TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

THIẾT KẾ KIẾN TRÚC PHẦN MỀM

NGÀNH: CÔNG NGHỆ THÔNG TIN

Đề Tài:

HỆ THỐNG ĐẶT VÉ XE ONLINE

SVTH: Trần Minh Phúc - 2274802010694

Châu Gia Kiệt - 2274802010449

Võ Đình Ngọc Bình – 2274802010066

LHP: 243_71ITSE41803_0101

GVHD: ThS. Nguyễn Thế Quang

TP. Hồ Chí Minh – 7/2025

MỤC LỤC

DANH MỤC HÌNH ẢNH

CÂU 1: TÌM TẤT CẢ ENTITY CHO HỆ THỐNG BÁN VÉ XE ONLINE

- **User role:** Khách Hàng, Admin, Staff nhà xe

- **System:**

  - **External:** Futa System, Vé Xe Rẻ System, An Vui System, Payment Gateway, Notification Services.
  - **Internal:** Mobile App, Backend System, Database, Search Services, Cache Layer, Intergration Services.

1.1 Khách Hàng

Entity Name: Khách hàng (End User)	**Entity ID:** UC_USER_001
Description: Khách hàng là người dùng cuối sử dụng hệ thống để tìm kiếm và đặt vé xe trực tuyến. Họ có thể là người dùng thông thường, khách hàng thường xuyên, hoặc khách hàng lần đầu sử dụng dịch vụ. Khách hàng tương tác với hệ thống thông qua mobile app (Flutter) để thực hiện các hoạt động mua vé.	
Provides Assumptions: - Khách hàng cung cấp thông tin tìm kiếm (nơi đi, nơi đến, ngày giờ) - Khách hàng cung cấp thông tin cá nhân để đặt vé - Khách hàng cung cấp thông tin thanh toán - Khách hàng gửi requests tìm kiếm địa điểm, tìm kiếm vé - Khách hàng gửi events đặt vé, hủy vé	
Requires Assumptions: - Khách hàng yêu cầu hệ thống trả về kết quả tìm kiếm nhanh chóng (< 5s) - Khách hàng yêu cầu interface thân thiện, dễ sử dụng - Khách hàng yêu cầu thông tin giá vé chính xác, cập nhật - Khách hàng yêu cầu xác nhận đặt vé qua email/SMS - Khách hàng yêu cầu lịch sử đặt vé và quản lý đơn hàng	
Identified Use Cases:	

UC1.1: Tìm kiếm địa điểm UC2.1: Tìm kiếm vé một chiều UC2.2: Tìm kiếm vé khứ hồi UC3.1: Xem lịch giá rẻ UC4.1: Đăng ký tài khoản UC4.2: Đăng nhập UC4.3: Đặt vé UC4.4: Thanh toán UC4.5: Quản lý đơn hàng UC4.7: Quản lý profile

1.2 Admin

Entity Name: Admin hệ thống (System Administrator)	**Entity ID:** UC_ADMIN_001
Description: Admin hệ thống là người quản lý và vận hành hệ thống bán vé xe online. Họ có quyền truy cập toàn bộ hệ thống, giám sát hoạt động 24/7, xử lý sự cố và đảm bảo hệ thống hoạt động ổn định. Admin tương tác với hệ thống thông qua dashboard quản trị.	
Provides Assumptions: - Admin cung cấp cấu hình hệ thống - Admin cung cấp thông tin nhà xe đối tác - Admin cung cấp business rules và policies - Admin gửi commands để khởi động/dừng services - Admin gửi sự kiện cảnh báo, thông báo	
Requires Assumptions: - Admin yêu cầu dashboard monitoring real-time - Admin yêu cầu logs chi tiết về hoạt động hệ thống - Admin yêu cầu báo cáo thống kê performance - Admin yêu cầu tools để troubleshoot issues	

- Admin yêu cầu hệ thống backup và recovery
Identified Use Cases: UC5.1: Monitor hệ thống UC5.2: Quản lý nhà xe đối tác UC5.3: Quản lý cấu hình UC5.4: Báo cáo thống kê UC5.5: Xử lý khiếu nại UC5.6: Quản lý người dùng

1.3 Hệ Thống Futa

Entity Name: Hệ thống Futa (Futa Booking System)	**Entity ID:** UC_FUTA_001
Description: Hệ thống bán vé của nhà xe Futa là external system cung cấp API để truy vấn thông tin vé, địa điểm, giá cả. Hệ thống có response time < 3s nhưng không hỗ trợ pagination. Đây là partner system cần được tích hợp để cung cấp dữ liệu vé cho USERs.	
Provides Assumptions: - Futa system cung cấp API endpoints để query vé - Futa system cung cấp thông tin địa điểm, tuyến đường - Futa system cung cấp real-time availability - Futa system gửi response trong < 3s - Futa system gửi data format JSON	
Requires Assumptions: - Futa system yêu cầu authentication để access API - Futa system yêu cầu rate limiting compliance - Futa system yêu cầu proper error handling - Futa system yêu cầu timeout handling (< 3s) - Futa system yêu cầu retry mechanism for failures	
Identified Use Cases: UC1.2: Lấy danh sách địa điểm từ Futa	

UC2.3: Truy vấn vé từ Futa UC2.7: Xử lý lỗi từ external systems

1.4 Hệ Thống Vé Xe Rẻ

Entity Name: Hệ thống Vé Xe Rẻ (VeXeRe System)	**Entity ID:** UC_VXR_001
Description: Hệ thống bán vé của Vé Xe Rẻ là external system có khối lượng vé lớn nhất, hỗ trợ pagination, response time < 5s. Đây là partner quan trọng với nhiều dữ liệu và cần special handling cho pagination.	
Provides Assumptions: - Hệ thống VXR cung cấp API phản hồi dạng phân trang - Hệ thống VXR cung cấp bộ dữ liệu lớn với nhiều vé - Hệ thống VXR cung cấp webhook để cập nhật thời gian thực - Hệ thống VXR gửi phản hồi trong < 5 giây - Hệ thống VXR gửi thông tin phân trang	
Requires Assumptions: - Hệ thống VXR yêu cầu xử lý tham số phân trang - Hệ thống VXR yêu cầu thiết lập endpoint webhook - Hệ thống VXR yêu cầu cấu hình kích thước trang phù hợp - Hệ thống VXR yêu cầu xử lý các request đồng thời - Hệ thống VXR yêu cầu đồng bộ hóa dữ liệu	
Identified Use Cases: UC1.3: Lấy danh sách địa điểm từ Vé Xe Rẻ UC2.4: Truy vấn vé từ Vé Xe Rẻ (với pagination) UC2.7: Xử lý lỗi từ external systems	

1.5 Hệ Thống An Vui

Entity Name: Hệ thống An Vui (An Vui Booking System)	**Entity ID:** UC_AV_001

Description: Hệ thống bán vé của nhà xe An Vui là external system có response time chậm nhất (< 15s), không hỗ trợ pagination. Cần special handling để không làm chậm overall system performance.
Provides Assumptions: - Hệ thống An Vui cung cấp các API endpoint (chậm) - Hệ thống An Vui cung cấp toàn bộ dữ liệu trong mỗi request - Hệ thống An Vui cung cấp thông tin địa điểm qua file - Hệ thống An Vui gửi phản hồi trong vòng < 15 giây - Hệ thống An Vui gửi dữ liệu lớn mà không sử dụng phân trang
Requires Assumptions: - An Vui system yêu cầu extended timeout handling - An Vui system yêu cầu async processing - An Vui system yêu cầu caching strategy - An Vui system yêu cầu fallback mechanism - An Vui system yêu cầu pre-loading popular routes
Identified Use Cases: UC1.4: Lấy danh sách địa điểm từ An Vui UC2.5: Truy vấn vé từ An Vui UC2.7: Xử lý lỗi từ external systems

1.6 Cổng Thanh Toán

Entity Name: Payment Gateway (Cổng thanh toán)	**Entity ID:** UC_PAYMENT_001
Description: Payment Gateway là hệ thống bên thứ 3 xử lý thanh toán cho các đơn hàng đặt vé. Hỗ trợ nhiều phương thức thanh toán như thẻ tín dụng, ví điện tử, chuyển khoản. Đây là critical component cho business operation.	
Provides Assumptions: - Payment Gateway cung cấp secure payment processing - Payment Gateway cung cấp multiple payment methods - Payment Gateway cung cấp transaction status updates	

- Payment Gateway gửi confirmation/failure notifications - Payment Gateway gửi transaction receipts
Requires Assumptions: - Payment Gateway yêu cầu secure API integration - Payment Gateway yêu cầu PCI compliance - Payment Gateway yêu cầu transaction logging - Payment Gateway yêu cầu error handling - Payment Gateway yêu cầu refund capabilities
Identified Use Cases: UC4.4: Thanh toán UC4.5: Quản lý đơn hàng (refund)

1.7 Mobile App

Entity Name: Mobile Application (Flutter App)	**Entity ID:** UC_MOBILE_001
Description: Mobile Application là frontend client được phát triển bằng Flutter, cung cấp giao diện người dùng cho USERs. Đây là primary interface cho tất cả user interactions và cần đảm bảo performance, usability tốt trên cả Android và iOS.	
Provides Assumptions: - Mobile app cung cấp user-friendly interface - Mobile app cung cấp real-time search capabilities - Mobile app cung cấp offline data caching - Mobile app gửi user interactions tới backend - Mobile app gửi analytics events	
Requires Assumptions: - Mobile app yêu cầu fast API responses từ backend - Mobile app yêu cầu reliable internet connectivity - Mobile app yêu cầu push notification setup - Mobile app yêu cầu secure authentication - Mobile app yêu cầu app store compliance	

Identified Use Cases: UC1.1: Tìm kiếm địa điểm UC2.1: Tìm kiếm vé UC2.2: Tìm kiếm vé khứ hồi UC3.1: Hiển thị lịch giá UC4.1- 4.7: Tất cả user functions

1.8 TicketSystem

Entity Name: Backend System (Java Spring Backend)	**Entity ID:** UC_BACKEND_001
Description: Backend System là core business logic layer được phát triển bằng Java Spring Framework. Xử lý tất cả business rules, API orchestration, data processing. Đây là heart của hệ thống, kết nối mobile app với databases và external systems.	
Provides Assumptions: - Backend cung cấp REST API endpoints - Backend cung cấp business logic processing - Backend cung cấp data validation và transformation - Backend cung cấp integration với external systems - Backend gửi processed data tới mobile app	
Requires Assumptions: - Backend yêu cầu database connectivity - Backend yêu cầu external API integration - Backend yêu cầu caching layer - Backend yêu cầu monitoring và logging - Backend yêu cầu scalability support	
Identified Use Cases: UC1.5: Đồng bộ dữ liệu địa điểm UC2.6: Tổng hợp kết quả tìm kiếm UC3.2: Cập nhật giá theo lịch trình UC4.3: Đặt vé	

UC4.4: Thanh toán

1.9 Tìm Kiếm

Entity Name: Search Service (Elasticsearch)	**Entity ID:** UC_SEARCH_001
Description: Search Service là specialized service cho location search với requirements về fast response time (< 1.5s). Sử dụng Elasticsearch để cung cấp fuzzy search, auto-complete, real-time suggestions cho địa điểm.	
Provides Assumptions: - Search service cung cấp fast search responses - Search service cung cấp fuzzy matching capabilities - Search service cung cấp auto-complete suggestions - Search service cung cấp indexed location data - Search service gửi search results tới backend	
Requires Assumptions: - Search service yêu cầu data indexing - Search service yêu cầu real-time updates - Search service yêu cầu performance optimization - Search service yêu cầu scaling capabilities - Search service yêu cầu monitoring	
Identified Use Cases: UC1.1: Tìm kiếm địa điểm theo từ khóa UC1.6: Cập nhật địa điểm real-time	

1.10 Cache Layer

Entity Name: Cache Layer (Redis Cache)	**Entity ID:** UC_CACHE_001

Description: Cache Layer sử dụng Redis để cache frequently accessed data như pricing calendar, search results, user sessions. Đây là performance optimization layer để giảm database load và improve response time.
Provides Assumptions: - Cache layer cung cấp fast data retrieval - Cache layer cung cấp session management - Cache layer cung cấp temporary data storage - Cache layer cung cấp pricing data cache - Cache layer gửi cached data tới backend
Requires Assumptions: - Cache layer yêu cầu TTL configuration - Cache layer yêu cầu cache invalidation strategy - Cache layer yêu cầu memory management - Cache layer yêu cầu high availability - Cache layer yêu cầu monitoring
Identified Use Cases: UC3.1: Hiển thị lịch giá rẻ nhất UC3.3: Cập nhật giá khi user search UC3.4: Tính toán giá rẻ nhất

CÂU 2: TÌM TẤT CẢ CÁC FUNCTIONAL CHO HỆ THỐNG BÁN VÉ XE ONLINE

2.1 Địa điểm (6 Use Case)

UC1.1: Tìm kiếm địa điểm theo từ khóa

UC1.2: Lấy danh sách địa điểm từ Futa

UC1.3: Lấy danh sách địa điểm từ Vé Xe Rẻ

UC1.4: Lấy danh sách địa điểm từ An Vui

UC1.5: Đồng bộ dữ liệu địa điểm

UC1.6: Cập nhật địa điểm real-time

Use Case Title: Tìm kiếm địa điểm theo từ khóa	**Use Case ID:** UC1.1
General Use Case Description: Khách hàng nhập từ khóa để tìm kiếm địa điểm (nơi đi hoặc nơi đến). Hệ thống sẽ gợi ý danh sách các địa điểm phù hợp một cách nhanh chóng và chính xác từ dữ liệu của các nhà xe đối tác.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_SEARCH_001, UC_CACHE_001	
Preconditions: – Khách hàng đã mở mobile app – Search service đang hoạt động – Dữ liệu địa điểm đã được đồng bộ từ các nhà xe – Cache layer sẵn sàng	
Primary Use Case Flow of Events:	
1.	Khách hàng nhập từ khóa tìm kiếm vào ô địa điểm
2.	Mobile app gửi request tới Backend System

3.	Backend kiểm tra Cache layer cho kết quả
4.	Cache hit: trả về kết quả từ cache
5.	Cache miss: gửi request tới Search Service
6.	Search Service thực hiện fuzzy search
7.	Search Service trả về danh sách địa điểm
8.	Backend cache kết quả và trả về Mobile app
9.	Mobile app hiển thị danh sách gợi ý
10.	Khách hàng chọn địa điểm mong muốn
Primary Use Case Post Conditions: - Danh sách địa điểm hiển thị trong < 1.5s - Kết quả được cache cho lần sau - Địa điểm đã chọn lưu vào session	
Alternate Use Case #1 Flow of Events: Search Service không phản hồi	
1.	Thực hiện bước 1-5 như primary flow
2.	Search Service timeout hoặc lỗi
3.	Backend fallback tới database query
4.	Thực hiện SQL LIKE query
5.	Trả về kết quả từ database
Alternate Use Case #1 Post Conditions: – Kết quả vẫn được trả về (chậm hơn) – Lỗi được ghi log – Functionality vẫn hoạt động	

Use Case Title: Lấy danh sách địa điểm từ Futa	**Use Case ID:** UC1.2
General Use Case Description: Hệ thống tự động lấy danh sách địa điểm từ hệ thống Futa thông qua API hoặc file JSON để cập nhật database địa điểm phục vụ tìm kiếm.	

Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_DB_001, UC_STAFF_001	
Preconditions: – API Futa đang hoạt động – Integration service đã được cấu hình – Database sẵn sàng nhận dữ liệu – Authentication với Futa thành công	
Primary Use Case Flow of Events:	
1.	**Scheduler trigger hoặc manual request**
2.	Integration Service gọi Futa API
3.	Futa System authenticate request
4.	Futa System trả về JSON locations
5.	Integration Service validate dữ liệu
6.	Transform dữ liệu về format chuẩn
7.	Backend System xử lý dữ liệu
8.	Database System cập nhật location table
9.	Search Service re-index dữ liệu mới
10.	Ghi log thành công
Primary Use Case Post Conditions: – Dữ liệu Futa được cập nhật trong database – Search index được refresh – Khách hàng thấy địa điểm mới từ Futa	
Alternate Use Case #1 Flow of Events: Futa API không phản hồi	
1.	**Thực hiện bước 1-2 như primary flow**
2.	Futa API timeout (> 3s)
3.	Integration Service ghi log lỗi
4.	Retry mechanism schedule lại

5.	Thông báo admin về lỗi
Alternate Use Case #1 Post Conditions: – Dữ liệu Futa không được cập nhật – Retry sẽ thực hiện sau – Admin được thông báo	

Use Case Title: Lấy danh sách địa điểm từ Vé Xe Rẻ	**Use Case ID:** UC1.3
General Use Case Description: Hệ thống lấy danh sách địa điểm từ Vé Xe Rẻ thông qua API hoặc webhook notification khi có cập nhật địa điểm mới.	
Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_VXR_001, UC_DB_001, UC_STAFF_001	
Preconditions: – API Vé Xe Rẻ đang hoạt động – Webhook endpoint đã được setup – Database sẵn sàng nhận dữ liệu – Authentication với VXR thành công	
Primary Use Case Flow of Events:	
1.	**Webhook trigger hoặc scheduled call**
2.	Integration Service gọi VXR API
3.	VXR System authenticate request
4.	VXR System trả về locations data
5.	Integration Service validate dữ liệu
6.	Transform dữ liệu về format chuẩn
7.	Backend System xử lý dữ liệu
8.	Database System cập nhật location table
9.	Search Service re-index dữ liệu mới
10.	Ghi log thành công

Primary Use Case Post Conditions:

- Dữ liệu VXR được cập nhật trong database
- Search index được refresh
- Khách hàng thấy địa điểm mới từ VXR

Alternate Use Case #1 Flow of Events:

VXR API trả về lỗi

1.	**Thực hiện bước 1-4 như primary flow**
2.	VXR API trả về error response
3.	Integration Service parse error
4.	Ghi log chi tiết về lỗi
5.	Schedule retry cho lần sau

Alternate Use Case #1 Post Conditions:

- Dữ liệu An Vui không được cập nhật
- Cached data được sử dụng
- Retry được schedule

Use Case Title: Lấy danh sách địa điểm từ An Vui	**Use Case ID:** UC1.4

General Use Case Description:

Hệ thống lấy danh sách địa điểm từ An Vui thông qua API hoặc cron job crawling dữ liệu, xử lý đặc biệt vì response time chậm.

Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_AV_001, UC_DB_001, UC_STAFF_001

Preconditions:

- API An Vui đang hoạt động
- Cron job đã được setup
- Database sẵn sàng nhận dữ liệu
- Extended timeout được cấu hình

Primary Use Case Flow of Events:

1.	**Cron job trigger hoặc manual request**
2.	Integration Service gọi An Vui API
3.	An Vui System authenticate request
4.	An Vui System trả về locations (chậm)
5.	Integration Service validate dữ liệu
6.	Transform dữ liệu về format chuẩn
7.	Backend System xử lý dữ liệu
8.	Database System cập nhật location table
9.	Search Service re-index dữ liệu mới
10.	Ghi log thành công
Primary Use Case Post Conditions: – Dữ liệu An Vui được cập nhật trong database – Search index được refresh – Khách hàng thấy địa điểm mới từ An Vui	
Alternate Use Case #1 Flow of Events: An Vui API timeout (> 15s)	
1.	**Thực hiện bước 1-2 như primary flow**
2.	An Vui API timeout sau 15s
3.	Integration Service ghi log timeout
4.	Fallback tới cached data nếu có
5.	Retry với exponential backoff
Alternate Use Case #1 Post Conditions: – Dữ liệu VXR không được cập nhật – Error được log chi tiết – Retry mechanism hoạt động	

Use Case Title: Đồng bộ dữ liệu địa điểm	**Use Case ID:** UC1.5

General Use Case Description: Hệ thống tự động đồng bộ dữ liệu địa điểm từ 3 nhà xe đối tác, merge và normalize dữ liệu để tạo database thống nhất phục vụ tìm kiếm.	
Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_DB_001, UC_SEARCH_001	
Preconditions: – Tất cả 3 API nhà xe đang hoạt động – Integration service đã được cấu hình – Database sẵn sàng nhận dữ liệu – Search service đang hoạt động	
Primary Use Case Flow of Events:	
1.	**Scheduler khởi động sync process**
2.	Integration Service gọi parallel tới 3 API
3.	Futa trả về dữ liệu (< 3s)
4.	VXR trả về dữ liệu (< 5s)
5.	An Vui trả về dữ liệu (< 15s)
6.	Integration Service merge dữ liệu
7.	Normalize và deduplicate
8.	Backend validate business rules
9.	Database cập nhật location table
10.	Search Service re-index toàn bộ
Primary Use Case Post Conditions: – Database có dữ liệu đồng bộ từ 3 nguồn – Search index được cập nhật – Khách hàng thấy thông tin tức thời	
Alternate Use Case #1 Flow of Events: Một hoặc nhiều API không phản hồi	
1.	**Thực hiện bước 1-5 như primary flow**
2.	An Vui timeout, Futa và VXR OK
3.	Integration Service merge 2/3 nguồn

4.	Đánh dấu An Vui data là stale
5.	Cập nhật partial data
Alternate Use Case #1 Post Conditions: – Hệ thống hoạt động với 2/3 nguồn dữ liệu – Partial data được cập nhật – Retry mechanism hoạt động	

Use Case Title: Đồng bộ dữ liệu địa điểm	**Use Case ID:** UC1.6
General Use Case Description: Hệ thống nhận và xử lý cập nhật địa điểm real-time từ các nhà xe để đảm bảo khách hàng thấy thông tin tức thời khi có thay đổi.	
Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_DB_001, UC_SEARCH_001, UC_CACHE_001	
Preconditions: – Webhook endpoints đã được setup – Real-time connection đang hoạt động – Database sẵn sàng nhận updates – Cache layer đã sẵn sàng	
Primary Use Case Flow of Events:	
1.	**Nhà xe gửi update notification**
2.	Integration Service nhận webhook
3.	Validate webhook signature
4.	Parse update data
5.	Transform về format chuẩn
6.	Backend System xử lý update
7.	Database cập nhật record tương ứng
8.	Search Service update index

9.	Cache layer invalidate related data
10.	Khách hàng thấy thông tin mới
Primary Use Case Post Conditions: – Dữ liệu được cập nhật tức thời – Search index được sync – Cache được invalidate – Khách hàng thấy thông tin real-tim	
Alternate Use Case #1 Flow of Events: Webhook signature invalid	
1.	**Thực hiện bước 1-3 như primary flow**
2.	Webhook signature không hợp lệ
3.	Integration Service reject request
4.	Ghi log security violation
5.	Thông báo admin về potential attack
Alternate Use Case #1 Post Conditions: – Update được reject – Security log được ghi – Admin được thông báo	

2.2 Tìm kiếm vé (7 Use Case)

UC2.1: Tìm kiếm vé một chiều

UC2.2: Tìm kiếm vé khứ hồi

UC2.3: Truy vấn vé từ Futa

UC2.4: Truy vấn vé từ Vé Xe Rẻ

UC2.5: Truy vấn vé từ An Vui

UC2.6: Tổng hợp kết quả tìm kiếm

UC2.7: Xử lý lỗi từ external systems

Use Case Title: Tìm kiếm vé một chiều	**Use Case ID:** UC2.1
General Use Case Description: Khách hàng nhập thông tin nơi đi, nơi đến, ngày đi để tìm vé một chiều. Hệ thống truy vấn parallel tất cả nhà xe và trả về kết quả trong < 5s.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_CACHE_001	
Preconditions: – Khách hàng đã chọn địa điểm đi và đến – Ngày đi hợp lệ – Các API nhà xe đang hoạt động – Integration service sẵn sàng	
Primary Use Case Flow of Events:	
1.	**Khách hàng nhập nơi đi, nơi đến, ngày đi**
2.	Mobile app validate thông tin
3.	Gửi request tới Backend System
4.	Backend kiểm tra cache cho query tương tự
5.	Cache miss: gửi tới Integration Service
6.	Integration Service gọi parallel 3 API
7.	Futa trả về kết quả (< 3s)
8.	VXR trả về kết quả (< 5s)
9.	An Vui trả về kết quả (< 15s)
10.	Integration aggregate kết quả sẵn sàng
Primary Use Case Post Conditions: – Danh sách vé hiển thị trong < 5s – Kết quả từ tất cả nhà xe có sẵn – Dữ liệu được cache	

– Khách hàng có thể chọn vé	
Alternate Use Case #1 Flow of Events: An Vui chậm > 5s	
1.	**Thực hiện bước 1-8 như primary flow**
2.	An Vui chưa phản hồi sau 5s
3.	Integration trả về kết quả Futa + VXR
4.	Hiển thị với thông báo "Đang tải thêm..."
5.	Khi An Vui phản hồi, cập nhật thêm
Alternate Use Case #1 Post Conditions: – Khách hàng nhận kết quả nhanh – System không bị block – Trải nghiệm không bị ảnh hưởng	

Use Case Title: Tìm kiếm vé khứ hồi	**Use Case ID:** UC2.2
General Use Case Description: Khách hàng nhập thông tin nơi đi, nơi đến, ngày đi, ngày về để tìm vé khứ hồi. Hệ thống tìm kiếm song song 2 chiều và suggest combo vé.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_CACHE_001	
Preconditions: – Khách hàng đã chọn địa điểm đi và đến – Ngày đi và ngày về hợp lệ – Các API nhà xe đang hoạt động – Integration service sẵn sàng	
Primary Use Case Flow of Events:	
1.	**Khách hàng nhập nơi đi, nơi đến, ngày đi, ngày về**
2.	Mobile app validate thông tin

3.	Gửi request tới Backend System
4.	Backend tạo 2 search queries (đi và về)
5.	Integration Service gọi parallel cho 2 queries
6.	Tìm kiếm vé đi: A → B ngày X
7.	Tìm kiếm vé về: B → A ngày Y
8.	Aggregate kết quả từ 2 searches
9.	Tạo combo suggestions
10.	Hiển thị vé đi và về separately + combos

Primary Use Case Post Conditions:

- Danh sách vé đi và về được hiển thị
- Combo suggestions được tạo
- Khách hàng có thể chọn từng vé hoặc combo
- Dữ liệu được cache

Alternate Use Case #1 Flow of Events:
Một chiều không có vé

1.	**Thực hiện bước 1-8 như primary flow**
2.	Vé đi có kết quả, vé về không có
3.	Hiển thị vé đi và thông báo vé về
4.	Suggest ngày về khác
5.	Cho phép book vé đi trước

Alternate Use Case #1 Post Conditions:

- Khách hàng vẫn có option book vé đi
- Alternative suggestions được provide
- Flexibility trong booking process

Use Case Title: Truy vấn vé từ Futa	**Use Case ID:** UC2.3
General Use Case Description:	

Hệ thống gọi API Futa để truy vấn vé xe theo tuyến đường và ngày cụ thể, xử lý response trong < 3s và không có phân trang.	
Entitles Involved: UC_INTEGRATION_001, UC_FUTA_001, UC_BACKEND_001	
Preconditions: – API Futa đang hoạt động – Authentication với Futa thành công – Request parameters hợp lệ – Network connection stable	
Primary Use Case Flow of Events:	
1.	**Integration Service nhận search request**
2.	Prepare API parameters cho Futa
3.	Gọi Futa API với timeout 3s
4.	Futa System authenticate request
5.	Futa System query internal database
6.	Futa trả về complete ticket list
7.	Integration Service validate response
8.	Transform dữ liệu về common format
9.	Return tickets tới Backend System
10.	Ghi log thành công
Primary Use Case Post Conditions: – Danh sách vé Futa được trả về – Dữ liệu đã được transform – Response time < 3s – Sẵn sàng cho aggregation	
Alternate Use Case #1 Flow of Events: Futa API timeout	
1.	**Thực hiện bước 1-3 như primary flow**
2.	Futa API không phản hồi trong 3s

3.	Integration Service timeout
4.	Ghi log timeout error
5.	Return empty result với error flag
Alternate Use Case #1 Post Conditions: – Futa data không available – Error được log – System vẫn hoạt động với 2 nguồn khác	

Use Case Title: Truy vấn vé từ VeXeRe	**Use Case ID:** UC2.4
General Use Case Description: Hệ thống gọi API Vé Xe Rẻ để truy vấn vé xe, xử lý pagination vì số lượng vé lớn, response time < 5s.	
Entitles Involved: UC_INTEGRATION_001, UC_VXR_001, UC_BACKEND_001	
Preconditions: – API Vé Xe Rẻ đang hoạt động – Authentication với VXR thành công – Request parameters hợp lệ – Pagination được support	
Primary Use Case Flow of Events:	
1.	**Integration Service nhận search request**
2.	Prepare API parameters với pagination
3.	Gọi VXR API page đầu tiên
4.	VXR System authenticate request
5.	VXR System query với pagination
6.	VXR trả về page 1 + pagination info
7.	Integration Service check total pages
8.	Gọi parallel multiple pages nếu cần

9.	Merge tất cả pages thành complete list
10.	Transform và return tới Backend
Primary Use Case Post Conditions: – Danh sách vé VXR complete được trả về – Pagination được handle properly – Response time < 5s – Sẵn sàng cho aggregation	
Alternate Use Case #1 Flow of Events: VXR API trả về quá nhiều pages	
1.	**Thực hiện bước 1-7 như primary flow**
2.	VXR trả về > 10 pages
3.	Integration Service limit tới 10 pages
4.	Gọi parallel 10 pages đầu tiên
5.	Merge và return với note "More results available"
Alternate Use Case #1 Post Conditions: – Partial results được trả về – Performance không bị ảnh hưởng – User được thông báo có thêm kết quả	

Use Case Title: Truy vấn vé từ An Vui	**Use Case ID:** UC2.5
General Use Case Description: Hệ thống gọi API An Vui để truy vấn vé xe, xử lý response time chậm < 15s, không có phân trang, có thể cần pre-loading.	
Entitles Involved: UC_INTEGRATION_001, UC_AV_001, UC_BACKEND_001, UC_CACHE_001	
Preconditions: – API An Vui đang hoạt động – Extended timeout đã được cấu hình	

– Request parameters hợp lệ – Cache layer sẵn sàng cho pre-loading	
Primary Use Case Flow of Events:	
1.	**Integration Service nhận search request**
2.	Check cache cho pre-loaded data
3.	Cache miss: gọi An Vui API
4.	An Vui System authenticate request
5.	An Vui System query (chậm)
6.	An Vui trả về complete ticket list
7.	Integration Service validate response
8.	Transform dữ liệu về common format
9.	Cache kết quả cho popular routes
10.	Return tickets tới Backend System
Primary Use Case Post Conditions: – Danh sách vé An Vui được trả về – Response time < 15s – Dữ liệu được cache – Sẵn sàng cho aggregation	
Alternate Use Case #1 Flow of Events: An Vui API timeout > 15s	
1.	**Thực hiện bước 1-5 như primary flow**
2.	An Vui API không phản hồi trong 15s
3.	Integration Service timeout
4.	Check cache cho stale data
5.	Return cached data với warning
Alternate Use Case #1 Post Conditions: – Stale cached data được sử dụng – User được thông báo data có thể cũ	

– System vẫn functional

Use Case Title: Tổng hợp kết quả tìm kiếm	**Use Case ID:** UC2.6
General Use Case Description: Hệ thống tổng hợp, merge và format kết quả từ 3 nhà xe thành danh sách thống nhất, sort theo business logic, đảm bảo < 5s.	
Entitles Involved: UC_INTEGRATION_001, UC_BACKEND_001, UC_FUTA_001, UC_VXR_001, UC_AV_001	
Preconditions: – Đã nhận responses từ ít nhất 1 nhà xe – Integration service sẵn sàng xử lý – Business rules đã được định nghĩa	
Primary Use Case Flow of Events:	
1.	**Integration Service nhận responses từ APIs**
2.	Validate data format từ mỗi nhà xe
3.	Transform về common format
4.	Handle pagination data từ VXR
5.	Merge tất cả tickets vào dataset
6.	Apply business rules (chênh lệch giá ≤ 10k)
7.	Sort theo price, time, rating
8.	Add metadata (nhà xe, availability)
9.	Apply caching strategy
10.	Return unified list tới Backend
Primary Use Case Post Conditions: – Danh sách thống nhất đã được format	

– Business rules đã được apply – Dữ liệu đã sort theo logic – Sẵn sàng hiển thị cho user	
Alternate Use Case #1 Flow of Events: Data inconsistency giữa nhà xe	
1.	**Thực hiện bước 1-3 như primary flow**
2.	Phát hiện data format không consistent
3.	Apply fallback transformation
4.	Log error cho monitoring
5.	Continue với best effort processing
Alternate Use Case #1 Post Conditions: – System vẫn hoạt động với data có sẵn – Errors được log – User experience không bị interrupt	

Use Case Title: Xử lý lỗi từ external systems	**Use Case ID:** UC2.7
General Use Case Description: Hệ thống xử lý các lỗi từ external systems (timeout, API error, network issues) để đảm bảo fault tolerance và user experience.	
Entitles Involved: UC_INTEGRATION_001, UC_BACKEND_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_CACHE_001	
Preconditions: – Integration service đang hoạt động – Error handling được cấu hình – Fallback mechanisms sẵn sàng – Cache layer available	
Primary Use Case Flow of Events:	
1.	**Integration Service detect API error**

2.	Classify error type (timeout, 5xx, network)
3.	Apply appropriate retry strategy
4.	Log error với chi tiết context
5.	Check cache cho fallback data
6.	Notify monitoring system
7.	Continue với available data
8.	Return partial results với error flags
9.	Update health check status
10.	Schedule retry nếu appropriate
Primary Use Case Post Conditions: – Lỗi được handle gracefully – User vẫn nhận được results – System stability được maintain – Errors được log và monitor	
Alternate Use Case #1 Flow of Events: Tất cả external systems đều lỗi	
1.	**Thực hiện bước 1-3 như primary flow**
2.	Tất cả 3 APIs đều không phản hồi
3.	Integration Service check cache
4.	Return cached data với warning
5.	Notify admin về critical failure
Alternate Use Case #1 Post Conditions: – Cached data được sử dụng – Critical alert được gửi – System vẫn partially functional	

2.3 Lịch giá (4 Use Case)

UC3.1: Hiển thị lịch giá rẻ nhất

UC3.2: Cập nhật giá theo lịch trình

UC3.3: Cập nhật giá khi user search

UC3.4: Tính toán giá rẻ nhất

Use Case Title: Hiển thị lịch giá rẻ nhất	**Use Case ID:** UC3.1
General Use Case Description: Hệ thống hiển thị calendar view với giá vé rẻ nhất cho mỗi ngày trong 1 tháng tới của tuyến đường cụ thể, đảm bảo nhanh và accurate.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_CACHE_001, UC_DB_001	
Preconditions: – Khách hàng đã chọn tuyến đường – Cache layer đã populate pricing data – Database có pricing history – Mobile app sẵn sàng render calendar	
Primary Use Case Flow of Events:	
1.	**Khách hàng chọn tuyến đường và xem lịch giá**
2.	Mobile app gửi request với route info
3.	Backend check cache cho pricing calendar
4.	Cache hit: trả về 30 ngày pricing data
5.	Backend format thành calendar structure
6.	Tính giá thấp nhất mỗi ngày từ tất cả nhà xe
7.	Add metadata (trends, offers, availability)

8.	Mobile app render calendar
9.	Khách hàng click ngày để xem chi tiết
10.	Drill down vào search results
Primary Use Case Post Conditions: – Calendar hiển thị nhanh (không chậm UI) – Giá phản ánh thông tin accurate – User có thể interact với calendar – Sẵn sàng cho booking flow	
Alternate Use Case #1 Flow of Events: Cache miss hoặc stale data	
1.	**Thực hiện bước 1-3 như primary flow**
2.	Cache miss hoặc data quá cũ
3.	Backend query database
4.	Trigger background job refresh cache
5.	Return data từ database
Alternate Use Case #1 Post Conditions: – User vẫn nhận được calendar – Cache được refresh background – Subsequent requests nhanh hơn	

Use Case Title: Cập nhật giá theo lịch trình	**Use Case ID:** UC3.2
General Use Case Description: Hệ thống tự động cập nhật giá vé theo lịch trình định sẵn (hourly/daily) để đảm bảo pricing calendar luôn fresh.	
Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_CACHE_001, UC_DB_001	
Preconditions:	

– Scheduler đã được cấu hình – APIs của nhà xe đang hoạt động – Cache layer sẵn sàng update – Database có pricing tables	
Primary Use Case Flow of Events:	
1.	Scheduler trigger pricing update job
2.	Backend System khởi động update process
3.	Integration Service gọi parallel 3 APIs
4.	Query pricing cho popular routes
5.	Nhận pricing data từ tất cả nhà xe
6.	Calculate lowest price mỗi ngày
7.	Update pricing calendar trong cache
8.	Store pricing history trong database
9.	Update calendar metadata
10.	Ghi log update thành công
Primary Use Case Post Conditions: - Pricing calendar được refresh - Cache có data mới nhất - History được lưu trong database - Next requests sẽ có data fresh	
Alternate Use Case #1 Flow of Events: Một số APIs không phản hồi	
1.	Thực hiện bước 1-5 như primary flow
2.	An Vui timeout, Futa và VXR OK
3.	Update calendar với 2/3 data sources
4.	Mark An Vui data là stale
5.	Schedule retry cho An Vui
Alternate Use Case #1 Post Conditions:	

- Calendar được partial update - Missing data được mark - Retry mechanism hoạt động

Use Case Title: Cập nhật giá khi user search	**Use Case ID:** UC3.3
General Use Case Description: Hệ thống cập nhật pricing calendar khi user thực hiện search để đảm bảo calendar reflect giá thực tế mới nhất.	
Entitles Involved: UC_USER_001, UC_BACKEND_001, UC_INTEGRATION_001, UC_CACHE_001, UC_DB_001	
Preconditions: – User đã thực hiện search – Search results có pricing data – Cache layer sẵn sàng update – Calendar component đang được hiển thị	
Primary Use Case Flow of Events:	
1.	User thực hiện search vé
2.	Backend nhận search results với pricing
3.	Extract pricing data từ search
4.	Compare với cached calendar prices
5.	Identify differences
6.	Update calendar cache với prices mới
7.	Update database pricing history
8.	Trigger calendar refresh trên UI
9.	User thấy calendar updated
10.	Pricing data stay fresh
Primary Use Case Post Conditions: – Calendar có pricing data mới nhất	

