

Chương này trình bày các giải pháp kỹ thuật chuyên sâu và những đóng góp nổi bật nhất của đề án trong việc giải quyết các thách thức của bài toán ứng dụng giao đồ ăn, đặc biệt là tính năng đặt hàng nhóm (TeamCart). Ba đóng góp chính được phân tích bao gồm: (i) việc áp dụng kiến trúc Clean Architecture kết hợp Domain-Driven Design để quản lý nghiệp vụ phức tạp, (ii) giải pháp xử lý đồng thời lạc quan (Optimistic Concurrency Control) giúp đảm bảo tính nhất quán dữ liệu, và (iii) kiến trúc đồng bộ hóa thời gian thực tối ưu hiệu năng. Các nội dung ở Chương 4 chỉ mô tả tổng quan, còn chi tiết giải pháp được trình bày tại đây.

0.1 Ứng dụng Clean Architecture và Domain-Driven Design trong quản lý nghiệp vụ phức tạp

Vấn đề

Trong quá trình phát triển các hệ thống thương mại điện tử hiện đại, đặc biệt là các tính năng đòi hỏi sự cộng tác cao như Giỏ hàng nhóm, logic nghiệp vụ thường trở nên phức tạp với hàng loạt các quy tắc về tính toán giá, chia sẻ chi phí và quản lý trạng thái. Nếu logic nghiệp vụ bị trộn lẫn với mã xử lý hạ tầng (cơ sở dữ liệu, API, giao diện), mã nguồn khó kiểm soát tính toàn vẹn dữ liệu và khó kiểm thử, đặc biệt ở các luồng tài chính nhạy cảm.

Giải pháp

Đề án áp dụng kiến trúc Clean Architecture kết hợp với DDD nhằm tách biệt triệt để logic nghiệp vụ khỏi hạ tầng. Lớp miền được tổ chức theo mô hình giàu hành vi, trong đó các aggregate như TeamCart tự quản lý các thao tác nghiệp vụ quan trọng (ví dụ LockForPayment, AddItem) và các bất biến dữ liệu. Các Value Object như Money và TeamCartId giúp đóng gói quy tắc xác thực ở mức nhỏ, hạn chế lỗi dữ liệu ngay tại biên của miền nghiệp vụ. Những quyết định này được hiện thực hóa rõ nhất ở các aggregate và dịch vụ miền được mô tả trong Chương 4, và được sử dụng trực tiếp trong các ca sử dụng TeamCart.

Kết quả

Giải pháp giúp giảm sự phụ thuộc chặt giữa nghiệp vụ và hạ tầng, từ đó tăng khả năng kiểm thử độc lập cho các kịch bản tính toán giá, phân bổ phí và chuyển trạng thái. Mã nguồn bám sát ngôn ngữ chung của nghiệp vụ, giúp đội ngũ dễ bảo trì và giảm rủi ro khi thay đổi quy tắc trong tương lai.

0.2 Giải pháp xử lý cạnh tranh dữ liệu với Optimistic Concurrency Control

Vấn đề

Một trong những thách thức lớn của TeamCart là xử lý đồng thời khi nhiều thành viên cùng thao tác trên một giỏ hàng tại cùng thời điểm. Các xung đột như cập nhật

số lượng và xóa món có thể gây mất cập nhật hoặc tạo trạng thái không nhất quán nếu không được kiểm soát. Cơ chế khóa bi quan ở cấp cơ sở dữ liệu tuy an toàn nhưng dễ làm giảm hiệu năng và gây độ trễ lớn cho trải nghiệm cộng tác.

Giải pháp

Giải pháp được lựa chọn là sử dụng Redis làm kho lưu trữ trạng thái thời gian thực, kết hợp cơ chế Optimistic Concurrency Control. Mỗi bản ghi TeamCart trong Redis mang một trường `Version`. Khi cập nhật, hệ thống đọc trạng thái hiện tại, áp dụng thay đổi, tăng `Version`, sau đó ghi lại bằng giao dịch có điều kiện (Check-And-Set) để đảm bảo dữ liệu chỉ được ghi khi trạng thái gốc chưa bị thay đổi. Nếu phát hiện xung đột, hệ thống tự động thử lại với khoảng trễ ngẫu nhiên (jittered backoff) nhằm giảm va chạm ở các thời điểm cao điểm.

Kết quả

Giải pháp giúp giảm nguy cơ mất cập nhật khi nhiều người dùng cùng thao tác, đồng thời hạn chế độ trễ do không cần khóa cứng tài nguyên. Việc tự động thử lại giúp duy trì trải nghiệm mượt mà và giảm tỷ lệ lỗi hiển thị ở các thao tác cộng tác.

0.3 Kiến trúc đồng bộ hóa thời gian thực hiệu năng cao

Vấn đề

Tính năng Giỏ hàng nhóm yêu cầu đồng bộ trạng thái tức thì giữa các thiết bị của thành viên. Mô hình hỏi vòng (polling) truyền thống gây lãng phí băng thông, tăng tải máy chủ và khó đạt độ trễ thấp khi nhiều người dùng thao tác đồng thời. Ngoài ra, khi triển khai trên nhiều máy chủ, hệ thống cần cơ chế để các kết nối ở các nút khác nhau vẫn nhận được cùng một sự kiện.

Giải pháp

Đồ án triển khai giao tiếp thời gian thực dựa trên SignalR với WebSockets để thiết lập kênh kết nối hai chiều ổn định. Các thành viên được gắn vào nhóm theo `TeamCartId`, từ đó mọi sự kiện thay đổi đều được phát đến đúng nhóm. Trạng thái TeamCart thời gian thực được lưu ở Redis; khi dữ liệu thay đổi, hệ thống phát thông báo nhẹ để máy khách đồng bộ lại trạng thái. Redis cũng được sử dụng để phát tán sự kiện cập nhật giữa các nút, giúp hệ thống hoạt động ổn định khi mở rộng nhiều máy chủ.

Kết quả

Giải pháp giúp giảm độ trễ trong cập nhật giỏ hàng nhóm, tăng tính nhất quán giữa các thiết bị và hỗ trợ mở rộng theo chiều ngang. Nhờ cơ chế thông báo nhẹ và lưu trạng thái tập trung, hệ thống giảm tải truy vấn lặp và cải thiện trải nghiệm cộng tác so với polling.

0.4 Kết chương

Chương này đã trình bày chi tiết ba giải pháp kỹ thuật và đóng góp quan trọng nhất của đồ án. Việc áp dụng thành công kiến trúc Clean Architecture kết hợp DDD đã tạo nên nền tảng vững chắc cho việc quản lý nghiệp vụ phức tạp. Giải pháp Optimistic Concurrency Control với Redis giải quyết triệt để bài toán cạnh tranh dữ liệu trong môi trường đa người dùng. Cuối cùng, kiến trúc đồng bộ hóa thời gian thực tối ưu đã mang lại trải nghiệm người dùng mượt mà và khả năng mở rộng hệ thống linh hoạt. Những giải pháp này không chỉ giải quyết các vấn đề hiện tại của dự án mà còn đặt nền móng cho việc phát triển các tính năng nâng cao trong tương lai, được tóm tắt và thảo luận trong Chương 6 kế tiếp.