TRƯỜNG ĐẠI HỌC SÀI GÒN

Khoa Công nghệ thông tin

BÁO CÁO BÀI TẬP Lab 3

KIỂM THỬ PHẦN MỀM

ĐỀ BÀI: THIẾT KẾ KIẾN TRÚC CHUẨN C4 MODEL

GV hướng dẫn: ThS. Đỗ Như Tài

Lớp: DCT122C3 – Mã học phần: 841408

Thành viên: Lê Song Nhật Quyền – 3122411174

Nguyễn Lê Nhật Minh – 3122411125

Trần Minh Trí – 3122411222

Bùi Văn Tiến – 3122411206

Thành phố Hồ Chí Minh, tháng 9 năm 2025

BẢNG PHÂN CÔNG CỦA NHÓM

Họ tên các thành viên thực hiện	Nội dung công việc	Tiến độ công việc
Lê Song Nhật Quyền (Trưởng nhóm)	**Bài 1.** Vẽ lại sơ đồ kiến trúc SPA (Single Page Application)	100%
	Bài 3. Vẽ sơ đồ API của hệ thống	100%
Trần Minh Trí	**Bài 6.** Vẽ sơ đồ C3 – Component (High-Level) và giải thích	100%
	Bài 7. Vẽ sơ đồ C3 – Component (Module-Level) sau	100%
Nguyễn Lê Nhật Minh	**Bài 2**. Vẽ sơ đồ triển khai CI/CD	100%
	Bài 8. Vẽ sơ đồ xử lý 1 request	100%
Bùi Văn Tiến	**Bài 4.** Vẽ sơ đồ C1 - System Context và giải thích	100%
	Bài 5. Vẽ sơ đồ C2 – Container và giải thích	100%
Lê Song Nhật Quyền Trần Minh Trí	Các hình trên mô tả hệ thống (theo C4 model) cho dự án *Modular Monolith with DDD* của Kamil Grzybek	100%
Nguyễn Lê Nhật Minh Bùi Văn Tiến	Các hình trên mô tả hệ thống (theo C4 model) cho dự án hệ thống quản lý thư viện trực tuyến	100%

LỜI CAM ĐOAN

Tôi tên là Lê Song Nhật Quyền, xin đại diện nhóm chịu trách nhiệm và cam đoan rằng:

Những kết quả nghiên cứu được trình bày trong bài tiểu luận là công trình của riêng chúng tôi dưới sự hướng dẫn của giảng viên ThS. Đỗ Như Tài

Chúng tôi đã không sao chép bất kỳ thông tin nào từ các nguồn khác mà không được ghi nhận. Chúng tôi cam đoan không vi phạm bất kỳ quyền sở hữu trí tuệ hoặc quyền tác giả của bất kỳ ai hoặc bất kỳ tổ chức nào.

Tôi cam đoan rằng những kết quả và nhận định đưa ra trong bài báo cáo là sự hiểu biết và đánh giá của chúng tôi dựa trên nghiên cứu tài liệu và kiến thức về ... . Chúng tôi đã cố gắng hết sức để cung cấp thông tin đầy đủ về các kiến thức của học phần kiểm thử phần mềm được đề cập trong bài. Chúng tôi cam đoan rằng bài tiểu luận này được thực hiện một cách độc lập và khách quan.

