaran dan para mitra yang terlibat dalam pembayaran dan transaksi.
memproses dan menghasilkan peristiwa untuk setiap status transaksi. Konsumen peristiwa, di sisi lain, adalah layanan atau
komponen yang mendengarkan dan menanggapi peristiwa. Posisi mereka
Dapat dibalik. Setelah suatu peristiwa terjadi, peristiwa tersebut dikonsumsi oleh satu atau lebih layanan yang bertanggung
jawab untuk menangani peristiwa tersebut, seperti memperbarui status pembayaran, mengirim pemberitahuan,
atau memicu tindakan lanjutan seperti pengembalian dana atau penolakan pembayaran.

Bagi platform pembayaran skala besar yang perlu memproses ribuan transaksi di berbagai pasar dan metode pembayaran, sistem berbasis peristiwa (event-driven system) memberikan beberapa keunggulan penting:

- Komunikasi Asinkron: Arsitektur berbasis peristiwa memungkinkan komponen untuk berkomunikasi secara asinkron, artinya mereka dapat terus memproses tugas lain tanpa menunggu satu respons pun. Hal ini mengurangi hambatan dan meningkatkan efisiensi sistem, yang penting dalam lingkungan berkinerja tinggi seperti pemrosesan pembayaran (?).

- Skalabilitas: Fitur komunikasi asinkron memungkinkan sistem berbasis peristiwa.
  untuk mencapai skalabilitas (?). Seiring meningkatnya volume transaksi, produsen peristiwa dapat terus menghasilkan peristiwa sementara konsumen melakukan penskalaan horizontal untuk memproses peristiwa tersebut.
  Hal ini memungkinkan sistem untuk mengurangi latensi dan meningkatkan throughput keseluruhan dengan menangani banyak peristiwa secara bersamaan.

- Pemrosesan Waktu Nyata: EDA ideal untuk sistem yang membutuhkan penanganan data secara waktu nyata.
  Dalam sistem pembayaran, pemrosesan transaksi secara real-time sangat penting untuk memastikan kecepatan.
  respons terhadap tindakan pelanggan. EDA memungkinkan platform pembayaran untuk memproses dan mengkonfirmasi transaksi secara instan, memberikan pengalaman pengguna yang lancar bagi pedagang maupun pelanggan.
  dan konsumen.

- Pemisahan Layanan: EDA memisahkan produsen dan konsumen peristiwa, memungkinkan setiap layanan untuk berkembang secara independen. Hal ini sangat bermanfaat bagi platform pembayaran yang mengintegrasikan berbagai metode pembayaran. Misalnya, integrasi
  GoCardless tidak mengharuskan perubahan pada layanan terkait pembayaran lainnya di Mollie's.
  platform.

2.2.2 Apache Kafka

Saat ini, sistem berbasis peristiwa biasanya didukung oleh teknologi seperti Apache Kafka (10). Apache Kafka adalah platform streaming peristiwa terdistribusi yang memainkan peran sentral dalam mengimplementasikan arsitektur berbasis peristiwa. Awalnya diperkenalkan oleh LinkedIn dan kemudian dijadikan open source, Kafka telah menjadi pilihan populer untuk sistem berbasis peristiwa, terutama di domain seperti layanan keuangan di mana pemrosesan data real-time sangat penting.

Kafka memisahkan komunikasi antar layanan melalui penggunaan produsen dan konstruktor.
Para produsen mengirimkan data peristiwa ke Kafka, dan konsumen mengambil data tersebut dengan berlangganan topik Kafka tertentu. Pustaka klien Kafka yang sederhana namun ampuh memungkinkan para produsen.

dan konsumen untuk berinteraksi dengan klaster Kafka melalui API komunikasi yang transparan. Salah satunya
Kekuatan utama Kafka terletak pada ekosistemnya, yang mencakup alat tambahan seperti Kafka.
Streams dan Kafka Connect, memperluas fungsionalitas Kafka untuk memenuhi berbagai kebutuhan pemrosesan data.
kebutuhan pemrosesan dan integrasi. Alat-alat ini memungkinkan kemampuan pemrosesan aliran data yang canggih dan integrasi
tanpa hambatan dengan sistem data lainnya, menjadikan Kafka solusi serbaguna untuk
saluran data waktu nyata (11).

Penyimpanan peristiwa Kafka terstruktur sebagai log yang hanya dapat ditambahkan (append-only log), dan konsumen dapat mengambilnya.
Peristiwa diproses menggunakan offset, yang melacak posisi peristiwa di dalam partisi. Hal ini memungkinkan pemrosesan peristiwa
yang andal dan berurutan. Setiap klaster Kafka terdiri dari broker Kafka,
dengan setiap broker memegang satu atau lebih partisi dari suatu topik. Kafka memastikan redundansi dan
Toleransi kesalahan melalui replikasi topik. Hal ini memungkinkan Kafka untuk mentolerir hingga n-1 kegagalan broker (di mana n
adalah jumlah replika) tanpa kehilangan data atau mengganggu pengiriman pesan. Dibandingkan dengan broker pesan lain, seperti
RabbitMQ, Kafka unggul dalam menangani beban kerja throughput tinggi dengan muatan pesan yang lebih pendek, yang
bermanfaat untuk
platform, khususnya untuk sistem pembayaran di mana banyak transaksi kecil diproses.
dengan cepat.

Kafka sangat penting untuk mengelola peristiwa pembayaran secara real-time. Ketika pembayaran dimulai, Kafka memastikan
bahwa peristiwa tersebut didistribusikan ke semua layanan yang diperlukan, sehingga memungkinkan pengelolaan independensi.
Pemrosesan transaksi, notifikasi, dll. Dengan mengadopsi Apache Kafka dan EDA, kita dapat
Memastikan integrasi metode pembayaran seperti Ideal yang lancar sambil mempertahankan skalabilitas,
daya tanggap, dan ketahanan. Pendekatan ini memungkinkan perusahaan fintech untuk memisahkan layanan,
Menanggapi peristiwa secara real-time dengan efisien, dan memenuhi kebutuhan pedagang dan konsumen yang beragam dan
terus berkembang di ranah pembayaran yang dinamis.

2.3 Pemantauan Data Waktu Nyata dan Kinerja Sistem

Karena keandalan dan efisiensi transaksi sangat penting dalam sistem pembayaran, pemantauan data secara real-time dan
manajemen kinerja sistem memainkan peran penting dalam menjaga keandalan dan efisiensi tersebut.
Stabilitas, keamanan, dan daya tanggap. Dengan semua jenis metode pembayaran yang terintegrasi dalam satu platform,
pemantauan menjadi semakin penting untuk mengelola kompleksitas yang meningkat.
dan lalu lintas. Hal ini memungkinkan platform pembayaran untuk secara proaktif mengidentifikasi dan mengatasi masalah sebelum
Hal ini berdampak pada pengguna. Dengan terus memantau metrik kunci seperti latensi transaksi, tingkat kesalahan, dan
pemanfaatan sumber daya, sistem dapat mendeteksi anomali sejak dini, sehingga memungkinkan tindakan cepat.
intervensi untuk meminimalkan waktu henti atau kegagalan transaksi.

Dalam pemrosesan pembayaran, masalah terpenting adalah tingginya—penundaan atau kegagalan transaksi dapat menyebabkan kerugian finansial yang signifikan, masalah kepatuhan, atau kehilangan pelanggan (12). Oleh karena itu, solusi pemantauan yang efektif harus memberikan visibilitas komprehensif ke semua bagian sistem, mulai dari permintaan API hingga interaksi basis data. Alat pemantauan modern seperti Datadog, Prometheus dan Grafana memberikan wawasan simultan tentang kesehatan sistem dengan cara Melacak metrik kunci seperti throughput transaksi, latensi, pemanfaatan sumber daya, dan kesalahan. tarif. Visibilitas waktu nyata ini sangat penting untuk:

- Pemantauan Terpusat: Mengumpulkan log dan data kinerja dari semua mi-
  Layanan CROSS dalam dasbor terpusat membantu mengidentifikasi berbagai penyebab masalah.
  terus-menerus. Ini penting ketika berurusan dengan arsitektur terdistribusi seperti mikroserver.
  kebiasaan buruk dan arsitektur berbasis peristiwa.

- Peringatan dan Notifikasi Waktu Nyata: Dengan mengatur peringatan berdasarkan ambang batas yang telah
  ditentukan (misalnya, latensi transaksi), platform dapat memberi tahu teknisi secara langsung.
  Ketika terjadi kesalahan, mengurangi waktu respons. Kapan pun hal yang tidak terduga terjadi.
  Ketika suatu kejadian terjadi di dalam aplikasi, kesalahan tersebut dicatat ke layanan pelaporan kesalahan untuk
  debugging.

- Pelacakan Metrik Kinerja: Memantau metrik penting seperti transaksi
  waktu respons, throughput permintaan, kinerja basis data, dan penggunaan sumber daya server
  memastikan bahwa platform tetap berada dalam batas kinerja yang dapat diterima. Selain itu,
  Wawasan waktu nyata tentang pemanfaatan sumber daya memungkinkan penskalaan layanan secara dinamis. Hal
  ini sangat berguna dalam arsitektur layanan mikro, di mana berbagai layanan dapat diskalakan.
  secara independen berdasarkan beban transaksi saat ini.

Mengenai alat pemantauan, ada banyak platform yang dapat digunakan untuk mengatasi hal tersebut. masalah.

- Datadog: Sebuah platform pemantauan dan analitik berskala cloud, Datadog menawarkan pemantauan dan analitik secara real-time.
  Wawasan tentang kinerja microservices. Terintegrasi dengan mulus dengan cloud.
  infrastruktur dan mendukung penskalaan otomatis serta peringatan berdasarkan indikator kinerja yang ditentukan
  khusus. Struktur harga modular Datadog memungkinkan pengguna untuk memilih fitur-fitur spesifik yang dibutuhkan.
  kemampuan pemantauan yang mereka butuhkan, memberikan fleksibilitas bagi organisasi dari berbagai ukuran,
  khususnya mereka yang mengelola layanan mikro dan infrastruktur cloud dinamis.

- Dynatrace: Platform kecerdasan perangkat lunak yang berfokus pada pemantauan full-stack dengan
  Otomatisasi berbasis AI dan analisis akar penyebab. Menawarkan penemuan otomatis dan
  Visibilitas waktu nyata di seluruh aplikasi, infrastruktur, dan pengalaman pengguna. Mesin AI Dyna-trace, Davis,
  mengotomatiskan deteksi masalah kinerja dan mengoreksinya.
  Hal ini mengarah pada akar permasalahan, sehingga ideal untuk lingkungan cloud-native yang kompleks.
  Sistem ini sangat cocok untuk perusahaan yang membutuhkan pengamatan mendalam, pemecahan masalah otomatis,
  dan pelacakan ujung-ke-ujung di seluruh infrastruktur hibrida dan cloud.

- Prometheus: Sebuah solusi pemantauan sumber terbuka, Prometheus mengkhususkan diri dalam pemantauan waktu.
  Pengumpulan data seri dan peringatan waktu nyata. Dapat diintegrasikan dengan layanan untuk
  mengumpulkan data kinerja secara real-time, sehingga ideal untuk memantau peristiwa berbasis kejadian.
  arsitektur. Prometheus menyimpan data secara lokal dan menyediakan sistem peringatan yang fleksibel, sehingga memudahkan
  Ini adalah pilihan utama untuk pemantauan kinerja infrastruktur dan aplikasi, terutama
  di lingkungan Kubernetes. Ini terintegrasi dengan baik dengan alat visualisasi seperti Grafana.
  dan banyak digunakan untuk memantau layanan mikro dan infrastruktur dinamis.

Pemantauan data secara real-time merupakan bagian yang sangat penting bagi platform pembayaran, untuk memastikan bahwa Mereka mampu menangani permintaan yang terus meningkat dari para pedagang dan konsumen. Dengan memanfaatkan hal tersebut, Dengan memanfaatkan pemantauan waktu nyata, perusahaan fintech dapat memenuhi SLA dan mengoptimalkan sumber daya. pemanfaatan, dan menyediakan layanan yang andal dan efisien kepada pengguna.

Bab ini memperkenalkan konsep-konsep utama yang diterapkan pada arsitektur baru, termasuk:
Arsitektur microservices, sistem berbasis event, dan pemantauan real-time. Integrasi teknologi-teknologi ini dapat membentuk strategi yang kohesif untuk mengatasi tantangan-tantangan kritis.
dari segi skalabilitas, toleransi kesalahan, dan responsivitas sistem. Arsitektur microservices memungkinkan...
memungkinkan penskalaan dan pengembangan layanan secara independen, mengurangi hambatan dan meningkatkan Modularitas. Sistem berbasis peristiwa, yang didukung oleh alat-alat seperti Apache Kafka, memisahkan layanan.
interaksi dan memungkinkan komunikasi asinkron dan waktu nyata untuk skenario dengan throughput tinggi. Sementara itu, alat pemantauan waktu nyata, seperti yang kami gunakan Datadog untuk Platform Mollie,
memberikan visibilitas penting terhadap kesehatan sistem, memfasilitasi manajemen kesalahan secara proaktif dan Optimalisasi kinerja. Bersama-sama, konsep-konsep ini meletakkan dasar bagi skalabilitas.
dan platform pembayaran yang andal. Bab-bab selanjutnya akan membahas lebih lanjut tentang bagaimana mereka diterapkan dan diimplementasikan.

