

EVENT-DRIVEN ARCHITECTURE

Bài tập vận dụng • 60 phút • 10 điểm • Không yêu cầu lập trình

Họ và tên: Lê Chí Hoàng Long..... MSSV: 23120294.....

TÌNH HUỐNG THỰC TẾ | ĐĂNG KÝ WORKSHOP

Khoa tổ chức workshop có **80 chỗ**. Sinh viên đăng ký qua ứng dụng và trả phí **100.000 đồng**. Hệ thống gồm các dịch vụ: **Registration** (đăng ký), **Seat** (giữ chỗ), **Payment** (thanh toán), **Notification** (gửi email) và **Analytics** (thống kê).

Quy tắc: Chỉ xác nhận đăng ký khi đã giữ chỗ và thanh toán thành công; thanh toán thất bại hoặc quá hạn thì phải giải phóng chỗ đã giữ. Email chỉ gửi sau khi xác nhận; Analytics gặp lỗi không được làm gián đoạn đăng ký. Một thông điệp có thể được giao lại khi hệ thống thử gửi lần nữa.

Ví dụ cách ghi luồng (không phải đáp án): *BookReturned* → *Fine Service* → *FineUpdated* (hệ thống thư viện).

01. Nhận diện sự kiện

10 phút • 2,0 điểm

- Đề xuất 01 initiating event và 02 processing events cho tình huống. Với mỗi event, nêu: điều gì đã xảy ra, dịch vụ phát và dịch vụ nhận/xử lý. Chọn 01 event, chỉ ra 02 trường dữ liệu cần có và giải thích lý do. Initiating event:
 - Initiating event: RegistrationRequest** - Sinh viên nhấn "Đăng ký"; đơn được tạo, trạng thái PENDING. Registration service nhận yêu cầu và phát sự kiện. Mediator nhận sự kiện để điều phối.
 - Processing events:**
 - + **SeatReversed** - Seat Service giữ 1 chỗ thành công và phát sự kiện. Mediator nhận kết quả để biết giữ chỗ đã thành công.
 - + **PaymentSucceeded** - Payment Service xác nhận đã thu 100.000 đồng thành công và phát sự kiện. Mediator nhận kết quả để biết thanh toán đã thành công.
 - Event chọn: PaymentSucceeded** - 2 trường dữ liệu cần có:
 - + **registrationId**: để Mediator xác nhận đúng đơn và khử trùng lặp khi event bị giao lại.
 - + **paymentId**: để định danh giao dịch để đối chiếu và phát hiện xử lý trùng.

02. Lựa chọn kiến trúc

15 phút • 2,5 điểm

- Bạn ưu tiên Broker topology hay Mediator topology? Nêu 02 lý do từ yêu cầu nghiệp vụ; phân tích 01 đánh đổi (trade-off) của phương án chọn và 01 khó khăn nếu dùng phương án còn lại. Có thể chọn phương án khác nhau nếu lập luận hợp lý.

Chọn Mediator topology.

Hai lý do từ nghiệp vụ:

- Với quy tắc "chỉ xác nhận khi đã giữ chỗ và thanh toán thành công" thì ta cần một nơi tập trung nắm kết quả của cả hai bước để quyết định. Do đó Mediator được lựa chọn để theo dõi kết quả của từng bước từ đó quyết định khi nào xác nhận đăng ký.
- Với quy tắc "thất bại thanh toán hoặc quá hạn thì phải giải phóng chỗ", việc bù trừ phải dựa trên trạng thái từng đơn (đã giữ, chưa xác nhận); Mediator theo dõi trạng thái này nên ra lệnh ReleaseSeat và quản lý timeout ở một nơi.

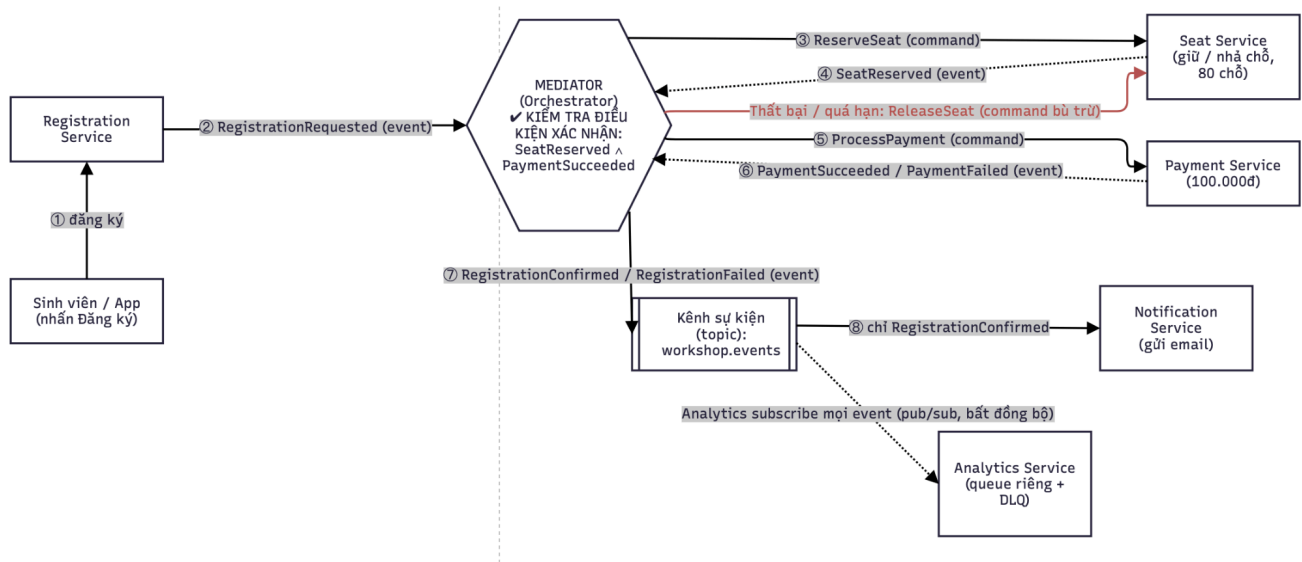
Đánh đổi của Mediator: Mediator trở thành điểm nghẽn và điểm lỗi đơn; các service phụ thuộc vào nó nên coupling cao hơn, khó mở rộng độc lập.

