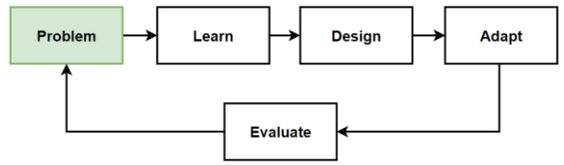
BÀI 8 – CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI

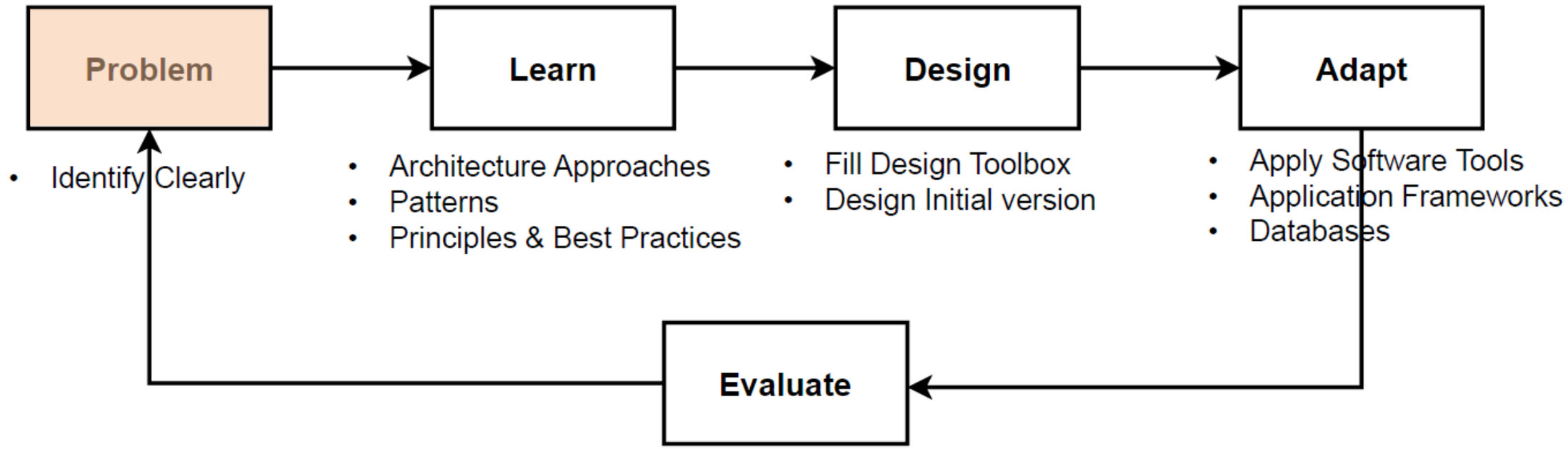
Singple Page Application (SPA)

Lợi ích và thách thức của SPA
Thiết kế ứng dụng thương mại điện tử với SPA

TS. Đỗ Như Tài
Đại Học Sài Gòn
dntai@sgu.edu.vn

Quy trình Thiết kế Kiến trúc Phần Mềm

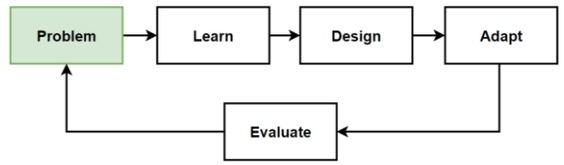




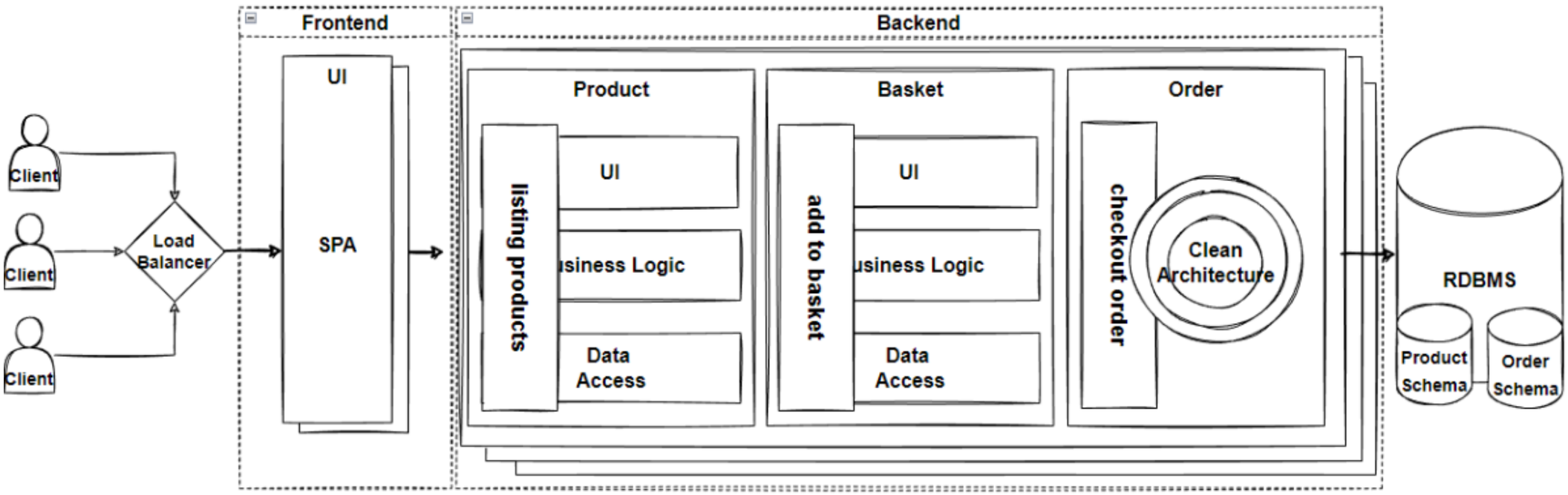
- What's wrong with this architecture ?
- how we can improve this current design ?

