ỦY BAN NHÂN DÂN TP . HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP GIỮA KỲ

Học phần: Kiểm thử phần mềm

Đề tài: **Bài tập 3 – Thiết kế kiến trúc chuẩn C4 model**

Giảng viên hướng dẫn: **TS. Đỗ Như Tài**

Nhóm sinh viên thực hiện:

STT	Họ và tên	MSSV
1	**Trang Gia Huy**	**3122411068**
2	**Nguyễn Lê Quỳnh Hương**	**3122411078**

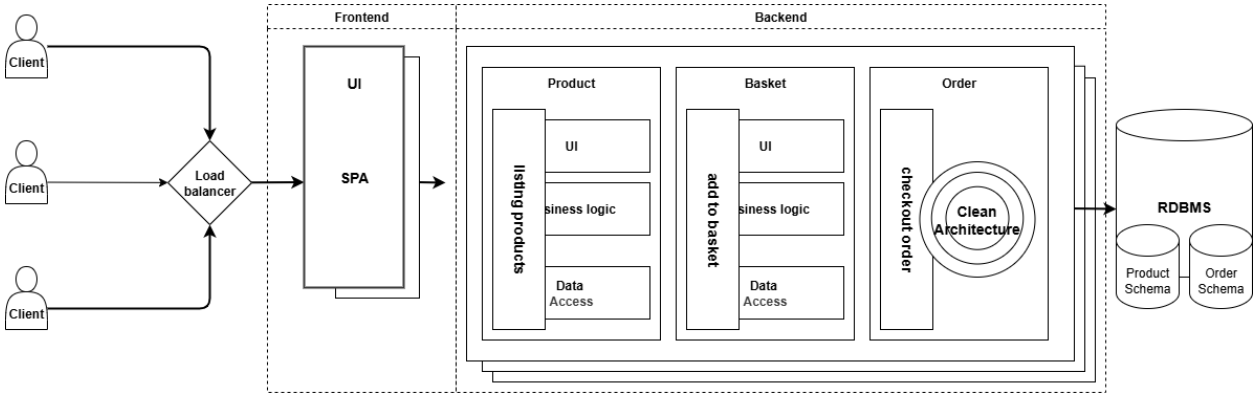
TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2025

MỤC LỤC

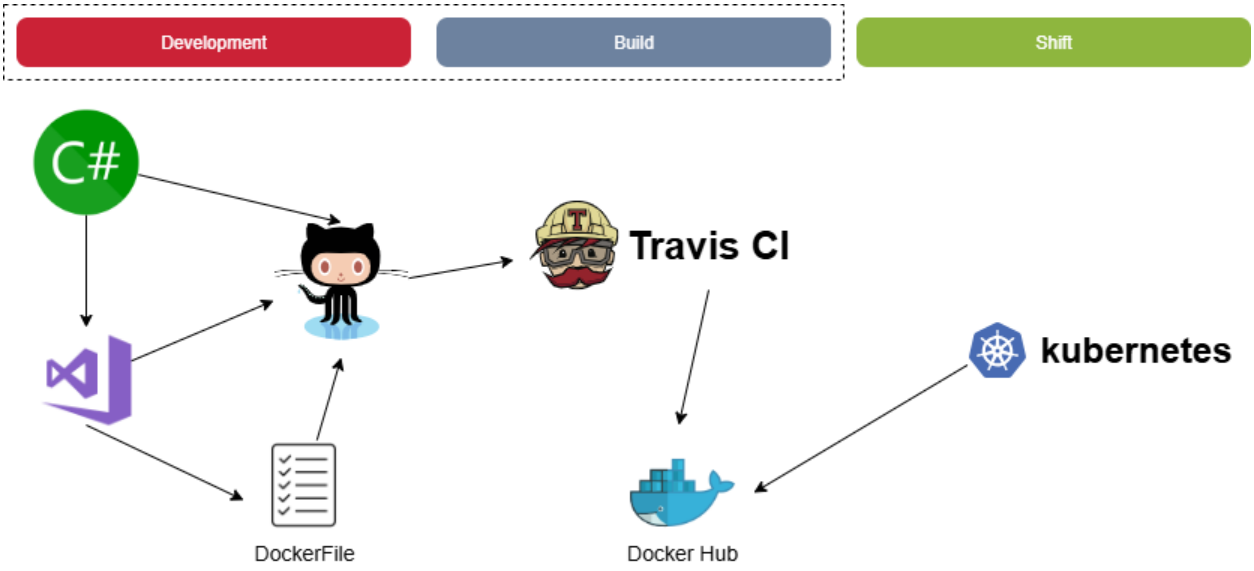
BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Họ và tên	MSSV	Nội dung công việc
1	Trang Gia Huy	3122411068	- Soạn báo cáo - Bài tập ứng dụng
2	Nguyễn Lê Quỳnh Hương	3122411078	- Bài 1, 2, 3, 4, 5, 6, 7, 8

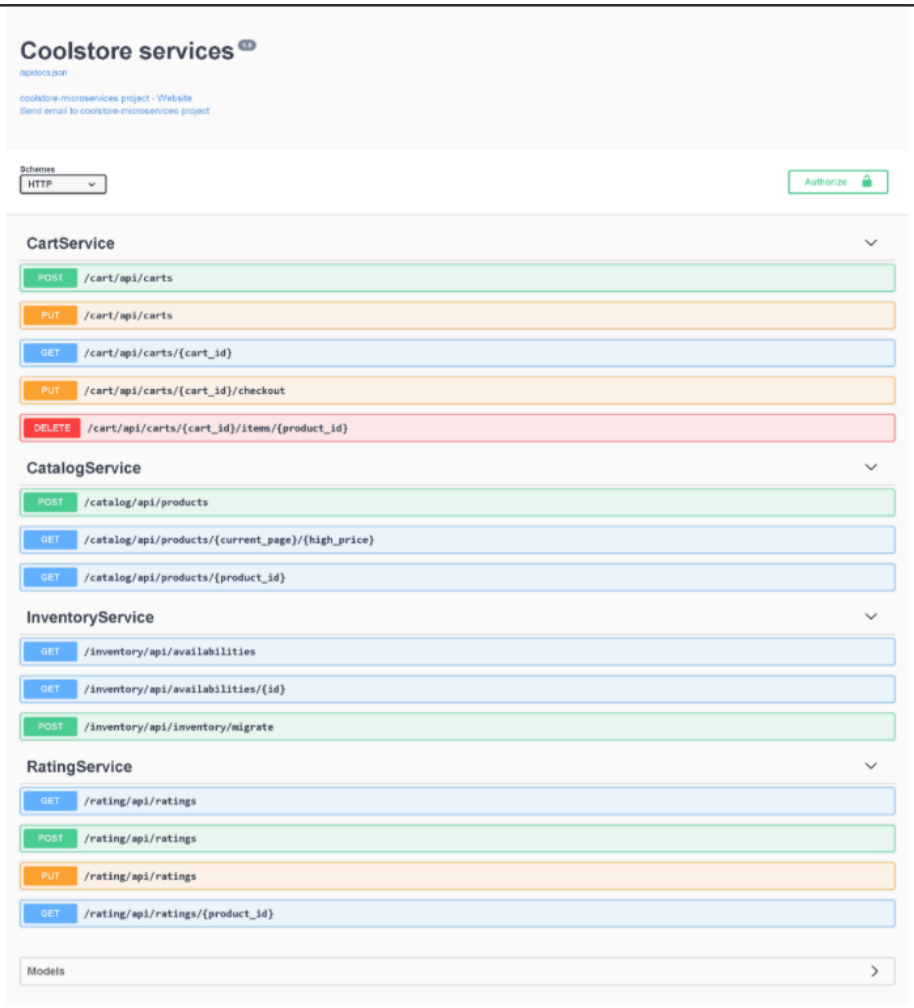
Bài 1. Vẽ lại sơ đồ kiến trúc SPA (Single Page Application):