Xin chân thành cảm ơn

Sinh viên thực hiện

Lê Song Nhật Quyền

Nguyễn Lê Nhật Minh

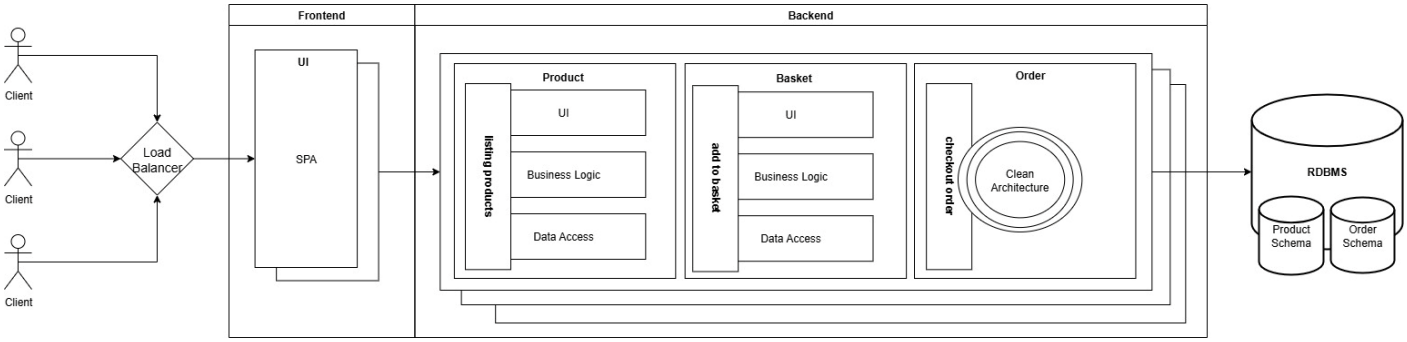
Trần Minh Trí

Bùi Văn Tiến

DANH MỤC HÌNH

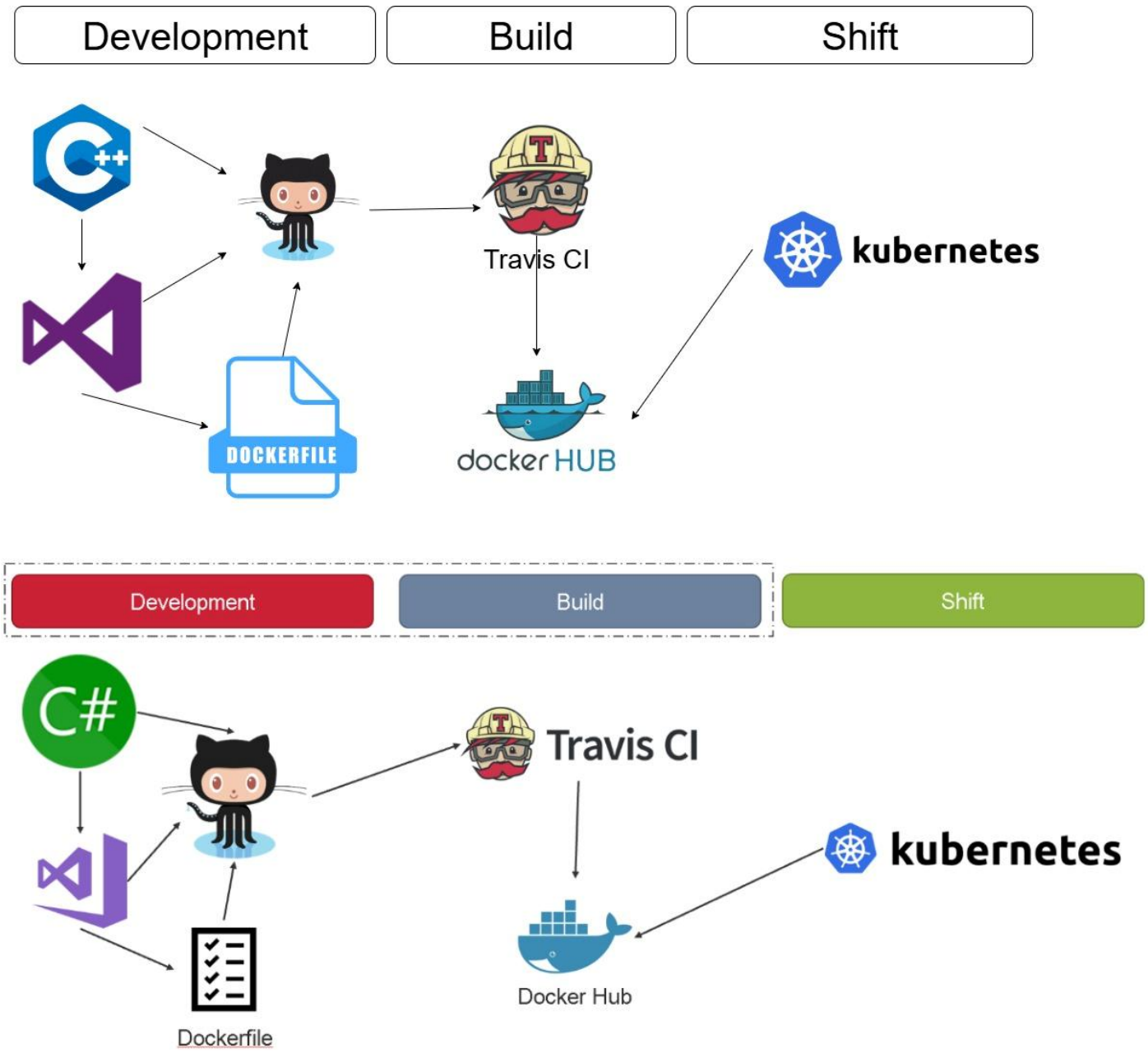
MỤC LỤC

Bài 1. Vẽ lại sơ đồ kiến trúc SPA (Single Page Application):



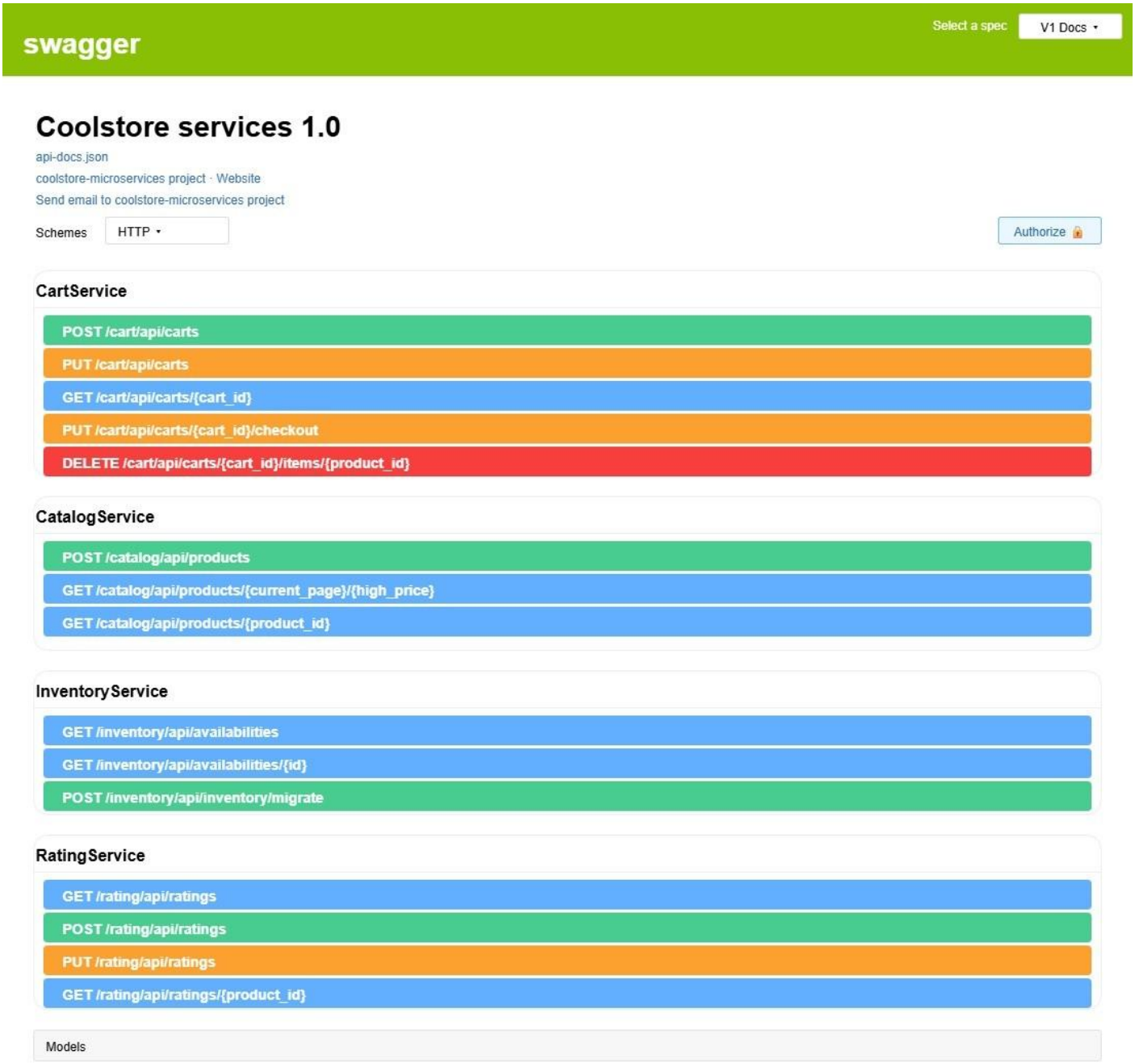
***Hình 1**. Tăng trải nghiệm người dùng với SPA*

