

# Phân tích thiết kế Hệ thống

Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2021

# Kiến trúc hệ thống

# Nội dung

- Phân mảnh và sơ đồ gói
- Sơ đồ thành phần và sơ đồ triển khai
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc

# Nội dung

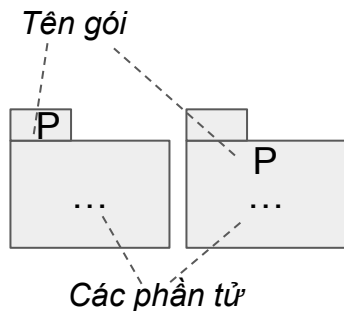
- Phân mảnh và sơ đồ gói
- Sơ đồ thành phần và sơ đồ triển khai
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc

# Phân mảnh

- Phân mảnh: Tạo hệ thống con với các lớp cộng tác.
- Các tương tác và các mối liên hệ có thể là cơ sở để xác định những lớp nào cần được gom cùng nhau
  - Các sơ đồ lớp
  - Các sơ đồ giao tiếp
    - Ví dụ, các thông điệp và liên kết trong sơ đồ giao tiếp
    - Nếu có nhiều thông điệp được gửi giữa 2 lớp thì đưa cả 2 lớp vào cùng 1 mảnh
  - Ma trận CRUDE
  - v.v..
- Tạo mảnh từ các lớp có bậc liên kết cao.

# Sơ đồ gói: Hệ ký hiệu

*Sơ đồ gói biểu diễn các gói cùng với các mối quan hệ.*

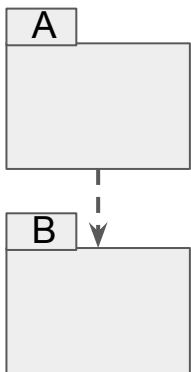


- **Gói/Package:**

- Nhóm lô-gic của nhiều phần tử, là không gian tên của các phần tử
- Có thể gom nhiều phần tử liên quan thành 1 thành phần bậc cao, giúp giản lược mô hình

- **Quan hệ phụ thuộc/Dependency:**

- Nếu B thay đổi thì A cũng sẽ thay đổi theo.
- A phụ thuộc vào B được biểu diễn bằng mũi tên nét đứt từ A tới B



# Sơ đồ gói: Hệ ký hiệu<sub>(2)</sub>

<<merge>>  
----->

- Quan hệ hợp nhất/Merge:
  - Nội dung của gói đích (theo mũi tên) được hợp nhất với nội dung của gói nguồn.
  - Các thành phần cùng tên cũng được hợp nhất.

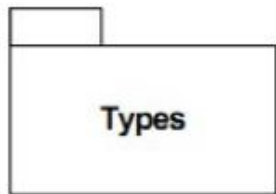
<<import>>  
----->

- Quan hệ nhập/Import
  - Thêm các phần tử của gói đích vào gói nguồn.
  - Các phần tử được thêm vào có thể được nhìn thấy từ bên ngoài / nhập công khai.

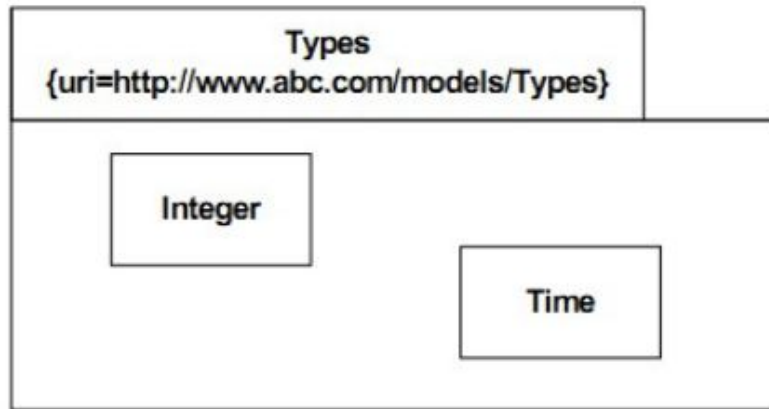
<<access>>  
----->

- Quan hệ truy cập/access
  - Thêm các phần tử của gói đích vào gói nguồn
  - Các phần tử được thêm vào không được nhìn thấy từ bên ngoài / nhập riêng tư

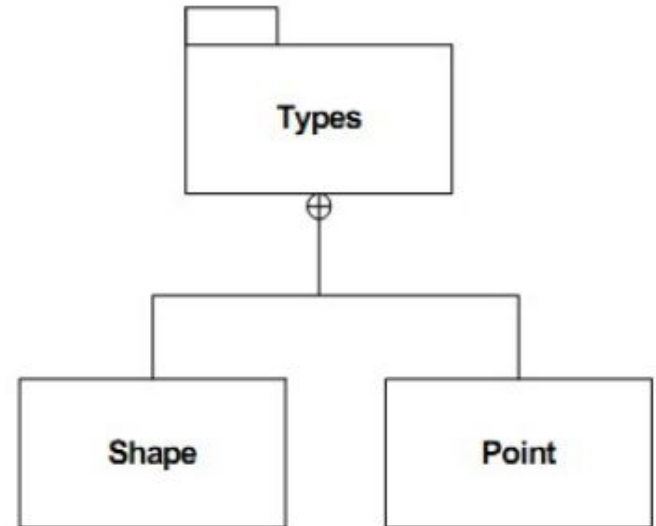
# Biểu diễn phần tử thuộc gói



Ẩn nội dung  
của gói Types



Các lớp Integer và Time thuộc gói  
Types

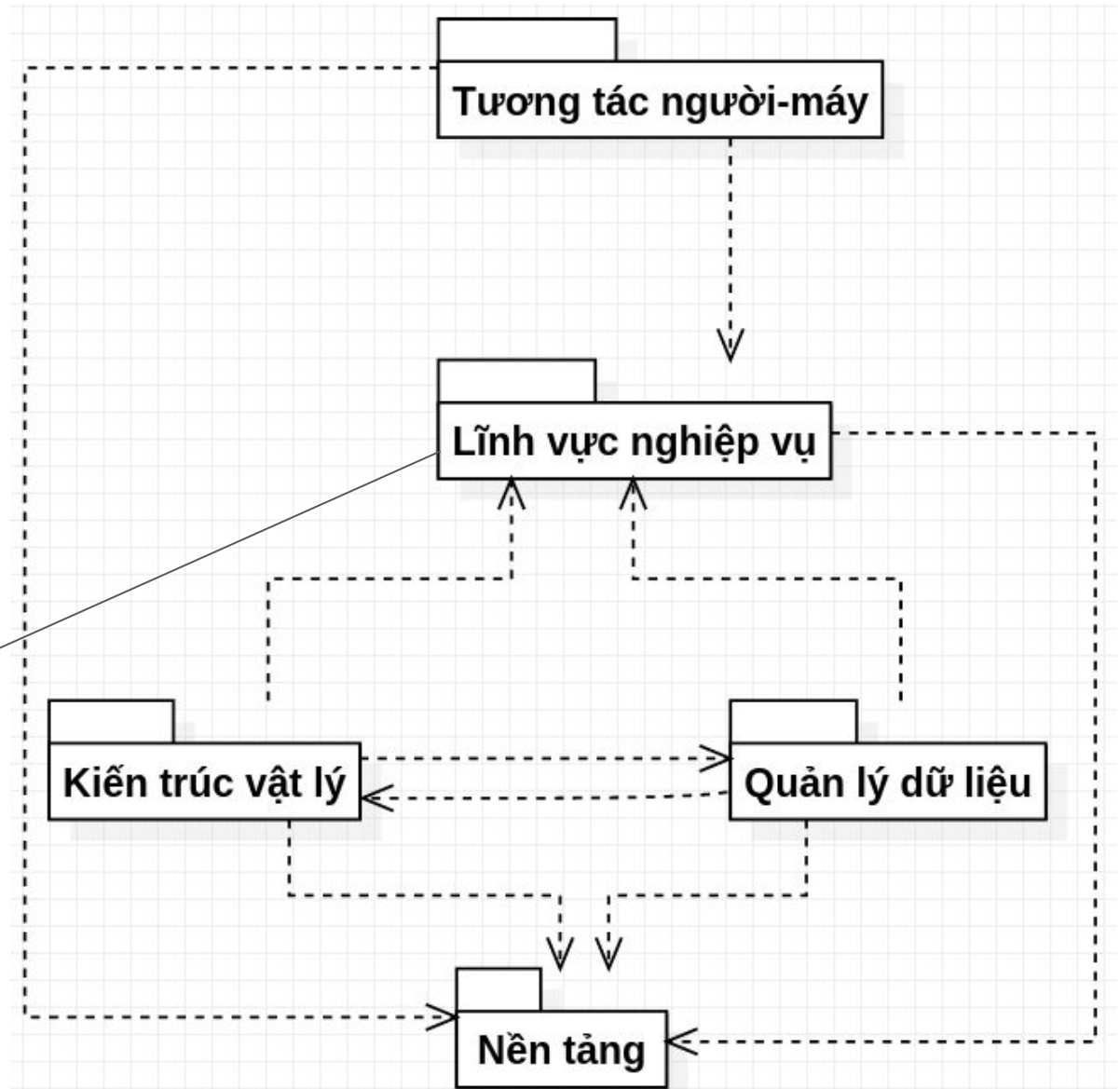


(1 trường hợp khác với) Các lớp  
Shape và Point thuộc gói Types



# Tổng quan kiến trúc 5 tầng

Hệ thống được tạo thành từ 5 tầng, mỗi tầng được biểu diễn như 1 gói khái quát trên sơ đồ.



# Tổng quan kiến trúc 5 tầng<sub>(2)</sub>

- **Tầng nền tảng:** Cung cấp các thành phần cơ bản để xây dựng các ứng dụng HĐT. Ví dụ lớp: Array, Map
- **Tầng nghiệp vụ:** Các xử lý nhằm đáp ứng các hoạt động nghiệp vụ, đã được đề cập tới ở bước phân tích, và tiếp tục được phát triển ở pha thiết kế. Ví dụ lớp: Order, Customer
- **Quản lý dữ liệu:** Các xử lý liên quan đến lưu trữ cố định. Ví dụ lớp: CustomerDAM, FileInputStream
- **Tương tác người - máy:** Triển khai giao diện tách rời với nghiệp vụ. Ví dụ lớp: Form, Button

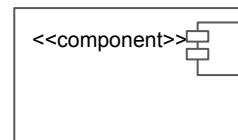
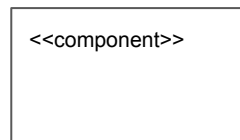
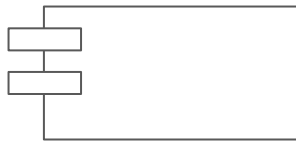
# Nội dung

- Phân mảnh và sơ đồ gói
- Sơ đồ thành phần và sơ đồ triển khai
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc

# Sơ đồ thành phần: Các ký hiệu

*Sơ đồ thành phần biểu diễn các thành phần khác nhau của hệ thống cùng với các giao diện và các mối quan hệ, và cho biết cấu trúc lô-gic của hệ thống*

- Có nhiều cách biểu diễn thành phần:



- Các giao diện được cung cấp và được yêu cầu được biểu diễn bằng các đường tròn và nửa đường tròn.



