NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng 01 năm 2025

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

ThS. Phạm Minh Đương

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng 01 năm 2025

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Đầu tiên, tôi xin gửi lời cảm ơn chân thành đến Khoa Kỹ thuật và Công nghệ, Bộ môn Công nghệ Thông tin đã tạo điều kiện thuận lợi cho tôi trong suốt quá trình học tập và hoàn thành đồ án thực tập chuyên ngành. Tiếp theo, tôi xin bày tỏ lòng biết ơn sâu sắc đến thầy ThS. Phạm Minh Đương đã truyền đạt kiến thức và hướng dẫn tôi trong quá trình làm đồ án.

Tôi đã cố gắng vận dụng những kiến thức đã học được trong các học kỳ qua để hoàn thành đồ án. Kết quả đạt được là cả quá trình nổ lực của tôi, tuy nhiên vẫn không tránh khỏi những thiếu sót trong quá trình nghiên cứu và trình bày. Rất kính mong sự góp ý của quý thầy (cô) để đồ án thực tập chuyên ngành của tôi được hoàn thiện hơn.

Xin trân trọng cảm ơn!

Sinh viên thực hiện

Dương Văn Hiệp

MỤC LỤC

DANH MỤC HÌNH ẢNH - BẢNG BIỂU

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Vấn đề nghiên cứu

Đề tài tập trung xây dựng một hệ thống quản lý và đặt vé xe cho hãng xe khách, bao gồm các chức năng quản lý cho nội bộ và chức năng dành cho khách hàng. Sử dụng công nghệ .NET Core và NextJS để phát triển. Vấn đề trọng tâm là tích hợp được các công nghệ hiện đại nhằm cung cấp trải nghiệm người dùng tốt, bảo đảm hiệu suất cao và khả năng mở rộng cho hệ thống.

Hướng tiếp cận

Frontend: Xây dựng giao diện người dùng bằng NextJS bao gồm cả trang chủ và trang quản trị. Giao diện được thiết kế thân thiện với người dùng, đơn giản dễ dàng sử dụng, hỗ trợ responsive tối ưu giao diện cho nhiều màn hình thiết bị.

Backend: Phát triển bằng .Net Core, cụ thể là ASP.NET Core Web API, xây dựng các API cho hệ thống, kết hợp với hệ quản trị cơ sở dữ liệu Sql Server cho hiệu suất cao.

Cách giải quyết vấn đề

Tiến hành phân tích yêu cầu nhằm xác định các tính năng cần thiết gồm cả các tính năng cho khách hàng như đặt vé xe, tra cứu thông tin và các tính năng cho quản trị quản lý các xe, chuyến đi, quản lý khách hàng,… Dựa vào đó, thiết kế cơ cở dữ liệu, xây dựng API, thiết kế và xây dựng giao diện với các chức năng đã đặt ra sử dụng các công nghệ đã chọn.

Một số kết quả đạt được

Dự án hoàn thiệt với các tính năng cần thiết cho một hệ thống đặt vé xe cho hãng xe khách bao gồm trang người dùng và trang quản trị, tối ưu hóa trải nghiệm người dùng. Backend sử dụng ASP.Net Core Web API đảm bảo tính an toàn về bảo mật cùng với tốc độ xử lý mạnh mẽ. Triển khai hệ thống lên nền tảng Điện toán đám mây, giúp người dùng có thể truy cập nhanh chống và thuận tiện.

MỞ ĐẦU

Lý do chọn đề tài

Trong giai đoạn chuyển đổi số hiện nay, ứng dụng công nghệ thông tin ngày càng phát triển trên mọi lĩnh vực, trong đó nhu cầu tra cứu và đặt trực tuyến các loại vé nói chung và đặt vé xe khách nói riêng là một tất yếu. Với hệ thống giao thông vận tải của quốc gia đang trên đà phát triển, các tuyến xe khách cũng đang ngày mở rộng, nhu cầu di chuyển giữa các tỉnh thành của người dân ngày càng tăng thì việc phát triển một ứng dụng công nghệ thông tin để đáp ứng các nhu cầu đó là rất cần thiết.

Mục đích nghiên cứu

Mục đích xây dựng một hệ thống đặt vé xe khách dành cho một hãng xe cụ thể, cung cấp các tính năng cho người dùng đặt vé cũng như tra cứu, thanh toán trực tuyến và quản lý dễ dàng cho phía quản trị. Tích hợp các công nghệ hiện đại và mạnh mẽ như .NET Core và NextJS, xây dựng trải nghiệm người dùng tốt, dễ dàng truy cập và sử dụng.

Đối tượng nghiên cứu

Đối tượng nghiên cứu là các công nghệ chính được sử dụng để xây dựng hệ thống bao gồm .NET Core cụ thể là ASP.NET Core Web API, hệ quản trị cơ sở dữ liệu Sql Server, NextJS cùng với một số thư viện hỗ trợ liên quan khác. Đề tài tập trung vào cách kết hợp các công nghệ này lại với nhau để tạo nên một hệ thống hiệu quả, đáp ứng yêu cầu của người dùng, dễ dàng triển khai, nâng cấp và bảo trì.

Phạm vi nghiên cứu

Giới hạn trong việc xây dựng một hệ thống website đặt vé xe khách và chức năng quản lý cho phía hãng xe. Bao gồm các nội dung như phân tích yêu cầu, thiết kế và triển khai hệ thống với các chức năng thiết yếu như đặt vé xe, tra cứu hành trình, thanh toán trực tuyến, quản lý xe, quản lý chuyến đi, quản lý khách hàng.

Phương pháp nghiên cứu

Nguyên cứu lý thuyết .Net Core và NextJS dựa trên các nền tảng trực tuyến như các khóa học trực tuyến, trang bài viết công nghệ, bên cạnh đó còn nghiên cứu dựa trên các tài liệu sách. Sau đó thực nghiệm áp dụng các kiến thức đã nghiên cứu vào xây dự hệ thống cho đề tài.

CHƯƠNG 1. TỔNG QUAN

1.1 Tổng quan về chủ đề

Nhằm tạo ra một nền tảng dành riêng cho một hãng xe cụ thể, cung cấp các tính năng tiện ích như tra cứu tuyến xe, đặt vé, thanh toán trực tuyến, đồng thời hỗ trợ quản lý hiệu quả cho phía quản trị viên. Hệ thống này sẽ đóng vai trò quan trọng trong việc nâng cao chất lượng dịch vụ vận tải, đáp ứng tốt hơn nhu cầu của khách hàng cũng như cải thiện hiệu quả vận hành cho doanh nghiệp.

