BÁO CÁO CLEAN ARCHITECTURE TRONG DỰ ÁN QUẢN LÝ THƯ VIỆN

1. Tổng quan dự án

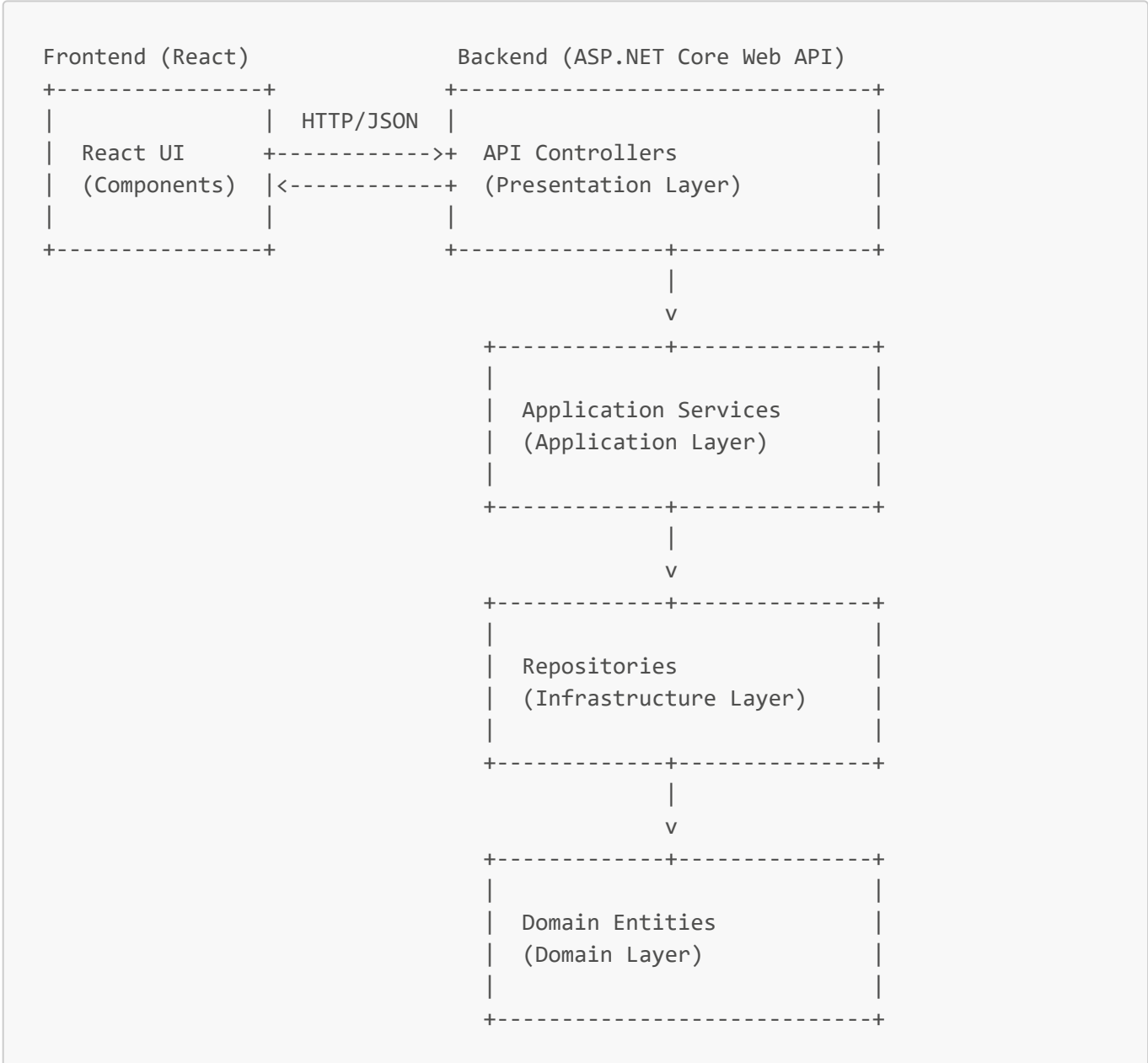
Dự án Quản lý Thư viện được xây dựng theo mô hình Clean Architecture với hai thành phần chính:

- **Backend**: ASP.NET Core Web API (C#)
- **Frontend**: React (JavaScript)

Dự án cho phép quản lý sách, độc giả và các hoạt động mượn/trả sách trong thư viện, sử dụng lưu trữ dữ liệu dạng file JSON.