Bài 2. Vẽ sơ đồ triển khai CI/CD:



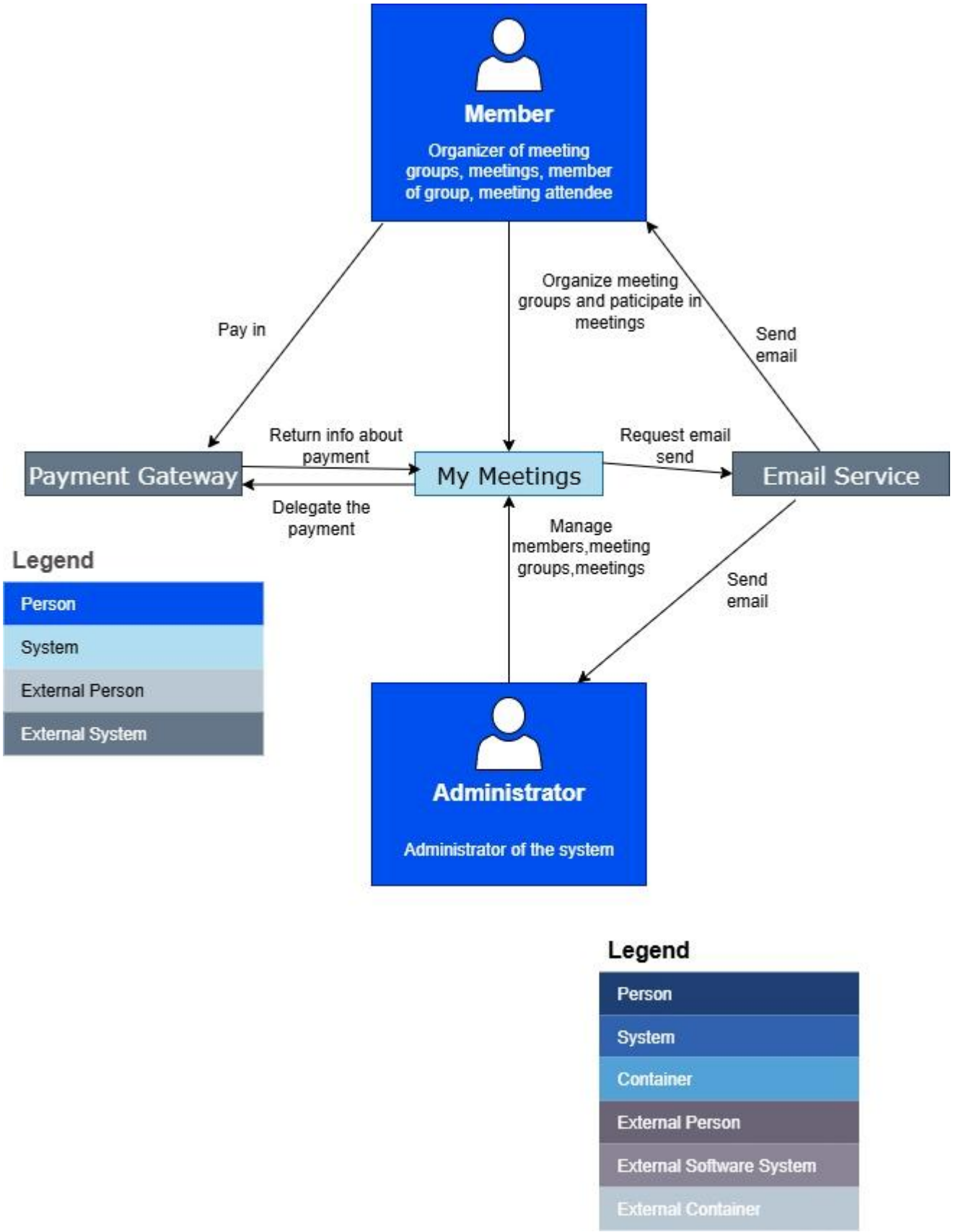
***Hình 2**. BT vẽ lại cho sơ đồ triển khai CI/CD*

Bài 3. Vẽ sơ đồ API của hệ thống:



