BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ISO 9001:2015

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên: Lê Thành Công

Giảng viên hướng dẫn: TS. Nguyễn Trịnh Đông

HẢI PHÒNG - 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

KẾT HỢP NODEJS VỚI MONGODB CHO BÀI TOÁN XÂY DỰNG HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU THỜI GIAN THỰC

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Lê Thành Công

Giảng viên hướng dẫn : TS. Nguyễn Trịnh Đông

HẢI PHÒNG - 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên : Lê Thành Công **Mã sinh viên :** 1612111012

Lớp : CT2001C

Ngành : Công Nghệ Thông Tin

Tên đề tài : Kết hợp NodeJS với MongoDB cho bài toán xây dựng hệ quản trị cơ sở dữ liệu thời gian thực

NHIỆM VỤ ĐỀ TÀI

1. **Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp**

   - Lập trình NodeJS, AnglurJS, Socket.IO, Express Framework.
   - Hệ quản trị cơ sở dữ liệu MongoDB
   - Kết nối NodeJS, Socket.IO với MongoDB

2. **Các tài liệu, số liệu cần thiết**

   Sử dụng số liệu thực tế thu thập trên mạng internet

3. **Địa điểm thực tập tốt nghiệp**

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên **:** Nguyễn Trịnh Đông

Học hàm, học vị **:** Tiến sĩ

Đơn vị công tác **:** Trường Đại học Quản lý và Công nghệ Hải Phòng

Nội dung hướng dẫn **:**

- Lập trình NodeJS, AnglurJS, Socket.IO, Express Framework.
- Hệ quản trị cơ sở dữ liệu MongoDB
- Kết nối NodeJS, Socket.IO với MongoDB

Đề tài tốt nghiệp được giao ngày 12 tháng 10 năm 2020

Yêu cầu phải hoàn thành xong trước ngày 31 tháng 12 năm 2020

Đã nhận nhiệm vụ ĐTTN	Đã giao nhiệm vụ ĐTTN
Sinh viên	**Giảng viên hướng dẫn**

Hải Phòng, ngày tháng năm 2021

TRƯỞNG KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: Nguyễn Trịnh Đông
Đơn vị công tác: Khoa Công nghệ Thông tin – Trường ĐHQL&CNHP
Họ và tên sinh viên: Lê Thành Công Ngành: Công nghệ Thông tin
Nội dung hướng dẫn:

- Lập trình NodeJS, AnglurJS, Socket.IO, Express Framework.
- Hệ quản trị cơ sở dữ liệu MongoDB
- Kết nối NodeJS, Socket.IO với MongoDB

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp

- Sinh viên chịu khó tìm hiểu kiến thức đã học và liên quan.
- Nghiêm túc thực hiện theo yêu cầu của giáo viên.

2. Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đã đề ra trong nhiệm vụ Đ.T. T.N trên các mặt lý luận, thực tiễn, tính toán số liệu…)

- Sự ứng dụng Hệ quản trị cơ sở dữ liệu MongoDB để phục vụ như một cơ sở dữ liệu thời gian thực có ý nghĩa thực tế cao. Khóa luận đã trình bày phần kiến thức cơ bản trong Chương 1. Chương 2 trình bày kỹ thuật sử dụng NodeJS và Socket.IO kết nối với cơ sở dữ liệu MongoDB để tạo ra cách thức xử lý dữ liệu theo hướng thời gian thực dựa trên sự phản hồi tức thời của hệ thống sử dụng phương pháp này. Chương 3 thử nghiệm với bài toán quản lý xe công ten nơ để minh họa kỹ thuật. Khóa luận đạt yêu cầu đề ra.

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp

Đạt ☐ Không đạt ☐ Điểm:

Hải Phòng, ngày tháng năm 2020
Giảng viên hướng dẫn
(Ký và ghi rõ họ tên)

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên:
…….....

Đơn vị công tác:
……..

Họ và tên sinh viên: …… Ngành: …… ….

Đề tài tốt nghiệp: .....

1. Phần nhận xét của giảng viên chấm phản biện

.....

.....

.....

.....

2. Những mặt còn hạn chế

.....

.....

.....

.....

3. Ý kiến của giảng viên chấm phản biện

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm:.....

Hải Phòng, ngày tháng năm 2021

Giảng viên chấm phản biện
(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên em xin chân thành cảm ơn các thầy, cô trong khoa Công Nghệ Thông Tin cũng như toàn thể mọi người trong ngôi trường Đại học Dân lập Hải Phòng đã tạo điều kiện thuận lợi cho em trong suốt quá trình học tập tại trường cũng như trong thời gian thực hiện đồ án tốt nghiệp.

Đặc biệt, em muốn gửi lời cảm ơn tới Thầy Nguyễn Trịnh Đông giảng viên trực tiếp hướng dẫn tận tình chỉ bảo giúp em khắc phục những khó khăn, thiếu sót để có thể hoàn thành các phần trong đồ án tốt nghiệp từ lý thuyết cho tới thực hành sử dụng công cụ. Với hiểu biết tìm tòi của bản thân và sự chỉ bảo hướng dẫn tận tình của giảng viên em đã cố gắng hoàn thành đồ án một cách tốt nhất có thể nhưng cũng không thể tránh được thiếu sót. Kính mong nhận được sự đóng góp ý kiến từ thầy cô để em có thể nâng cao cũng như bổ sungthêm kiến thức cho bản thân, hoàn thiện đồ án với một kết quả tốt và hoàn chỉnh hơn

Em xin chân thành cảm ơn!

Hải Phòng, ngày tháng năm

Sinh viên thực hiện

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Từ đầy đủ	Diễn giải
HTTP	*Hypertext Transfer Protocol*	HTTP là giao thức truyền tải siêu văn bản. Đây là giao thức tiêu chuẩn cho World Wide Web (www) để truyền tải dữ liệu dưới dạng văn bản, âm thanh, hình ảnh, video từ Web Server tới trình duyệt web của người dùng và ngược lại.
URL	*Uniform Resource Locator*	URL là địa chỉ tài nguyên của một web
HTML	*Hypertext Markup Language*	HTML là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web trên World Wide Web. Cùng với CSS và JavaScript
TCP/IP	*Transmission Control Protocol/ Internet Protocol*	TCP/IP là một bộ giao thức trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet
W3C	*World Wide Web Consutirum*	W3C là một quy chuẩn thiết kế chung được rất nhiều nhà thiết kế web sử dụng như thước đo đánh giá mức độ hoàn thiện của những website đó.
RFC	*Request For Comment*	RFC là tập hợp những tài liệu về kiến nghị, đề xuất và những lời bình luận liên quan trực tiếp hoặc gián tiếp đến công nghệ, nghi thức mạng INTERNET
OSI	*Open Systems Interconnection Reference Model*	OSI là một thiết kế dựa vào nguyên lý tầng cấp, lý giải một cách trừu tượng kỹ thuật kết nối truyền thông giữa các máy tính và thiết kế giao thức mạng giữa chúng

DANH SÁCH CÁC HÌNH

DANH SÁCH CÁC BẢNG

GIỚI THIỆU

Hiện nay, tốc độ khoa học phát triển rất nhanh, đặc biệt trong lĩnh vực Công nghệ Thông tin. Các yêu cầu của các hệ thống phần mềm cần phát triển nhanh, chất lượng tốt, chi phí giá thành giảm, v.v. Lựa chọn hệ quản trị cơ sở dữ liệu là một trong những yếu tố dẫn đến thành công của dự án. Tuy nhiên mỗi loại cơ sở dữ liệu lại có ưu nhược điểm khác nhau, tuy vào bài toán để chọn cơ sở dữ liệu phù hợp. Để đáp ứng yêu tố nhanh, và tức thời trong hệ thống phần mềm, người ta sẽ chọn giải pháp sử dụng cơ sở dữ liệu thời gian thực. Nhưng loại cơ sở dữ liệu này có chi phí vận hành lớn, trong khi đó nhiều dự án chỉ có nguồn kinh phí hạn hẹp.

Dựa trên những hệ quản trị cơ sở dữ liệu hiện tại, MongoDB là một trong những hệ quản trị cơ sở dữ liệu mạnh, mã nguồn mở, tương thích với nhiều hệ điều hành như Windows, Ubuntu, CentOS, v.v. Kết hợp với sự nâng cấp của engine JavaScript phiên bản v8, và platform NodeJS do Google phát triển đã tạo ra bước đột phá cho phép tạo ra nhiều cách cải tiến hiệu năng hệ thống phần mềm. Sự cải thiện tốc độ xử lý và truyền tải dữ liệu nhờ sự nâng cấp lên thành phiên bản HTTP/2.0 năm 2015 làm cho việc phát triển phần mềm trên nền Web ngày càng hiệu quả.

Trên cơ sở các công nghệ phát triển và hướng sử dụng hệ quản trị cơ sở dữ liệu MongoDB có nhiều ưu điểm. Em đã chọn đề tài "**Kết hợp NodeJS với MongoDB cho bài toán xây dựng hệ quản trị cơ sở dữ liệu thời gian thực**." với mong muốn tìm hiểu thêm công nghệ mới để áp dụng cho tương lai nghề nghiệp. Khóa luận có các phần chính được trình bày theo trình tự sau:

Giới thiệu

Chương 1: *Kiến thức nền tảng, chương này tổng hợp các kiến thức cơ bản làm cơ sở lý luận cho các chương tiếp theo.*

Chương 2: *Kết hợp NodeJS với MongoDB, trình bày các yếu tố kỹ thuật kết hợp giữa NodeJS với MongoDB để xử lý dữ liệu có tính phản hồi nhanh.*

Chương 3: *Thử nghiệm hệ thống, trình bày phần áp dụng các kiến thức ở các chương trên thử nghiệm với bài toán quản lý vận tải đơn giản.*

Kết luận

CHƯƠNG 1: KIẾN THỨC NỀN TẢNG

Chương này trình bày các kiến thức nền tàng, tổng hợp các kiến thức cơ bản làm cơ sở lý luận cho các chương tiếp theo cũng như là các phương pháp tiếp cận để giải quyết đề tài.

1.1. Đặt vấn đề

Các hệ thống phần mềm hiện nay đòi hỏi chạy trên đa nền tảng, hiệu quả, tương tác thân thiện với người dùng. Phần mềm phát triển trên nền Web là một trong những loại ứng dụng đáp ứng được các yêu cầu khắt khe đó. Tuy nhiên, các yếu tố quan trọng anh hướng đến hiệu năng của các ứng dụng Web là việc trao đổi dữ liệu dựa trên Internet và cụ thể là trên giao thức HTTP và xử lý dữ liệu trong các hệ quản trị cơ sở dữ liệu. Do khoa học phát triển, những hạn chế kể trên đã được khắc phục. Giao thức HTTP đã được nâng cấp thành HTTP 2.0 truyền hai chiều (bidirection), các hệ quản trị cơ sở dữ liệu hướng đến xử lý dữ liệu theo thời gian thực, đặc biệt là sự ra đời cảu hệ quản trị cơ sở dữ liệu NoSQL (Not only SQL) đã cải thiện tốc độ xử lý dữ liệu đáng kể. Trong đó điển hình là hệ quản trị cơ sở dữ liệu MongoDB. MongoDB là một hệ quản trị cơ sở dữ liệu hướng tài liệu. Nghĩa là dữ liệu không chỉ chứa trong các bảng như cơ sở dữ liệu quan hệ mà được lưu trữ ở dạng JSON. Điều này giúp cho việc truy cập và xử lý nhanh hơn rất nhiều so với các hệ quản trị cơ sở dữ liệu quan hệ. Trong chương này, khóa luận trình bày kiến thức nền tảng liên quan để làm cơ sở cho các nội dung tiếp theo

1.2. Cơ sở dữ liệu hướng tài liệu

Cơ sở dữ liệu hướng tài liệu (Document Oriented Database) là cơ sở dữ liệu lưu trữ dữ liệu một cách tự do không theo một lược đồ nhất định. Mỗi bản ghi không cần phải có cấu trúc cố định, các bản ghi khách nhau có thể có nhiều cột khách nhau. Loại dữ liệu trong mỗi cột bản ghi cũng có thể khác nhau. Một cột có thể có nhiều hơn một mảng và các bản ghi có thể có cấu trúc lồng nhau. Dữ liệu được đóng gói thành từng tài liệu, tài liệu có thể lưu dưới dạng JSON, XML, v.v.

Ví dụ:

```
{
        "_id": ObjectId("5fe87df4ccac7508fc2c52a4")
        "name: "Roanldo"
        "position": "striker"
}
```

<contact>

```
<firstname>Bob</firstname>
<lastname>Smith</lastname>
<phone type="Cell"> (123) 555-0178</phone>
<phone type="Work"> (890) 555-0133</phone>
<address>
 <type>Home</type>
 <street1>123 Back St.</street1>
 <city>Boys</city>
 <state>AR</state>
 <zip>32225</zip>
 <country>US</country>
 </address>

</contact>
```

Khi phát sinh việc chèn dữ liệu, tức là thêm một trường mới thì ta chỉ cần bổ sung một trường vào đối tượng JSON/XML là được chứ nó không cố định như số cột trong bảng của cơ sở dữ liệu quan hệ. Một số hệ quản trị cơ sở dữ liệu hướng tài liệu như MongoDB, CouchDB, Elasticsearch, v.v.

1.3. Giao thức HTTP

1.3.1 Giới thiệu HTTP

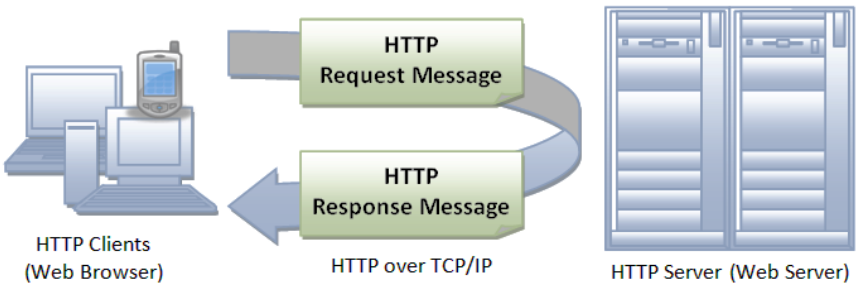
Giao thức HTTP là một trong các giao thức chuẩn sử dụng để trao đổi dữ liệu trên Internet, giao thức này được dùng để trao đổi thông tin giữa bên cung cấp dịch vụ (Web server) và bên sử dụng dịch vụ (Web client) trong mô hình Client/Server. Giao thức HTTP là một giao thức thuộc tầng ứng dụng, nằm trên cặp giao thức tầng giao vận & tầng mạng là TCP/IP.

1.3.2. Lịch sử phát triển

- *Giao thức HTTP và chuẩn HTML được Tim Berners-Lee đề xuất vào năm năm 1989 tại CERN và được các tổ chức IETF và World Wide Web Consortium (W3C) công nhận và đã công bố ra hàng loạt các phiên bản RFC (Request for Comments).*
- *Phiên bản đầu tiên của HTTP là HTTP v0.9 được đưa ra năm 1991.*
- *Vào giữa thập niên 90, David Ragger đã tăng tính bảo mật, mở rộng các thẻ meta-rich decription và thêm các phương thức khác cung với các trường header nhằm mục đích biểu diễn được đa dạng các loại dữ liệu.*
- *Giao thức HTTP từ lúc công bố đến nay đã trải qua nhiều phiên bản 1.x (1.0, 1.1, 1.2,1.3), đến ngày nay là phiên bản 2.x.*

- *HTTP/2 được công bố trong bản RFC 7540 vào tháng 7 năm 2015. [TL1]*

1.3.3. Nguyên lý hoạt động của HTTP



Hình 1. 1: Nguyên lý hoạt động của HTTP

Header: Chứa các thông tin về địa chỉ xuất phát của gói, địa chỉ đích đến và các thông tin như loại dữ liệu, dung lượng dữ liệu.

Payload: Chứa các gói dữ liệu cần được truyền tải.

Footer: Chứa các thông tin dùng để phát hiện và chỉnh sửa lỗi trong quá trình truyền.

Các hệ thống sử dụng giao thức HTTP hoạt động theo nguyên lý **Client – Server**. Theo nguyên lý này các thiết bị đóng vai trò làm máy khách (Client) sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi phản hồi thông tin từ máy chủ. Giao thức HTTP là một giao thức phi trạng thái (stateless protocol). Hay nói cách khác, yêu cầu hiện tại (request) không biết những gì đã xảy ra trong yêu cầu trước đó. HTTP cho phép tạo các yêu cầu gửi và nhận các kiểu dữ liệu, do đó cho phép xây dựng hệ thống độc lập với dữ liệu được truyền.