Quy trình Thiết kế Kiến trúc Phần Mềm

Vấn đề: Cải thiện trải nghiệm khách hàng với SPA

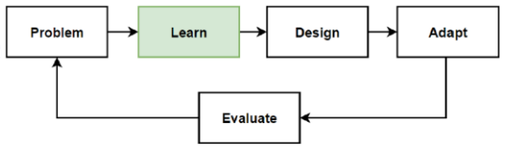


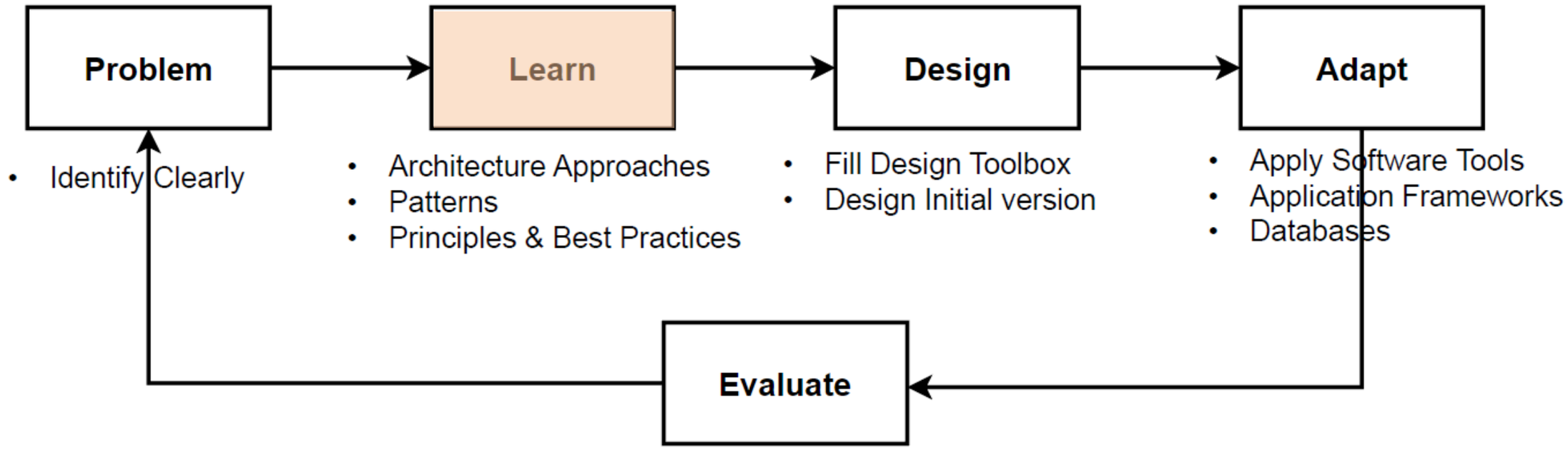
- **Các vấn đề**:
  - Doanh nghiệp thương mại điện tử đang phát triển.
  - Cần cải thiện trải nghiệm khách hàng với giao diện người dùng (UI) tách biệt và đa kênh (Omnichannel)
  - Các trang phản hồi nhanh (Responsive Pages) sử dụng SPA (Single Page Application)
  - Khách hàng kỳ vọng trải nghiệm đa kênh (Omnichannel)
- **Giải pháp**:
  - Tách riêng phần trình bày với SPA
  - Tách biệt Giao diện người dùng (FrontEnd) và Hệ thống xử lý (BackEnd)
  - Kiến trúc Headless (không phụ thuộc giao diện cụ thể)



Tăng trải nghiệm người dung với SPA

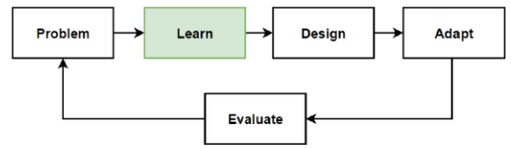
Quy trình Thiết kế Kiến trúc Phần Mề



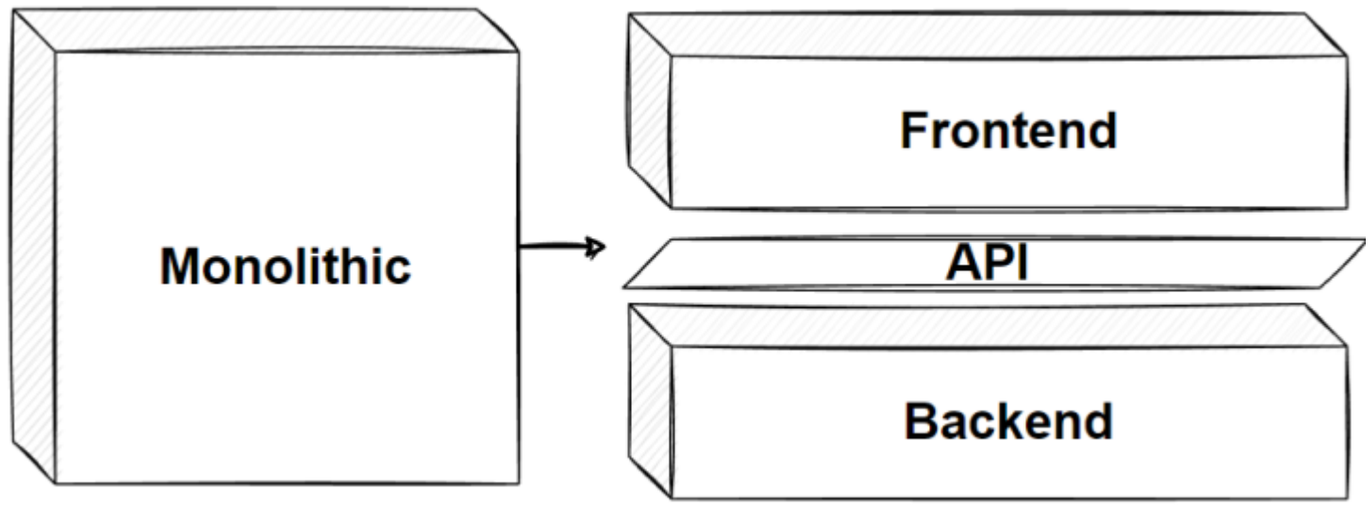


- What's wrong with this architecture ?
- how we can improve this current design ?