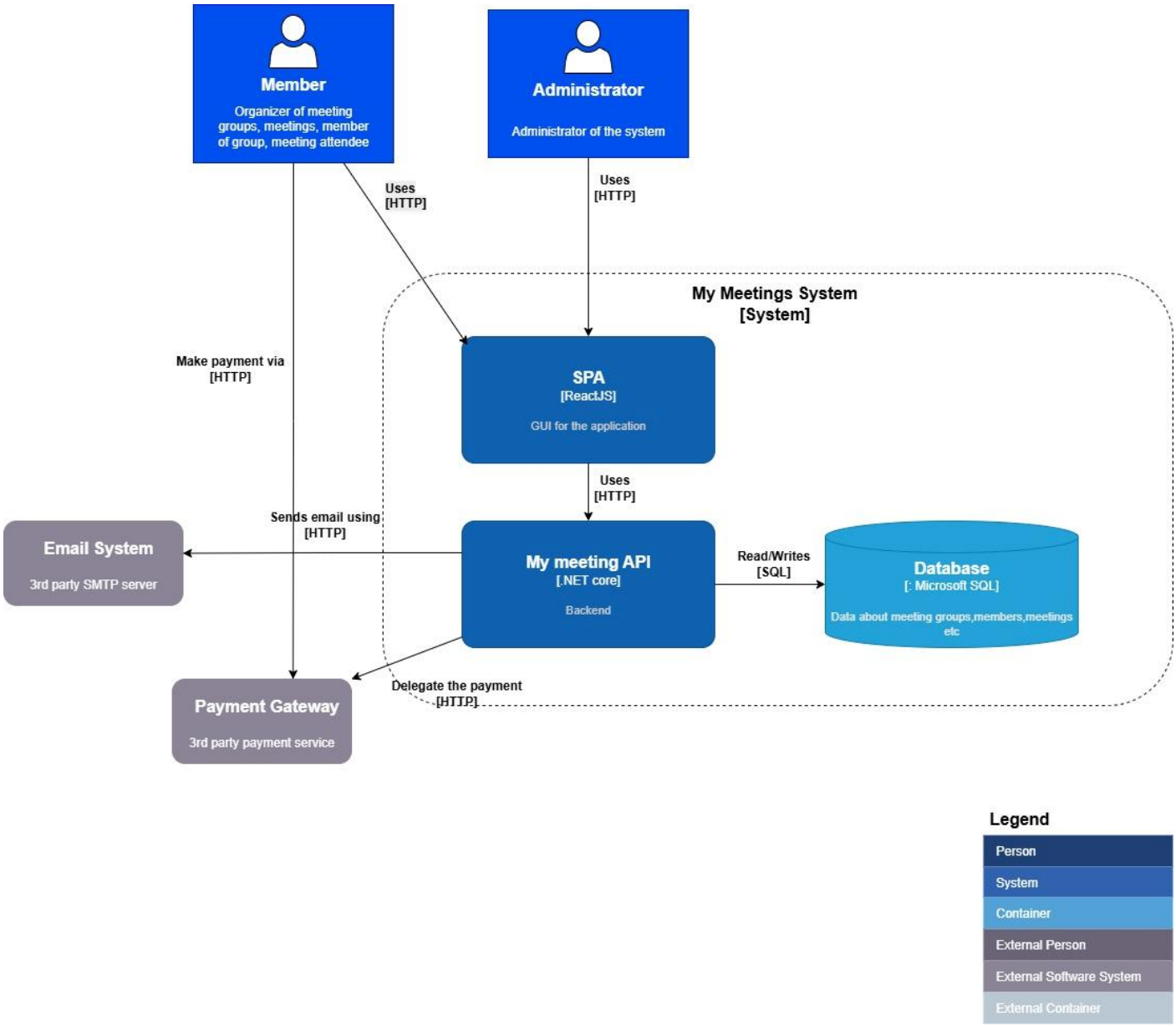
***Hình 3**. BT vẽ lại cho sơ đồ API của Coolstore website (sử dụng Swagger)*

Bài 4. Vẽ sơ đồ C1 - System Context và giải thích:



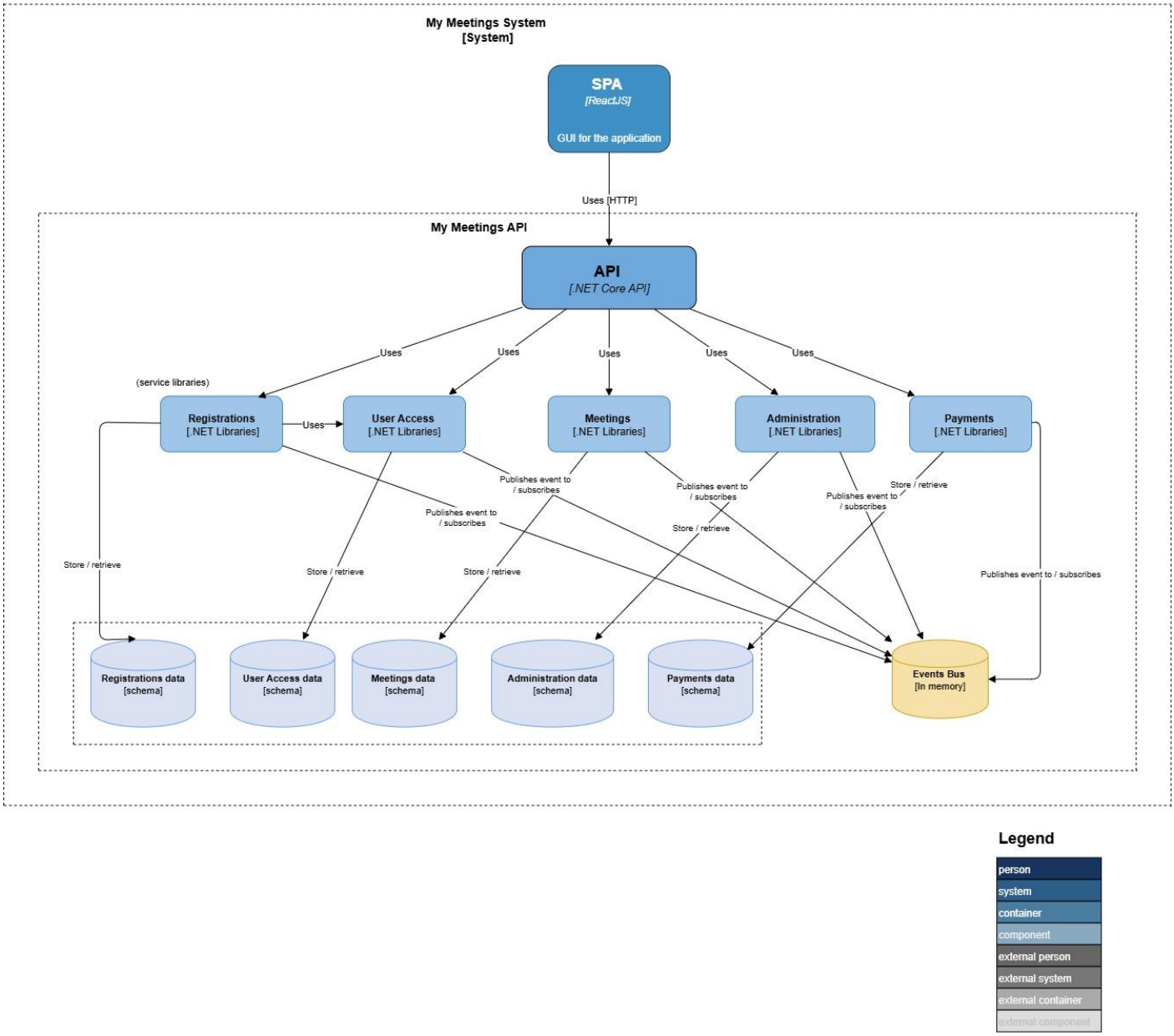
***Hình 4**. BT vẽ lại cho phần C1 – System Context trong C4 model*