3

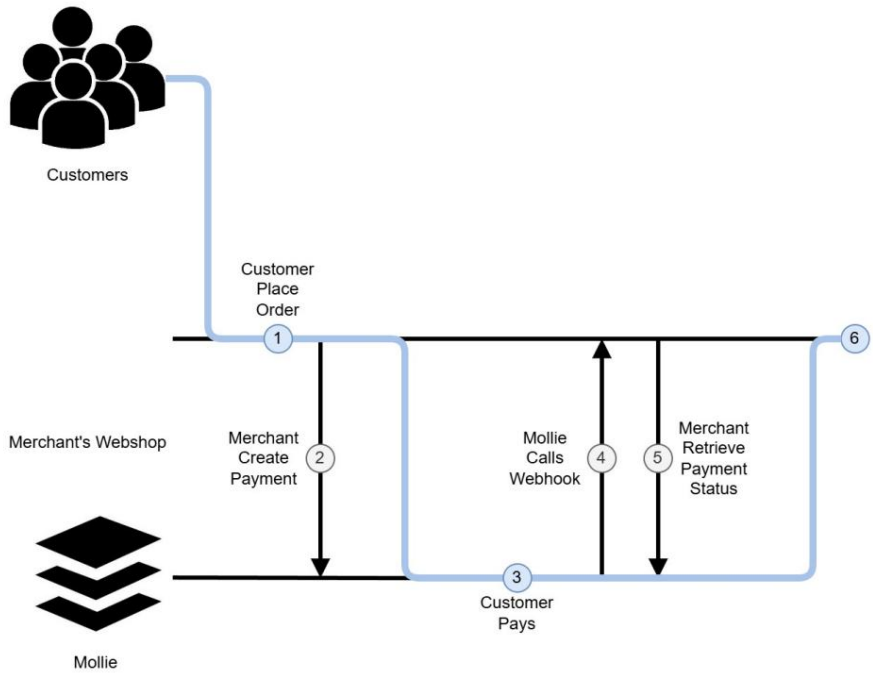
Pekerjaan Terkait

Meningkatnya kompleksitas sistem pembayaran dan kebutuhan akan solusi yang skalabel dan tahan terhadap kesalahan telah mendorong penelitian signifikan ke dalam desain arsitektur backend, yang digerakkan oleh peristiwa (event-driven). sistem, dan modularitas melalui layanan mikro. Bagian ini meninjau studi-studi yang relevan dan menyoroti kontribusi, keterbatasan, dan penerapannya pada permasalahan penelitian.

3.1 Desain API Backend untuk Transaksi Non-Tunai

Adam et al.(13) mengembangkan server backend berbasis REST API untuk mendukung pembayaran tanpa uang tunai Sistem untuk komunitas ritel skala kecil. Desainnya mencakup fitur-fitur penting seperti:
Sistem ini menggunakan pendaftaran pengguna, manajemen saldo, dan pemrosesan transaksi, dll.
Node.js untuk logika sisi server dan MongoDB untuk manajemen data. Mekanisme otentikasi berbasis token (JWT) diimplementasikan untuk memastikan interaksi API yang aman.
Sistem ini bertujuan untuk memungkinkan integrasi tanpa hambatan di berbagai platform klien melalui sistem terpusat. Backend, didasarkan pada fleksibilitas API REST untuk komunikasi antar layanan. Mereka
Mereka menguji sistem backend mereka di bawah lalu lintas API yang tinggi, mensimulasikan 100 permintaan bersamaan per Kedua, mencapai tingkat keberhasilan 76,92% di 13 fitur. Fitur terkait transaksi.
menunjukkan keandalan yang lebih rendah—45% untuk mengurangi saldo pembeli dan 65% untuk menambah saldo penjual. ances—menyoroti masalah skalabilitas dan inefisiensi dalam penanganan kueri basis data di bawah beban berat.

Meskipun API REST menawarkan fleksibilitas untuk sistem skala kecil, sifat stateless-nya membatasi fleksibilitas tersebut. Membutuhkan interaksi basis data yang sering, sehingga menimbulkan latensi dan membatasi skalabilitas. Hal ini Masalah menjadi kritis dalam lingkungan pembayaran frekuensi tinggi, di mana ribuan transaksi per detik membutuhkan pemrosesan waktu nyata (14). Penelitian ini mengatasi tantangan ini dengan merancang backend yang dioptimalkan untuk pemrosesan data frekuensi tinggi. Alih-alih

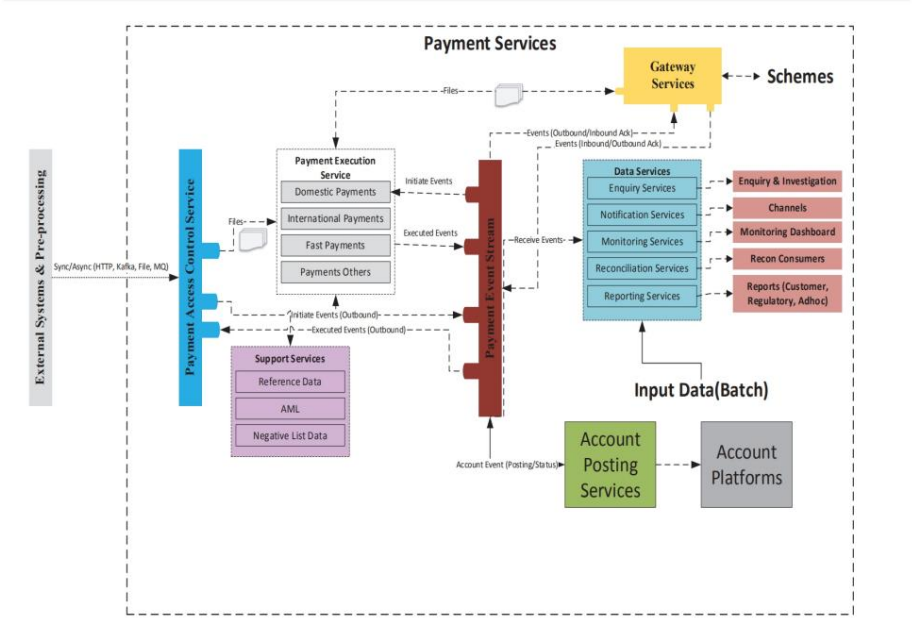


Gambar 3.1: Alur API di Mollie

Dengan mengandalkan sepenuhnya pada API REST, sistem ini mengintegrasikan komunikasi berbasis peristiwa (yang akan dibahas kemudian) untuk mengurangi ketergantungan pada basis data, meningkatkan throughput, dan meningkatkan toleransi kesalahan. Dengan memperluas karya dasar Adam dkk. ke platform berskala besar, studi ini menawarkan Arsitektur backend yang tangguh dan dirancang khusus untuk memenuhi tuntutan sistem pembayaran modern.

Studi lain oleh Aué et al.(15) tentang kesalahan dalam integrasi web API dalam skala besar Sistem pembayaran, menganalisis lebih dari 2,43 juta respons kesalahan API dari platform pembayaran Adyen. Studi ini mengkategorikan kesalahan integrasi API ke dalam 11 penyebab umum. Misalnya- Kesalahan tersebut disebabkan oleh beberapa hal, seperti input pengguna yang tidak valid atau hilang, izin yang tidak memadai, dan kesalahan internal. Mereka menemukan bahwa sebagian besar kesalahan berasal dari data yang tidak valid atau hilang, serta masalah integrasi pihak ketiga. juga penting. Studi tersebut menyimpulkan bahwa konsumen API sering mengandalkan dokumentasi resmi. Namun, mereka menghadapi tantangan yang berbeda karena kurangnya panduan tentang penanganan kesalahan dan proses pemulihan.

Sementara studi mereka berfokus pada mendiagnosis kesalahan dalam API yang ada, penelitian ini membahas hal tersebut. Masalah-masalah ini dapat diatasi dengan desain API yang lebih kuat dan tahan terhadap kesalahan untuk platform pembayaran. Kami- Dengan menggunakan arsitektur berbasis peristiwa, sistem secara proaktif meminimalkan kesalahan dengan memisahkan layanan dan memvalidasi permintaan API sebelum diproses. Fitur flag juga diperkenalkan untuk... Memenuhi persyaratan khusus dari berbagai metode pembayaran dan pedagang. Beberapa pembayaran



Gambar 3.2: Arsitektur Berbasis Peristiwa untuk Layanan Pembayaran oleh Vangala dkk.

Metode ini memerlukan informasi lebih lanjut, seperti alamat penagihan dan pengiriman saat membuat pesanan. Misalnya, pembayaran. Alat pemantauan waktu nyata semakin meningkatkan deteksi kesalahan dan memberikan peringatan langsung, memungkinkan penyelesaian kesalahan yang cepat dan mengurangi waktu henti. Pendekatan ini memastikan integrasi API yang terukur dan andal untuk sistem pembayaran frekuensi tinggi, mengatasi keterbatasan desain berbasis REST tradisional.

3.2 Arsitektur Berbasis Peristiwa dan Streaming

Arsitektur berbasis peristiwa dan streaming telah menjadi penting dalam layanan keuangan karena kemampuannya untuk menangani aliran data waktu nyata dan meningkatkan efisiensi sistem. Vangala dkk.(16) menyelidiki peran Apache Kafka dalam mengelola transaksi throughput tinggi. Tidak hanya menyoroti kemampuan Kafka untuk memisahkan layanan, tetapi juga mengurangi latensi dan meningkatkan toleransi kesalahan. Studi mereka menunjukkan bahwa model event-streaming Kafka memungkinkan setiap layanan untuk memproses peristiwa transaksi secara independen seperti yang ditunjukkan pada gambar 3.2, sehingga meningkatkan Ketahanan dengan menghilangkan ketergantungan sinkron yang dapat menyebabkan hambatan.

Berdasarkan temuan Vangala dkk., tesis ini memanfaatkan Kafka dalam integrasi pembayaran Mollie untuk menangani peristiwa transaksi di berbagai penyedia pembayaran. Dengan mengadopsi Pendekatan event-streaming berbasis Kafka, integrasi ini dapat mencapai latensi rendah dan kinerja tinggi. Skalabilitas, menyediakan fondasi yang fleksibel untuk menangani transaksi waktu nyata. Memperluas

Mengenai hal ini, Vyas et al.(17) mengevaluasi kinerja Apache Kafka dalam streaming data real-time. Studi mereka menyoroti efisiensi Kafka dalam menangani volume data besar dengan minimal
Pemanfaatan memori dengan menggunakan penyimpanan pesan berbasis disk. Mereka juga mengeksplorasi hal-hal penting. metrik kinerja seperti throughput dan latensi untuk menunjukkan bagaimana konfigurasi pa-
Parameter seperti partisi dan interval polling memengaruhi kinerja sistem.

Meskipun memiliki manfaat, penerapan EDA dalam sistem transaksi keuangan bervolume tinggi juga menghadirkan tantangan, termasuk duplikasi peristiwa, peristiwa di luar urutan,
dan penyeimbangan beban, seperti yang disebutkan dalam makalah. Selama pengembangan, masalah serupa diamati, terutama ketika pemberitahuan peristiwa berulang menyebabkan operasi yang berlebihan, yang mengarah pada terhadap peringatan palsu dan penurunan efisiensi. Untuk menyelesaikan konflik, strategi idempotensi (unik) (pengidentifikasi peristiwa dan penanda waktu) diperkenalkan.

Pemantauan waktu nyata dan pelacakan terdistribusi, seperti yang direkomendasikan oleh kedua tim, berperan penting. memainkan peran penting dalam memastikan kinerja sistem. Dalam tesis ini, alat pemantauan yang digunakan adalah... Digunakan untuk melacak metrik kinerja seperti latensi, throughput, dan tingkat kesalahan. Ini
Metrik memberikan wawasan yang dapat ditindaklanjuti untuk mengidentifikasi dan mengatasi hambatan, serta memastikan stabilitas. penanganan transaksi. Langkah-langkah proaktif tersebut tidak hanya meningkatkan keandalan sistem tetapi juga Menawarkan pengalaman pembayaran yang lancar, memenuhi tuntutan ketat dari layanan keuangan.

3.3 Arsitektur Monolitik & Mikroservis

Dalam beberapa tahun terakhir, microservices telah menjadi pendekatan arsitektur yang populer bagi organisasi yang beralih dari sistem monolitik tradisional. Sebuah studi kasus (18) tentang mendesain ulang
platform integrasi sektor real estat dari Arsitektur Berorientasi Layanan (SOA) ke
Arsitektur microservices menyoroti manfaat utama dari pergeseran ini. Transisi ini memungkinkan kemandirian. Peningkatan skala layanan, yang memungkinkan alokasi sumber daya ke hambatan spesifik tanpa memengaruhi seluruh platform secara bersamaan. Siklus penerapan menjadi jauh lebih singkat,
dengan pembaruan yang diterapkan pada layanan individual tanpa mengakibatkan gangguan sistem secara keseluruhan. Selain itu, komunikasi berbasis antrian yang melekat pada layanan mikro juga meningkat.
Toleransi kesalahan dengan mengisolasi kegagalan pada layanan tertentu, mencegah gangguan yang meluas. Peningkatan manajemen sumber daya dan kemampuan untuk menggunakan teknologi spesifik untuk berbagai keperluan. Layanan tersebut semakin meningkatkan skalabilitas dan efisiensi operasional.

Namun, studi tersebut menunjukkan bahwa transisi ke arsitektur microservices juga menghadirkan tantangan, termasuk-kompleksitas pengelolaan ketergantungan antar layanan dan komunikasi asinkron
aliran kation. Studi ini mengusulkan penggunaan teknologi service mesh untuk menyederhanakan layanan. interaksi dan alat pelacakan terdistribusi untuk meningkatkan kemampuan pengamatan dan pemecahan masalah.

Berdasarkan temuan ini, microservices sangat menguntungkan untuk kebutuhan dengan frekuensi tinggi.
Lingkungan transaksi di bidang fintech. Komunikasi berbasis peristiwa, pada gilirannya, mengurangi layanan.
penggabungan dan meningkatkan isolasi kesalahan. Desain modular ini memastikan integrasi dari
metode pembayaran baru atau fungsionalitas pendaftaran tanpa mengganggu stabilitas sistem. Dengan
Memisahkan interaksi basis data dalam layanan mikro khusus, seperti yang ditunjukkan pada
Dalam studi ini, konsistensi transaksi dipertahankan sambil menyederhanakan akses dan pengelolaan data.
Pendekatan ini menjaga fleksibilitas dan ketahanan, menjadikannya pilihan ideal untuk
domain dinamis yang didorong oleh skalabilitas.