Kiến trúc Headless và Giao diện UI-SPA tách biệt

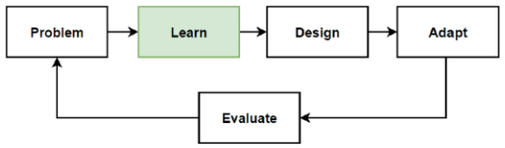


- **Kiến trúc Headless** tách **phần giao diện người dùng (frontend)** khỏi **phần xử lý phía sau (backend)** của ứng dụng. Tách biệt giữa giao diện và logic nghiệp vụ.

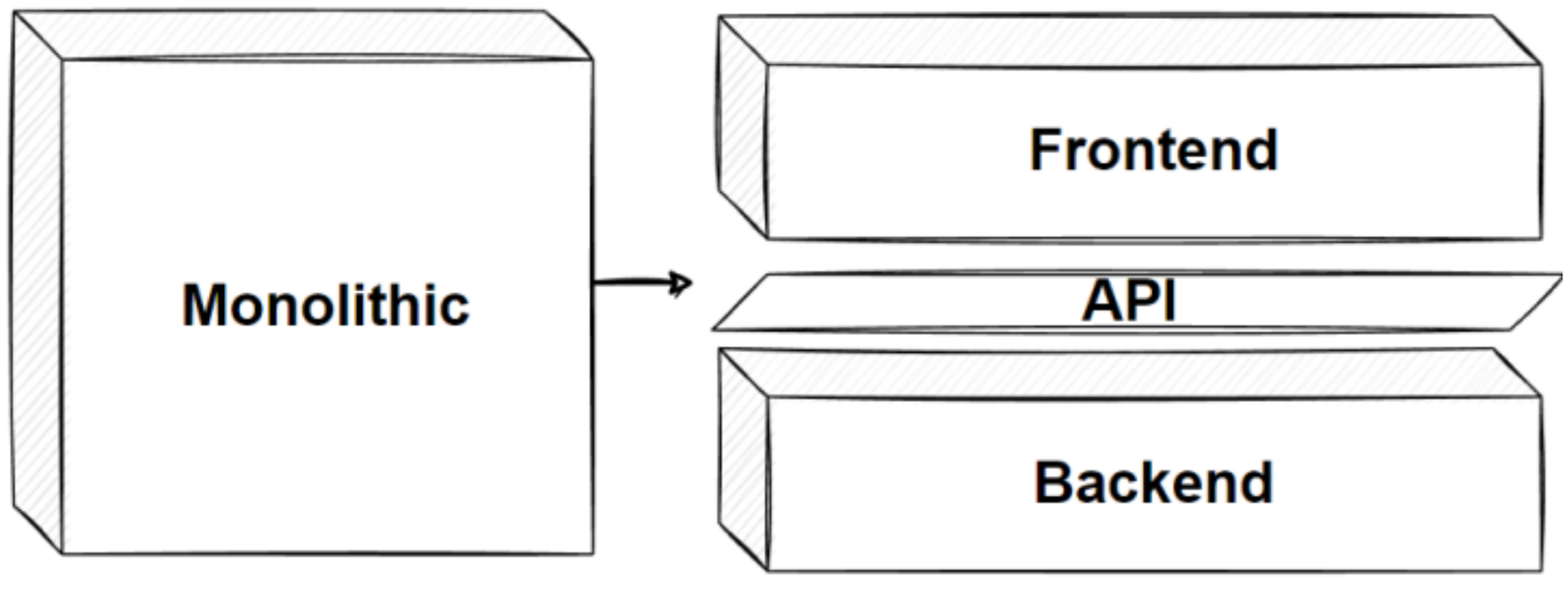


- Kiến trúc Headless nhấn mạnh vào việc **tách rời frontend và backend**, và đây là bước đầu tiên trước khi chuyển sang mô hình microservices.
- Kiến trúc này sử dụng **API** để kết nối giữa ứng dụng frontend và backend.
- **Giao diện lập trình ứng dụng (API)** là phần mềm trung gian giúp giao tiếp giữa các ứng dụng với nhau.
- **REST APIs** được tạo ra từ ứng dụng backend sẽ được sử dụng bởi ứng dụng frontend. Việc gọi API này được thực hiện trong ứng dụng **SPA** ở phía frontend.
- **Ứng dụng một trang (SPA)** là các ứng dụng chạy trong **một trang web duy nhất**, không cần tải lại toàn bộ trang, và có thể cập nhật **một phần của trang một cách linh hoạt** (responsive).