Bài 5. Vẽ sơ đồ C2 – Container và giải thích:



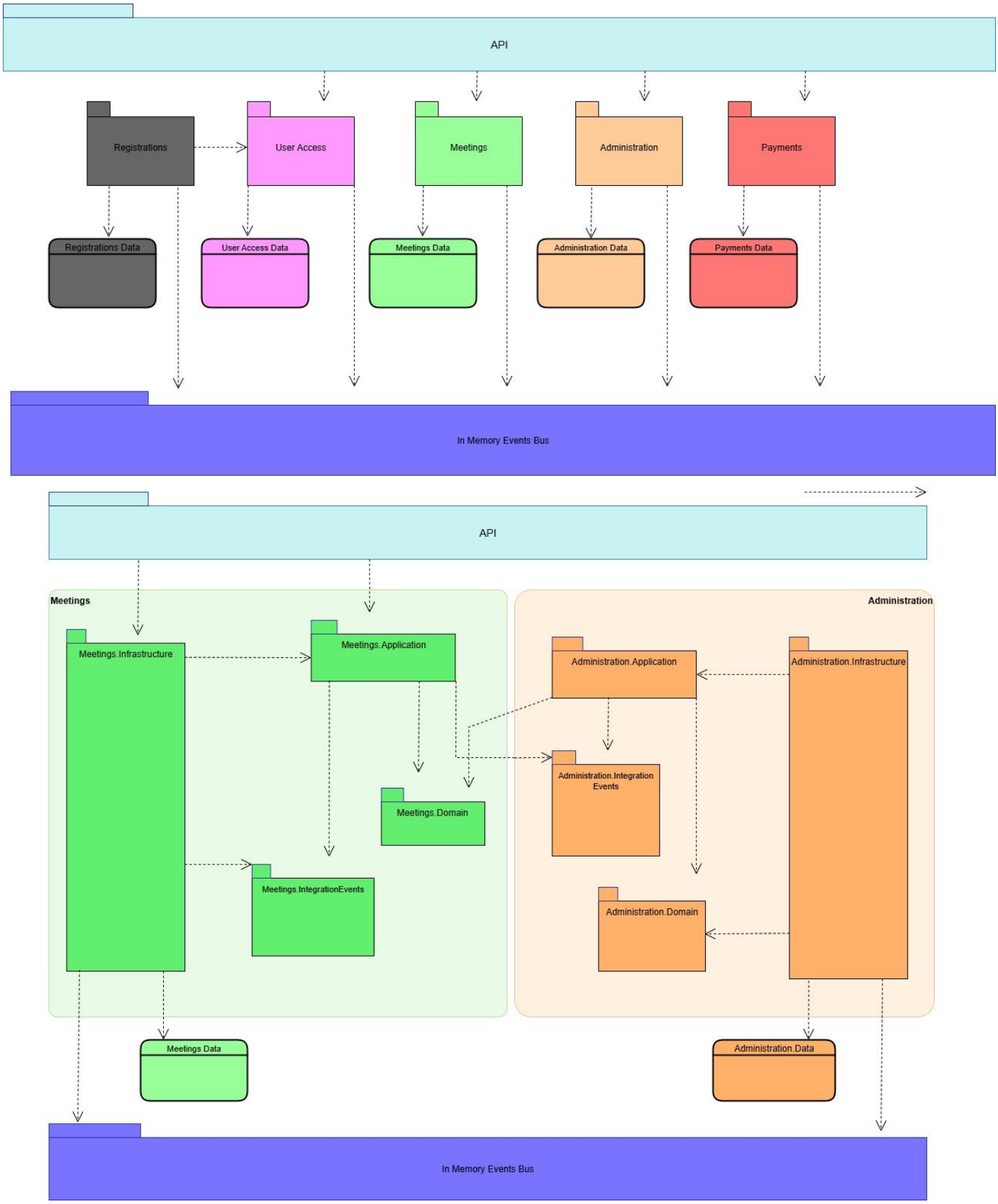
***Hình 5**. BT vẽ lại cho phần C2 – Container trong C4 model*

Bài 6. Vẽ sơ đồ C3 – Component (High-Level) và giải thích:



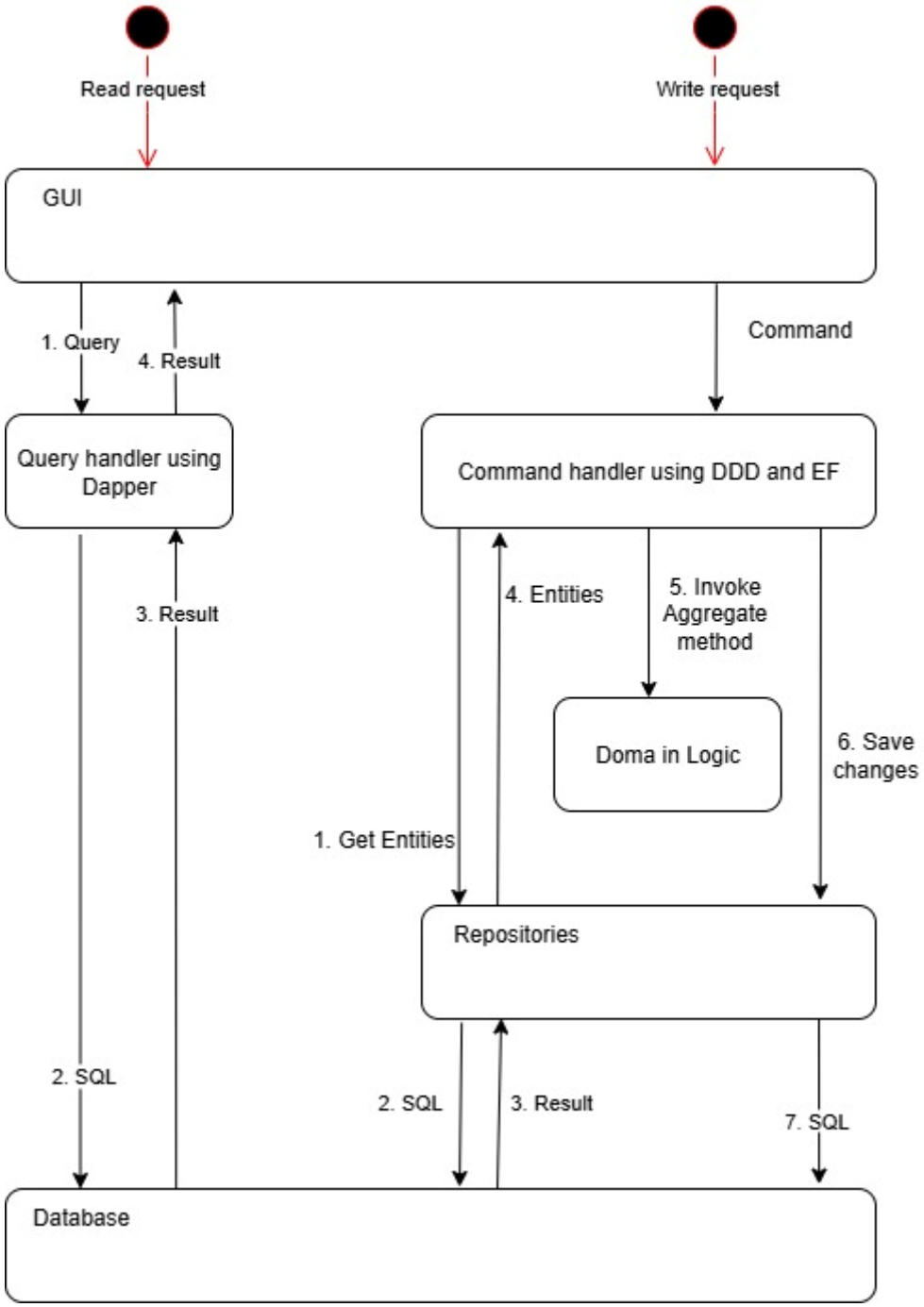
***Hình 6**. BT vẽ lại cho phần C3 – Component (High-Level) trong C4 model*

Bài 7. Vẽ sơ đồ C3 – Component (Module-Level) sau:



Hình 7. *BT vẽ lại cho phần C3 – Component (Module-Level) trong C4 model*

Bài 8. Vẽ sơ đồ xử lý 1 request:



Hình 8. *BT vẽ lại cho sơ đồ xử lý 1 request*

VÍ DỤ MINH HỌA (BEST PRATICES)

Tham khảo:

https://github.com/kgrzybek/modular-monolith-with-ddd https://vietnam-devs.github.io/coolstore-microservices

Tổng quan kiến trúc

Các hình trên mô tả hệ thống (theo tinh thần C4) dự án *Modular Monolith with DDD* của Kamil Grzybek.

- **C1**: Hệ thống Meetup-like với users/admin/payer, quy trình đề xuất-duyệt nhóm, cuộc họp, thanh toán.
- **C2**: Monolith theo module, **API mỏng + Event Bus trong bộ nhớ**, module sở hữu dữ liệu riêng.
- **C3**: Mỗi module tách **Application/Domain/Infrastructure/IntegrationEvents**, xử lý CQRS + events.
- **C4**: Code-level có ví dụ aggregate cho **Meeting Group** (được đính kèm sơ đồ trong README).

Mô tả chi tiết:

C1 — System Context (tóm lược)

- Hệ thống là một **Modular Monolith** phục vụ miền "Meeting Groups/Meetup": người dùng đăng ký tài khoản, đề xuất/duyệt nhóm họp, tạo cuộc họp, tham gia, bình luận; có thanh toán phí đăng ký/fee sự kiện.
- Các actor/chức năng cấp cao được hiện thực qua các **module** sau: **User Access**, **Registrations**, **Meetings**, **Administration**, **Payments**; API mỏng ở phía trước, và một **In-Memory Events Bus** dùng để tích hợp bất đồng bộ giữa các module.

C2 — Container View

- **API (ASP.NET Core REST)**
  Vai trò: nhận request → xác thực/ủy quyền (dựa vào User Access) → gửi **Command/Query** tới module đích → trả response. Không chứa business logic. API nói chuyện với module qua giao diện nhỏ (ports) do từng module cung cấp.
- **Các Module nghiệp vụ** (nằm trong cùng tiến trình — monolith, nhưng đóng gói chặt chẽ và độc lập dữ liệu):
  - **User Access**: xác thực/ủy quyền.
  - **Registrations**: đăng ký người dùng.
  - **Meetings**: bounded context tạo nhóm họp/cuộc họp, quản lý tham dự, bình luận…
  - **Administration**: duyệt/khước từ **Meeting Group Proposal** và các tác vụ quản trị.
  - **Payments**: thanh toán subscription, fee cuộc họp, gia hạn.
  - **In-Memory Events Bus**: cơ chế Pub/Sub để **các module chỉ tích hợp bất đồng**

bộ bằng sự kiện (cấm gọi trực tiếp qua method).

- **Các giả định/luật tích hợp quan trọng**
  - API không có logic ứng dụng;
  - API ↔ Module qua interface nhỏ (Command/Query);
  - **Module-to-Module chỉ qua Event Bus**;
  - **Mỗi module sở hữu dữ liệu riêng trong schema riêng** (không chia sẻ), và có thể tách ra DB riêng khi cần;
  - Chỉ được phụ thuộc vào **assemblies IntegrationEvents** của module khác;
  - **Mỗi module có Composition Root/IoC riêng** và được **API khởi tạo**;
  - Mức đóng gói cao (public tối thiểu).