Giao diện được cung cấp

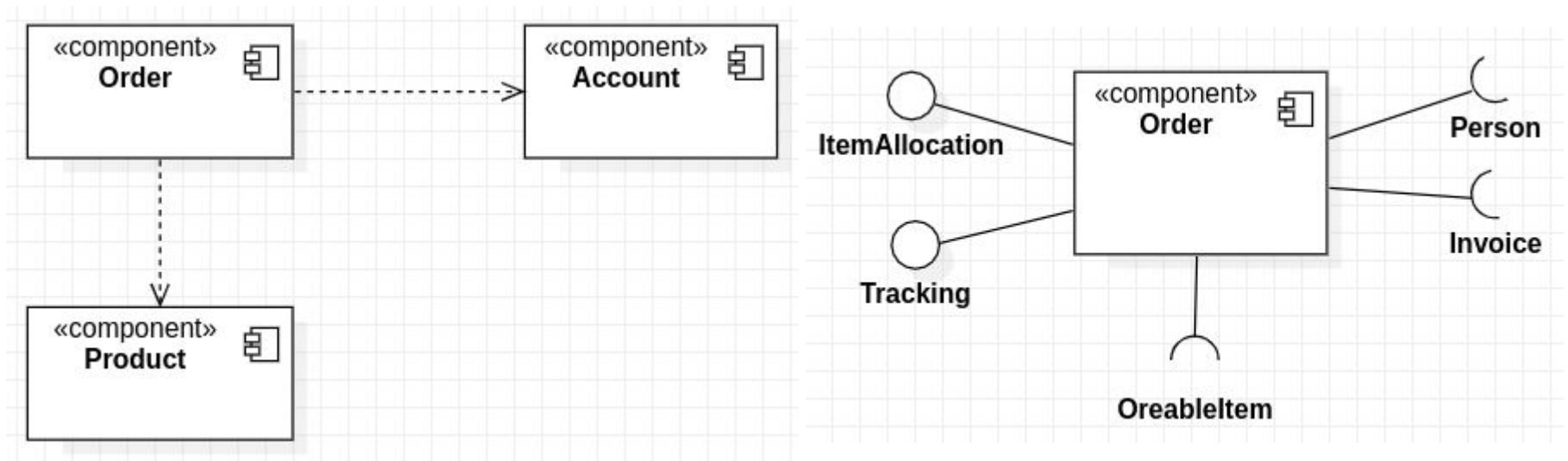


Giao diện được yêu cầu

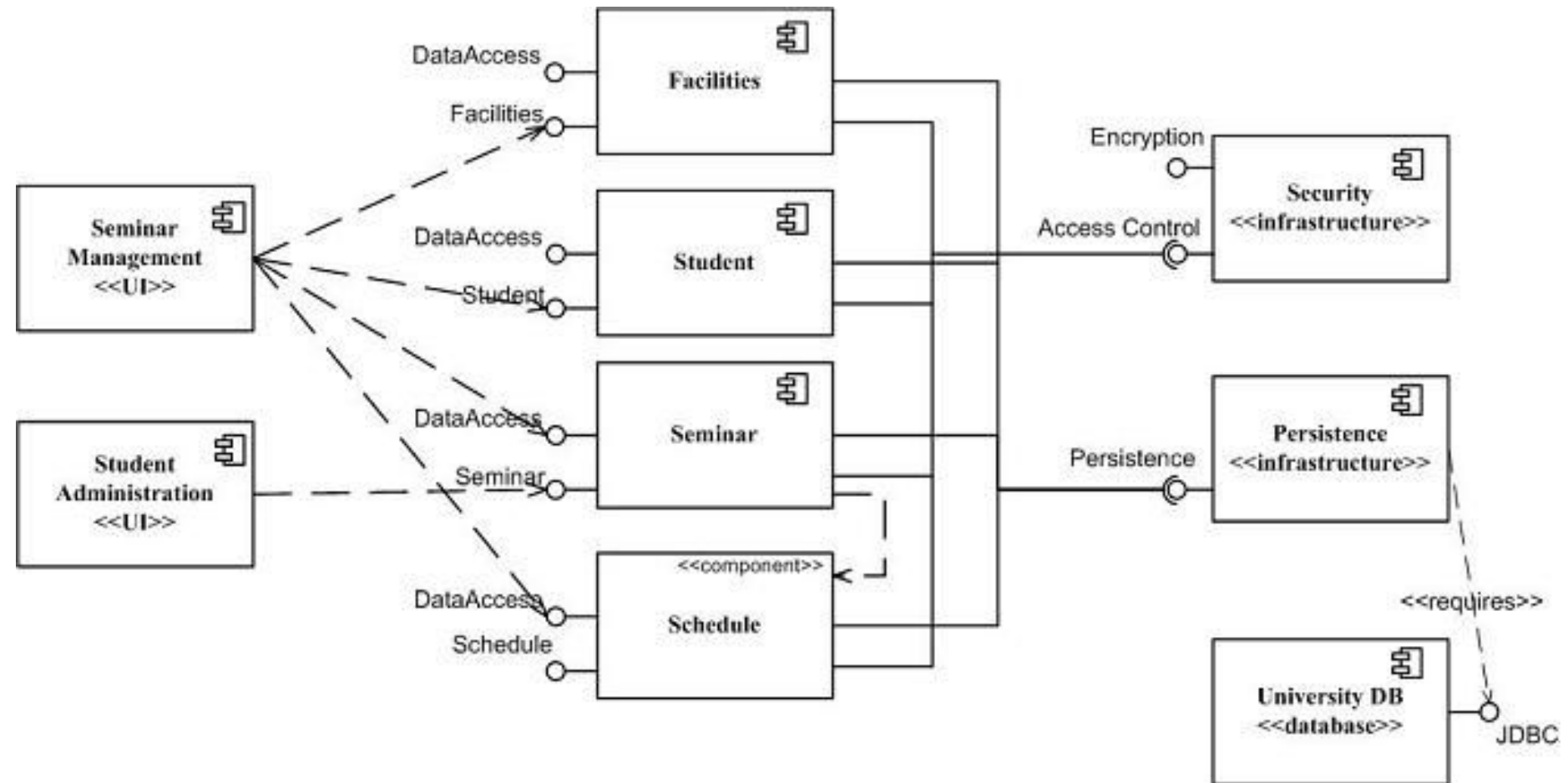
- Quan hệ phụ thuộc được biểu diễn bằng mũi tên nét đứt



# Biểu diễn thành phần và các giao diện



# Sơ đồ thành phần trong UML 2.x



UML 2.0

# Sơ đồ triển khai

- Biểu diễn kiến trúc vật lý của phần cứng và phần mềm của hệ thống được triển khai
- Các nút
  - Thường là các thiết bị tính toán: Máy vi tính, điện thoại thông minh, v.v..
  - Môi trường thực thi: Linux, macOS, Windows, ...
  - Chứa các thành phần phần mềm
- Biểu diễn mối quan hệ vật lý giữa phần cứng và phần mềm trong hệ thống được triển khai
  - Mô tả cách hệ thống tương tác với môi trường bên ngoài

# Sơ đồ triển khai: Artifact

Artifact biểu diễn một thành phần thực được triển khai trong các nút.

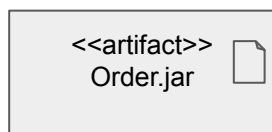
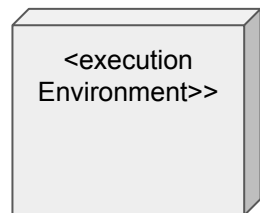


Một số artifact tiêu chuẩn:

- Tập - `<<file>>` một tập cụ thể trong hệ thống tập
- Tài liệu - `<<document>>` một tài liệu, có thể là mã nguồn hoặc tập thực thi
- Thư viện - `<<library>>` một thư viện tĩnh hoặc động
- Tập thực thi - `<<executable>>` một chương trình có thể được thực thi trên máy tính.



# Sơ đồ triển khai: Hệ ký hiệu



- Nút là tài nguyên tính toán:
  - Thiết bị
  - Môi trường thực thi
  - Có thể được kết nối để biểu diễn các kênh trao đổi thông tin theo hình trạng mạng

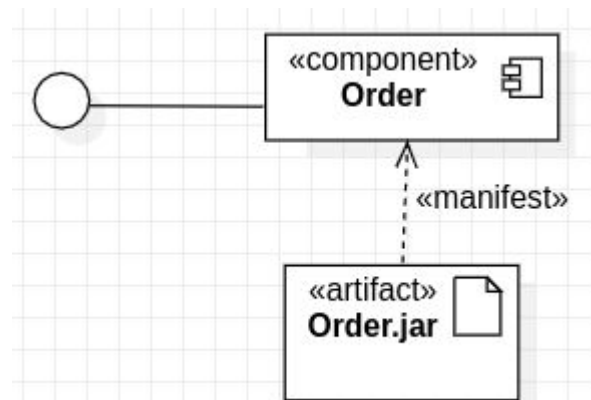
- Thành phẩm / Artifact:
  - Thành phần cụ thể của hệ thống
  - Các phân kiểu:
    - <<mã nguồn>> / <<source>>
    - <<tệp thực thi>> / <<executable>>
    - <<jar>>