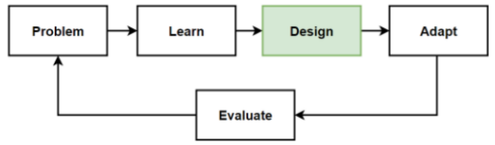
Lợi ích của Kiến trúc Headless

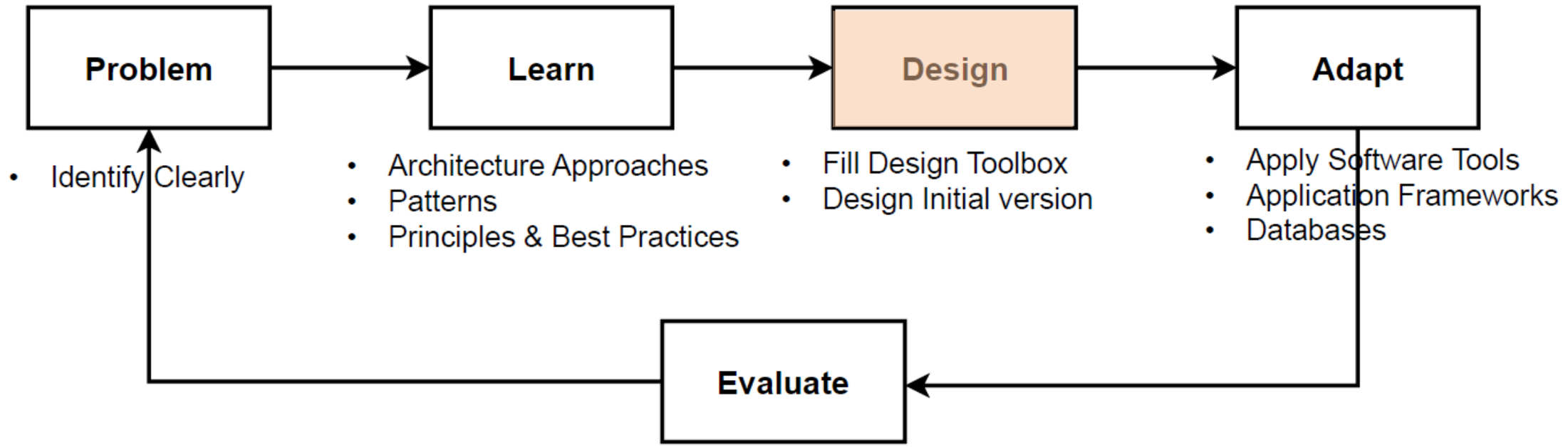


- Kiến trúc headless **tách biệt phần giao diện người dùng (frontend) khỏi phần xử lý phía sau (backend)** của ứng dụng. Tách biệt giữa UI và logic nghiệp vụ.
- **Linh hoạt sử dụng bất kỳ framework giao diện nào** hoặc tiếp cận hiện đại trong phát triển web.
- **Dễ dàng chia sẻ dịch vụ** giữa tất cả các kênh khác nhau.
- **Cập nhật giao diện một cách độc lập** với các dịch vụ phía sau.
- **Đổi mới và thử nghiệm** với các kênh mới.
- Hiệu suất tốt hơn



Quy trình Thiết kế Kiến trúc Phần Mềm

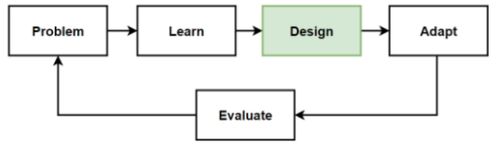




- What's wrong with this architecture ?
- how we can improve this current design ?

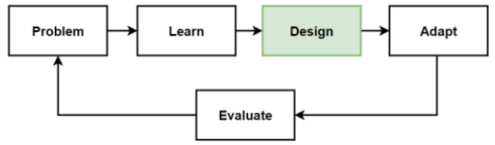
Quy trình Thiết kế Kiến trúc Phần Mềm

Trước khi thiết kế – Chúng ta có gì trong hộp công cụ thiết kế?

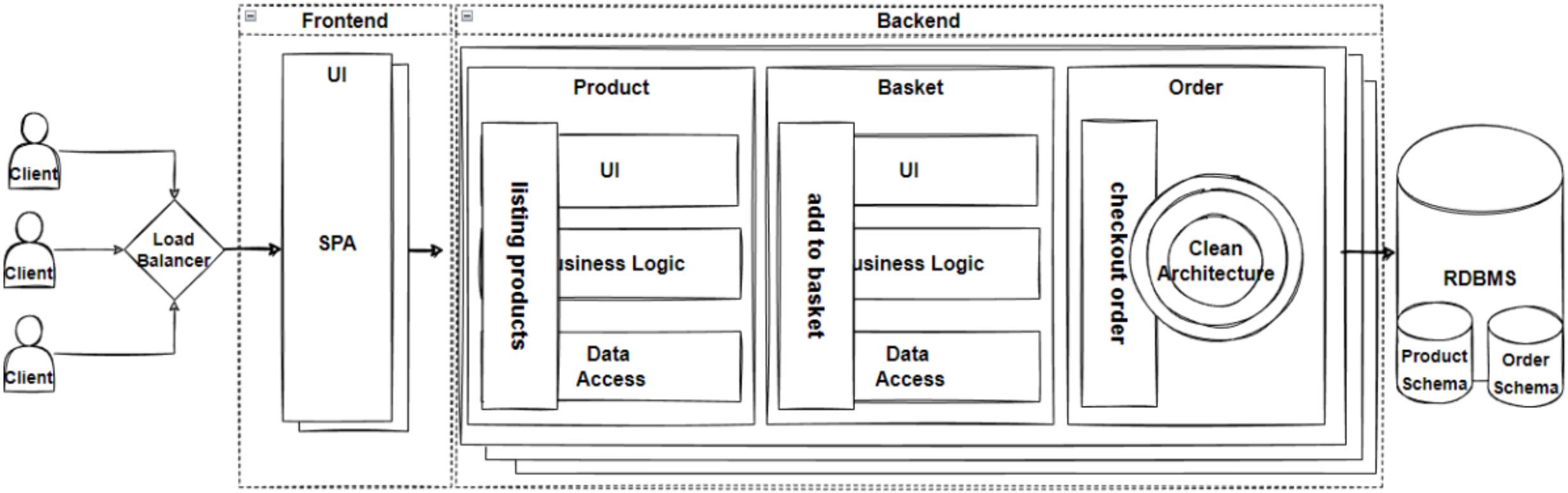


Kiến trúc	Mẫu & Nguyên tắc	Yêu cầu phi chức năng	Yêu cầu chức năng
• Kiến trúc nguyên khối • Kiến trúc phân lớp • Kiến trúc Clean • **Kiến trúc nguyên mẫu module** • **Kiến trúc headless**	• DRY (Không lặp lại chính mình) • KISS (Giữ đơn giản, đừng phức tạp) • YAGNI (Bạn sẽ không cần nó) • Seperation of Concerns (SoC) • SOLID • Quy tắc phụ thuộc • **Ưu tiên tiếp cận nguyên khối** • **Phân tách UI**	• Độ sẵn sàng • Số lượng người dùng đồng thời nhỏ • Khả năng bảo trì • Tính linh hoạt • Dễ kiểm thử • Tính khả mở • **Độ tin cậy** • **Tính tái sử dụng**	• Liệt kê sản phẩm • Lọc sản phẩm theo thương hiệu và danh mục • Thêm sản phẩm vào giỏ hàng • Áp dụng mã giảm giá • Thanh toán giỏ hàng và tạo đơn hàng • Xem danh sách đơn hàng cũ và lịch sử các mặt hàng đã đặt