– Cache được sync với reality – User experience consistent – Pricing accuracy improved	
Alternate Use Case #1 Flow of Events: Search results incomplete	
1.	Thực hiện bước 1-4 như primary flow
2.	Search results missing pricing từ 1 nhà xe
3.	Update calendar với available data
4.	Keep existing prices cho missing data
5.	Mark incomplete update
Alternate Use Case #1 Post Conditions: – Partial calendar update – Existing data preserved – Update status tracked	

Use Case Title: Tính toán giá rẻ nhất	**Use Case ID:** UC3.4
General Use Case Description: Hệ thống tính toán và so sánh giá từ tất cả nhà xe để xác định giá rẻ nhất cho mỗi ngày trong calendar.	
Entitles Involved: UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_CACHE_001	
Preconditions: – Có pricing data từ ít nhất 1 nhà xe – Business rules đã được định nghĩa – Cache layer sẵn sàng store results	
Primary Use Case Flow of Events:	
1.	Backend System nhận pricing data
2.	Group pricing theo date và route

3.	Compare prices từ tất cả nhà xe
4.	Apply business rules (valid pricing)
5.	Calculate lowest price mỗi ngày
6.	Identify price trends
7.	Generate price recommendations
8.	Store results trong cache
9.	Update calendar metadata
10.	Return calculated prices
Primary Use Case Post Conditions: – Lowest prices được xác định – Price trends được calculate – Results được cached – Calendar sẵn sàng hiển thị	
Alternate Use Case #1 Flow of Events: Missing pricing data	
1.	Thực hiện bước 1-3 như primary flow
2.	Missing pricing từ 1 hoặc 2 nhà xe
3.	Calculate với available data
4.	Mark dates với incomplete data
5.	Provide best estimate
Alternate Use Case #1 Post Conditions: – Best available prices calculated – Incomplete data marked – Estimates provided	

2.4 User & Booking (7 Use Case)

UC4.1: Đăng ký người dùng

UC4.2: Đăng nhập/Đăng xuất

UC4.3: Đặt vé

UC4.4: Thanh toán

UC4.5: Quản lý đơn hàng

UC4.6: Gửi thông báo

UC4.7: Quản lý profile

Use Case Title: Đăng ký người dùng	**Use Case ID:** UC4.1
General Use Case Description: Người dùng mới tạo tài khoản trong hệ thống để có thể đặt vé, lưu lịch sử và nhận thông báo.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_DB_001, UC_NOTIFY_001	
Preconditions: – Mobile app đã được cài đặt – User có email và số điện thoại hợp lệ – Backend system đang hoạt động – Database sẵn sàng nhận user data	
Primary Use Case Flow of Events:	
1.	User mở app và chọn "Đăng ký
2.	Mobile app hiển thị form đăng ký
3.	User nhập thông tin: email, phone, password
4.	Mobile app validate input
5.	Gửi registration request tới Backend
6.	Backend validate uniqueness
7.	Create user record trong database
8.	Generate verification code
9.	Notification service gửi verification
10.	User nhận và nhập verification code

Primary Use Case Post Conditions:

- User account được tạo trong database
- Verification email/SMS được gửi
- User sẵn sàng verify account
- Registration flow hoàn tất

Alternate Use Case #1 Flow of Events:

Email hoặc phone đã tồn tại

1.	Thực hiện bước 1-6 như primary flow
2.	Backend detect duplicate email/phone
3.	Return error message
4.	Mobile app hiển thị error
5.	Suggest login hoặc reset password

Alternate Use Case #1 Post Conditions:

- Registration bị reject
- User được thông báo duplicate
- Alternative options provided

Use Case Title: Đăng nhập/Đăng xuất	**Use Case ID:** UC4.2

General Use Case Description:

User đăng nhập vào hệ thống để access các chức năng cá nhân và đăng xuất khi hoàn tất.

Entitles Involved:

UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_DB_001, UC_CACHE_001

Preconditions:

- User đã có account
- Mobile app đã cài đặt
- Backend authentication service hoạt động
- Database user table available

Primary Use Case Flow of Events:	
1.	User mở app và chọn "Đăng nhập"
2.	Mobile app hiển thị login form
3.	User nhập email/phone và password
4.	Mobile app validate input
5.	Gửi registration request tới Backend
6.	Backend authenticate credentials
7.	Generate JWT token
8.	Cache user session
9.	Return token tới mobile app
10.	Mobile app store token và redirect
Primary Use Case Post Conditions: – User được authenticate – JWT token generated – Session được cache – User access các chức năng cá nhân	
Alternate Use Case #1 Flow of Events: Sai credentials	
1.	Thực hiện bước 1-6 như primary flow
2.	Backend detect invalid credentials
3.	Return authentication error
4.	Mobile app hiển thị error message
5.	Suggest login hoặc reset password
Alternate Use Case #1 Post Conditions: – Login bị reject – User được thông báo lỗi – Recovery options provided	

Use Case Title: Đặt vé	**Use Case ID:** UC4.3

General Use Case Description:

Khách hàng chọn vé từ search results và thực hiện đặt vé, tạo booking record và reserve ghế.

Entitles Involved:

UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_DB_001

Preconditions:

- User đã login
- User đã chọn vé từ search results
- Vé vẫn available
- Thông tin user đã validate

Primary Use Case Flow of Events:

1.	User chọn vé và click "Đặt vé"
2.	Mobile app hiển thị form thông tin
3.	User nhập thông tin hành khách
4.	Mobile app validate và gửi booking request
5.	Backend check availability từ partner
6.	Partner API reserve seat (temp hold)
7.	Backend create booking record
8.	Generate booking ID và expiration
9.	Return booking confirmation
10.	Mobile app hiển thị booking summary

Primary Use Case Post Conditions:

- Booking record tạo trong database
- Seat được reserve temporary
- Booking ID generated
- 15 phút để complete payment

Alternate Use Case #1 Flow of Events:

Vé đã hết chỗ

1.	Thực hiện bước 1-6 như primary flow
2.	Partner API báo "No seats available"
3.	Backend ghi log và return error
4.	Mobile app hiển thị "Vé đã hết chỗ"
5.	Suggest alternative tickets
Alternate Use Case #1 Post Conditions: – Booking không được tạo – Error message rõ ràng – Alternatives provided	

Use Case Title: Thanh toán	**Use Case ID:** UC4.4
General Use Case Description: Khách hàng thanh toán cho booking đã tạo thông qua payment gateway và confirm booking với nhà xe.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_PAYMENT_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_DB_001, UC_NOTIFY_001	
Preconditions: – Booking record đã tạo và chưa expire – Payment gateway hoạt động – User đã chọn payment method – Seat vẫn reserved	
Primary Use Case Flow of Events:	
1.	User chọn payment method
2.	Mobile app redirect tới payment gateway
3.	User nhập payment information
4.	Payment gateway process payment
5.	Backend nhận payment confirmation
6.	Backend confirm booking với partner

7.	Partner issue ticket
8.	Backend update booking status "CONFIRMED"
9.	Trigger notification service
10.	Mobile app hiển thị success
Primary Use Case Post Conditions: – Payment successful – Booking confirmed – Ticket issued – Confirmation sent via email/SMS	
Alternate Use Case #1 Flow of Events: Payment failed	
1.	Thực hiện bước 1-4 như primary flow
2.	Payment gateway return failed
3.	Backend ghi log payment error
4.	Keep booking "PENDING"
5.	Mobile app hiển thị error với retry
Alternate Use Case #1 Post Conditions: – Payment failure logged – Booking vẫn active (chưa expire) – Retry options available	

Use Case Title: Quản lý đơn hàng	**Use Case ID:** UC4.5
General Use Case Description: Khách hàng xem lịch sử đặt vé, track booking status, và thực hiện cancel/refund nếu cần.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_DB_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_PAYMENT_001	

Preconditions: – User đã login – User có booking history – Database có booking records – Mobile app có booking management UI	
Primary Use Case Flow of Events:	
1.	User mở "Đơn hàng của tôi
2.	Mobile app request booking history
3.	Backend query database cho user bookings
4.	Return booking list với status
5.	Mobile app hiển thị booking list
6.	User chọn booking để xem chi tiết
7.	Mobile app hiển thị booking details
8.	User có thể cancel nếu eligible
9.	Backend process cancellation
10.	Update booking status và refund
Primary Use Case Post Conditions: – Booking history displayed – User có thể track status – Cancellation processed nếu có – Refund initiated nếu applicable	
Alternate Use Case #1 Flow of Events: Booking không thể cancel	
1.	Thực hiện bước 1-8 như primary flow
2.	User click cancel nhưng past deadline
3.	Backend check cancellation policy
4.	Return "Cannot cancel" với reason
5.	Mobile app hiển thị policy explanation
Alternate Use Case #1 Post Conditions:	

– Cancellation request rejected – Policy explanation provided – Booking remains active

Use Case Title: Gửi thông báo	**Use Case ID:** UC4.6
General Use Case Description: Hệ thống gửi thông báo tới khách hàng về booking confirmation, payment status, trip reminders qua email/SMS.	
Entitles Involved: UC_BACKEND_001, UC_NOTIFY_001, UC_USER_001, UC_DB_001	
Preconditions: – Notification service hoạt động – USER có email/phone valid – Notification templates đã setup – Events trigger notification	
Primary Use Case Flow of Events:	
1.	Backend trigger notification event
2.	Notification service nhận event
3.	Select appropriate template
4.	Populate template với booking data
5.	Determine delivery channels
6.	Send email notification
7.	Send SMS notification
8.	Send push notification
9.	Track delivery status
10.	Log notification history
Primary Use Case Post Conditions: – Notification sent via multiple channels – Delivery status tracked	

– USER receives timely updates – Notification history logged	
Alternate Use Case #1 Flow of Events: Notification delivery failed	
1.	Thực hiện bước 1-8 như primary flow
2.	Email/SMS delivery failed
3.	Notification service retry
4.	Try alternative channels
5.	Log delivery failure
Alternate Use Case #1 Post Conditions: – Retry mechanism activated – Alternative channels used – Delivery failure logged	

Use Case Title: Quản lý profile	**Use Case ID:** UC4.7
General Use Case Description: Khách hàng xem và cập nhật thông tin cá nhân, thay đổi password, quản lý preferences.	
Entitles Involved: UC_USER_001, UC_MOBILE_001, UC_BACKEND_001, UC_DB_001, UC_NOTIFY_001	
Preconditions: – User đã login – Database có user profile – Mobile app có profile management UI – Backend validation rules setup	
Primary Use Case Flow of Events:	
1.	User mở "Thông tin cá nhân
2.	Mobile app request profile data

3.	Backend query user profile
4.	Return profile information |
5.	Mobile app hiển thị profile form
6.	User update thông tin
7.	Mobile app validate changes
8.	Send update request tới Backend
9.	Backend update database
10.	Confirm update success
Primary Use Case Post Conditions: – Profile updated successfully – Changes reflected in database – User receives confirmation – Updated info available system-wide	
Alternate Use Case #1 Flow of Events: Email/phone change cần verification	
1.	Thực hiện bước 1-8 như primary flow
2.	User change email/phone
3.	Backend require verification
4.	Send verification code
5.	User nhập verification code
Alternate Use Case #1 Post Conditions: – Sensitive changes require verification – Verification process completed – Changes applied sau verification	

2.5 Quản Trị (6 Use Case)

UC5.1: Monitor hệ thống

UC5.2: Quản lý nhà xe đối tác

UC5.3: Quản lý cấu hình

UC5.4: Báo cáo thống kê

UC5.5: Xử lý khiếu nại

UC5.6: Quản lý người dùng

Use Case Title: Monitor hệ thống	**Use Case ID:** UC5.1
General Use Case Description: Admin giám sát real-time system health, performance metrics, và phát hiện issues để maintain 24/7 uptime.	
Entitles Involved: UC_ADMIN_001, UC_BACKEND_001, UC_DB_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001	
Preconditions: – Admin dashboard setup – Monitoring tools configured – Alert systems hoạt động – Admin có access rights	
Primary Use Case Flow of Events:	
1.	Admin truy cập monitoring dashboard
2.	Dashboard hiển thị real-time metrics
3.	Monitor API response times
4.	Check database performance
5.	Monitor search service
6.	Check cache hit rates
7.	Monitor booking success rates
8.	Review error logs
9.	Check resource utilization
10.	Alert nếu threshold violation
Primary Use Case Post Conditions: – System health visible	

– Issues identified và escalated – Performance metrics tracked – Uptime requirements monitored	
Alternate Use Case #1 Flow of Events: System outage detected	
1.	Monitoring detect service outage
2.	Alert gửi tới admin team
3.	Admin investigate root cause
4.	Apply immediate fixes
5.	Track recovery time
Alternate Use Case #1 Post Conditions: – Issue resolved/mitigated – Downtime minimized – Post-mortem report created	

Use Case Title: Quản lý nhà xe đối tác	**Use Case ID:** UC5.2
General Use Case Description: Admin quản lý thông tin nhà xe đối tác, configure APIs, add/remove partners, monitor integration health.	
Entitles Involved: UC_ADMIN_001, UC_BACKEND_001, UC_INTEGRATION_001, UC_FUTA_001, UC_VXR_001, UC_AV_001, UC_DB_001	
Preconditions: – Admin có partner management access – Partner configuration UI available – Integration service hoạt động – Database partner tables setup	
Primary Use Case Flow of Events:	
1.	Admin mở partner management

2.	Backend return partner list
3.	Admin chọn partner để configure
4.	Hiển thị partner configuration
5.	Admin update API endpoints
6.	Admin update authentication
7.	Admin test connection
8.	Backend validate configuration
9.	Save partner configuration
10.	Integration service reload config

Primary Use Case Post Conditions:

- Partner configuration updated
- API integration tested
- New config applied
- Integration health monitored

Alternate Use Case #1 Flow of Events:

Add new partner

1.	Admin click "Add new partner"
2.	Hiển thị partner setup form
3.	Admin nhập partner information
4.	Admin configure API settings
5.	Test integration

Alternate Use Case #1 Post Conditions:

- New partner added
- Integration configured
- Ready for production

Use Case Title: Quản lý cấu hình	**Use Case ID:** UC5.3

General Use Case Description:

Admin quản lý system configuration, business rules, feature flags, và operational parameters.

Entitles Involved: UC_ADMIN_001, UC_BACKEND_001, UC_DB_001, UC_CACHE_001	
Preconditions: – Admin có configuration access – Configuration management UI available – Backend configuration system setup – Database config tables available	
Primary Use Case Flow of Events:	
1.	Admin mở system configuration
2.	Backend return current config
3.	Admin xem configuration categories
4.	Admin chọn category để edit
5.	Update configuration values
6.	Admin validate changes
7.	Backend apply configuration
8.	Update cache và services
9.	Test configuration changes
10.	Confirm changes applied
Primary Use Case Post Conditions: – Configuration updated – Changes applied system-wide – Services reloaded với new config – Changes tested và verified	
Alternate Use Case #1 Flow of Events: Configuration validation failed	
1.	Thực hiện bước 1-6 như primary flow
2.	Backend validation detect error
3.	Return validation error
4.	Admin fix configuration
5.	Retry validation

Alternate Use Case #1 Post Conditions:
– Invalid configuration rejected – Error message provided – Admin can retry with fixes

Use Case Title: Báo cáo thống kê	**Use Case ID:** UC5.4
General Use Case Description: Hệ thống generate báo cáo business metrics, performance analytics, compliance reports cho management.	
Entitles Involved: UC_ADMIN_001, UC_BACKEND_001, UC_DB_001	
Preconditions: – Database có sufficient transaction data – Reporting system setup – Admin có reporting access – Report templates available	
Primary Use Case Flow of Events:	
1.	Admin chọn report type và time range
2.	Backend query database cho metrics
3.	Calculate conversion rates
4.	Calculate revenue per partner
5.	Analyze pricing compliance
6.	Generate performance charts
7.	Create summary với insights
8.	Export report tới PDF/Excel
9.	Admin review report
10.	Distribute report tới stakeholders
Primary Use Case Post Conditions: – Accurate report generated – Business metrics evaluated	

– Compliance checked – Insights provided	
Alternate Use Case #1 Flow of Events: Insufficient data cho report	
1.	Thực hiện bước 1-2 như primary flow
2.	Database không có enough data
3.	Backend return data availability notice
4.	Admin adjust time range
5.	Generate report với available data
Alternate Use Case #1 Post Conditions: – Report generated với available data – Data limitations noted – Alternative time range used	

Use Case Title: Quản lý người dùng	**Use Case ID:** UC5.6
General Use Case Description: Admin quản lý user accounts, permissions, handle account issues, perform user-related administrative tasks.	
Entitles Involved: UC_ADMIN_001, UC_USER_001, UC_BACKEND_001, UC_DB_001, UC_NOTIFY_001	
Preconditions: – Admin có user management access – User management UI available – Database user tables accessible – Admin tools configured	
Primary Use Case Flow of Events:	
1.	Admin mở user management
2.	Backend return user list
3.	Admin search/filter users

4.	Admin chọn user để manage
5.	View user profile và history
6.	Admin perform actions (reset, disable)
7.	Backend update user record
8.	Log admin action
9.	Notify user nếu cần
10.	Confirm action completed
Primary Use Case Post Conditions: – User account updated – Admin action logged – User notified về changes – Changes applied system-wide	
Alternate Use Case #1 Flow of Events: User account locked	
1.	Thực hiện bước 1-6 như primary flow
2.	Admin unlock user account
3.	Backend remove account lock
4.	Send unlock notification
5.	User có thể login lại
Alternate Use Case #1 Post Conditions: – Account unlocked – User can access system – Unlock action logged	

CÂU 3: TÌM TẤT CẢ CÁC QUALITY ATTRIBUTES CHO HỆ THỐNG BÁN VÉ XE ONLINE

QAS 1: Tìm Kiếm Vé

- **Mô tả**: Tìm kiếm vé từ nhiều nhà xe đối tác
- **Loại**: Performance
- **Kịch bản**:
    1. **Stimulus**: Khách hàng nhập thông tin tìm kiếm vé: nơi đi, nơi đến, ngày đi.
    2. **Source of Stimulus**: Khách hàng muốn đặt vé xe trong giờ cao điểm.
    3. **Environment**: Runtime, peak hours, over 1000 concurrent users.
    4. **Artifacts**: Search service, Integration service, External APIs (Futa API, VXR API, An Vui API), Cache layer, Backend system.
    5. **Response**: Hiển thị danh sách vé từ tất cả các nhà xe có sẵn, được tổng hợp và sort theo giá/thời gian.
    6. **Response measure**:
        - Thời gian phản hồi tổng < 5s cho kết quả đầy đủ
        - Futa API: < 3s
        - VXR API: < 5s (với phân trang)
        - An Vui API: < 15s (có thể bất đồng bộ)
        - Hỗ trợ 1000+ người dùng đồng thời
    7. **Architecture risk**:
        - Hết thời gian chờ từ An Vui API (15s) có thể làm chậm phản hồi tổng thể
        - Ảnh hưởng đến kiểm thử chất lượng tìm kiếm địa điểm nếu cơ sở dữ liệu quá tải
        - Ảnh hưởng đến kiểm thử chất lượng thanh toán nếu hệ thống quá tải
        - Phân trang VXR có thể tạo điểm nghẽn