1.3.4. Uniform Resource Locator (URL)

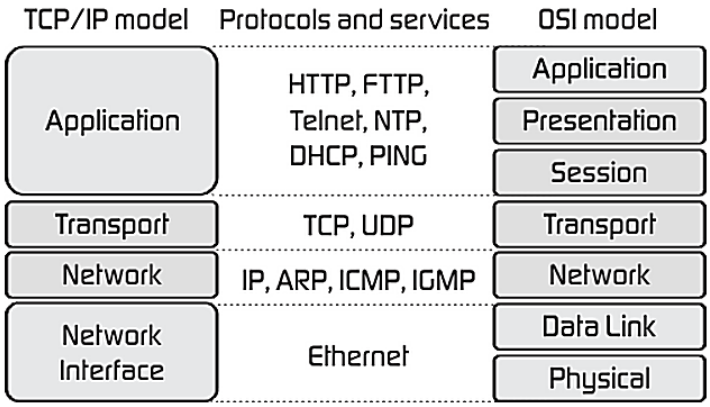
Một **URL** (*Uniform Resource Locator*) được sử dụng để xác định duy nhất một tài nguyên trên Web. Một URL có cấu trúc như sau:

Protocol://hostname:port/path-and-file-name

Trong một URL có 4 thành phần:

- **Protocol**: giao thức tầng ứng dụng được sử dụng bởi client và server
- **Hostname**: tên DNS domain
- **Port**: Cổng TCP để server lắng nghe request từ client
- **Path-and-file-name:** Tên và vị trí của tài nguyên yêu cầu

1.3.5. Giao thức TCP/IP



Hình 1. 2: Kiến trúc giao thức TCP/IP so với OSI

Các giao thức được phân chia thành các tầng, Trong đó **TCP/IP** có 4 tầng mỗi tầng lại sử dụng các giao thức ở tầng dưới để đạt đc mục đích của mình.

- **Layer 1: Network Access Layer**

Network Access Layer: Quy ước về cách thức dữ liệu được gửi qua mạng bởi các thiết bị phần cứng trực tiếp giao tiếp với môi trường mạng, chẳng hạn như cáp đồng trục, cáp quang hay dây đồng xoắn đôi. Các giao thức bao gồm trong Network Access Layer là Ethernet, Token Ring, FDDI, X.25, Frame Relay…

- **Layer 2: Internet Layer**

Internet Layer: Đóng gói dữ liệu vào các gói chúng lại dưới dạng các gói tin thông giao thức **Internet Protocol**, chứa địa chỉ nguồn và đích (địa chỉ logic hoặc địa chỉ IP) được sử dụng để chuyển tiếp các gói tin giữa các máy chủ và qua các mạng.

Mục đích của Transport Layer là cho phép các thiết bị trên máy chủ nguồn và đích đến trao đổi dữ liệu. Transport Layer sẽ xác định mức độ service và trạng thái của kết nối được sử dụng khi vận chuyển dữ liệu.

- **Layer 3: Transport Layer**

Mục đích của **Transport Layer** là cho phép các thiết bị trên máy chủ nguồn và đích đến trao đổi dữ liệu với nhau. Service và trạng thái kết nối được sử dụng khi vận chuyển sẽ được Transport Layer xác định mức độ nào

Khi dùng trình duyệt truy cập Web bạn sẽ thường gặp các thông báo lỗi khác nhau như sau:

- Lỗi 404 hay Http 404 tức là lỗi không tồn tại địa chỉ bạn đang truy cập
- Lỗi 401: lỗi này bạn truy cập vào nơi yêu cầu xác thực, nhưng không vượt qua được sẽ có lỗi này.

- Lỗi 500: lỗi này thường do Web server mà bạn truy cập bị lỗi nên không thể truy cập vào được.
- Ngoài ra Http 200 tức là bạn truy cập thành công.

1.4. Giao thức HTTP 2.0

1.4.1. Giới thiệu HTTP 2.0

HTTP/2 là cuộc cách mạng giao thức truyền siêu văn bản (Hypertext Transfer Protocol) (HTTP) mới nhất tính đến thời điểm này. HTTP là giao thức mạng được sử dụng để yêu cầu và nhận page cùng dữ liệu trên môi trường World Wide Web. Công nghệ mới này đang dần thay thế chuẩn HTTP/1.1 đã được sử dụng rộng rãi trong hơn hai thập kỷ gần đây

1.4.2. Nguyên lí hoạt động

HTTP/1.1 đã ra đời gần 20 năm và với các ứng dụng web (web application) như hiện nay, giao thức này đang trở nên lạc hậu. HTTP/2.0 ra đời với rất nhiều những nâng cấp.

Ghép kênh (Multiplexed)

HTTP có một vấn đề gọi là **head-of-line blocking**, chỉ có phép được thực hiện với mỗi kết nối. HTTP/1.1 đã cố gắng giải quyết vấn đề này bằng các luồng song song **(pipelining)**, nhưng không thể giải quyết triệt để (ví dụ một truy vấn mà bị lỗi không nhận được phản hồi sẽ làm gián đoạn toàn bộ các truy vấn tiếp theo).

Hơn nữa, cơ chế luồng song song (pipelining) cũng rất phức tạp trong vận hành bởi vì cần phải xử lý các truy vấn thật cẩn thận mới đảm bảo được phản hồi tương ứng với truy vấn. Client buộc phải sử dụng một cách chuẩn đoán để xác định cần gửi truy vấn nào vào kết nối nào. Vì thông thường, một trang web cần tới 10 (có thể hơn) các kết nối, nên hiệu suất có thể bị ảnh hưởng nghiêm trọng khi có những truy vấn lỗi.

Multiplexing giải quyết vấn đề này bằng cách cho phép nhiều truy vấn và phản hồi cùng một lúc. Về phía client, chỉ cần một kết nối đến máy chủ là có thể tải toàn bộ dữ liệu cần thiết.

Chính cơ chế gửi và nhận dữ liệu của HTTP/2.0 giúp nó dễ dàng triển khai **multiplexing**. Theo đó, HTTP/2.0 cho phép client và server chia nhỏ dữ liệu thành các frame hoàn toàn độc lập với nhau. Chúng có thể được gửi và nhận song song, xen kẽ nhau và ghép nối lại thành những thông điệp hoàn chỉnh tại đích đến. Điều này giúp việc gửi và nhận dữ liệu cực kỳ hiệu quả mà không hề gặp phải **head-of-line blocking** như HTTP/1.1 vì các frame hoàn toàn độc lập với nhau.

HTTP/2.0 sử dụng dữ liệu nhị phân

Dữ liệu nhị phân rõ ràng là dữ liệu dễ xử lý nhất đối với máy tính. HTTP/1.1 sử dụng dữ liệu dạng văn bản (Nó sẽ được mã hoá bởi máy tính, và tiếp tục được mã hoá ở những tầng dưới). Dữ liệu nhị phân cho phép chúng ta phân tích nó dễ dàng hơn, nó dễ được nén hơn và quan trong nhất là nó không dư thừa. Dữ liệu dang văn bản của HTTP/1.1 thường xuyên có những thứ không cần thiết như dấu cách, các dấu xuống dòng…

Cũng vì vậy với HTTP/2.0, chúng ta có một phương thức duy nhất để phân tích dữ liệu, trong khi HTTP/1.1 có tới 4 cách khác nhau. HTTP/1.1 sử dụng văn bản nên một gói tin của nó chứa rất nhiều thông tin và bắt buộc phải sử dụng dấu xuống dòng để ngăn cách. HTTP/2.0 thì hoàn toàn khác, mọi giao tiếp của client và máy chủ đều được chia thành các thông điệp và đơn vị truyền tin (frame), tất cả chúng đều là dữ liệu binary. Có nhiều loại frame khác nhau như HEADER, DATA, SETTINGS…

Bằng việc sử dụng dữ liệu nhị phân, chúng ta có một số khái niệm sau trong giao tiếp HTTP/2.0

- Stream: Một dòng dữ liệu hai chiều trong một kết nối, trong 1 stream có thể có nhiều thông điệp (message)
- Message (thông điệp): Một chuỗi các frame (đơn vị truyền tin) có liên quan đến nhau để hoàn thành một truy vấn hoặc phản hồi.
- Frame (Đơn vị truyền tin): Đây là đơn vị nhỏ nhất trong giao tiếp HTTP/2.0, như đã nói ở trên, có nhiều loại frame khác nhau để hoàn thiện một message.

Mọi kết nối HTTP/2.0 đều thực hiện qua TCP, trên đó giao tiếp được thực hiện hai chiều nhờ vào stream. Mỗi stream thì đều có định danh và độ ưu tiên nhờ đó nó có thể truyền tải các thông điệp giữa client và server một cách chính xác. Mỗi một thông điệp là một gói tin HTTP hoàn chỉnh (ví dụ là truy vấn hoặc phản hồi từ máy chủ), thông điệp sẽ có nhiều đơn vị truyền tin (frame). Ví dụ frame header chứa các HTTP header, frame data chứa dữ liệu chính cần truyền tải. Đơn vị truyền tin truyền tải các thông điệp khác nhau và các stream khác nhau có thể được truyền tải xen kẽ nhau và sẽ được ghép nối khi chúng đến đích nhờ vào định danh của chúng được gắn với các thông điệp và stream. Chính nhờ cơ chế truyền tải dữ liệu nhị phân này mà HTTP/2.0 cho phép chúng ta gửi và nhận nhiều thông tin hơn trong cùng một kết nối.

HTTP/2.0 có cơ chế server push

Khi trình duyệt truy vấn một trang web, máy chủ sẽ trả về HTML và sau đó trình duyệt phải phân tích HTML nay và nếu cần thì tiếp tục truy vấn để tải về JS, CSS, v.v...

Server push cho phép máy chủ không cần chờ truy vấn từ trình duyệt và tự quyết định những gì là quan trọng (thường dựa vào cache) và push luôn cho trình duyệt trước.

Vậy tại sao lại cần server push? Như đã nói ở trên, một trang web có trung bình 80 assets, vậy việc giảm thiểu độ trễ khi phải phân tích rồi gửi yêu cầu truy vấn những thứ này sẽ giảm rất nhiều thời gian tải trang. Server hiểu rất rõ về những thứ client cần và nó push luôn cho client, vậy là khỏi mất công phải chờ.

HTTP/2.0 thực hiện push một cách tự động nên nó mang lại nhiều lợi ích hơn. Ví dụ có thể dùng cache ở phía client, chuyển trang không cần tải lại dữ liệu này, các assets khác nhau có thể multiplexing làm chúng được tải nhanh hơn. Tuy nhiên, tính năng này nếu bị lạm dụng, nó sẽ phản tác dụng khiến hiệu suất giảm đi. Làm thế nào để sử dụng một cách đúng đắn thì cũng là một vấn đề khá khó.

HTTP/2.0 nén các header

Mỗi một kết nốt HTTP đều phải có header. HTTP là một giao thức phi trạng thái. Vì vậy, mọi kết nối đều là một cặp gửi - nhận truy vấn. Mỗi kết nối đều phải gửi kèm các thông tin liên quan đến kết nối đó (kể cả nó giống hệt các kết nối khác) trong header.

Khi được tiêu chuẩn hoá vào năm 2005, có 116 header khác nhau. Phần lớn chúng là những yêu cầu bắt buộc phải gửi trong truy vấn cho các trang web (mà rất nhiều thứ là thông tin bị lặp), vì vậy, HTTP/2.0 sử dụng HPACK để nén các header.

HTTP/1.1 gửi các dưới dạng text (văn bản) cho mỗi truy vấn hay phản hồi. Và nhiều trong số chúng là giống nhau khi chúng ta truy cập một trang web. Cơ chế nén header của HTTP/2.0 cho phép chúng ta giảm bớt khá nhiều thông tin dư thừa nhờ phương thức sau:

- Sử dụng Huffman code để mã hoá các header, nhờ đó mà giảm được kích thước gói tin.
- Yêu cầu client và server chia sẽ một bảng chỉ mục các header đã được gửi và nhận từ trước, nhờ đó mà những header lặp lại sẽ được bỏ qua.

Huffman code cho phép các giá trị được nén trước khi gửi đi, và bảng chỉ mục các header đã được gửi từ trước cho phép chúng ta tránh những dữ liệu dư thừa bằng cách chỉ cần gửi định danh của các header này và cả client và server có thể lấy ra giá trị của chúng đã được lưu từ trước đó.

Mỗi truy vấn cần trung bình 1400 byte cho header và một trang web có trung bình 80 assets, mỗi assets cần một truy vấn riêng biệt. Tổng cộng phải mất đến 1MB cho riêng header để tải trang. Vì vậy rõ ràng nén được lượng dữ liệu này sẽ có ích rất lớn.

Lúc đầu, HTTP/2.0 sử dụng **gzip** để nén, tuy nhiên, phương thức này có một lỗ hổng **Compression Ratio Info-leak Made Easy** (CRIME) nên HTTP/2.0 chuyển sang dùng phương thức nén khác an toàn hơn.

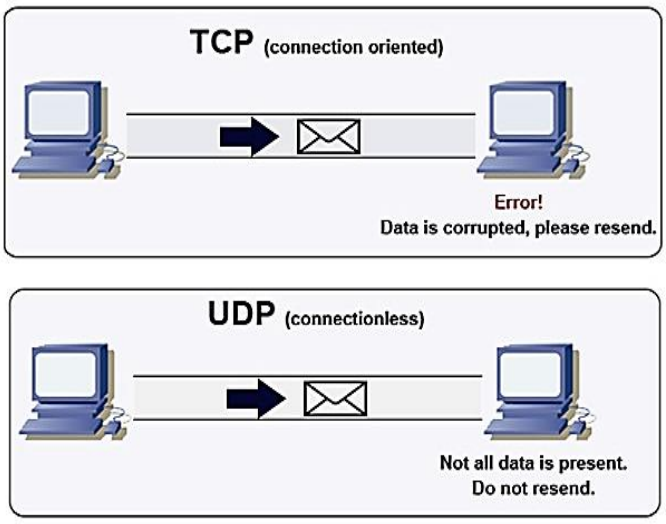
1.5. WebSocket

1.5.1. Giới thiệu Socket

Socket là một điểm cuối (end-point) của liên kết giao tiếp hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 tiến trình nói chuyện với 1 tiến trình khác.

Các lớp Socket được sử dụng để tiến hành kết nối giữa client và server. Nó được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể xác định ứng dụng mà dữ liệu sẽ được gửi tới

1.5.2. Nguyên lí hoạt động của Socket



Hình 1. 3: Sơ đồ hoạt động của Socket trong việc truyền nhận dữ liệu

Socket giúp lập trình viên kết nối các ứng dụng để truyền và nhận dữ liệu trong môi trường có kết nối Internet bằng cách sử dụng phương thức TCP/IP và UDP.

Khi cần trao đổi dữ liệu cho nhau thì 2 ứng dụng cần phải biết thông tin IP và port bao nhiêu của ứng dụng kia. Có rất nhiều dạng socket khác nhau phụ thuộc vào sự khác biệt giữa cách truyền dữ liệu (protocol). Dạng phổ biến nhất là TCP và UDP.

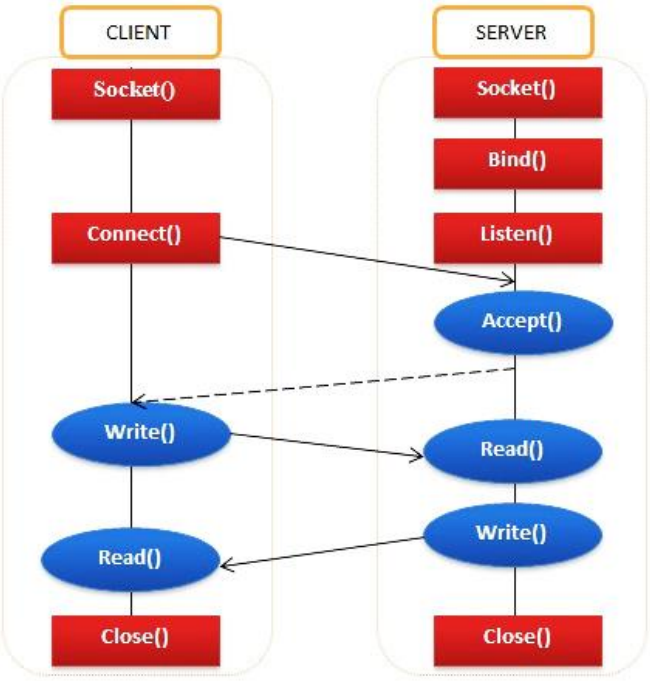
2 ứng dụng cần truyền thông tin phải đáp ứng điều kiện sau thì socket mới có thể hoạt động:

- Ứng dụng có thể nằm cùng trên một máy hoặc 2 máy khác nhau
- Trong trường hợp 2 ứng dụng cùng nằm trên một máy, số hiệu cổng không được trùng nhau.

1.5.3. Phân loại Socket

Stream Socket

Dựa trên giao thức TCP (Tranmission Control Protocol) stream socket thiết lập giao tiếp 2 chiều theo mô hình client và server. Được gọi là socket hướng kết nối.