Selain itu, arsitektur seperti itu dapat secara signifikan meningkatkan keamanan dan privasi data.
Li(19) menyajikan kerangka desain yang kuat untuk sistem pembayaran elektronik yang disesuaikan dengan fi-
lembaga keuangan, dengan menekankan protokol keamanan yang ketat, skalabilitas, dan transaksi.
Kecepatan—persyaratan inti dalam lingkungan berisiko tinggi seperti keuangan. Studi ini membahas...
Beberapa prinsip desain termasuk kontrol akses berlapis, enkripsi data, dan keamanan ujung-ke-ujung untuk kepatuhan
terhadap standar peraturan seperti PSD2 dan GDPR. Re-
Pencarian ini menguraikan persyaratan keamanan tingkat tinggi yang harus dipenuhi dalam sistem pembayaran apa pun,
yang memberikan wawasan yang sangat berharga untuk penelitian kami. Namun, penelitian ini masih berfokus pada hal-hal tradisional.
model pemrosesan pembayaran dan kurangnya penekanan pada modularitas.

Sebaliknya, tesis ini dibangun di atas kerangka keamanan mereka dengan menggunakan arsitektur microservices.
Arsitektur khusus untuk mengintegrasikan berbagai layanan dalam ekosistem Mollie. Ini
Pendekatan ini memanfaatkan layanan mikro untuk mengisolasi fungsi keuangan yang sensitif, memastikan bahwa
Protokol keamanan di sekitar setiap modul selaras dengan kebutuhan kepatuhan sekaligus mendukung pemrosesan
waktu nyata. Lebih lanjut, penelitian ini memperkenalkan arsitektur berbasis peristiwa (event-driven architecture/EDA).
sebagai lapisan tambahan untuk menangani transaksi asinkron, memberikan ketahanan dan
Peningkatan keamanan dalam lingkungan pemrosesan waktu nyata.

3.4 Refleksi

Studi-studi sebelumnya di atas telah membahas aspek-aspek individual dari desain backend, event-
sistem yang digerakkan, dan layanan mikro. Namun, penelitian kami menggabungkan elemen-elemen ini untuk
mengusulkan solusi holistik untuk sistem transaksi frekuensi tinggi. Berdasarkan asinkron
Dengan menangani event dan memodularisasi layanan, karya ini menjembatani kesenjangan dalam skalabilitas, isolasi kesalahan.
tion, dan kinerja waktu nyata. Tidak seperti karya-karya sebelumnya, yang terutama berfokus pada diagnosis
Dengan mempertimbangkan masalah yang ada, penelitian ini secara proaktif merancang dan mengimplementasikan sistem yang mengurangi masalah tersebut.
tantangan umum seperti duplikasi acara dan ketergantungan antar layanan.

3. KARYA TERKAIT

Tesis ini memberikan kontribusi dengan menyediakan arsitektur yang terukur dan andal yang dirancang khusus untuk memenuhi tuntutan platform pembayaran yang terus meningkat. Tesis ini tidak hanya membahas tantangan teknis tetapi juga masalah operasional, seperti kemudahan penerapan dan pemeliharaan. Studi ini menawarkan kerangka kerja komprehensif yang dapat diterapkan tidak hanya pada platform pembayaran tetapi juga pada sistem skala besar apa pun yang membutuhkan kinerja dan skalabilitas yang kuat.

4

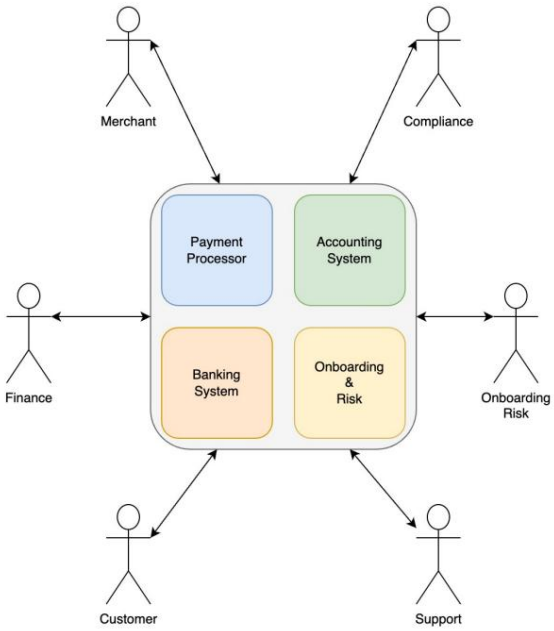
Desain

Pada bab ini, kami memberikan gambaran umum tingkat tinggi tentang arsitektur sistem, yang merinci hal-hal berikut: Struktur monolitik sebelumnya dan pendekatan modular yang diusulkan. Kami berfokus pada transisi dari arsitektur monolitik ke Layanan Otonom Independen (IAS), menyoroti manfaat dalam skalabilitas, toleransi kesalahan, dan efisiensi operasional. Layanan-layanan ini
Berkomunikasi melalui aliran peristiwa dan API, memastikan konsistensi data, pemrosesan yang efisien, dan isolasi kesalahan.

4.1 Arsitektur Sebelumnya

Arsitektur sebelumnya didasarkan pada struktur monolitik, di mana berbagai fungsi-fungsi...
Hubungan seperti pemrosesan pembayaran, pendaftaran pengguna, penilaian risiko, dan akuntansi, terjalin erat. terintegrasi dalam satu basis kode dan lingkungan basis data. Meskipun pengaturan ini menyediakan
model penerapan yang mudah dan menyederhanakan beberapa aspek berbagi data di seluruh
modul.

Seperti yang ditunjukkan pada gambar 4.1, semua layanan dan fungsionalitas terintegrasi ke dalam Arsitektur monolitik, yang menyebabkan latensi rendah dan kompleksitas operasional. Sebagai trans-
Seiring meningkatnya volume aksi, arsitektur monolitik kesulitan untuk menanganinya.
Input/output data dalam jumlah besar secara efisien. Peningkatan skala membutuhkan penambahan sumber daya ke seluruh sistem. alih-alih ke layanan individual. Pendekatan penskalaan vertikal ini mahal dan tidak efisien.
efisien. Sistem ini gagal menyeimbangkan sumber daya berdasarkan komponen-komponen tertentu. Misalnya, pembayaran. Prosesor membutuhkan sumber daya komputasi yang lebih banyak daripada layanan lainnya. Namun, hal itu tidak terjadi. menyediakan sumber daya yang cukup yang menyebabkan waktu respons yang lama. Selain itu, arsitektur monolitik tersebut kurang memiliki isolasi antar layanan. Kegagalan pada satu komponen dapat merusak dan
dapat menyebabkan seluruh sistem mengalami kerusakan. Misalnya, jika layanan akuntansi mengalami kesalahan.



Gambar 4.1: Arsitektur Monolitik Platform Mollie

Saat pedagang melakukan onboarding, hal itu berpotensi memengaruhi seluruh sistem, dan mengganggu proses tersebut. Layanan penting lainnya, seperti pemrosesan pembayaran. Hal ini dapat menyebabkan risiko tinggi terhadap layanan. keandalan.

Bagi pengembang, memperbarui atau menerapkan fitur baru memerlukan penerapan ulang seluruh monosistem. aplikasi litik. Proses menjalankan pipeline di GitLab membutuhkan waktu sekitar 40 menit. Pertama. Bahkan kesalahan kode kecil pun memaksa pengembang untuk menunggu alur kerja penerapan yang panjang. untuk mengidentifikasi masalah, memperlambat siklus pengembangan. Ini juga berarti tim pengembang harus bekerja dalam basis kode bersama, yang menciptakan hambatan dalam proses pengembangan. Saat pertama kali bergabung dengan perusahaan, saya menghabiskan hampir tiga bulan untuk membiasakan diri dengan basis kode karena terlalu besar untuk memahami setiap bagiannya, termasuk pengujian unit dan lainnya. pengujian integrasi. Selain itu, karena semua layanan berbagi basis data dan lingkungan yang sama, Menerapkan protokol keamanan secara efektif merupakan tantangan. Data sensitif yang digunakan oleh Sistem perbankan atau manajemen risiko dapat diakses di seluruh modul karena semua orang dapat menggunakannya. Token terenkripsi yang sama, meningkatkan risiko kebocoran data. Struktur monolitik. upaya yang rumit untuk memenuhi standar kepatuhan, dan sulit untuk mengisolasi dan mengamankannya. alur data spesifik.

Kesimpulannya, meskipun arsitektur monolitik pada awalnya memberikan solusi yang sederhana dan terpusat- Pendekatan yang terstruktur saat pertama kali membuat platform ini, mulai kesulitan untuk menangani pertumbuhan yang terus meningkat.

Kebutuhan berubah seiring berjalannya waktu. Toleransi kesalahan yang rendah, kurangnya skalabilitas, dan keterbatasan keamanan. Mengidentifikasi kebutuhan akan arsitektur yang lebih modular, terukur, dan tangguh. Meskipun transisi itu kompleks, mengadopsi struktur berbasis microservices, atau Independent Autonomous Services (IAS), akan memberikan fondasi yang lebih kuat untuk pertumbuhan perusahaan.

4.2 Desain Arsitektur

Bagian ini menjelaskan arsitektur tingkat tinggi untuk platform Mollie. Fokusnya adalah pada...
komponen-komponen besar dan interaksi di antara mereka. Solusi yang diusulkan mencakup keduanya.
Titik di cakrawala dan cara strategis untuk mencapainya. Tujuan utamanya adalah untuk menangani jumlah transaksi yang terus meningkat dan membatasi insiden. Seiring bertambahnya jumlah pengguna.
Seiring pertumbuhan ekonomi, kita menghadapi tantangan eksternal dan internal yang harus diatasi. Dari faktor eksternal, terdapat semakin banyak transaksi dan tuntutan yang lebih tinggi terhadap pemrosesan transaksi.
per detik dan penggunaan puncak platform kami. Selain itu, pelanggan bervolume tinggi mengharapkan tingkat kualitas dan stabilitas yang lebih tinggi. Di sisi lain, tenaga kerja teknik yang terus bertambah.
yang perlu bekerja secara independen dan otonom untuk tim internal. Selain itu, keinginan untuk mengurangi kompleksitas dan keterkaitan basis kode saat ini untuk mempercepat
Waktu untuk mencapai kompetensi bagi para insinyur baru sangat penting mengingat fase pertumbuhan yang pesat, serta membantu tim yang ada agar dapat bergerak lebih cepat dan dengan risiko yang lebih rendah. Oleh karena itu, langkah-langkah baru ini diperlukan.
Arsitektur harus sesuai dengan persyaratan yang menghilangkan kendala saat ini dan menyediakan fondasi yang stabil untuk pengembangan di masa mendatang. Berikut adalah persyaratan fungsional tingkat tinggi.
harus dipenuhi:

4.2.1 Persyaratan Utama

- Ketahanan Komponen: Komponen individual harus mampu mengalami kegagalan tanpa berdampak.
  Proses bisnis penting seperti pemrosesan pembayaran.

- Toleransi Kesalahan: Sistem harus dirancang untuk menangani kegagalan dengan baik.
  pemrosesan pembayaran.

- Skalabilitas: Arsitektur harus mendukung penskalaan horizontal untuk menangani peningkatan
  volume transaksi dapat diprediksi.

- Ketersediaan: Sistem harus mencakup beberapa zona ketersediaan dan mendukung failover.
  di berbagai wilayah.

- Ramah Tata Kelola: Arsitektur harus menerapkan solusi yang menyederhanakan tata kelola sistem dan infrastruktur.
- Standardisasi Alat: Minimalkan penggunaan alat yang berlebihan dengan fungsi yang serupa.
- Keamanan: Solusi harus memungkinkan pembuatan komponen yang aman secara default. Sistem harus menyediakan alat bagi para insinyur dan DevOps yang dapat mengungkap cacat keamanan sejak dini dalam siklus pengembangan.

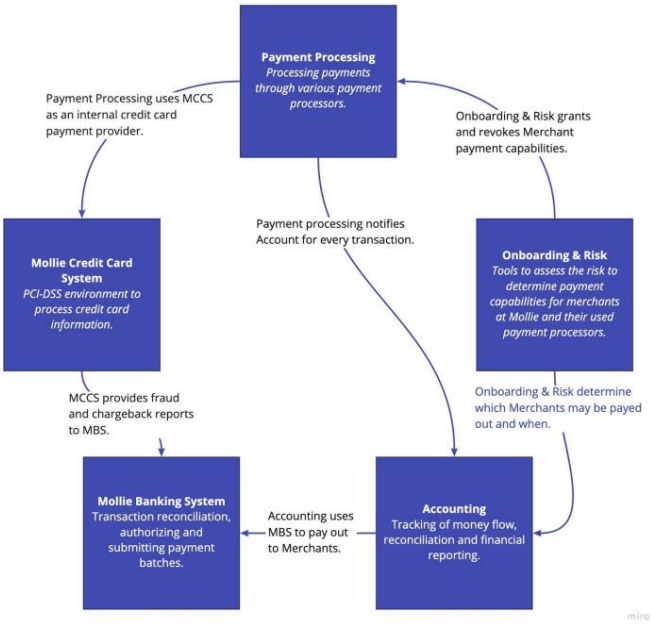
4.2.2 Tujuan Sekunder

- Modularitas: Struktur modular memungkinkan prediktabilitas yang lebih tinggi dan pengiriman yang lebih cepat. fungsi baru.
- Desain Cloud-Native: Manfaatkan solusi cloud-native untuk meningkatkan skalabilitas dan keandalan.
- Penyederhanaan Debugging: Arsitektur harus meminimalkan waktu yang dihabiskan di lingkungan produksi. debugging.
- Batasan Layanan: Batasan yang jelas antara layanan memungkinkan konsumen untuk tetap mendapatkan layanan yang sesuai. tidak terpengaruh oleh perubahan internal.
- Solusi Standar: Utamakan solusi standar industri daripada solusi yang dibuat khusus.
- Konsistensi Pola: Lebih menyukai pola yang dapat diulang daripada implementasi sekali saja.