QAS 2: Tìm Kiếm Địa Điểm

- **Mô tả**: Gợi ý địa điểm tự động khi user nhập từ khóa
- **Loại**: Performance
- **Kịch bản**:
    1. **Stimulus**: Khách hàng gõ từ khóa tìm kiếm địa điểm (nơi đi/nơi đến).
    2. **Source of Stimulus**: Khách hàng cần tìm kiếm địa điểm để bắt đầu booking flow.
    3. **Environment**: Runtime, normal operation, more than 100 concurrent search requests.
    4. **Artifacts**: Search service, Location database, Cache layer, Elasticsearch, Mobile app.
    5. **Response**: Hiển thị danh sách gợi ý địa điểm phù hợp với từ khóa.
    6. **Response measure**:
        - Thời gian phản hồi < 1.5s với 100 requests đồng thời
        - Data response < 200KB
        - Cache hit rate > 80%
        - Search accuracy > 90%
    7. **Architecture risk**:
        - Ảnh hưởng đến hiệu năng của tìm vé nếu cơ sở dữ liệu quá tải
        - Ảnh hưởng đến thanh toán nếu dịch vụ tìm kiếm quá tải
        - Elasticsearch ngừng hoạt động → chuyển dự phòng sang truy vấn cơ sở dữ liệu (chậm hơn)
        - Trượt bộ nhớ đệm → tăng tải cho cơ sở dữ liệu

QAS 3: Hiển Thị Lịch Giá

- **Mô tả**: Hiển thị calendar với giá rẻ nhất không làm chậm UI
- **Loại**: Performance
- **Kịch bản**:

1. **Stimulus**: Khách hàng chọn tuyến đường và yêu cầu xem lịch giá 1 tháng.
2. **Source of Stimulus**: Khách hàng muốn tìm ngày có giá tốt nhất để đi.
3. **Environment**: Runtime, normal operation, multiple concurrent schedule requests.
4. **Artifacts**: Cache layer, Pricing service, Database, Mobile app UI, Background scheduler.
5. **Response**: Hiển thị calendar view với giá rẻ nhất mỗi ngày trong 1 tháng.
6. **Response measure**:
    - Hiển thị nhanh, không làm chậm giao diện người dùng
    - Loading time < 2s
    - Cache hit rate > 90%
    - UI responsive trong quá trình load
7. **Architecture risk**:
    - Trượt bộ nhớ đệm → truy vấn cơ sở dữ liệu → giao diện người dùng bị treo
    - Tập dữ liệu lớn → áp lực bộ nhớ ứng dụng di động
    - Yêu cầu lịch đồng thời → điểm nghẽn cơ sở dữ liệu
    - Cập nhật giá ngầm → xóa hiệu lực bộ nhớ đệm → giảm hiệu năng

QAS 4: Hoạt Động 24/7

- **Mô tả**: Hệ thống bán vé hoạt động liên tục 24/7
- **Loại**: Availability
- **Kịch bản**:
    1. **Stimulus**: Khách hàng truy cập hệ thống bất kỳ lúc nào trong ngày.
    2. **Source of Stimulus**: Khách hàng cần đặt vé, admin cần monitor hệ thống.
    3. **Environment**: Runtime, 24/7 operation, including maintenance periods.

4. **Artifacts**: Load balancer, Multiple server instances, Database cluster, External APIs, Monitoring system.
5. **Response**: Hệ thống phản hồi và cho phép thực hiện tất cả chức năng.
6. **Response measure**:
    - Thời gian hoạt động ≥ 99.9% (8.76 giờ ngừng hoạt động/năm)
    - Thời gian trung bình khôi phục (MTTR) < 15 phút
    - Triển khai không gián đoạn
    - Suy giảm có kiểm soát khi dịch vụ gặp sự cố
7. **Architecture risk**:
    - Điểm lỗi đơn lẻ tại cơ sở dữ liệu → hệ thống ngừng hoạt động
    - Mất kết nối API bên ngoài → mất một phần chức năng
    - Sự cố triển khai → gián đoạn dịch vụ
    - Điểm mù giám sát → sự cố không được phát hiện

QAS 5: Tích Hợp Nhà Xe Mới

- **Mô tả**: Dễ dàng tích hợp thêm nhà xe mới vào hệ thống
- **Loại**: Scalability/Modifiability
- **Kịch bản**:
    1. **Stimulus**: Business team muốn tích hợp nhà xe mới vào hệ thống.
    2. **Source of Stimulus**: Yêu cầu mở rộng kinh doanh, hợp tác mới.
    3. **Environment**: Development/Production environment, system modification phase.
    4. **Artifacts**: Integration service, API gateway, Configuration management, Database schema, Partner adapters.
    5. **Response**: Nhà xe mới được tích hợp và có thể bán vé qua platform.
    6. **Response measure**:
        - Thời gian tích hợp < 2 weeks
        - Không ảnh hưởng đến đối tác hiện tại
        - Tích hợp dựa trên cấu hình
        - Kiểm thử tự động cho đối tác mới

- Không thay đổi mã nguồn trong hệ thống lõi

7. **Architecture risk**:
   - Logic viết cứng trong mã nguồn → khó mở rộng
   - Liên kết chặt chẽ → ảnh hưởng đến đối tác hiện tại
   - Cấu hình thủ công → lỗi con người
   - Không có API chuẩn hóa → tích hợp phức tạp
   - Thay đổi cấu trúc cơ sở dữ liệu → hệ thống ngừng hoạt động

QAS 6: Partner Api Failures

- **Mô tả**: Một hãng xe lỗi không ảnh hưởng đến các hãng khác
- **Loại**: Fault Tolerance/Reliability
- **Kịch bản**:
  1. **Stimulus**: API của một nhà xe (Futa/VXR/An Vui) bị timeout hoặc return error.
  2. **Source of Stimulus**: Sự cố hệ thống bên ngoài, sự cố mạng, bảo trì từ đối tác.
  3. **Environment**: Runtime, normal operation, external dependency failure.
  4. **Artifacts**: Integration service, Circuit breaker, Cache layer, Retry mechanism, Fallback service.
  5. **Response**: Hệ thống tiếp tục hoạt động với data từ 2 partners còn lại.
  6. **Response measure**:
     - Hệ thống tiếp tục hoạt động với 2/3 đối tác
     - Dịch vụ suy giảm dưới 33% chức năng
     - Tự động khôi phục khi đối tác hoạt động trở lại
     - Không lỗi hiển thị với người dùng
     - Phản hồi nhanh khi lỗi < 5 giây
  7. **Architecture risk**:
     - Không có circuit breaker (mạch ngắt) → sự cố dây chuyền
     - Tài nguyên dùng chung → một lỗi ảnh hưởng tất cả

- Không có dữ liệu dự phòng → mất toàn bộ tính năng
- Thời gian chờ dài → trải nghiệm người dùng kém
- Không giám sát → sự cố không được phát hiện

QAS 7: Không Gián Đoạn

- **Mô tả**: Chức năng tìm kiếm địa điểm luôn có sẵn, không bị gián đoạn
- **Loại**: Reliability
- **Kịch bản**:
    1. **Stimulus**: Khách hàng thực hiện tìm kiếm địa điểm trong bất kỳ điều kiện nào.
    2. **Source of Stimulus**: Khách hàng bắt đầu booking journey.
    3. **Environment**: Runtime, varying load conditions, potential service degradation..
    4. **Artifacts**: Search service, Location database, Cache layer, Backup systems, Health monitoring.
    5. **Response**: Tìm kiếm địa điểm luôn trả về kết quả cho user.
    6. **Response measure**:
        - Độ sẵn sàng dịch vụ > 99.95%
        - Không có thất bại dịch vụ hoàn toàn
        - Cơ chế dự phòng luôn hoạt động
        - Thời gian trung bình giữa các lần hỏng (MTBF) > 720 giờ
        - Suy giảm có kiểm soát khi xảy ra sự cố
    7. **Architecture risk**:
        - Cơ sở dữ liệu đơn → điểm lỗi đơn lẻ
        - Không nhân bản dữ liệu → rủi ro mất dữ liệu
        - Phụ thuộc vào bộ nhớ đệm → dịch vụ lỗi khi bộ nhớ đệm ngừng hoạt động
        - Không kiểm tra tình trạng hệ thống → sự cố âm thầm (không phát hiện)
        - Cạn kiệt tài nguyên → dịch vụ không sẵn sàng

QAS 8: Trải Nghiệm Người Dùng

- **Mô tả**: Giao diện nhanh chóng, thuận tiện cho người dùng
- **Loại**: Usability
- **Kịch bản**:
    1. **Stimulus**: Khách hàng sử dụng mobile app để tìm kiếm và đặt vé.
    2. **Source of Stimulus**: Tương tác của người dùng cuối với luồng đặt chỗ.
    3. **Environment**: Runtime, mobile devices, varying network conditions.
    4. **Artifacts**: Mobile app (Flutter), UI/UX components, API responses, Loading states.
    5. **Response**: User hoàn thành booking journey một cách thuận tiện.
    6. **Response measure**:
        - Tỷ lệ hoàn tất đặt chỗ > 85%
        - Thời gian đặt chỗ trung bình < 3 phút
        - Điểm hài lòng của người dùng > 4.5/5
        - Tỷ lệ crash của ứng dụng < 0.1%
        - Trạng thái tải cho tất cả thao tác
    7. **Architecture risk**:
        - API phản hồi chậm → người dùng bỏ đi
        - Luồng giao diện phức tạp → người dùng bối rối
        - Không có khả năng hoạt động ngoại tuyến → trải nghiệm di động kém
        - Trạng thái không nhất quán → người dùng khó chịu
        - Không xử lý lỗi → ứng dụng bị crash

QAS 9: Thời Gian Deploy

- **Mô tả**: Hệ thống deploy trong ≤ 30s và sẵn sàng bán hàng
- **Loại**: Deployability
- **Kịch bản**:

1. **Stimulus**: DevOps team thực hiện deployment của version mới.
2. **Source of Stimulus**: Code changes, bug fixes, feature updates.
3. **Environment**: Deployment phase, production environment.
4. **Artifacts**: CI/CD pipeline, Container orchestration, Database migrations, Health checks.
5. **Response**: Hệ thống được deploy và sẵn sàng serve requests.
6. **Response measure**:
   - Triển khai không gián đoạn
   - Tự động rollback < 60s nếu fail
   - Tỷ lệ kiểm tra sức khỏe thành công 100%
   - Di chuyển cơ sở dữ liệu < 10s
7. **Architecture risk**:
   - Database migration lâu → vượt quá 30 giây
   - Các thành phần phụ thuộc không sẵn sàng → triển khai thất bại
   - Không kiểm tra sức khỏe hệ thống → triển khai phiên bản lỗi
   - Bước thủ công → lỗi con người
   - Không có chiến lược rollback → thời gian ngừng hoạt động kéo dài

QAS 10: Cấu Hình Dễ Dàng

- **Mô tả**: Cấu hình số items trên 1 trang nhanh chóng và dễ dàng
- **Loại**: Modifiability/Configurability
- **Kịch bản**:
  1. **Stimulus**: Admin cần thay đổi số items hiển thị trên 1 trang search results.
  2. **Source of Stimulus**: Thay đổi yêu cầu nghiệp vụ, tối ưu hóa hiệu năng.
  3. **Environment**: Runtime, administrative configuration interface.

4. **Artifacts**: Configuration management, Admin dashboard, Backend API, Cache layer.
5. **Response**: Cấu hình mới được apply và hiệu lực ngay lập tức.
6. **Response measure**:
    - Thời gian thay đổi cấu hình < 1 minute
    - Không cần khởi động lại hệ thống
    - Hiệu lực ngay lập tức với tất cả người dùng
    - Không cần triển khai mã nguồn
    - Quay lui cấu hình < 30s
7. **Architecture risk**:
    - Giá trị viết cứng trong mã nguồn → cần chỉnh sửa mã nguồn
    - Không kiểm tra tính hợp lệ → cấu hình không hợp lệ
    - Không có hiệu lực ngay lập tức → người dùng bối rối
    - Không có quay lui → mắc kẹt với cấu hình lỗi
    - Bộ nhớ đệm không được cập nhật → hành vi không nhất quán

QAS 11: Xác Thực Và Thanh Toán

- **Mô tả**: Bảo mật thông tin người dùng và giao dịch thanh toán
- **Loại**: Security
- **Kịch bản**:
    1. **Stimulus**: Khách hàng thực hiện đăng nhập và thanh toán.
    2. **Source of Stimulus**: Người dùng cuối, đối tượng độc hại tiềm ẩn..
    3. **Environment**: Thời gian chạy, mạng internet công cộng, giao diện di động/web.
    4. **Artifacts**: Authentication service, Payment gateway, Encryption, JWT tokens, SSL/TLS.
    5. **Response**: Thông tin được bảo mật và giao dịch được xử lý an toàn.
    6. **Response measure**:
        - Tuân thủ PCI DSS

- Thời hạn hết hiệu lực của JWT token < 24 giờ
- Mã hóa SSL/TLS 100%
- Không ghi dữ liệu nhạy cảm vào log
- Tỷ lệ phát hiện gian lận > 99%

7. **Architecture risk**:
   - Xác thực yếu → truy cập trái phép
   - Rò rỉ dữ liệu thanh toán → trách nhiệm tài chính
   - Không mã hóa → chặn dữ liệu
   - Tấn công SQL injection → rò rỉ dữ liệu
   - Không có nhật ký kiểm tra → vấn đề tuân thủ

QAS 12: Cập Nhật Tức Thời

- **Mô tả**: Cập nhật địa điểm từ nhà xe phải được khách hàng thấy tức thời
- **Loại**: Data Consistency
- **Kịch bản**:
  1. **Stimulus**: Nhà xe cập nhật thông tin địa điểm mới.
  2. **Source of Stimulus**: Nhân viên hệ thống đối tác (Futa/VXR/An Vui).
  3. **Environment**: Thời gian chạy, đồng bộ dữ liệu thời gian thực.
  4. **Artifacts**: Integration service, Database, Cache layer, Real-time sync, Webhook handlers.
  5. **Response**: Khách hàng thấy thông tin địa điểm mới ngay lập tức.
  6. **Response measure**:
     - Thời gian lan truyền dữ liệu < 5s
     - Xóa hiệu lực bộ nhớ đệm < 1s
     - 100% nhất quán dữ liệu giữa tất cả người dùng
     - Không hiển thị dữ liệu cũ
     - Tỷ lệ đồng bộ thời gian thực thành công > 99%
  7. **Architecture risk**:
     - Bộ nhớ đệm không bị xóa hiệu lực → dữ liệu cũ

- Cập nhật bất đồng bộ → tạm thời không nhất quán
- Sự cố mạng → lỗi đồng bộ
- Khóa cơ sở dữ liệu → trì hoãn cập nhật
- Không xử lý xung đột → hỏng dữ liệu

CÂU 4: TÌM TECHNICAL CONSTRAINTS CHO HỆ THỐNG BÁN VÉ XE ONLINE

TC1: Platform & Technology Constraints

TC1.1: Mobile Application Technology

Constraint: Mobile app phải được phát triển bằng Flutter

Căn cứ từ đề bài: *"Phát triển mobile app dùng flutter"*

Rationale:

- Cross-platform development cho cả Android và iOS
- Single codebase giảm effort và maintenance cost
- Consistent UI/UX experience across platforms
- Hot reload cho faster development cycle

Implications:

- Development team cần Flutter expertise
- Dart programming language required
- Native features cần Flutter plugins
- App store compliance cho cả Android và iOS
- Performance considerations cho cross-platform

TC1.2: Backend Technology

Constraint: Backend system phải sử dụng Java + Spring Framework

Căn cứ từ đề bài: "Phát triển BE dùng java, spring framework"

Rationale:

- Enterprise-grade framework với proven scalability
- Rich ecosystem cho integration và microservices
- Strong community support và documentation
- Mature security features
- Easy to find skilled developers

Implications:

- JVM runtime environment required
- Spring Boot cho rapid development
- Spring Security cho authentication
- Spring Integration cho external APIs
- Memory management và garbage collection tuning

TC1.3: Database Technology

Constraint: Database phải sử dụng PostgreSQL

Căn cứ từ đề bài: "database postgres"

Rationale:

- ACID compliance cho transaction reliability
- Advanced indexing cho search performance
- JSON support cho flexible data storage
- Proven scalability cho large datasets
- Open-source với no licensing cost

Implications:

- SQL expertise required
- Database design cho performance optimization
- Connection pooling cho concurrent access
- Backup và recovery strategies
- Performance tuning cho search queries

TC2: Performance Constraints

TC2.1: Deployment Time Constraint

Constraint: Deploy time ≤ 30 seconds và sẵn sàng bán hàng

Căn cứ từ đề bài: "Deploy time ≤ 30s sẵn sàng bán hàng"

Rationale:

- Minimize business impact during deployments
- Enable frequent releases
- Reduce deployment risk

- Support CI/CD best practices

Implications:

- Containerization required (Docker)
- Orchestration platform (Kubernetes)
- Health check mechanisms
- Database migration strategies
- Rolling deployment pattern
- Automated testing pipeline

TC2.2: Uptime Constraint

Constraint: Hệ thống phải hoạt động 24/7

Căn cứ từ đề bài: "bán hàng liên tục 24/7"

Rationale:

- Business continuity requirement
- Global USER base across timezones
- Competitive advantage
- Revenue protection

Implications:

- High availability architecture
- Load balancing và failover
- Monitoring và alerting systems
- Disaster recovery planning
- No single point of failure
- Scheduled maintenance strategies

TC2.3: Response Time Constraints

Constraint: API response time limits cho external partners

Căn cứ từ đề bài: "vé xe rẻ <5s, futa < 3s, an vui < 15s"

Rationale:

- Partner SLA requirements

- User experience expectations
- System performance targets
- Business process efficiency

Implications:

- Timeout handling mechanisms
- Circuit breaker patterns
- Caching strategies
- Parallel processing
- Async/await patterns
- Performance monitoring

TC3: Data Size Constraints

TC3.1: Current Data Size Limit

Constraint: Dung lượng data hiện tại < 1MB

Căn cứ từ đề bài: "Dung lượng hiện tại < 1MB"

Rationale:

- Mobile data usage optimization
- Fast loading times
- Battery life preservation
- Network cost reduction

Implications:

- Data compression techniques
- Pagination implementation
- Incremental data loading
- Cache size management
- Image optimization
- JSON response optimization

TC3.2: Future Data Size Limit

Constraint: Dung lượng data tương lai < 5MB

Căn cứ từ đề bài: "Dung lượng tương lai < 5MB"

Rationale:

- Scalability planning
- Future growth accommodation
- Performance sustainability
- Storage cost management

Implications:

- Scalable architecture design
- Database partitioning strategies
- CDN implementation
- Data archiving policies
- Performance impact assessment
- Storage optimization techniques

