

BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN
ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

QUẢN TRỊ CƠ SỞ DỮ LIỆU HIỆN ĐẠI



Nhóm thực hiện: QTCSDLHD-04

GV phụ trách: GV. Nguyễn Trần Minh Thứ – GV. Phạm Minh Tú – GV. Tiết Gia Hồng

Quản trị cơ sở dữ liệu hiện đại
HỌC KỲ II – NĂM HỌC 2023-2024

BẢNG THÔNG TIN CHI TIẾT NHÓM

Mã nhóm:	QTCSDLHD-04	
Tên nhóm:	QTCSDLHD-04	
Số lượng:	5	
MSSV	Họ tên	Email
18120564	Lâm Hồng Thành	18120564@student.hcmus.edu.vn
20120168	Quách Bảo Quân	20120168@student.hcmus.edu.vn
20120187	Nguyễn Viết Thái	20120187@student.hcmus.edu.vn
20120552	Văn Đức Phước	20120552@student.hcmus.edu.vn
20120575	Nguyễn Khắc Tấn	20120575@student.hcmus.edu.vn

Bảng phân công & đánh giá hoàn thành công việc			
Công việc thực hiện	Người thực hiện	Mức độ hoàn thành	Đánh giá của nhóm
<ul style="list-style-type: none"> - Mô tả quy trình nghiệp vụ cho hệ thống - Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy xuất hệ thống của nghiệp vụ tra cứu hóa đơn vé - Phân tích ưu nhược điểm của từng loại NoSQL database - Thiết kế cơ sở dữ liệu cho nghiệp vụ tra cứu hóa đơn vé - Viết tài liệu 	Lâm Hồng Thành	10/10	10/10
<ul style="list-style-type: none"> - Mô tả quy trình nghiệp vụ cho hệ thống - Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy xuất hệ thống của nghiệp vụ tìm kiếm chuyển đi theo bộ lọc - Phân tích ưu nhược điểm của từng loại NoSQL database 	Quách Bảo Quân	10/10	10/10

<ul style="list-style-type: none"> - Thiết kế cơ sở dữ liệu cho nghiệp vụ tìm kiếm chuyển đi theo bộ lọc - Viết tài liệu 			
<ul style="list-style-type: none"> - Mô tả quy trình nghiệp vụ cho hệ thống - Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy xuất hệ thống của nghiệp vụ thanh toán - Phân tích ưu nhược điểm của từng loại NoSQL database - Thiết kế cơ sở dữ liệu cho nghiệp vụ thanh toán - Viết tài liệu 	Nguyễn Viết Thái	10/10	10/10
<ul style="list-style-type: none"> - Mô tả quy trình nghiệp vụ cho hệ thống - Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy xuất hệ thống của nghiệp vụ tìm kiếm chuyển đi - Phân tích ưu nhược điểm của từng loại NoSQL database - Thiết kế cơ sở dữ liệu cho nghiệp vụ tìm kiếm chuyển đi - Viết tài liệu 	Văn Đức Phước	10/10	10/10
<ul style="list-style-type: none"> - Mô tả quy trình nghiệp vụ cho hệ thống - Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy xuất hệ thống của nghiệp vụ đặt vé - Phân tích ưu nhược điểm của từng loại NoSQL database - Thiết kế cơ sở dữ liệu cho nghiệp vụ đặt vé - Viết tài liệu 	Nguyễn Khắc Tấn	10/10	10/10

Mục Lục

A.	Yêu cầu của Đồ án.....	4
B.	Kết quả.....	5
I.	Mô tả quy trình nghiệp vụ	5
1.	Quy trình tìm chuyến	5
2.	Quy trình tìm kiếm theo bộ lọc.....	7
3.	Quy trình đặt vé	9
4.	Quy trình thanh toán	11
5.	Quy trình tra cứu hóa đơn	13
II.	Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy suất hệ thống.....	15
1.	Tìm kiếm chuyển đi	15
2.	Tìm kiếm chuyển đi theo bộ lọc	16
3.	Đặt vé.....	16
4.	Thanh toán hoá đơn	17
5.	Tra cứu hoá đơn/vé	17
III.	Ưu nhược.....	17
IV.	Lựa chọn và thiết kế cơ sở dữ liệu	19
1.	Nghiệp vụ tìm kiếm chuyển xe.....	19
2.	Nghiệp vụ tìm kiếm chuyển xe theo bộ lọc.....	20
3.	Nghiệp vụ đặt vé.....	21
4.	Nghiệp vụ thanh toán vé.....	24
5.	Nghiệp vụ tìm kiếm hóa đơn, vé	25

YÊU CẦU ĐỒ ÁN

Loại bài tập	<input checked="" type="checkbox"/> Lý thuyết <input type="checkbox"/> Thực hành <input type="checkbox"/> Đồ án <input checked="" type="checkbox"/> Bài tập
Ngày bắt đầu	01/06/2024
Ngày kết thúc	13/06/2024

A. Yêu cầu của Đồ án

Sinh viên khảo sát quy trình tìm chuyến, đặt vé và thanh toán của hãng xe Phương Trang (<https://futabus.vn>):

- o Mô tả lại các quy trình nghiệp vụ của hệ thống Futabus xác định và mô tả các chức năng sẽ xây dựng trong hệ thống.

- o Phân tích các yêu cầu lưu trữ, khả năng mở rộng và nhu cầu truy xuất của các chức năng đã mô tả trong hệ thống.

- o Lập bảng phân tích ưu nhược điểm của MongoDB, Redis, Cassandra và Neo4J

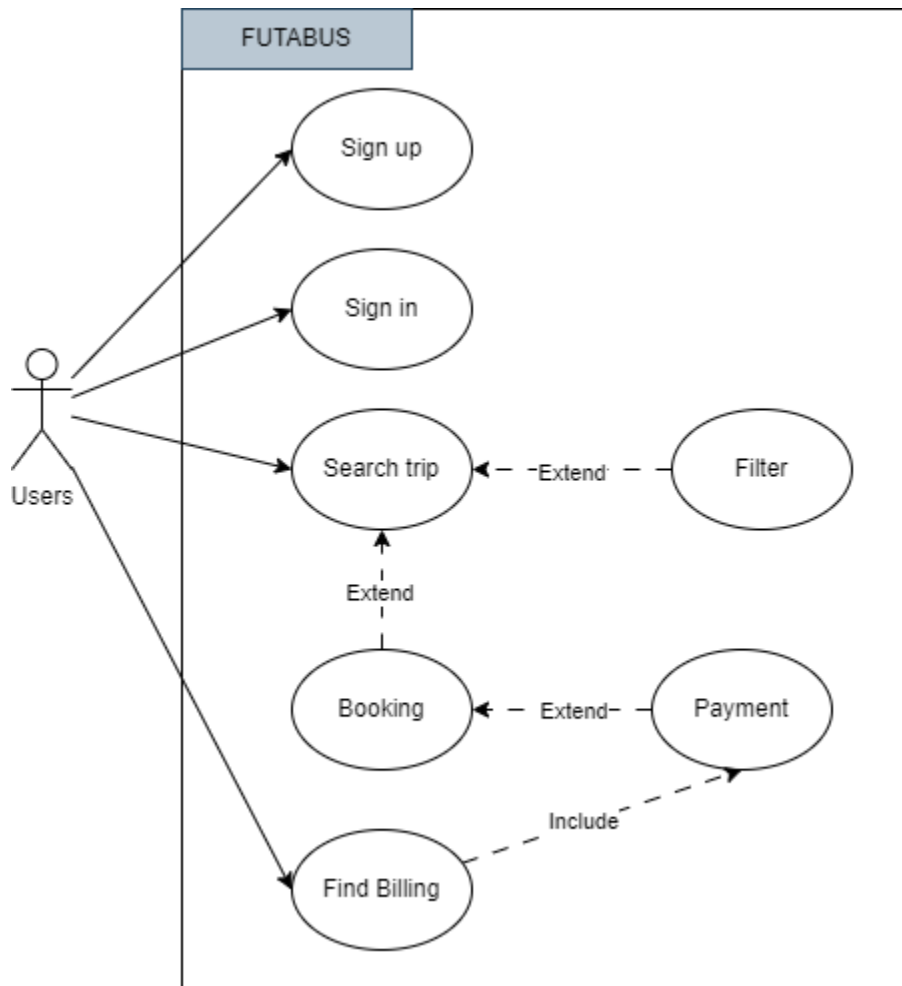
- o Dựa vào bảng phân tích và yêu cầu lưu trữ hãy chọn và thiết kế dữ liệu phù hợp với các quy trình đã phân tích ở trên (NoSQL/ SQL).