- Đường truyền được biểu diễn như liên kết
- Triển khai thành phẩm trên nút

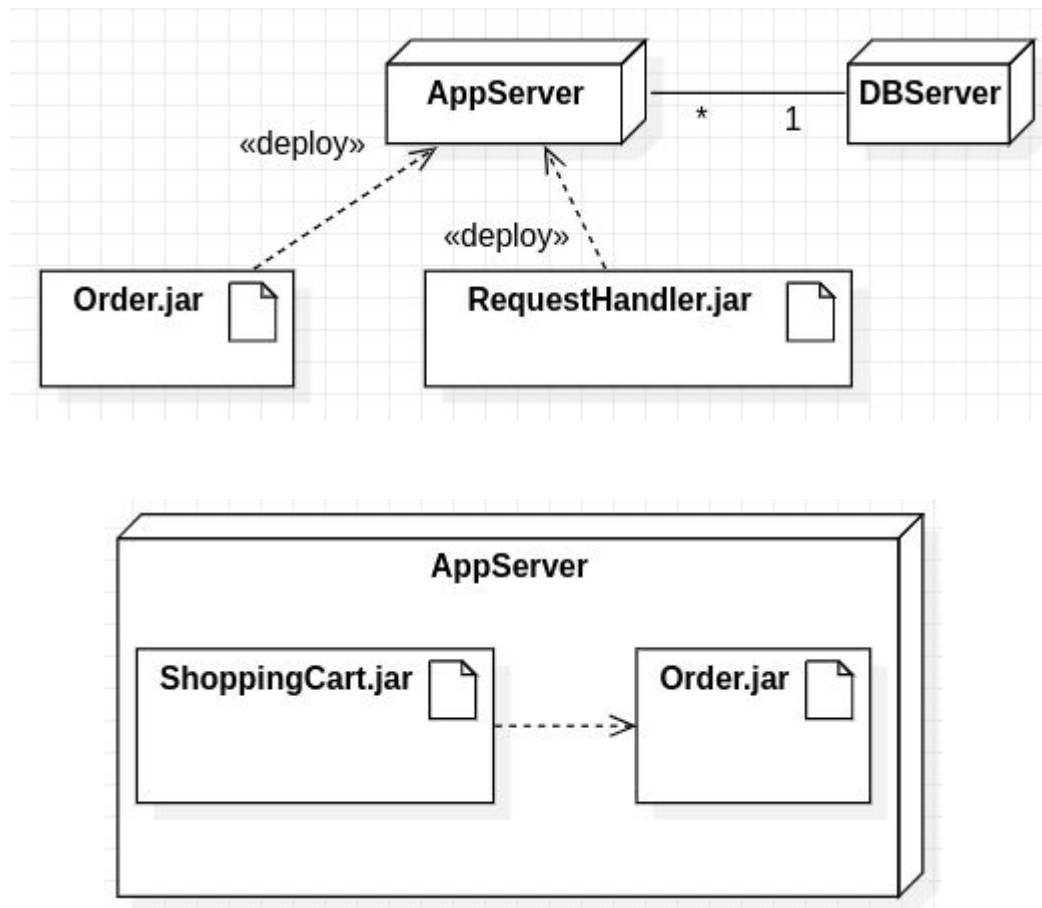
# Quan hệ manifest

- Biểu diễn triển khai vật lý của 1 hoặc nhiều phần tử của mô hình dưới dạng thành phẩm.
- Thường được sử dụng để biểu diễn triển khai của thành phần dưới dạng thành phẩm
  - Mỗi thành phần có thể được triển khai như 1 thành phẩm
  - Mỗi thành phẩm có thể bao gồm nhiều thành phần



- Ngoài ra có thể sử dụng với bất kỳ phần tử nào khác nếu có thể là thành phần của gói / `PackageableElement`

# Biểu diễn cách triển khai thành phẩm



# Nội dung

- Phân mảnh và sơ đồ gói
- Sơ đồ thành phần và sơ đồ triển khai
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc

# Các thành phần kiến trúc

- Kiến trúc công nghệ
  - Máy tính, mạng máy tính và hình trạng mạng, và phần mềm hệ thống
  - Hình thành hạ tầng hỗ trợ các phần mềm và dịch vụ được công ty cung cấp
- Kiến trúc ứng dụng
  - Các chương trình phần mềm / các hệ thống thông tin công ty cần để hỗ trợ các hoạt động của nó.
  - Được triển khai trên kiến trúc công nghệ bằng cách cài đặt các thành phần phần mềm trên các thiết bị phần cứng và kết nối chúng bằng các giao thức và đường truyền mạng.

*Mục đích thiết kế kiến trúc tổng quan là xác định cách bố trí các thành phần phần mềm trong các phần cứng.*

# Các thành phần kiến trúc cơ bản

- Thành phần phần cứng
  - Máy khách / máy tính cá nhân
  - Máy chủ
  - Hạ tầng mạng
- Thành phần phần mềm
  - Thành phần lưu trữ dữ liệu
  - Thành phần truy cập dữ liệu
  - Thành phần ứng dụng
  - Thành phần trình diễn

# Môi trường thực thi

- Máy chủ - Quản lý các tài nguyên dùng chung và cho phép người dùng và các máy tính khác truy cập đến những tài nguyên đó.
- Máy khách / Máy tính cá nhân
  - Máy bàn, máy tính xách tay, máy tính bảng, điện thoại thông minh, v.v..

# Điện toán đám mây

- Cho thuê hạ tầng phần cứng
  - Máy chủ ở trên "Đám mây"
  - Máy khách là thiết bị của người dùng
- "Đám mây"
  - Trung tâm dữ liệu, nội hoặc ngoại; hoặc
  - Dịch vụ được cung cấp bởi nhà cung cấp
  - Tích hợp nhiều công nghệ:
    - Ảo hóa
    - Kiến trúc hướng dịch vụ
    - Mạng lưới tính toán / Grid computing



# Hạ tầng mạng

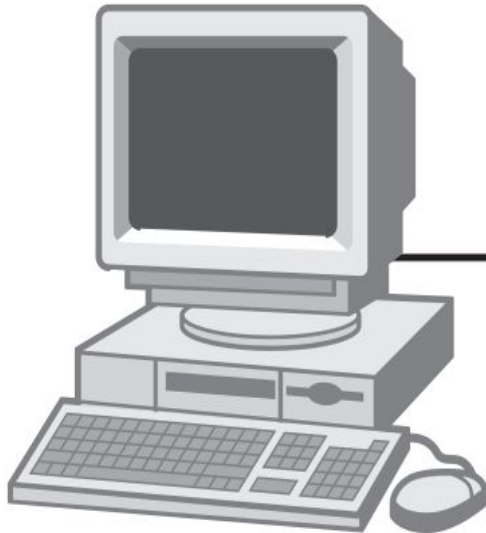
- Có thể bao gồm phần cứng, phần mềm, phương tiện truyền tín hiệu
- Hạ tầng Internet
  - Các đường truyền băng thông lớn và máy tính tốc độ cao
  - Được sở hữu bởi các chính phủ và các công ty viễn thông
- Mạng địa phương (LAN)
  - Mạng nhỏ cho 1 hệ thống thông tin
- Mạng toàn cầu (WWW)
  - Tất cả các tài nguyên được kết nối và truy cập qua Internet

# Các giao thức

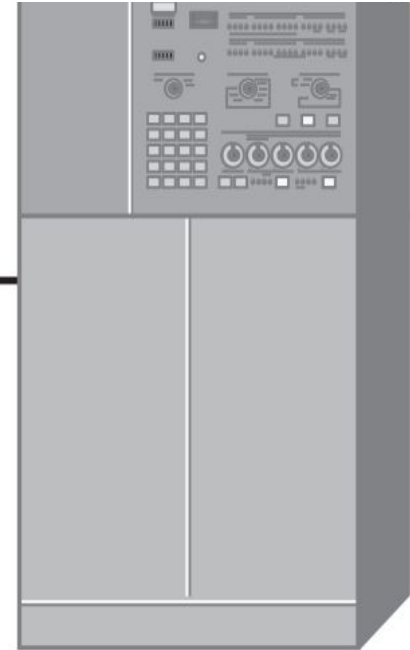
- Giao thức:
  - Một tập ngôn ngữ và quy tắc để đảm bảo giao tiếp và trao đổi dữ liệu giữa phần cứng và phần mềm
- Các giao thức mạng:
  - Mạng riêng tư ảo (VPN)
    - Tạo mạng riêng tư trên Internet bằng cách sử dụng các công nghệ bảo mật và mã hóa

# Kiến trúc dựa trên máy chủ

Máy khách/Cổng  
vận hành



Máy chủ



Trình diễn  
Ứng dụng  
Truy cập dữ liệu  
Lưu trữ dữ liệu

# Phần mềm như 1 dịch vụ (SaaS)

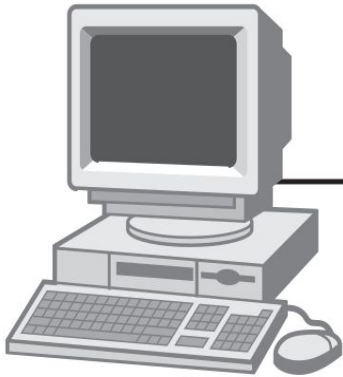
- Thường được thực hiện với hạ tầng điện toán đám mây
- Phần mềm không được cài trên thiết bị của người dùng
- Các dịch vụ ứng dụng được truy cập từ xa (thường qua trình duyệt)
- Dữ liệu người dùng được cô lập và được lưu trên máy chủ dùng chung

# Dịch vụ Web

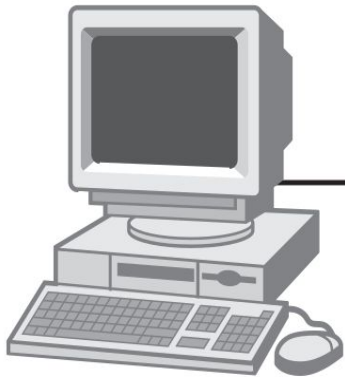
- Chức năng phần mềm được thực hiện bằng các quy chuẩn Web
  - Truy cập qua URL
  - Dữ liệu đầu vào được gửi qua URL
  - Được thực thi từ xa
  - Dữ liệu được trả về trong trang Web