2. Luồng hoạt động và dữ liệu

2.1. Sơ đồ kiến trúc tổng thể



2.2. Luồng xử lý yêu cầu điển hình

Quy trình xử lý dữ liệu từ frontend đến backend tuân theo các bước sau:

1. **React Component gửi yêu cầu**: Người dùng tương tác với giao diện (ví dụ: nhấn nút "Thêm sách")
2. **HTTP Request**: Component gọi API service để gửi request đến backend
3. **API Controller**: Controller nhận request, xác thực dữ liệu đầu vào
4. **Application Service**: Controller gọi service tương ứng để xử lý nghiệp vụ
5. **Repository**: Service sử dụng repository để truy xuất/cập nhật dữ liệu
6. **Domain Logic**: Các quy tắc nghiệp vụ được kiểm tra và áp dụng
7. **Lưu trữ dữ liệu**: Repository lưu dữ liệu vào file JSON
8. **Phản hồi**: Kết quả được trả về theo chiều ngược lại

3. Điểm mạnh của mô hình Clean Architecture

3.1. Tách biệt các mối quan tâm

Mỗi lớp trong kiến trúc có vai trò và nhiệm vụ riêng biệt:

- **Domain Layer**: Chứa các entity và business rules cốt lõi
- **Application Layer**: Chứa các use cases và dịch vụ ứng dụng
- **Infrastructure Layer**: Chứa các triển khai cụ thể (repositories, email service, etc.)
- **Presentation Layer**: Chứa các API controllers xử lý yêu cầu HTTP

3.2. Dễ dàng thay đổi và mở rộng

Nhờ vào nguyên tắc phụ thuộc hướng vào trong (Dependency Inversion), dự án có thể dễ dàng thay đổi chi tiết triển khai mà không ảnh hưởng đến logic nghiệp vụ. Ví dụ:

Thay đổi từ Entity Framework Core sang lưu trữ File JSON:

- Chỉ cần tạo các class triển khai repository mới (`FileBookRepository`, `FileReaderRepository`, `FileLoanRepository`)
- Thay đổi đăng ký DI trong `Program.cs`
- Không cần chỉnh sửa Domain Entities, Application Services, hay Controllers

Đoạn code đăng ký phụ thuộc ban đầu với Entity Framework:

```
// Đăng ký DbContext
builder.Services.AddDbContext<LibraryDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"
)));

// Đăng ký Repositories
builder.Services.AddScoped<IBookRepository, BookRepository>();
builder.Services.AddScoped<IReaderRepository, ReaderRepository>();
builder.Services.AddScoped<ILoanRepository, LoanRepository>();
```

Thay đổi sang lưu trữ File JSON:

```
// Đăng ký File repositories
var dataFolder = Path.Combine(Directory.GetCurrentDirectory(), "Data");
builder.Services.AddScoped<IBookRepository>(provider =>
    new FileBookRepository(Path.Combine(dataFolder, "books.json")));
builder.Services.AddScoped<IReaderRepository>(provider =>
    new FileReaderRepository(Path.Combine(dataFolder, "readers.json")));
builder.Services.AddScoped<ILoanRepository>(provider =>
    new FileLoanRepository(Path.Combine(dataFolder, "loans.json"),
        provider.GetRequiredService<IBookRepository>(),
        provider.GetRequiredService<IReaderRepository>()));
```

3.3. Khả năng kiểm thử cao

Clean Architecture cho phép kiểm thử từng lớp một cách độc lập:

- **Domain Entities**: Có thể kiểm thử business rules mà không cần database
- **Application Services**: Có thể sử dụng mock repositories để kiểm thử use cases
- **Infrastructure**: Có thể kiểm thử các repository implementation riêng biệt

3.4. Độc lập với frameworks

Logic nghiệp vụ không phụ thuộc vào framework. Nếu muốn chuyển từ ASP.NET Core sang một framework khác, chỉ cần thay đổi Presentation Layer.