Hình 1. 4: Stream Socket

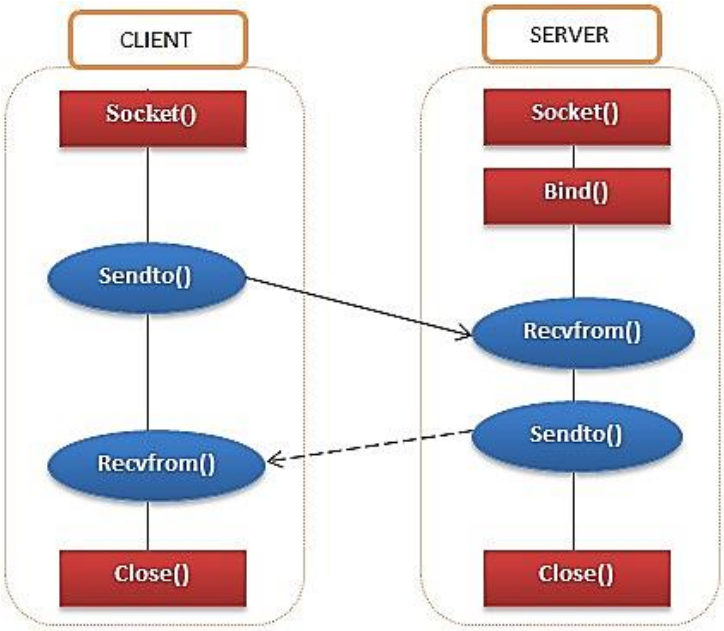
Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng tuần tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn

Đặc điểm tóm gọn:

- Có một đường kết nối (địa chỉ IP) giữa 2 tiến trình.
- Một trong hai tiến trình kia phải đợi tiến trình này yêu cầu kết nối.
- Mô hình client /sever thì sever lắng nghe và chấp nhận từ client.
- Mỗi thông điệp gửi phải có xác nhận trả về.
- Các gói tin chuyển đi tuần tự.

Datagram Socket

Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 tiến trình. Tức là nó cung cấp phương thức phi kết nối cho việc gửi và nhận gói tin. Gọi là socket không hướng kết nối



Hình 1. 5: Datagram Socket

Do không yêu cầu thiết lập kết nối, không phải có những cơ chế phức tạp. Nên tốc độ giao thức khá nhanh, thuận tiện cho các ứng dụng truyền dữ liệu nhanh như chat, game online…

Đặc điểm:

- Hai tiến trình liên lạc với nhau không kết nối trực tiếp
- Thông điệp gửi đi phải kèm theo thông điệp người nhận
- Thông điệp có thể gửi nhiều lần
- Người gửi không chắc chắn thông điệp đến tay người nhận.
- Thông điệp gửi sau có thể đến trước và ngược lại.

- Để có thể thực hiện các cuộc giao tiếp, một trong 2 quá trình phải công bố port của socket mà mình đang sử dụng.

1.5.4. Giới thiệu Web Socket

Websocket là giao thức hỗ trợ giao tiếp hai chiều giữa client và server để tạo một kết nối trao đổi dữ liệu. Giao thức này không sử dụng HTTP mà thực hiện nó qua TCP. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

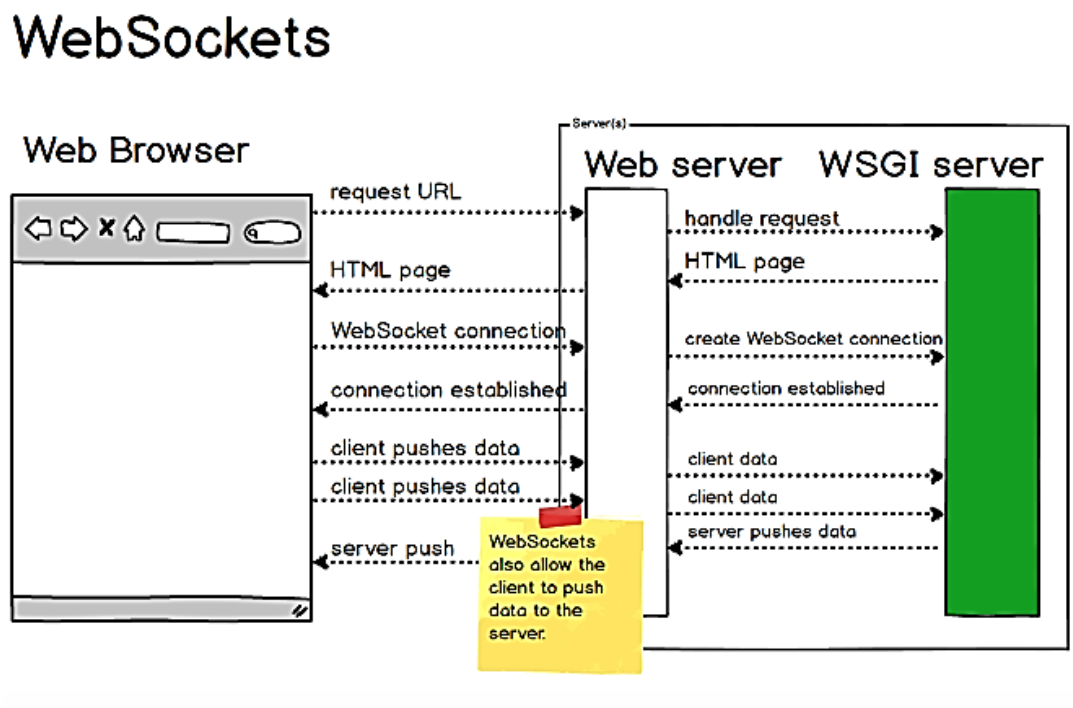
Ưu điểm:

- Tăng tốc độ truyền tải thông tin giữa 2 chiều
- Dễ phát hiện và xử lý trong trường hợp có lỗi xảy ra
- Dễ dàng sử dụng, không cần cài đặt thêm các phần mềm bổ sung khác
- Không cần sử dụng nhiều phương pháp kết nối khác nhau

Nhược điểm:

- Chưa hỗ trợ trên tất cả các trình duyệt
- Với các dịch vụ có phạm vi yêu cầu, Websocket chưa hỗ trợ hoàn toàn.

1.5.5. Cấu trúc của Web Socket



Hình 1. 6: Sơ đồ hoạt động của WebSockets

Giao thức chuẩn thông thường của WebSocket là ws://, giao thức secure là wss://. Chuẩn giao tiếp là String và hỗ trợ buffered arrays và blobs.

1.5.6. Các thuộc tính của WebSocket

THUỘC TÍNH	MÔ TẢ
readyState	Diễn tả trạng thái kết nối. Nó có các giá trị sau: • Giá trị 0: kết nối vẫn chưa được thiết lập (WebSocket.CONNECTING) • Giá trị 1: kết nối đã thiết lập và có thể giao tiếp (WebSocket.OPEN) • Giá trị 2: kết nối đang qua handshake đóng (WebSocket.CLOSING) • Giá trị 3: kết nối đã được đóng (WebSocket.CLOSED)
bufferedAmount	Biểu diễn số byte của UTF-8 mà đã được xếp hàng bởi sử dụng phương thức send()

Bảng 1. 1 : Bảng thuộc tính của WebSocket

```
switch (socket.readyState) {
  case WebSocket.CONNECTING:
    // phần code
    break;
  case WebSocket.OPEN:
    // phần code
    break;
  case WebSocket.CLOSING:
    // phần code
    break;
  case WebSocket.CLOSED:
    // do something
    break;
  default:
    // phần code
    break;
}
```

1.5.7. Các sự kiện WebSocket

SỰ KIỆN	EVENT HANDLER	MÔ TẢ
open	onopen	Khi một WebSocket chuyển sang trạng thái mở, "onopen" sẽ được gọi.
message	onmessage	Khi WebSocket nhận dữ liệu từ Server.
error	onerror	Có bất kỳ lỗi nào trong giao tiếp.
close	onclose	Kết nối được đóng. Những sự kiện được truyền cho "onclose" có ba tham số là "code", "reason", và "wasClean".

Bảng 1. 2: Bảng các sự kiện WebSocket

Event handlers có thể được tạo ra bằng cách sử dụng phương thức **addEventListener()**.

Ví dụ:

Onopen

```
socket.onopen = function(event) {

};
```

Sử dụng **addEventListener()**

Onmessage

```
socket.addEventListener("open", function(event) {

});
```

Sử dụng **addEventListener()**

```
socket.addEventListener("message", function(event) {
  var data = event.data;
});
```

Onerror

```
socket.onerror = function(event) {

};
```

Sử dụng **addEventListener()**

```
socket.addEventListener("error", function(event) {

  // handle error event

});
```

Onclose

```
socket.onclose = function(event) {
  var code = event.code;
  var reason = event.reason;
  var wasClean = event.wasClean;
  // handle close event
};
```

Thực hiện handler Onclose sự kiện sử dụng **addEventListener()**

```
socket.addEventListener("close", function(event) {
  var code = event.code;
  var reason = event.reason;
  var wasClean = event.wasClean;
  // handle close event
});
```

1.5.8. Các phương thức của WebSocket

PHƯƠNG THỨC	MÔ TẢ
send()	send(data) gửi dữ liệu tới server. Message data là string, ArrayBuffer, blob.
close()	Đóng kết nối đang tồn tại.

Bảng 1. 3: Bảng phương thức của WebSocket

Ví dụ:

```
var data = new ArrayBuffer(1000000);

socket.send(data);

if (socket.bufferedAmount === 0) {
```

```
  // Gửi dữ liệu
 }
 else {
  // Gửi dữ liệu thất bại
}
```

1.6. MongoDB

1.6.1. Giới thiệu MongoDB

MongoDB là hệ CSDL mã nguồn mở, là CSDL phi quan hệ hay còn gọi là NoSQL (None-Relationship SQL hay còn gọi là Not only SQL). NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value. NoSQL ra đời như là sự bổ sung cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng, khả năng mở rộng. Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc. NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng. NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn. Ví dụ như FaceBook sử dụng Cassandra do FaceBook phát triển, Google phát triển và sử dụng BigTable. MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.

Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là **collection** thay vì bảng. So với RDBMS thì trong MongoDB **collection** ứng với table, còn **document** sẽ ứng với **row**, MongoDB sẽ dùng các **document** thay cho row trong RDBMS.

Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định. Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

1.6.2. Một số câu lệnh cơ bản trên MongoDB

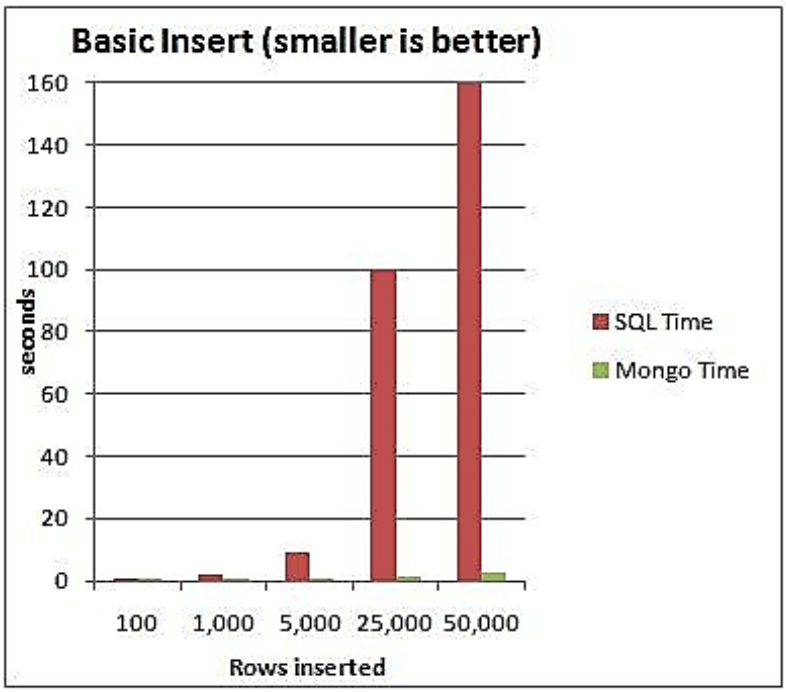
Mục này so sánh một số câu lệnh giữa MongoDB và MySQL để làm nổi bật cách xử lý dữ liệu của MongoDB.

Lệnh	MySQL	MongoDB
Tạo csdl	CREATE DATABASE test;	use test;
Tạo bảng	CREATE TABLE students (ten_cot - kieu_du_lieu);	db.createCollection('students');
Tạo bản ghi	INSERT INTO studetns ('name', 'gender') VALUES('thanh', 'male');	db.students.insert({ name:'thanh', gender: 'male'});
Cập nhật	UPDATE students SET name = 'thanh update' WHERE id = 1;	db.students.update({ _id: 1 },{$set:{ name: 'thanh update' }});
Xóa bản ghi	DELETE FROM students Where id = 1;	db.students.remove({ _id: 1});
Tìm kiếm all	SELECT * FROM students;	db.students.find({});
Tìm kiếm	SELECT * FROM students WHERE name = 'thanh';	db.students.find({ name: 'thanh' });

Bảng 1. 4: Bảng câu lệnh cơ bản trên MongoDB

1.6.3. Ưu điểm của MongoDB

- Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, do đó việc chèn thêm dữ liệu vào không bị hạn chế.
- Dữ liệu trong MongoDB không bị ràng buộc như cơ sở dữ liệu quan hệ, không kết nối. Do đó, khi bổ sung, xóa hay cập nhật sẽ không bị mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
- MongoDB rất dễ mở rộng. Trong MongoDB có một khái niệm cụm dữ liệu (cluster), là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster.
- Trường dữ liệu "_id" luôn được tự động đánh chỉ mục để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
- Khi có một truy vấn dữ liệu, các bản ghi được ghi tạm thời lên bộ nhớ Ram, để phục vụ các truy vấn tiếp theo, vì vậy sự xử lý cũng diễn ra nhanh hơn mà không cần phải đọc lại từ ổ cứng.
- Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn khi thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.



Hình 1. 7: So sánh thời gian chèn dữ liệu của MongoDB với SQL

1.6.4. Nhược điểm của MongoDB

Hệ quản trị cơ sở dữ liệu MongoDB có nhiều ưu điểm, tuy nhiên cũng có một số nhược điểm. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với MongoDB thì chuyên gia phải tự xử lý các mối quan hệ giữa các dữ liệu. Bộ nhớ lưu trữ bị tăng, do dữ liệu lưu dưới dạng key-value, các bộ dữ liệu chỉ khác về giá trị do đó các Khóa sẽ bị lặp lại. MongoDB cũng không hỗ trợ liên kết nên dữ liệu bị dư thừa. Một điều nữa, khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau một khoảng thời gian 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống thiết bị lưu trữ, điều này sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện.

1.6.5. Các ứng dụng cần MongoDB

Dù MongoDB có nhiều ưu điểm, nhưng tùy vào quy mô hệ thống để lựa chọn. Ví dụ như các hệ thống yêu cầu phản hồi nhanh, Các hệ thống dữ liệu lớn với yêu cầu truy vấn nhanh hay các hệ thống có lượng request lớn thì MongoDB sẽ là sự lựa chọn ưu tiên hơn CSDL quan hệ. Tùy theo dự án và trường hợp cụ thể để sử dụng CSDL quan hệ hay sử dụng MongoDB đem lại hiệu quả cao.

1.7. NodeJs

1.7.1. Giới thiệu

NodeJS là một platform mới, xây dựng trên Chrome Javascript Runtime (V8) nhằm phát triển các ứng dụng phía máy chủ nhanh chóng và dễ mở rộng. Một số điểm mạnh của NodeJS:

1.7.2. Những ứng dụng nên viết bằng Nodejs

Rõ ràng, không phải cứ hot và mới là Nodejs làm gì cũng tốt, ví dụ như một ứng dụng cần tính ổn định cao, logic phức tạp thì các ngôn ngữ PHP hay Ruby… vẫn là sự lựa chọn tốt hơn. Còn dưới đây là những ứng dụng có thể và nên viết bằng Nodejs:

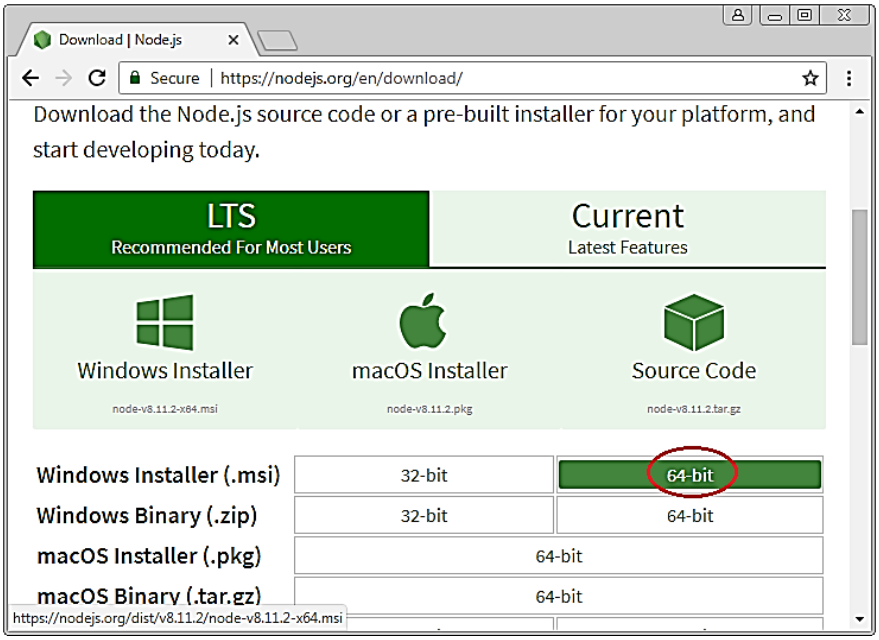
- **Websocket server**: Các máy chủ web socket như là Online Chat, Game Server…
- **Fast File Upload ClientL:** là các chương trình upload file tốc độ cao.
- **Ad Server**: Các máy chủ quảng cáo.
- **Cloud Services**: Các dịch vụ đám mây.
- **RESTful API**: đây là những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.
- **Any Real-time Data Application**: bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực. Micro Services: Ý tưởng của micro services là chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ và kết nối chúng lại với nhau. Nodejs có thể làm tốt điều này.