# Kiến trúc dựa trên máy khách

Các máy khách

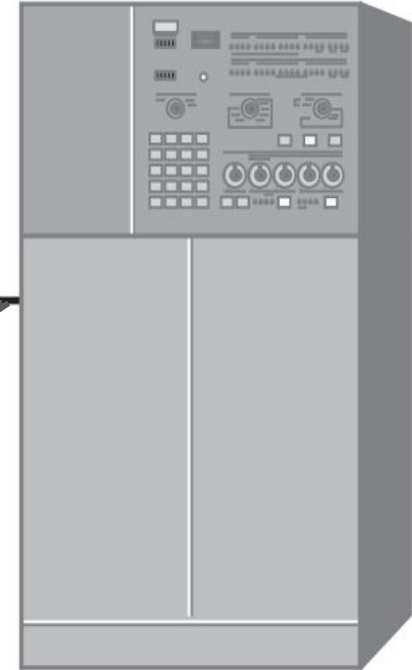


Trình diễn, ứng dụng  
Truy cập dữ liệu



Trình diễn, ứng dụng  
Truy cập dữ liệu

Máy chủ



Lưu trữ dữ liệu

# Kiến trúc máy khách-máy chủ

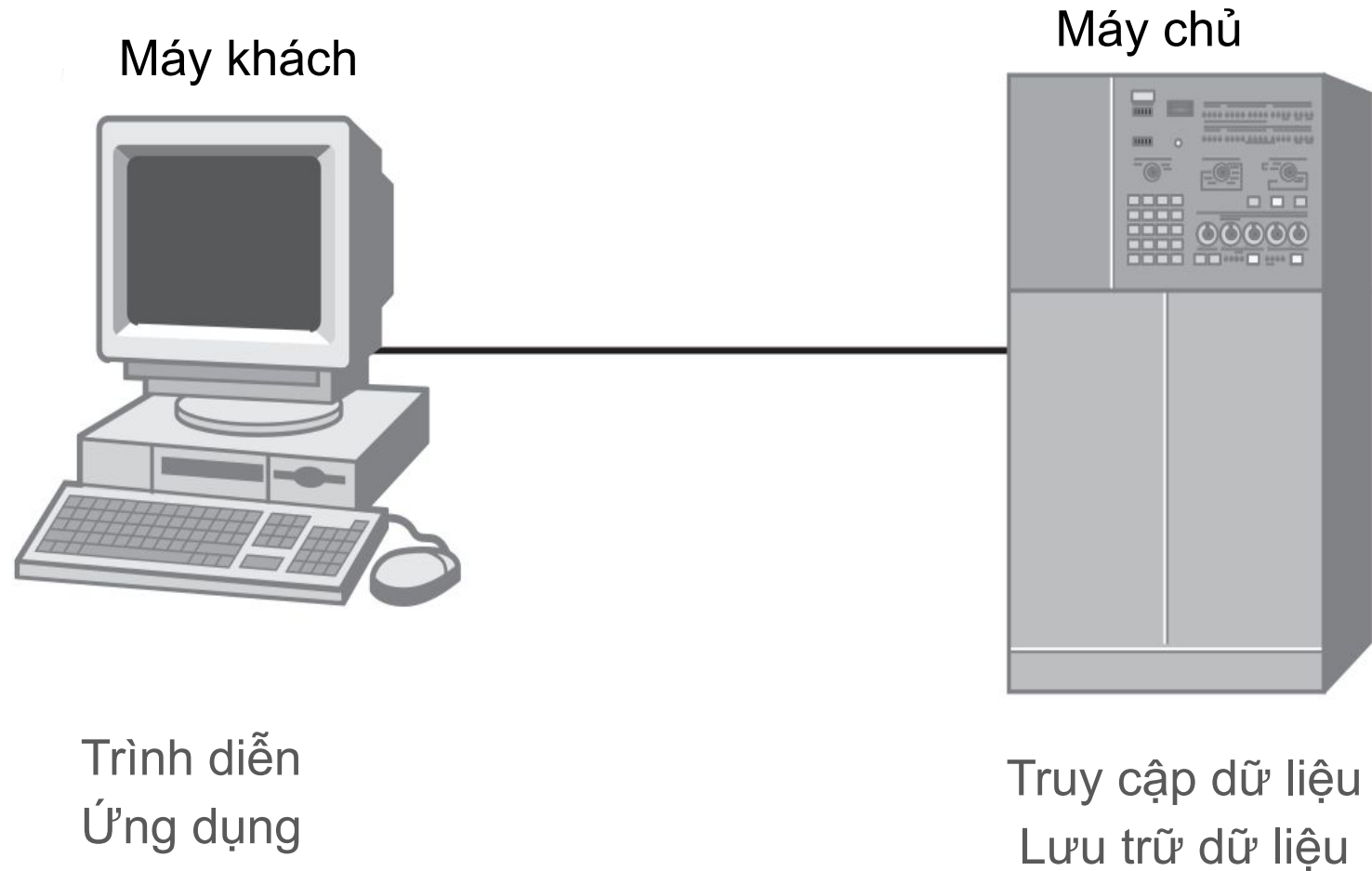
- Hệ thống được thiết kế với 1 phần ở máy chủ và 1 phần ở máy khách, cố gắng phân chia hợp lý khối lượng tính toán
- Kiến trúc phổ biến nhất trong các hệ thống hiện đại
- Khối lượng tính toán của máy khách dao động
  - Máy khách mỏng chỉ thực hiện lô-gic trình diễn
  - Máy khách dày thực hiện lô-gic trình diễn và lô-gic ứng dụng
- Khả năng mở rộng cao với chi phí từng phần
- Phức tạp hơn phải phát triển ứng dụng cho máy khách, máy chủ và đảm bảo tương tác phân tán.

# Các dãy máy khách-máy chủ

- Các dãy máy khách-máy chủ được định nghĩa dựa trên cách phân mảnh lô-gic:
  - 2-dãy: 1 máy chủ chịu trách nhiệm lưu trữ và truy cập dữ liệu; máy khách có trách nhiệm xử lý lô-gic ứng dụng và lô-gic trình diễn
  - 3-dãy: Lô-gic truy cập và lưu trữ dữ liệu trên 1 máy chủ; lô-gic ứng dụng trên 1 máy chủ khác; máy khách có trách nhiệm xử lý lô-gic trình diễn.
  - n-dãy: Lô-gic ứng dụng được phân chia trên nhiều máy chủ, lô-gic dữ liệu trên 1 máy khác chủ khác
    - Phổ biến trong các ứng dụng thương mại điện tử
    - Cân bằng tải tốt hơn
    - Khả năng mở rộng cao hơn hệ thống 2 hoặc 3 dãy
    - Nhu cầu sử dụng mạng cao hơn



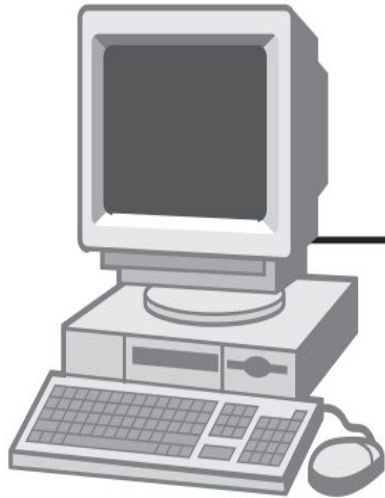
# Kiến trúc máy khách-máy chủ



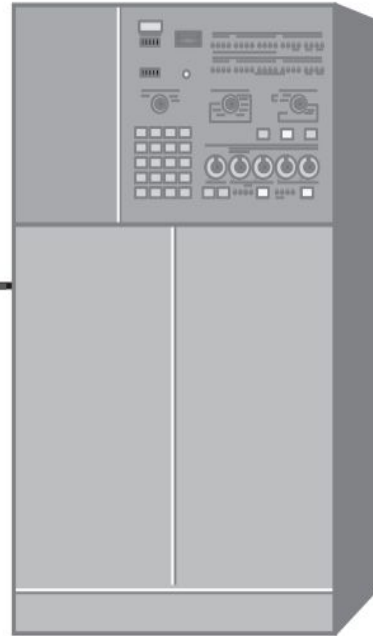
*Kiến trúc 2 dây (sử dụng 2 nhóm máy tính)*