4. Ví dụ thực tế về luồng dữ liệu

4.1. Thêm sách mới

1. **Frontend**:
    - Người dùng điền form thêm sách và nhấn nút "Thêm sách"
    - React component gọi API service: `createBook(bookData)`
2. **Backend**:
    - `BooksController.CreateBook()` nhận request, xác thực dữ liệu
    - Controller gọi `BookService.AddBookAsync(bookDto)`
    - Service kiểm tra xem ISBN đã tồn tại chưa qua `_bookRepository.GetByISBNAsync(bookDto.ISBN)`
    - Nếu hợp lệ, tạo entity mới: `new Book(bookDto.ISBN, bookDto.Title, bookDto.Author, bookDto.PublicationYear)`
    - Gọi `_bookRepository.AddAsync(book)` để lưu sách
    - `FileBookRepository` chuyển đổi entity thành JSON và lưu vào file
3. **Phản hồi**:
    - Controller trả về status code 201 Created với ID của sách mới
    - Frontend hiển thị thông báo thành công và chuyển hướng đến trang danh sách sách

4.2. Mượn sách

1. **Frontend**:
    - Người dùng chọn sách, độc giả và ngày hạn trả, rồi nhấn "Tạo phiếu mượn"
    - React component gọi API service: `createLoan(loanData)`
2. **Backend**:
    - `LoansController.CreateLoan()` nhận request
    - Controller gọi `LoanService.CreateLoanAsync()`
    - Service lấy thông tin sách và độc giả qua repository
    - Kiểm tra xem sách có sẵn không
    - Tạo entity Loan mới và lưu qua repository
    - Cập nhật trạng thái sách thành "không có sẵn"
3. **Phản hồi**:
    - Controller trả về status code 201 Created
    - Frontend hiển thị thông báo thành công và chuyển hướng tới danh sách phiếu mượn

5. Hướng dẫn sử dụng dự án

5.1. Yêu cầu hệ thống

- **.NET SDK**: Phiên bản 7.0 hoặc mới hơn
- **Node.js**: Phiên bản 16.0 hoặc mới hơn
- **npm**: Phiên bản 8.0 hoặc mới hơn

5.2. Thiết lập Backend (ASP.NET Core)

1. **Mở Terminal/Command Prompt và di chuyển đến thư mục backend**:

```
cd LibraryManager.API
```

2. **Khôi phục các gói NuGet**:

```
dotnet restore
```

3. **Xây dựng ứng dụng**:

```
dotnet build
```

4. **Chạy ứng dụng**:

```
dotnet run
```

Backend API sẽ được chạy tại `https://localhost:7233` (hoặc một port khác tùy cấu hình).

5.3. Thiết lập Frontend (React)

1. **Mở Terminal/Command Prompt khác và di chuyển đến thư mục frontend**:

```
cd library_manager_front_end
```

2. **Cài đặt các gói phụ thuộc**:

```
npm install
```

3. **Chạy ứng dụng React**:

```
npm start
```

Frontend sẽ được chạy tại `http://localhost:3000`.

5.4. Lưu ý quan trọng

- Đảm bảo backend đang chạy trước khi khởi động frontend
- Kiểm tra cấu hình CORS trong backend để cho phép kết nối từ frontend
- Trong file `src/services/api.js`, đảm bảo baseURL đúng với địa chỉ backend
- Dữ liệu sẽ được lưu trong thư mục `Data` bên trong thư mục backend, dưới dạng các file JSON

5.5. Các chức năng chính

1. **Quản lý sách**:
    - Xem danh sách sách
    - Thêm, sửa, xóa sách
    - Xem thông tin chi tiết sách
2. **Quản lý độc giả**:
    - Xem danh sách độc giả
    - Thêm, sửa, xóa độc giả
    - Xem thông tin chi tiết độc giả
3. **Quản lý mượn/trả sách**:
    - Tạo phiếu mượn mới

- Xem danh sách phiếu mượn (tất cả và đang hoạt động)
- Trả sách

6. Kết luận

Mô hình Clean Architecture giúp tạo ra ứng dụng có cấu trúc rõ ràng, dễ mở rộng và dễ bảo trì. Việc tách biệt các lớp và tuân theo nguyên tắc phụ thuộc hướng vào trong giúp ứng dụng linh hoạt hơn trước các thay đổi, đặc biệt là thay đổi về cơ sở dữ liệu hoặc frameworks.

Dự án Quản lý Thư viện là một ví dụ cụ thể về cách triển khai Clean Architecture trong thực tế, kết hợp frontend React và backend ASP.NET Core để tạo ra một ứng dụng hoàn chỉnh.