1.7.3. Cài đặt NodeJs

Download NodeJs

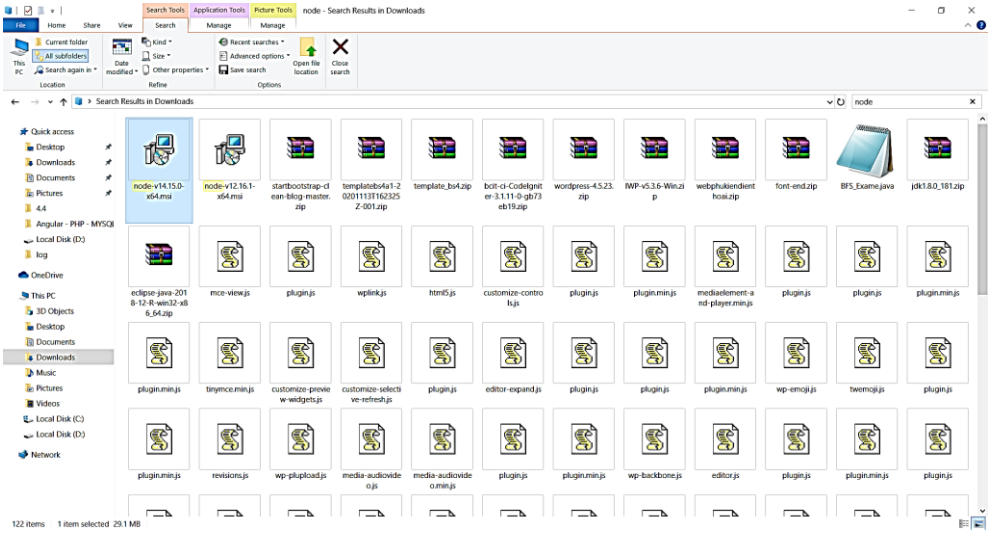
Để download NodeJS bạn cần truy cập vào địa chỉ dưới đây:

https://nodejs.org/en/download/



Hình 1. 8: Trang chủ NodeJS

Sau khi download thành công bạn có một file

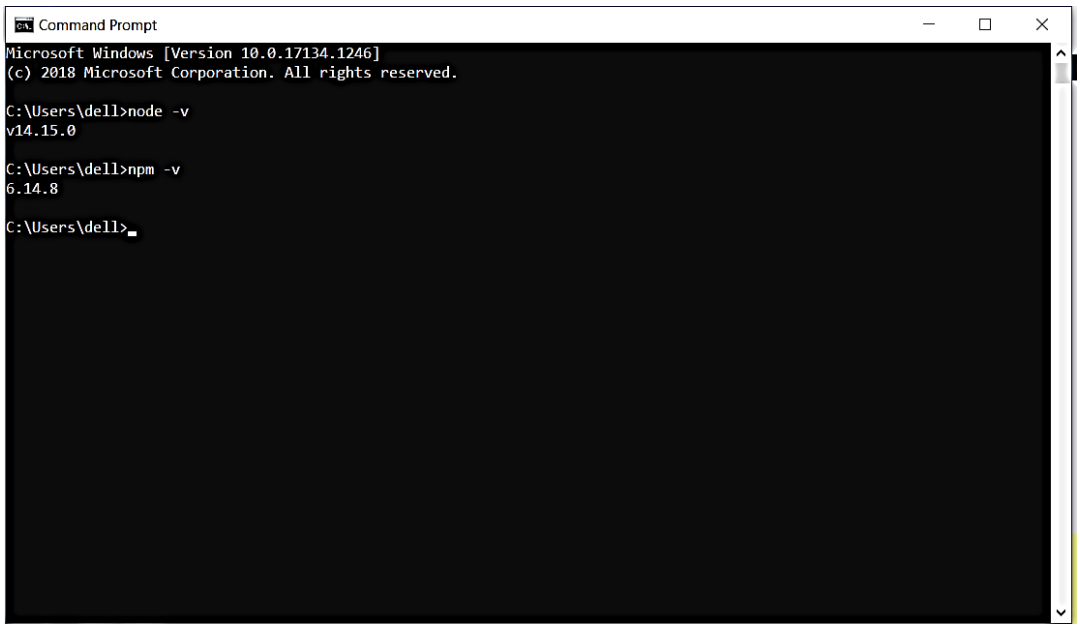


Hình 1. 9: Chọn file cài đặt

Cài đặt NodeJS trên Windows rất đơn giản, chấp nhận các tùy chọn mặc định và nhấn "**Next … Next**" cho tới bước cuối cùng

Theo mặc định, phần mềm NPM cũng được cài đặt vào hệ thống của bạn. Đây là một phần mềm quản lý các thư viện Javascript

Kiểm tra và cấu hình



Hình 1. 10:Chọn file cài đặt

1.8. Express

1.8.1. Giới thiệu Express

Express là một framework giành cho Nodejs. Nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web cũng như trên các ứng dụng di động. Express hỗ rợ các phương thức HTTP và midleware tạo ra môt API vô cùng mạnh mẽ và dễ sử dụng. Có thể tổng hợp một số chức năng chính của express như sau:

- Thiết lập các lớp trung gian để trả về các HTTP request
- Định nghĩa router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL
- Cho phép trả về các trang HTML dựa vào các tham số.

1.8.2. Cài đặt Express

Để cài đặt express, vào Terminal và gõ lệnh sau:

$ mkdir express

$ cd express

$npm install express --save

Với câu lệnh trên sẽ lưu phần cài đặt trong thư mục ***node_modules*** và tạo thư mục ***express*** trong đó. Cần cài đặt thêm một số **module** quan trọng đi cùng **express** như sau:

- body-parser: Đây là lớp trung gian, xỷ lý JSON, text và mã hóa URL.

- cookie-parser: Chuyển đổi header của cookie và phân bố đến các req.cookies
- multer: Xử lí phần multipart/form-data

Để cài đặt các module trên, lần lượt chạy các lệnh:

$ npm install body-parser --save

$ npm install cookie-parser --save

$ npm install multer –save

1.9. Resful API

1.9.1. Giới thiệu RestFul API

API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML

REST: REST là viết tắt của REpresentational State Transfer. REST là tập hợp các hướng dẫn và kiến trúc sử dụng cho việc truyền dữ liệu. REST áp dụng phổ biến cho các Web app, nhưng cũng hoàn toàn có thể sử dụng cho các phần mềm nói chung. Vì thế RESTful API là các API mà tuân theo các kiến trúc và quy tắc của REST

1.9.2. Đặc điểm của Resful API

- Nhất quán xuyên suốt các API
- Stateless existence
- Sử dụng HTTP status code khi có thể
- Sử dụng URL Endpoint có phân tầng logic
- Đánh version trong URL thay vì trong HTTP Headers

Phương thức truy cập:

- GET để lấy về 1 đối tượng
- POST để tạo ra đối tượng mới
- PUT để sửa đổi hoặc thay thế đối tượng
- DELETE để xoá đi đối tượng
- Tương tự như CRUD: Create, read, update, delete

1.10. Angurlar Js

1.10.1. Giới thiệu Angular

AngularJS là một framework có cấu trúc cho các ứng dụng web động. Nó cho phép bạn sử dụng HTML như là ngôn ngữ mẫu và cho phép bạn mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng của bạn một cách rõ ràng và súc tích

Hai tính năng cốt lõi: Data binding và Dependency injection của AngularJS loại bỏ phần lớn code mà bạn thường phải viết. Nó xảy ra trong tất cả các trình duyệt, làm cho nó trở thành đối tác lý tưởng của bất kỳ công nghệ Server nào.

Angular hoạt động là dạng đơn trang, sử dụng API để lấy data, nên cầm nắm vững kĩ thuật DHTML, AJAX. Dữ liệu ở dạng JSON hoặc là XML. Phát triển dựa trên JavaScript.

Khả năng tương thích cao, tự động xử lý mã javascript để phù hợp vứi mỗi trình duyệt. Tạo ứng dụng client - serve theo mô hình MVC. Mã nguồn mở, miễn phí hoàn toàn và được sủ dụng rộng rãi.

1.10.2. Các tính năng cơ bản

- Scope: là đối tượng có nhiệm vụ giao tiếp giữa controller và view của ứng dụng.
- Controller: xử lí dữ liệu cho đối tượng $scope, từ đây bên views sẽ sử dụng các dữ liệu trong scope để hiển thị ra tương ứng.
- Data-binding: tự động đồng bộ dữ liệu giữa model và view
- Service: là singleton object được khởi tạo 1 lần duy nhất cho mỗi ứng dụng, cung cấp các phương thức lưu trữ dữ liệu có sãn. ($http, $httpBackend, $sce, $controller, $document, $compile, $parse, $rootElement, $rootScope…)
- Filter: Lọc các tập con từ tập item trong các mảng và trả về các mảng mới.
- Directive: dùng để tạo các thẻ HTML riêng phục vụ những mục đích riêng. AngularJS có những directive có sẵn như ngBind, ngModel…
- Temple: một thành phần của view, hiển thị thông tin từ controller
- Routing: chuyển đổi giữa các action trong controller, qua lại giữa các view.
- MVC & MVVM: mô hình thiết kế để phân chia các ứng dụng thành nhiều phần khác nhau (gọi là Model, View và Controller) mỗi phần có một nhiệm vụ nhất định. AngularJS không triển khai MVC theo cách truyền thống, mà gắn liền hơn với Model-View-ViewModel.
- Deep link: Liên kết sâu, cho phép bạn mã hóa trạng thái của ứng dụng trong các URL để nó có thể bookmark với công cụ tìm kiếm. Các ứng dụng có thể được phục hồi lại từ các địa chỉ URL với cùng một trạng thái.
- Dependency Injection: AngularJS có sẵn một hệ thống con dependency injection để giúp các lập trình viên tạo ra các ứng dụng dễ phát triển, dễ hiểu và kiểm tra

Các Components chính

- ng-app: định nghĩa và liên kết một ứng dụng AngularJS tới HTML.
- ng-model: gắn kết giá trị của dữ liệu ứng dụng AngularJS đến các điều khiển đầu vào HTML.
- ng-bind: gắn kết dữ liệu ứng dụng AngularJS đến các thẻ HTM

Ưu điểm

- Cung cấp khả năng tạo ra các Single Page Aplication dễ dàng.
- Cung cấp khả năng data binding tới HTML, khiến cho người dùng cảm giác linh hoạt, thân thiện.
- Dễ dàng Unit test
- Dễ dàng tái sử dụng component

Nhược điểm

- Không an toàn: được phát triển từ javascript cho nên ứng dụng được viết bởi AngularJS không an toàn. Nên có sự bảo mật và xác thực phía server sẽ giúp ứng dụng trở nên an toàn hơn.
- Nếu người sử dụng ứng dụng của vô hiệu hóa JavaScript thì sẽ chỉ nhìn thấy trang cơ bản.

CHƯƠNG 2: KẾT HỢP NODEJS VỚI MONGODB

Chương này của đồ án tập trung vào phần giới thiệu cơ sở dữ liệu thời gian thực. Trình bày kĩ thuật kết hợp NodeJs với MongoDB thành cơ sở dữ liệu như thời gian thực và được trình bay ở các phần tiếp theo

2.1 Cơ sở dữ liệu thời gian thực

2.1.1 Giới thiệu về cơ sơ dữ liệu thoi gian thực

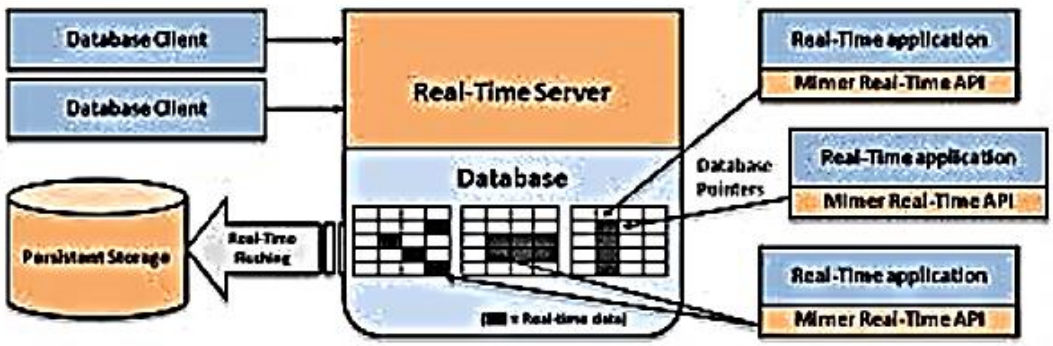
Trong thời kì phát triển Web nổi trội như hiện này. Cụm từ realtime chắc không còn quá xa lạ với mọi người. Thời gian thực (realtime) ám chỉ rằng ứng dụng phản hồi lại lại người dùng một cách tức thời mà người dùng không cần phải refesh lại

Một cơ sở dữ liệu thời gian thực là một hệ thống cơ sở dữ liệu trong đó sử dụng thời gian thực để xử lý khối lượng công việc cho các bài toán đang có thay đổi liên tục về dữ liệu. Ví dụ các hệ thống bán hàng có các sản phẩm có giá trị thay đổi liên tục như vàng, dầu mỏ... hoặc các loại dữ liệu khác. Xử lý thời gian thực có nghĩa là một giao dịch được xử lý đủ nhanh và cho kết quả phản hồi và có tác động ngay lập tức.

Với việc truyền tải thông tin đến người dùng một cách tức thời thì thời gian thực (realtime) là tính năng không thể thiếu trong hầu hết các ứng dụng yêu thích hiện nay. Có thể nói đến một số các lĩnh vực áp dụng cơ sở dữ liệu thời gian thực như: Facebook có thông báo (Notification) trên ứng dụng điện thoại hay như Uber với hiệu ứng dò tìm địa điểm ngay lập tức, tính năng đa người dùng của Google Docs. Hoặc là các trang thương mại điện tử có thể kiểm tra đơn hàng của mình 1 cách tức thì khi sản phẩm đến tay người mua

Các cơ sở dữ liệu thời gian thực là các công cụ quản lý hữu ích cho các đối tượng như cán bộ kế toán, các ngân hàng, hệ thống pháp luật, các hồ sơ y tế, hệ thống đa phương tiện, hệ thống kiểm soát quá trình, hệ thống dự phòng thảm hoạ, và hệ thống phân tích dữ liệu khoa học

Điều này khác với cơ sở dữ liệu truyền thống chứa những dữ liệu bền vững, chủ yếu không bị ảnh hưởng bởi thời gian. Ví dụ, một thị trường chứng khoán thay đổi rất nhanh chóng và gần như liên tục



Hình 2. 1: Minh họa kiến trúc cơ sở dữ liệu thời gian thực

2.1.2 So sánh cơ sở dữ liệu thời gian thực và cơ sở dữ liệu truyền thống

Sự khác biệt giữa cơ sở dữ liệu truyền thống và cơ sở dữ liệu thời gian thực

Cơ sở dữ liệu thời gian thực thực chất là một cơ sở dữ liệu truyền thống nhưng lại được cộng thêm có những phần mở rộng để cung cấp thêm nhiều chức năng giúp cho các cơ sở dữ liệu này có thể đưa ra những phản hồi đáng tin cậy. Một hệ thống có hiệu quả cần phải có khả năng xử lý các truy vấn thời gian gần như tức thời, trả về tạm thời những dữ liệu hợp lệ.

Cơ sở dữ liệu thời gian thực sử dụng cơ chế hạn chế thời gian để thể hiện một phạm vi nhất định của giá trị mà các dữ liệu được coi là đúng. Phạm vi này được gọi là giá trị thời gian, trong khi đó một cơ sở dữ liệu thông thường không thể làm việc trong những trường hợp này các dữ liệu mà đại diện bị ràng buộc. Ví dụ sản phẩm là vàng có thuộc tính giá trị biểu diễn bởi đồng đô la, nhưng đồng đô la lại chịu ảnh hưởng lên xuống liên tục do các yếu tố khác tác động dẫn đến giá vàng nếu muốn chính xác thì phải liên tục cập nhật theo tỷ giá của đồng đô la. Các cơ sở dữ liệu truyền thống có sử ràng buộc về dữ liệu nên khi có nhiều trường thay đổi thì cần phải tạo thêm rất nhiều bảng.