Tập trung vào việc nghiên cứu với trọng tâm chính gồm các khía cạnh sau:

Phân tích nhu cầu và nghiệp vụ đặt vé xe khách: Tìm hiểu quy trình đặt vé truyền thống và các vấn đề mà cả hành khách lẫn nhà xe thường gặp phải. Từ đó, xác định các yêu cầu nghiệp vụ cần thiết như tra cứu tuyến xe, chọn ghế ngồi, đặt vé, thanh toán trực tuyến, và quản lý vé cho nhà xe.

Thiết kế và triển khai hệ thống:

Phía Client: Xây dựng giao diện thân thiện, dễ sử dụng cho khách hàng đặt vé và quản trị viên quản lý.

Phía Backend: Tập trung vào việc thiết kế cơ sở dữ liệu và triển khai các API để xử lý dữ liệu, đảm bảo hệ thống hoạt động ổn định, bảo mật và hiệu quả.

Tích hợp công nghệ hiện đại: Đồ án sẽ tập trung nghiên cứu việc áp dụng các công nghệ tiên tiến như .NET Core để xây dựng backend mạnh mẽ và NextJS để phát triển frontend. Sự kết hợp này nhằm tối ưu hóa hiệu suất, tính bảo mật, và khả năng mở rộng của hệ thống.

Giải pháp thanh toán trực tuyến: Nghiên cứu và tích hợp các cổng thanh toán phổ biến, đảm bảo quy trình thanh toán diễn ra nhanh chóng, an toàn và phù hợp với người dùng Việt Nam.

Trải nghiệm người dùng: Nghiên cứu các yếu tố để tối ưu hóa trải nghiệm người dùng, bao gồm hiệu suất tải trang, thiết kế giao diện trực quan, thân thiện và khả năng truy cập dễ dàng trên nhiều thiết bị.

Quản lý và vận hành hệ thống: Tìm hiểu các chức năng quản lý dành cho nhà xe, bao gồm quản lý lịch trình, ghế trống, báo cáo doanh thu, và thống kê dữ liệu khách hàng để cải thiện hiệu quả hoạt động.

1.2 Các công nghệ sử dụng

.Net Core: Một nền tảng phát triển ứng dụng mã nguồn mở, đa nền tảng do Microsoft phát triển, được sử dụng rộng rãi trong việc xây dựng các ứng dụng web và API mạnh mẽ. Với hiệu suất cao, khả năng mở rộng tốt và tích hợp nhiều thư viện hỗ trợ, .NET Core cung cấp một nền tảng vững chắc để phát triển backend cho hệ thống. Đặc biệt, tính năng hỗ trợ đa nền tảng giúp hệ thống có thể triển khai trên Windows, Linux hoặc macOS, mang lại tính linh hoạt và tiết kiệm chi phí. Ngoài ra, các công cụ tích hợp của .NET Core giúp tối ưu hóa quá trình phát triển và bảo trì ứng dụng, đảm bảo hiệu quả vận hành trong dài hạn.

SQL Server: hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ, được biết đến với khả năng xử lý khối lượng lớn dữ liệu và bảo mật cao. Đây là một lựa chọn lý tưởng để lưu trữ thông tin liên quan đến tuyến xe, thông tin người dùng, vé đặt và các giao dịch thanh toán. SQL Server cung cấp các tính năng như tối ưu hóa truy vấn, quản lý giao dịch hiệu quả và bảo mật dữ liệu ở mức độ cao. Tích hợp với .NET Core, SQL Server giúp đảm bảo tính ổn định và hiệu năng trong việc quản lý và truy xuất dữ liệu của hệ thống.

NextJS: Một framework phát triển giao diện người dùng mạnh mẽ dựa trên React, cung cấp khả năng kết xuất phía máy chủ (Server-side Rendering) và tạo trang tĩnh (Static Generation). NextJS giúp tối ưu hóa tốc độ tải trang, cải thiện SEO và mang lại trải nghiệm người dùng mượt mà. Với khả năng tích hợp linh hoạt, framework này giúp xây dựng giao diện trực quan, tương tác cao và dễ sử dụng trên nhiều loại thiết bị. Ngoài ra, NextJS hỗ trợ quản lý state hiệu quả, định tuyến động và tích hợp API, giúp frontend hoạt động trơn tru và dễ dàng kết nối với backend xây dựng bằng .NET Core.

Sự kết hợp giữa .NET Core, SQL Server và NextJS tạo nên một hệ thống toàn diện, vừa mạnh mẽ, ổn định về mặt xử lý dữ liệu, vừa thân thiện và tối ưu về trải nghiệm người dùng.

1.3 Các ứng dụng tương tự hiện có

1.3.1 Nền tảng VeXeRe

VeXeRe là nền tảng đặt vé xe khách trực tuyến lớn nhất tại Việt Nam, được thành lập vào năm 2013. Nền tảng này cung cấp dịch vụ đặt vé cho hơn 700 hãng xe, bao phủ hơn 2.600 tuyến đường trong nước và quốc tế.

Các dịch vụ chính của VeXeRe

Đặt vé xe khách trực tuyến: Người dùng có thể tìm kiếm và đặt vé xe khách một cách nhanh chóng thông qua trang web hoặc ứng dụng di động của VeXeRe. Nền tảng này cung cấp thông tin về lịch trình, giá vé và các hãng xe, giúp người dùng so sánh và lựa chọn dịch vụ phù hợp.

Đặt vé máy bay và tàu hỏa: Ngoài vé xe khách, VeXeRe còn mở rộng dịch vụ cho phép người dùng đặt vé máy bay và vé tàu hỏa, tạo sự thuận tiện trong việc lập kế hoạch di chuyển.

Hệ thống quản lý cho hãng xe (Bus Management System - BMS): VeXeRe cung cấp phần mềm quản lý cho các hãng xe, giúp họ tối ưu hóa doanh thu, giảm chi phí và nâng cao hiệu quả hoạt động.

Hệ thống quản lý đại lý (Agent Management System): Hơn 5.000 đại lý sử dụng hệ thống này để tăng doanh số bán vé và phục vụ khách hàng tốt hơn.