- o Cài đặt thiết kế vật lý và các yêu cầu truy xuất cho các chức năng đã mô tả (Mỗi sinh viên cài đặt 1 chức năng demo kết nối với HQT CSDL NoSQL)

- o Thực hiện thử nghiệm với các câu truy vấn với giả định tần suất lớn và các nhu cầu mở rộng. Quan sát thực nghiệm và đề xuất giải pháp cải thiện hiệu quả truy xuất của bản thiết kế và đảm bảo ràng buộc toàn vẹn.

B. Kết quả

I. Mô tả quy trình nghiệp vụ



1. Quy trình tìm chuyến

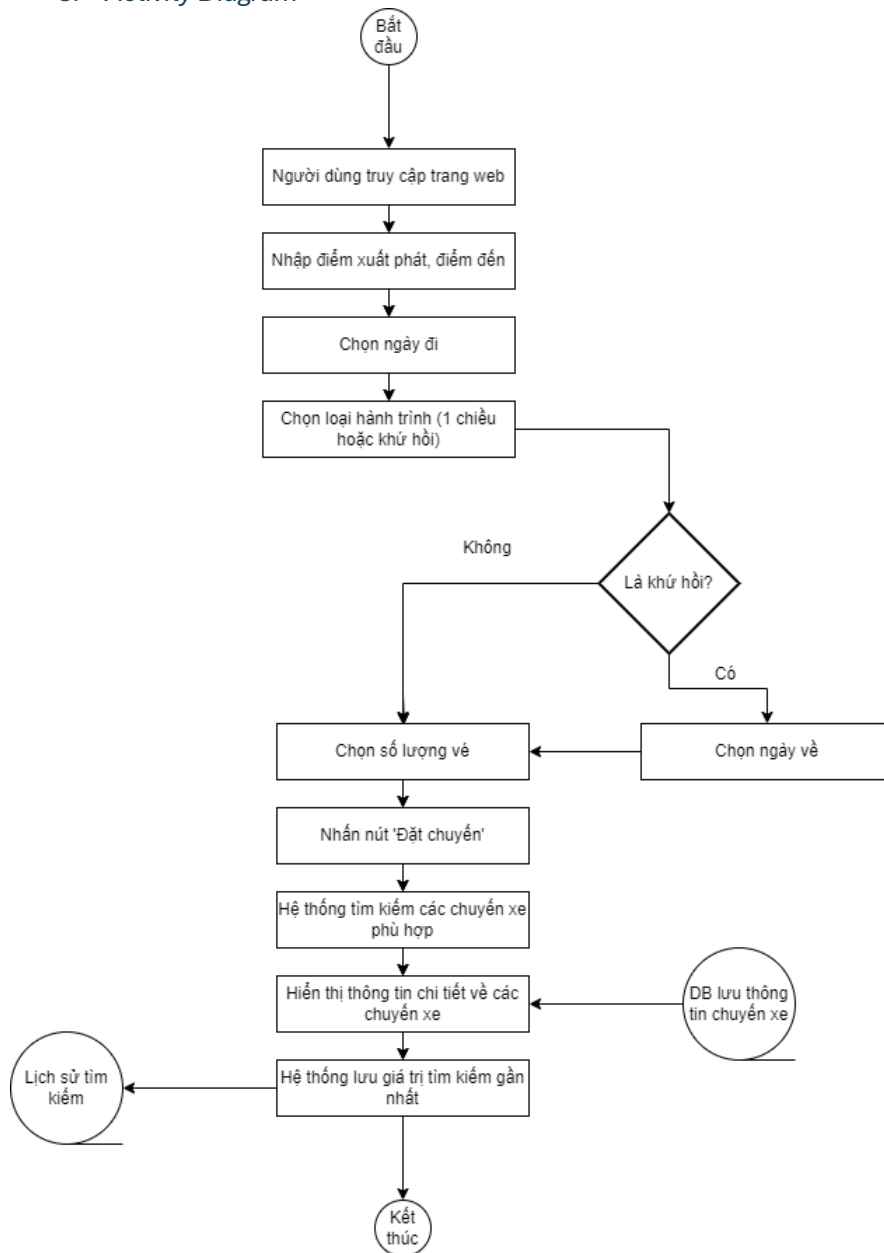
1. Mô tả

- Người dùng truy cập trang web, nhập điểm xuất phát, điểm đến, chọn ngày đi, chọn loại hành trình (1 chiều hoặc khứ hồi) và số lượng vé. Nếu là khứ hồi yêu cầu chọn ngày về.
- Người dùng bấm nút đặt chuyến.
- Hệ thống tìm kiếm các chuyến xe phù hợp và hiển thị cho người dùng kèm thông tin như thời gian dự kiến đến, còn bao nhiêu chỗ trống, lịch trình, chính sách và thông tin trung chuyển, hình ảnh/video, tiện ích, giá vé, bộ lọc (giờ đi, loại xe, hàng ghế, tầng trên hoặc dưới).
- Hệ thống lưu giá trị tìm kiếm gần nhất để người dùng có thể truy cập nhanh.

2. Đặc tả Usecase

Usecase	Nội dung
Tên Usecase	Tìm chuyến
Mô tả	Người dùng tìm kiếm các chuyến xe từ điểm đi đến điểm đến theo ngày và số lượng vé.
Actor	Khách hàng
Điều kiện kích hoạt	Khi người dùng nhập thông tin cần thiết và nhấn nút tìm chuyến xe.
Tiền điều kiện	Không.
Hậu điều kiện	Hiển thị danh sách chuyến xe hợp yêu cầu
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng nhập thông tin điểm đi, điểm đến, ngày đi, và số lượng vé. 2. Hệ thống tìm kiếm các chuyến xe phù hợp. 3. Hiển thị kết quả tìm kiếm.
Luồng sự kiện phụ	<p>A1. Tại 1, nếu chọn hành trình là khứ hồi:</p> <ol style="list-style-type: none"> 1. Chọn ngày về. 2. Tiếp tục bước 2 ở luồng chính. <p>A2. Tại 2, nếu không có chuyến xe phù hợp:</p> <ol style="list-style-type: none"> 1. Hiển thị thông báo kết quả 2. Quay lại bước 1 ở luồng chính.