TC4: Integration Constraints

TC4.1: External Api Integration

Constraint: Tích hợp với 3 hệ thống partner khác nhau

Căn cứ từ đề bài: "Futa, Vé Xe Rẻ, An Vui"

Rationale:

- Business partnership requirements
- Market coverage expansion
- Revenue diversification
- USER choice provision

Implications:

- API gateway implementation
- Multiple authentication mechanisms

- Data format standardization
- Error handling strategies
- Rate limiting compliance
- API versioning management

TC4.2: Pagination Constraint

Constraint: Xử lý khác nhau cho pagination

Căn cứ từ đề bài: "Vé Xe Rẻ có hỗ trợ trả theo trang, Futa và An Vui không"

Rationale:

- Partner system limitations
- Data volume management
- Performance optimization
- API design differences

Implications:

- Flexible integration layer
- Data aggregation strategies
- Memory management
- Response time optimization
- Concurrent request handling
- Result merging algorithms

TC5: Fault Tolerance Constraints

TC5.1: Isolation Constraint

Constraint: Một hãng lỗi không ảnh hưởng đến các hãng khác

Căn cứ từ đề bài: "Bất kỳ một hãng nào có lỗi thì cũng không ảnh hưởng đến các hãng khác"

Rationale:

- Service reliability
- User experience protection

- Business continuity
- Risk mitigation

Implications:

- Circuit breaker pattern
- Bulkhead isolation
- Timeout mechanisms
- Fallback strategies
- Independent service instances
- Error containment

TC5.2: Real-Time Update Constraint

Constraint: Cập nhật tức thời cho khách hàng

Căn cứ từ đề bài: "khi có cập nhật địa điểm từ các hãng xe thì khách hàng phải được thấy tức thời"

Rationale:

- Data accuracy requirements
- User experience expectations
- Business process efficiency
- Competitive advantage

Implications:

- Real-time synchronization
- Cache invalidation strategies
- WebSocket connections
- Event-driven architecture
- Conflict resolution mechanisms
- Data consistency patterns

TC6: Scalability Constraints

TC6.1: Extensibility Constraint

Constraint: Dễ dàng tích hợp nhà xe mới

Căn cứ từ đề bài: "Trong tương lai tích hợp thêm các nhà xe khác"

Rationale:

- Business growth support
- Market expansion capability
- Architecture flexibility
- Development efficiency

Implications:

- Plugin architecture
- Configuration-driven integration
- Abstract interfaces
- Standardized protocols
- Automated onboarding
- Minimal code changes

TC7: Development Constraints

TC7.1: Timeline Constraint

Constraint: 6 tháng development time

Căn cứ từ đề bài: "Thời gian phát triển: 6 tháng"

Rationale:

- Business launch timeline
- Market opportunity window
- Budget limitations
- Resource allocation

Implications:

- Agile development methodology
- MVP approach
- Parallel development streams
- Risk management
- Quality assurance planning

- Scope prioritization

TC7.2: Team Size Constraint

Constraint: 5 nhân sự development team

Căn cứ từ đề bài: "5 nhân sự"

Rationale:

- Budget constraints
- Team management efficiency
- Communication overhead
- Skill distribution

Implications:

- Role specialization
- Knowledge sharing
- Code review processes
- Documentation requirements
- Training needs
- Collaboration tools

CÂU 5: BUSINESS CONSTRAINTS

BC1: Project Constraints

BC1.1: Development Timeline

Constraint: Thời gian phát triển tối đa 6 tháng

Căn cứ từ đề bài: "Thời gian phát triển: 6 tháng"

Business Rationale:

- Market window opportunity
- Competitive advantage timing
- Investment return expectations
- Partnership commitments with bus companies

Business Impact:

- Scope prioritization required
- MVP approach mandatory
- Feature phasing strategy
- Risk of delayed market entry
- Budget overrun if delayed
- Mitigation Strategies:
- Agile development methodology
- Parallel development streams
- Early prototype validation
- Risk-based development planning

BC1.2: Team Size Limitation

Constraint: Đội ngũ phát triển tối đa 5 nhân sự

Căn cứ từ đề bài: "5 nhân sự"

Business Rationale:

- Budget constraints
- Team management efficiency

- Communication overhead minimization
- Cost-effective development

Business Impact:

- Limited parallel development capacity
- Skill specialization requirements
- Knowledge sharing critical
- Single point of failure risks
- Productivity optimization essential
- Mitigation Strategies:
- Cross-functional training
- External vendor partnerships
- Automated testing/deployment
- Clear role definitions

BC2: Revenue & Performance Constraints

BC2.1: Conversion Rate Commitment

Constraint: Tỉ lệ conversion tối thiểu 100 lượt search / 1 vé bán

Căn cứ từ đề bài: "Tỉ lệ bán cam kết 100 lượt search / 1 vé bán"

Business Rationale:

- Revenue generation targets
- Partner commission structures
- Platform profitability
- Market penetration goals

Business Impact:

- UX optimization critical
- Conversion funnel analysis required
- A/B testing necessary
- Marketing cost calculations

- Revenue forecasting baseline
- Mitigation Strategies:
- User behavior analytics
- Conversion optimization techniques
- Price comparison features
- Streamlined booking process
- Trust-building elements

BC2.2: Pricing Compliance

Constraint: Giá bán chênh lệch không được vượt quá 10k VND

Căn cứ từ đề bài: "Giá bán chênh lệch không được vượt quá 10k so với giá gốc"

Business Rationale:

- Partner contract terms
- Price competitiveness
- USER trust maintenance
- Regulatory compliance

Business Impact:

- Automated pricing monitoring required
- Partner relationship management
- Margin limitation
- Compliance reporting needs
- Potential revenue loss
- Mitigation Strategies:
- Real-time price monitoring
- Dynamic pricing algorithms
- Partner negotiation
- Compliance dashboard
- Alert systems

BC3: Operational Constraints

BC3.1: 24/7 Availability Requirement

Constraint: Hệ thống phải hoạt động liên tục 24/7

Căn cứ từ đề bài: "bán hàng liên tục 24/7"

Business Rationale:

- USER service expectations
- Revenue maximization
- Multi-timezone operations
- Competitive advantage

Business Impact:

- High infrastructure costs
- Monitoring system requirements
- Support team coverage
- Maintenance window challenges
- SLA commitments
- Mitigation Strategies:
- High availability architecture
- Automated monitoring
- Incident response procedures
- Planned maintenance strategies
- USER communication protocols

BC3.2: Multi-Partner Integration

Constraint: Tích hợp với 3 nhà xe trong giai đoạn đầu

Căn cứ từ đề bài: "Futa, Vé Xe Rẻ, An Vui"

Business Rationale:

- Market coverage
- Revenue diversification
- USER choice provision

– Risk distribution

Business Impact:

- Complex integration requirements
- Multiple contract negotiations
- Varied technical standards
- Coordination challenges
- Dependency management
- Mitigation Strategies:
- Standardized integration framework
- Phased integration approach
- Partner relationship management
- Technical standardization
- Contingency planning

BC4: Scalability Constraints

BC4.1: Future Expansion Capability

Constraint: Khả năng tích hợp thêm nhà xe trong tương lai

Căn cứ từ đề bài: "Trong tương lai tích hợp thêm các nhà xe khác"

Business Rationale:

- Business growth strategy
- Market expansion
- Revenue scaling
- Competitive positioning

Business Impact:

- Scalable architecture requirements
- Standardization needs
- Investment in flexibility
- Partner onboarding processes

- Maintenance complexity
- Mitigation Strategies:
- Plugin architecture design
- Configuration-driven integration
- Standard API protocols
- Automated onboarding
- Scalable infrastructure

BC4.2: Data Volume Growth

Constraint: Chuẩn bị cho tăng trưởng dữ liệu từ 1MB hiện tại lên 5MB

Căn cứ từ đề bài: "Dung lượng hiện tại < 1MB, dung lượng tương lai < 5MB"

Business Rationale:

- User base expansion
- Feature enhancement
- Data richness improvement
- Service quality increase

Business Impact:

- Infrastructure scaling costs
- Performance optimization needs
- Storage strategy planning
- User experience considerations
- Technical debt management
- Mitigation Strategies:
- Incremental scaling approach
- Performance monitoring
- Data optimization techniques
- CDN implementation
- Caching strategies

BC5: Compliance & Legal Constraints

BC5.1: Partner Contract Compliance

Constraint: Tuân thủ các điều khoản hợp đồng với nhà xe đối tác

Căn cứ từ đề bài: Implicit từ partnership requirements

Business Rationale:

- Legal obligations
- Partnership maintenance
- Revenue protection
- Risk mitigation

Business Impact:

- SLA monitoring requirements
- Reporting obligations
- Audit trail needs
- Compliance costs
- Legal risk exposure
- Mitigation Strategies:
- Automated compliance monitoring
- Regular audit processes
- Documentation standards
- Legal review procedures
- Partner communication protocols

BC5.2: Financial Transaction Compliance

Constraint: Tuân thủ quy định về giao dịch tài chính

Căn cứ từ đề bài: Implicit từ payment processing

Business Rationale:

- Legal requirements
- USER protection

- Financial security
- Regulatory compliance

Business Impact:

- Security investment requirements
- Audit and reporting costs
- Compliance training needs
- Process documentation
- Risk management overhead
- Mitigation Strategies:
- PCI DSS compliance
- Regular security audits
- Staff training programs
- Process documentation
- Third-party compliance services

BC6: User Experience Constraints

BC6.1: User Experience Standards

Constraint: Trải nghiệm người dùng phải nhanh chóng và thuận tiện

Căn cứ từ đề bài: "nhanh chóng, thuận tiện cho người dùng"

Business Rationale:

- USER satisfaction
- Conversion rate optimization
- Brand reputation
- Competitive advantage

Business Impact:

- UX research investments
- User testing requirements
- Performance optimization costs

- Design iteration needs
- USER feedback systems
- Mitigation Strategies:
- User-centered design approach
- Performance monitoring
- A/B testing programs
- USER feedback collection
- Continuous improvement process

BC6.2: Real-Time Information Accuracy

Constraint: Thông tin phải được cập nhật tức thời cho khách hàng

Căn cứ từ đề bài: "khách hàng phải được thấy tức thời"

Business Rationale:

- USER trust
- Service reliability
- Competitive advantage
- Operational efficiency

Business Impact:

- Real-time system requirements
- Infrastructure complexity
- Monitoring system needs
- Data synchronization costs
- USER expectation management
- Mitigation Strategies:
- Real-time data architecture
- Monitoring and alerting systems
- Fallback mechanisms
- USER communication protocols
- Performance optimization

BC7: Market Constraints

BC7.1: Competitive Response Time

Constraint: Thời gian phản hồi phải c경쟁 với thị trường

Căn cứ từ đề bài: "trả về kết quả nhanh nhất có thể"

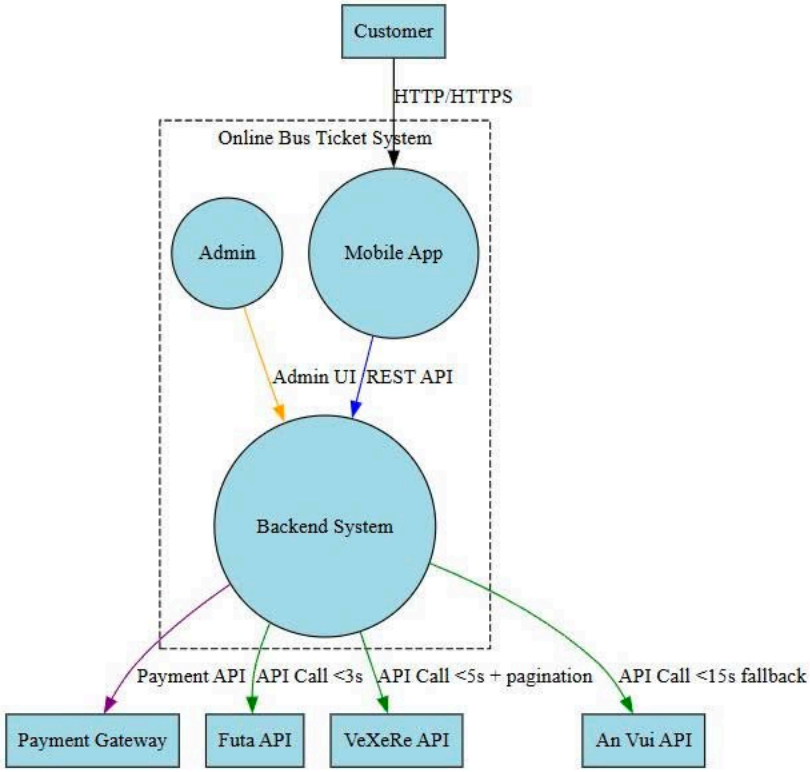
Business Rationale:

- Market competitiveness
- USER retention
- Service differentiation
- Market share protection

Business Impact:

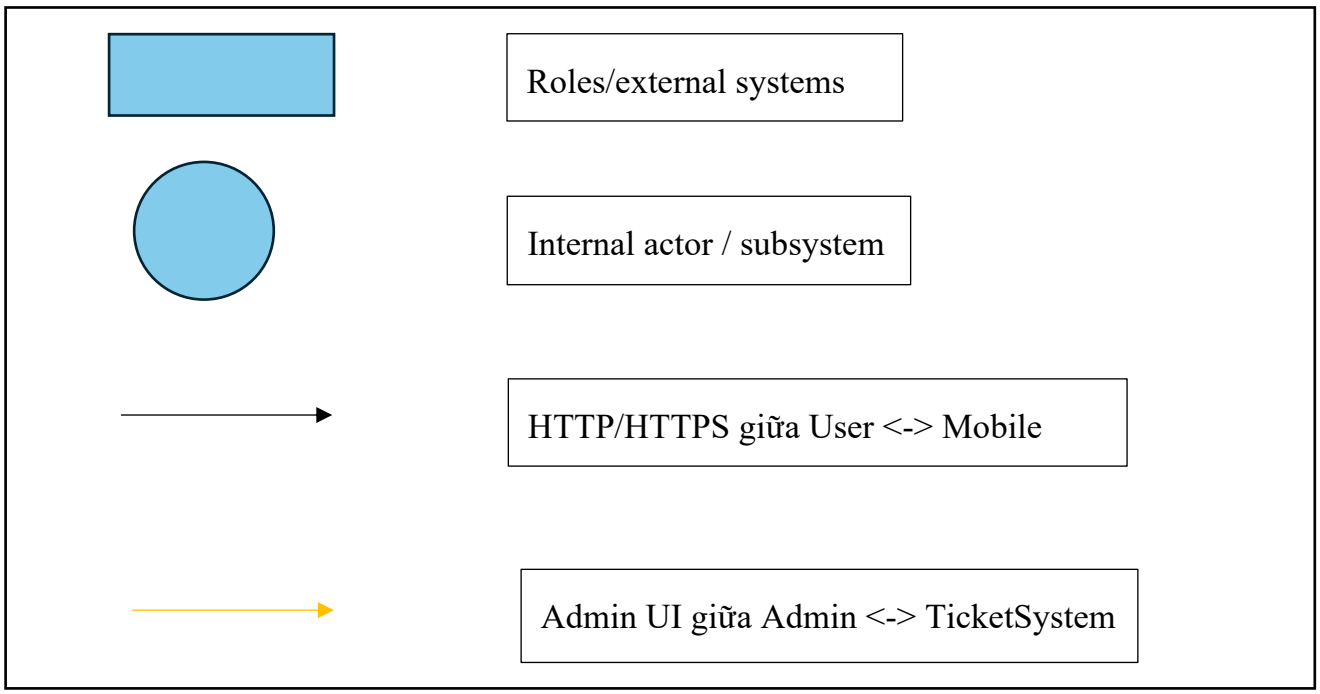
- Performance investment needs
- Technology advancement requirements
- Monitoring and optimization costs
- USER expectation management
- Competitive analysis needs
- Mitigation Strategies:
- Performance benchmarking
- Technology optimization
- Competitive analysis
- USER feedback monitoring
- Continuous improvement

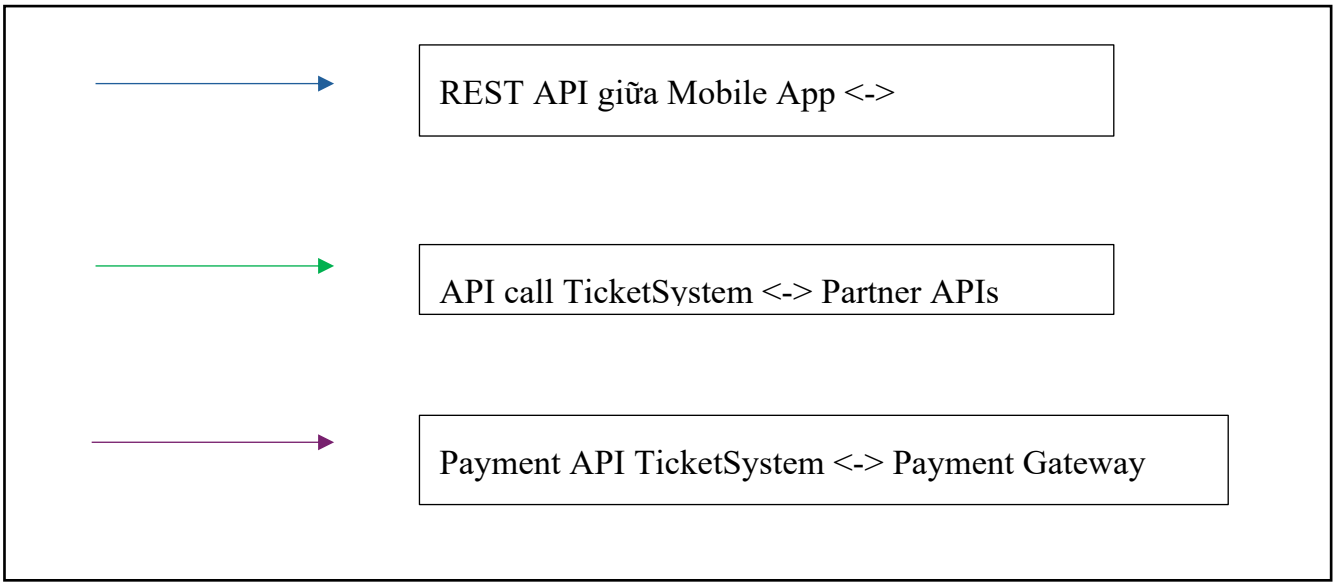
CÂU 6: THIẾT KẾ SƠ ĐỒ NGỮ CẢNH CHO HỆ THỐNG BÁN VÉ XE ONLINE



Hình 1. Sơ đồ ngữ cảnh hệ thống đặt vé xe online

Chú thích

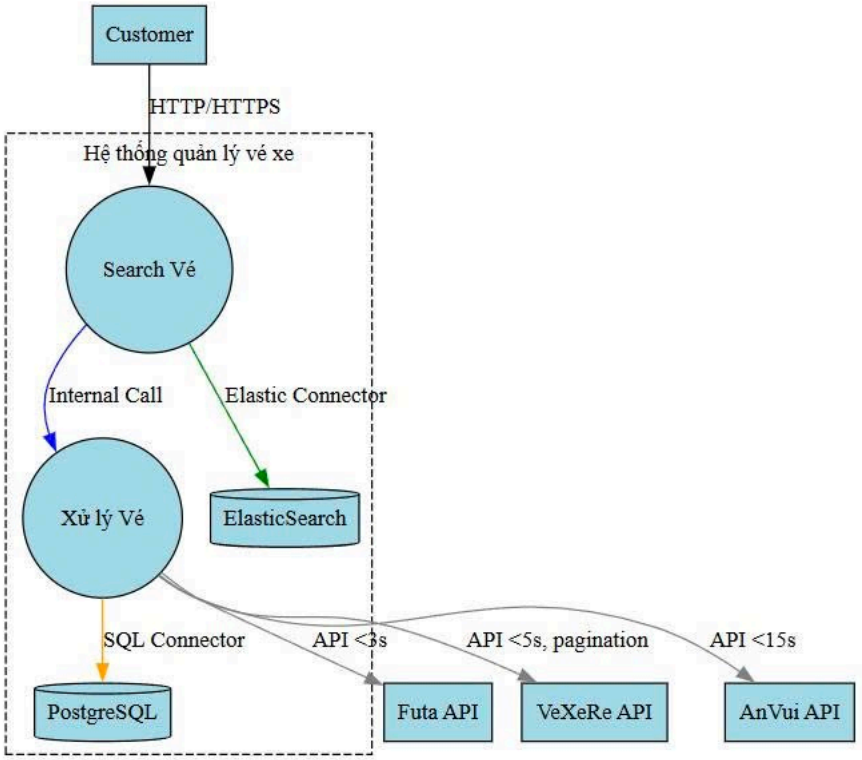



REST API giữa Mobile App <->
API call TicketSystem <-> Partner APIs
Payment API TicketSystem <-> Payment Gateway