VeXeRe cung cấp ứng dụng di động trên cả hai nền tảng iOS và Android, cho phép người dùng tìm kiếm và đặt vé xe khách, vé máy bay và vé tàu hỏa một cách thuận tiện. Ứng dụng này cũng cung cấp các tính năng như hủy vé, theo dõi xe trực tuyến và nhiều ưu đãi hấp dẫn.

Tuy nhiên, do có quá nhiều chức năng trong một hệ thống, VeXeRe làm cho khách hàng khó chọn lựa và sử dụng. Các hãng xe chưa được độc quyền riêng về hệ thống, cũng như khách hàng.

1.3.2 Hệ thống Futabus

Futabus.vn là trang web chính thức của Công ty Cổ phần Xe Khách Phương Trang (FUTA Bus Lines), một trong những hãng xe khách lớn và uy tín tại Việt Nam

thời điểm hiện tại. Trang web cung cấp các dịch vụ chính như đặt vé xe trực tuyến, tra cứu lịch trình, và cập nhật tin tức liên quan đến hoạt động của hãng.

Các tính năng chính của Futabus

Đặt vé trực tuyến: Người dùng có thể dễ dàng chọn điểm đi, điểm đến, ngày khởi hành và số lượng vé, sau đó tiến hành đặt vé và thanh toán trực tuyến. Quy trình đặt vé được hướng dẫn chi tiết trên trang web.

Tra cứu lịch trình: Cung cấp thông tin về các tuyến xe, giờ khởi hành, loại xe và giá vé, giúp khách hàng lựa chọn chuyến đi phù hợp.

Thông tin khuyến mãi: Cập nhật các chương trình ưu đãi, giảm giá dành cho khách hàng.

Hỗ trợ khách hàng: Cung cấp thông tin liên hệ, tổng đài hỗ trợ và các câu hỏi thường gặp để giải đáp thắc mắc của khách hàng.

Ngoài ra, FUTA Bus Lines còn cung cấp ứng dụng di động FUTA Bus trên các nền tảng iOS và Android, giúp người dùng đặt vé và theo dõi thông tin chuyến đi một cách thuận tiện.

CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT

2.1 Giới thiệu về NextJS

2.1.1 Tổng quan về NextJS

NextJS là một framework mã nguồn mở được phát triển dựa trên React, cho phép xây dựng các ứng dụng web với nhiều tính năng tối ưu hóa, bao gồm Server-Side Rendering (SSR) và Static Site Generation (SSG). Được phát triển bởi công ty Vercel, NextJS giúp tăng cường hiệu suất và khả năng SEO cho các ứng dụng web bằng cách cho phép máy chủ xử lý dữ liệu và tạo HTML trước khi gửi đến trình duyệt.

2.1.2 Ưu nhược điểm của NextJS

Ưu điểm:

Tối ưu hóa SEO: Hỗ trợ Server-Side Rendering (SSR) và Static Site Generation (SSG), giúp các trang web được lập chỉ mục dễ dàng hơn, cải thiện thứ hạng trên các công cụ tìm kiếm.

Hiệu suất cao: NextJS cho phép tạo ra các trang web tải nhanh và mượt mà nhờ các tính năng như SSG và SSR, mang lại trải nghiệm người dùng tốt hơn.

Hệ thống Route tự động: Tự động tạo các route dựa trên cấu trúc thư mục, giúp việc quản lý và phát triển ứng dụng trở nên dễ dàng hơn.

Hỗ trợ CSS và JavaScript hiện đại: Cho phép nhập tệp CSS trực tiếp trong tệp JavaScript và hỗ trợ các tính năng JavaScript hiện đại, giúp phát triển ứng dụng nhanh chóng và hiệu quả.

Tích hợp API routes: NextJS cho phép tạo các API routes để xử lý các yêu cầu HTTP, giúp xây dựng các API RESTful một cách dễ dàng.

Nhược điểm:

Giới hạn trong hệ thống định tuyến: NextJS bị giới hạn việc chỉ sử dụng bộ định tuyến dựa trên cấu trúc tệp, không thể tùy chỉnh cách xử lý các route.

Hệ sinh thái và thư viện còn hạn chế: ít thư viện hơn so với các framework khác, có thể gây khó khăn khi tìm kiếm giải pháp cho các yêu cầu cụ thể.

2.1.3 Cấu trúc của NextJS

Cấu trúc thư mục hiện tại của NextJS như sau:

```
my-nextjs-app/
|- .next/
|- node_modules/
|- public/
|- src/
|  |- app/
|  |- components/
|  |- styles/
|  |- utils/
|- package.json
|- next.config.js
|- README.md
```

Trong đó:

.next/: Thư mục này được tạo ra sau khi build dự án, chứa các tệp liên quan đến quá trình biên dịch và tối ưu hóa.

node_modules/: Chứa các module và thư viện mà dự án phụ thuộc, được quản lý bởi npm hoặc yarn.

public/: Lưu trữ các tệp tĩnh như hình ảnh, favicon, và các tài nguyên khác. Các tệp trong thư mục này có thể được truy cập trực tiếp thông qua đường dẫn URL.

src/: Thư mục gốc cho mã nguồn ứng dụng, giúp tổ chức code một cách rõ ràng và dễ quản lý.

app/: Chứa các thành phần ứng dụng và định tuyến. Trong NextJS, có thể tạo các trang bằng cách thêm các tệp vào thư mục này, NextJS sẽ tự động thiết lập các route dựa trên cấu trúc thư mục.

components/: Lưu trữ các component React dùng chung trong ứng dụng, giúp tái sử dụng và quản lý code hiệu quả.

styles/: Chứa các tệp CSS hoặc các giải pháp styling khác cho ứng dụng, giúp tách biệt phần giao diện và logic.

utils/: Lưu trữ các hàm tiện ích, helper functions, và logic chung được sử dụng trong nhiều phần của ứng dụng.

package.json: Tệp này chứa thông tin về dự án, bao gồm các gói phụ thuộc, script, và các cấu hình khác.

next.config.js: Tệp cấu hình cho NextJS, cho phép tùy chỉnh các thiết lập mặc định của framework theo nhu cầu dự án.

README.md: Tệp mô tả dự án, thường được sử dụng để cung cấp thông tin tổng quan, hướng dẫn cài đặt và sử dụng.