2.1.3 Một số ứng dụng

- Giám sát hệ thống phần mềm: máy ảo, container, dịch vụ, ứng dụng
- Giám sát hệt thống vật lý: Thiết bị, máy móc, thiết bị kết nối, môi trường, nhà cửa
- Ứng dụng theo dõi tài sản: Xe, xe tải
- Hệ thống giao dịch tài chính: Chứng khoán, tiền điện tử
- Ứng dụng tổ chức sự kiện: Theo dõi dữ liệu tương tác của khách hàng
- Và nhiều ứng dùng khác

2.2 Sử dụng MongoDB như cơ sở dữ liệu thời gian thực với NodeJS

Hiện nay ứng dụng Web đã phát triển rất mạnh mẽ khác xa với những ngày đầu nó xuất hiện. Bây giờ muốn có một trang web thu hút thì phải có giao diện bắt mắt, nội dung hay, hiệu ứng độc lạ và rất nhiều yếu tố khác nữa. Yếu tố phản hồi nhanh chóng cũng là một trong những yếu tố được rất nhiều người dùng quan tâm, kèm theo đó là vô số các kỹ thuật mới được áp dụng để phục vụ cho quá trình này nhằm đem lại trải nghiệm mới mẻ, đầy hứng thú và cũng không kém phần tiện dụng cho người dùng. Công nghệ web thời gian thực (realtime) ngày càng trở nên phổ biết. Có nhiều công nghệ, phương pháp giúp xây dựng ứng dụng thời gian thực:

- AJAX LONG-POLLING:
- SERVER SENT EVENTS (SSE)
- COMET
- SOCKET.IO

Để xây dựng một ứng dụng realtime thì chúng ta cần phải sử dụng Socketio. Socketio có khả năng giúp những bên có địa điểm khác nhau có thể thực hiện dễ dàng hơn các kết nối với nhau để có thể truyền tải được dữ liệu một cách nhanh chóng và lập tức nhất thông qua hệ thống server trung gian

2.2.1 Thư viện SocketIO

Giới thệu về SocketIO

Socket.io là một module trong Node.js được phát triển vào năm 2010. Nó được phát triển để sử dụng các kết nối mở để tạo điều kiện giao tiếp thời gian thực, trả về giá trị thực ở tại thời điểm đó. Socket.io cho phép giao tiếp hai chiều giữa máy khách và máy chủ. Giao tiếp hai chiều được bật khi máy khách có Socket.io trong trình duyệt và máy chủ cũng đã tích hợp gói Socket.io

Socket.io xây dựng dựa trên Engine.IO, đầu tiên nó sẽ thiết lập một kết nối long-polling, sau đó cố gắng nâng cấp lên các kết nối khác tốt hơn giống như WebSocket.

Ngoài Socket.io chúng ta còn có một vài kết nối khác như:

Trong long-polling, client sẽ gửi yêu cầu giống AJAX đến máy chủ. Với mỗi lần nhận được yêu cầu, máy chủ sẽ gửi phản hồi lại nếu & khi có cập nhật mới. Tại đây, clients sẽ liên tục & định kỳ yêu cầu cập nhật từ máy chủ, thông qua các kết nối TCP riêng biệt, làm tắc nghẽn lưu lượng mạng.

Trong short-polling, clients định kỳ gửi yêu cầu đến máy chủ để hỏi xem có gì mới không. Máy chủ không đợi, nhưng gửi lại nếu có cập nhật hoặc chỉ có tin

nhắn trống. Ở đây, mạng thậm chí còn tắc nghẽn hơn với các yêu cầu liên tục này, ngay cả khi không có bản cập nhật.

Trong WebSockets, sẽ luôn có một kết nối TCP giữa clients và server. Có luồng dữ liệu hai chiều giữa clients và server cũng như tính chất thời gian thực do luôn kết nối TCP mở. Trong các phương thức, có tiềm năng rất lớn để tăng tốc độ trong WebSockets. Dung lượng phần header của giao thức HTTP là 100 byte, trong khi phần header của socket chỉ là 2 byte. Vì vậy, sau khi sử dụng HTTP ban đầu, Sockets có thể giao tiếp với tài nguyên ít hơn nhiều. Với nhiều số lượng yêu cầu được gửi đến thì nó cũng sẻ làm tăng thời gian phản hồi từ server tới clients.

Cài đặt SocketIO

Để cài đặt Socket.io trong dự án của mình ban cần phải cài đặt ở 2 phía đó là server và client. Socket.io sẽ đảm nhận kết nối giữa 2 phía, thông thường các API của 2 phia sẽ tương tự giống nhau.

2.2.2 So sánh MongoDb với Firebase

Giới thiệu Firebase

Firebase không chỉ là một cơ sở dữ liệu. Đây là một giải pháp hoàn chỉnh được sử dụng để xây dựng web và ứng dụng di động. Google sở hữu Dịch vụ phụ trợ theo thời gian thực này. Cơ sở dữ liệu thời gian thực của Google Firebase hoàn hảo cho các ứng dụng cần xử lý dữ liệu trong thời gian thực trên nhiều thiết bị.

Dịch vụ cơ sở dữ liệu của Firebase được gọi là Cloud Firestore. Nó hoạt động gần như thời gian thực, tìm nạp các thay đổi từ cơ sở dữ liệu của bạn khi chúng xảy ra. Firestore là một phần của các dịch vụ Cloud Firebase, có nghĩa là nó hoạt động hoàn hảo với tất cả các sản phẩm Firebase khác.

Ưu điểm của Firebase:

- Thư viện khách hàng mạnh mẽ
- Hỗ trợ đầy đủ cho chế độ ngoại tuyến
- Bộ quy tắc bảo mật toàn diện
- Công cụ duyệt dữ liệu dễ sử dụng

MongoDB

MongoDB được phát triển và quản lý bởi MongoDB Inc. Không giống như Firebase cung cấp một hệ sinh thái dịch vụ hoàn chỉnh, MongoDB chỉ là một cơ sở dữ liệu hướng văn bản (rất mạnh). Nhắc tơi MongoDB là nhắc tới tính mở rộng và tính linh hoạt, 2 yếu tố được xem xét khi phát triển MongoDB. Nó cung cấp khả năng truy vấn mạnh mẽ, mặc dù chỉ cung cấp tập trung vào một dịch vụ tập trung

vào việc lưu trữ dữ liệu nhưng MongoDB vẫn được sử dụng rộng rãi vì khả năng lưu trữ mạnh mẽ mà nó mang lại. Đối với các nhà phát triển việc sử dụng MongoDB như được tiếp thêm sức mạnh trong việc phát triển ứng dụng. MongoDB đáp ứng nhu cầu phát triển của họ để ứng dụng lưu trữ dữ liệu hiệu quả.

Tên	Firebase Realtime Database	MongoDB
Mô tả	Kho tài liệu thời gian thực được lưu trữ trên đám mây. Ứng dụng khách iOS, Android và JavaScript chia sẻ một phiên bản Cơ sở dữ liệu thời gian thực và tự động nhận các bản cập nhật với dữ liệu mới nhất.	Một trong những kho lưu trữ tài liệu quan trọng nhất có sẵn ở Cloud Serivce. Được quản lí và triển khai hoàn toàn trên Cloud Service
Cơ sở dữ liệu chính	Document store	Document store
Cơ sở dữ liệu phụ		Search engine
Website và tài liệu	firebase.google.com/-products/realtime-database firebase.google.com/docs/-database	www.mongodb.com docs.mongodb.com/manual
Nhà phát triển	Google	MongoDB, Inc
Năm phát hành	2012	2009
Phiên bản phát hành hiện tại		4.4.2, November 2020
License	commercial	Open Source

Chỉ dựa trên cloud	có	không
Được viết bơi ngôn ngữ		C++
Server operating systems	hosted	Linux OS X Solaris Windows
Data scheme	schema-free	schema-free
Typing	Có	Có
Hỗ trợ XML	Không	
Secondary indexes	Có	Có
SQL	Không	Read-only SQL queries via the MongoDB Connector for BI
	Android iOS JavaScript API RESTful HTTP API	Giao thức độc quyền sử dụng JSON
Các ứng dụng	Firebase phù hợp nhất cho các ứng dụng quy mô nhỏ	MongoDB hoàn hảo cho các ứng dụng quy mô lớn
Hỗ trợ các ngôn ngữ lập trình	Java JavaScript Objective-C	Actionscript , C, C#, C++, Clojure, ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Scala, Smalltalk
Server-side scripts	Giới hạn chức năng khi sử dụng 'rules'	JavaScript
Triggers	Được gọi khi dữ liệu thay	Có

	đổi	
Phân vùng		Sharding
Replication		Master-slave
MapReduce	Không	Có
Khái niệm nhất quán	- Sự nhất quán cuối cùng mạnh với CRDT. - Tính nhất quán cuối cùng.	- Tính nhất quán cuối cùng. - Tính nhất quán ngay lập tức.
Foreign keys	Không	Không
Transaction concepts	Có	
Concurrency	Có	Có
Durability	Có	Có
In-memory		Có
Phương thức người dùng	Kiểm soát truy cập dựa trên mật khẩu đơn giản.	Quyền truy cập cho user và role

Bảng 2. 1: So sánh thành phần MongoDB với Firebase

Các công ty hàng đầu sử dụng MongoDB:

- Adobe
- SEGA
- eBay
- Trò chơi
- Verizon
- eHarmony

Các công ty hàng đầu sử dụng Firebase:

- Venmo
- Lyft
- Duolingo
- Thời báo New York
- Alibaba
- Shazam

2.3 Sử dụng thư viện SocketIO xây dựng ứng dụng cơ sở dữ liệu thời gian thực

Trên nền tảng NodeJS, thiết lập các thông số để cài đặt hệ thống sao cho kết nối giữa client với server và MongoDb.

2.3.1 Thiết lập cấu hình

Khởi tạo hệ thống: Khởi tạo hệ thống bằng câu lệnh:

npm init

NodeJs sẽ tạo ra một file **package.json.** Tệp tin này ghi cấu hình cần thiết cho ứng dụng để có thể chạy được trên server, các thông tin tối thiểu như phiên bản, tên, description… Mọi thông tin trong package.json được viết dưới định dạng JSON.

Cài đặt Express:

npm install --save express

Khi express được cài đặt vào hệ thống, ứng dụng sẽ tạo ra một thư mục chứa các thư viện của express. Nếu hệ thống cần thêm các thư viện khác, thì các thư viện sẽ được bổ sung vào trong thư mục này.

Cài đặt mongoose:

npm install –save mongoose

Cài đặt socket.io ở Server

npm install --save socket.io

Cài đặt socket.io ở Client

Ta chỉ cần import thư viện của Socket.IO bởi vì khi tạo Socket.IO ở server thì thư viện này đã được cài tự động rồi

<script src="/socket.io/socket.io.js"></script>

Cài đặt MongoDb

MongoDB có một hàm được gọi là **change streams** cho phép bạn lắng nghe cơ sở dữ liệu của mình**.** Đầu tiền bạn cần phải chuyển đổi phiên bản mongo cục bộ của mình thành mongo relicate sets (bản sao). Chỉ cần mở của số dòng lệnh và chạy lệnh sau

$ mongod - cổng 27017 --replSet rs0

Lệnh trên bắt đầu phiên bản mongo dưới dạng có replicaSettên là rs0.

Bây giờ với mongo đang chạy như replicaSet, trước khi tạo cơ sở dữ liệu mới, chúng ta phải khởi tạo bộ nhân bản.

Để làm như vậy, hãy mở một cửa sổ mới và thực hiện như sau

$ mongo

$ rs.initiate ()

Khởi tạo HTTP Server kết hợp Socket.io

File điều khiển chính ở đây là index.js. Trong file index.js chúng ta sẽ khởi tạo như sau

```
var express = require("express");
var app = express();
app.set("view engine", "ejs");
app.set("views", "./views");
var server = require("http").Server(app);
var io = require("socket.io")(server);
// Chỉ ra đường dẫn chứa css, js, images...
app.use(express.static("public"));
  //Tạo router
  app.get("/", function (req, res) {
    res.sendFile(path.join(__dirname + '/views/index.html'));
});
// Tạo Socket
io.on("connection", function(socket){
        console.log('Welcome to server');
        socket.on('send', function (data) {
          io.sockets.emit('send', data);
   });
})
// Khởi tạo 1 server listen tại 1 port
server.listen(3000);
```

Trong file này có 4 phần chính như đã ghi chú

Đầu tiên là sẽ khởi tảo HTTP Server bằng cách kết nối phần Express và kết nối phần Socket.io. Cài đặt phần view engine để hệ thống biết được mình đang sử dụng thư viện ejs.

Các file mà khách hàng truy cập sẽ ở trong Public bằng cách sử dụng hàm express.static. Thư mục view sẽ là thư mục chưa phần layout của dự án

Tạo Route là app.get để hiển thị dữ liệu. Tất cả các file cần hiển thị sẽ khai báo vào hàm app.get

Cuối cùng là phần tạo Socket sử dụng function **Connection** của Socket để bắt sự kiện khi có 1 client kết nối đến server và thực hiện hiển thị ra một thông điệp gì đó. Sẽ có một sự kiện gửi từ client lên server cụ thể ở đây là send. Server sẽ nhận được thông điệp từ client và gửi data sang các kết nối khác của server. Đây cũng là phần thời gian thực của Socket

Tạo thư mục Model trong dự án

Tạo một thư mục có tên là Models, chúng ta sẽ tiến hành khai báo các thuộc tính của UserDb (Collection) và xuất ra tên bảng trong cơ sở dữ liệu

```
var mongoose = require("mongoose");
var Schema = mongoose.Schema;
var UserSchema = new mongoose Schema({
  name: {
    type: String,
    required: [true, 'Need User name']
  }
});
module.exports = mongoose.model("User", UserSchema);
```

Giả sử bây giờ có 2 User ở trong User Collections của UserDB và muốn tương tác UserDB đối với bất cứ thay đổi nào trên User Collections. Trong index**.js** chuỗi kết nối sẽ như thế này:

```
request('./models/connection').connectMongoDB('mongodb://localhost/User DB? replicaSet = rs0');
```

Yêu cầu kết nối từ các bảng đã được khởi tạo trong thư mục Models tới cơ sở dữ liệu Mongo. Ta sẽ tiến hành request các connection đến các bảng tạo ở trong thư mục Models đến cơ sở dữ liệu MongoDB cục bộ

Trong đó:

- Mongodb:// là định nghĩa giao thức
- Localhost/UserDB là phần cơ sở dữ liệu muốn kết nối tới
- *replicaSet = rs0' là là tên của ReplicaSet mà đang kết nối*

Chuyển dbURL sang connection modul của mình

```
const socket_io = require('socket.io');
var io = socket_io();
const User = require('../models/User');
const changeStream = User.watch();
changeStream.on('change', (change) => {
   console.log(change); // Hiển thị thay đổi
   io.emit('changeData', change);
});
io.on('connection', function () {
   console.log('connected');
});
var socket = io;
module.exports = socket
```

Tiến hành khai báo các bảng (collection) trong cơ sở dữ liệu bằng cách chỉ ra các đường dẫn đi vào thư mục có chứa các tập tin chưa thuộc tính của bảng. Lắng nghe bất cứ dữ liệu thay đổi nào trong User bằng phương thức Watch().

Trình nghe sự kiện Change được cung cấp bởi phương thức ChangStream. Nó sẽ bị gọi lại dữ liệu khi có bất kí tham số nào thay đổi. Để kiểm tra xem nó có hoạt động hay không

CHƯƠNG 3: THỬ NGHIỆM HỆ THỐNG

Để thử nghiệm các kỹ thuật được trình bày trong chương hai. Chương ba khóa luận chọn bài toán quản lý vận tải để triển khai.

Quản lý đội xe vận tải là một công việc gặp rất nhiều khó khăn, phức tạp. Cùng với phương pháp quản lý đội xe thủ công, thiếu minh bạch của điều phối, tài xế và các bộ phận khác trong doanh nghiệp. Từ đó tạo nên chi phí vận hành lớn, thất thoát doanh thu, khó kiểm soát vận chuyển hàng hóa.

Quản lý toàn diện bao gồm việc bảo trì, giám sát xe, giám sát nhiên liệu, quản lý tài xế để đảm bảo sự an toàn cho tài xế và xe, giúp làm giảm đến mức tối thiểu rủi ro về đầu tư cũng như quá trình vận hành xe

Đảm bảo chi phí tổng thể cho quá trình vận chuyển luôn giữ mức ổn định tối thiểu. Giám sát xe hiệu quả bằng việc hỗ trợ giám sát qua hệ thống GPS để biết chính xác thời gian vận hành, vị trí hiện tại của đội xe từ đó lên kế hoạch vận chuyển hợp lí nhất…

Quản lý tổng quan các hoạt động của tài xế một cách cụ thể nhất như: thái độ phục vụ, chạy quá tốc độ, nghỉ quá giờ quy định, quá thời hạn đăng kiểm xe…

Cung cấp cho nhà quản lý những thông tin cốt yếu trong việc bảo trì xe cũng như can bằng hiệu quả kinh tế: tổng số nhiên liệu đã sử dụng, số km đã đi, mức dầu, mức nước, doanh thu từng xe, từng chuyến, theo các mốc thời gian.