4.2.3 Arsitektur yang Diusulkan

Dengan mengikuti aturan dan tujuan di atas, arsitektur yang diusulkan ditunjukkan pada gambar 4.2, dengan Dengan memecah struktur monolitik, layanan dibagi menjadi area yang memberikan nilai paling besar kepada pelanggan. Area nilai ini dipengaruhi oleh domain operasional Mollie (pembayaran online) dan layanan yang diberikan kepada pedagang. Pemrosesan pembayaran merupakan inti dari layanan ini. Alat tambahan mendukung Onboarding & Risk, memungkinkan mereka untuk menilai risiko dan mengelola kemampuan pembayaran pedagang. Arsitektur ini juga memisahkan akuntansi, yang mendukung proses bisnis sekunder, dari layanan pembayaran utama.

Dengan mengidentifikasi area-area kunci ini dan ketergantungannya, kita dapat mendefinisikan batasan-batasan yang berbeda. Diagram tersebut menggambarkan gambaran umum tingkat tinggi, menunjukkan bagaimana platform monolitik yang besar tersebut disegmentasikan menjadi unit-unit logis yang lebih kecil dengan batasan interaksi yang jelas. Setiap unit, atau kontainer, mewakili layanan independen. Kontainer bersifat mandiri,



Gambar 4.2: Arsitektur Platform Mollie yang Diusulkan

dengan sumber daya mereka sendiri, dan berkomunikasi satu sama lain. Setiap layanan dan fungsionalitas diekspos secara eksternal melalui load balancer.

Komunikasi antar layanan dicapai melalui aliran peristiwa dan API. Setiap layanan memiliki sumber daya sendiri, seperti basis data dan penyimpanan. Komunikasi antar layanan mengikuti protokol transaksional (misalnya, HTTP). Data asinkron ditransfer melalui aliran peristiwa, yang memungkinkan model publikasi-berlangganan untuk komunikasi. Alih-alih mengintegrasikan-

Melalui akses basis data bersama, aliran peristiwa memungkinkan komunikasi yang terpisah. Pesan

Di dalam sistem, data bersifat tetap (immutable), dan data tidak dapat diakses secara langsung. Struktur ini meningkatkan pemisahan dan memungkinkan skalabilitas yang fleksibel.

Mekanisme aliran peristiwa akan menjadi sumber daya bersama untuk semua konteks. Akses

Informasi tersebut diorganisir berdasarkan aliran. Aliran dapat direpresentasikan sebagai topik, antrean, atau pertukaran. Aliran peristiwa digunakan untuk komunikasi, bukan antrean pekerjaan.

Semua komunikasi yang terdapat dalam aliran peristiwa harus sesuai dengan standar umum.

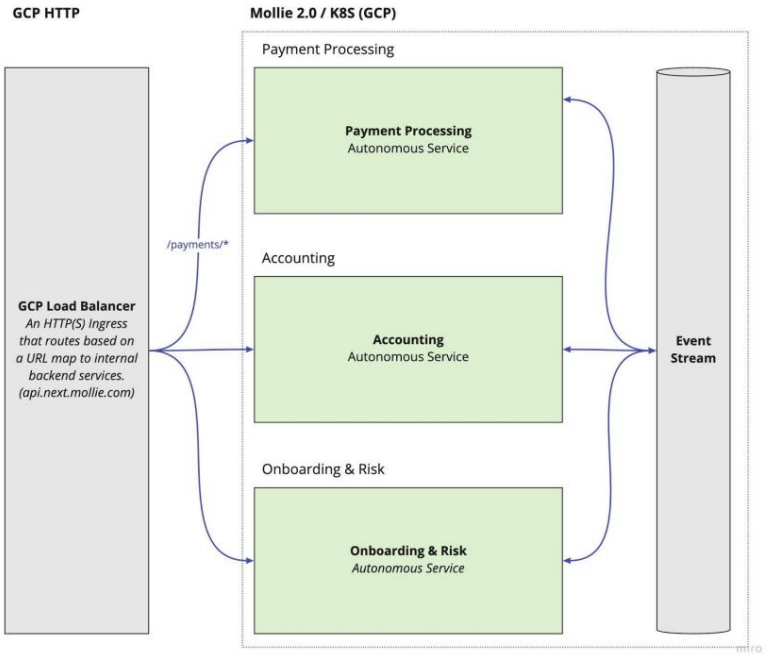
format. Implementasi akan dipilih berdasarkan penyebut umum tertinggi.

Mekanisme tersebut harus dapat diskalakan secara horizontal dan memiliki izin berbasis pengguna. Tujuannya adalah...

Salah satu tujuan penggunaan format umum ini adalah tata kelola. Oleh karena itu, izin harus dikelola.

dengan cara konfigurasi sebagai kode, memastikan kita dapat mengauditnya dan mendapatkan gambaran umum yang deterministik.

tentang akses informasi. Kami ingin membuat seperangkat alat yang memastikan komponen-komponen tersebut.



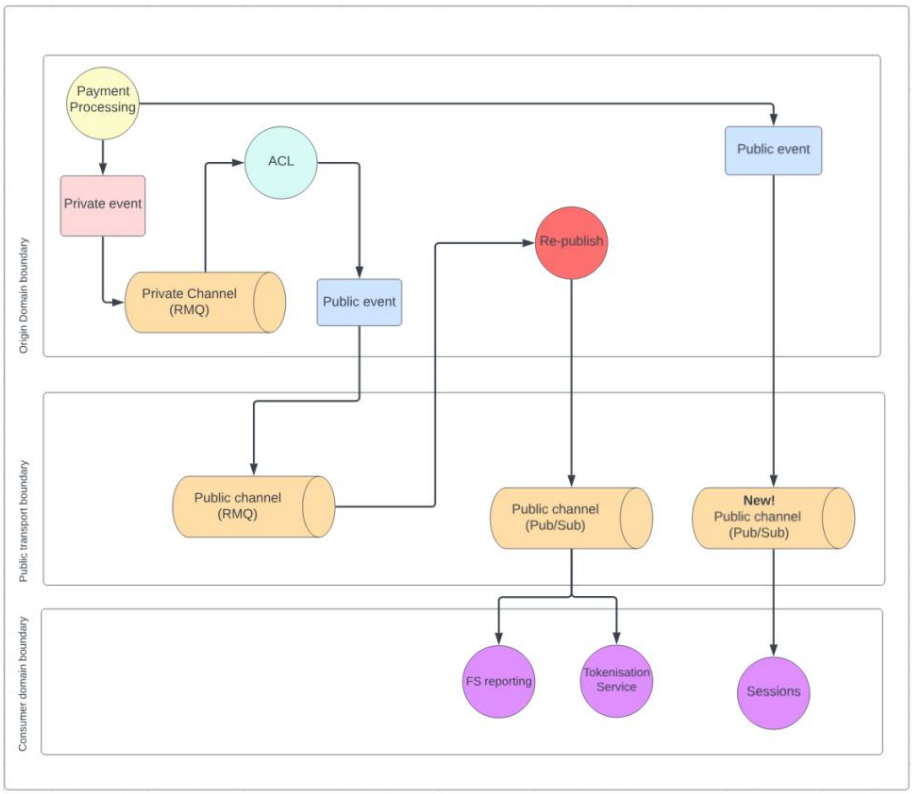
Gambar 4.3: Arsitektur Layanan Berbasis GCP untuk Pemrosesan Pembayaran

aman dan sesuai standar secara default. Isolasi sumber daya, komunikasi terstandarisasi, dan
Kontrol akses memainkan peran besar dalam mencapai hal tersebut.

Singkatnya, arsitektur yang diusulkan didasarkan pada Layanan Otonom Independen.
(IAS) dan layanan mikro. Ini adalah unit mandiri yang selaras dengan aliran nilai Mollie, menyediakan struktur yang terukur,
tangguh, dan modular yang mendukung pertumbuhan di masa depan dan
kemampuan beradaptasi.

4.3 Mengoptimalkan Pengiriman Peristiwa

Dalam arsitektur saat ini, setiap domain menyediakan aliran peristiwa untuk memungkinkan asinkron.
integrasi di seluruh layanan. Kami membedakan antara acara publik dan privat menggunakan
antarmuka penanda, namun masih ada beberapa ambiguitas tentang saluran pengiriman mana (topik, pertukaran, atau antrean)
yang ditujukan untuk konsumsi umum atau internal. Pengaturan yang ada menyebabkan penundaan yang signifikan (hingga 80
detik) bagi sistem hilir untuk mengonsumsi peristiwa,
Hal ini tidak dapat diterima untuk kasus penggunaan baru yang membutuhkan pemrosesan mendekati waktu nyata, khususnya
untuk peristiwa yang berkaitan dengan pembayaran. Keterlambatan pengiriman ini mengurangi efektivitas layanan publik.
Peristiwa dalam kasus penggunaan yang membutuhkan latensi minimal. Untuk memperjelas dan menyederhanakan perbedaan ini,



Gambar 4.4: Arsitektur Pengiriman Peristiwa yang Dioptimalkan

RFC ini mengusulkan konvensi standar untuk secara eksplisit membedakan antara publik dan
saluran pribadi.

Untuk mengatasi keterlambatan tersebut, mekanisme penerbitan langsung untuk sistem pengiriman peristiwa adalah...
diperkenalkan. Ini akan melibatkan penambahan tabel kotak keluar sekunder untuk menangani penerbitan langsung.
sebagai bagian dari proses bisnis yang memicu peristiwa-peristiwa ini. Memanfaatkan data yang sudah ada.
yang dihasilkan selama pemrosesan pembayaran dan alur kerja transaksi, acara publik akan di-
Langsung dikirim ke saluran publik baru menggunakan pengaturan kotak keluar ini. Selama migrasi,
Kami akan melanjutkan proses yang ada untuk mengubah acara privat menjadi acara publik dan mendistribusikan kembali
acara publik dari RMQ ke Pub/Sub, meminimalkan gangguan sambil secara bertahap
Mengadopsi pengaturan baru.

Untuk implementasi, tabel outbox baru dibuat untuk mencatat peristiwa publik. Tabel ini
akan digunakan untuk mengirimkan peristiwa secara langsung ke saluran publik yang telah ditentukan, melewati saluran utama.
Jalur pengiriman berbasis RMQ, sehingga menghilangkan penundaan dalam konsumsi peristiwa untuk hilir.
Selain itu, saluran publik baru diperkenalkan khusus untuk peristiwa yang terjadi hampir secara real-time.
Konsumen hilir secara bertahap akan beradaptasi dengan saluran-saluran ini, sehingga memastikan transisi yang lancar.

dengan dampak minimal. Untuk mengurangi risiko, baik pengiriman RMQ-ke-Pub/Sub yang ada maupun pengiriman langsung yang baru masih bekerja sama. Ini memberikan solusi cadangan sekaligus memastikan keandalan bagi konsumen yang belum beralih ke saluran baru.

5

Pelaksanaan

Sebagai respons terhadap tantangan kompleks yang dihadirkan oleh arsitektur sebelumnya, kami terus mengadopsi arsitektur modular dan terpisah untuk menyederhanakan pengembangan (integrasi).
(metode pembayaran baru untuk tim kami) dan mengoptimalkan skalabilitas sistem. Bagian implementasi ini menguraikan langkah-langkah yang diambil untuk memisahkan komponen inti, mendesain sistem berbasis plugin. arsitektur, dan mengintegrasikan metode pembayaran baru dengan cara yang mudah. Teknologi utama-teknologi dan metodologi—termasuk gateway, microservices, dan integrasi plugin lapisan-lapisan tersebut digunakan untuk memastikan kekokohan dan fleksibilitas dalam ekosistem Mollie.

5.1 Memisahkan Arsitektur

Karena arsitektur sebelumnya sangat besar dan rumit, kami terus berusaha untuk mengurangi... menggabungkan struktur monolitik yang sebelumnya mengatur platform Mollie dengan-tanpa memengaruhi fungsionalitas yang ada. Sistem monolitik yang ada membutuhkan biaya yang besar. sumber daya untuk ditingkatkan atau diperbarui, sehingga menjadi mahal dan memakan waktu. Beralih ke pendekatan modular, Arsitektur yang terpisah memungkinkan setiap komponen berfungsi secara independen, sehingga mengurangi kompleksitas teknis. utang kal dan meningkatkan skalabilitas. Berikut ini menunjukkan proses decoupling yang diikuti dalam tiga langkah.

5.1.1 Pindah Berdasarkan Namespace

Dua layanan utama yang biasanya kami berikan, yaitu Pembayaran dan Onboarding, telah diidentifikasi. sebagai modul inti yang penting untuk manajemen transaksi keuangan dan kontrol akses pengguna. Setiap layanan dirancang sebagai modul otonom yang mampu bertransisi ke masa depan. microservice lengkap jika diperlukan oleh tuntutan skalabilitas. Layanan Pembayaran mengelola

Semua aspek transaksi keuangan, termasuk interaksi API untuk pengembalian dana dan penanganan webhook. Sebaliknya, layanan Onboarding berfokus pada otentikasi pengguna.

Kode dari direktori application/classes dipindahkan ke folder khusus untuk setiap domain. Kepemilikan kode diklarifikasi, kode yang memiliki kepemilikan bersama perlu dipisahkan. Selama tahap ini masih akan ada keterkaitan antar domain. Kode yang menggunakan kode dari domain lain masih diperbolehkan pada tahap ini. Kita membuka kunci kuantifikasi keterkaitan domain dengan menghitung impor lintas domain dan penggunaan kelas. Ada empat aturan.