C3 — Component View (mẫu cấu trúc bên trong từng module)

Mỗi **module** tuân theo **Clean Architecture** và chia thành các *submodules/assemblies* chuẩn:

- **Application**: xử lý use case (Command/Query), domain events, integration events, internal commands.
- **Domain**: mô hình domain (DDD) của **Bounded Context** tương ứng.
- **Infrastructure**: khởi tạo module, nền tảng chạy nền, truy cập dữ liệu, giao tiếp Event Bus & hệ thống ngoài.
- **IntegrationEvents**: hợp đồng sự kiện công bố ra Event Bus (đây là **điểm duy nhất** cho module khác phụ thuộc).

Nguyên tắc domain áp dụng: encapsulation cao (mặc định private/internal), persistence-ignorant (POCO, không phụ thuộc hạ tầng), domain giàu hành vi, tránh "primitive obsession" bằng Value Object, đặt tên theo ngôn ngữ nghiệp vụ, dễ kiểm thử.

Dòng chảy xử lý (Request & CQRS)

1. **API** nhận HTTP request, xác thực/ủy quyền bằng **User Access** →
2. API gửi **Command/Query** tới **Application** của module đích →
3. Application gọi **Domain** để thực thi nghiệp vụ, phát **Domain Event** nội bộ nếu có →
4. Khi cần tích hợp liên-module, module **publish Integration Event** lên **In-Memory Events Bus** để module khác **subscribe** và phản ứng bất đồng bộ. (Phong cách EDA trong một monolith.)

Dữ liệu & biên giới module

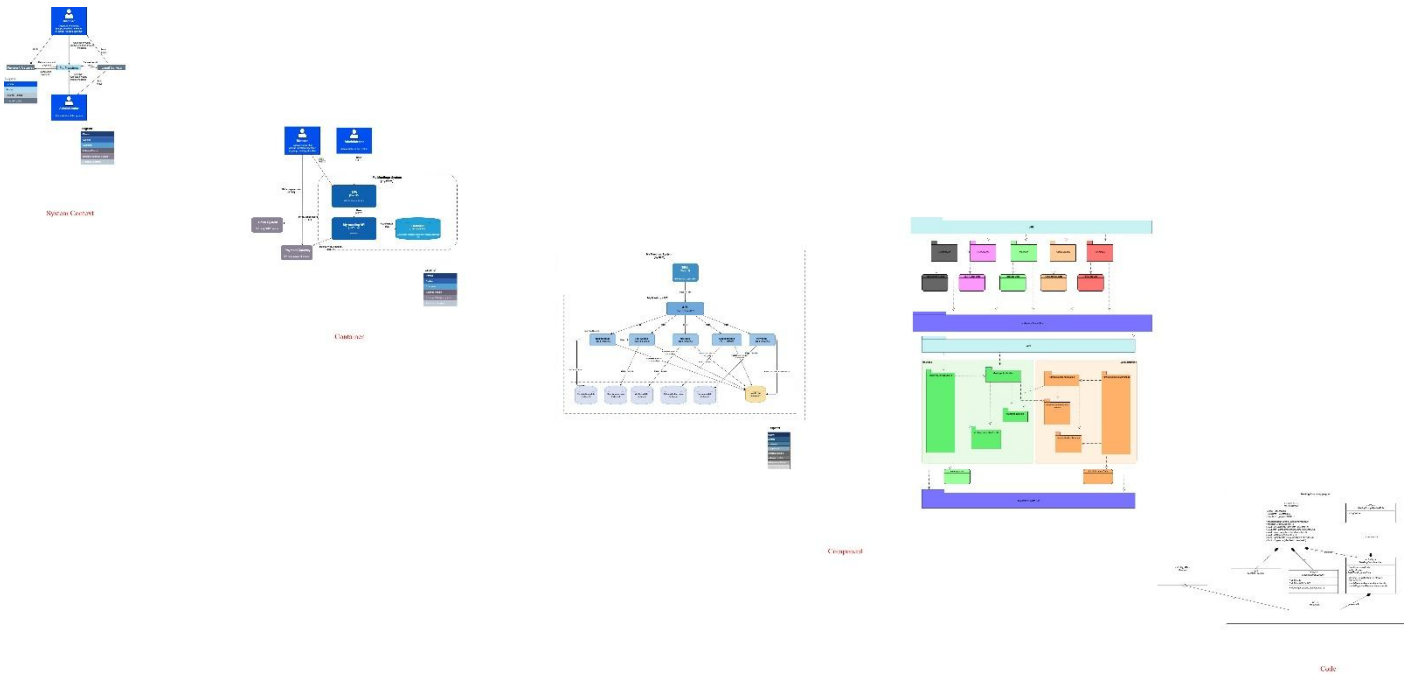
- **Data ownership**: mỗi module quản lý **schema dữ liệu riêng**, không chia sẻ bảng; có thể nâng cấp thành DB riêng nếu cần.
- **Giao tiếp dữ liệu** giữa module: qua **Integration Events** (không đọc chéo DB, không gọi method chéo).

Bảo mật & khởi tạo

- **User Access** phụ trách authN/authZ; API dựa vào đây trước khi ủy quyền use case vào module.
- **Khởi tạo module**: API gọi **Initialize(...)** tĩnh của từng module trong Startup, truyền cấu hình (connection string, logger, email config…), dựng Composition Root, khởi động **Quartz** (jobs) và **Events Bus** cho module.

Kiểm thử & chất lượng (khung hỗ trợ trong repo)

Repo minh họa: **unit test domain**, **integration test**, **system integration testing**, **event sourcing**, **database change management**, **CI**, **static analysis**, **mutation testing**—dùng để đảm bảo kiến trúc & hiện thực ở mức production-ready.