3. Activity Diagram



2. Quy trình tìm kiếm theo bộ lọc

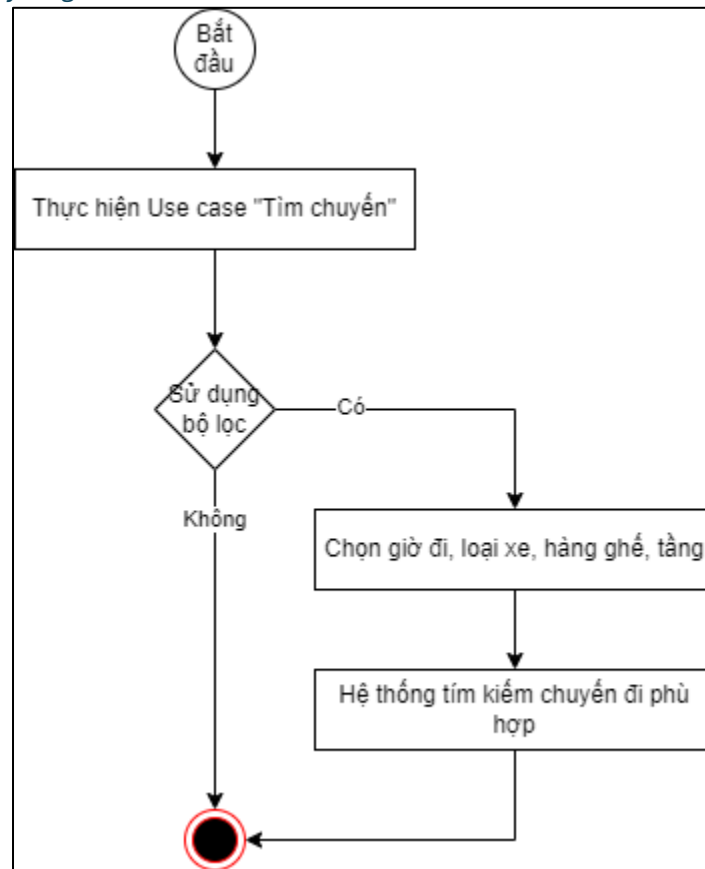
1. Mô tả

- Sau khi người dùng tìm kiếm chuyến, sẽ có 1 bộ lọc để có thể tìm kiếm chi tiết hơn theo thời gian khởi hành, loại xe, hàng ghế và tầng.
- Hệ thống sẽ tìm kiếm những chuyến xe phù hợp với yêu cầu chi tiết của người dùng.

2. Đặc tả Use case

Usecase	Nội dung
Tên Usecase	Tìm kiếm theo bộ lọc
Mô tả	Người dùng tìm kiếm chi tiết chuyến xe theo yêu cầu
Actor	Khách hàng
Điều kiện kích hoạt	Khi người dùng đã thực hiện use case tìm chuyến xe
Tiền điều kiện	Khi người dùng đã thực hiện use case tìm chuyến xe
Hậu điều kiện	Hiển thị danh sách chuyến xe hợp yêu cầu
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người chọn thông tin giờ đi, loại xe <u>loại xe, hàng ghế, tầng.</u> 2. Hệ thống tìm kiếm các chuyến xe phù hợp. 3. Hiển thị kết quả tìm kiếm.
Luồng sự kiện phụ	A1. Tại 2, nếu không có chuyến xe phù hợp: <ol style="list-style-type: none"> 1. Hiển thị thông báo kết quả 2. Quay lại bước 1 ở luồng chính.

3. Activity diagram



3. Quy trình đặt vé

1. Mô tả

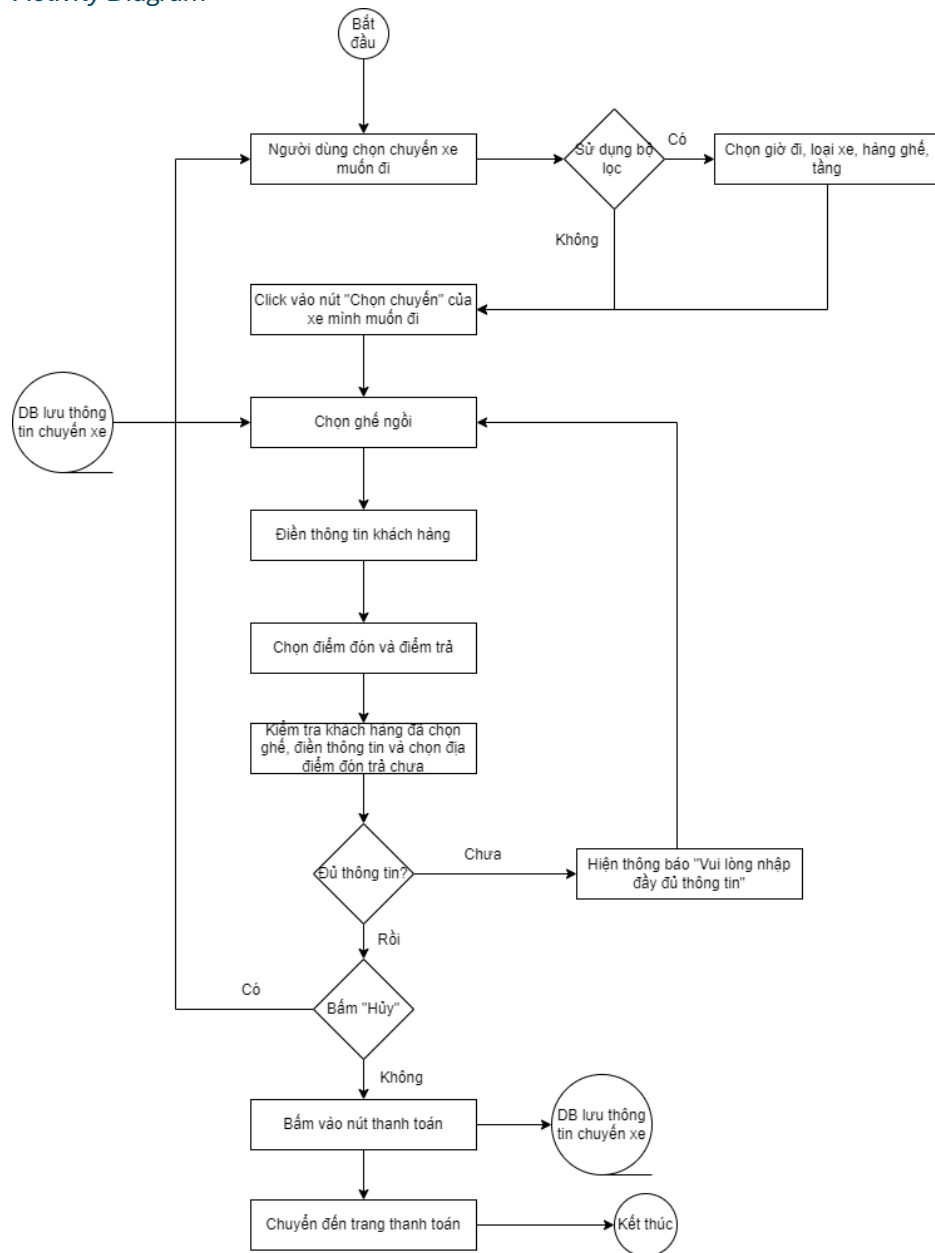
- Người dùng chọn chuyến xe muốn đặt vé (có thể dùng bộ lọc: giờ đi, loại xe, hàng ghế, tầng để tìm kiếm dễ dàng hơn), chọn ghế ngồi, xem thông tin chi tiết (lịch trình, tiện ích, chính sách, thông tin ghế, giá tiền cần thanh toán), và điền thông tin khách hàng (họ và tên, số điện thoại, email).
- Người dùng chọn điểm đón và điểm trả (có thể chọn trung chuyển và nhập địa chỉ cần đến).
- Người dùng bấm nút Thanh toán.