# Kiến trúc máy khách-máy chủ: 3 dãy

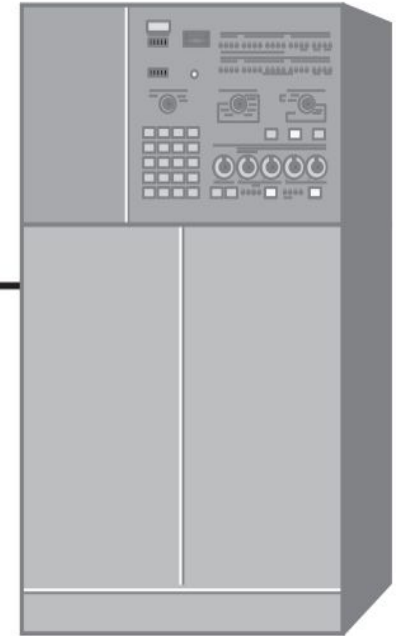
Máy khách



Máy chủ ứng dụng



Máy chủ CSDL

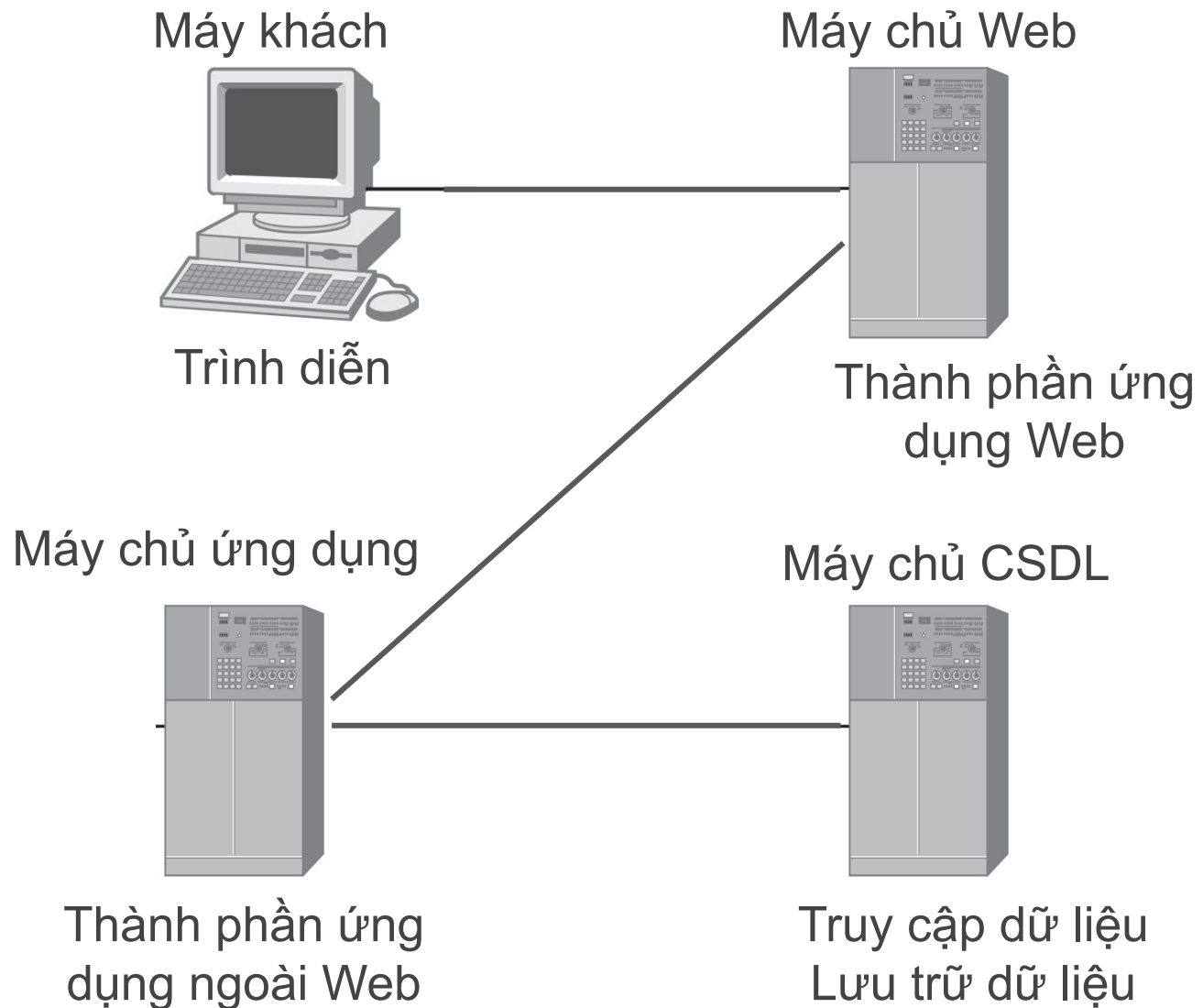


Trình diễn

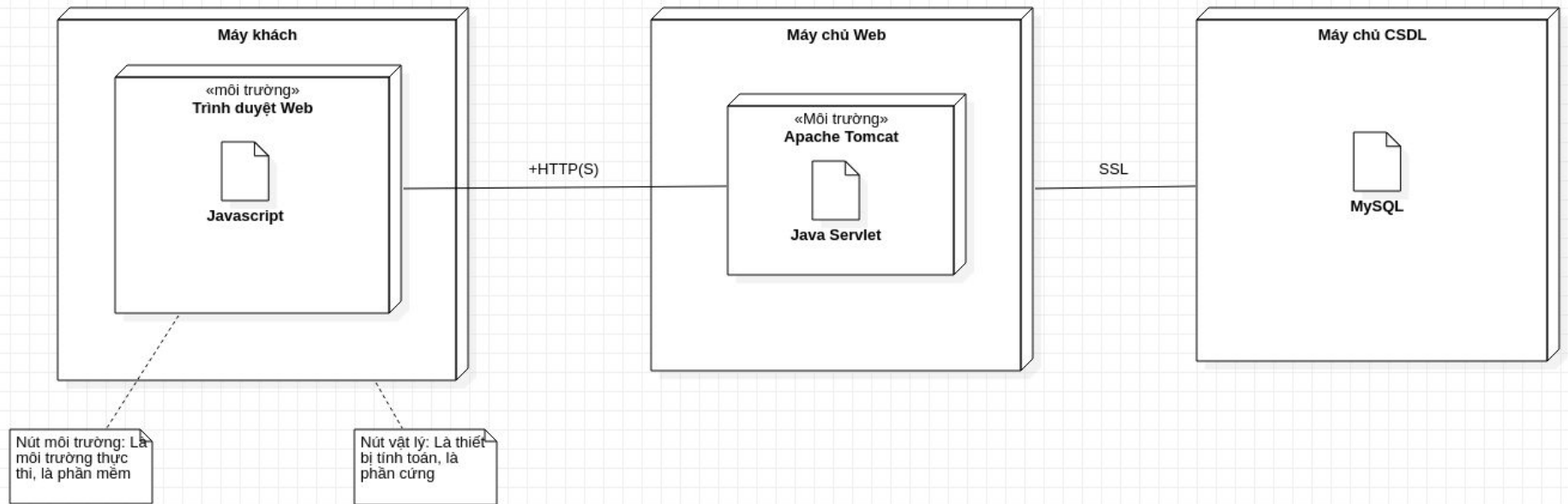
Ứng dụng

Truy cập dữ liệu  
Lưu trữ dữ liệu

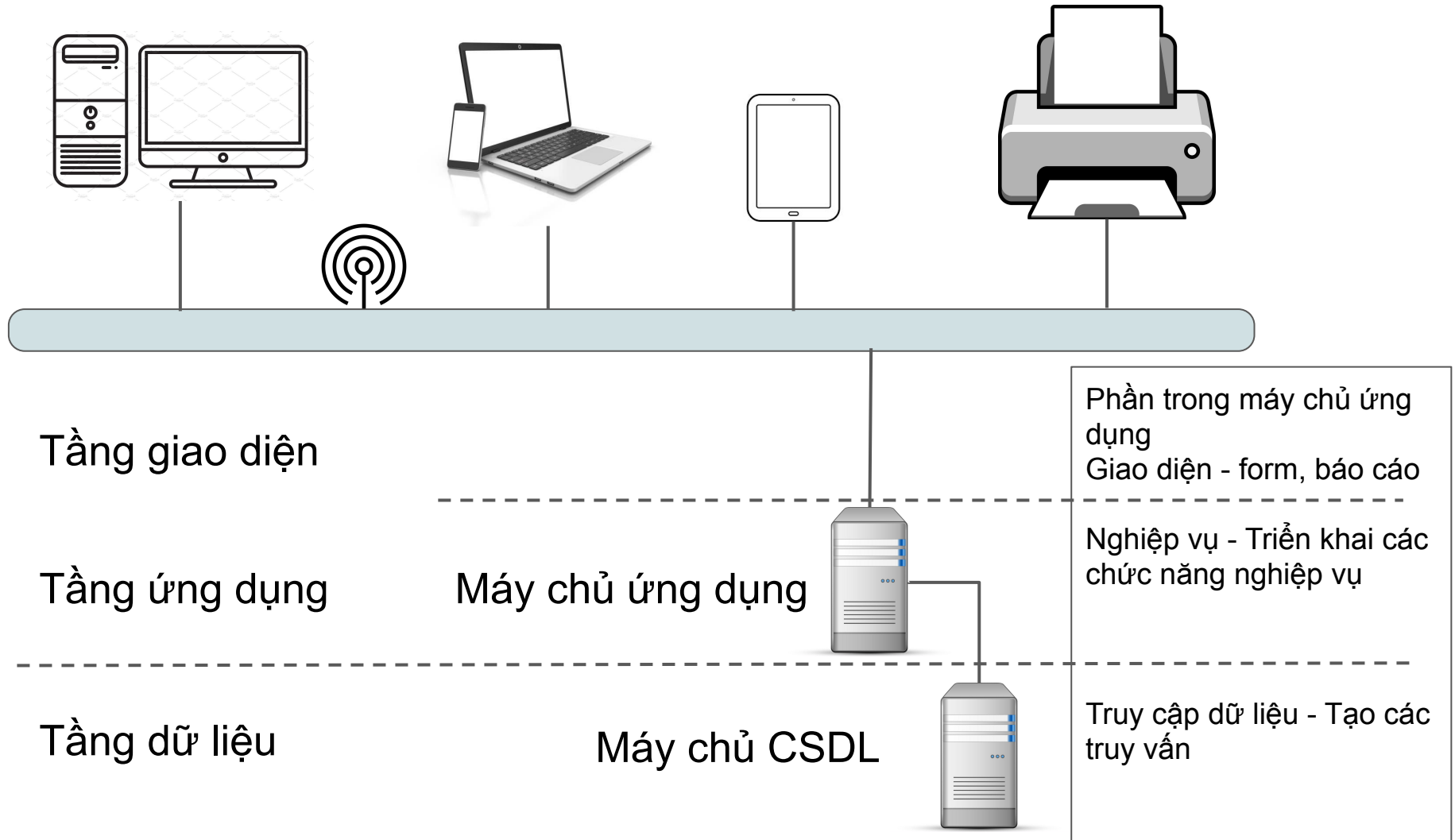
# Kiến trúc máy khách-máy chủ: 4-dãy



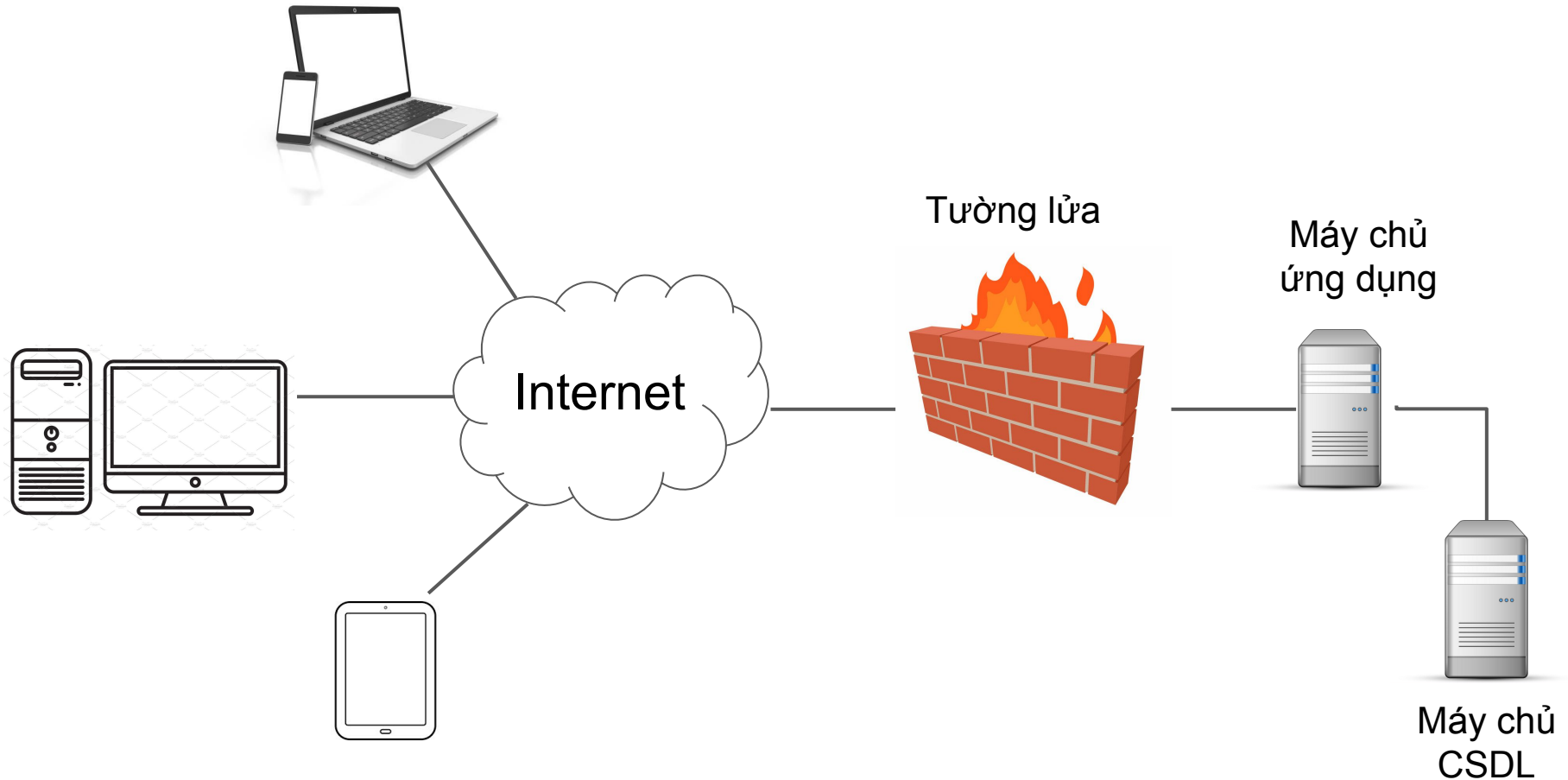
# Triển khai hệ thống



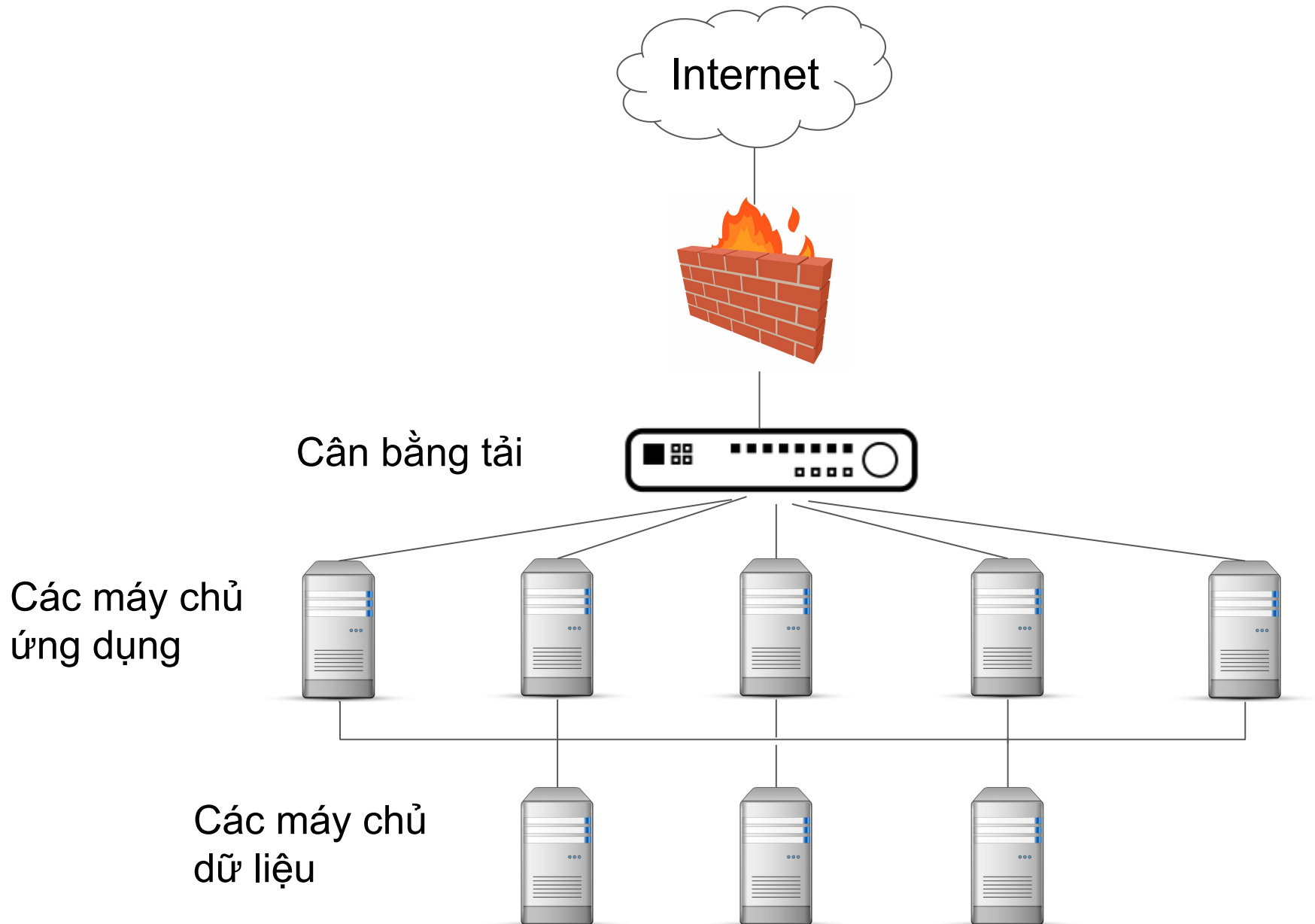
# Kiến trúc 3 tầng: Triển khai nội bộ




# Kiến trúc 3 tầng: Triển khai trên internet



# Kiến trúc 3 tầng: Mở rộng hệ thống



# Nội dung

- Phân mảnh và sơ đồ gói
  - Sơ đồ thành phần và sơ đồ triển khai
  - Các thành phần kiến trúc Hệ thống