Ngoài ra, còn có thể tạo thêm cái thư mục khác tùy theo nhu cầu thực tế của từng dự án.

2.1.4 Page Router và App Router trong NextJS

Trong Next.js, có hai hệ thống định tuyến chính: Pages Router và App Router.

Bảng 1 So sánh Page Router và App Router

	Page Router	***App Router***
Cấu trúc thư mục	Sử dụng thư mục **pages/**, mỗi tệp tương ứng với một route.	Sử dụng thư mục **app/**, tạo các route lồng nhau thông qua cấu trúc thư mục.
Thành phần	Mặc định sử dụng thành phần phía client.	Mặc định sử dụng các thành phần phía server, giúp tối ưu hiệu suất.
Gọi dữ liệu	Sử dụng các hàm **getStaticProps**, **getServerSideProps,** **getInitialProps** để lấy dữ liệu.	Sử dụng hàm **fetch** trực tiếp trong các thành phần để lấy dữ liệu.

Layouts	Layouts tĩnh, không hỗ trợ lồng nhau một cách linh hoạt.	Hỗ trợ layouts động và có thể lồng nhau, cho phép cấu trúc giao diện phức tạp hơn.
Server Components	Không hỗ trợ.	Có hỗ trợ, tận dụng lợi thế của React Server Components.

Lựa chọn giữa Pages Router và App Router:

Dự án mới: Nên sử dụng App Router để tận dụng các tính năng hiện đại và hiệu suất cao.

Yêu cầu đơn giản: Pages Router phù hợp với các dự án nhỏ, yêu cầu định tuyến đơn giản.

Yêu cầu phức tạp: App Router thích hợp cho các ứng dụng lớn với cấu trúc định tuyến và dữ liệu phức tạp.

2.2 Giới thiệu về Bootstrap

2.2.1 Tổng quan về Bootstrap

Bootstrap là một framework front-end mã nguồn mở, được phát triển bởi Mark Otto và Jacob Thornton tại Twitter vào năm 2011. Bootstrap giúp các nhà phát triển tạo ra các trang web và ứng dụng web dễ dàng và nhanh chóng với giao diện responsive và thống nhất.

Bootstrap bao gồm các thành phần như CSS, JavaScript và HTML, cung cấp nhiều công cụ để xây dựng giao diện người dùng (UI) đẹp mắt, dễ sử dụng và dễ duy trì. Những thành phần quan trọng trong Bootstrap bao gồm:

Grid system: Hệ thống lưới 12 cột linh hoạt cho phép bạn bố trí các phần tử trên trang web một cách dễ dàng và responsive.

Components: Các thành phần giao diện người dùng như buttons, navigation bars, modals, alerts, cards, forms, v.v.

Utilities: Các lớp tiện ích như margin, padding, background color, text alignment, v.v.

JavaScript plugins: Các plugin như dropdowns, carousels, modals, tooltips giúp trang web thêm tính tương tác mà không cần phải viết nhiều mã JavaScript.

2.2.2 Ưu nhược điểm của Bootstrap

Ưu điểm

Tiết kiệm thời gian phát triển: Bootstrap cung cấp các thành phần và giao diện sẵn có giúp nhà phát triển tiết kiệm thời gian thiết kế và lập trình giao diện.

Responsive mặc định: Các trang web xây dựng bằng Bootstrap sẽ tự động điều chỉnh giao diện để hiển thị đẹp trên tất cả các thiết bị (máy tính, điện thoại, máy tính bảng).

Tính nhất quán: Bootstrap giúp đảm bảo tính nhất quán về giao diện và trải nghiệm người dùng trên toàn bộ trang web, nó cung cấp các quy chuẩn thiết kế sẵn.

Dễ học và sử dụng: Với tài liệu hướng dẫn chi tiết và một cộng đồng lớn, Bootstrap dễ dàng cho người mới bắt đầu học và sử dụng.

Tương thích với nhiều trình duyệt: Bootstrap đảm bảo tính tương thích cao với các trình duyệt phổ biến như Chrome, Firefox, Safari, Edge, giúp giảm thiểu công sức kiểm tra và sửa lỗi.

Cộng đồng phát triển lớn: Có một cộng đồng rộng lớn, hỗ trợ tài liệu phong phú, mã nguồn mở, giúp dễ dàng tìm kiếm các giải pháp và ý tưởng.

Nhược điểm

Giao diện giống nhau: Nếu không tùy chỉnh, các trang web dùng Bootstrap có thể trông khá giống nhau, thiếu sự độc đáo và sáng tạo.

Kích thước lớn của tệp: Mặc dù Bootstrap cung cấp khả năng tùy chỉnh, nhưng nếu dùng toàn bộ framework, kích thước của các tệp CSS và JavaScript có thể khá lớn, ảnh hưởng đến tốc độ tải trang.

Không linh hoạt cho thiết kế phức tạp: Khi bạn cần một thiết kế rất đặc biệt hoặc muốn kiểm soát chi tiết từng phần tử, Bootstrap có thể không linh hoạt và yêu cầu nhiều sự tùy chỉnh.

2.3 Giới thiệu về ASP.NET Core

2.3.1 Tổng quan về ASP.NET Core

ASP.NET Core là một tập hợp các thư viện chuẩn như một framework để xây dựng ứng dụng web. ASP.NET Core không phải là phiên bản tiếp theo của ASP.NET mà là một cái tên mới được xây dựng từ đầu. Có một sự thay đổi lớn về kiến trúc và kết quả là nó gọn hơn, phân chia module tốt hơn. ASP.NET Core có thể chạy trên cả .NET Core hoặc full .NET Framework.

.NET Core là môi trường thực thi. Nó được thiết kế lại hoàn toàn của .NET Framework. Mục tiêu chính của .NET Core là hỗ trợ phát triển ứng dụng đa nền tảng cho ứng dụng .NET. Nó được hỗ trợ trên Windows, MacOS và Linux. .NET Core là một framework mã nguồn mở được xây dựng và phát triển bởi Microsoft và cộng đồng .NET trên Github

Nó cũng triển khai đặc điểm của .NET Standard. .NET Standard là một đặc tả chuẩn của .NET API hướng tới hỗ trợ trên tất cả các triển khai của nền tảng .NET. Nó định nghĩa một tập các quy tắc thống nhất cần thiết để hỗ trợ tất cả các ứng dụng trên nền .NET.