2. Đặc tả Usecase

Usecase	Nội dung
Tên Usecase	Đặt vé
Mô tả	Người dùng chọn chuyến xe và số ghế, sau đó tiến hành đặt vé.

Actor	Khách hàng
Điều kiện kích hoạt	Khách hàng bấm nút tìm chuyến xe.
Tiền điều kiện	Khách hàng đã tìm kiếm chuyến xe phù hợp với yêu cầu.
Hậu điều kiện	Khách hàng chuyển đến trang thanh toán.
Luồng sự kiện chính	<p>Bước 1: Người dùng chọn chuyến xe và số ghế.</p> <p>Bước 2: Hệ thống hiển thị chi tiết chuyến đi và tổng chi phí.</p> <p>Bước 3: Người dùng xác nhận đặt vé.</p>
Luồng sự kiện phụ	<ul style="list-style-type: none"> - Nếu người dùng chưa chọn ghế, thông báo chọn ghế và quay lại bước 1 của luồng sự kiện chính. - Nếu người dùng chọn hủy, quay lại trang tìm chuyến.

3. Activity Diagram



4. Quy trình thanh toán

1. Mô tả

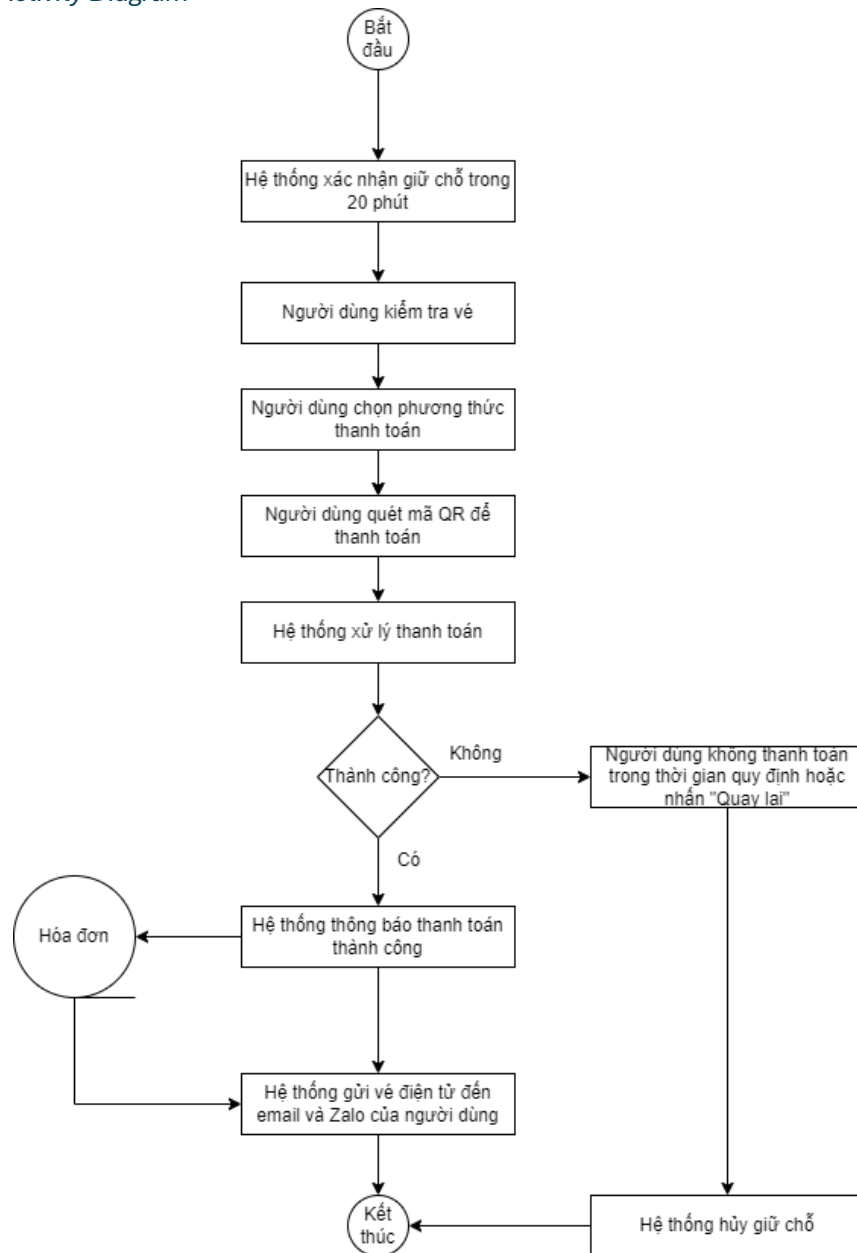
- Người dùng chọn phương thức thanh toán (thẻ tín dụng, ví điện tử, ngân hàng trực tuyến), kiểm tra giá vé, và quét QR để thanh toán.
- Sau khi người dùng thanh toán. Hệ thống xử lý thanh toán và xác nhận thành công.

- Hệ thống gửi vé điện tử đến email và Zalo của người dùng cùng thông tin vé và thông tin khách.

2. *Đặc tả Usecase*

Usecase	Nội dung
Tên Usecase	Đặt vé
Mô tả	Người dùng chọn chuyến xe và số ghế, sau đó tiến hành đặt vé.
Actor	Khách hàng
Điều kiện kích hoạt	Khách hàng scan QR thành công và được hệ thống ghi nhận
Tiền điều kiện	Khách hàng đã đặt vé.
Hậu điều kiện	
Luồng sự kiện chính	<p>Bước 1: Hệ thống xác nhận giữ chỗ trong 20 phút để người dùng chọn hình thức thanh toán.</p> <p>Bước 2: Người dùng quét mã QR để thanh toán.</p> <p>Bước 3: Thanh toán thành công, người dùng nhận vé và hóa đơn qua email và Zalo.</p>
Luồng sự kiện phụ	- Nếu người dùng không thanh toán trong thời gian quy định hoặc nhấn "Quay lại", hệ thống sẽ hủy giữ chỗ.

3. Activity Diagram



5. Quy trình tra cứu hóa đơn

1. Mô tả

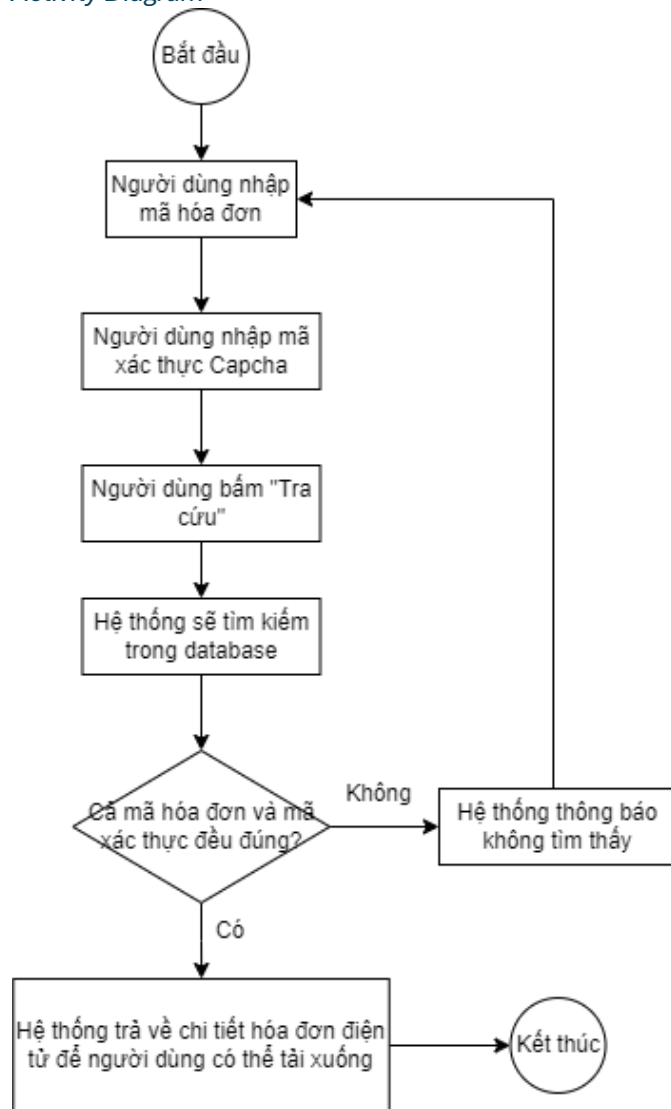
- Người dùng nhập mã hóa đơn.
- Người dùng nhập Capcha.
- Nếu nhập sai mã hóa đơn --> Thông báo “Không tìm thấy mã hóa đơn”
- Nếu nhập sai mã Capcha --> Thông báo “Mã xác thực không đúng”