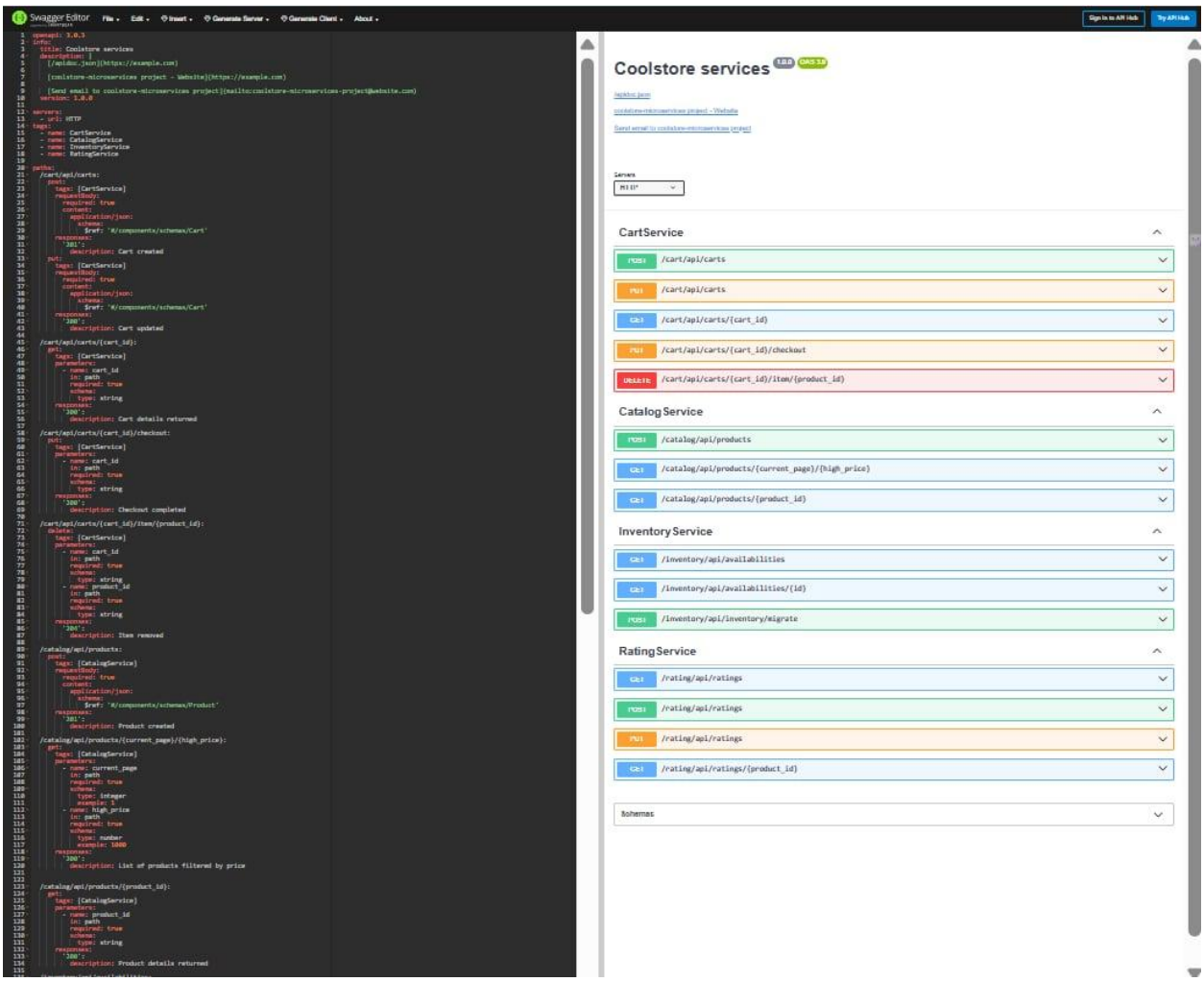


Bài 2. Vẽ sơ đồ triển khai CI/CD:

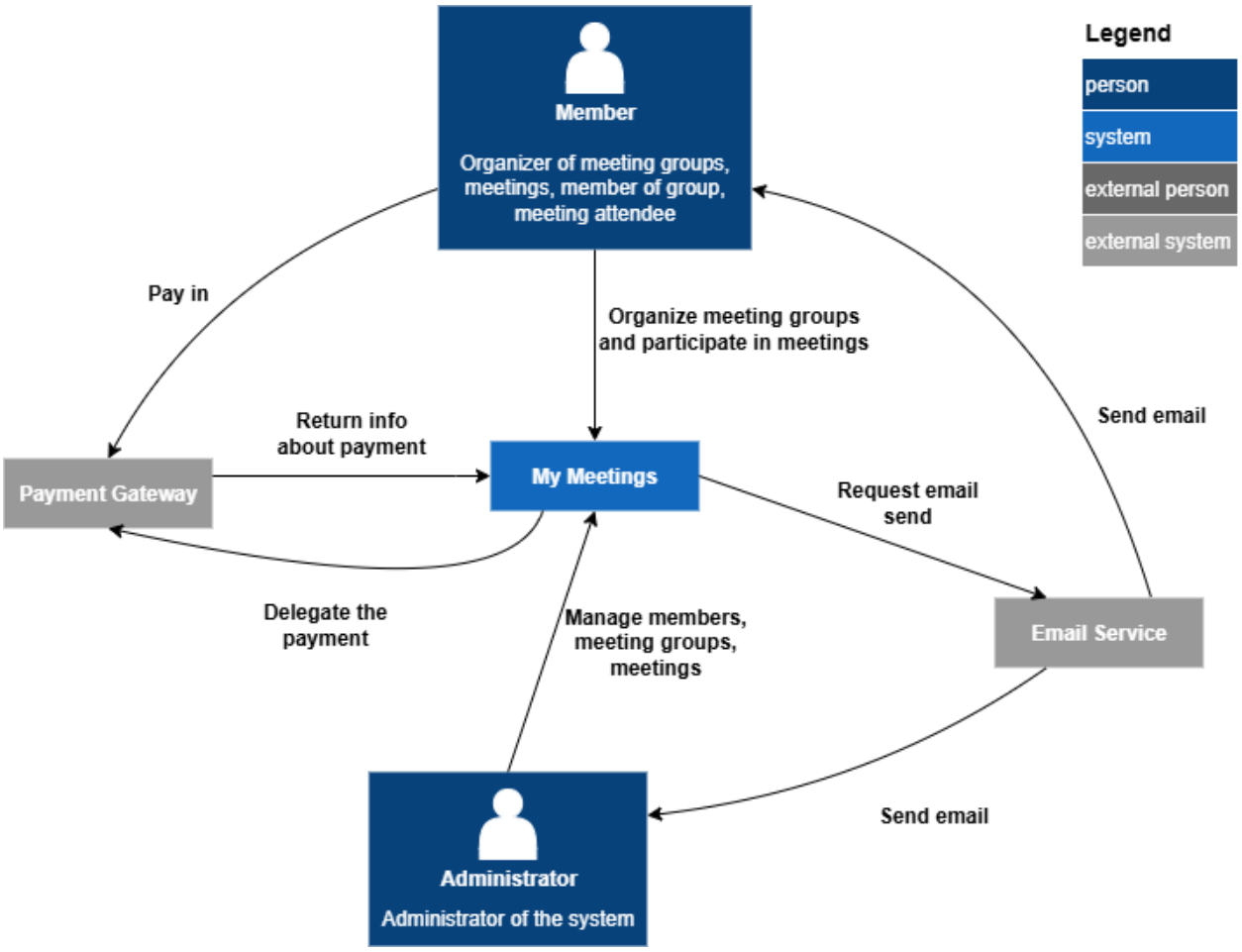


Bài 3. Vẽ sơ đồ API của hệ thống:




Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
About
Sign In to API Hub
Try API Hub
openapi: 3.0.3
info:
title: Coolstore services
description: |
[/apidoc.json](http://example.com)
[coolstore-microservices project - Website](https://example.com)
[Send email to coolstore-microservices project](mailto:coolstore-microservices-project@website.com)
version: 1.0.0
servers:
- url: HTTP
tags:
- name: CartService
- name: CatalogService
- name: InventoryService
- name: RatingService
paths:
/cart/api/carts:
post:
tags: [CartService]
requestBody:
required: true
content:
application/json:
schema:
$ref: '#/components/schemas/Cart'
responses:
'201':
description: Cart created
put:
tags: [CartService]
requestBody:
required: true
content:
application/json:
schema:
$ref: '#/components/schemas/Cart'
responses:
'200':
description: Cart updated
/cart/api/carts/{cart_id}:
get:
tags: [CartService]
parameters:
- name: cart_id
in: path
required: true
schema:
type: string
responses:
'200':
description: Cart details returned
/cart/api/carts/{cart_id}/checkout:
put:
tags: [CartService]
parameters:
- name: cart_id
in: path
required: true
schema:
type: string
responses:
'200':
description: Checkout completed
/cart/api/carts/{cart_id}/item/{product_id}:
delete:
tags: [CartService]
parameters:
- name: cart_id
in: path
required: true
schema:
type: string
- name: product_id
in: path
required: true
schema:
type: string
responses:
'200':
description: Item removed
/catalog/api/products:
post:
tags: [CatalogService]
requestBody:
required: true
content:
application/json:
schema:
$ref: '#/components/schemas/Product'
responses:
'201':
description: Product created
/catalog/api/products/{current_page}/{high_price}:
get:
tags: [CatalogService]
parameters:
- name: current_page
in: path
required: true
schema:
type: integer
example: 1
- name: high_price
in: path
required: true
schema:
type: number
example: 1000
responses:
'200':
description: List of products filtered by price
/catalog/api/products/{product_id}:
get:
tags: [CatalogService]
parameters:
- name: product_id
in: path
required: true
schema:
type: string
responses:
'200':
description: Product details returned
Coolstore services
1.0.0
OAS 3.0
/apidoc.json
coolstore-microservices project - Website
Send email to coolstore-microservices project
Servers
HTTP
CartService
/cart/api/carts
/cart/api/carts
/cart/api/carts/{cart_id}
/cart/api/carts/{cart_id}/checkout
/cart/api/carts/{cart_id}/item/{product_id}
Catalog Service
/catalog/api/products
/catalog/api/products/{current_page}/{high_price}
/catalog/api/products/{product_id}
Inventory Service
/inventory/api/availabilities
/inventory/api/availabilities/{id}
/inventory/api/inventory/migrate
RatingService
/rating/api/ratings
/rating/api/ratings
/rating/api/ratings
/rating/api/ratings/{product_id}
Schemas

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



Sơ đồ C1 - System Context

Giải thích: sơ đồ này cung cấp cái nhìn tổng quan về hệ thống phần mềm My Meetings và cách nó tương tác với các đối tượng bên ngoài, bao gồm người dùng và các hệ thống khác. Nó giúp trả lời các câu hỏi sau

- Hệ thống đang xây dựng là gì: hệ thống My Meetings.
- Ai là người sử dụng nó: Member và Administrator.
- Hệ thống phù hợp với môi trường hiện có như thế nào: nó tương tác với Payment Gateway và Email Service.

Cấu trúc: sơ đồ thể hiện hệ thống My Meetings được bao quanh bởi các tác nhân (actors) và hệ thống phần mềm khác mà nó tương tác. Sơ đồ này tập trung vào bức tranh lớn, không đi sâu vào chi tiết công nghệ, giúp cả người có chuyên môn và người không có chuyên môn đều có thể hiểu được.

Các thành phần: sơ đồ gồm 2 thành phần chính là người dùng và hệ thống phần mềm

Người dùng:

- Member (Thành viên): Là người tổ chức các nhóm họp, các cuộc họp, và là người tham dự cuộc họp. Họ sử dụng hệ thống để "tổ chức các nhóm họp và tham gia vào các cuộc họp."

- Administrator (Quản trị viên): Là người quản trị hệ thống. Họ sử dụng hệ thống để "quản lý các thành viên, các nhóm họp, và các cuộc họp."

.Hệ thống bên ngoài

- Payment Gateway: Một hệ thống thanh toán bên ngoài. Hệ thống My Meetings "ủy quyền thanh toán" cho nó. Thành viên tương tác với nó để "trả tiền vào," và nó sẽ "trả lại thông tin về thanh toán" cho My Meetings.

- Email Service: Một hệ thống dịch vụ email bên ngoài. My Meetings "yêu cầu gửi email" đến nó, và nó sẽ "gửi email" đến Thành viên và Quản trị viên.

Tương tác:

- My Meetings ↔ Member: "Tổ chức các nhóm họp và tham gia vào các cuộc họp."

- My Meetings ↔ Administrator: "Quản lý các thành viên, các nhóm họp, các cuộc họp."

- My Meetings ↔ Payment Gateway: "Ủy quyền thanh toán."

- Payment Gateway ↔ My Meetings: "Trả lại thông tin về thanh toán."