Thiết kế: Kiến trúc nguyên khối module với SPA

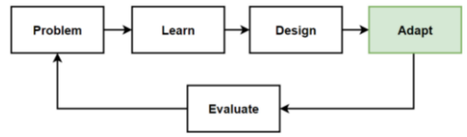


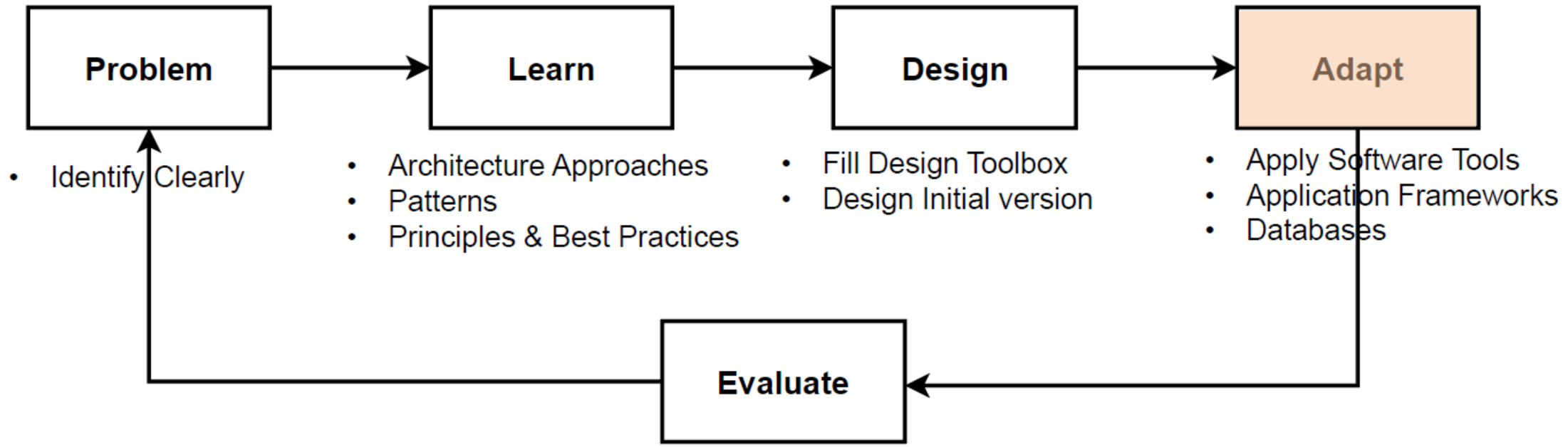
- **Client (Người dùng)** truy cập hệ thống thông qua **Load Balancer (Cân bằng tải)** để đảm bảo phân phối lưu lượng đều.
- **Frontend (Giao diện phía người dùng)** gồm:
  - **SPA (Ứng dụng một trang)**
  - **UI (Giao diện người dùng)**
- **Backend (Hệ thống xử lý phía sau)** gồm một ứng dụng nguyên khối nhưng được **tách thành các mô-đun riêng biệt**:
  - **Product**, **Order, Basket**, **Payment, Shipment**, **Reporting Module**
- Dữ liệu được quản lý tập trung qua **RDBMS**, nhưng có thể tách riêng theo **schema từng mô-đun** (ví dụ: Product Schema, Order Schema).



Kiến trúc nguyên khối module với SPA

Quy trình Thiết kế Kiến trúc Phần Mềm

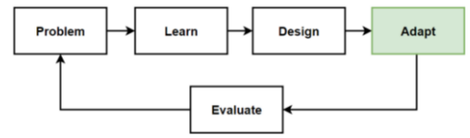




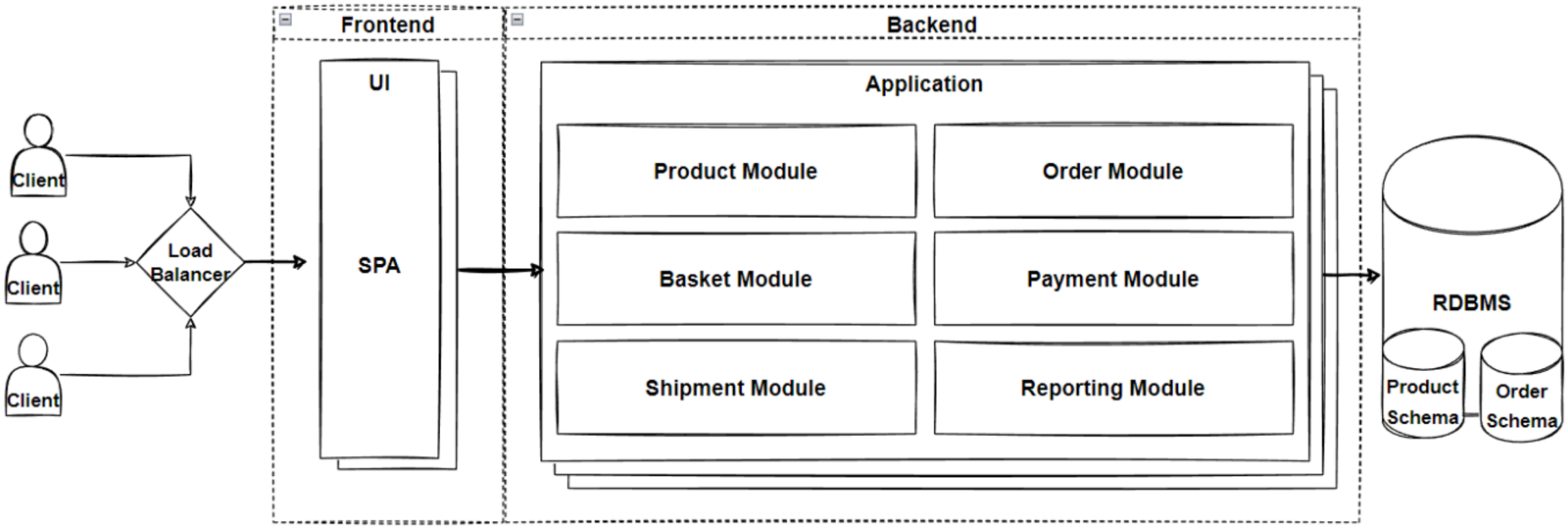
- What's wrong with this architecture ?
- how we can improve this current design ?