2. Đặc tả Usecase

Usecase	Nội dung
---------	----------

Tên Usecase	Tra cứu hóa đơn
Mô tả	Người dùng nhập mã hóa đơn để có thể xem được thông tin chi tiết hóa đơn của mình
Actor	Khách hàng
Điều kiện kích hoạt	Khách hàng nhập đúng mã hóa đơn và mã xác thực Capcha
Tiền điều kiện	Khách hàng đã thanh toán hóa đơn.
Hậu điều kiện	Khách hàng nhận được hóa đơn điện tử
Luồng sự kiện chính	<p>Bước 1: Người dùng nhập mã hóa đơn</p> <p>Bước 2: Người dùng nhập mã xác thực</p> <p>Bước 3: Hệ thống trả ra chi tiết hóa đơn của mã hóa đơn đó</p>
Luồng sự kiện phụ	<ul style="list-style-type: none"> - Nếu người dùng nhập sai mã hóa đơn --> Trả về thông báo “Không tìm thấy mã hóa đơn”. - Nếu người dùng nhập sai mã xác thực --> Trả về thông báo “Mã xác thực không đúng”.

3. Activity Diagram



II. Phân tích yêu cầu lưu trữ, khả năng mở rộng, nhu cầu truy suất hệ thống

1. Tìm kiếm chuyển đi

1. Yêu cầu lưu trữ

- Thông tin địa điểm: tỉnh/thành phố, văn phòng
- Thông tin tuyến xe: điểm xuất phát, điểm đến, ngày xuất phát

2. Khả năng mở rộng

- Cơ sở dữ liệu phân tán (Distributed Database): Sử dụng các hệ thống cơ sở dữ liệu phân tán như Cassandra hoặc MongoDB để xử lý lưu lượng truy cập lớn và đảm bảo tính khả dụng cao.
- Bộ nhớ đệm (Caching): Sử dụng các công cụ bộ nhớ đệm như Redis hoặc Memcached để lưu trữ các truy vấn tìm kiếm phổ biến.

3. *Các chức năng hệ thống và nhu cầu truy suất*
 - Tìm kiếm chuyển xe
2. Tìm kiếm chuyển đi theo bộ lọc
 1. *Yêu cầu lưu trữ*
 - Thông tin địa điểm: tỉnh/thành phố, văn phòng
 - Thông tin tuyến xe: điểm xuất phát, điểm đến, ngày xuất phát, giờ đi, loại xe, tầng, giờ dự kiến đến, số lượng ghế còn lại, giá vé
 - Thông tin lịch trình
 - Thông tin chính sách, trung chuyển
 2. *Khả năng mở rộng*
 - Cơ sở dữ liệu phân tán (Distributed Database): Sử dụng các hệ thống cơ sở dữ liệu phân tán như Cassandra hoặc MongoDB để xử lý lưu lượng truy cập lớn và đảm bảo tính khả dụng cao.
 - Bộ nhớ đệm (Caching): Sử dụng các công cụ bộ nhớ đệm như Redis hoặc Memcached để lưu trữ các truy vấn tìm kiếm phổ biến.
 - Load Balancer:
 - o Phân phối tải đều giữa các máy chủ xử lý yêu cầu tìm kiếm.
 - o Đảm bảo không có máy chủ nào bị quá tải.
 3. *Các chức năng hệ thống và nhu cầu truy suất*
 - Tìm kiếm chuyển xe
3. Đặt vé
 1. *Yêu cầu lưu trữ*
 - Thông tin ghế: hàng ghế, tầng vị trí, trạng thái, loại xe
 - Thông tin khách hàng: Tên, sdt, email
 - Thông tin điểm đón, điểm trả:
 - Thông tin vé: mã vé, tổng tiền, trạng thái, thông tin khách hàng, thông tin ghế, thông tin đón trả
 2. *Khả năng mở rộng*
 - Tần suất truy cập lớn vào dịp lễ hay cuối tuần, sử dụng các hệ thống cơ sở dữ liệu phân tán để xử lý lưu lượng truy cập lớn và đảm bảo tính khả dụng cao.
 - Sử dụng các công cụ bộ nhớ đệm như Redis hoặc Memcached để lưu trữ các truy vấn tìm kiếm phổ biến, đảm bảo tốc độ phản hồi nhanh.
 3. *Các chức năng hệ thống và nhu cầu truy suất*
 - Thao tác chọn, hủy một ghế, với tần suất truy suất rất lớn, đặc biệt vào các dịp cuối tuần hay lễ.
 - Thao tác chọn điểm đón trả, tần suất trung bình, tùy thuộc vào thời điểm.
 - Thao tác thanh toán, tần suất trung bình.

4. Thanh toán hoá đơn

1. Yêu cầu lưu trữ

- Hóa đơn: mã hóa đơn, trạng thái, ngày đặt, thông tin vé, tổng tiền, thông tin thanh toán
- Thanh toán: mã thanh toán, tên thanh toán, ngày thanh toán

2. Khả năng mở rộng

- Cơ sở dữ liệu phân tán (Distributed Database): Sử dụng các hệ thống cơ sở dữ liệu phân tán như Cassandra hoặc MongoDB để xử lý lưu lượng truy cập lớn và đảm bảo tính khả dụng cao.
- Bộ nhớ đệm (Caching): Sử dụng các công cụ bộ nhớ đệm như Redis hoặc Memcached để lưu trữ các truy vấn tìm kiếm phổ biến.
- Load Balancer:
 - o Phân phối tải đều giữa các máy chủ xử lý yêu cầu tìm kiếm.
 - o Đảm bảo không có máy chủ nào bị quá tải.

3. Các chức năng hệ thống và nhu cầu truy suất

- Thanh toán

5. Tra cứu hoá đơn/vé

1. Yêu cầu lưu trữ

- Hóa đơn: mã hóa đơn, trạng thái, ngày đặt, thông tin vé, tổng tiền, thông tin thanh toán
- Thanh toán: mã thanh toán, ngày thanh toán, trạng thái, phương thức thanh toán.

2. Khả năng mở rộng

- Cơ sở dữ liệu phân tán (Distributed Database): Sử dụng các hệ thống cơ sở dữ liệu phân tán như Cassandra hoặc MongoDB để xử lý lưu lượng truy cập lớn và đảm bảo tính khả dụng cao.
- Load Balancer:
 - o Phân phối tải đều giữa các máy chủ xử lý yêu cầu tìm kiếm.
 - o Đảm bảo không có máy chủ nào bị quá tải.