2.3.2 Các đặc tính quan trọng của ASP.NET Core

Đa nền tảng: Hỗ trợ chạy trên nhiều hệ điều hành, bao gồm Windows, macOS và Linux, cho phép phát triển và triển khai ứng dụng trên nhiều môi trường khác nhau.

Hiệu suất cao: Được tối ưu hóa để cung cấp hiệu suất vượt trội, ASP.NET Core giúp giảm thời gian phản hồi và tăng khả năng xử lý của ứng dụng.

Mô hình lập trình MVC và Web API hợp nhất: ASP.NET Core kết hợp mô hình MVC (Model-View-Controller) và Web API, cho phép xây dựng cả giao diện người dùng và dịch vụ web trong cùng một framework.

Hỗ trợ Dependency Injection: Framework tích hợp sẵn cơ chế Dependency Injection, giúp quản lý phụ thuộc giữa các thành phần trong ứng dụng một cách hiệu quả.

Cấu hình linh hoạt: ASP.NET Core cung cấp hệ thống cấu hình mạnh mẽ, hỗ trợ nhiều nguồn cấu hình khác nhau như tệp JSON, biến môi trường và tham số dòng lệnh, giúp dễ dàng quản lý cấu hình trong các môi trường khác nhau.

Hỗ trợ các framework phía client hiện đại: ASP.NET Core tích hợp tốt với các framework phía client như Angular, React và Vue.js, hỗ trợ phát triển ứng dụng web hiện đại và tương tác.

Bảo mật tích hợp: Framework cung cấp các tính năng bảo mật mạnh mẽ như xác thực và phân quyền, giúp bảo vệ ứng dụng khỏi các mối đe dọa và tấn công phổ biến.

Hỗ trợ phát triển và triển khai nhanh chóng: ASP.NET Core hỗ trợ quy trình phát triển nhanh chóng với khả năng biên dịch và triển khai linh hoạt, giúp giảm thời gian phát triển và triển khai ứng dụng.

2.3.3 Các nhánh của ASP.NET

Có hai nhánh của ASP.NET cho đến hiện tại là ASP.NET và ASP.NET Core.

ASP.NET là phiên bản hiện tại của ASP.NET và nó cần .NET Framwork để chạy.

ASP.NET Core được thiết kế để xây dựng các ứng dụng web hiện đại, hỗ trợ đa nền tảng và hiệu suất cao. Nó hợp nhất ASP.NET MVC và ASP.NET Web API, cung cấp một nền tảng linh hoạt và mạnh mẽ cho việc phát triển ứng dụng web.

Việc lựa chọn giữa ASP.NET và ASP.NET Core phụ thuộc vào yêu cầu cụ thể của dự án và môi trường triển khai.

2.3.4 ASP.NET và ASP.NET Core

Bảng 2 So sánh giữa APS.NET và ASP.NET Core

ASP.NET	***ASP.NET CORE***
Nền tảng đã có từ lâu	Hoàn toàn được thiết kế mới
Chạy trên .NET Framwork	Chạy trên cả .NET Core và .NET Framework

Chỉ trên Windows	Chạy trên tất cả các OS sử dụng .NET Core
Nền tảng ổn định với tính năng phong phú	Chưa hoàn chỉnh nhưng mong đợi sẽ hoàn chỉnh trong tương lai
WebForms được hỗ trợ	Không hỗ trợ WebForms
System.web.dll cồng kềnh	Nhỏ, nhẹ và module hóa
Bản quyền của Microsoft	ASP.NET Core là mã nguồn mở

2.3.5 Ưu nhược điểm của ASP.NET Core

Ưu điểm:

Hiệu năng cao: ASP.NET Core được tối ưu hóa để cung cấp hiệu suất tốt hơn so với các phiên bản trước, đồng thời tối ưu hóa việc sử dụng hệ thống tài nguyên.

Đa nền tảng: Nhờ vào .NET Core, có thể xây dựng phần mềm trên Windows, Linux, macOS một cách dễ dàng.

Mã nguồn mở: ASP.NET Core là một mã nguồn mở, một xu thế mà các ngôn ngữ lập trình hiện nay hướng đến.

Tích hợp tốt với các framework phía client: ASP.NET Core được thiết kế tích hợp với nhiều client side frameworks một cách liên tục bao gồm AngularJS, ReactJS,..

Hỗ trợ triển khai trên đám mây: ASP.NET Core được thiết kế để tối ưu development framework cho những ứng dụng cái mà được chạy on-promise hay được triển khai trên đám mây.

Nhược điểm:

Số lượng thư viện hỗ trợ hạn chế: Có ít thư viện hơn so với .NET Framework có hệ sinh thái và nhiều thư viện hỗ trợ.

Yêu cầu công cụ phát triển: Nếu không có Visual Studio thì việc phát triển khó khăn hơn.

Không hỗ trợ Web Forms: ASP.NET Core không hỗ trợ cho Web Forms.

2.4 Giới thiệu về RESTful API

2.4.1 Tổng quan về RESTful API

RESTful API (Representational State Transfer Application Programming Interface) là một tiêu chuẩn thiết kế API cho các ứng dụng web, giúp quản lý và tương tác với các tài nguyên hệ thống như tệp văn bản, hình ảnh, âm thanh, video, đặc biệt là dữ liệu động. RESTful API sử dụng các phương thức HTTP tiêu chuẩn như GET, POST, PUT và DELETE để thực hiện các thao tác tương ứng với việc truy xuất, tạo mới, cập nhật và xóa tài nguyên.

2.4.2 Các thành phần chính của RESTful API

API (Application Programming Interface): Là tập hợp các quy tắc cho phép một ứng dụng hoặc thành phần tương tác với ứng dụng hoặc thành phần khác, thường trả về dữ liệu ở các định dạng như JSON hoặc XML.

REST (Representational State Transfer): Là một kiểu kiến trúc sử dụng các phương thức HTTP đơn giản để tạo giao tiếp giữa các máy, cho phép gửi yêu cầu HTTP như GET, POST, DELETE đến một URL để xử lý dữ liệu.

2.4.3 Nguyên tắc hoạt động của RESTful API

RESTful API hoạt động dựa trên giao thức HTTP, với các phương thức chính:

GET: Truy xuất một tài nguyên hoặc danh sách tài nguyên.

POST: Tạo mới một tài nguyên.

PUT: Cập nhật thông tin cho một tài nguyên.