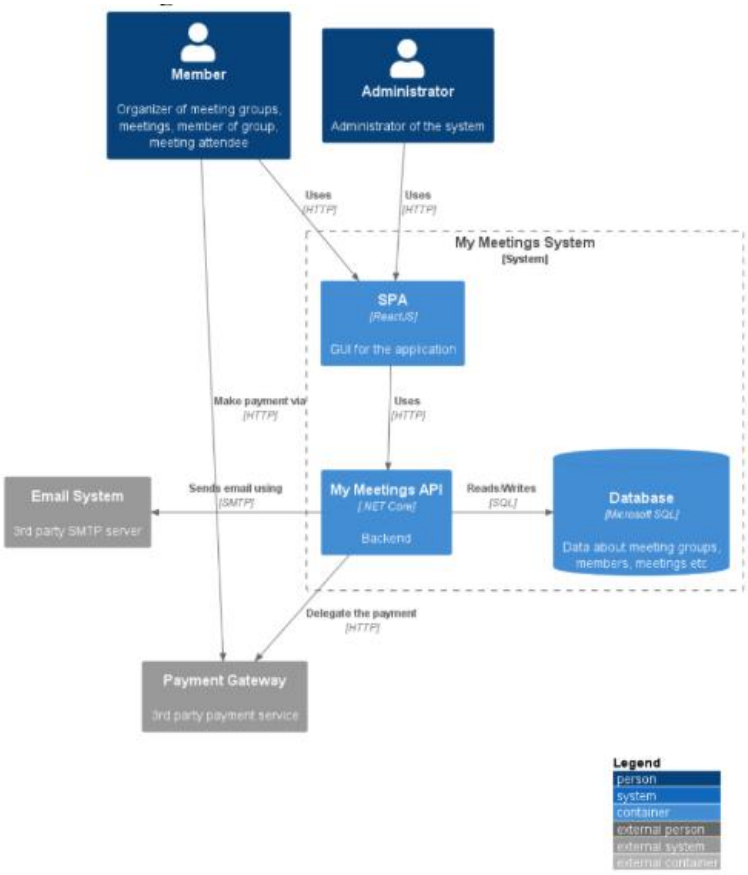
- My Meetings ↔ Email Service: "Yêu cầu gửi email."

- Email Service ↔ Member/Administrator: "Gửi email."

Đối tượng xem: phù hợp với cả người có chuyên môn kỹ thuật và người không có chuyên môn, bao gồm cả đội ngũ phát triển và các bên liên quan bên ngoài, giúp mọi người hiểu

được vai trò của nó trong môi trường hiện tại. Đây là điểm khởi đầu cho các cuộc thảo luận và là công cụ phân tích yêu cầu hiệu quả.

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



Giải thích: sơ đồ Container giúp làm rõ các quyết định công nghệ chính, cách các trách nhiệm được phân chia giữa các thành phần (containers) và cách chúng tương tác với nhau. Sơ đồ này cung cấp một cái nhìn chi tiết hơn về bên trong hệ thống so với sơ đồ ngữ cảnh (System Context Diagram)

Cấu trúc: 3 container chính:

1. SPA (Single-Page Application) [ReactJS]: Đây là giao diện người dùng (GUI) của ứng dụng.

Công nghệ: được xây dựng bằng ReactJS, chạy trên trình duyệt của người dùng.

- Member và Administrator tương tác trực tiếp với SPA này bằng giao thức HTTP. SPA này sử dụng HTTP để gọi đến My Meetings API.

2. My Meetings API [.NET Core]: Đây là phần backend của hệ thống, xử lý logic nghiệp vụ và giao tiếp với cơ sở dữ liệu.

Công nghệ: được xây dựng bằng .NET Core.

- Nhận yêu cầu từ SPA bằng giao thức HTTP. Đọc/ghi dữ liệu từ Database bằng giao thức SQL. Gửi email bằng giao thức SMTP đến Email System. Ủy quyền thanh toán cho Payment Gateway bằng HTTP.

3. Database [Microsoft SQL]: nơi lưu trữ tất cả dữ liệu của hệ thống, bao gồm thông tin về các nhóm họp, thành viên, cuộc họp, v.v.

Công nghệ: Sử dụng Microsoft SQL.

- My Meetings API đọc và ghi dữ liệu từ cơ sở dữ liệu này bằng giao thức SQL.

Tương tác với người dùng và hệ thống

- Member: Sử dụng SPA để tương tác với hệ thống. Thực hiện thanh toán trực tiếp với Payment Gateway qua giao thức HTTP.

- Administrator: Sử dụng SPA để quản lý hệ thống.

- Payment Gateway: Là một dịch vụ thanh toán của bên thứ ba. My Meetings API ủy quyền thanh toán cho nó bằng giao thức HTTP.

- Email System: Là một máy chủ SMTP của bên thứ ba. My Meetings API gửi email đến nó bằng giao thức SMTP.

Luồng hoạt động

- Người dùng truy cập hệ thống thông qua giao diện người dùng, đó là SPA (ReactJS).

- SPA gửi các yêu cầu đến My Meetings API (Backend) bằng giao thức HTTP.

- My Meetings API xử lý các yêu cầu này:

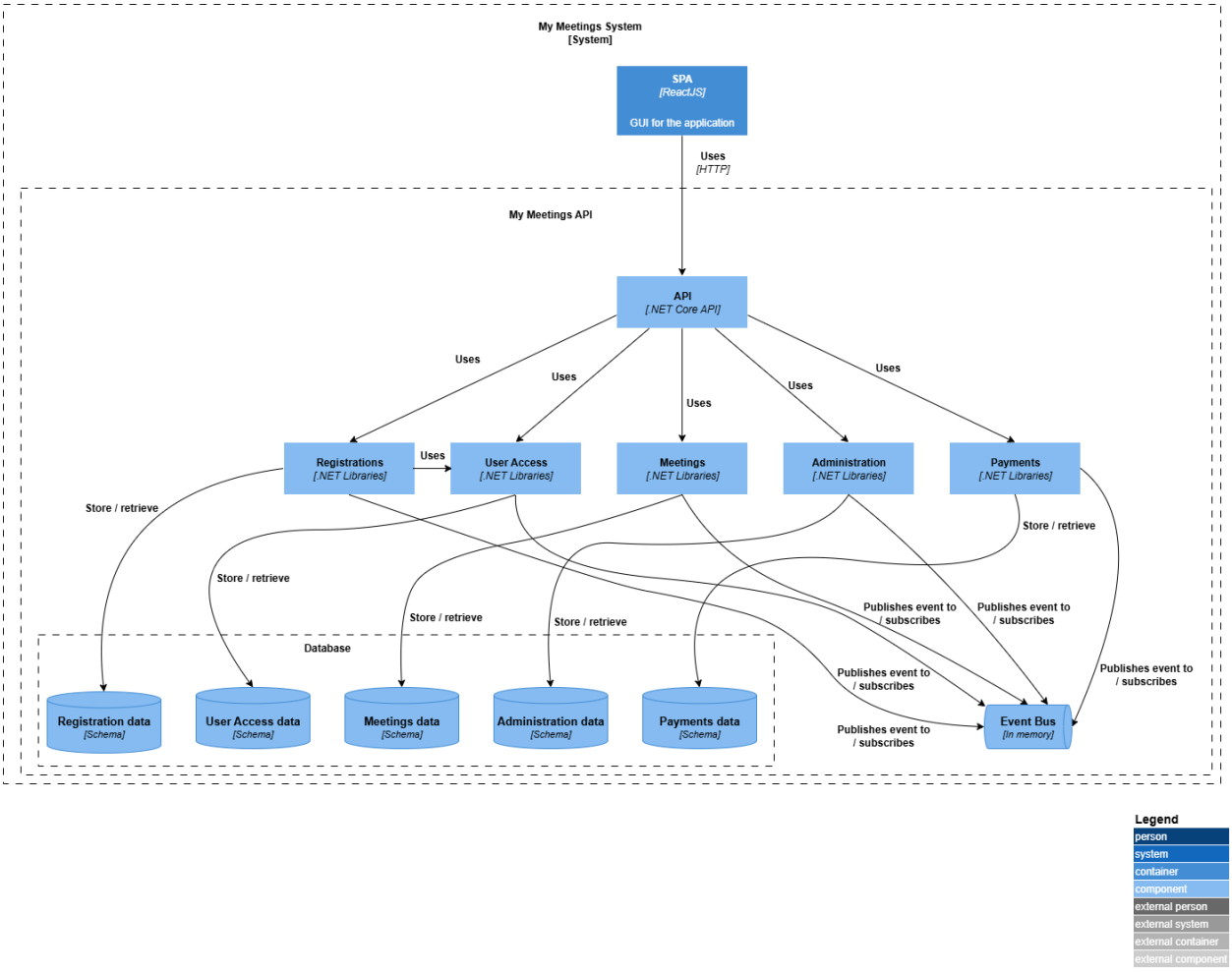
  - Nếu cần lưu trữ hoặc truy xuất dữ liệu, nó sẽ kết nối và tương tác với Database bằng SQL.