3. Các chức năng hệ thống và nhu cầu truy suất

- Xem thông tin hóa đơn/vé

III. Ưu nhược

Cơ sở Dữ liệu	Ưu điểm	Nhược điểm
Redis	<ul style="list-style-type: none">- Tốc độ cao do lưu trữ dữ liệu trong bộ nhớ.- Hỗ trợ nhiều cấu trúc dữ liệu như chuỗi, danh sách, tập hợp, và bản đồ.	<ul style="list-style-type: none">- Lưu trữ dữ liệu trong bộ nhớ giới hạn bởi kích thước RAM.- Không phù hợp cho lưu trữ dữ liệu lớn và phức tạp.

	<ul style="list-style-type: none"> - Phù hợp cho các ứng dụng yêu cầu tốc độ truy cập nhanh như caching, hàng đợi, và phiên làm việc. - Hỗ trợ tính năng sao chép và phân tán dữ liệu. 	<ul style="list-style-type: none"> - Không hỗ trợ truy vấn phức tạp và tìm kiếm văn bản đầy đủ.
MongoDB	<ul style="list-style-type: none"> - Dễ dàng mở rộng theo chiều ngang nhờ tính năng sharding. - Lưu trữ dữ liệu dưới dạng JSON, linh hoạt cho các tài liệu không cấu trúc. - Hỗ trợ các truy vấn phức tạp, lập chỉ mục, và tìm kiếm văn bản đầy đủ. - Phù hợp cho các ứng dụng yêu cầu tính linh hoạt trong cấu trúc dữ liệu. 	<ul style="list-style-type: none"> - Sử dụng nhiều dung lượng lưu trữ hơn so với các cơ sở dữ liệu quan hệ. - Không hỗ trợ tốt cho các giao dịch phức tạp. - Tính nhất quán cuối cùng (eventual consistency) có thể gây ra các vấn đề trong một số ứng dụng.
Cassandra	<ul style="list-style-type: none"> - Khả năng mở rộng theo chiều ngang mạnh mẽ, phù hợp cho các ứng dụng lớn và phân tán. - Hỗ trợ tính nhất quán cuối cùng và chịu được lỗi cao. - Hiệu suất ghi cao, phù hợp cho các ứng dụng ghi dữ liệu liên tục như log, sensor data. - Không có một điểm lỗi (no single point of failure). 	<ul style="list-style-type: none"> - Khó học và quản lý hơn so với một số cơ sở dữ liệu khác. - Không hỗ trợ tốt cho các truy vấn phức tạp và tìm kiếm văn bản đầy đủ. - Tính nhất quán cuối cùng có thể gây ra vấn đề trong một số trường hợp cần tính nhất quán cao.
Neo4j	<ul style="list-style-type: none"> - Tối ưu hóa cho việc lưu trữ và truy vấn dữ liệu dạng đồ thị. - Hỗ trợ truy vấn phức tạp và tìm kiếm các mối quan hệ trong dữ liệu một cách hiệu quả. - Dễ dàng mô hình hóa các mối quan hệ phức tạp trong dữ liệu. - Phù hợp cho các ứng dụng yêu cầu phân tích mạng, khuyến nghị sản phẩm, và quản lý mối quan hệ khách hàng. 	<ul style="list-style-type: none"> - Không mở rộng theo chiều ngang dễ dàng như các cơ sở dữ liệu khác. - Hiệu suất có thể giảm khi dữ liệu không ở dạng đồ thị. - Cộng đồng và tài liệu hỗ trợ có thể ít hơn so với một số cơ sở dữ liệu phổ biến khác

IV. Lựa chọn và thiết kế cơ sở dữ liệu

1. Nghiệp vụ tìm kiếm chuyến xe

1. *Phân tích*

MongoDB có thể là lựa chọn tốt nhất cho nghiệp vụ tìm kiếm chuyến xe.

- Tính linh hoạt trong cấu trúc dữ liệu: MongoDB lưu trữ dữ liệu dưới dạng document JSON, rất linh hoạt cho việc lưu trữ các thông tin khác nhau về chuyến xe.
- Khả năng truy vấn mạnh mẽ: MongoDB hỗ trợ các truy vấn phức tạp, bao gồm các bộ lọc, sort, rất phù hợp cho nghiệp vụ tìm kiếm với nhiều điều kiện khác nhau. Bên cạnh đó, MongoDB hỗ trợ tốt việc index và tối ưu hóa truy vấn -> Tăng hiệu suất truy vấn.
- Khả năng mở rộng: MongoDB có khả năng mở rộng tốt với sharding và replicate, giúp xử lý tốt khi lượng dữ liệu và lưu lượng truy cập tăng lên.
- Hiệu suất truy xuất cao: MongoDB có thể sử dụng index để tăng tốc độ truy vấn, điều này rất quan trọng cho nghiệp vụ tìm kiếm chuyến xe.

Bên cạnh đó, một số lý do mà các loại cơ sở dữ liệu khác không phù hợp là:

- Redis: Khả năng mở rộng hạn chế, vì redis lưu dữ liệu trong bộ nhớ. Và chỉ tìm kiếm nhanh đối với dữ liệu đơn giản. Còn đối với dữ liệu phức tạp, cần kết hợp nhiều điều kiện lọc thì Redis rất hạn chế.
- Cassandra: Tối ưu cho các truy vấn ghi nhanh và không phù hợp cho các truy vấn phức tạp hoặc tìm kiếm dựa trên nhiều bộ lọc khác nhau. Dữ liệu không quá lớn để triển khai bằng Cassandra.
- Neo4j: Không có mối quan hệ phức tạp nên không thích hợp để sử dụng.
- SQL: Schema cứng, yêu cầu phải define rõ ràng schema trước. Khả năng mở rộng theo chiều dọc gây tốn kém tài nguyên.

2. *Thiết kế Model*

```
const SeatSchema = new mongoose.Schema({
  seatNumber: { type: String, required: true },
  row: { type: String, enum: ["front", "middle", "back"], required: true },
  deck: { type: String, enum: ["upper", "lower"], required: true },
  available: { type: Boolean, required: true, default: true },
});

const TripSchema = new mongoose.Schema({
  departure: { type: String, required: true },
  destination: { type: String, required: true },
  departureDate: { type: Date, required: true },
  departureTime: { type: String, required: true },
  vehicleType: {
    type: String,
    enum: ["seat", "bed", "limousine"],
    required: true,
  },
  seats: [SeatSchema]
});
```

3. *Thiết kế Index*

Tạo index cho các trường thường xuyên được sử dụng để tìm kiếm và lọc, ví dụ: departure, destination, departureDate

```
db.trips.createIndex({ departure: 1, destination: 1, departureDate: 1});
```

2. Nghiệp vụ tìm kiếm chuyến xe theo bộ lọc

1. *Phân tích*

MongoDB có thể là lựa chọn tốt nhất cho nghiệp vụ tìm kiếm chuyến xe theo bộ lọc:

- Tính linh hoạt trong cấu trúc dữ liệu: MongoDB lưu trữ dữ liệu dưới dạng document JSON, rất linh hoạt cho việc lưu trữ các thông tin khác nhau về chuyến xe.
- Khả năng truy vấn mạnh mẽ: MongoDB hỗ trợ các truy vấn phức tạp, bao gồm các bộ lọc, sort, và projection, rất phù hợp cho nghiệp vụ tìm kiếm với nhiều điều kiện khác nhau.
- Khả năng mở rộng: MongoDB có khả năng mở rộng tốt với sharding và replicate, giúp xử lý tốt khi lượng dữ liệu và lưu lượng truy cập tăng lên.
- Hiệu suất truy xuất cao: MongoDB có thể sử dụng index để tăng tốc độ truy vấn, điều này rất quan trọng cho nghiệp vụ tìm kiếm chuyến xe.