Quy trình Thiết kế Kiến trúc Phần Mềm

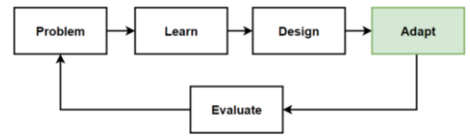
Triển khai: Kiến trúc nguyên khối module dung SPA



- **Cân bằng tải (Load Balancer):**
  - Apache LB
  - NGINX
- **Các ứng dụng giao diện một trang (Frontend SPAs):**
  - Angular, Vue, React
- **Ứng dụng Backend**
  - Một tệp JAR / WAR duy nhất; Chạy trên Tomcat Container
- **Cơ sở dữ liệu (Database):**
  - Oracle, Postgres, SQL Server

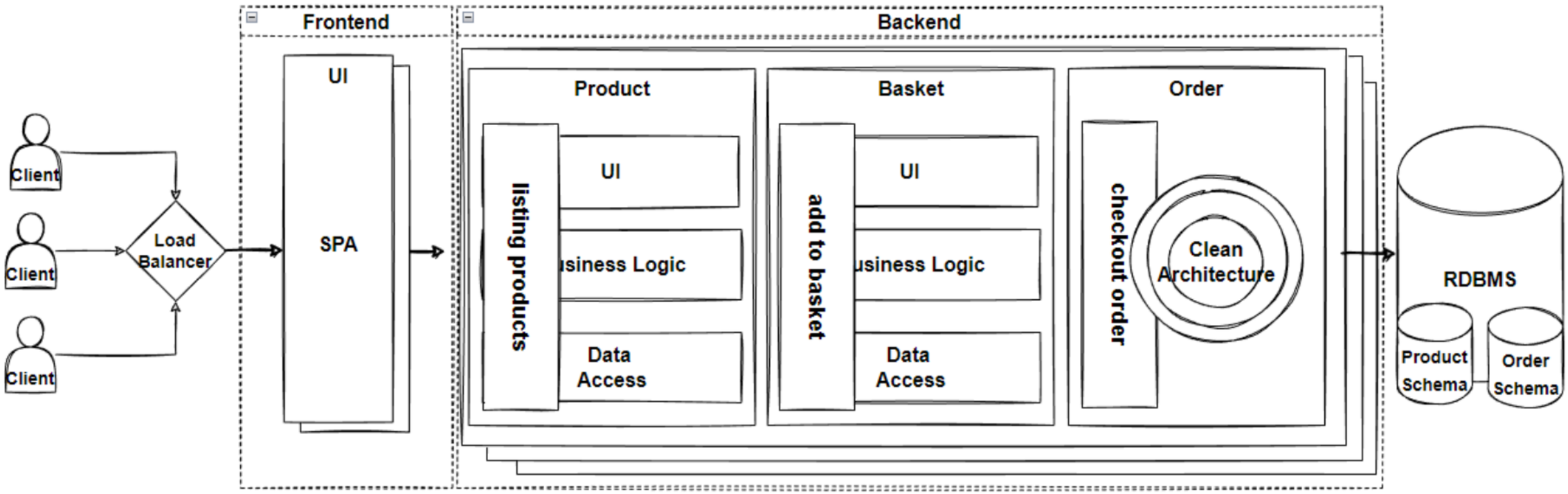


Kiến trúc nguyên khối module với SPA



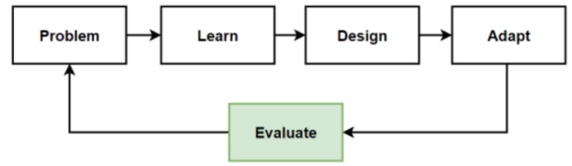
Minh họa: Kiến trúc Nguyên khối module dung SPA – Đánh giá mã nguồn

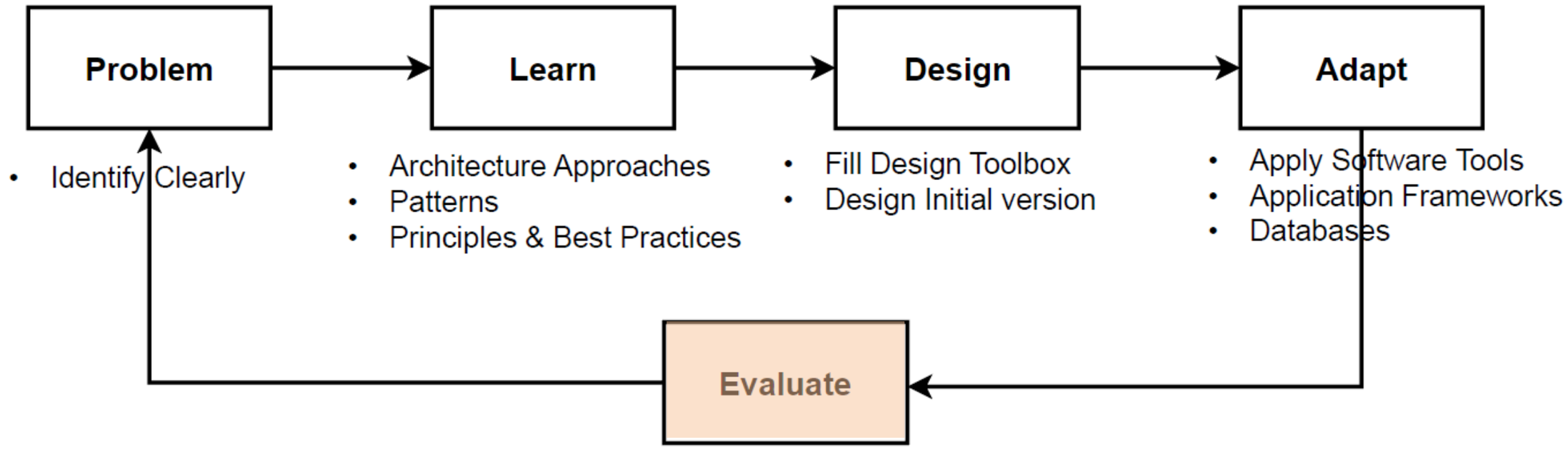
- **Minh họa**: Kamil Grzybek – Nguyên khối mô-đun với DDD
  - https://github.com/kgrzybek/modular-monolith-with-ddd
  - https://github1s.com/kgrzybek/modular-monolith-with-ddd