  - Nếu cần gửi thông báo email, nó sẽ gửi yêu cầu đến Email System bằng giao thức SMTP.

  - Nếu có giao dịch thanh toán, nó sẽ ủy quyền cho Payment Gateway thông qua HTTP, và người dùng sẽ tương tác trực tiếp với cổng thanh toán này.

  - Payment Gateway và Email System là các hệ thống bên ngoài, được sử dụng như các dịch vụ hỗ trợ cho hệ thống My Meetings.

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



Giải thích: sơ đồ Component giúp chúng ta hiểu rõ hơn về cấu trúc bên trong của container My Meetings API. Nó cho thấy các khối xây dựng cốt lõi (components), trách nhiệm của từng thành phần, và cách chúng tương tác với nhau để xử lý các yêu cầu từ giao diện người dùng (SPA) và tương tác với các hệ thống khác.

Các thành phần: container My Meetings API được chia thành năm thành phần (components) chính, tất cả đều là các thư viện .NET Libraries:

1. Registrations:

- Chịu trách nhiệm xử lý các tác vụ liên quan đến việc đăng ký người dùng mới.
- Sử dụng component User Access để kiểm tra quyền truy cập và lưu trữ/truy xuất dữ liệu từ schema Registration data trong Database.

2. User Access:

- Quản lý quyền truy cập và xác thực người dùng.

- Được sử dụng bởi Registrations, Meetings, và Administration. Lưu trữ/truy xuất dữ liệu từ schema User Access data trong Database.

3. Meetings:

- Chịu trách nhiệm quản lý các cuộc họp, bao gồm tạo, sửa, xóa và các chức năng liên quan.

- Sử dụng User Access để kiểm tra quyền. Lưu trữ/truy xuất dữ liệu từ schema Meetings data trong Database.

4. Administration:

- Cung cấp các chức năng quản trị hệ thống, cho phép quản lý toàn bộ hệ thống.

- Sử dụng User Access để kiểm tra quyền. Lưu trữ/truy xuất dữ liệu từ schema Administration data trong Database.

5. Payments:

- Xử lý logic liên quan đến các giao dịch thanh toán.

- Lưu trữ/truy xuất dữ liệu từ schema Payments data trong Database.

Tương tác giữa các thành phần:

- Event Bus [In memory]:

- Một hệ thống truyền tin nội bộ (in-memory) cho phép các thành phần gửi và nhận các sự kiện.

- Các component Registrations, User Access, Meetings, Administration, và Payments đều có thể xuất bản (publishes event to) hoặc đăng ký (subscribes) các sự kiện trên Event Bus. Điều này cho phép các thành phần hoạt động một cách độc lập và giảm sự phụ thuộc trực tiếp.

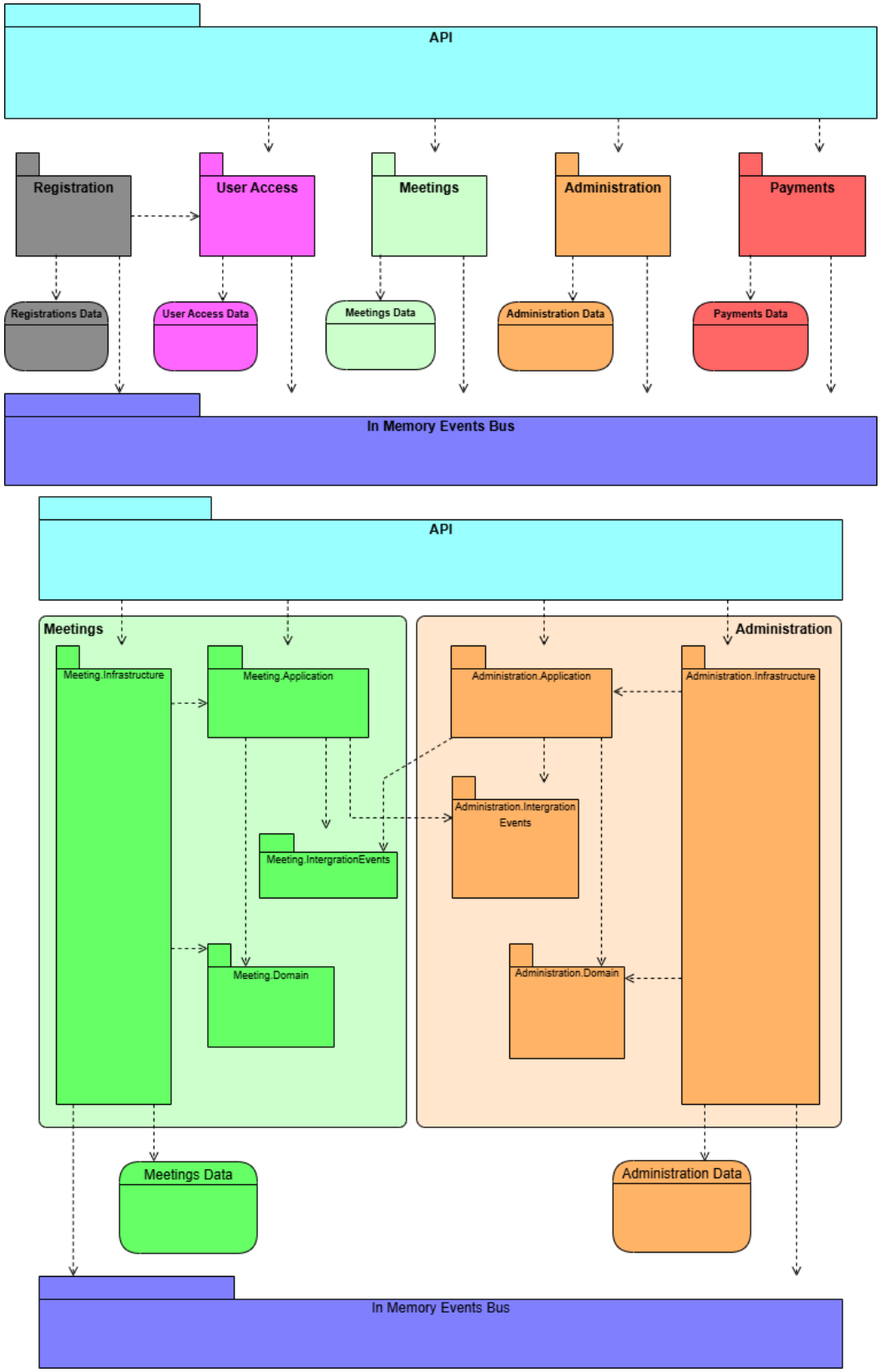
Tương tác với các Container khác và cơ sở dữ liệu:

- SPA (Single-Page Application): Tất cả các thành phần trong My Meetings API được sử dụng bởi API, vốn là điểm truy cập chính cho SPA.