Bên cạnh đó, một số lý do mà các loại cơ sở dữ liệu khác không phù hợp là:

- Redis: Khả năng mở rộng hạn chế, vì redis lưu dữ liệu trong bộ nhớ. Và chỉ tìm kiếm nhanh đối với dữ liệu đơn giản. Còn đối với dữ liệu phức tạp, cần kết hợp nhiều điều kiện lọc thì Redis rất hạn chế.
- Cassandra: Tối ưu cho các truy vấn ghi nhanh và không phù hợp cho các truy vấn phức tạp hoặc tìm kiếm dựa trên nhiều bộ lọc khác nhau. Dữ liệu không quá lớn để triển khai bằng Cassandra.
- Neo4j: Không có mối quan hệ phức tạp nên không thích hợp để sử dụng.
- SQL: Schema cứng, yêu cầu phải define rõ ràng schema trước. Khả năng mở rộng theo chiều dọc gây tốn kém tài nguyên.

2. Thiết kế Model

```
const SeatSchema = new mongoose.Schema({
  seatNumber: { type: String, required: true },
  row: { type: String, enum: ["front", "middle", "back"], required: true },
  deck: { type: String, enum: ["upper", "lower"], required: true },
  available: { type: Boolean, required: true, default: true },
});

const TripSchema = new mongoose.Schema({
  departure: { type: String, required: true },
  destination: { type: String, required: true },
  departureDate: { type: Date, required: true },
  departureTime: { type: String, required: true },
  vehicleType: {
    type: String,
    enum: ["seat", "bed", "limousine"],
    required: true,
  },
  seats: [SeatSchema]
});
```

3. Thiết kế Index

Tạo index cho các trường thường xuyên được sử dụng để tìm kiếm và lọc, Ngoài tạo index cho các trường dữ liệu departure, destination, departureDate. Bổ sung index cho vehicleType, row, deck, và departureTime cho bảng Trip

```
db.trips.createIndex({ vehicleType: 1, "seats.row": 1, "seats.deck": 1,
"departureTime": 1 });
```

3. Nghiệp vụ đặt vé

1. Phân tích

Nghiệp vụ đặt vé chia thành hai giai đoạn sử dụng kết hợp cơ sở dữ liệu NoSQL và cơ sở dữ liệu quan hệ, ở giai đoạn người dùng chọn ghế, bỏ chọn ghế sẽ sử dụng Redis, còn ở giai đoạn lưu trữ thông tin vé sẽ sử dụng Postgre.

Giai đoạn người dùng chọn, bỏ chọn ghế:

- Người dùng chọn, bỏ chọn ghế, yêu cầu dữ liệu được đọc ghi nhanh chóng, đảm bảo hiệu suất cũng như trải nghiệm của người dùng, vì vậy sử dụng key-value store là hợp lý, đảm bảo tính sẵn sàng của dữ liệu.
- Key-value store hỗ trợ lưu trữ dữ liệu linh hoạt về key và value, giúp dễ dàng quản lý và lưu trữ.
- Quy trình đặt vé yêu cầu tính nhất quán cao, và redis cung cấp cơ chế sao lưu và phục hồi dữ liệu như snapshotting và AOF.
- Khả năng xử lý đồng thời: có các cơ chế locking, giúp xử lý đồng thời nhiều thao tác từ các user khác nhau một cách hiệu quả. Điều này rất quan trọng đối với các hệ thống thanh toán, nơi mà tính nhất quán và độ chính xác của dữ liệu là cực kỳ quan trọng.

- Sử dụng Redis giúp giảm tải cho hệ quản trị cơ sở dữ liệu chính, giảm các truy vấn chọn, bỏ chọn giúp tăng hiệu suất cho toàn hệ thống.

Giai đoạn hệ thống lưu trữ vé khi thanh toán thành công:

- Ở giai đoạn xử lý đặt vé, yêu cầu tính nhất quán và toàn vẹn dữ liệu (ACID: Atomicity, Consistency, Isolation, Durability), nên để đảm bảo sự chính xác của nghiệp vụ, nên sử dụng cơ sở dữ liệu quan hệ, ở nghiệp vụ này sẽ sử dụng Postgre để lưu trữ thông tin đặt vé.
- Cơ sở dữ liệu quan hệ có khả năng xử lý giao dịch mạnh mẽ, đảm bảo thao tác đặt vé thành công, nếu các thao tác kiểm tra, cập nhật trước đó thành công.

2. Thiết kế schema

Thông tin chuyến đi (route) sẽ nhận được từ nghiệp vụ tìm kiếm chuyến đi, nó sẽ bao gồm thông tin cơ bản như schema sau:

```

Run | New Tab | Copy
CREATE TABLE IF NOT EXISTS routes (
  route_id INT PRIMARY KEY,
  arrival_time TIMESTAMP NOT NULL,
  departure_time TIMESTAMP NOT NULL,
  name VARCHAR(255) NOT NULL,
  origin_code VARCHAR(255) NOT NULL,
  dest_code VARCHAR(255) NOT NULL,
  origin_office_id INT REFERENCES offices(office_id),
  dest_office_id INT REFERENCES offices(office_id),
  prices FLOAT NOT NULL
);

```

Thông tin về điểm đón, điểm trả (pickUpPoint):

```

Run | New Tab | Copy
CREATE TABLE IF NOT EXISTS pickup_points (
  id INT PRIMARY KEY,
  address VARCHAR(255) NOT NULL,
  name VARCHAR(255) NOT NULL,
  office_id VARCHAR(255) NOT NULL,
  phone VARCHAR(255) NOT NULL
);

```

Thông tin vé (ticket) sẽ bao gồm thông tin cơ bản chuyến đi, thông tin cơ bản điểm đón, trả, và thông tin người dùng:

```

> Run | New Tab | Copy
CREATE TABLE IF NOT EXISTS tickets (
  ticket_id SERIAL PRIMARY KEY,
  user_id INT NOT NULL,
  user_name VARCHAR(255) NOT NULL,
  user_phone VARCHAR(255) NOT NULL,
  route_id INT NOT NULL,
  route_name VARCHAR(255) NOT NULL,
  pickUp_arrival_id INT NOT NULL,
  pickUp_arrival_address VARCHAR(255) NOT NULL,
  pickUp_departure_id INT NOT NULL,
  pickUp_departure_address VARCHAR(255) NOT NULL,
  list_seat_id INT[] NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  total FLOAT NOT NULL,
  UNIQUE (ticket_id, user_id)

```

Các thông tin liên quan khác như văn phòng (office), người dùng (user) không liên quan trực tiếp đến nghiệp vụ nên sẽ không thể hiện ở đây.

3. Thiết kế bộ nhớ đệm

Mục tiêu sử dụng bộ nhớ đệm để tăng hiệu suất, và giảm tải cho cơ sở dữ liệu chính ở bước người dùng chọn, bỏ chọn ghế trong một chuyến đi đã được chọn trước đó, nên sẽ thiết kế key-value store như sau:

- Key: Order:user_id
- Field: Route:route_id:seat_id
- Value: 0, 1, 2

Lưu dữ liệu thông tin chọn, bỏ chọn ghế cho user_id, chi tiết cho từng route_id, seat_id với giá trị 0 và 1, tương ứng với bỏ chọn và chọn, giá trị 2, có nghĩa là ghế đã được thanh toán.