DELETE: Xóa một tài nguyên.

Các phương thức này tương ứng với các thao tác CRUD (Create, Read, Update, Delete) trong quản lý dữ liệu.

2.4.4 Ưu nhược điểm của RESTful API

Ưu điểm

Đơn giản và dễ sử dụng: Sử dụng các phương thức HTTP tiêu chuẩn giúp việc phát triển và tích hợp trở nên dễ dàng hơn.

Tính mở rộng: Cho phép các ứng dụng kết nối và trao đổi dữ liệu một cách dễ dàng, hỗ trợ việc mở rộng và thêm tính năng mới.

Tính độc lập: Client và server hoạt động độc lập, giúp việc bảo trì và nâng cấp ứng dụng thuận tiện hơn.

Khả năng tương thích: Tương thích với nhiều ứng dụng khác nhau, hỗ trợ tích hợp dễ dàng giữa các hệ thống.

Nhược điểm

Không lưu trạng thái (Stateless): RESTful API hoạt động theo nguyên tắc không lưu trạng thái, nghĩa là mỗi yêu cầu từ client đến server đều độc lập và không lưu trữ thông tin về các yêu cầu trước đó. Có thể làm tăng khối lượng dữ liệu truyền tải, vì mỗi yêu cầu phải chứa đầy đủ thông tin cần thiết, dẫn đến tốn tài nguyên và có thể ảnh hưởng đến hiệu suất.

Bảo mật hạn chế: Do tính chất mở và không lưu trạng thái, RESTful API có thể gặp khó khăn trong việc bảo mật, đặc biệt khi truyền tải dữ liệu nhạy cảm. Việc bảo vệ dữ liệu đòi hỏi các biện pháp bổ sung như xác thực, mã hóa và quản lý phiên làm việc, điều này có thể phức tạp và tốn kém.

Khó khăn trong việc quản lý phiên (Session Management): Vì RESTful API không lưu trạng thái, việc quản lý phiên làm việc của người dùng trở nên phức tạp hơn. Điều này đòi hỏi các giải pháp bổ sung để theo dõi và quản lý trạng thái người dùng, tăng thêm độ phức tạp cho ứng dụng.

Giới hạn về giao thức: RESTful API chủ yếu hoạt động trên giao thức HTTP, điều này có thể là hạn chế nếu ứng dụng yêu cầu sử dụng các giao thức khác hoặc cần tính năng không được hỗ trợ bởi HTTP.

Chi phí phát triển và bảo trì: Việc thiết kế và triển khai RESTful API đòi hỏi kiến thức chuyên sâu và kinh nghiệm, dẫn đến chi phí phát triển và bảo trì có thể cao hơn, đặc biệt đối với các hệ thống phức tạp.

2.5 Giới thiệu về SQL Server

2.5.1 Tổng quan về SQL Server

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) do Microsoft phát triển, được thiết kế để lưu trữ và quản lý dữ liệu một cách hiệu quả. Nền tảng này hỗ trợ ngôn ngữ truy vấn Transact-SQL (T-SQL), cho phép người dùng thực hiện các thao tác như tạo, đọc, cập nhật và xóa dữ liệu trong cơ sở dữ liệu. SQL Server được tối ưu để hoạt động trên các môi trường cơ sở dữ liệu lớn, có thể phục vụ hàng ngàn người dùng đồng thời và tích hợp tốt với các sản phẩm khác của Microsoft như Internet Information Services (IIS) và Visual Studio. Hệ thống này cung cấp các tính năng mạnh mẽ như bảo mật nâng cao, khả năng mở rộng và hiệu suất cao, đáp ứng nhu cầu của các ứng dụng doanh nghiệp và hệ thống xử lý dữ liệu phức tạp.

2.5.2 Ưu nhược điểm của SQL Server

Ưu điểm

Hiệu suất mạnh mẽ: SQL Server cung cấp khả năng tối ưu hóa truy vấn và tích hợp với các tính năng mở rộng như tích hợp dữ liệu với SSIS, phân tích dữ liệu với SSAS, và hỗ trợ tạo báo cáo với SSRS.

Quản lý dự án doanh nghiệp: SQL Server là sự lựa chọn hàng đầu cho các doanh nghiệp lớn với các yêu cầu quản lý dự án phức tạp.

Tài liệu học tập phong phú: Với sự phổ biến rộng rãi, SQL Server có nhiều tài liệu học tập, sách và tài liệu trực tuyến, giúp việc học và phát triển dự án trở nên dễ dàng.

Nhược điểm

Phụ thuộc vào hệ điều hành Windows: SQL Server được thiết kế tối ưu với hệ điều hành Windows, điều này có thể giới hạn sự linh hoạt trong việc triển khai trên các hệ điều hành khác.

Quản lý phức tạp: SQL Server có nhiều tính năng, nhưng điều này cũng có nghĩa rằng việc quản lý và cấu hình có thể trở nên phức tạp đối với người mới bắt đầu.

2.6 Giới thiệu cổng thanh toán PayOS

2.6.1 Tổng quan

PayOS là cổng thanh toán đầu tiên tại Việt Nam áp dụng mô hình A2A (Account to Account), cho phép tiền chuyển trực tiếp từ tài khoản người mua đến tài khoản người bán mà không qua trung gian. Cổng thanh toán này phù hợp cho các doanh nghiệp và cá nhân kinh doanh trong nước, đặc biệt là những người muốn giảm chi phí và đơn giản hóa thủ tục thanh toán.

2.6.2 Ưu nhược điểm của cổng thanh toán PayOS

Ưu điểm

Tiết kiệm chi phí: Do không qua các trung gian như VISA hay PayPal, payOS giúp doanh nghiệp giảm đến 99% chi phí so với các cổng thanh toán truyền thống.

Thủ tục đơn giản: Cho phép cá nhân và doanh nghiệp đăng ký và sử dụng dễ dàng, không yêu cầu giấy phép kinh doanh hay hợp đồng phức tạp.

Nhanh chóng: Tiền được chuyển trực tiếp vào tài khoản người bán mà không bị giữ lại ở trung gian, giúp cải thiện dòng tiền và quản lý tài chính hiệu quả hơn.

Hỗ trợ đa kênh: payOS hỗ trợ nhận thanh toán qua nhiều hình thức như ứng dụng điện thoại, web, giúp doanh nghiệp tiếp cận khách hàng trên nhiều nền tảng khác nhau.