Di Sini:

- Pindahkan semua kode dari direktori application/classes ke namespace khusus domain.
  (misalnya, Pembayaran, Pendaftaran Pengguna).
- Kode yang berkaitan dengan logika bisnis harus dipindahkan ke nama domain spesifiknya masing-masing.
  kecepatan untuk memastikan bahwa logika spesifik domain tetap terenkapsulasi.
- Kode yang bergantung pada basis data ditempatkan secara eksklusif di dalam namespace khusus domain.
  untuk menghilangkan akses basis data lintas domain.
- Fungsionalitas dasar yang penting di berbagai domain telah dipindahkan ke satu platform yang lebih luas.
  ruang nama aplikasi.

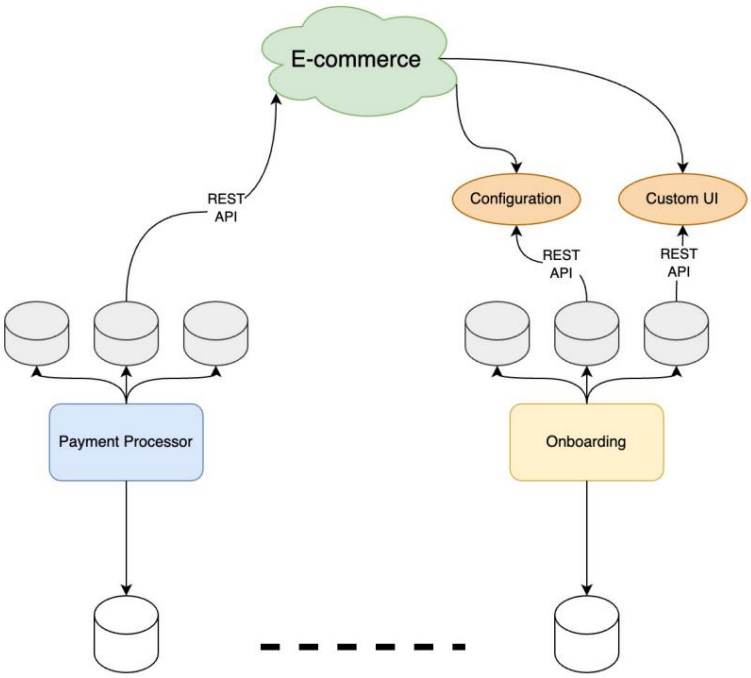
5.1.2 Menghilangkan Akses Langsung Lintas Domain ke Basis Data

Langkah selanjutnya bertujuan untuk menghilangkan akses basis data langsung antar domain, mengalihkan ketergantungan data setiap domain ke sistem penyimpanan independen. Dengan memisahkan basis data, setiap layanan dapat memelihara basis datanya sendiri tanpa gangguan lintas domain, yang sangat penting untuk skalabilitas dan keamanan.

Tujuannya adalah untuk mencapai nol akses basis data langsung antar domain. Kemajuan dapat dilacak di GCP yang memantau ketergantungan basis data. Metrik ini divisualisasikan pada dasbor. Hal ini memungkinkan pemantauan waktu nyata dan memastikan bahwa setiap layanan mematuhi batasan domainnya masing-masing. Langkah ini membangun fondasi untuk sistem yang sepenuhnya modular di mana setiap domain dapat ditingkatkan skalanya dan dioperasikan secara independen tanpa referensi silang langsung.

5.1.3 Menerapkan kembali titik integrasi melalui jaringan

Dengan kode dan basis data yang dibagi menjadi modul-modul berbeda, sudah saatnya untuk mengimplementasikan ulang semua titik integrasi menggunakan API berbasis jaringan atau aliran peristiwa. Transisi ini memungkinkan komunikasi waktu nyata antar layanan sekaligus memastikan pemisahan (decoupling), karena titik-titik integrasi



Gambar 5.1: Gerbang E-commerce

tidak lagi melibatkan panggilan langsung lintas batas domain. Dengan memanfaatkan titik akhir API atau
Dengan antrian pesan untuk interaksi, integrasi menjadi terukur dan tidak bergantung pada lokasi.
Hal ini berpotensi mendukung migrasi ke layanan berbasis cloud seperti GCP dalam jangka panjang.
API atau struktur berbasis peristiwa memungkinkan setiap domain untuk berkomunikasi secara asinkron,
Memberikan ketahanan dan fleksibilitas.

Untuk memastikan transisi yang sukses, pelacakan kemajuan sangat diperlukan.
Keefektifan transisi ini memerlukan pemantauan untuk memastikan tidak ada pelanggaran terhadap integrasi langsung
antar sub-domain. Hal ini memastikan bahwa setiap domain beroperasi sebagai unit yang mandiri.
siap untuk penskalaan dan penerapan secara independen.

5.1.4 Manajemen Gerbang dan Alur Kerja

Untuk mengelola dan mengoordinasikan alur kerja di seluruh layanan yang terpisah, gateway memainkan peran penting.
Gateway ini memproses peristiwa bisnis tingkat tinggi menjadi informasi yang dapat ditindaklanjuti.
alur kerja. Ini secara sempurna mengabstraksikan logika yang mendasarinya dan memungkinkan peralihan yang mulus antar alur kerja.
layanan.

Sebagai contoh, jika suatu peristiwa pembayaran memicu alur kerja pembayaran terpisah, gateway akan mendeteksinya.

Kebutuhan ini kemudian memicu langkah-langkah yang sesuai. Sentralisasi manajemen alur kerja ini menyederhanakan interaksi layanan. Selama gateway, seperti yang ditunjukkan pada gambar 5.1, menanganinya terlebih dahulu, layanan tidak perlu mengetahui detail implementasi spesifik dari setiap alur kerja. Gateway meningkatkan modularitas dengan berfungsi sebagai lapisan abstraksi, yang memungkinkan modifikasi alur kerja yang fleksibel tanpa perubahan langsung pada komponen inti.

5.1.5 Arsitektur Berbasis Peristiwa

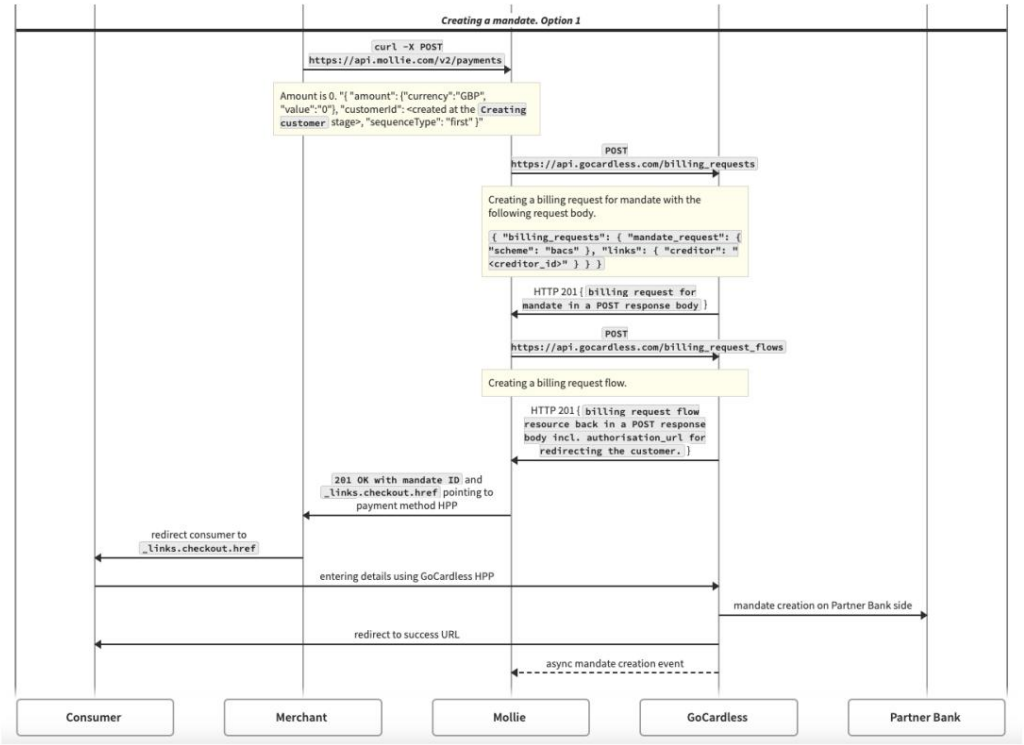
Untuk mendukung interaksi waktu nyata antara komponen yang terpisah, kami memilih arsitektur berbasis peristiwa (event-driven architecture/EDA) sebagai solusinya. Dengan menggunakan Apache Kafka untuk manajemen peristiwa, EDA memfasilitasi komunikasi asinkron. Hal ini meminimalkan ketergantungan antar layanan secara bersamaan. Dengan menerapkan Kafka untuk komunikasi antar layanan, kami memastikan setiap layanan beroperasi secara independen. Ketika suatu tindakan menghasilkan atau menerima, maka tindakan tersebut mengonsumsi atau menghasilkan suatu peristiwa sebagai perantara tanpa memengaruhi seluruh sistem.

5.2 Integrasi Mitra

Bagian utama yang saya kerjakan dalam proyek ini adalah pemeliharaan metode pembayaran yang ada dan integrasi metode pembayaran baru ke dalam platform. Arsitektur yang terpisah (decoupled architecture) memberikan dukungan untuk integrasi metode pembayaran yang lebih efisien. Proses integrasi sebelumnya membutuhkan perubahan kode yang signifikan dan konfigurasi manual, yang menyebabkan waktu pengembangan yang lebih lama. Di bawah arsitektur baru, setiap metode pembayaran diperlakukan sebagai modul plugin, yang mencakup semua persyaratan khusus platform. Setiap tim dapat berkolaborasi dengan API yang konsisten, berfokus pada domain mereka sendiri. Hal ini membuat integrasi metode pembayaran baru lebih efisien dan memastikan konsistensi di seluruh integrasi.

5.2.1 Arsitektur Berbasis Plugin

Setiap platform e-commerce yang didukung (misalnya, Shopify, Lightspeed, Wix), sebagai hulu dari keseluruhan proses, beroperasi melalui modul plugin (20). Modul ini berfungsi sebagai perantara antara sistem internal Mollie dan platform e-commerce tertentu. Struktur modul plugin mengikuti model 80/20, di mana 80% fungsionalitas dibagi di seluruh integrasi, dengan 20% sisanya disesuaikan dengan persyaratan platform individual. Pendekatan ini memungkinkan sistem internal Mollie untuk berkomunikasi secara efisien dengan platform eksternal. Hal ini tidak hanya mengisolasi logika spesifik platform di dalam setiap plugin, tetapi juga memungkinkan kinerja yang konsisten di berbagai lingkungan. Dalam arsitektur ini, platform eksternal dapat berkomunikasi dengan platform Mollie dengan lancar, tanpa mengubah API apa pun.



Gambar 5.2: Alur Kerja Pembuatan Mandat - GoCardless

persyaratan selama transisi. Setiap modul plugin mengelola konfigurasi khusus platform secara internal, sehingga memisahkan persyaratan pengaturan dari sistem inti Mollie.
Struktur ini meminimalkan kebutuhan akan adaptasi khusus pada basis kode utama Mollie.

Arsitektur ini mendukung setiap modul plugin untuk terhubung secara independen ke layanan khusus melalui API REST, menawarkan kustomisasi yang fleksibel untuk beragam platform e-commerce. persyaratan. Selain itu, ini mempercepat proses platform eksternal baru tetapi juga menyediakan kerangka kerja modular yang terukur yang selaras dengan tujuan strategis Mollie untuk menciptakan
Sistem integrasi e-commerce yang tangguh.

5.2.2 Integrasi Metode Pembayaran Berbasis Alur Kerja

Di sisi lain, metode pembayaran berperan sebagai hilir bagi kami, menangani hal-hal berikut:
Pembayaran dan transaksi lebih berbasis perbankan. Sedangkan untuk proses pengembangannya, setelah metode pembayaran baru diidentifikasi, alur kerja akan dimulai di dalam gateway.
untuk mengelola siklus pembayaran, memastikan bahwa prosesnya berjalan lancar mulai dari otorisasi awal hingga penyelesaian dan pemrosesan pengembalian dana selanjutnya. Proses ini didorong oleh interaksi. dengan Layanan Pembayaran dan komponen berbasis peristiwa. Integrasi pembayaran baru

Metode yang kami gunakan mengikuti alur kerja yang jelas dan terstruktur. Tim kami bekerja pada layanan pembayaran dan layanan onboarding untuk mendukung berbagai langkah dalam alur kerja, menyediakan penanganan standar untuk tugas-tugas seperti membuat dan memproses pembayaran/mandat, mengelola pembayaran berulang, dan menangani pengembalian dana dan penolakan pembayaran. Namun, metode pembayaran yang berbeda mungkin memiliki persyaratan yang berbeda dalam hal detail pembayaran. Saya mengambil tahapan alur kerja integrasi metode pembayaran yang berbasis di Inggris, GoCardless, sebagai contoh.
untuk mengilustrasikan di sini 5.1.

Selain langkah-langkah umum yang disebutkan di atas, masih ada beberapa tahapan lain yang mungkin dilakukan. perlu dilakukan, seperti layanan penyelesaian, rekonsiliasi, dan pemberitahuan, yang memastikan bahwa semua dana ditransfer dengan benar dan menyediakan layanan yang disesuaikan sesuai dengan aturan. atau kebijakan dari berbagai negara. Bagi GoCardless, mengirimkan pemberitahuan adalah wajib.
Pembayaran debit langsung kepada pelanggan karena peraturan perbankan Inggris.

5.3 Pemantauan & Penerapan Berkelanjutan

Untuk mendukung ketersediaan dan keandalan tinggi yang dibutuhkan dalam domain pembayaran, proses pemantauan dan penerapan diperlukan untuk berbagai layanan dalam arsitektur baru.