CÂU 7: THIẾT KẾ BA PERPECTIVE

7.1 Dynamic Perpective

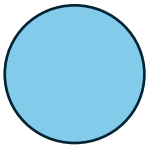
QA1: Performance – Tìm kiếm vé xe



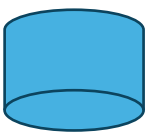
Hình 2. Biểu đồ phân rã chức năng "Tìm kiếm vé" theo thuộc tính Performance

Chú thích

Role, Futa API, VXR API, AnVui API

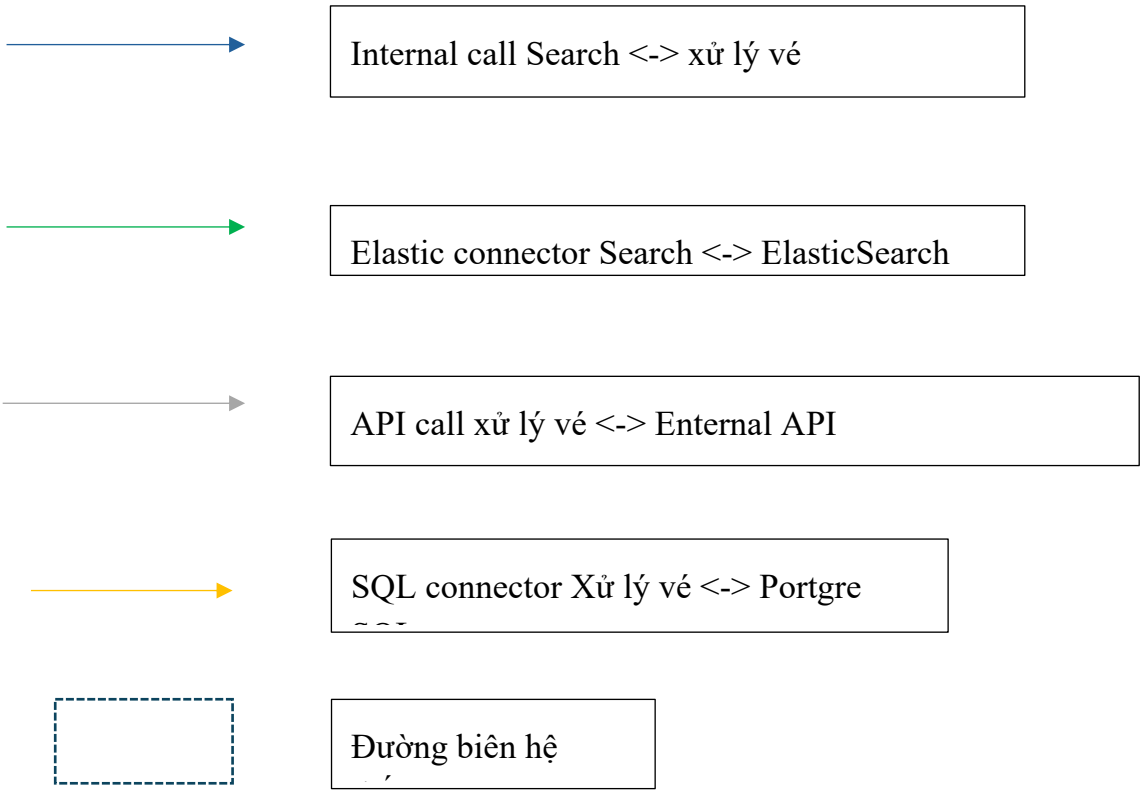


Search Vé, Xử lý Vé



ElasticSearch Postgre

HTTP/HTTPS request-reponse User <-> Search vé


Internal call Search <-> xử lý vé
Elastic connector Search <-> ElasticSearch
API call xử lý vé <-> Enternal API
SQL connector Xử lý vé <-> Portgre
Đường biên hệ

Lý do thiết kế:

- Đạt Performance target:

- Response ≤ 3s (trừ An Vui ≤ 15s).
- 1000+ concurrent users.
- ElasticSearch query cho fast lookup.

- Architectural pattern:

- ElasticSearch + fallback Postgres.
- Microservice separation (Search vé và Xử lý vé).

- Static support:

- Module Search Service tách biệt với Process Service → dễ maintain, easy to scale.

Trách nhiệm từng phần tử:

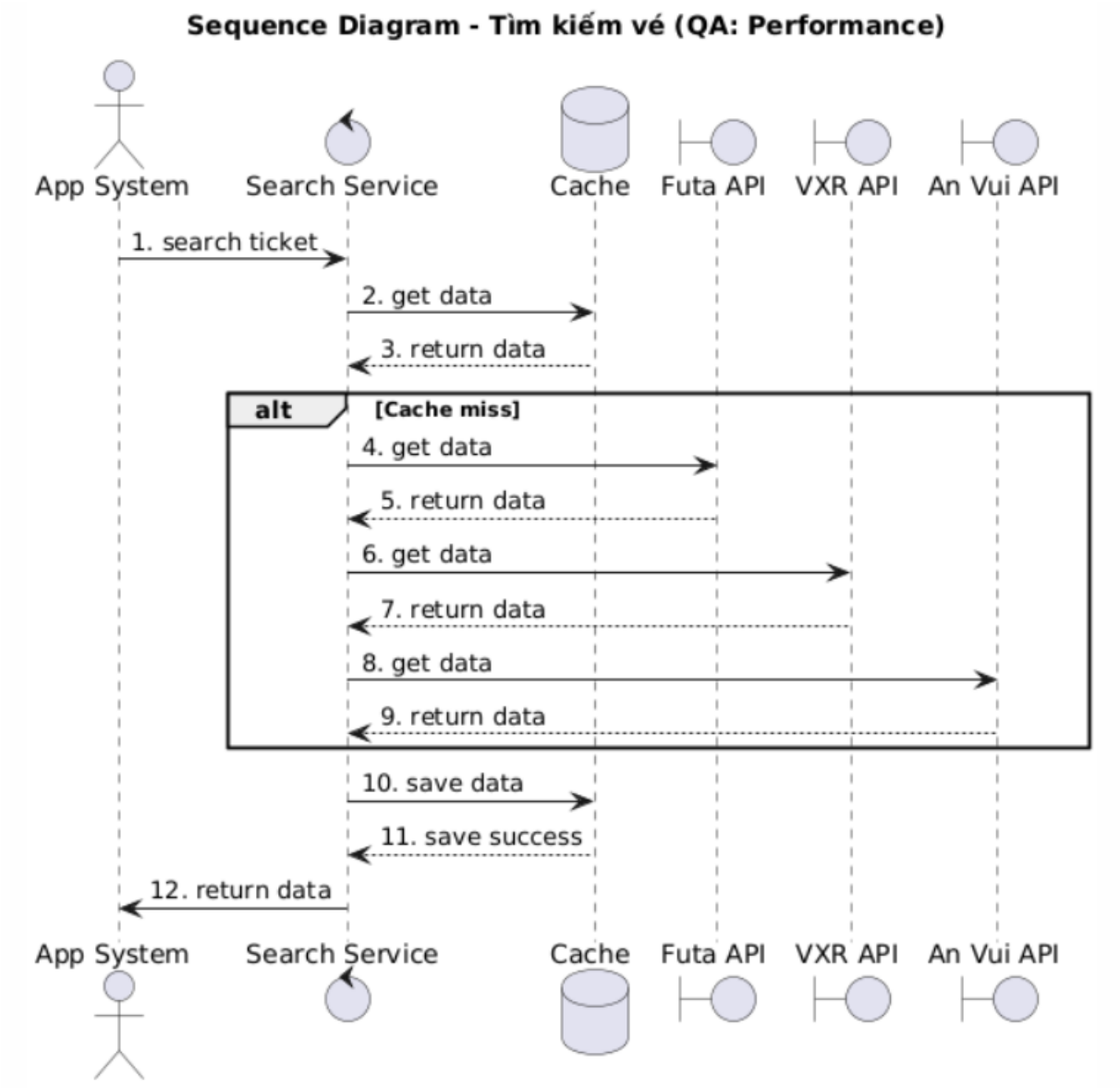
Tên phần tử	Mô tả
Customer	Actor thực hiện search vé.
Search Vé	Service phục vụ search UI request (giao tiếp end-user).
Xử lý Vé	Backend service orchestrate các call external APIs + DB.
ElasticSearch	Fast search index vé theo địa điểm/ngày.
PostgreSQL	Persist vé & booking info.
Futa API	External API cung cấp vé Futa.
VXR API	External API cung cấp vé VeXeRe (có pagination).
AnVui API	External API cung cấp vé An Vui (chậm hơn).

Trách nhiệm mối quan hệ:

Tên mối quan hệ	Mô tả
Customer ↔ Search Vé	HTTP/HTTPS search request/response
Search Vé ↔ ElasticSearch	Elastic Connector query vé

Search Vé ↔ Xử lý Vé	Internal call trigger orchestration
Xử lý Vé ↔ Postgres	SQL connector store/retrieve vé
Xử lý Vé ↔ Futa API	API call (<3s)
Xử lý Vé ↔ VXR API	API call (<5s, pagination)
Xử lý Vé ↔ AnVui API	API call (<15s)

Sequence Diagram cho UC "Search Ticket:



Hình 3. Sequence Diagram cho Use Case: Tìm kiếm vé (UC2.1)

Lý do thiết kế:

- Module hóa theo Domain-driven design:
  - Service boundary rõ ràng → dễ maintain/scale.
  - Backend stateless, dễ deploy distributed.
- API Gateway tách riêng:
  - Quản lý authentication/throttling/routing.

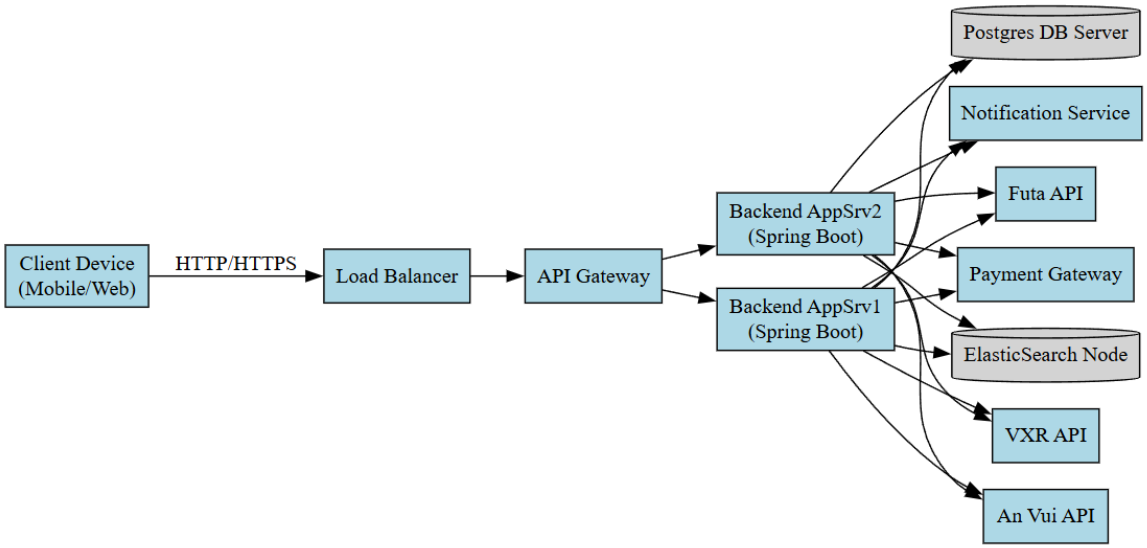
Trách nhiệm từng phần tử (Element Responsibility):

Module	Responsibility
Mobile App UI	Flutter UI cho end-user.
API Gateway	Entry point, auth, route request đến backend service.
Search Service	Xử lý nghiệp vụ tìm kiếm vé, gọi partner API.
Booking Service	Xử lý logic giữ vé, tạo booking record.
Payment Service	Xử lý nghiệp vụ thanh toán online qua payment gateway.
Notification Service	Gửi notification SMS/email/push.
Partner Integration Module	Adapter cho API Futa/VXR/An Vui, abstract detail API từng đối tác.
Admin Management Module	Web UI cho admin thao tác quản trị, monitoring.
Postgres Repository Module	Persistency cho booking, user, location data.
Cache Repository Module	ElasticSearch phục vụ caching search result / location.

Trách nhiệm từng mối quan hệ:

Mối quan hệ	Protocol / Description
Mobile App → API Gateway	HTTP/HTTPS REST API.
API Gateway → Search Service	REST API call, orchestrate search logic.
API Gateway → Booking Service	REST API call, orchestrate booking.
API Gateway → Payment Service	REST API call, payment handling.
Search Service → Partner Integration	Call adapter cho từng API partner.
Services → Postgres Repo Module	JDBC/Postgres SQL connector.
Services → Cache Repo Module	ElasticSearch connector.
Admin UI → API Gateway → Admin Module	HTTP/HTTPS cho admin monitoring/config.

Physical Perspective



Lý do thiết kế:

- Load Balancer (LB):
  - Phân tải cho AppSrv nodes.
  - Tăng tính scalability + availability.
- API Gateway riêng:
  - Quản lý auth, throttling, routing request.

- Backend AppSrv:
    - Stateless, dễ scale-out theo horizontal scaling.
- Postgres DB Server + ElasticSearch:
    - Separation of concern (DB cho transactional data, ES cho caching/querying fast search).

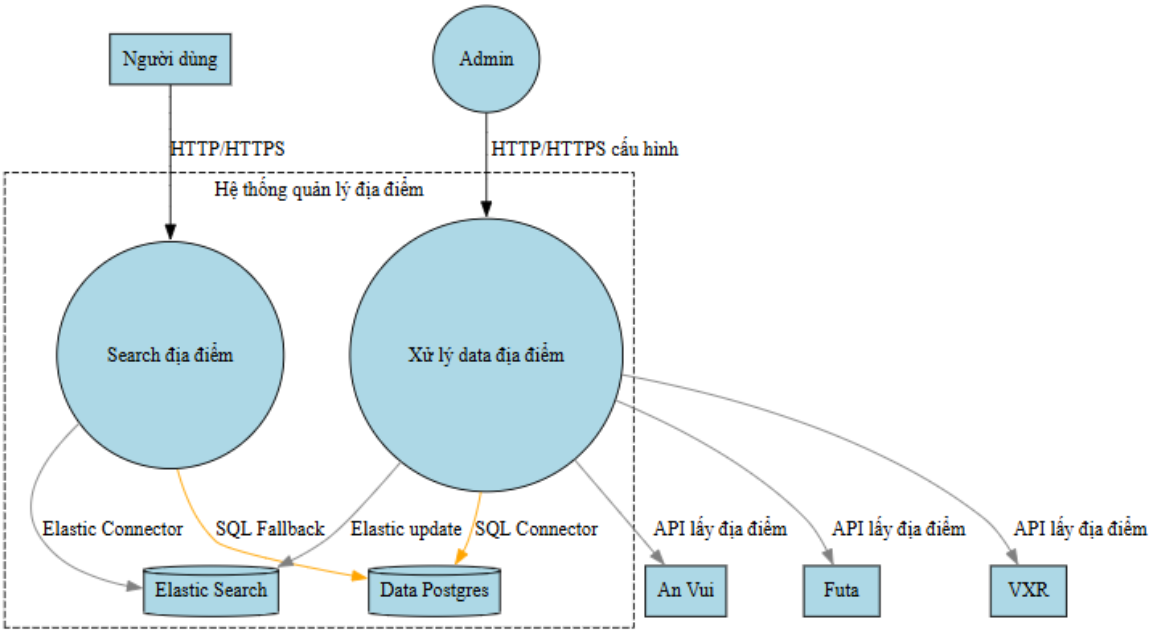
Trách nhiệm từng phần tử:

Physical Node	Responsibility
Client	Device user chạy Flutter app.
Load Balancer	Reverse proxy phân tải HTTP request.
API Gateway	Xử lý auth, route đến backend service đúng endpoint.
AppSrv1, AppSrv2	Backend app instances (Spring Boot) — xử lý nghiệp vụ.
Postgres DB Server	Transactional data storage.
ElasticSearch Node	Fast search cache + indexing cho query địa điểm, vé xe.
External API Systems	Partner API endpoints (Futa, VXR, An Vui, Payment, Notification).

Trách nhiệm mối quan hệ (connector):

Mối quan hệ	Protocol / Description
Client → LB	HTTP/HTTPS requests from end user.
LB → API Gateway	HTTP/HTTPS reverse proxy, load balancing.
API Gateway → AppSrv	Internal REST API call.
AppSrv → Postgres DB	JDBC/Postgres SQL connection.
AppSrv → ElasticSearch Node	ElasticSearch connector.
AppSrv → Partner API	API Call, REST API external partner.
AppSrv → Payment Gateway	API Call, payment handling.
AppSrv → Notification Service	API Call, multi-channel notify.

QA2: Đảm bảo chức năng gợi ý địa điểm luôn luôn hoạt động và cập nhật tức thời khi admin cấu hình dữ liệu mới.



Hình 4. Biểu đồ phân rã chức năng "Tìm kiếm địa điểm" theo thuộc tính Real-time Update

Chú thích:

Lý do thiết kế

- Đảm bảo Availability:
  - Tách Search địa điểm ra thành 1 service độc lập với chức năng khác.

- ElasticSearch dùng riêng cho search → không bị DB lock/block.