3.1 Phát biểu bài toán

Một công ty có một đội xe có 8 chiếc xe vận tải rơ moóc. Khi nhận được đơn hàng, quản lý sẽ tiến hành làm lệnh điều xe và tiến hành điều phối xe với thông tin tuyến đường cần đi. Mỗi một đầu xe sẽ được ghép vào với rơ mooc để hoàn tất một xe tải. Sau đó sẽ tiến hành bàn giao xe cho nhân viên. Tại một thời điểm lái xe sẽ được bàn giao một xe và được bắn vùng theo các thông tin tuyến đương như ở trên lệnh điều xe đã duyệt

3.2. Xác định yêu cầu của hệ thống

Từ dữ liệu thu thập được trong quá trình khảo sát, có thể xác định được hệ thống gồm những chức năng chính sau đây:

3.2.1. Yêu cầu phi chức năng

- Giao diện hài hòa, dễ sử dụng, thân thiện với người dung.
- Bảo mật tương đối tốt, dễ bảo trì.

3.2.2. Yêu cầu chức năng:

- Yêu cầu hệ thống Quản lý tài khoản, đăng nhập
- Yêu cầu nghiệp vụ Quản lý nhân viên lái xe, đầu xe, mooc xe, tuyến đường, lệnh điều xe, điều khiển xe

3.3. Xác định các tác nhân, các UC sử dụng và biểu đồ UC

3.3.1. Các tác nhân

- Quản lý
- Người dùng
- Người quản trị hệ thống

3.3.2. Các UseCase sử dụng

Đăng nhập

UC1	Đăng nhập
1	Đăng nhập hệ thống
2	Thoát khỏi hệ thống
3	Đăng kí tài khoản mới

Bảng 3. 1: Bảng use case đăng nhập

Quản lí đầu xe

UC2	Quản lý đầu xe
1	Nhập đầu xe
2	Sửa đầu xe
3	Xóa đầu xe

Bảng 3. 2: Bảng use case quản lý đầu xe

Quản lý mooc xe

UC3	Quản lý mooc xe
1	Nhập mooc xe
2	Sửa mooc xe
3	Xóa mooc xe

Bảng 3. 3: Bảng use case của quản lý mooc xe

Quản lý tuyến đường

UC4	Quản lý tuyến đường
1	Nhập tuyến đường
2	Sửa tuyến đường
3	Xóa tuyến đường

Bảng 3. 4: Bảng use case của quản lý tuyến đường

Quản lý lệnh điều xe

UC5	Quản lý lênh điều xe
1	Nhập lệnh điều xe
2	Sửa lệnh điều xe
3	Xóa lệnh điều xe

Bảng 3. 5: Bảng use case quản lý lệnh điều xe

Quản lý lái xe

UC6	Quản lý lái xe
1	Nhập thông tin lái xe
2	Sửa thông tin lái xe
3	Xóa thông tin lái xe

Bảng 3. 6: Bảng use case quản lý lái xe

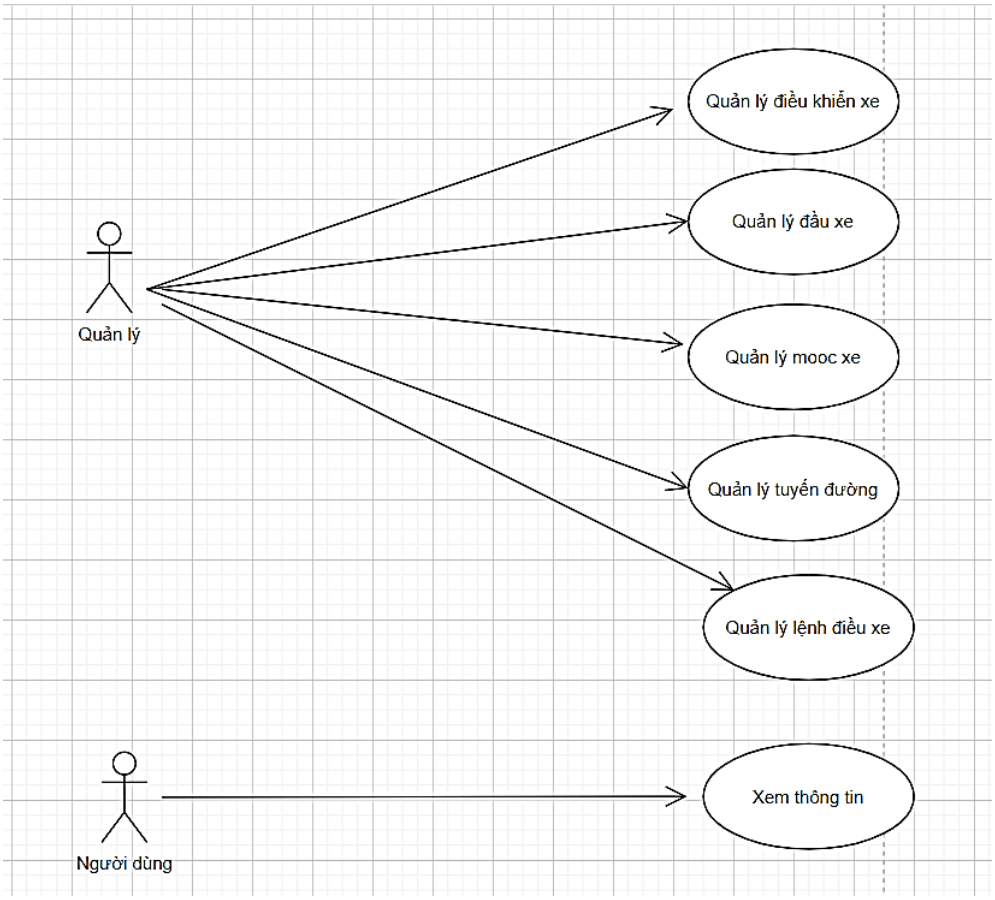
Quản lý điều khiển xe

UC6	Quản lý điều khiển xe
1	Nhập thông tin điều khiển xe
2	Sửa thông tin điều khiển xe
3	Xóa thông tin điều khiển xe

Bảng 3. 7: Bảng use case quản lý điều khiển xe

3.4. Biểu đồ các use case

3.4.1. Biểu đồ use case tổng quát



Hình 3. 1: Biểu đồ Use Case tổng quát

Tên usecase: xây dựng phần mềm quản lí xe rơ mooc

Tên tác nhận: Quản lý, quản trị hệ thống, người dùng

Chức năng use case: Biểu đồ này cho thấy các use case sẽ tương tác với nhau thông qua phần mềm quản lí xe container

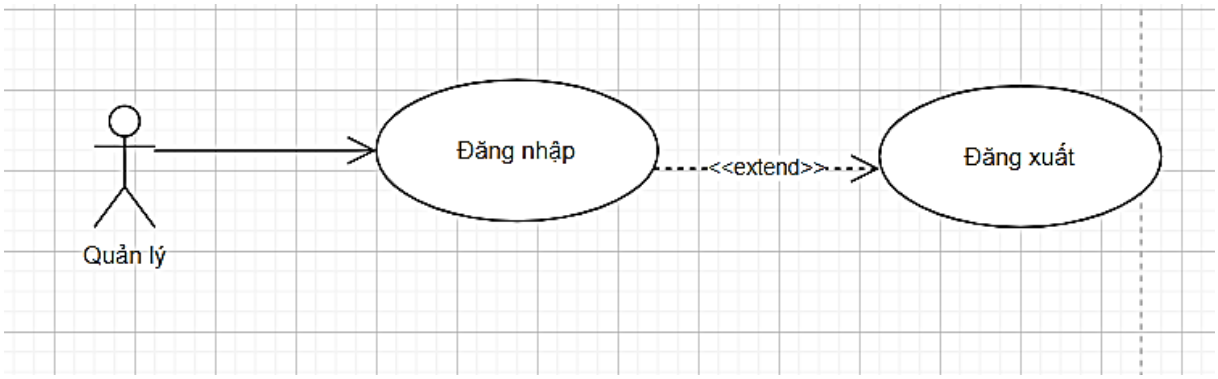
Dòng sự kiện chính:

- Quản lý được lựa chọn tất cả các chức năng mình được phân trên hệ thống
- Người dung xem những thông tin được phân
- Hệ thống sẽ kiểm tra tính hợp lệ và xác nhận
- Thông tin sẽ được lưu vào cơ sở dữ liệu
- Hệ thống sẽ báo thành công
- Người dùng thoát khỏi chức năng khi ấn vào nút “Thoát”

Dòng sử kiện phụ:

- Nếu thông tin người dùng nhập vào không đúng thì hệ thông sẽ báo là không hợp lệ
- Hệ thống sẽ yêu cầu nhập lại thông tin

3.4.2. Biểu đồ Use case đăng nhập



Hình 3. 2: Biểu đồ Use Case đăng nhập

Đặc tả use case:

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn chức năng đăng nhập vào hệ thống	
	2. Hiện thị giao diện đăng nhập
3. Nhập thông tin đăng nhập	
	4. Kiểm tra thông tin và xác nhận

Bảng 3. 8: Bảng đặc tả use case đăng nhập

Tên use case: đăng nhập

Tác nhận: quản lý

Chức năng use case: cho phép quản lí đăng nhập vào hệ thống

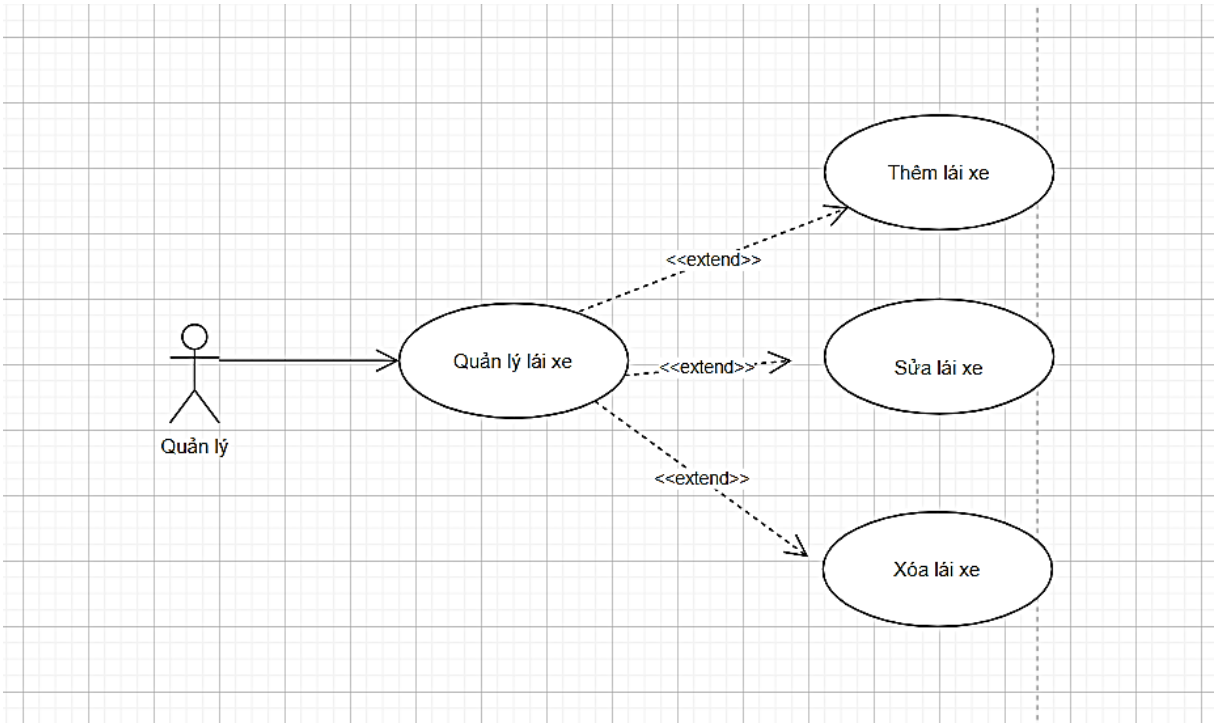
Dòng sự kiện chính:

- Người dùng đăng nhập vào Website
- Hệ thống yêu cầu người dùng đăng nhập
- Người dùng nhập Tên đăng nhập và Mật khẩu
- Hệ thống kiểm tra tính hợp lệ thông tin đăng nhập của người dùng và cho phép người dùng đăng nhập vào hệ thống
- Nếu người dùng mà chưa nhập thông tin đăng nhập mà ấn vào nút đăng nhập thì hệ thống sẽ phản hồi lại là không hợp lệ

Dòng sự kiện phụ:

- Người dùng mà nhập sai tài khoản và mật khẩu thì website sẽ báo lỗi và yêu cầu người dùng đăng nhập lại, nếu người dùng không đăng nhập được thì không vào được website

3.4.3. Biểu đồ Use case quản lý lái xe



Hình 3. 3: Biểu đồ Use Case quản lý xe

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn quản lý lái xe	
	2. Hiện thị danh sách lái xe
3. Thêm mới lái xe	
4. Sửa lái xe	
5. Xóa lái xe	
	6. Kiểm tra thông tin và xác nhận

Bảng 3. 9: Bảng đặc tả use case quản lý lái xe

Tên use case: quản lý lái xe

Tác nhận: quản lý

Chức năng use case: cho phép quản lí thêm, xóa, sửa thông tin lái xe

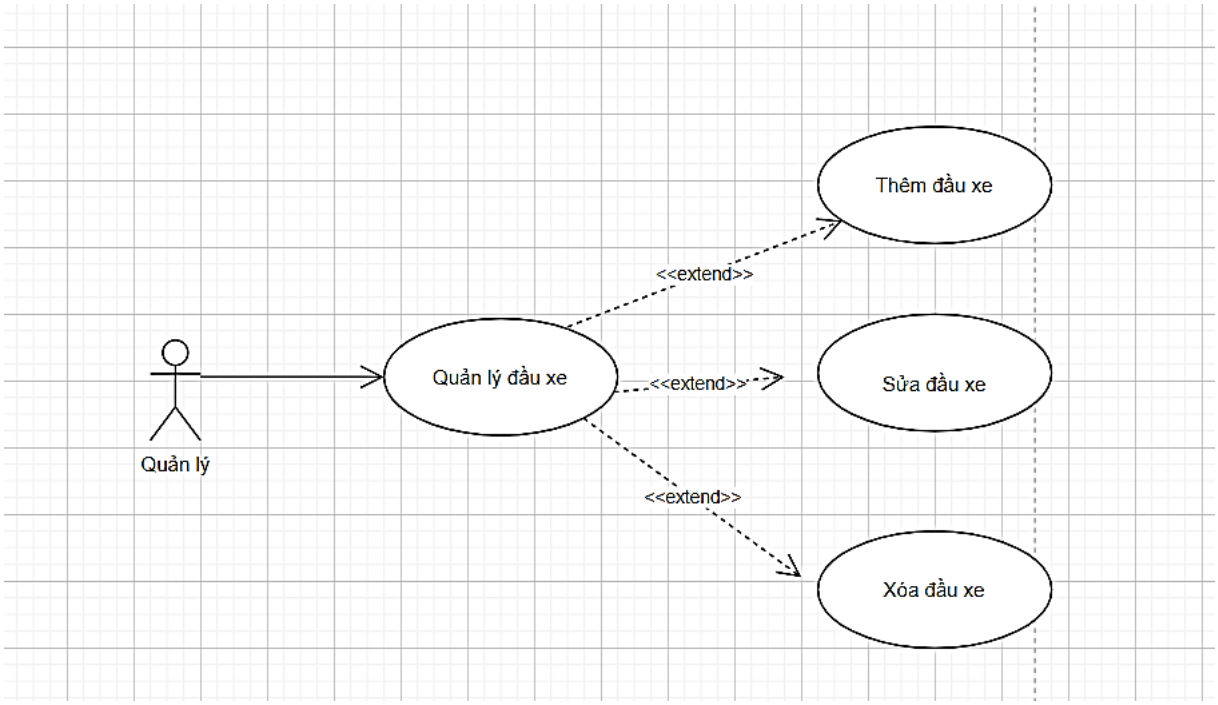
Dòng sự kiện chính:

- Quản lý chọn quản lý lái xe
- Hệ thống hiển thị danh sách các lái xe
- Quản lý thêm mới lái xe
- Quản lý sửa lái xe
- Quản lí xóa lái xe

Dòng sự kiện thay thế:

- Xóa lái xe đồng nghĩa với việc là xóa tất cả thông tin của lái xe

3.4.4. Biểu đồ Use case quản lý đầu xe



Bảng 3. 10: Biểu đồ Use Case quản lý đầu xe

Đặc tả use case

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn quản đầu xe	
	2. Hiện thị danh sách đầu xe
3. Thêm mới đầu xe	
4. Sửa đầu xe	
5. Xóa đầu xe	
	6. Kiểm tra thông tin và xác nhận