Nhược điểm

Hạn chế trong thanh toán quốc tế: Hiện tại, payOS chưa hỗ trợ thanh toán quốc tế, thẻ tín dụng hoặc các hình thức cà thẻ, điều này có thể là hạn chế đối với doanh nghiệp có khách hàng quốc tế.

Phạm vi ngân hàng hỗ trợ: Mặc dù payOS đã liên kết với nhiều ngân hàng lớn tại Việt Nam, nhưng có thể chưa hỗ trợ tất cả các ngân hàng, điều này có thể ảnh hưởng đến khả năng tiếp cận của một số khách hàng.

2.7 Giới thiệu về Docker

2.7.1 Tổng quan về Docker

Docker là một nền tảng mã nguồn mở cho phép đóng gói, triển khai và chạy ứng dụng trong các container. Container là các đơn vị phần mềm nhẹ, độc lập và có thể di chuyển, chứa tất cả mọi thứ cần thiết để chạy ứng dụng, bao gồm mã nguồn, thư viện, công cụ và các phụ thuộc khác. Docker giúp việc triển khai ứng dụng trở nên dễ dàng, nhanh chóng và nhất quán trên mọi môi trường.

Các thành phần chính của Docker

Docker Engine: Phần mềm chạy và quản lý các container.

Docker Images: Các bản sao của môi trường ứng dụng được đóng gói sẵn, có thể tái sử dụng và chia sẻ.

Docker Containers: Các phiên bản chạy của Docker images, giúp cô lập các ứng dụng và môi trường của chúng.

Docker Hub: Một kho chứa Docker images công cộng.

2.7.2 Ưu nhược điểm của Docker

Ưu điểm

Tiết kiệm tài nguyên: Chỉ cần một hệ điều hành duy nhất, giúp giảm thiểu sự lãng phí.

Nhanh chóng và linh hoạt: Containers nhẹ hơn máy ảo (VMs), khởi động và triển khai nhanh hơn đáng kể.

Khả năng mở rộng: Dễ dàng quản lý nhiều ứng dụng mà không làm tăng đáng kể chi phí phần cứng.

Độc lập và nhất quán: Container đảm bảo rằng ứng dụng sẽ chạy giống nhau trên mọi môi trường – từ máy cá nhân, máy chủ cho đến đám mây.

Nhược điểm

Hạn chế trên Windows và macOS: Trên hai hệ điều hành này, Docker không thể chạy trực tiếp mà phải thông qua một máy ảo Linux, điều này có thể ảnh hưởng đến hiệu năng khi sử dụng.

Quản lý trạng thái: Việc quản lý trạng thái và dữ liệu trong container có thể phức tạp, đặc biệt khi cần duy trì dữ liệu lâu dài.

Bảo mật: Mặc dù Docker cung cấp các tính năng bảo mật, nhưng việc cấu hình và quản lý bảo mật đúng cách đòi hỏi kiến thức chuyên sâu.

2.8 Giới thiệu về Docker Compose

2.8.1 Tổng quan về Docker Compose

Docker Compose là một công cụ hỗ trợ xác định và chạy các ứng dụng đa container. Docker Compose có thể định nghĩa tất cả các dịch vụ, mạng và volumes cần thiết cho ứng dụng trong một tệp cấu hình duy nhất (thường là docker-compose.yml). Điều này giúp việc triển khai và quản lý các ứng dụng phức tạp trở nên dễ dàng hơn.

Các tính năng chính của Docker Compose

Định nghĩa dịch vụ: Khai báo các container cần thiết cho ứng dụng.

Quản lý mạng: Tự động tạo và quản lý các mạng giữa các container.

Quản lý volumes: Quản lý dữ liệu và trạng thái của ứng dụng.

Quản lý môi trường: Thiết lập các biến môi trường cho các container.

2.8.2 Ưu nhược điểm của Docker Compose

Ưu điểm

Quản lý ứng dụng đa container: Giúp dễ dàng định nghĩa và quản lý các ứng dụng phức tạp với nhiều dịch vụ.

Tự động hóa: Tự động tạo và kết nối các container, mạng và volumes, giảm thiểu công sức cấu hình thủ công.

Tính nhất quán: Đảm bảo rằng ứng dụng chạy giống nhau trên mọi môi trường, từ phát triển đến sản xuất.

Nhược điểm

Hạn chế về quy mô: Docker Compose phù hợp cho môi trường phát triển và thử nghiệm, nhưng có thể không phù hợp cho các ứng dụng quy mô lớn hoặc phức tạp.

Quản lý trạng thái: Việc quản lý trạng thái và dữ liệu trong các container có thể phức tạp, đặc biệt khi cần duy trì dữ liệu lâu dài.

2.9 Các nghiệp vụ liên quan

2.10 Các công trình nghiên cứu liên quan

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Mô tả bài toán

Trong bối cảnh nhu cầu di chuyển ngày càng tăng cao, việc đặt vé xe khách theo phương thức truyền thống thường gặp nhiều bất cập như mất thời gian, khó khăn trong việc tra cứu thông tin tuyến xe, tình trạng quá tải hoặc thiếu minh bạch trong việc quản lý vé. Những hạn chế này không chỉ gây phiền toái cho hành khách mà còn làm giảm hiệu quả vận hành của các doanh nghiệp vận tải. Vì vậy, bài toán đặt ra là cần phát triển một hệ thống đặt vé xe khách trực tuyến, cho phép người dùng dễ dàng tra cứu thông tin tuyến xe, đặt vé, chọn ghế và thanh toán trực tuyến nhanh chóng, an toàn. Đồng thời, hệ thống cũng phải hỗ trợ nhà xe quản lý lịch trình, số lượng vé đã bán, ghế trống và các báo cáo doanh thu một cách hiệu quả, góp phần cải thiện chất lượng dịch vụ và tối ưu hóa hoạt động kinh doanh.

3.2 Đặc tả các yêu cầu chức năng

3.2.1 Yêu cầu chức năng

Dành cho khách hàng

Đăng ký và đăng nhập: Khách hàng có thể tạo tài khoản, đăng nhập để sử dụng các dịch vụ của hệ thống.

Tra cứu thông tin tuyến xe: Cho phép khách hàng tìm kiếm các tuyến xe dựa trên điểm đi, điểm đến, thời gian khởi hành, loại xe, và hãng xe.