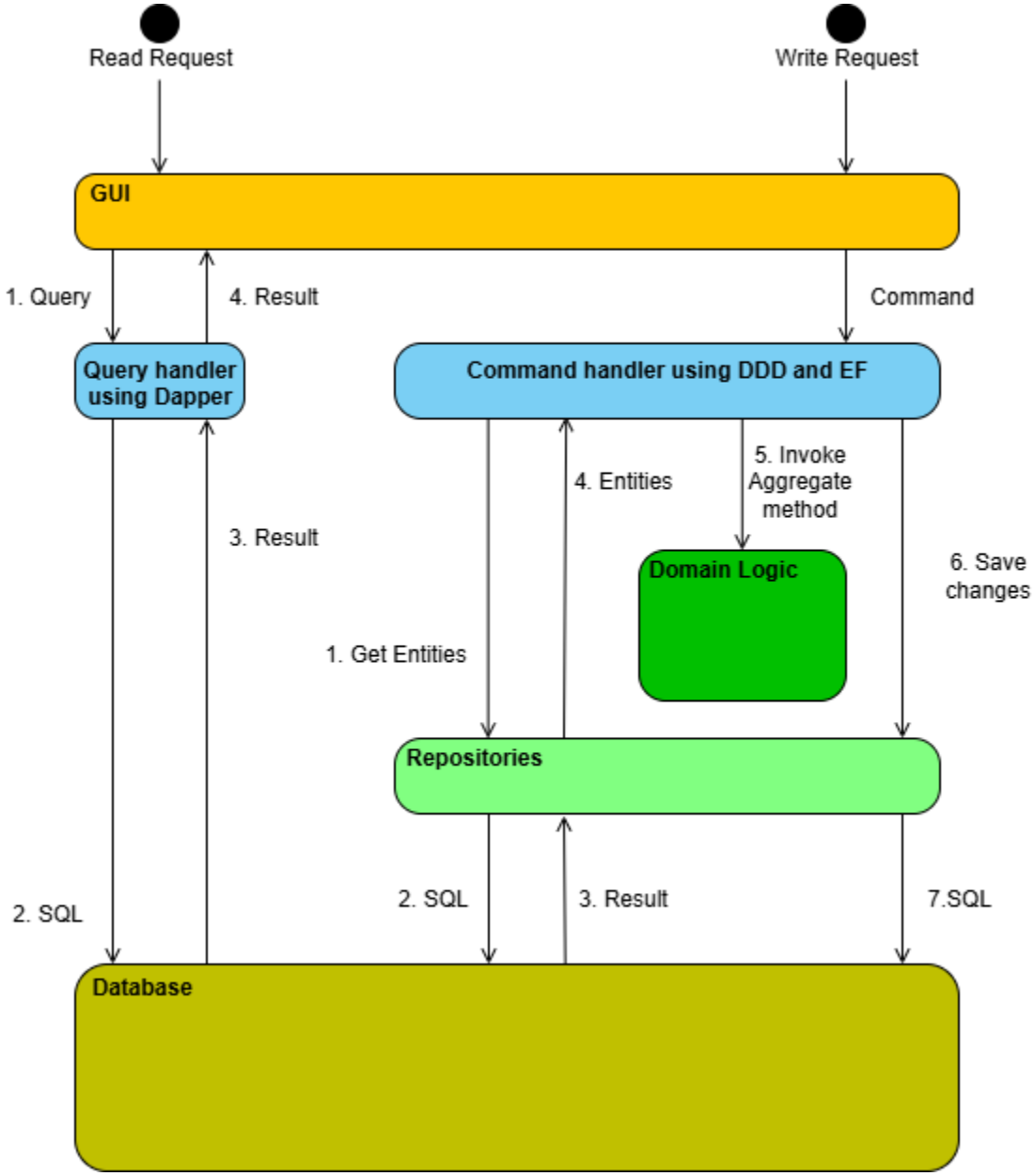
- Database: Mỗi thành phần trong API đều có một schema riêng biệt trong Database để lưu trữ dữ liệu chuyên biệt của nó (Registration data, User Access data, Meetings

data, Administration data, Payments data). Mối liên kết này giúp làm rõ cách dữ liệu được tổ chức và quản lý.

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



Bài 8. Vẽ sơ đồ xử lý 1 request:



Bài 9. 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:

- o Người đọc: Tìm kiếm sách, mượn sách online, gia hạn, xem lịch sử mượn.
- o Thủ thư/Admin: Quản lý kho sách, xử lý yêu cầu mượn/trả, theo dõi phí trễ hạn.
- o 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:**

- o Vẽ sơ đồ Context: Thư viện online, Người dùng, Thủ thư, Hệ thống thanh toán.
- o Vẽ sơ đồ Container: Web App, Mobile App, Database, Payment Service.
- o Vẽ sơ đồ Component: Bên trong Web App, có module tìm kiếm, module mượn/trả, module quản lý tài khoản.

o 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...).

9.1. Các quyết định thiết kế chính:

1. Kiến trúc frontend

- Quyết định: Sử dụng Single Page Application (SPA) (React, Angular, Vue).
- Lý do:
  - Trải nghiệm người dùng mượt (không reload trang).
  - Phù hợp khi hệ thống nhiều thao tác AJAX (search sách, lọc, giỏ mượn/trả…).
  - Giảm tải server, vì logic render chuyển sang client.
- Trade-offs: Tăng độ phức tạp frontend, cần API ổn định.

2. Tách Mobile và Web

- Quyết định: Tách mobile app (React Native/Flutter) và web app.
- Lý do:
  - Độc giả (reader) có nhu cầu truy cập mọi lúc, mọi nơi → mobile.
  - Quản trị viên (librarian/admin) cần giao diện quản lý chuyên sâu → web app.
  - Trải nghiệm UX khác nhau: mobile thiên về thao tác nhanh, web thiên về thao tác phức tạp.
- Trade-offs: Tốn thêm effort phát triển và bảo trì 2 client.

3. API Layer

- Quyết định: Dùng REST API (hoặc GraphQL nếu muốn).
- Lý do:
  - Dễ tích hợp với web, mobile, hệ thống ngoài (payment, email).
  - REST phù hợp vì tài nguyên (books, loans, users) rất tự nhiên.
- Trade-offs: Phải versioning API khi thay đổi.

4. Kiến trúc Backend

- Quyết định: Dùng Clean Architecture (Application, Domain, Infrastructure, Integration Events).
- Lý do:
  - Giữ domain tách biệt → dễ test, dễ bảo trì.
  - Cho phép thay đổi database/adapter mà không ảnh hưởng domain logic.
- Trade-offs: Tốn effort setup ban đầu, hơi "nặng" với dự án nhỏ.