Kiến trúc nguyên khối module với SPA

Quy trình Thiết kế Kiến trúc Phần Mềm

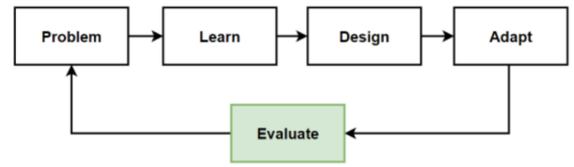




- What's wrong with this architecture ?
- how we can improve this current design ?

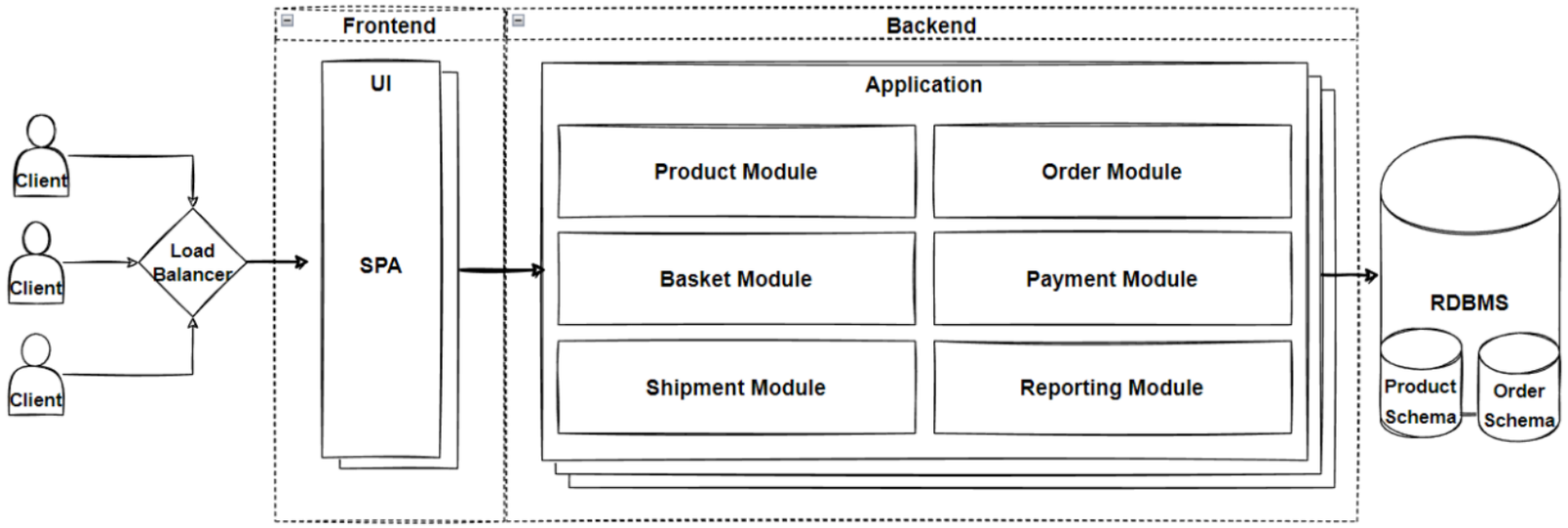
Quy trình Thiết kế Kiến trúc Phần Mềm

Đánh giá: Kiến trúc Nguyên khối module dùng SPA



Lợi ích

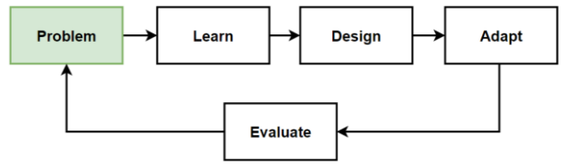
- Phát triển, gỡ lỗi và triển khai dễ dàng
- Đóng gói logic nghiệp vụ
- Mã có thể tái sử dụng, dễ dàng tái cấu trúc
- Quản lý phụ thuộc và phân chia nhóm tốt hơn
- Có thể cập nhật giao diện (frontend) độc lập, giao diện linh hoạt