***Hình 9**. Hình vẽ cấu trúc chuẩn C4 model (System Context -> Code) cho dự án Modular Monolith with DDD của Kamil Grzybek*

BÀI TẬP ỨNG DỤNG

HỆ THỐNG QUẢN LÝ THƯ VIỆN TRỰC TUYẾN

Business Context

- **Mục tiêu kinh doanh**: Giúp thư viện hiện đại hóa việc mượn/trả sách, giảm phụ thuộc vào thủ thư, tăng trải nghiệm người dùng, và quản lý kho sách hiệu quả.
- **Các tác nhân chính**:
  - **Người đọc**: Tìm kiếm sách, mượn sách online, gia hạn, xem lịch sử mượn.
  - **Thủ thư/Admin**: Quản lý kho sách, xử lý yêu cầu mượn/trả, theo dõi phí trễ hạn.
  - **Hệ thống thanh toán**: Xử lý phí thành viên hoặc phí trễ hạn.
- **Ràng buộc & giá trị**: Hệ thống phải ổn định, bảo mật dữ liệu người đọc, cho phép nhiều người dùng
  đồng thời.

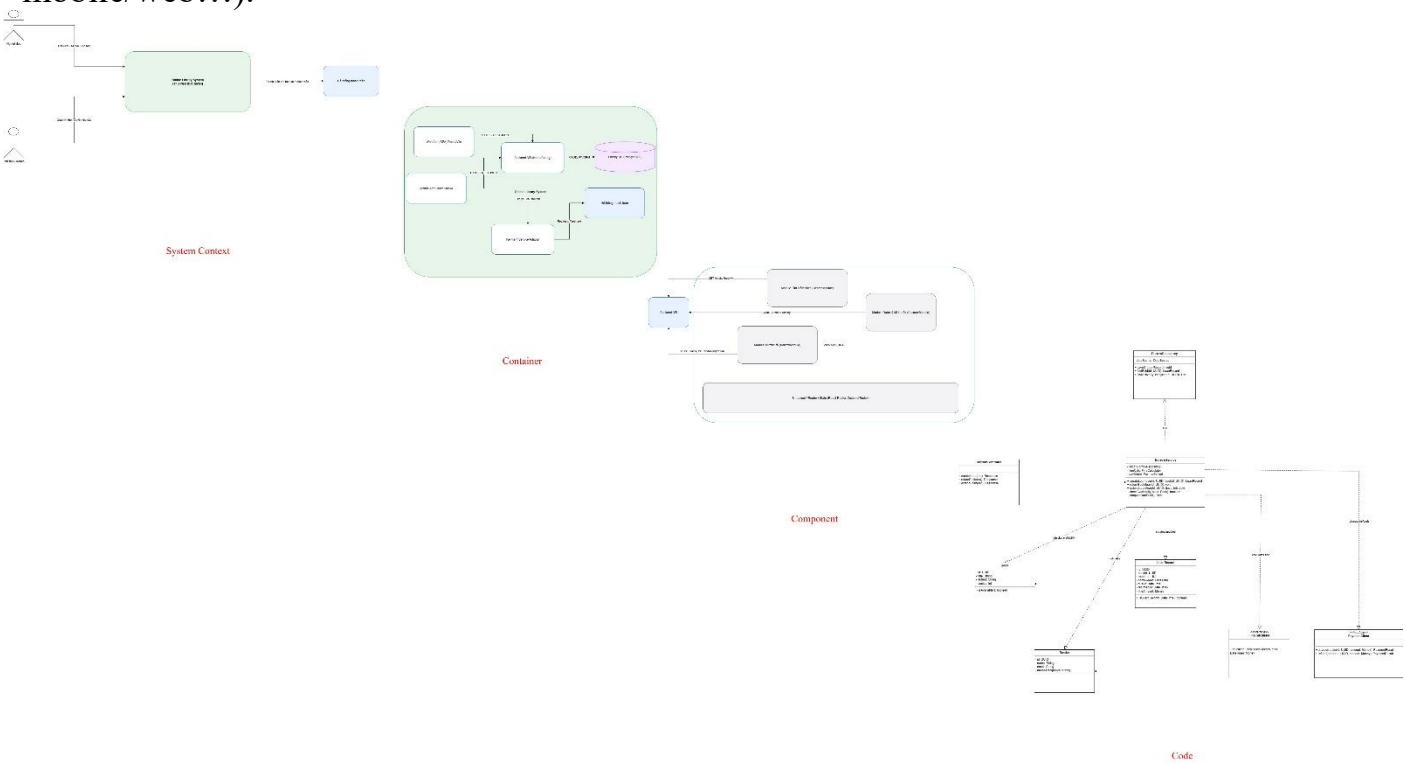
Business Context = **Mục tiêu kinh doanh + Các tác nhân + Giá trị mang lại + Ràng buộc chính**

Thiết kế kiến trúc phần mềm theo chuẩn C4 model theo 4 mức của C4:

- **Level 1 (Context Diagram)**: Hệ thống và các tác nhân bên ngoài.
- **Level 2 (Container Diagram)**: Các container (ứng dụng web, mobile app, database, service...).
- **Level 3 (Component Diagram)**: Các thành phần trong một container quan trọng.
- **Level 4 (Code/Implementation Diagram)**: Chi tiết lớp hoặc cấu trúc code.

Yêu cầu:

- **Mô tả**: Xây dựng hệ thống cho phép người dùng mượn/trả sách, quản lý kho sách, và thanh toán phí trễ hạn.
- **Yêu cầu**:
  - Vẽ sơ đồ **Context**: Thư viện online, Người dùng, Thủ thư, Hệ thống thanh toán.
  - Vẽ sơ đồ **Container**: Web App, Mobile App, Database, Payment Service.
  - Vẽ sơ đồ **Component**: Bên trong Web App, có module tìm kiếm, module mượn/trả, module quản lý tài khoản.
  - Vẽ sơ đồ **Code**: Ví dụ class diagram cho module "Quản lý mượn sách".
- Bạn có thể vẽ bằng **PlantUML, Structurizr DSL, hoặc draw.io** để thể hiện trực quan.
- Khi làm bài tập, hãy mô tả **các quyết định thiết kế chính** (ví dụ: tại sao dùng SPA, tại sao tách mobile/web…).



***Hình 10**. Hình vẽ cấu trúc chuẩn C4 model (System Context -> Code) cho dự án hệ thống quản lý thư viện trực tuyến*

TÀI LIỆU THAM KHẢO

[1] Tài liệu bài giảng của giảng viên Ths. Đỗ Như Tài cung cấp trên lớp