LAB 5 - Thiết kế Kiến trúc Hệ thống

1. Tên kiến trúc hệ thống: Kiến trúc 3-tier

1.1 Mô tả tổng quan:

Kiến trúc 3-tier chia hệ thống thành 3 tầng riêng biệt:

Tầng	Mô tả
Presentation Tier	Giao diện người dùng (Web/App) để nhập liệu, hiển thị thông tin
Logic Tier	Xử lý nghiệp vụ: quản lý đơn hàng, hóa đơn, vật tư, nhân viên...
Data Tier	Lưu trữ dữ liệu trong hệ quản trị cơ sở dữ liệu (SQL Server, MySQL...)

1.2 Lý do lựa chọn:

- Phù hợp với hệ thống có nhiều nghiệp vụ và người dùng.
- Dễ bảo trì, mở rộng từng tầng mà không ảnh hưởng đến tầng khác.
- Tăng tính bảo mật khi tầng logic xử lý dữ liệu thay vì giao diện truy cập trực tiếp.

2. Phân tích các tầng/thành phần

2.1 Tên tầng/thành phần & Vai trò:

Tầng	Thành phần	Vai trò
Presentation	Web/App UI	Giao diện nhập đơn hàng, xem hóa đơn, quản lý vật tư
Logic	API, Controller	Xử lý logic: tạo đơn hàng, tính tổng tiền, kiểm tra tồn kho
Data	Database	Lưu trữ bảng: NhanVien, DonHang, HoaDon, VatTu, KhachHang...

2.2 Mối quan hệ giữa các tầng:

- UI gửi yêu cầu đến tầng Logic qua API.
- Logic xử lý và truy vấn dữ liệu từ tầng Data.
- Kết quả trả về UI để hiển thị cho người dùng.
  Ví dụ 1: Tạo đơn hàng mới
- Presentation Tier (UI): Người dùng nhập thông tin đơn hàng qua form web/app (chọn vật tư, số lượng, ghi chú…).
- Logic Tier (API/Controller): Controller nhận dữ liệu → gọi hàm TaoDonHang() trong lớp DonHangBUS để xử lý nghiệp vụ (kiểm tra tồn kho, tính tổng tiền…).

- Data Tier (Database): DonHangBUS gọi DonHangDAL để lưu thông tin đơn hàng vào bảng DonHang và chi tiết vào bảng ChiTietDonHang.

Ví dụ 2: Hiển thị danh sách hóa đơn

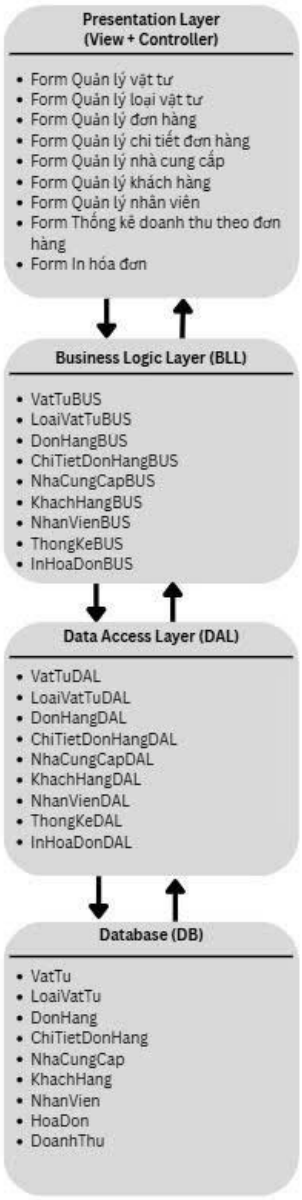
- Presentation Tier (UI): Người dùng chọn chức năng “Xem hóa đơn”.
- Logic Tier (API/Controller): Controller gọi HoaDonBUS.LayDanhSachHoaDon() để lấy dữ liệu.
- Data Tier (Database): HoaDonBUS gọi HoaDonDAL để truy vấn bảng HoaDon, sau đó trả kết quả về UI để hiển thị.

Ví dụ 3: Cập nhật thông tin vật tư

- Presentation Tier (UI): Người dùng chỉnh sửa thông tin vật tư (tên, đơn giá, số lượng…).
- Logic Tier (API/Controller): Controller gọi VatTuBUS.CapNhatVatTu() để kiểm tra dữ liệu hợp lệ.
- Data Tier (Database): VatTuBUS gọi VatTuDAL.Update() để cập nhật vào bảng VatTu.

3. Sơ đồ kiến trúc

3.1 Hình vẽ sơ đồ



3.2 Giải thích tóm tắt

- Người dùng nhập liệu trên giao diện (ví dụ: phiếu nhập vật tư).
- Controller nhận yêu cầu → gửi đến **BLL** để xử lý nghiệp vụ.
- **BLL** gọi **DAL** để lưu vào **Database**.
- Khi cần thống kê, dữ liệu từ DB sẽ đi ngược lại qua các tầng để hiển thị trên UI.

4. Nhận xét

4.1 Điểm mạnh

- Dễ bảo trì, dễ mở rộng khi thêm chức năng mới (ví dụ quản lý nhà cung cấp).
- Tách biệt rõ ràng nghiệp vụ và dữ liệu.
- Có thể triển khai nhiều giao diện khác nhau (WinForms, Web, Mobile) mà không ảnh hưởng tới phần nghiệp vụ.

4.2 Hạn chế

- Tốn công thiết kế ban đầu, phải tách riêng nhiều lớp.
- Với hệ thống nhỏ, có thể làm tăng độ phức tạp không cần thiết.
- Hiệu năng có thể chậm hơn một chút do qua nhiều tầng.

Kết luận

Kiến trúc 3-tier phù hợp với hệ thống nhiều nghiệp vụ, giúp dễ bảo trì, mở rộng và tăng bảo mật. Tuy nhiên, thiết kế ban đầu tốn công, có thể gây phức tạp cho hệ thống nhỏ và ảnh hưởng nhẹ đến hiệu năng. Cần cân nhắc kỹ trước khi áp dụng.