Khó khăn nếu dùng Broker: không service nào nắm trạng thái tổng; phải tự ghép SeatReserved và PaymentSucceeded, bù trừ rài rác ở nhiều service, khó theo dõi và gỡ lỗi.

03. Thiết kế luồng xử lý

20 phút • 3,0 điểm

- Vẽ sơ đồ từ lúc nhấn "Đăng ký" đến khi gửi email: có các service, kênh sự kiện/hàng đợi (nếu dùng), mũi tên truyền event/command. Đánh dấu nơi kiểm tra điều kiện xác nhận và cách Analytics nhận thông tin. Viết 02 câu giải thích vì sao luồng của bạn vận hành được.



Giải thích vì sao luồng vận hành được: Mediator điều phối thứ tự các bước và kiểm tra đủ hai điều kiện trước khi xác nhận, vì vậy không xác nhận đăng ký khi chưa giữ chỗ hoặc chưa thanh toán thành công. Analytics nhận sự kiện độc lập nên lỗi thống kê không làm gián đoạn đăng ký; các consumer vẫn cần xử lý message giao lại theo cách idempotent.

04. Phân tích sự cố

15 phút • 2,5 điểm

- (a) Seat đã giữ chỗ nhưng Payment không phản hồi trong 30 giây: đăng ký phải ở trạng thái nào và xử lý chỗ ra sao? (b) Yêu cầu đăng ký bị nhận lặp hai lần: rủi ro gì và phòng tránh thế nào? Mỗi ý trình bày theo “nguyên cơ → xử lý → lý do”.

(a) Payment không phản hồi sau 30 giây

- Nguyên cơ:** chiếm chỗ vô thời hạn; hoặc nhả chỗ quá sớm trong khi thanh toán thực ra đã thành công.
- Xử lý:** đăng ký ở trạng thái PENDING_PAYMENT (chưa xác nhận); Mediator truy vấn/thử lại trạng thái thanh toán. Hết hạn giữ chỗ mà vẫn không có kết quả → EXPIRED và gửi ReleaseSeat. Nếu PaymentSucceeded đến muộn sau khi đã nhả chỗ → hoàn tiền.
- Lý do:** đúng quy tắc (không xác nhận khi chưa thanh toán, quá hạn thì giải phóng chỗ), tránh mất chỗ của người khác và mất tiền người đăng ký.

(b) Yêu cầu đăng ký bị nhận lặp hai lần

- Nguyên cơ:** giữ 2 chỗ cho 1 sinh viên, thu tiền 2 lần, gửi 2 email.
- Xử lý:** idempotency key (registrationId/requestId) trên mọi event/command; consumer lưu id đã xử lý và bỏ qua bản trùng; ràng buộc duy nhất (studentId, workshopId) ở Seat.
- Lý do:** thông điệp có thể được giao lại (at-least-once), nên xử lý lặp phải cho cùng kết quả như xử lý một lần.

TIÊU CHÍ CHẤM ĐIỂM

C1 • 2,0 đ 03 event phù hợp (0,75); đúng vai trò các service (0,75); payload và lý do (0,50).

C2 • 2,5 đ Lựa chọn và 02 lý do bám yêu cầu (1,25); đánh đổi (0,75); so sánh phương án còn lại (0,50).

C3 • 3,0 đ Service/kênh/hướng truyền rõ (0,75); luồng và điều kiện xác nhận đúng (1,50); Analytics và giải thích (0,75).

C4 • 2,5 đ Timeout: trạng thái và giải phóng chỗ có lý do (1,25); thông điệp trùng: rủi ro và biện pháp phù hợp (1,25).

QUY ĐỊNH SỬ DỤNG AI Được tra cứu ý nghĩa từ khóa (event, broker, mediator, idempotency...) hoặc ví dụ khác; không dùng AI để tạo trọn đáp án hoặc sơ đồ cho đề này. Tự giải thích và bảo vệ quyết định. Nếu có dùng AI, cuối bài ghi 1-3 từ khóa đã tra và diễn giải ngắn bằng lời của mình.

THAM KHẢO Slide EDA: trang 4-6, 11-13, 18-21 | [Microsoft Learn: EDA](#) | [AWS: Coordination](#) | [AWS: Pub/Sub](#)