  - Mẫu kiến trúc
- 

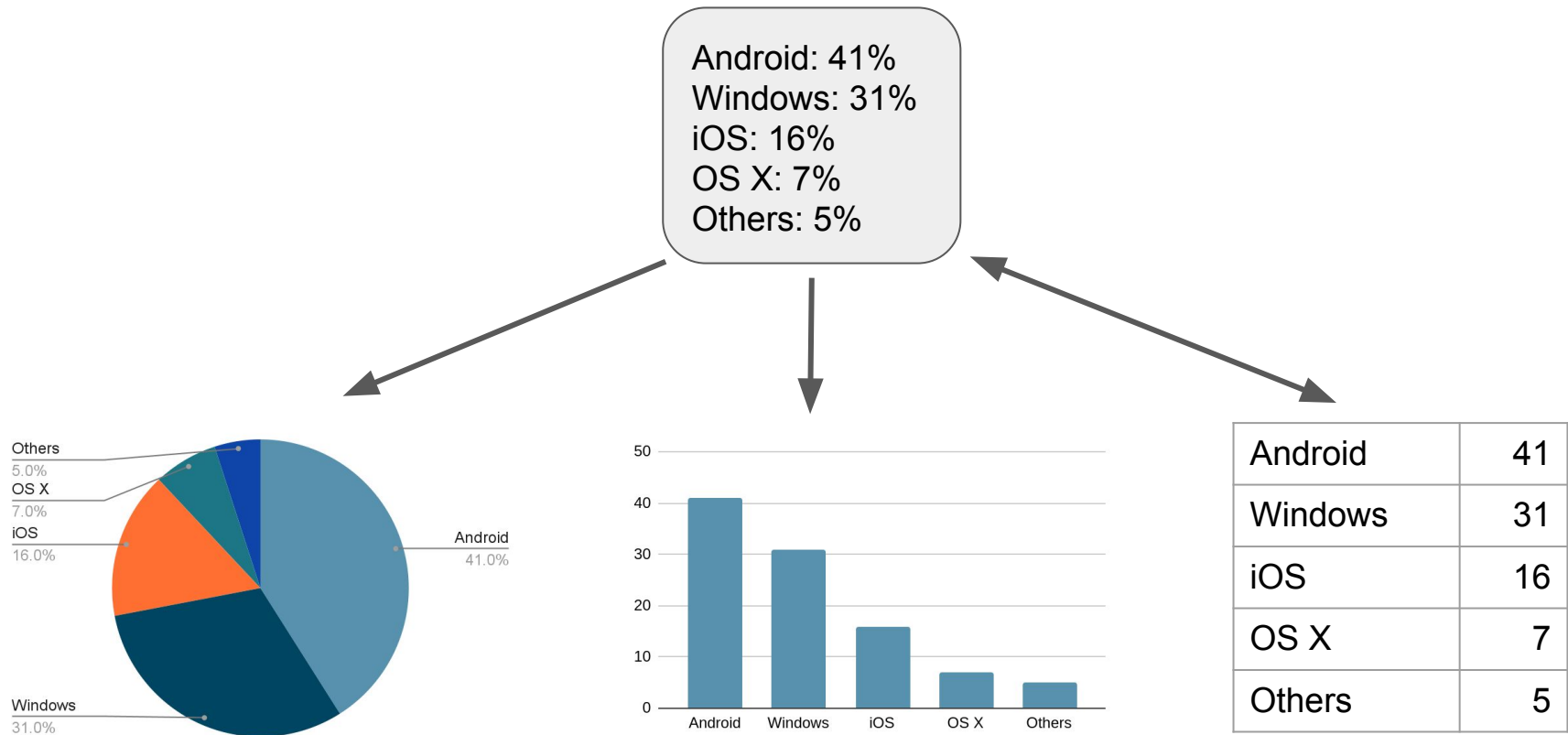


# Kiến trúc MVC

- Mẫu kiến trúc Model-View-Controller (MVC) phân mảnh 1 chương trình tương tác thành 3 thành phần.
  - Mô hình/Model chứa dữ liệu và triển khai chức năng lõi
  - Khung nhìn/View hiển thị thông tin cho người dùng
  - Điều khiển/Controller tiếp nhận dữ liệu vào của người dùng
- Khung nhìn và điều khiển kết hợp lại thành giao diện người dùng.
- Mỗi thành phần có thể được triển khai bằng 1 lớp.
- Cập nhật theo các thay đổi trong mô hình có thể được thực hiện theo các cơ chế:
  - Đẩy / Push: Mô hình thông báo về các thay đổi.
  - Kéo / Pull: Khung nhìn và điều khiển đọc dữ liệu cập nhật từ mô hình khi có thay đổi.