5.3.1 Otomatisasi Penerapan dan Pengujian

Pipeline integrasi dan penyebaran berkelanjutan (CI/CD) banyak digunakan untuk penyebaran dan pengujian platform (21). Ketika arsitektur telah ditetapkan, penyebaran dan pengujian otomatis untuk setiap layanan dapat diproses secara independen. Modularitas arsitektur baru mendukung strategi rilis bergulir. Setiap modul dapat diperbarui tanpa memengaruhi layanan lain. Pipeline ini mempercepat siklus pengembangan dan meminimalkan waktu henti, meningkatkan pengalaman pengguna dan memfasilitasi respons yang lebih cepat terhadap

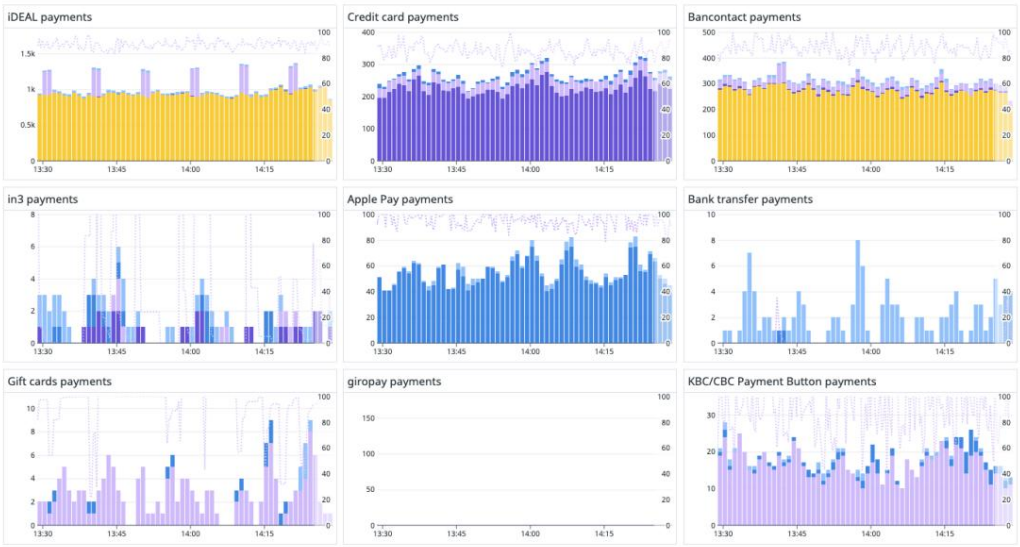
tuntutan pasar.

Setiap permintaan penggabungan (merge request/MR) ke cabang master memicu pipeline deployment otomatis. Hal ini memungkinkan ketertelusuran untuk setiap rilis, dengan pengidentifikasi commit memastikan kemungkinan untuk melakukan rollback dalam keadaan darurat. Tombol pengaktifan/penonaktifan fitur digunakan untuk mengontrol rilis fitur tertentu kepada pengguna atau grup yang ditargetkan, meningkatkan fleksibilitas dan mengurangi risiko deployment. pengelolaan.

Selain itu, setiap layanan memiliki konfigurasi pencatatan log, pemantauan, dan peringatan, seperti yang akan kita bahas di bagian selanjutnya. Datadog dan alat lainnya melacak metrik penting seperti tingkat kesalahan dan waktu respons, dengan peringatan yang diatur untuk setiap kegagalan yang tidak terduga dalam pipeline. Hal ini memastikan respons cepat terhadap masalah selama penerapan.

Panggung	Deskripsi
Proses Pendaftaran Pedagang	Proses pendaftaran pedagang dapat dilakukan secara manual atau otomatis, termasuk pengaturan kredensial dan konfigurasi yang diperlukan untuk pemrosesan pembayaran. Selain itu, terdapat metode pembayaran. Pemeriksaan aktivasi berdasarkan informasi pedagang. dan persyaratan perusahaan metode pembayaran.
Membuat Mandat	Untuk metode yang memerlukan otorisasi berbasis mandat (seperti debit langsung), alur kerja dimulai dengan membuat mandat untuk mengumpulkan pembayaran dan pembayaran selanjutnya dapat dipotong dari mandat yang sama. Seperti yang ditunjukkan pada 5.2
Pemrosesan Pembayaran	Setelah mandat ditetapkan, pembayaran dapat dibuat dan diproses menggunakan API Pembayaran Mollie. Ini melibatkan pembuatan permintaan otorisasi, pengambilan pembayaran, dan melakukan pemeriksaan penipuan yang diperlukan.
Pembayaran Berulang	Sistem harus mendukung pembayaran berulang jika mandat diizinkan, terutama untuk layanan berbasis langganan atau keanggotaan. Ini melibatkan pembuatan instruksi pembayaran berulang, pengelolaan jadwal pembayaran, dan penanganan perpanjangan otomatis.
Penanganan Webhook	Webhook memainkan peran penting dalam komunikasi dengan sistem eksternal. Untuk memproses webhook, sistem mampu menerima peristiwa masuk seperti otorisasi pembayaran, aktivasi mandat, atau pembaruan lain yang terkait dengan transaksi pembayaran (termasuk pembayaran, mandat, dan aktivasi). Peristiwa ini memicu tindakan lanjutan untuk memperbarui status transaksi, memberi tahu pengguna, atau memulai pengembalian dana dan penolakan pembayaran.
Pengembalian Dana dan Pembatalan Transaksi Pengolahan	Pengembalian dana dan pembatalan transaksi ditangani melalui alur kerja khusus yang bereaksi terhadap peristiwa seperti sengketa pelanggan atau kesalahan pembayaran, memastikan dana yang benar dikembalikan kepada pelanggan atau dipotong. dari para pedagang. Yang perlu diperhatikan adalah bahwa Pengembalian dana lebih umum daripada penarikan dana (chargeback). Penarikan dana (chargeback) tidak selalu didukung oleh semua metode pembayaran.

Tabel 5.1: Tahapan Integrasi Metode Pembayaran



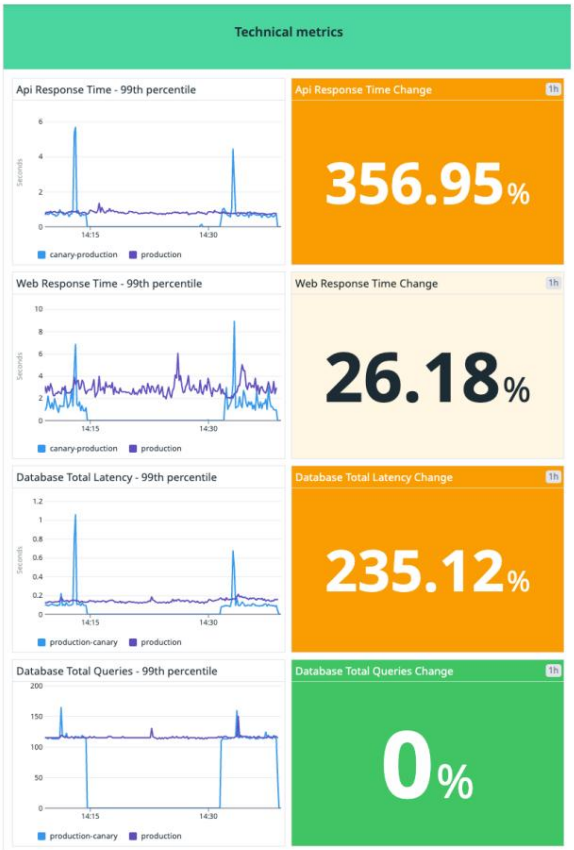
Gambar 5.3: Dasbor DataDog - Ikhtisar Metode Pembayaran

5.3.2 Pemantauan Waktu Nyata

Untuk memastikan kekokohan, kinerja, dan keandalan semua layanan di dalam platform Mollie-
Sebagai alat pemantauan waktu nyata, DataDog telah digunakan di seluruh sistem inti dan plugin. DataDog berfungsi sebagai alat pemantauan utama di Mollie, di mana metrik kinerja utama, seperti latensi transaksi, tingkat kesalahan, dan kesehatan sistem, divisualisasikan pada
Dasbor terpusat. Dasbor ini memungkinkan deteksi dan penyelesaian masalah dengan cepat.
yang memberikan gambaran komprehensif tentang kinerja operasional di berbagai metode pembayaran dan titik integrasi.

Dengan dasbor terpusat, kita dapat memperoleh wawasan tentang operasi secara real-time dan Kesehatan berbagai layanan. Pertama dan terpenting, tinjauan operasional secara real-time.
Terdapat dasbor yang menampilkan data langsung tentang volume pembayaran dan waktu pemrosesan di berbagai metode pembayaran seperti pada gambar 5.3, memungkinkan visibilitas cepat ke dalam setiap operasional. fluktuasi. Selain itu, sistem ini melacak tingkat otorisasi, penolakan, dan kesalahan untuk setiap pembayaran, memungkinkan pemecahan masalah proaktif di tingkat skema atau pengakuisisi. Ditambah lagi, terdapat dasbor yang memantau kinerja dan keandalan operasi perintah, yang penting untuk mengatur peristiwa asinkron di seluruh platform. Dasbor ini berfungsi untuk
sebagai alat penting bagi tim teknik Mollie, memungkinkan kami untuk memantau kesehatan operasional, dan bereaksi terhadap kegagalan yang tidak terduga secara tepat waktu.

Berdasarkan Kubernetes, kami menyebarkan layanan yang mengekspos metrik Prometheus (22), yang dikumpulkan oleh Agen Datadog. Anotasi pod Kubernetes digunakan untuk menentukan yang mana



Gambar 5.4: Dasbor DataDog - Metrik Teknis

Metrik yang dikumpulkan memungkinkan pemantauan yang selektif dan efisien. Beberapa metrik utama yang digunakan oleh platform Mollie meliputi: 1. Waktu respons: mengukur latensi untuk panggilan API penting (misalnya, pembayaran, otorisasi) untuk melacak kinerja dan mendeteksi perlambatan, seperti yang ditunjukkan pada gambar.
5.4. 2. Tingkat kesalahan: melacak jumlah kesalahan yang ditemui dalam interaksi layanan, yang
Menunjukkan potensi masalah dalam layanan atau ketergantungan eksternal. 3. Volume transaksi: pro-
memberikan wawasan tentang beban transaksi dengan menghitung peristiwa pembayaran atau otentikasi, memungkinkan
Mollie akan memantau penggunaan sistem dan menyesuaikannya sesuai kebutuhan. 4. Sumber daya komputasi: mengukur-
Memastikan penggunaan CPU, memori, dan sumber daya lainnya di seluruh layanan untuk menjamin efisiensi sumber daya.
alokasi.

Secara keseluruhan, kerangka kerja pemantauan waktu nyata ini memastikan bahwa Mollie mempertahankan kualitas layanan yang tinggi dan memenuhi standar kinerja. Arsitektur modular membantu mengelola berbagai layanan serta mendeteksi dan menyelesaikan potensi kesalahan dengan lebih efisien.

5. IMPLEMENTASI

6

Evaluasi

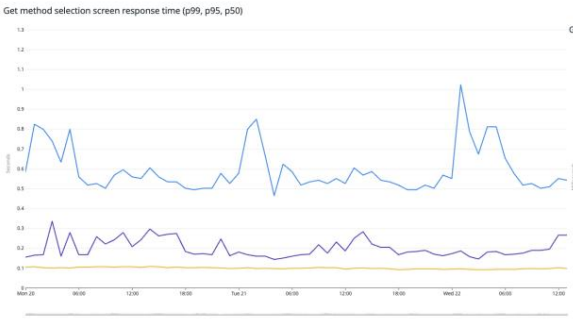
Bab ini bertujuan untuk mengevaluasi kinerja, keandalan, dan skalabilitas arsitektur microservices dan event-driven yang diusulkan, menggunakan metrik real-time dari Datadog (23). Dengan memeriksa metrik-metrik kunci, seperti waktu respons API, latensi basis data, dan peringatan kesalahan, bagian ini memvalidasi efektivitas arsitektur dalam mendukung pemrosesan pembayaran frekuensi tinggi. Temuan ini juga menyoroti area untuk peningkatan kinerja dan sumber daya komputasi, yang berkontribusi pada optimasi sistem yang berkelanjutan.

6.0.1 Waktu Respons API

Pertama-tama, arsitektur saat ini telah memberikan kontribusi signifikan terhadap peningkatan kinerja API (24), seperti yang ditunjukkan pada dasbor waktu respons Datadog untuk beberapa API 6.1.

Sebagai contoh, kita ambil salah satu dasbor waktu respons API yang detail. Grafik waktu respons yang diperbarui untuk API "Get Method Selection" menunjukkan peningkatan signifikan dibandingkan data sebelumnya, seperti yang ditunjukkan pada gambar 6.2 dan gambar 6.3. Saat ini, waktu respons persentil ke-99 stabil antara 600-700 milidetik, persentil ke-95 sekitar 350-450 milidetik, dan persentil ke-50 sekitar 200-250 milidetik.

Gambar 6.1: Gambaran Umum Waktu Respons API



Gambar 6.2: Waktu Respons Sebelumnya untuk API Pemilihan Metode Get

Gambar 6.3: Waktu Respons Saat Ini untuk Get API Pemilihan Metode

Ini merupakan pengurangan yang signifikan dari puncak sebelumnya yaitu 1 detik atau lebih pada data sebelumnya, yang menunjukkan bahwa optimasi terbaru telah efektif.

Perbaikan tersebut menunjukkan bahwa sistem tersebut telah secara efektif mengurangi hambatan-hambatan yang sebelumnya ada. yang berdampak pada kinerja. Peningkatan tersebut dianggap berasal dari tiga aspek. Pertama, karena arsitektur yang terpisah, data yang sering diakses dapat

Pertama, dengan melakukan caching, beban pada layanan backend berkurang. Kedua, pendistribusian lalu lintas yang lebih merata di seluruh instance API meningkatkan stabilitas dan keseimbangan beban. Selanjutnya, optimasi basis data dan kueri mengurangi waktu pengambilan data dan secara langsung berkontribusi pada peningkatan respons API. Evaluasi dan peningkatan ini menunjukkan ketahanan arsitektur dan

Kemampuan beradaptasi dalam menangani volume transaksi secara efisien. Namun, pemantauan berkelanjutan tetap diperlukan. Disarankan untuk memastikan peningkatan ini tetap berlaku dalam berbagai kondisi beban.