5. Database

- Quyết định: Dùng Relational Database (PostgreSQL/MySQL).
- Lý do:
  - Dữ liệu library (books, loans, users) có cấu trúc rõ ràng, nhiều quan hệ.
  - Cần hỗ trợ transaction mạnh (mượn sách, trả sách).
- Trade-offs: Ít linh hoạt hơn NoSQL khi cần scale horizontal.

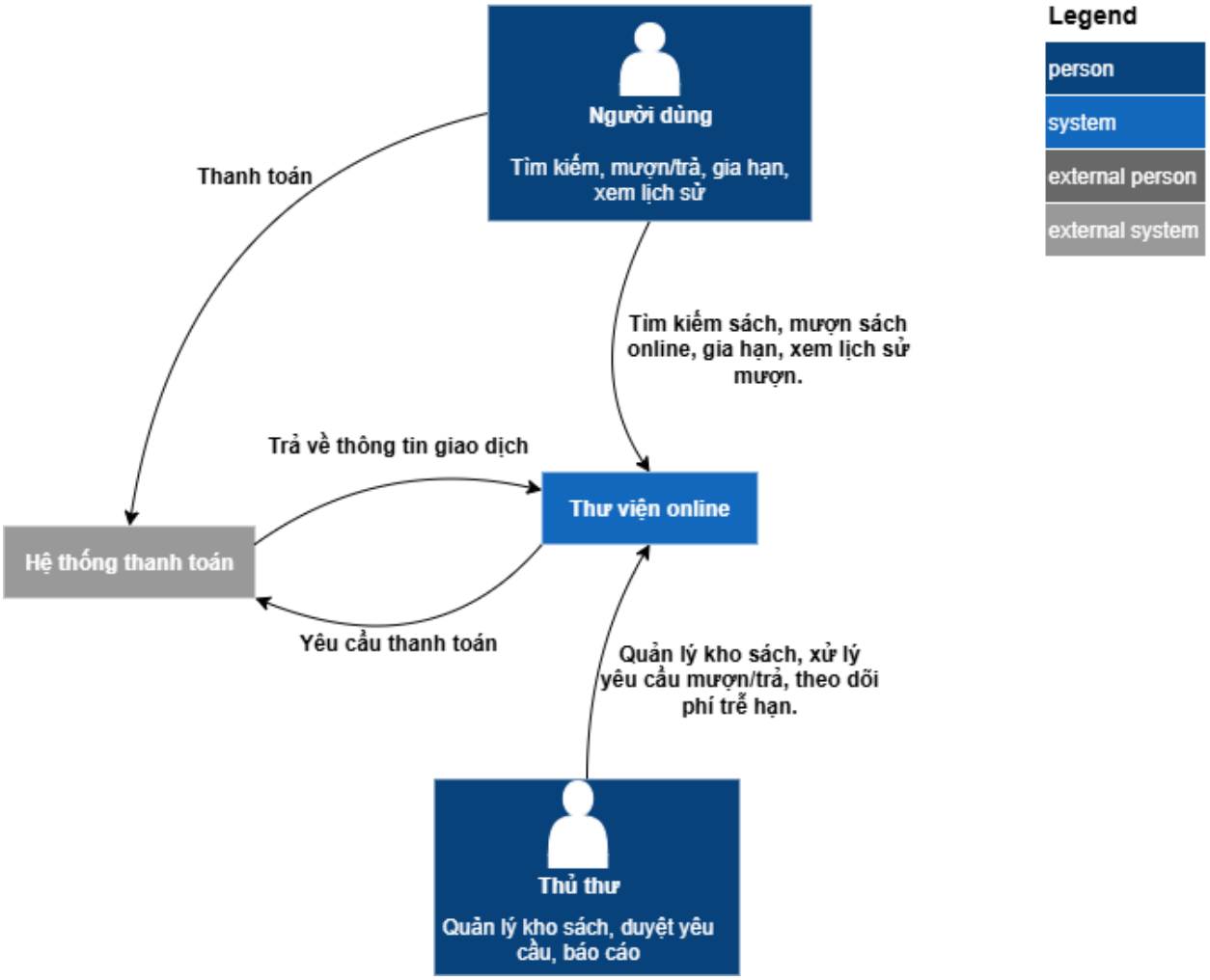
6. Event-driven Integration

- Quyết định: Dùng Integration Events qua message broker (Kafka/RabbitMQ) để tách Notifications/Reports khỏi core system.
- Lý do:
  - Giảm coupling (mượn sách xong mới publish event, notification/analytics xử lý async).
  - Hệ thống chính không bị block bởi service phụ.
- Trade-offs: Tăng độ phức tạp triển khai, cần thêm message broker.