# Trình diễn thông tin

## Ví dụ dữ liệu thị phần hệ điều hành

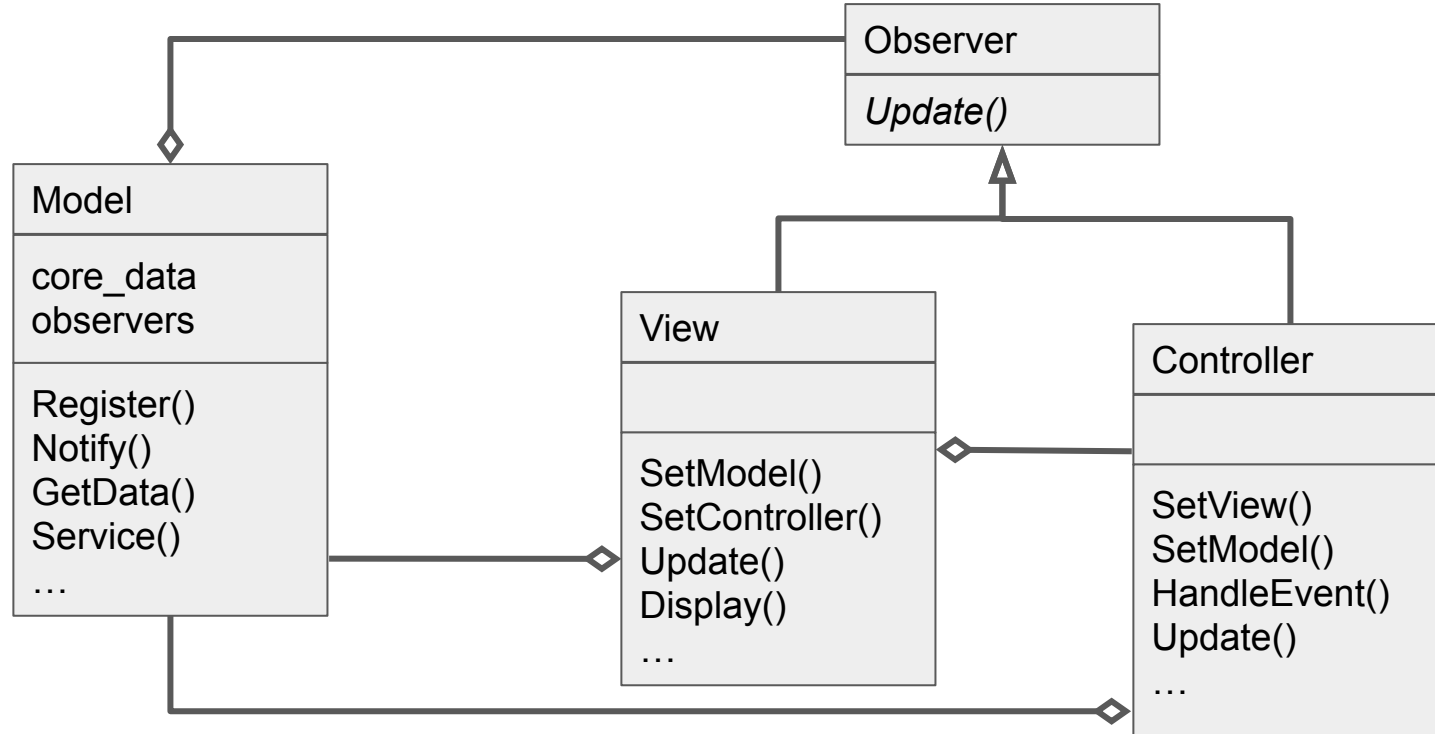


# Ứng dụng tương tác với giao diện linh động

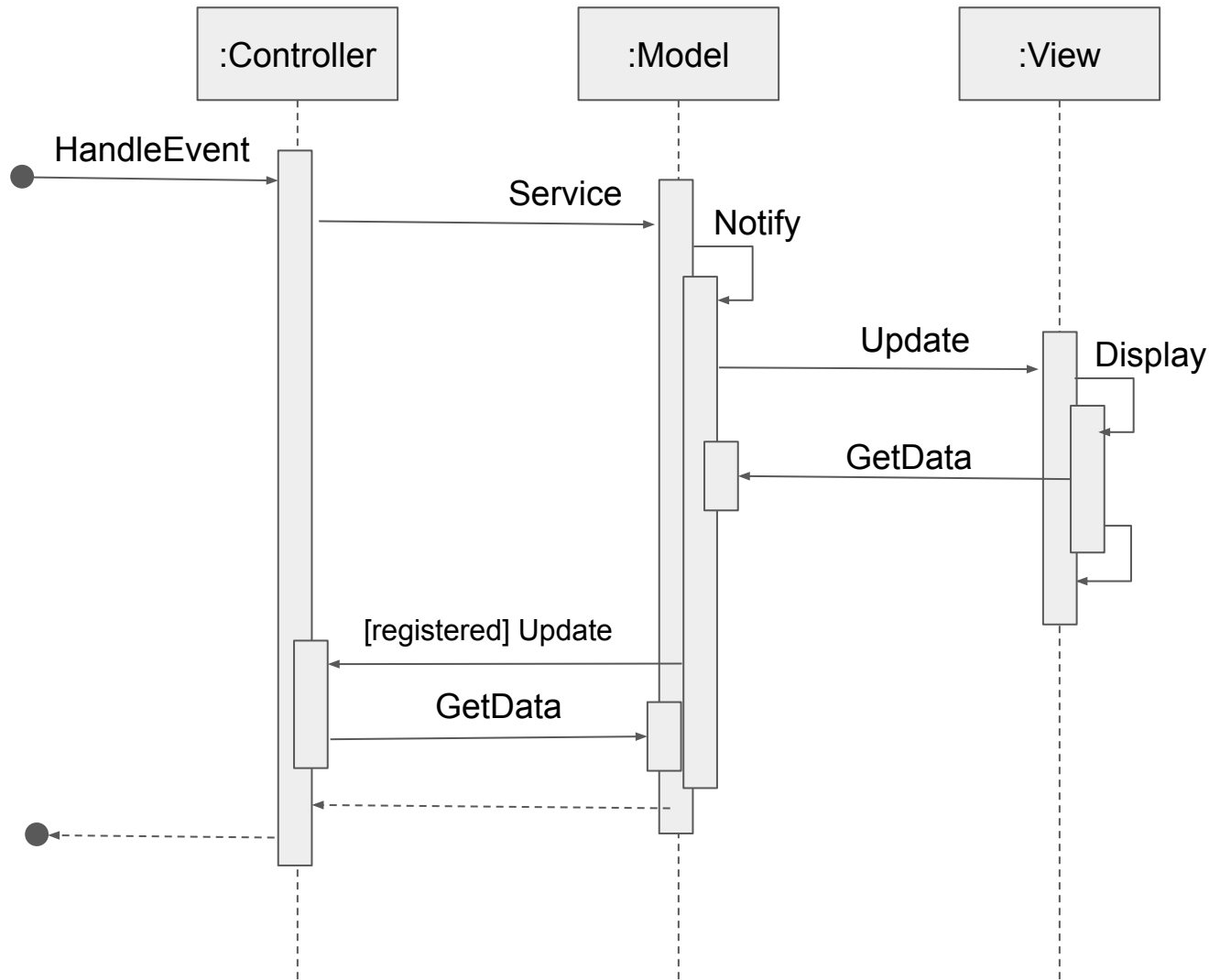
- Giao diện người dùng là thành phần có thể thay đổi vì nhiều nguyên nhân khác nhau:
  - Khi mở rộng chức năng của chương trình,
  - Các yêu cầu khác nhau của người dùng, v.v.
- Phân mảnh chương trình theo MVC cho phép giữ nguyên lô-gic ứng dụng (thành phần Mô hình) trong khi thay đổi thành phần giao diện, sử dụng đồng thời nhiều giao diện, và bổ xung giao diện mới.

# Triển khai MVC theo cơ chế đẩy dữ liệu

*Trong triển khai này mô hình thông báo khung nhìn và điều khiển khi có thay đổi*



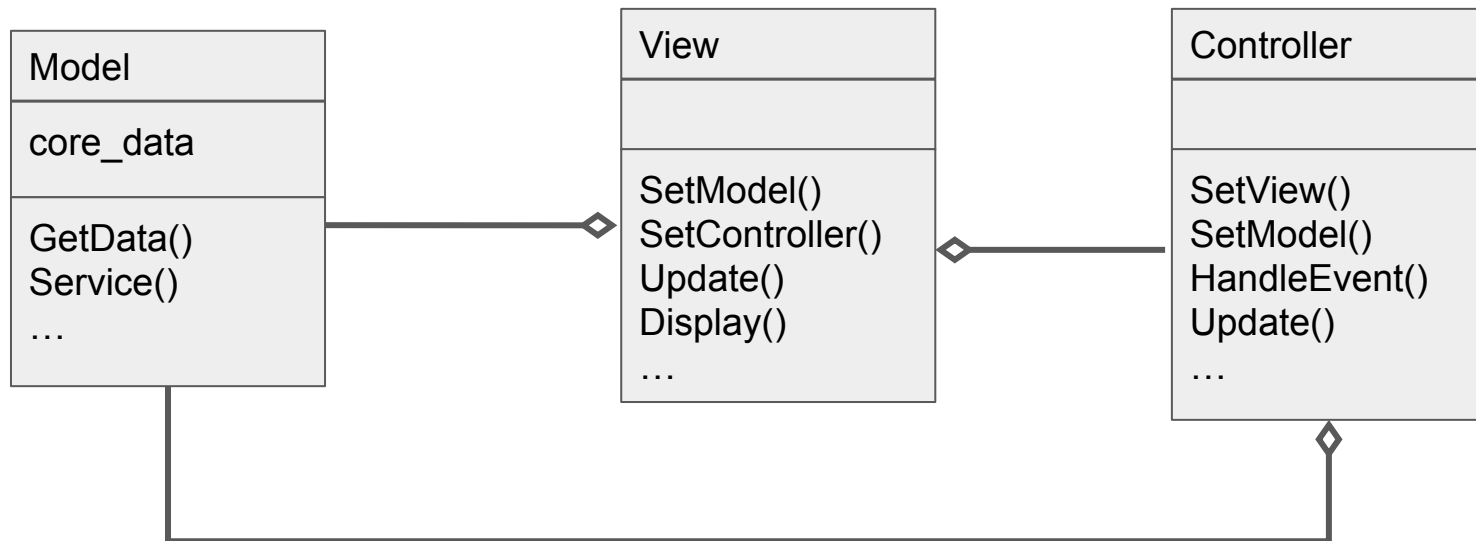
# Vòng đời thông điệp theo cơ chế đẩy



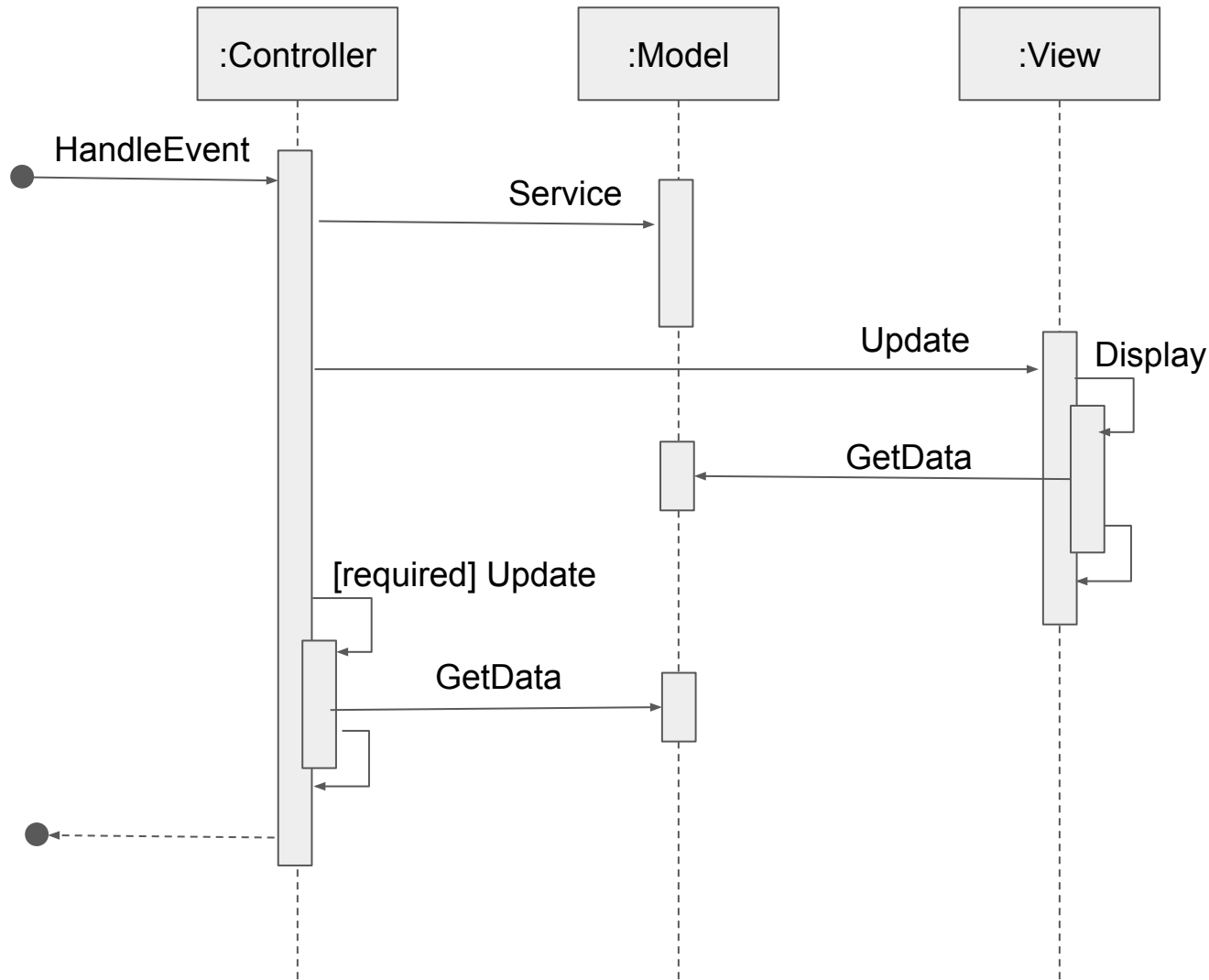
*Mô hình thông báo các thay đổi*

# Triển khai MVC theo cơ chế kéo dữ liệu:

*Trong triển khai này khung nhìn và điều khiển đọc dữ liệu từ mô hình khi có thay đổi cần cập nhật:*