- Đảm bảo cập nhật tức thời:
  - Khi Xử lý địa điểm lấy API từ nhà xe → update cả Elastic + DB.
  - Không dùng batch → update near real-time.
- Đảm bảo scalable:
  - Dữ liệu chỉ 1–5MB, có thể chứa full trong bộ nhớ ElasticSearch.
  - Tối ưu search qua index.

Trách nhiệm từng phần tử

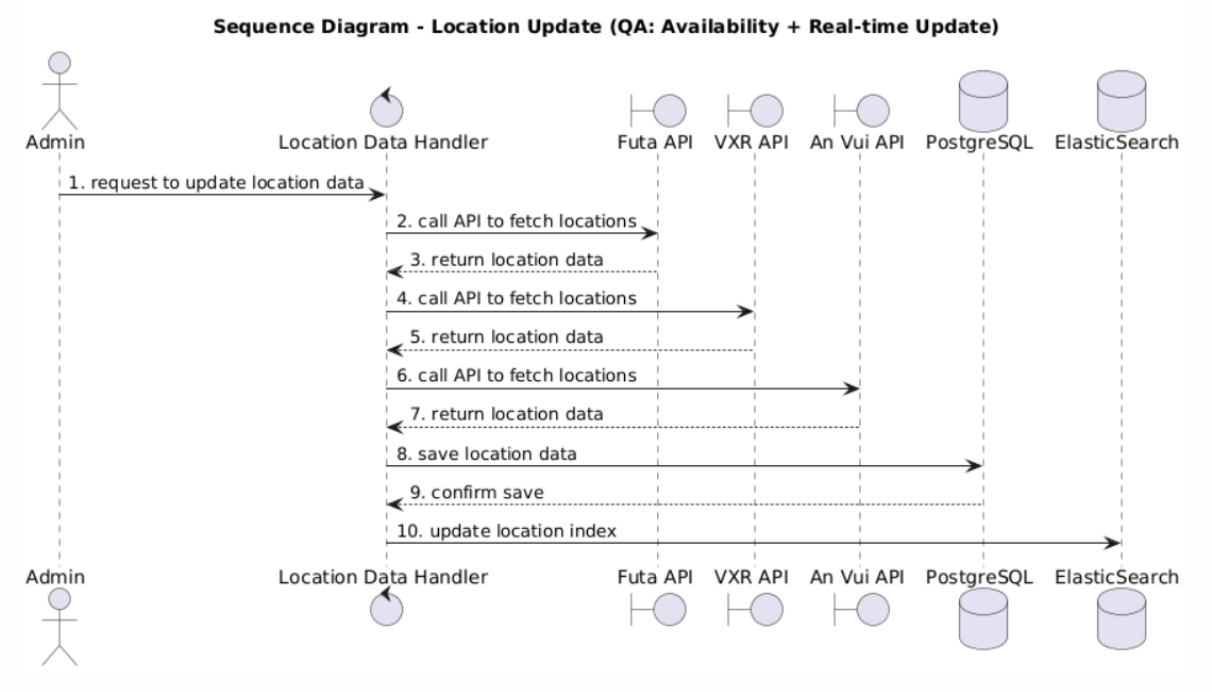
Tên phần tử	Mô tả
Người dùng	Actor chính sử dụng chức năng gợi ý địa điểm
Admin	Người cấu hình, cập nhật địa điểm từ API
Search địa điểm	Nhận input keyword từ user, query ElasticSearch
Xử lý data địa điểm	Giao tiếp API nhà xe, chuẩn hóa, cập nhật DB + Elastic
ElasticSearch	Nơi index toàn bộ địa điểm để search cực nhanh
Postgres	Lưu bản ghi chính thức địa điểm (source of truth)
Futa/VXR/An Vui	Các nhà xe cung cấp API trả về danh sách địa điểm

Trách nhiệm từng mối quan hệ

Mối quan hệ	Mô tả
Người dùng → Search	Gửi keyword để search
Search → Elastic	Truy vấn index tìm địa điểm
Search → Postgres	Nếu không có trong cache → fallback query
Admin → Xử lý	Gửi yêu cầu cập nhật địa điểm

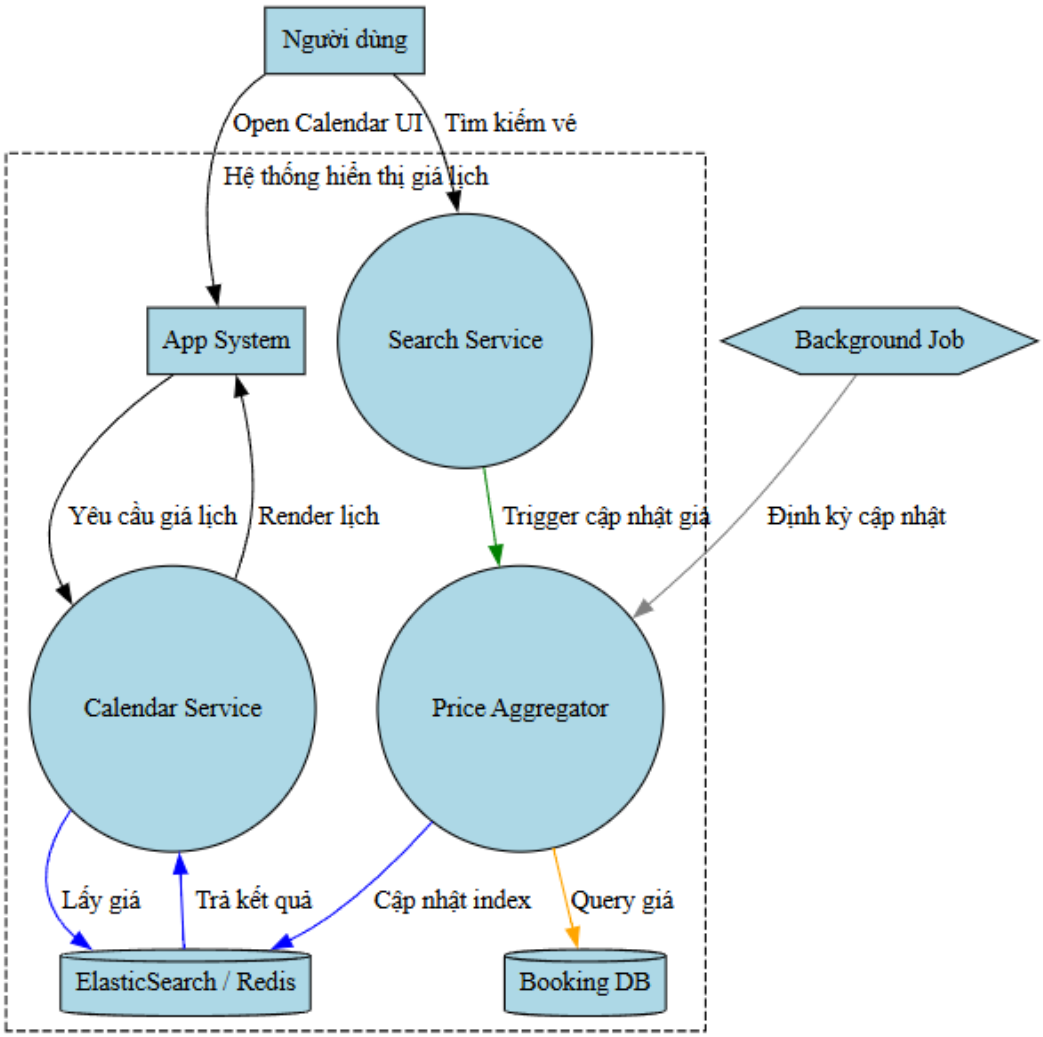
Xử lý → Futa/VXR/An Vui	Gọi API lấy danh sách địa điểm
Xử lý → Postgres	Lưu bản ghi chuẩn
Xử lý → Elastic	Update lại index real-time

Sequence Diagram



Hình 5. Sequence Diagram cho Use Case: Hiển thị lịch giá rẻ (UC3.1)

QA3: Đảm bảo việc hiển thị lịch giá rẻ diễn ra nhanh, không làm gián đoạn trải nghiệm người dùng và luôn phản ánh dữ liệu gần đúng với giá thực tế.



Hình 6. Biểu đồ phân rã chức năng "Hiển thị lịch giá" theo thuộc tính Usability & Performance

Chú thích:

Lý do thiết kế kiến trúc

- Đạt:

- Hiệu năng (Performance):
    - Dùng cache (ElasticSearch/Redis) để hiển thị giá nhanh.
    - Không truy vấn trực tiếp DB mỗi lần người dùng mở calendar.
- Tươi mới dữ liệu (Freshness):

- Có 2 cơ chế cập nhật:
    - Tự động sau mỗi lần người dùng search vé
    - Background job định kỳ (6h/lần)
- Tách biệt chức năng (Loose Coupling):
    - Calendar Service hoạt động độc lập → không ảnh hưởng Search

- Chưa đạt:

- Không đảm bảo luôn đúng giá "thực tế" vì phụ thuộc độ trễ sync từ booking DB.
- Nếu job delay hoặc search không có → có thể dữ liệu bị trễ vài giờ.

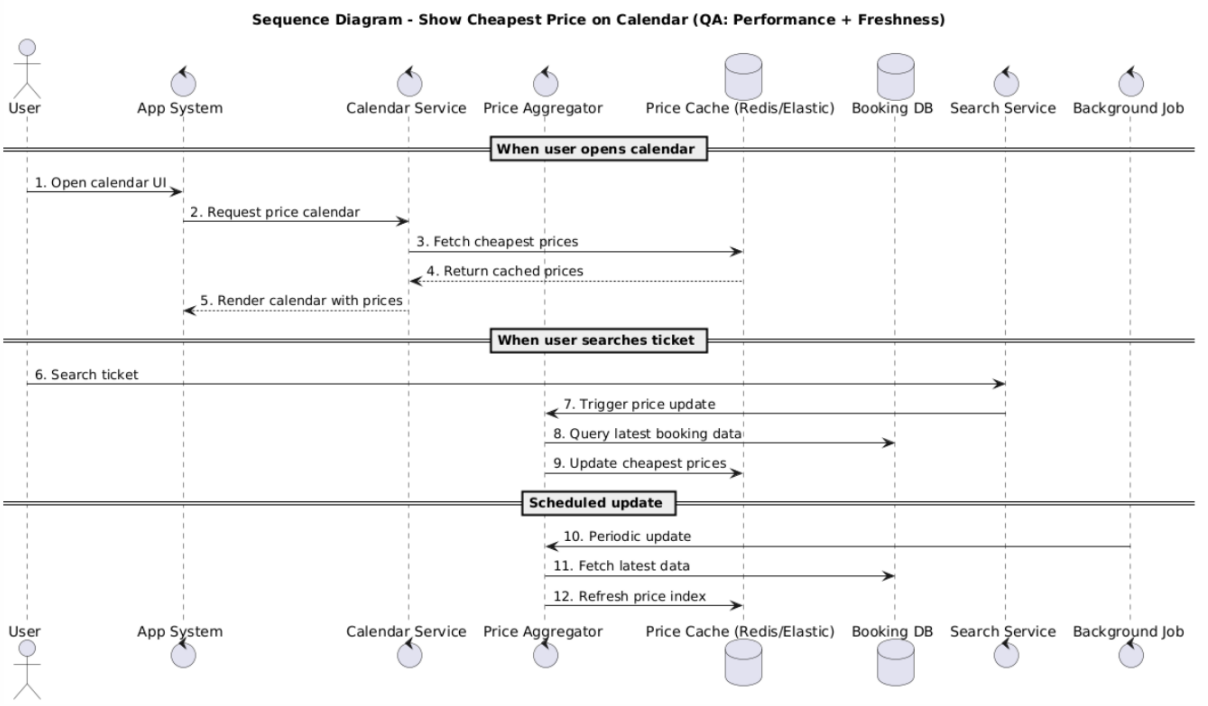
Trách nhiệm phần tử:

Thành phần	Trách nhiệm chính
App System	Hiển thị UI calendar và gọi Calendar Service
Calendar Service	Nhận request và trả về lịch giá (cache)
Price Aggregator	Tính toán giá rẻ nhất từ nhiều nguồn → ghi cache
Cache	Lưu giá rẻ nhất dạng index (hiển thị nhanh)
Booking DB	Lưu thông tin vé và giá thực tế đã được người dùng booking
Search Service	Gửi trigger update khi có tìm vé mới
Background Job	Cập nhật giá tự động theo lịch định sẵn

Trách nhiệm từng mối quan hệ

Mối quan hệ	Mô tả giao tiếp
App → Calendar Service	Gửi yêu cầu hiển thị lịch
Calendar → Cache	Truy vấn nhanh theo tuyến – ngày
Search → Aggregator	Khi user tìm vé, hệ thống trigger cập nhật giá vào cache
Aggregator → DB	Lấy lịch sử giá thực tế gần nhất
Aggregator → Cache	Ghi lại giá rẻ nhất (per route, per day)

Job → Aggregator	Tự động cập nhật mà không cần tương tác người dùng

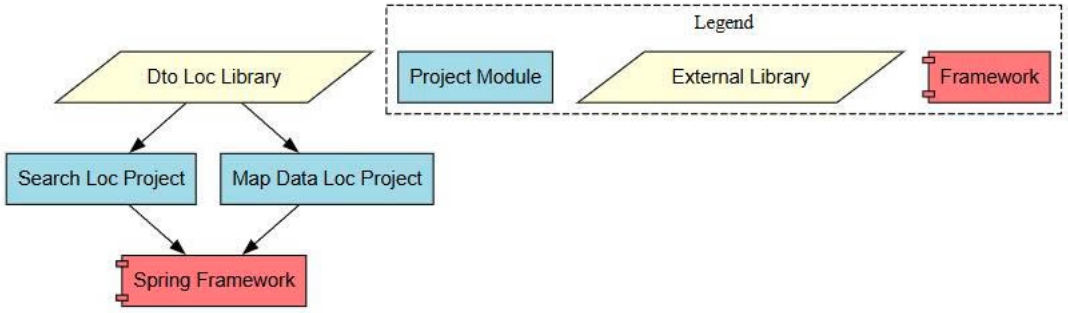


7.2 Static View

QA1: Chọn địa điểm

- QA: Khi thay đổi chức năng xử lý map địa điểm hay có địa điểm mới thì không ảnh hưởng đến việc tìm kiếm địa điểm của khách hàng. Thời gian thêm mới hoặc thay đổi phải tốn chi phí thấp nhất và cho ra đường nhanh nhất có thể.

Bản vẽ:



Chú thích:

Ký hiệu	Ý nghĩa
Hình ngũ giác	Module/Project
Hình lục giác	Library/Framework ngoài
A → B	A phụ thuộc B (A dùng B)

Lý do thiết kế:

- Tách module độc lập giúp:
  - Dễ thêm/tích hợp địa điểm mới từ nhà xe
  - Không ảnh hưởng search logic
  - DTO riêng dùng chung → giảm phụ thuộc
  - Spring framework hỗ trợ dependency injection

Trách nhiệm từng phần tử:

Tên phần tử	Mô tả
Dto loc lib	Khai báo cấu trúc dữ liệu địa điểm chung
Search loc project	Xử lý tìm kiếm địa điểm của người dùng
Map data loc	Giao tiếp lấy danh sách địa điểm từ đối tác (Futa, An Vui, VXR)
Spring framework	Thư viện hỗ trợ dependency & wiring các module

Trách nhiệm mối quan hệ:

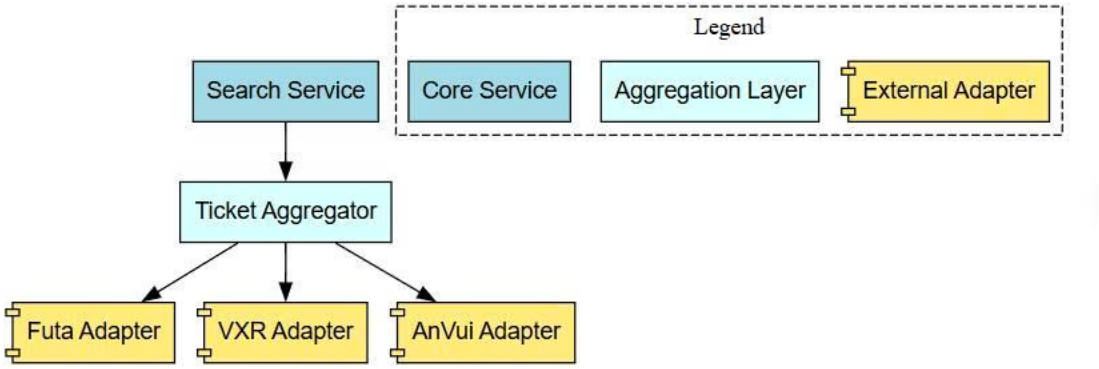
Tên quan hệ	Mô tả

A uses B	Module A phụ thuộc thư viện hoặc project B

QA2: Tìm kiếm vé

QA: Khi tích hợp thêm nhà xe (Futa, Vexere, An Vui…) không làm ảnh hưởng đến chức năng tìm kiếm vé hiện tại. Việc tích hợp nên không cần sửa code cũ.

Bản vẽ:



Chú thích:

Hình chữ nhật	Module logic
Adapter	Giao tiếp từng nhà xe
A → B	Gọi adapter động qua interface

Lý do thiết kế:

- Dùng pattern plugin/module adapter:
  - Thêm nhà xe mới → tạo adapter mới → không đụng search
  - Interface chuẩn hóa giữa Aggregator ↔ Adapter
  - Dễ mở rộng, dễ test độc lập

Trách nhiệm từng phần tử:

Tên phần tử	Mô tả

Search Service	Xử lý logic search UI, nhận yêu cầu từ người dùng
Ticket Aggregator	Tổng hợp vé từ nhiều nguồn adapter
FutaAdapter	Giao tiếp API Futa
VXRAdapter	Giao tiếp VXR
AnVuiAdapter	Giao tiếp An Vui

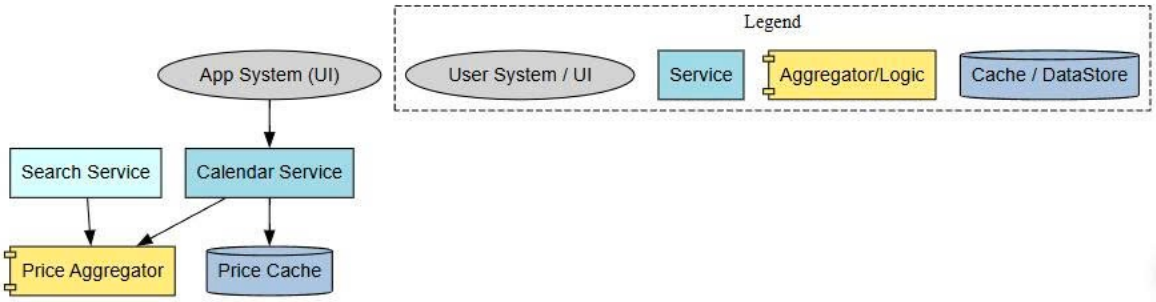
Trách nhiệm mối quan hệ:

Tên quan hệ	Mô tả
A uses B	Aggregator gọi adapter theo interface runtime

QA3: Hiển thị giá trên lịch

QA: Khi muốn thay đổi logic hiển thị lịch giá (ví dụ thay đổi tính toán, thay đổi cache) thì không ảnh hưởng UI hoặc logic tìm kiếm vé.