4. Thiết kế index, partition

Ở giai đoạn tạo vé mới trong cơ sở dữ liệu quan hệ Postgre, yêu cầu đọc thông tin cơ bản của chuyến đi, và điểm đón trả, cần tạo indexing.

```

> Run | New Tab
CREATE INDEX idx_routes_name ON routes(name);
> Run | New Tab
CREATE INDEX idx_pickup_points_address ON pickup_points(address);
> Run | New Tab
CREATE INDEX idx_pickup_points_name ON pickup_points(name);
> Run | New Tab | Copy

```


Lưu trữ thông tin vé hợp lý, đảm bảo hiệu suất khi truy vấn các vé, ta nên partitioning ở bảng lưu trữ vé theo tháng năm.

4. Nghiệp vụ thanh toán vé

1. Phân tích

PostgreSQL có thể là lựa chọn tốt nhất cho nghiệp vụ thanh toán.

- Tính toàn vẹn dữ liệu: PostgreSQL cung cấp các ràng buộc toàn vẹn dữ liệu như khóa chính, khóa ngoại, và ràng buộc duy nhất, giúp đảm bảo tính nhất quán và chính xác của dữ liệu.
- Tính bảo mật cao: SQL Server hỗ trợ nhiều lớp bảo mật, bao gồm xác thực người dùng, mã hóa dữ liệu và các quyền hạn truy cập chi tiết, giúp bảo vệ dữ liệu thanh toán nhạy cảm.
- Hỗ trợ giao dịch (transaction support): PostgreSQL hỗ trợ các giao dịch ACID (Atomicity, Consistency, Isolation, Durability), rất quan trọng trong việc đảm bảo các giao dịch thanh toán được thực hiện chính xác và an toàn.

Bên cạnh đó, một vài lí do khiến các cơ sở dữ liệu khác không phù hợp bằng SQL là:

- MongoDB: MongoDB không hỗ trợ ràng buộc khóa ngoại, dẫn đến khó đảm bảo tính nhất quán dữ liệu.
- Redis: Không phù hợp cho dữ liệu quan hệ. Redis là cơ sở dữ liệu trong bộ nhớ (in-memory database) chủ yếu cho cache và không phù hợp cho việc lưu trữ dữ liệu quan hệ.
- Cassandra: Cassandra không hỗ trợ ràng buộc khóa ngoại và có giới hạn trong việc đảm bảo tính nhất quán dữ liệu. Việc thiết kế schema trong Cassandra yêu cầu kiến thức sâu về mô hình dữ liệu phân tán, phức tạp hơn PostgreSQL.
- Neo4j: Neo4j là cơ sở dữ liệu đồ thị, tối ưu cho việc lưu trữ và truy vấn dữ liệu đồ thị hơn là dữ liệu quan hệ. Mặc dù có hỗ trợ giao dịch ACID nhưng Neo4j không được thiết kế tối ưu cho các ứng dụng thanh toán đòi hỏi tính nhất quán và toàn vẹn dữ liệu cao.

2. Thiết kế model

```
CREATE TABLE IF NOT EXISTS "Payments"
```

```
(
```

```
    "PaymentID " integer NOT NULL DEFAULT nextval("Payments_PaymentID_seq"::regclass),
```

```
    "TicketID" serial NOT NULL,
```

```
    "PaymentMethod" character varying(50) COLLATE pg_catalog."default" NOT NULL,
```

```
    "PaymentAmount" numeric(10,2) NOT NULL,
```

"PaymentStatus" character varying(20) COLLATE pg_catalog."default" NOT NULL,

"PaymentDate" timestamp without time zone NOT NULL,

"CustomerName" character varying(50) COLLATE pg_catalog."default" NOT NULL,

"CustomerEmail" character varying(50) COLLATE pg_catalog."default" NOT NULL,

"CustomerPhone" integer NOT NULL,

CONSTRAINT "Payments_pkey" PRIMARY KEY ("PaymentID ")

FOREIGN KEY ("TicketID") REFERENCES public."Tickets" ("TicketID")

)

3. *Thiết kế index*

Tạo index cột CustomerName vì sẽ có nhiều truy xuất để tìm kiếm thanh toán theo tên khách hàng → cài index cho cột này sẽ tăng hiệu suất.

CREATE INDEX IF NOT EXISTS idx_customername

ON public."Payments" USING btree ("CustomerName")

5. Nghiệp vụ tìm kiếm hóa đơn, vé

1. *Phân tích*

- Hiệu suất truy xuất cao: MongoDB có thể sử dụng index để tăng tốc độ truy vấn (vì trong nghiệp vụ này, ta chỉ để khách hàng tìm kiếm theo mã hóa đơn --> cài index cho cột này).
- Khả năng mở rộng: MongoDB có khả năng mở rộng tốt với sharding và replicate, giúp xử lý tốt khi lượng dữ liệu và lưu lượng truy cập tăng lên. (Do đây là nhà xe lớn tại Việt Nam nên số lượng hóa đơn sẽ tăng lên nhanh chóng).
- Tính linh hoạt trong cấu trúc dữ liệu: MongoDB lưu trữ dữ liệu dưới dạng document JSON, rất linh hoạt cho việc lưu trữ các thông tin khác nhau về hóa đơn mà không cần dùng phép join.

2. Thiết kế model

- Mỗi document có cấu trúc như sau:

```
{
  "_id": ObjectId('668c1999706f1bbac4e3e730')
  "company": Object
    "name": "FUTA Bus Lines"
    "branch": "CHI NHÁNH THÀNH PHỐ HỒ CHÍ MINH - CÔNG TY CỔ PHẦN XE KHÁCH PHƯƠNG TRAN..."
    "address": "486-486A Lê Văn Lương, Phường Tân Phong, Quận 7, Tp Hồ Chí Minh"
    "phone": "19006067"
    "tax_code": "0312241579-033"
  "ticket_info": Object
    "ticket_id": "V5428BT"
    "serial_no": "2307926"
    "price": "395,000 VND"
    "price_details": "Giá trên đã bao gồm thuế GTGT: 8% và bảo hiểm hành khách"
    "route": "BX Miền Tây (TPHCM) - BX An Giang"
    "seat_no": "A20"
    "bus_no": "58A-518.15"
  "departure": Object
    "date": "30-07-2024"
    "time": "20:32"
    "issue_date": "25-06-2024"
  "issuer": Object
    "name": "CHI NHÁNH THÀNH PHỐ HỒ CHÍ MINH - CÔNG TY CỔ PHẦN XE KHÁCH PHƯƠNG TRAN..."
    "tax_code": "0105232093"
    "website": "https://hoadon.futabus.vn"
    "payment_code": "0373407267"
    "seller": "CYBERLOTUS"
    "note": "Bản thể hiện của hóa đơn điện tử"
}
```

3. Thiết kế index

Tạo index cho cột ticket_info.serial_no, vì chức năng này là tìm kiếm theo mã hóa đơn --> cài index cho thuộc tính này sẽ tăng hiệu suất câu truy vấn này đáng kể.

Chưa cài index:

Query Performance Summary

1 documents returned

100000 documents examined

72 ms execution time


Is not sorted in memory


0 index keys examined


⚠ No index available for this query. ⚠


Cài index cho cột ticket_info.serial_no


Query Performance Summary

 1 documents returned

 1 documents examined

 **0 ms** execution time

 **Is not** sorted in memory

 1 index keys examined

Query used the following index:

ticket_info.serial_no ↑