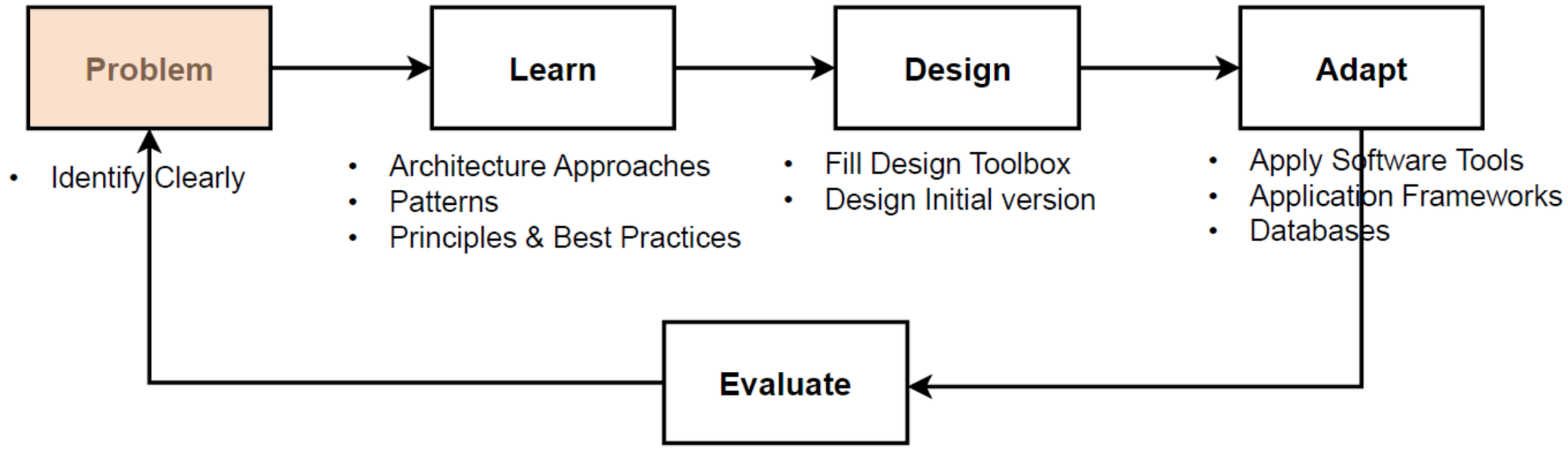


Hạn chế

- Giới hạn khả năng mở rộng, cơ sở dữ liệu không thể mở rộng
- Chúng tôi có một cơ sở dữ liệu quan hệ lớn, không thể mở rộng và trở thành điểm nghẽn của kiến trúc. Hàng triệu yêu cầu gây ra lỗi timeout.
- **Vẫn là kiến trúc nguyên khối (Monolithic) và gặp vấn đề về khả năng mở rộng**
- Không thể mở rộng các mô-đun một cách độc lập
- Không thể triển khai các mô-đun một cách độc lập
- **Vẫn là kiến trúc nguyên khối và gặp vấn đề về triển khai**

Quy trình Thiết kế Kiến trúc Phần Mềm

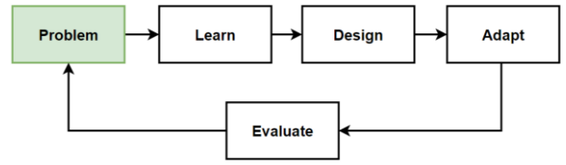




- What's wrong with this architecture ?
- how we can improve this current design ?

Quy trình Thiết kế Kiến trúc Phần Mềm

Vấn đề: Cải thiện trải nghiệm khách hàng với SPA

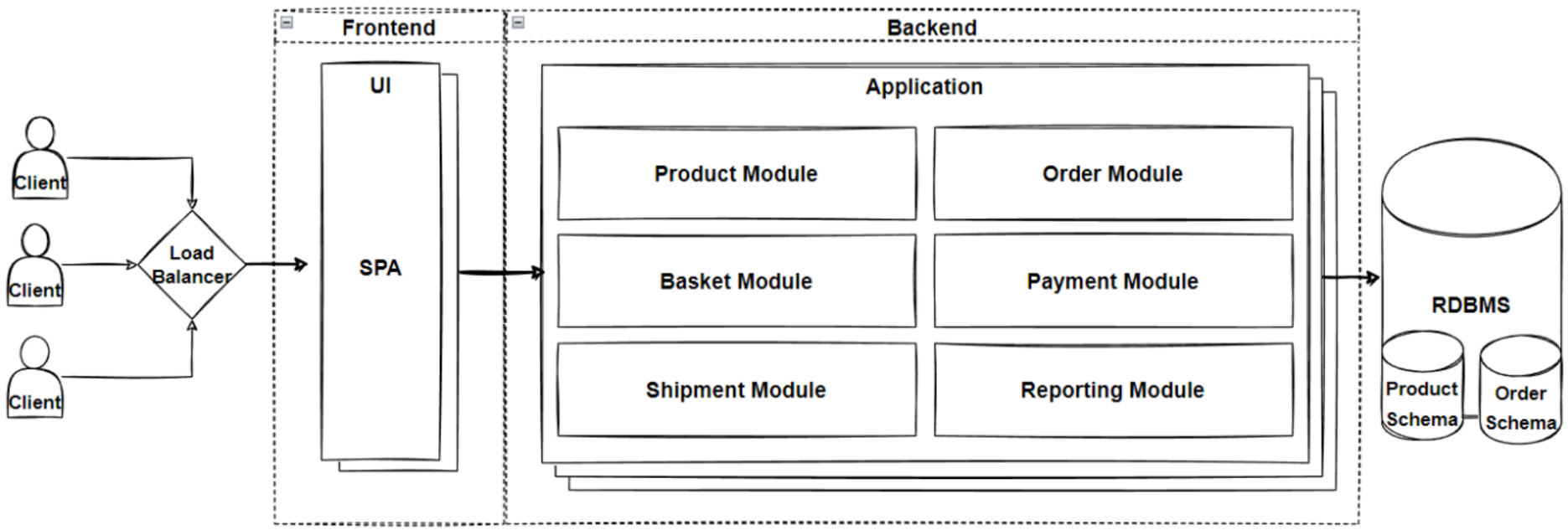


❖ Vấn đề hiện tại

- ❑ Doanh nghiệp TMĐT đang phát triển
- ❑ Các nhóm nghiệp vụ được tổ chức **tách biệt theo phòng ban**: Sản phẩm, Bán hàng, Thanh toán
- ❑ Các nhóm muốn **làm việc linh hoạt (agile)** và **thêm tính năng mới ngay lập tức** để cạnh tranh trên thị trường
- ❑ Liên tục đổi mới và thử nghiệm các tính năng mới càng sớm càng tốt
- ❑ **Triển khai tính năng ngay lập tức**, không phải chờ lịch triển khai
- ❑ **Mở rộng linh hoạt** trong các thời điểm cao điểm như đợt giảm giá Black Friday
- ❑ Xử lý hàng **triệu yêu cầu** với độ trễ chấp nhận được và hiệu suất cao hơn
- ❑ Không chỉ thay đổi công nghệ mà còn cần **thay đổi tổ chức là bắt buộc**

❖ Giải pháp đề xuất

- ❑ Kiến trúc Microservices



Kiến trúc nguyên khối module với SPA

CÁM ƠN ĐÃ CHÚ Ý LANG NGHE!