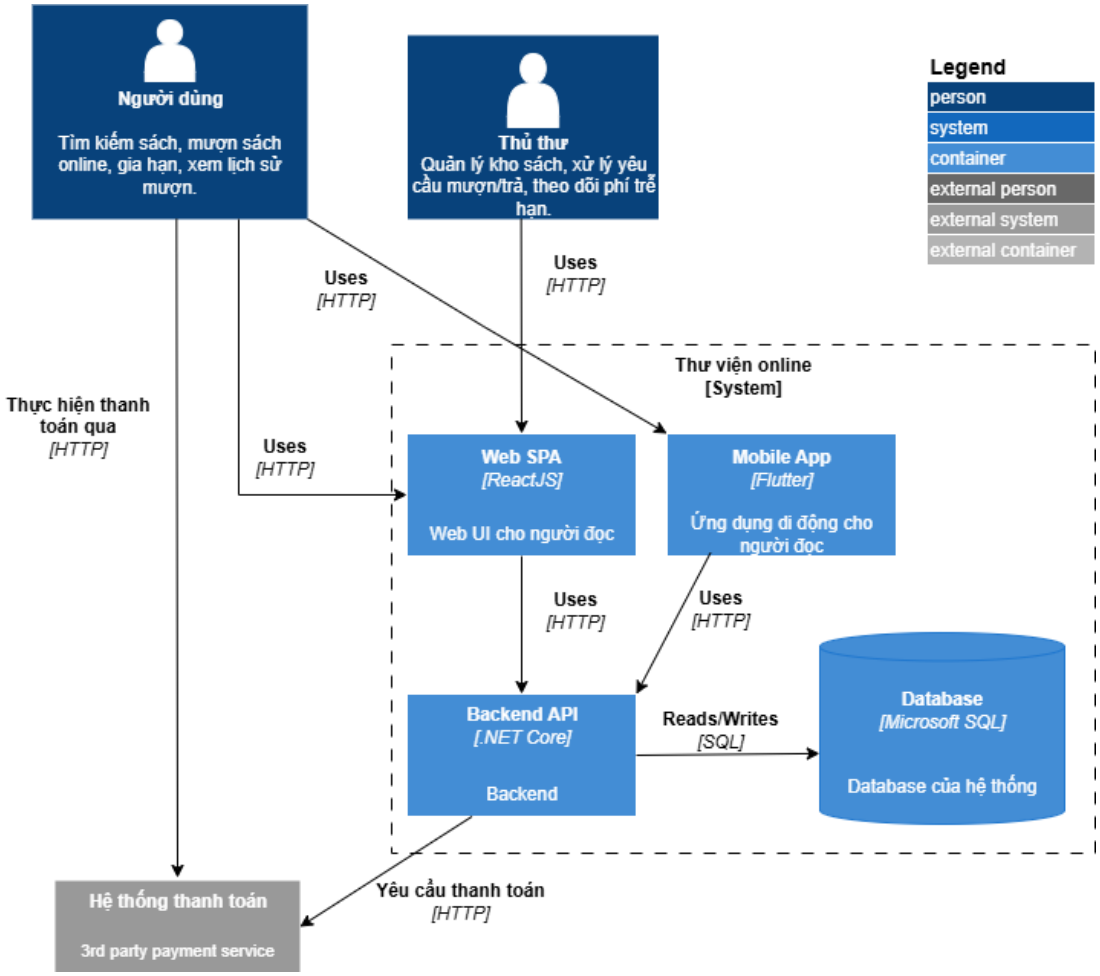
7. Triển khai

- Quyết định: Triển khai trên Cloud (AWS/Azure/GCP), dùng container (Docker/K8s).
- Lý do:
  - Dễ scale khi lượng user tăng.
  - Tận dụng service cloud (RDS, email, monitoring).
- Trade-offs: Phụ thuộc cloud vendor, chi phí cao hơn self-host.

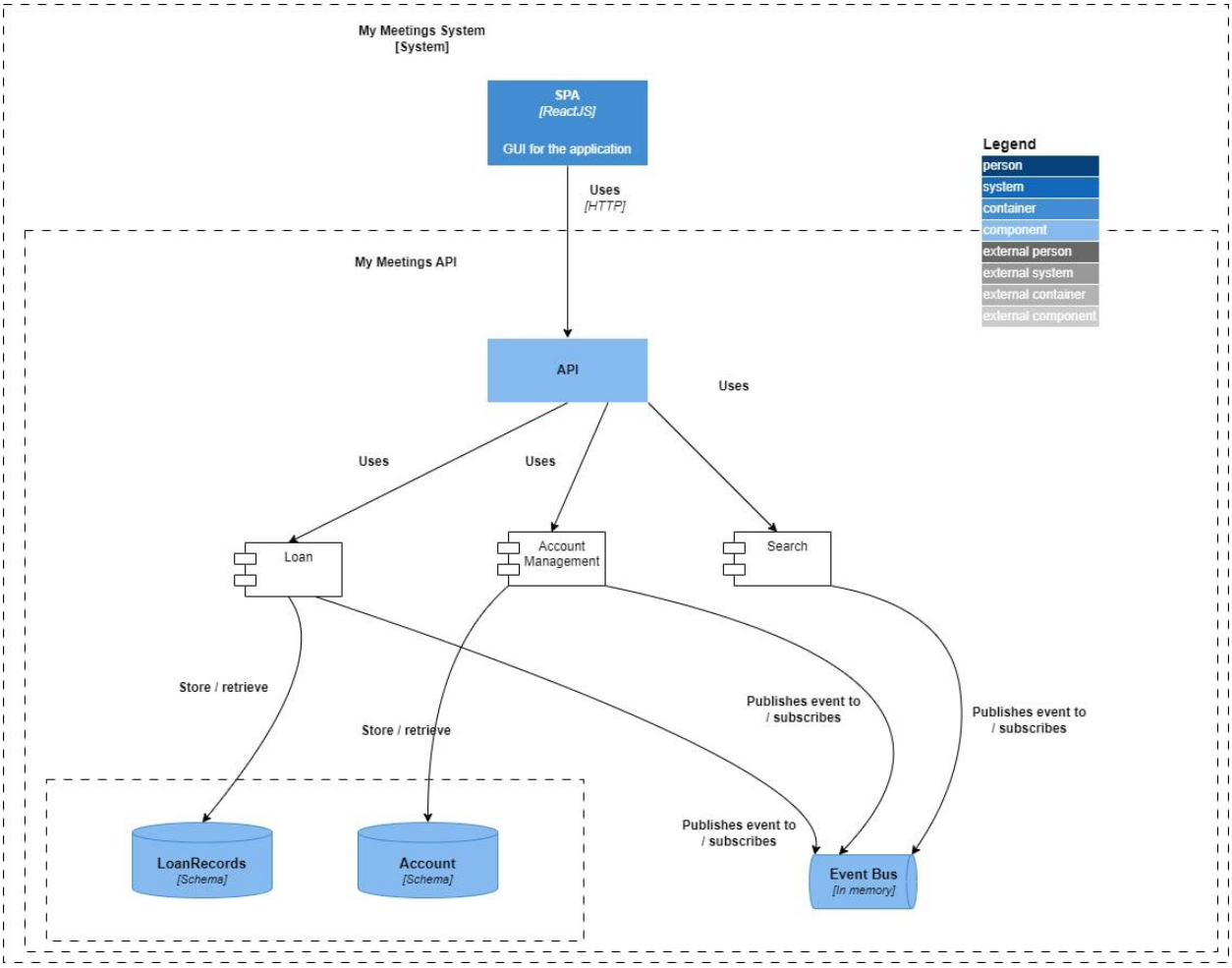
9.2. Level 1 (Context Diagram): Hệ thống và các tác nhân bên ngoài



9.3. Level 2 (Container Diagram): Các container (ứng dụng web, mobile app, database, service...)



9.4. Level 3 (Component Diagram): Các thành phần trong một container quan trọng



Container My Meetings API được chia thành ba thành phần (components) chính:

1. Loan:

- Chịu trách nhiệm xử lý các tác vụ liên quan đến dữ liệu vay.
- Lưu trữ/truy xuất dữ liệu từ schema Loan data trong Database.

2. Account Management:

- Quản lý dữ liệu tài khoản và các chức năng liên quan.
- Lưu trữ/truy xuất dữ liệu từ schema Account Management data trong Database.

3. Search:

- Cung cấp chức năng tìm kiếm trong hệ thống.
- Không trực tiếp lưu trữ dữ liệu, nhưng sử dụng dữ liệu từ các thành phần khác.

Tương tác giữa các thành phần:

Event Bus [In memory]:

- Một hệ thống truyền tin nội bộ (in-memory) cho phép các thành phần gửi và nhận các sự kiện.
- Các component Loan, Account Management, và Search đều có thể xuất bản (publishes event to) hoặc đăng ký (subscribes) các sự kiện trên Event Bus. Điều này cho phép các thành phần hoạt động một cách độc lập và giảm sự phụ thuộc trực tiếp.

Tương tác với các Container khác và cơ sở dữ liệu:

- SPA (Single-Page Application): Tất cả các thành phần trong My Meetings API được sử dụng bởi API, vốn là điểm truy cập chính cho SPA thông qua giao thức HTTP.
- Database: Mỗi thành phần trong API đều có một schema riêng biệt trong Database để lưu trữ dữ liệu chuyên biệt của nó (Loan data, Account Management data). Mối liên kết này giúp làm rõ cách dữ liệu được tổ chức và quản lý.

9.5. Level 4 (Code/Implementation Diagram): Chi tiết lớp hoặc cấu trúc code.