Đặt vé: Cung cấp chức năng chọn tuyến xe, số ghế, và xác nhận đặt vé. Hệ thống phải hiển thị số ghế còn trống theo thời gian thực.

Thanh toán trực tuyến: Tích hợp phương thức thanh toán chuyển khoản và hệ thống tự động xác nhận để người dùng thanh toán nhanh chóng và an toàn.

Quản lý thông tin cá nhân và lịch sử đặt vé: Khách hàng có thể cập nhật thông tin cá nhân, xem và quản lý các vé đã đặt (bao gồm hủy vé hoặc thay đổi lịch trình, nếu cần).

Nhận thông báo: Hệ thống gửi thông báo qua email hoặc tin nhắn về tình trạng vé (đặt thành công, hủy, hoặc thay đổi).

Dành cho quản trị viên

Quản lý tài khoản: Cho phép thêm, chỉnh sửa, hoặc xóa tài khoản quản trị viên hoặc nhân viên hỗ trợ.

Quản lý tuyến xe: Cung cấp chức năng tạo mới, cập nhật, và xóa tuyến xe, bao gồm thông tin như điểm đi, điểm đến, lịch trình, giá vé, và số lượng ghế.

Quản lý vé và ghế ngồi: Hiển thị danh sách vé đã đặt, số ghế còn trống theo từng tuyến xe, cho phép kiểm tra và xác nhận vé.

Báo cáo và thống kê: Hỗ trợ tạo báo cáo doanh thu, số lượng vé bán ra, và tình trạng hoạt động của các tuyến xe theo ngày, tuần, tháng hoặc năm.

Quản lý cổng thanh toán: Cấu hình và theo dõi các giao dịch thanh toán trực tuyến, đảm bảo tính minh bạch và đồng bộ với hệ thống.

Hỗ trợ khách hàng: Cung cấp công cụ trả lời các thắc mắc của người dùng, xử lý các yêu cầu hủy vé hoặc thay đổi lịch trình.

3.2.2 Yêu cầu phi chức năng

Tính bảo mật: Hệ thống phải đảm bảo an toàn cho thông tin cá nhân và giao dịch của người dùng.

Tính khả dụng: Hệ thống cần hoạt động ổn định, đáp ứng lưu lượng truy cập lớn mà không bị gián đoạn.

Bảo đảm hiệu suất: Có khả năng xử lý lượng lớn truy cập cùng lúc, thời gian hệ thống phản hồi nhanh, không bị gián đoạn trong các giờ cao điểm.

Tính dễ sử dụng: Giao diện cần thân thiện, trực quan, dễ dàng sử dụng cho cả người dùng lần đầu.

Khả năng mở rộng: Hệ thống phải hỗ trợ mở rộng để tích hợp thêm các tính năng mới trong tương lai, như kết nối với nhiều hãng xe khác nhau hoặc bổ sung phương thức thanh toán mới.

3.3 Kiến trúc hệ thống

3.4 Thiết kế dữ liệu

3.4.1 Lược đồ cơ sở dữ liệu

3.4.1.1 Mô hình thực thể kết hợp

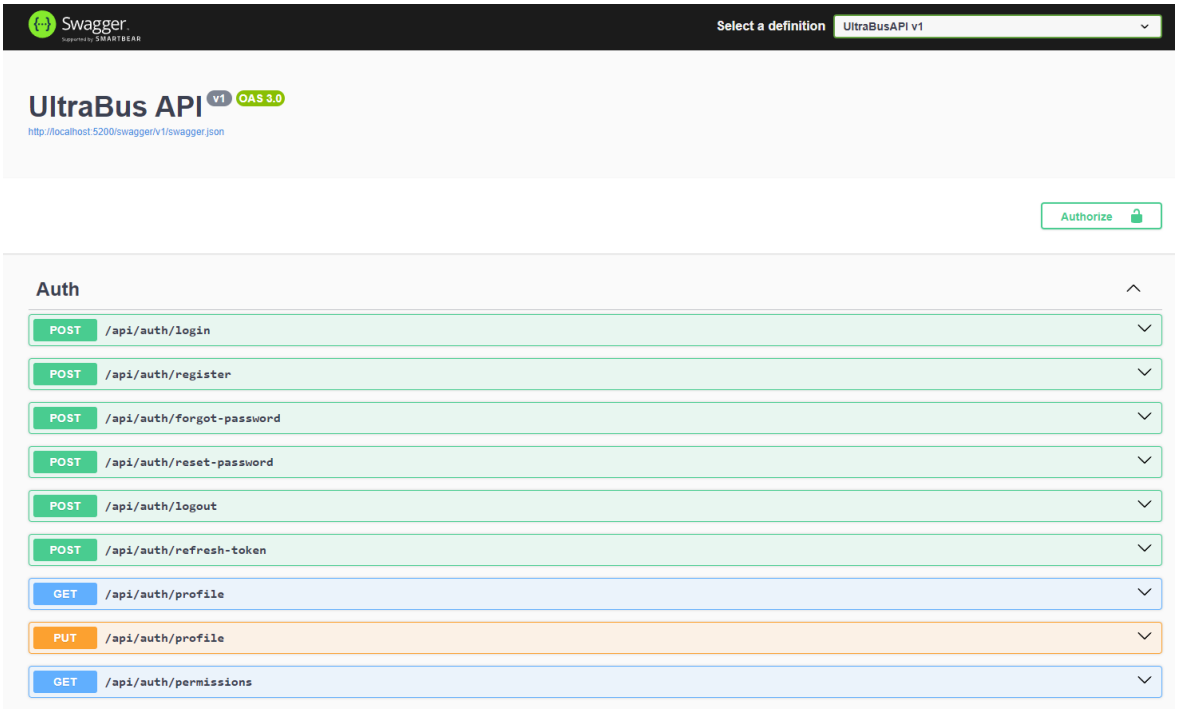
3.4.1.2 Mô hình vật lý

3.4.2 Danh sách các thực thể

3.4.3 Chi tiết các thực thể

CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU

4.1 Các API đã thực hiện



Hình 1 Các API đăng nhập, xác thực và phân quyền

4.2 Giao diện trang chủ

4.3 Giao diện trang quản trị

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

5.2 Hướng phát triển

DANH MỤC TÀI LIỆU THAM KHẢO

PHỤ LỤC