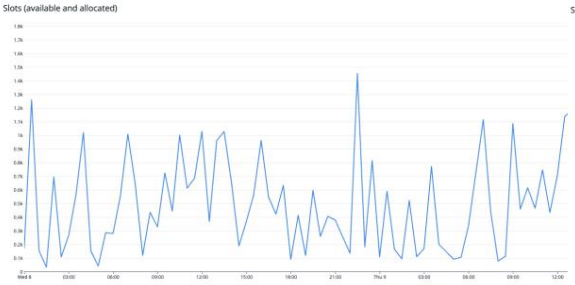
6.1 Kinerja Basis Data

Bagian ini menunjukkan bagaimana kinerja berubah dari tingkat basis data. Mengevaluasi kinerja basis data sangat penting untuk memahami efisiensi dan keandalan sistem, terutama

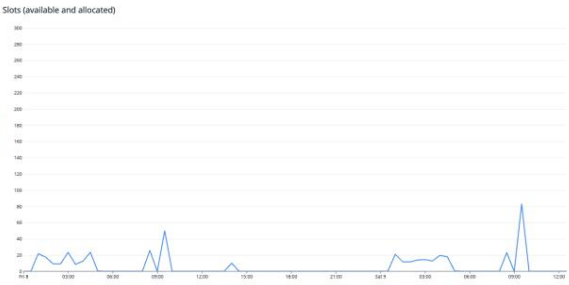
dalam platform transaksi frekuensi tinggi. Aspek-aspek kunci dari kinerja basis data meliputi:

latensi kueri, alokasi sumber daya, dan manajemen beban kerja (25). Penanganan elemen-elemen ini secara efisien memastikan bahwa sistem dapat mendukung pemrosesan waktu nyata dengan penundaan minimal, yang sangat penting untuk mempertahankan pengalaman pengguna yang responsif.

Gambar 6.4 dan 6.5 menunjukkan alokasi slot dari basis data yang mendasarinya. Pada data sebelumnya, alokasi slot menunjukkan variabilitas yang tinggi, dengan puncak hingga 1,5 ribu slot selama periode tertentu. Fluktuasi ini menunjukkan inefisiensi dalam manajemen kueri, yang berpotensi menyebabkan penundaan. dan peningkatan latensi kueri selama periode permintaan tinggi. Dasbor terbaru menunjukkan sebuah pola alokasi slot yang lebih stabil dan berkurang, dengan puncak yang sekarang hampir tidak mencapai sekitar



Gambar 6.4: Alokasi Slot Basis Data Sebelumnya tion



Gambar 6.5: Alokasi Slot Basis Data Saat Ini

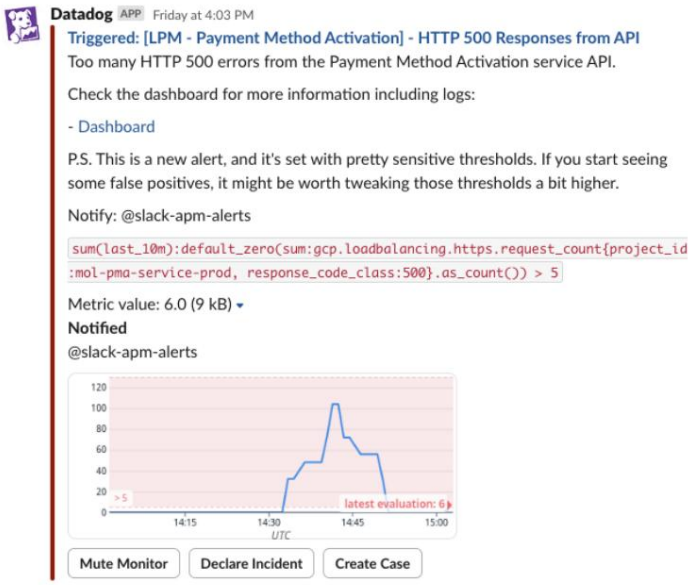
250 slot. Perubahan ini menunjukkan adanya peningkatan dalam distribusi beban kerja dan kueri.
Pengoptimalan telah mengurangi kebutuhan alokasi sumber daya yang berlebihan. Akibatnya, kueri
Latensi lebih rendah. Penggunaan slot yang lebih stabil telah berkontribusi pada pemrosesan kueri yang lebih cepat, yang pada gilirannya mendukung waktu respons API yang efisien. Alokasi slot yang lebih konsisten mencegah
pemborosan sumber daya dan memungkinkan basis data untuk menangani beban puncak dengan lebih mudah diprediksi.

6.2 Penanganan Kesalahan dan Peringatan dalam Arsitektur Berbasis Peristiwa ture

Meskipun kinerja platform secara keseluruhan telah meningkat, masih ada beberapa hal yang tidak terduga.
Kesalahan disebabkan oleh masalah kode atau operasi yang salah. Oleh karena itu, penting untuk mendeteksi kesalahan tersebut secara instan, terutama di lingkungan yang berisiko tinggi seperti platform pembayaran.
Untuk melakukan perbaikan tepat waktu seperti penggantian atau perbaikan cepat, deteksi kesalahan secara real-time sangat diperlukan. penting.

Untuk mencapai hal ini, kami menggunakan layanan notifikasi Slack, yang terintegrasi dengan Datadog, memungkinkan tim untuk menerima peringatan langsung untuk kesalahan kritis. Gambar 6.6 menunjukkan bagaimana Datadog
Bot memberi tahu tim tentang kesalahan HTTP 500 untuk API Aktivasi Metode Pembayaran. Slack
Peringatan tersebut menunjukkan bahwa hal itu berpotensi disebabkan oleh pengecualian yang tidak ditangani atau masalah di sisi server.
Dalam arsitektur berbasis peristiwa, penanganan kesalahan sangat penting untuk menjaga kelancaran komunikasi.
di antara layanan.

Untuk mengurangi kesalahan-kesalahan ini, mekanisme penanganan kesalahan diterapkan dengan membuat pencatatan (logging) yang detail. juga membantu menyelesaikan akar penyebab masalah dengan lebih cepat. Selain itu, menyiapkan peringatan yang lebih modular di Data-Anjing pelacak memungkinkan deteksi dini kesalahan tanpa adanya positif palsu yang berlebihan. Hal ini sebagian besar terjadi karena anjing pelacak. meningkatkan toleransi kesalahan dalam jangka panjang. Selain itu, metrik kunci seperti waktu pemuatan, tingkat kesalahan,
dan pemanfaatan sumber daya digunakan untuk mengidentifikasi hambatan kinerja dan meningkatkan sistem.



Gambar 6.6: Peringatan Datadog Slack untuk Kesalahan HTTP 500 pada API Aktivasi Metode Pembayaran

keandalan. Wawasan ini memvalidasi penerapan pemantauan dalam skenario dunia nyata dan selaras dengan temuan kami tentang mengintegrasikan strategi pemantauan hibrida untuk peningkatan kinerja (26).

7

Diskusi

Bab ini memberikan pembahasan komprehensif tentang refleksi utama proyek, dan membahas tantangan & keterbatasan, dan menguraikan arah potensial untuk pengembangan di masa depan dan peningkatan platform.

7.1 Refleksi

Transisi dari arsitektur monolitik ke arsitektur microservices dan event-driven telah secara signifikan-
Secara signifikan meningkatkan kinerja, skalabilitas, dan ketahanan platform pembayaran Mollie.
Dengan memisahkan sistem yang sangat besar, proyek ini berhasil dalam penskalaan independen dan peningkatan yang lebih besar.
Toleransi kesalahan. Penerapan pendekatan berbasis peristiwa dengan Apache Kafka semakin meningkatkan...
Mampu memproses data secara real-time dan asinkron, yang penting dalam menangani transaksi tinggi.
volume secara efisien. Hasilnya menggarisbawahi keunggulan microservices. Karena itu
Semakin tinggi modularitasnya, semakin baik isolasi kesalahan dan semakin singkat siklus pengembangannya.
Struktur modular memungkinkan platform untuk diterapkan lebih mudah dan merespons dengan cepat terhadap evolusi.
memenuhi permintaan pasar. Sementara itu, sistem berbasis peristiwa meningkatkan aliran data antara
layanan-layanan tersebut juga menurunkan latensi dan menjaga responsivitas sistem.

Dengan bantuan pemantauan waktu nyata menggunakan alat DataDog, hal ini sangat berharga.
untuk memberikan wawasan mendalam tentang kinerja sistem dan mendukung manajemen kesalahan proaktif.
Manajemen. Para insinyur mampu mendeteksi kesalahan atau kegagalan dalam waktu singkat, sehingga membantu.
Platform ini dirancang untuk menjaga ketersediaan tinggi dan meminimalkan waktu henti. Arsitektur ini
Peningkatan tersebut telah terbukti efektif dalam memenuhi tujuan inti proyek, yaitu secara substansial
Meningkatkan keandalan dan mendukung pertumbuhan berkelanjutan serta skalabilitas Mollie di masa depan.

7.2 Tantangan dan Keterbatasan

Meskipun perbaikan arsitektur telah membawa banyak manfaat, beberapa tantangan tetap ada.
Tantangan dan keterbatasan masih tetap ada. Banyak adaptasi telah diterapkan untuk melakukan dekopling.
arsitektur tanpa mengganggu fungsionalitas, namun beberapa masalah tetap ada (27). Salah satunya
Isu utamanya adalah duplikasi peristiwa dan pemrosesan di luar urutan. Peristiwa duplikat mengakibatkan pengulangan...
pemicu dundant, yang memengaruhi sistem hilir, sementara peristiwa yang tidak berurutan dapat menyebabkan
Ketidakkonsistenan dalam alur kerja. Meskipun kami mencoba menerapkan berbagai strategi anti-duplikasi-
Dengan mempertimbangkan strategi dan pengurutan berdasarkan stempel waktu, solusi-solusi ini terkadang tidak cukup, yang mana pro-
Hal ini menimbulkan kesalahan baru dan perlu kita perbaiki. Selain itu, hal ini menambah beban komputasi dan meningkatkan
kompleksitas. Kelemahan teknis lainnya adalah kinerja waktu nyata. Sistem pembayaran-
Sistem ini membutuhkan sifat waktu nyata yang memerlukan waktu respons API dan latensi penanganan peristiwa.
untuk tetap berada dalam ambang batas yang ketat. Ketergantungan eksternal tertentu, seperti API pihak ketiga,
Terkadang gagal memenuhi persyaratan ini, menyebabkan hambatan dalam alur kerja. Mencapai
Latensi yang konsisten tetap menjadi tantangan yang berkelanjutan. Dengan kata lain, satu respons tertunda.
Hal ini dapat menyebabkan komunikasi berikut menjadi lebih tertunda karena meningkatnya API internal.
komunikasi.

Di sisi lain, hal ini juga menghadirkan tantangan operasional seperti biaya sumber daya yang tinggi.
Karakteristik modular dari microservices menuntut sumber daya khusus untuk setiap layanan,
termasuk basis data terisolasi, pipeline penyebaran independen, dan konfigurasi pemantauan.
Pengaturan ini memberikan manfaat besar untuk skalabilitas dan isolasi kesalahan, tetapi juga
meningkatkan jejak sumber daya secara keseluruhan dan biaya operasional. Dengan kata lain,
Meskipun pendekatan modular menyederhanakan penskalaan dan fleksibilitas, mengintegrasikan beberapa metode pembayaran
Gateway dapat menimbulkan kompleksitas dalam menangani alur kerja yang beragam (28). Selain itu, transisi dari arsitektur
monolitik ke arsitektur microservices membutuhkan pelatihan yang substansial untuk
tim teknik. Meskipun modularitas memungkinkan tim untuk berspesialisasi di bidang tertentu, mereka
tidak mampu memahami bagaimana sistem bekerja secara keseluruhan. Sebaliknya, memahami hal-hal baru
paradigma seperti komunikasi berbasis peristiwa dan manajemen basis data terdistribusi mungkin
menyebabkan kesulitan dalam belajar.

Singkatnya, meskipun platform ini telah membuat kemajuan signifikan dalam mengatasi warisan masa lalunya,
Keterbatasan ini, tantangan-tantangan ini menyoroti area-area yang membutuhkan perhatian berkelanjutan. Mengatasi hal tersebut
Keterbatasan ini akan sangat penting untuk mempertahankan pertumbuhan, meningkatkan efisiensi operasional, dan
Memenuhi tuntutan pengguna dan mitra yang terus meningkat.

7.3 Pekerjaan di Masa Depan

Berdasarkan keterbatasan yang telah dibahas di atas, beberapa area untuk perbaikan dan arah masa depan diusulkan untuk lebih meningkatkan kinerja, skalabilitas, dan kemampuan adaptasi platform.

- Manajemen Acara Tingkat Lanjut: Mengembangkan algoritma yang lebih efisien dan ringan untuk penghapusan duplikasi peristiwa guna mengurangi beban komputasi. Memanfaatkan peristiwa unik. Pengklasifikasi atau penerapan protokol konsensus terdistribusi dapat membantu mengurangi masalah duplikasi dan urutan yang tidak sesuai tanpa mengorbankan kinerja. Untuk peristiwa dengan prioritas lebih tinggi, mekanisme dapat diterapkan untuk memprioritaskan peristiwa prioritas tinggi dalam antrian pemrosesan untuk memastikan alur kerja kritis mempertahankan latensi di bawah satu detik bahkan dalam kondisi beban puncak. Seperti yang disarankan oleh Petrov dkk. (29), kemajuan di masa depan dalam pemrosesan peristiwa dapat memperoleh manfaat signifikan dari integrasi kecerdasan buatan dan penyeimbangan beban adaptif.