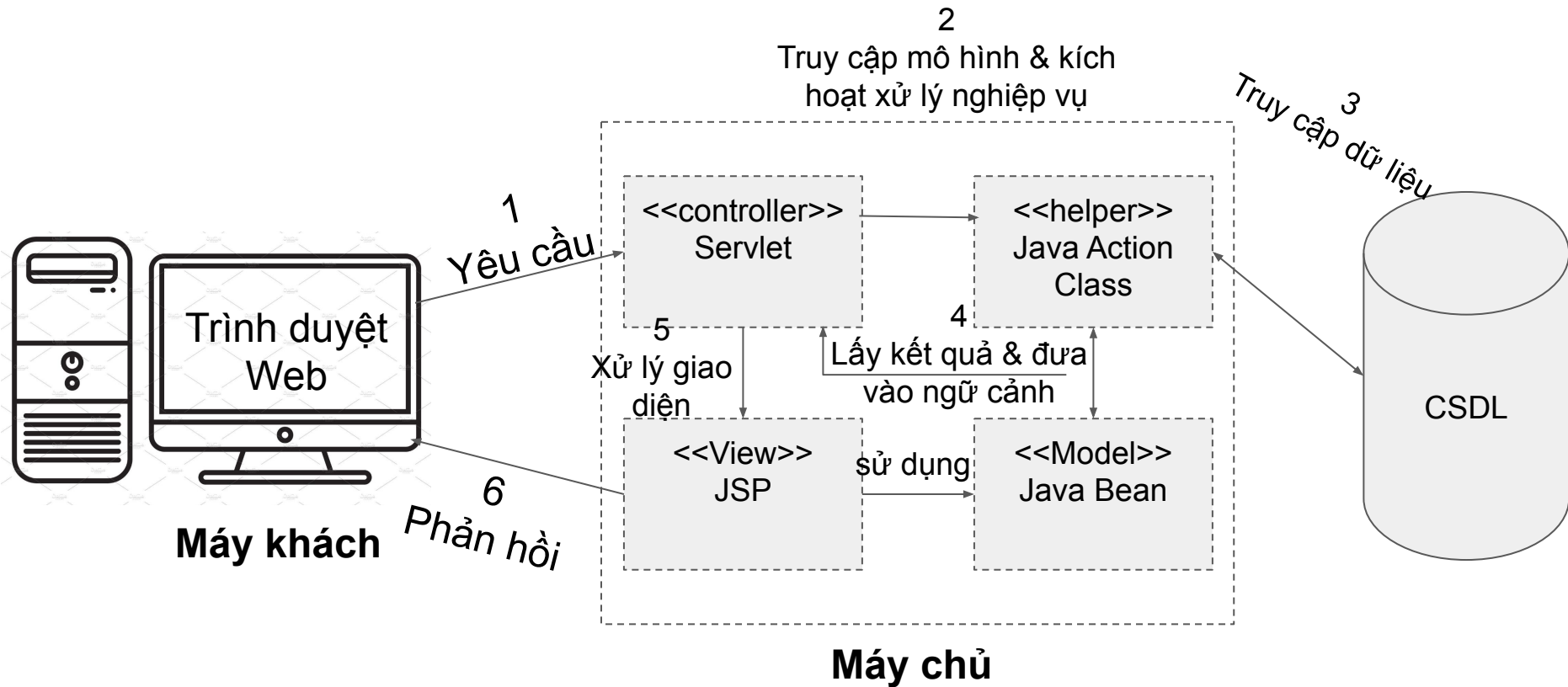
# Vòng đời thông điệp theo cơ chế kéo



*\*Khung nhìn và Điều khiển đọc dữ liệu khi cần cập nhật*

# Nền tảng Struts

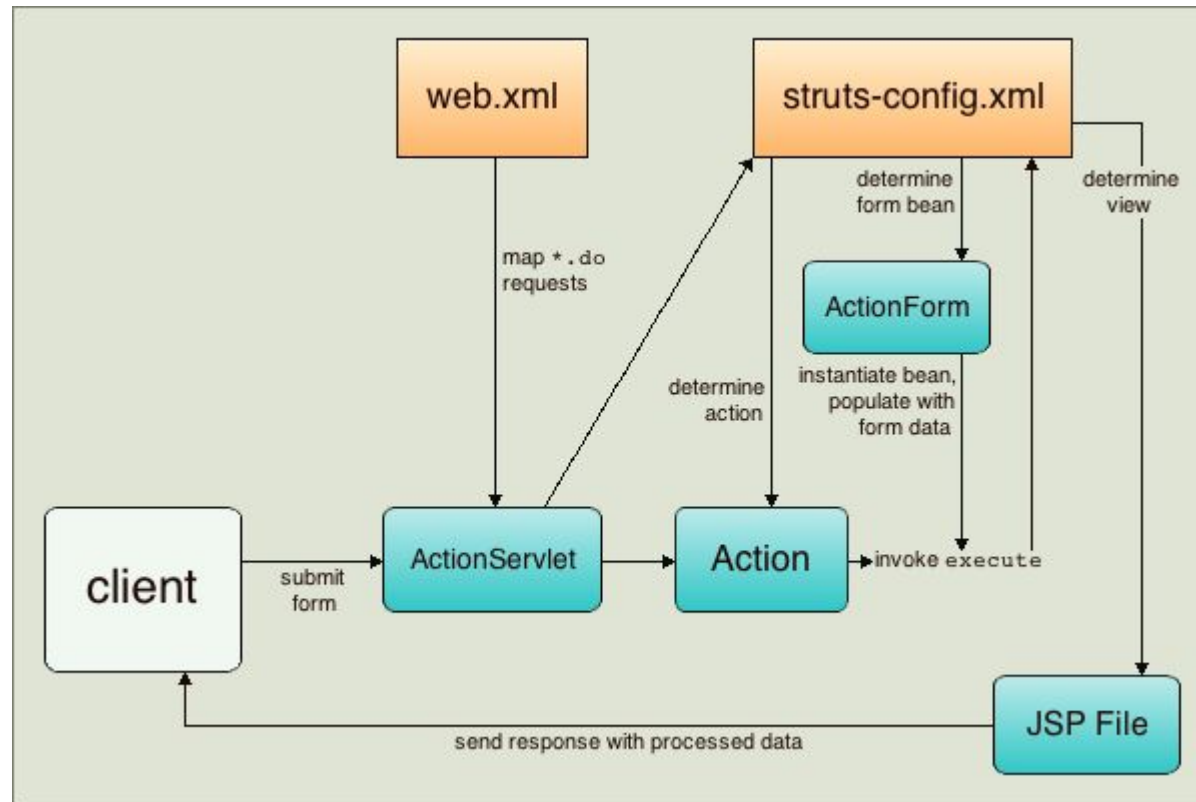
*Kiến trúc JSP Model 2: Tách biệt lô-gic trình diễn và lô-gic ứng dụng tương tự như trong MVC.*



*\* JSP Model 2 không định nghĩa định dạng cụ thể của Model*

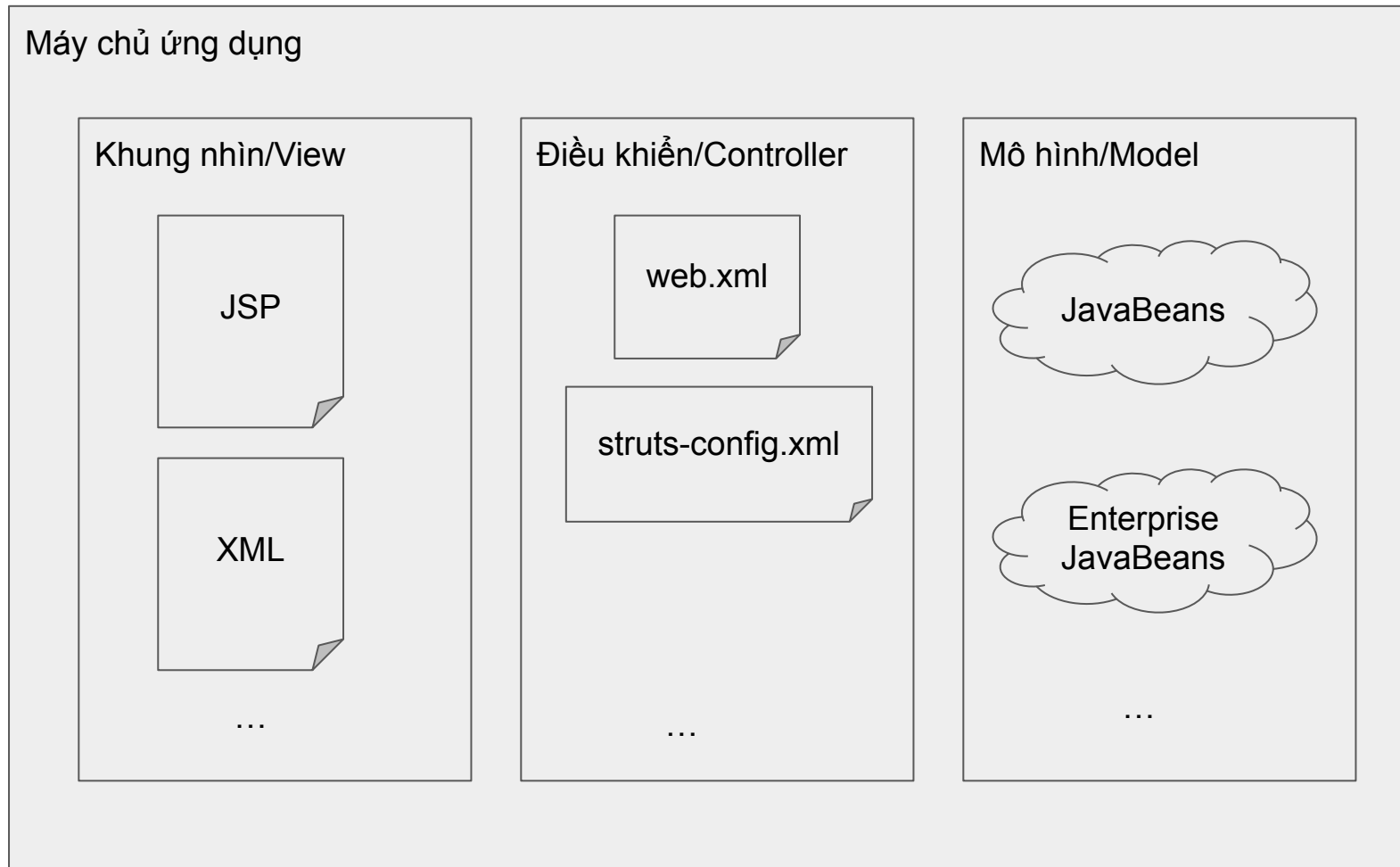


# Kiến trúc ứng dụng Web: Nền tảng Struts



[apache.org]

# JSP Model 2 và MVC



# Bài tập

Phân tích các vấn đề với thiết kế sau và phác thảo giải pháp hướng tới liên kết lỏng giữa các thành phần: Vẽ sơ đồ lớp và sơ đồ tuần tự.