Bản vẽ:



Chú thích:

Hình chữ nhật	Module chính logic
Hình trụ	Cache hoặc hệ thống lưu giá rẻ nhất
A → B	Module gọi nhau

Lý do thiết kế:

- Dùng cấu trúc tách biệt:

- Calendar tách khỏi search
- Price aggregator có thể tái sử dụng cho search lẫn lịch
- Cache trung gian giúp tăng tốc độ truy cập
- Thay đổi trong logic tính toán không ảnh hưởng App hoặc Search

Trách nhiệm từng phần tử:

Tên phần tử	Mô tả
App System	Giao diện người dùng, hiển thị lịch giá
Calendar Service	Lấy giá rẻ nhất từ cache và format theo lịch
Price Aggregator	Tính toán giá rẻ nhất từ lịch sử booking
Price Cache	Lưu cache giá cho mỗi tuyến/ngày
Search Service	Kích hoạt cập nhật giá khi user search vé

Trách nhiệm mối quan hệ:

Tên quan hệ	Mô tả
A uses B	Calendar → Cache, Aggregator → DB + Cache

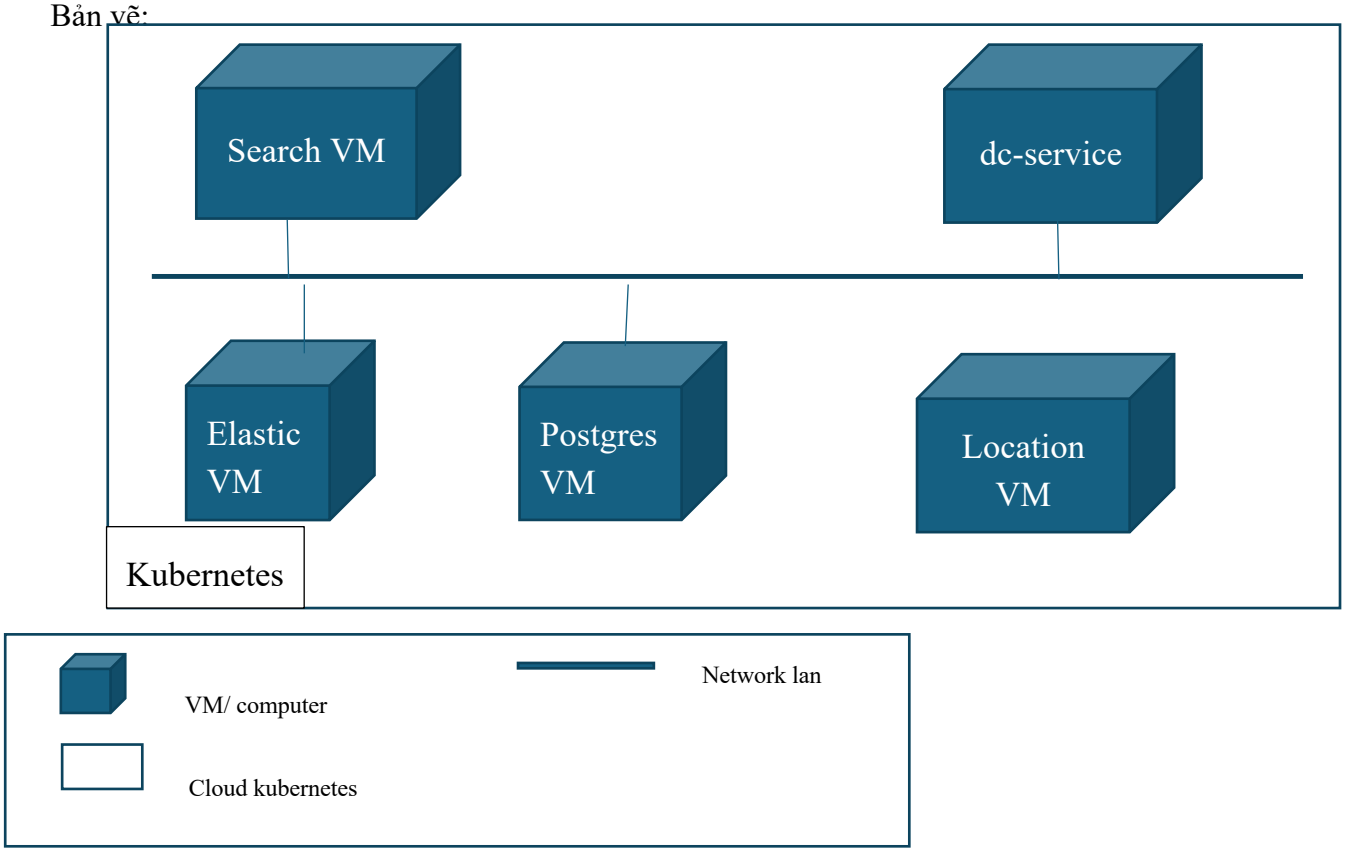
7.3 Physical View

QA1: Chọn Địa Điểm

QA1: Hệ thống phải đảm bảo việc cập nhật địa điểm mới không làm gián đoạn tìm kiếm địa điểm, và cần được triển khai tự động, nhanh chóng (≤ 5s), có khả năng scale tốt.

Bản vẽ:

Bản vẽ:



Lý do thiết kế:

- Tách module dc-service giúp cập nhật địa điểm mà không ảnh hưởng Search VM
- Dùng Kubernetes để tự động scale dc-service khi có nhiều điểm cần cập nhật
- Ghi log cập nhật địa điểm về Elastic VM
- Dữ liệu được ghi xuống Postgres VM dùng chung cho cả Search và Location

Trách nhiệm từng phần tử:

Tên phần tử	Mô tả
Search VM	Xử lý truy vấn địa điểm từ phía người dùng; truy vấn vào DB để lấy danh sách gợi ý, kết quả tìm kiếm

dc-service	Xử lý cập nhật địa điểm mới từ các plugin, đồng bộ dữ liệu xuống Postgres; có thể hot-deploy, autoscale không ảnh hưởng Search
Location VM	Cung cấp API danh sách địa điểm hiện tại cho các dịch vụ khác như đặt vé, chọn chỗ ngồi
Elastic VM	Ghi log cập nhật địa điểm, hỗ trợ tracking, logging, debugging các vấn đề dữ liệu
Postgres VM	Lưu trữ danh sách địa điểm chính thức; được truy cập bởi cả Search VM và Location VM

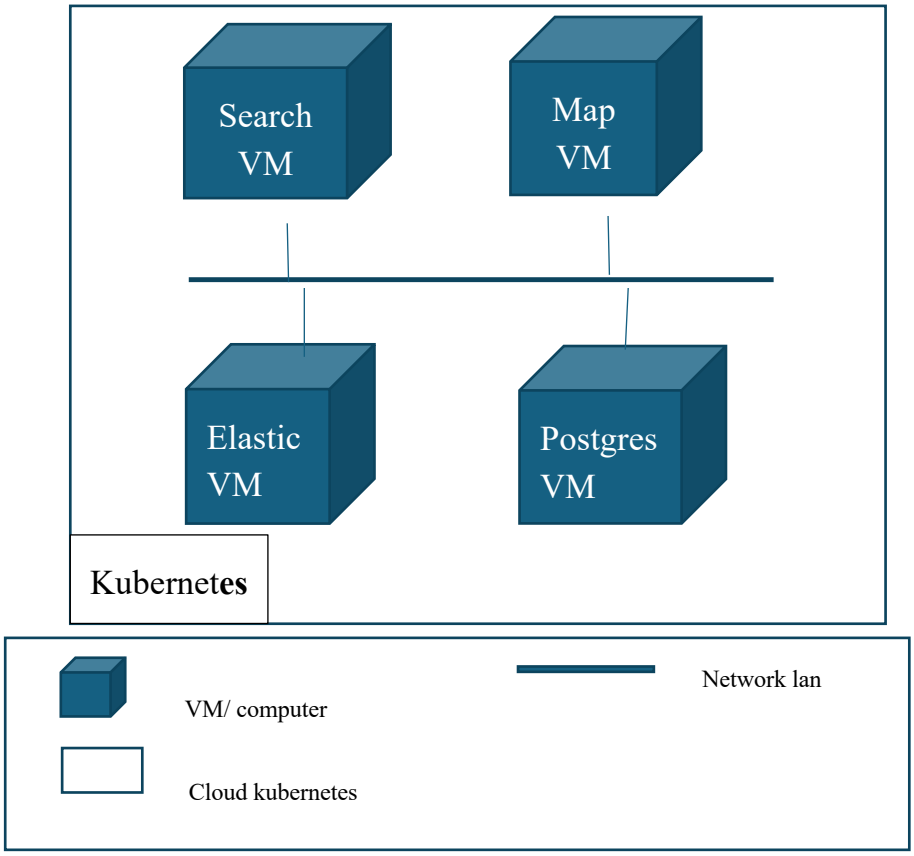
Giải thích mối quan hệ:

Tên quan hệ	**Mô tả**
dc → Postgres	Ghi địa điểm mới
dc → Elastic	Ghi log cập nhật
dc → Search	Gửi tín hiệu sync cache (nếu có)
CI/CD → dc VM	Tự động build và deploy khi thêm plugin địa điểm

QA2: Tìm Kiếm Vé

QA2: Khi tích hợp thêm nhà xe mới (plugin mới), hệ thống phải deploy module mới < 5s, không ảnh hưởng đến các nhà xe hiện tại.

Bản vẽ:



Lý do thiết kế:

- Mỗi adapter là một container riêng biệt → có thể hot-deploy vào cluster.
- Aggregator nằm trong Search VM, gọi các adapter qua network nội bộ.
- Adapter có thể scale riêng nếu một nhà xe quá tải (VD: Futa giờ cao điểm).
- Cấu trúc phù hợp với kiến trúc plugin.

Trách nhiệm từng phần tử:

Tên phần tử	Mô tả
Search VM	Xử lý tìm kiếm vé
Map VM	Xử lý hiển thị bản đồ
Adapter-VXR VM	Tích hợp plugin nhà xe VXR
Adapter-Futa VM	Tích hợp plugin nhà xe FUTA

Giải thích mối quan hệ:

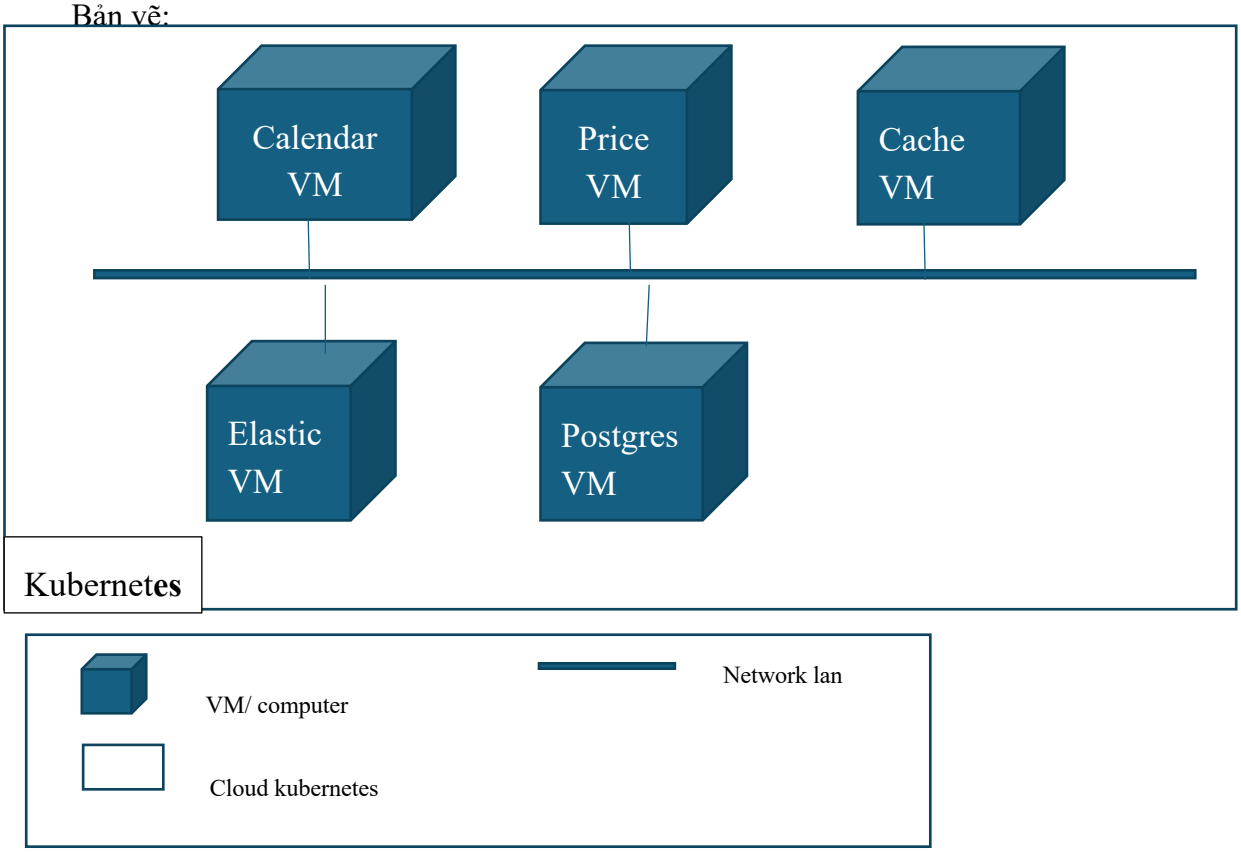
Tên quan hệ	Mô tả
Network lan	Kết nối nội bộ giữa các VM, đảm bảo low latency

Kubernetes	Quản lý tự động container và hỗ trợ autoscaling

QA3: Hiển thị giá rẻ trên lịch

QA3: Cập nhật giá rẻ trên lịch phải tự động, luôn có giá mới nhất. Hệ thống phải deploy lại các thay đổi trong lịch giá mà không làm gián đoạn UI.

Bản vẽ:



Lý do thiết kế:

- Tách riêng Price VM để xử lý logic tính toán giá rẻ → không ảnh hưởng UI.
- Calendar VM chỉ hiển thị dữ liệu từ cache → đảm bảo không gián đoạn UI khi backend thay đổi.
- Cache VM giúp lấy dữ liệu giá rẻ nhanh chóng, hỗ trợ real-time / near real-time.
- Dùng Postgres VM để lưu dữ liệu giá vé gốc từ nhà xe → phục vụ cho việc tái tính giá rẻ.
- Elastic VM ghi lại log thay đổi giá phục vụ giám sát, phân tích.
- Dùng Kubernetes để scale Price VM theo lưu lượng truy cập hoặc theo batch cập nhật giá.
- Triển khai thêm 2 Cache VM để hỗ trợ high - availability và load balancing.

Trách nhiệm từng phần tử:

Tên phần tử	Mô tả
Calendar VM	Giao diện frontend hiển thị lịch và giá rẻ mỗi ngày. Không reload khi dữ liệu thay đổi.
Price VM	Dịch vụ xử lý logic tính toán giá rẻ dựa vào dữ liệu giá gốc. Kết quả được đẩy vào cache.
Cache VM	Lưu giá rẻ đã tính toán để trả về cho frontend. Dùng Redis hoặc Memcached. Có thể có nhiều instance để scale.
Elastic VM	Ghi log các lần cập nhật giá, hỗ trợ tracking, debugging hoặc phân tích lịch sử thay đổi giá.
Postgres VM	Lưu trữ toàn bộ giá vé gốc từ các nhà xe (nguồn raw data để tính giá rẻ).

Giải thích mối quan hệ:

Tên quan hệ	Mô tả
Network lan	Kết nối nội bộ tốc độ cao giữa các VM trong cùng một cluster
Calendar → Cache	Calendar VM gọi API hoặc đọc trực tiếp từ Cache VM để lấy giá rẻ.
Price → Postgres	Price VM truy vấn giá gốc từ Postgres để tính giá rẻ.
Price → Cache	Price VM đẩy giá rẻ đã tính vào Cache để frontend sử dụng.
Price → Elastic	Log quá trình cập nhật hoặc tính giá vào Elastic VM.

CÂU 9: THIẾT KẾ 2 MAPPING VIEW

9.1 Mapping từ Static → Dynamic

Static Element	Dynamic Service

Map Data Loc, Dto Loc Lib, Search Loc Project	Search-Location Service
Ticket Aggregator, các Adapter (Futa, VXR, An Vui)	Search-Ticket Service
Price Aggregator, Price Cache, Calendar Module	Calendar Service
Booking, Payment, Notification libs	Booking Service
Integration Gateway	Adapter-Hub Service
Auth, User Profile	Identity Service
Shared Infra Libs (Spring, Elastic, Redis, Postgres)	Infra-Runtime

9.2 Mapping từ Dynamic → Physical

Dynamic Service	**Physical Deployment (VM / Node)**
Postgres Cluster	pg-vm-[primary/replica]
Search-Ticket Svc, Search-Location Svc	search-vm-pool (K8s deployment, HPA)
Calendar Svc	calendar-vm
Price Aggregator	price-vm
Adapter-Futa / VXR / An Vui	adapter-futa-vm, ...
Identity Svc	auth-vm
Elastic Cluster	elastic-vm-[1-3]
Redis Cluster	cache-vm-[1-n]

CÂU 10: REQUIREMENT TRACEABILITY MATRIX

Requirement satisfied	**Design decision**	**Element**	**Relationship**	**Design artifact reference**	**Comment**
QA1 – Performance (Tìm vé ≤ 5 s)	Parallel query + Redis cache	Search-Ticket Svc, Adapter-Hub, Redis	Search-Ticket → Cache → Adapter	Hình 2.1.1 (Dynamic UC2.1)	Cache hit > 80 %

QA2 – Real-time location update	Event-driven indexing	Map-Sync Svc, Elastic, Kafka	Event → Map-Sync → Elastic	Hình 2.2.1	SLA cập nhật ≤ 1.5 s
QA3 – Calendar price freshness	Scheduled + on-demand recompute	Price Aggregator, Calendar Svc, Redis	Job/Search → Aggregator → Cache	Hình 2.3.1	Δ giá ≤ 15 phút
UC2.1 – Search one-way ticket	Aggregator pattern	Search-Ticket Svc, Adapter set	Aggregator calls Adapter*	Seq-Diagram 88	Isolate partner SLA
UC3.1 – Show price calendar	CQRS read-model	Calendar Svc, Redis	Calendar reads cache	Static View QA3	No DB lock
UC4.4 – Payment	External gateway abstraction	Payment Adapter, Booking Svc	Booking ↔ Payment GW	Dynamic UC4	PCI-DSS scope shrink
QA6 – Partner API failure isolation	Circuit-breaker + timeout	Adapter pods, Hystrix/K8s CB	Adapter ↔ Partner	Fault-Tolerant view	Degrade ≤ 33 %
QA4 – 24/7 availability (99.9 %)	N-Way active, rolling deploy	K8s cluster, LB, Canary	LB → svc v1/v2	Physical View	MTTR < 15'
BC1 – 6-month time-to-market	Domain slicing, MVP waves	All services	Sprint roadmap → Arch	Gantt	Scope re-prioritise