- Alat Pengembang yang Ditingkatkan: Lingkungan staging tidak dapat mendukung beberapa API online, yang menyebabkan banyak masalah dalam melakukan pengujian setelah penerapan. Membangun alat untuk mensimulasikan interaksi antar layanan yang kompleks dalam lingkungan pengujian akan membantu pengembang mengidentifikasi dan menyelesaikan masalah dengan lebih cepat. Selain itu, dokumentasi yang komprehensif sangat penting bagi perusahaan seperti Mollie untuk berkembang saat ini. Untuk semua layanan mikro dan Alur kerja, dokumentasi yang detail dan terpusat harus digunakan untuk memfasilitasi proses yang lebih cepat. Membimbing para insinyur baru dan meningkatkan kolaborasi antar tim.

- Kinerja Waktu Nyata: Karena ketika kita mencoba menyelidiki kegagalan atau kesalahan transaksi tertentu, sulit untuk menemukan log terkait di GCP atau DataDog.
  Oleh karena itu, perlu untuk mendeteksi kinerja sistem secara real-time dengan informasi dan log yang detail yang dapat membantu menyelesaikan masalah dengan lebih efisien. Selain itu, untuk mengoptimalkan gateway API agar dapat menangani lalu lintas tinggi dengan latensi yang berkurang, kita dapat mencoba beberapa teknologi mutakhir seperti HTTP/3 dan pembatasan laju adaptif. Mekanisme percobaan ulang dan caching lokal untuk panggilan API pihak ketiga juga akan dipertimbangkan untuk meminimalkan dampak penundaan eksternal pada alur kerja internal.

7. DISKUSI

8

Kesimpulan

Makalah ini mengilustrasikan transisi dari arsitektur monolitik ke mikroserisasi modular.
kelemahan dan arsitektur berbasis peristiwa dalam konteks platform pembayaran frekuensi tinggi di
Mollie. Dengan menerapkan pendekatan modular, penelitian ini mengatasi tantangan utama terkait skalabilitas, toleransi kesalahan, dan efisiensi operasional, memastikan sistem tersebut sesuai untuk
Pemrosesan transaksi bervolume tinggi secara real-time. Melalui desain dan implementasi
Dengan arsitektur yang dioptimalkan, karya ini menunjukkan bagaimana komunikasi asinkron dan
Modularisasi meningkatkan kinerja sekaligus mengurangi latensi. Alat pemantauan waktu nyata.
seperti DataDog, diimplementasikan dengan Google Cloud Platform (30), lebih memfasilitasi proaktif
Manajemen kesalahan, berkontribusi pada keandalan dan kemudahan pemeliharaan sistem.

Temuan ini pertama-tama menunjukkan bahwa penerapan arsitektur microservices berhasil memisahkan fungsionalitas yang terintegrasi erat. Transisi ini memfasilitasi penskalaan independen.
dan modularitas yang ditingkatkan. Hasil kinerja DataDog mengkonfirmasi bahwa mikroser-
Arsitektur vices secara efektif mengatasi tantangan skalabilitas dan toleransi kesalahan. Kedua, arsitektur berbasis peristiwa (event-driven architecture/EDA) juga memainkan peran penting dalam mengoptimalkan waktu nyata.
pemrosesan dan pemisahan layanan. Dengan memanfaatkan Apache Kafka untuk komunikasi asinkron dan streaming peristiwa, platform ini mengurangi latensi, meningkatkan responsivitas, dan
Meminimalkan ketergantungan antar layanan. Selain itu, integrasi pemantauan waktu nyata-
Alat pemantauan seperti DataDog secara instan mencerminkan kesehatan sistem dan sangat meningkatkan kinerja sistem.
Keandalan. Mekanisme peringatan waktu nyata memungkinkan penyelesaian masalah secara proaktif, sementara jalur penerapan independen memungkinkan pembaruan tanpa gangguan layanan.
yang menjawab pertanyaan penelitian ketiga.

Terlepas dari peningkatan signifikan yang telah dicapai, penelitian ini menyoroti area-area yang perlu diperbaiki.
perhatian yang berkelanjutan, seperti duplikasi peristiwa, pengumpulan log kesalahan yang terbatas, dan eksternal
Kemacetan akibat ketergantungan API. Tantangan terkait beban sumber daya dan kompleksitasnya.

8. KESIMPULAN

Transisi menuju model microservices juga menggarisbawahi perlunya penyempurnaan berkelanjutan. (31). Ke depannya, temuan tesis ini mengusulkan solusi yang baik untuk inovasi.
Dalam arsitektur platform pembayaran, manajemen peristiwa tingkat lanjut, peningkatan alat bantu pengembang, dan pemantauan kinerja waktu nyata yang dioptimalkan menonjol sebagai area kunci untuk eksplorasi berkelanjutan. Peningkatan ini menawarkan wawasan berharga bagi akademisi dan industri dalam lanskap teknologi keuangan yang terus berkembang.

Referensi

[1] Situs Web Resmi Mollie BV. https://www.mollie.com/. Diakses: 2024-09-03.
1

[2] Aathira S Nair dan P Kannan. Metode Pembayaran Digital: Tantangan dan Peluang. Jurnal Studi Namibia: Sejarah Politik Budaya, 37:367–376,
Tahun 2023. 1

[3] Inc. Datadog. Datadog: Platform Pemantauan dan Keamanan untuk Aplikasi Cloud-aplikasi. Sumber Daya Online, 2024. Diakses: 2024-11-30. 2

[4] Aparna Naik, Janhavi Choudhari, Vedanti Pawar, dan Sanjay Shitole. Membangun platform edtech menggunakan microservices dan Docker. Pada tahun 2021 IEEE Konferensi Internasional Seksi Pune (PuneCon), halaman 1–6. IEEE, 2021. 3

[5] Paul Voigt dan Axel Von dem Bussche. Peraturan Perlindungan Data Umum Uni Eropa (GDPR). Panduan Praktis, Edisi ke-1, Cham: Springer International Publishing, 10(3152676):10–5555, 2017. 3

[6] Brenda M Michelson. Tinjauan arsitektur berbasis peristiwa. Patricia Seybold Grup, 2(12):10–1571, 2006. 3

[7] Jungho Kang. Pembayaran seluler di lingkungan Fintech: tren, tantangan keamanan, dan layanan. Komputasi dan Ilmu Informasi yang berpusat pada manusia, 8(1):32, 2018. 4

[8] Saulo Soares De Toledo, Antonio Martini, Agata Przybyszewska, dan Dag IK Sjøberg. Hutang teknis arsitektur dalam layanan mikro: studi kasus di perusahaan besar. Dalam Konferensi Internasional IEEE/ACM 2019 tentang Hutang Teknis (TechDebt), halaman 78–87. IEEE, 2019. 5

[9] Muhammad Waseem, Peng Liang, Mojtaba Shahin, Amleto Di Salle, dan Gastón Márquez. Desain, pemantauan, dan pengujian sistem microservices: Perspektif praktisi. Jurnal Sistem dan Perangkat Lunak, 182:111061, 2021. 7

[10] Nishant Garg. Apache kafka. Packt Publishing Birmingham, Inggris, 2013. 9

[11] Neha Narkhede, Gwen Shapira, dan Todd Palino. Kafka: panduan definitif: Pemrosesan data dan aliran data secara real-time dalam skala besar. " O'Reilly Media, Inc.", 2017. 10

[12] Henk Grent, Aleksei Akimov, dan Maurício Aniche. Mengidentifikasi batasan parameter secara otomatis dalam API web yang kompleks: Studi kasus di adyen. Dalam Konferensi Internasional IEEE/ACM ke-43 tentang Rekayasa Perangkat Lunak: Rekayasa Perangkat Lunak dalam Praktik (ICSE-SEIP) tahun 2021, halaman 71–80. IEEE, 2021. 11

[13] Bob Maulana Adam, Adnan Rachmat Anom Besari, dan Mochamad Mobed Bachtiar. Desain sistem server backend berbasis pada REST API untuk sistem pembayaran tanpa uang tunai di komunitas ritel. Pada tahun 2019. Simposium Elektronika Internasional (IES), halaman 208–213. IEEE, 2019. 13

[14] Ersin Ünsal, Bilgehan Öztekin, Murat Çavuÿ, dan Suat Özdemir. Membangun-Membangun ekosistem fintech: Desain dan pengembangan gerbang API fintech dengan cara tersebut. Pada simposium internasional tahun 2020 tentang jaringan, komputer, dan komunikasi. (ISNCC), halaman 1–5. IEEE, 2020. 13

[15] Joop Aué, Maurício Aniche, Maikel Lobbezoo, dan Arie van Deursen. Studi eksplorasi tentang kesalahan dalam integrasi API web di perusahaan pembayaran skala besar. Dalam Prosiding Konferensi Internasional ke-40 tentang Rekayasa Perangkat Lunak: Rekayasa Perangkat Lunak dalam Praktik, halaman 13–22, 2018. 14

[16] Sreenivas Rao Vangala, Bharath Kasimani, dan Ravi Kiran Mallidi. Mi-Pendekatan Arsitektur Berbasis Peristiwa dan Streaming untuk Layanan Pembayaran dan Penyelesaian Perdagangan. Dalam Konferensi Internasional ke-2 tentang Teknologi Cerdas (CONIT) tahun 2022, halaman 1–6. IEEE, 2022. 15

[17] Shubham Vyas, Rajesh Kumar Tyagi, Charu Jain, dan Shashank Sahu. Evaluasi kinerja Apache Kafka – sebuah platform modern untuk streaming data waktu nyata. Dalam Konferensi Internasional ke-2 tentang Praktik Inovatif dalam Teknologi dan Manajemen (ICIPTM) tahun 2022, 2, halaman 465–470. IEEE, 2022. 16

[18] Ervin Djogic, Samir Ribic, dan Dzenana Donko. Monolitik hingga mikro-
Desain ulang platform integrasi berbasis peristiwa. Pada tahun 2018, Konferensi Internasional ke-41.
Konvensi tentang Teknologi Informasi dan Komunikasi, Elektronika dan Mikroelektronika (MIPRO),
halaman 1411–1414. IEEE, 2018. 16

[19] Shujing Li. Penelitian tentang desain sistem pembayaran elektronik keuangan-
perusahaan komersial. Pada tahun 2021, Konferensi Internasional ke-2 tentang E-Commerce dan Internet
Teknologi (ECIT), halaman 91–94. IEEE, 2021. 17

[20] Mathieu Acher, Anthony Cleve, Philippe Collet, Philippe Merle, Laurence Duchien, dan Philippe
Lahire. Ekstraksi dan evolusi model variabilitas arsitektur dalam sistem berbasis plugin. Pemodelan
Perangkat Lunak & Sistem,
13:1367–1394, 2014. 30

[21] Charanjot Singh, Nikita Seth Gaba, Manjot Kaur, dan Bhavleen Kaur.
Perbandingan berbagai alat CI/CD yang terintegrasi dengan platform cloud.
Konferensi Internasional ke-9 tentang Komputasi Awan, Ilmu Data & Rekayasa tahun 2019
(Confluence), halaman 7–12. IEEE, 2019. 32

[22] Walid Maalej dan Martin P Robillard. Pola pengetahuan dalam dokumentasi referensi API. Transaksi
IEEE tentang Rekayasa Perangkat Lunak, 39(9):1264–
1282, 2013. 34

[23] Xian Jun Hong, Hyun Sik Yang, dan Han Kim Muda. Analisis kinerja-
Analisis API RESTful dan RabbitMQ untuk aplikasi web microservice. Dalam
Konferensi Internasional Konvergensi Teknologi Informasi dan Komunikasi (ICTC) 2018, halaman
257–259. IEEE, 2018. 37

[24] Alberto Martin-Lopez, Sergio Segura, dan Antonio Ruiz-Cortés. Katalog ketergantungan antar-
parameter dalam API web RESTful. Dalam Layanan-
Komputasi Berorientasi: Konferensi Internasional ke-17, ICSOC 2019, Toulouse, Prancis,
28–31 Oktober 2019, Prosiding 17, halaman 399–414. Springer, 2019. 37

[25] Stavros Christodoulakis. Implikasi dari asumsi tertentu dalam basis data
Evaluasi kinerja. ACM Transactions on Database Systems (TODS),
9(2):163–186, 1984. 38

[26] Linh Pham. Pemantauan pengguna nyata untuk aplikasi web internal. 2020. 40

[27] Konrad Gos dan Wojciech Zabierowski. Perbandingan arsitektur microservice dan monolitik. Dalam Konferensi Internasional IEEE XVI tentang Teknologi dan Metode Perspektif dalam Desain MEMS (MEMSTECH) tahun 2020, halaman 150–153. IEEE, 2020. 42

[28] Muchsin Hisyam dan Ida Bagus Kertyayana Manuaba. Model Integrasi tentang Berbagai Gerbang Pembayaran untuk Skenario Pembayaran Terpisah Online. Dalam Konferensi Internasional Manajemen dan Teknologi Informasi (ICIMTech) 2022, halaman 122–126. IEEE, 2022. 42

[29] YUREVNA AVKSENTIEVA dan VLADIMIROVICH BRYUKHANOV. Bajingan-Masalah sewa dan metode pemrosesan peristiwa dalam sistem dengan arsitektur berbasis peristiwa. Jurnal Teknologi Informasi Teoretis dan Terapan, 99(9), 2021. 43

[30] Inc. Google. Google Cloud Platform: Infrastruktur dan Layanan yang Dapat Diperluas untuk Aplikasi Modern. Sumber Daya Online, 2024. Diakses: 2024-11-30. 45

[31] Chien-Chang Liu, Chien-Chang Huang, Chia-Wei Tseng, Yao-Tsung Yang, dan Li-Der Chou. Manajemen sumber daya layanan dalam komputasi tepi berdasarkan layanan mikro. Dalam Konferensi Internasional IEEE 2019 tentang Internet of Things Cerdas (SmartIoT), halaman 388–392. IEEE, 2019. 46

Lampiran