Bảng 3. 11: Bảng đặc tả use case quản lý đầu xe

Tên use case: quản lý đầu xe

Tác nhận: quản lý

Chức năng use case: cho phép quản lí thêm, xóa, sửa thông tin đầu xe

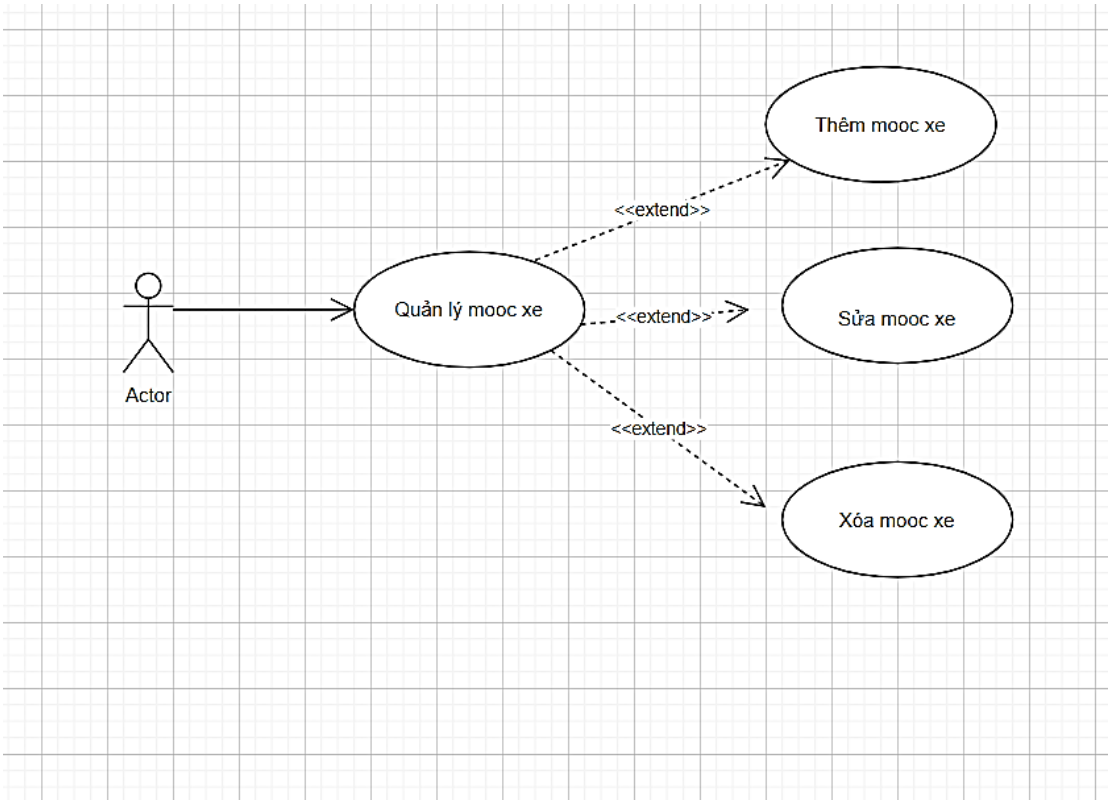
Dòng sự kiện chính:

- Quản lý chọn quản lý đầu xe
- Hệ thống hiển thị danh sách các đầu xe
- Quản lý thêm mới đầu xe
- Quản lý sửa đầu xe
- Quản lí xóa đầu xe

Dòng sự kiện thay thế:

- Xóa đầu xe đồng nghĩa với việc là xóa tất cả thông tin của đầu xe\

3.4.5. Biểu đồ Use case quản lý mooc xe



Hình 3. 4: Biểu đồ Use case quản lý mooc xe

Đặc tả use case

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn quản mooc xe	
	2. Hiện thị danh sách mooc xe
3. Thêm mới mooc xe	
4. Sửa mooc xe	
5. Xóa mooc xe	
	6. Kiểm tra thông tin và xác nhận

Bảng 3. 12: Bảng đặc tả use case Quản lý mooc xe

Tên use case: quản lý mooc xe

Tác nhận: quản lý

Chức năng use case: cho phép quản lí thêm, xóa, sửa thông tin mooc xe

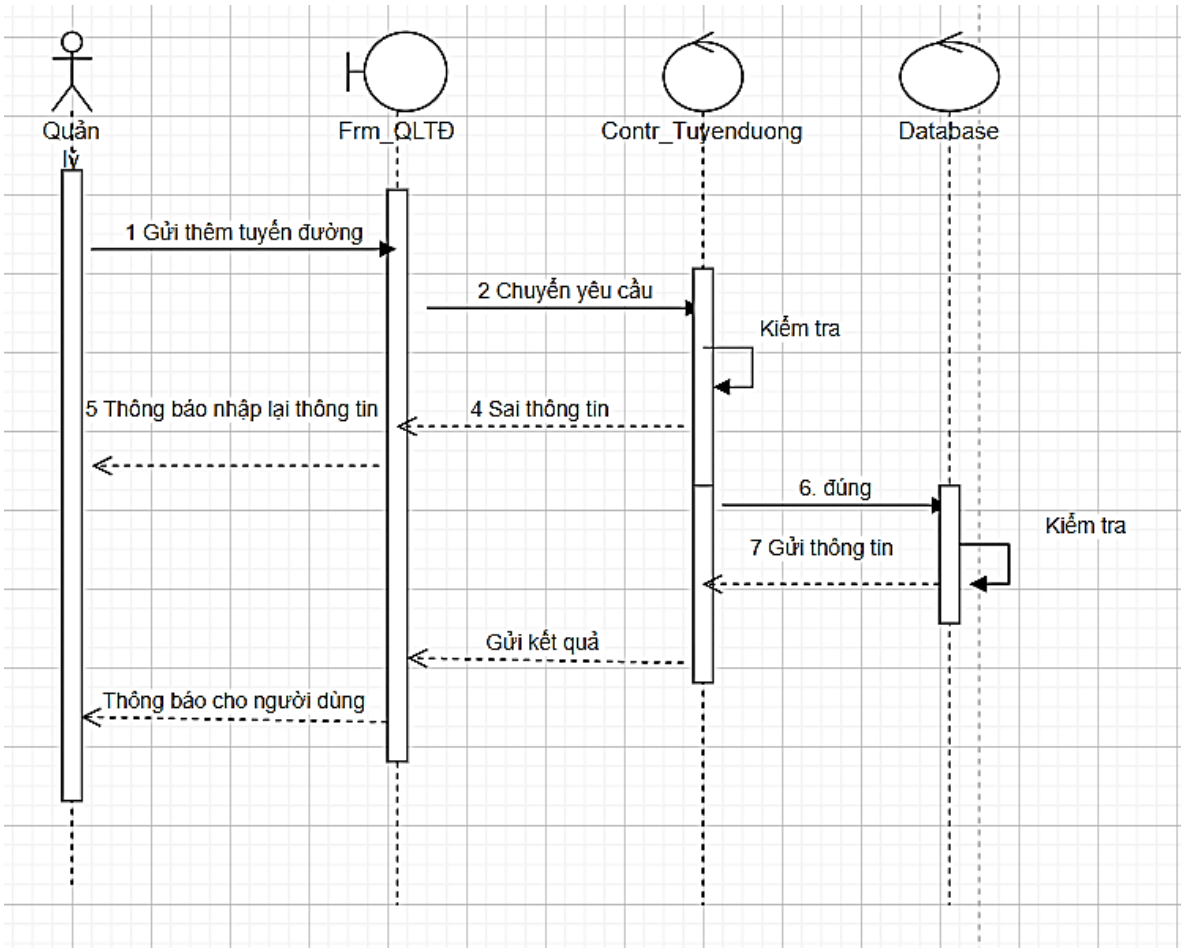
Dòng sự kiện chính:

- Quản lý chọn quản lý mooc xe
- Hệ thống hiển thị danh sách các mooc xe
- Quản lý thêm mới mooc xe
- Quản lý sửa mooc xe
- Quản lí xóa mooc xe

Dòng sự kiện thay thế:

- Xóa mooc xe đồng nghĩa với việc là xóa tất cả thông tin của mooc xe

3.4.6. Biểu đồ use case quản lý tuyến đường



Hình 3. 5: Biểu đồ Use case quản lý tuyến đường

Đặc tả use case

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn quản lý tuyến đường	
	2. Hiện thị danh sách tuyến đường
3. Thêm mới tuyến đường	
4. Sửa tuyến đường	
5. Xóa tuyến đường	
	6. Kiểm tra thông tin và xác nhận

Bảng 3. 13: Bảng đặc tả use case quản lý tuyến đường

Tên use case: quản lý tuyến đường

Tác nhận: quản lý

Chức năng use case: cho phép quản lí thêm, xóa, sửa thông tin tuyến đường

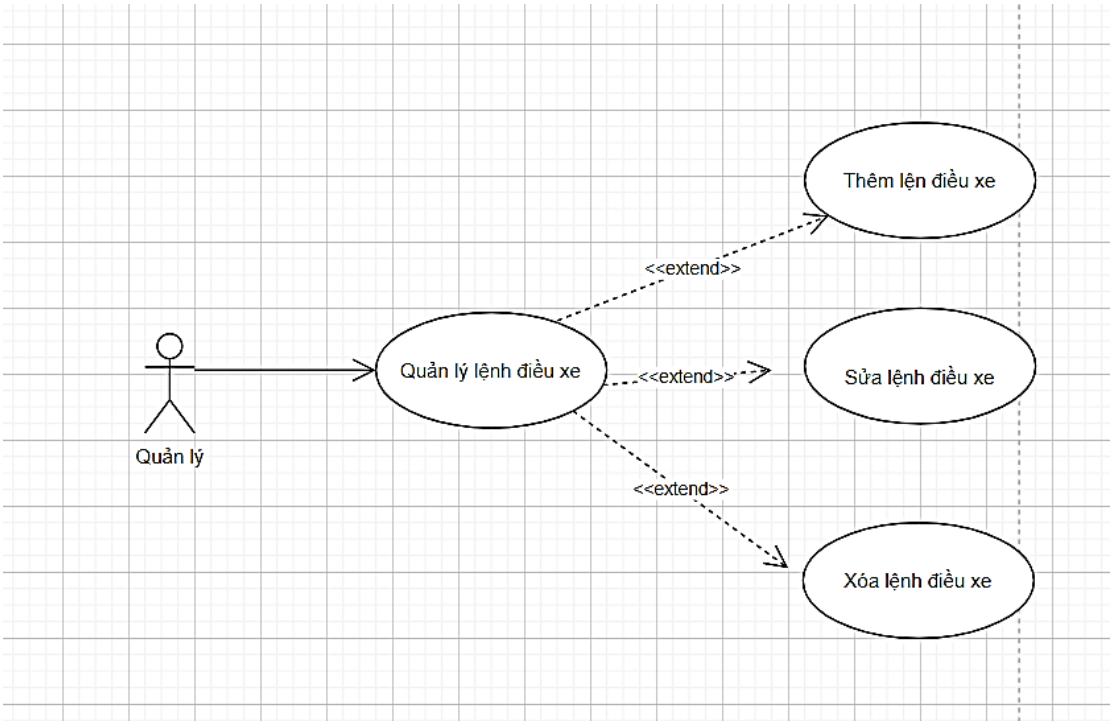
Dòng sự kiện chính:

- Quản lý chọn quản lý tuyến đường
- Hệ thống hiển thị danh sách các tuyến đường
- Quản lý thêm mới tuyến đường
- Quản lý sửa tuyến đường
- Quản lí xóa tuyến đường

Dòng sự kiện thay thế:

- Xóa tuyến đường đồng nghĩa với việc là xóa tất cả thông tin của tuyến đường

3.4.7. Biểu đồ use case quản lý lệnh điều xe



Hình 3. 6: Biểu đồ Use case quản lý tuyến đường

Đặc tả use case

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn quản lý lệnh điều xe	
	2. Hiện thị danh sách lệnh điều xe
3. Thêm mới lệnh điều xe	
4. Sửa tuyến lệnh điều xe	
5. Xóa tuyến lệnh điều xe	
	6. Kiểm tra thông tin và xác nhận

Bảng 3. 14: Bảng đặc tả use case Lệnh điều xe

Tên use case: quản lý lệnh điều xe

Tác nhận: quản lý

Chức năng use case: cho phép quản lí thêm, xóa, sửa thông tin lệnh điều xe

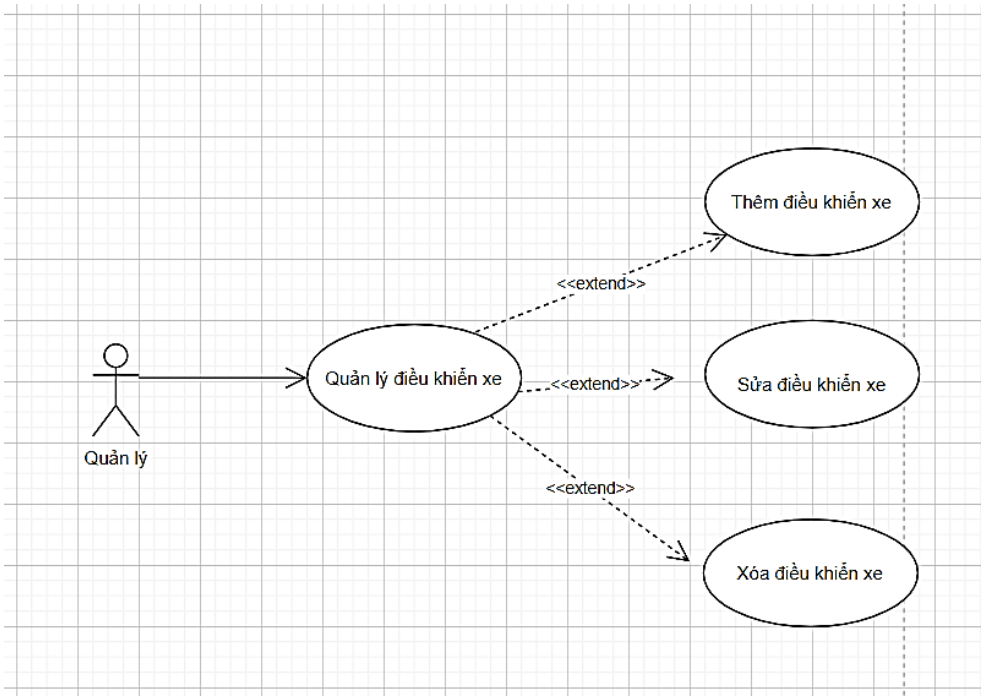
Dòng sự kiện chính:

- Quản lý chọn quản lý lệnh điều xe
- Hệ thống hiển thị danh sách các lệnh điều xe
- Quản lý thêm mới lệnh điều xe
- Quản lý sửa tuyến lệnh điều xe
- Quản lí xóa tuyến lệnh điều xe

Dòng sự kiện thay thế:

- Xóa lệnh điều đồng nghĩa với việc là xóa tất cả thông tin của lệnh điều xe

3.4.8. Biểu đồ use case quản lý điều khiển xe



Hình 3. 7: Biểu đồ Use case quản lý điều khiển xe

Đặc tả use case

Hoạt động của tác nhân	Phản ứng của hệ thống
1. Chọn quản lý điều khiển xe	
	2. Hiện thị danh sách điều khiển xe
3. Thêm mới điều khiển xe	
4. Sửa thông tin điều khiển xe	
5. Xóa thông tin điều khiển xe	
	6. Kiểm tra thông tin và xác nhận

Bảng 3. 15: Bảng đặc tả use case lệnh điều khiển xe

Tên use case: quản lý lệnh điều xe

Tác nhận: quản lý

Chức năng use case: cho phép quản lí thêm, xóa, sửa thông tin điều khiển xe

Dòng sự kiện chính:

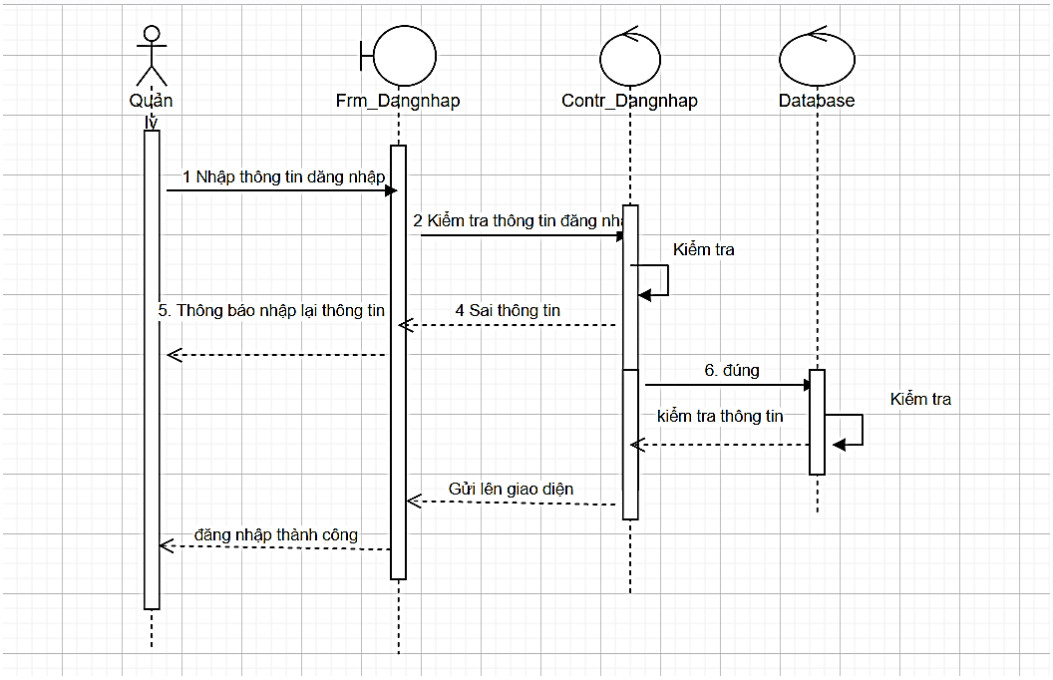
- Quản lý chọn quản lý lệnh điều xe
- Hệ thống hiển thị danh sách các điều khiển xe
- Quản lý thêm mới lệnh điều khiển xe
- Quản lý sửa điều khiển xe
- Quản lí xóa điều khiển xe

Dòng sự kiện thay thế:

- Xóa lệnh điều đồng nghĩa với việc là xóa tất cả thông tin của lệnh điều xe

3.5. Biểu đồ tuần tự

3.5.1. Biểu đồ tuần tự chức năng đăng nhập

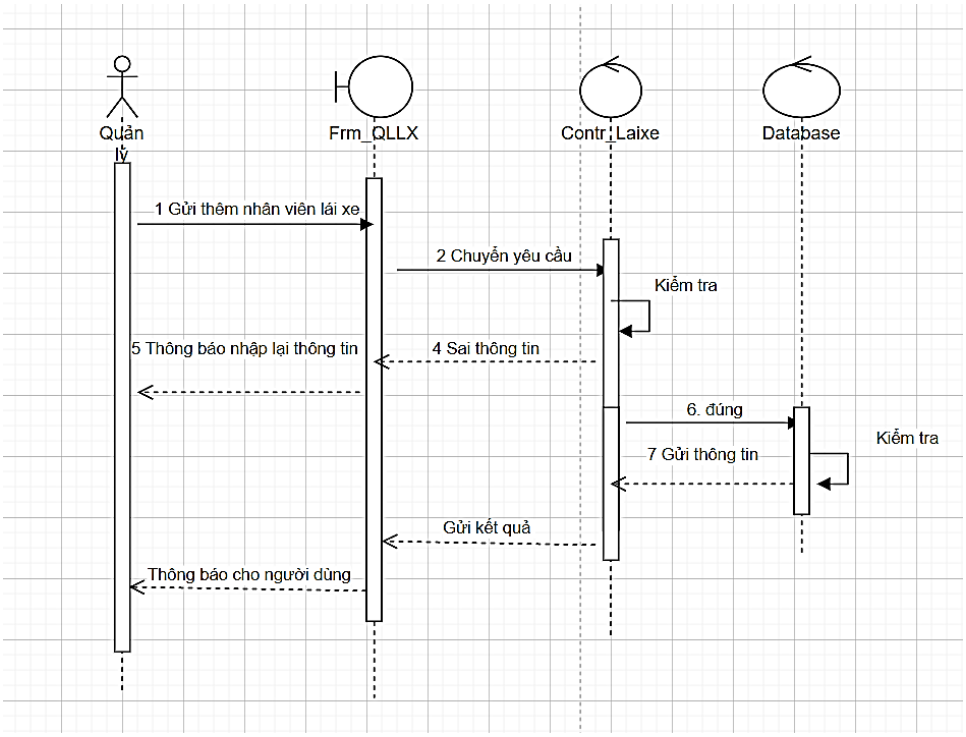


Hình 3. 8: Biểu đồ tuần tự chức năng đăng nhập

Mô tả tóm tắt:

- Khách hàng nhập email và mật khẩu để đăng nhập vào hệ thống.
- Hệ thống sẽ kiểm tra và đối chiếu với thông tin Khách hàng đã đăng ký trong cơ sở dữ liệu.
- Nếu thông tin đúng Hệ thống sẽ gửi thông báo đăng nhập thành công cho Khách hàng.
- Sau khi đăng nhập thành công người dùng có thể thao tác trên giao diện trang chủ của hệ thống với chức năng và quyền hạn của user được cung cấp.
-

3.5.2. Biểu đồ tuần tự cho chức năng thêm lái xe

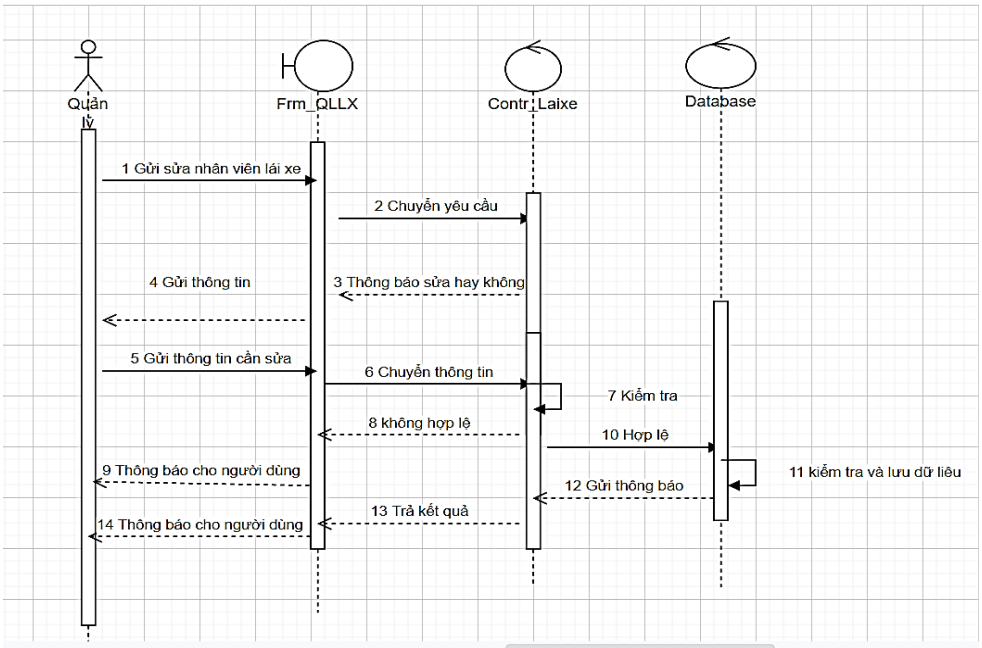


Hình 3. 9: Biểu đồ tuần tự chức năng thêm lái xe:

Mô tả tóm tắt:

- Khách hàng chọn chức năng thêm lái xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo thêm thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại

3.5.3. Biểu đồ tuần tự cho chức năng sửa lái xe

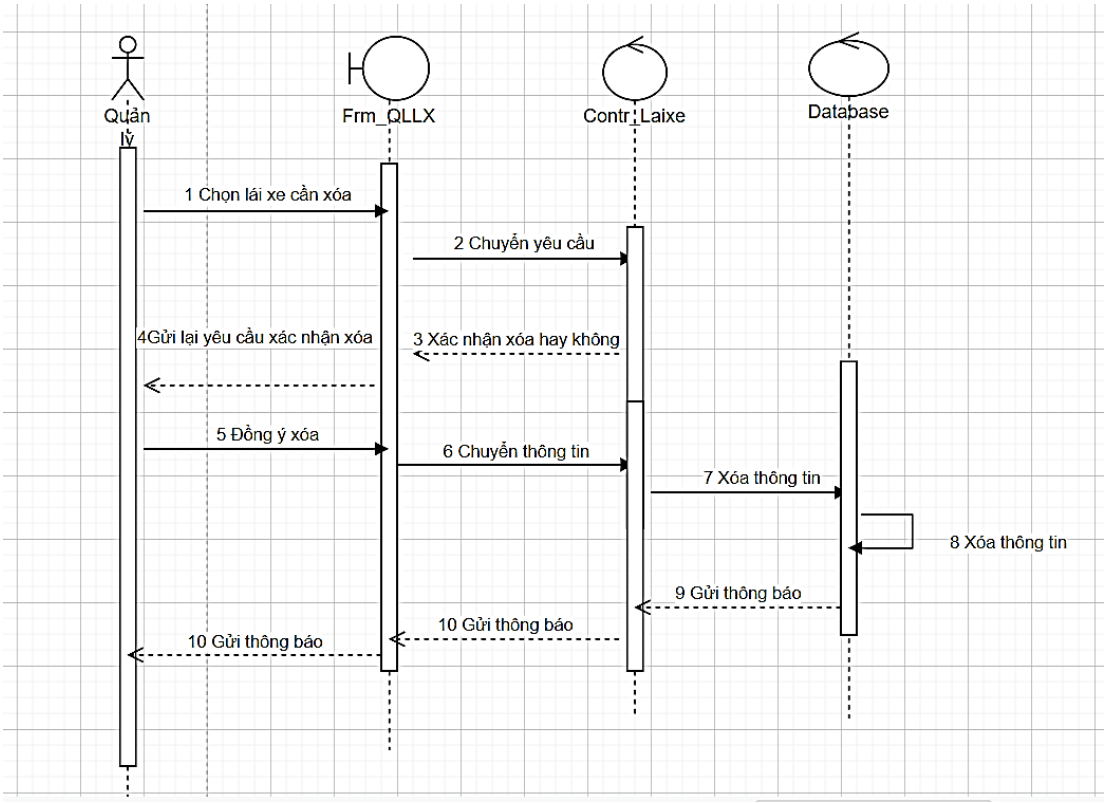


Hình 3. 10: Biểu đồ tuần tự chức năng sửa lái xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng sửa lái xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.4. Biểu đồ tuần tự cho chức năng xóa lái xe

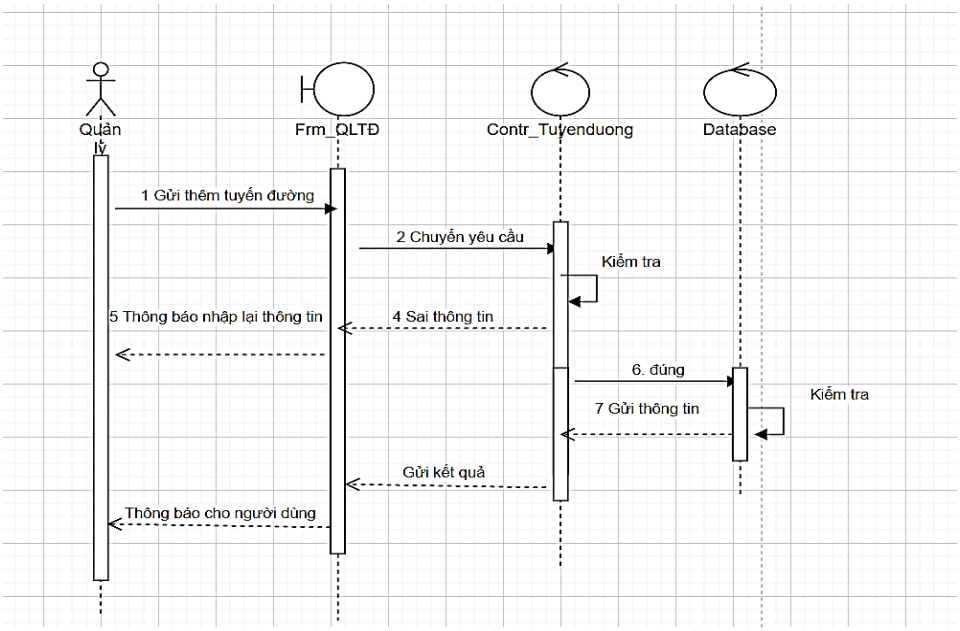


Hình 3. 11: Biểu đồ tuần tự chức năng xóa lái xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng xóa lái xe
- Hệ thống sẽ thông báo cho người dùng là muốn xóa hay không
- Nếu khách hàng đông ý thì hệ thống sẽ kiểm tra và xóa thông tin
- Còn nếu người dùng không đồng ý thì sẽ không thực hiện xóa

3.5.5. Biểu đồ tuần tự cho chức năng thêm tuyến đường

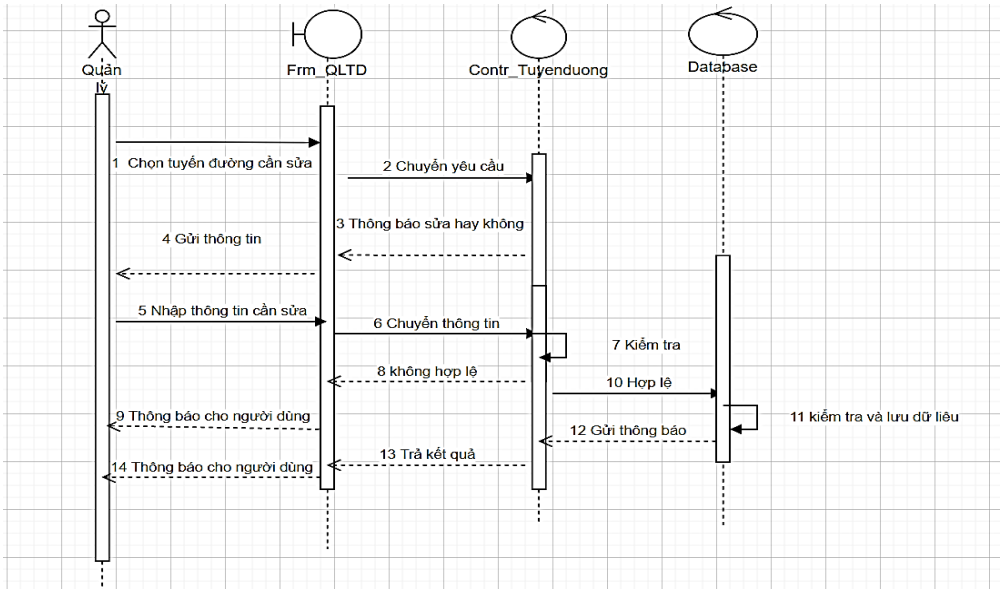


Hình 3. 12: Biểu đồ tuần tự chức năng thêm tuyến đường

Mô tả tóm tắt:

- Khách hàng chọn chức năng thêm tuyến đường
- Nếu thông tin đúng hệ thống sẽ gửi thông báo thêm thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại

3.5.6. Biểu đồ tuần tự cho chức năng sửa tuyến đường

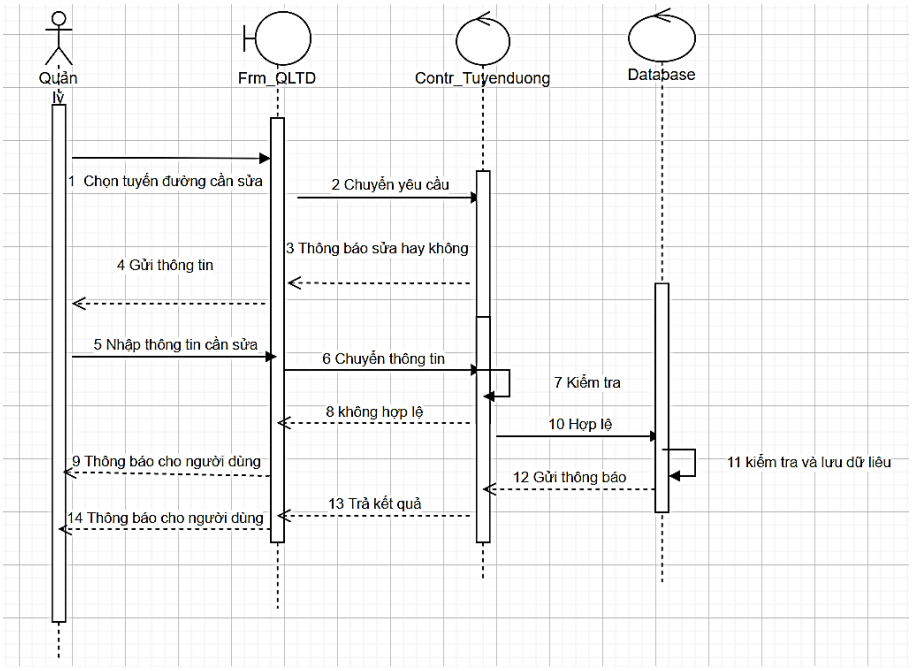


Hình 3. 13: Biểu đồ tuần tự chức năng sửa tuyến đường

Mô tả tóm tắt:

- Khách hàng chọn chức năng sửa tuyến xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.7. Biểu đồ tuần tự cho chức năng xóa tuyến đường

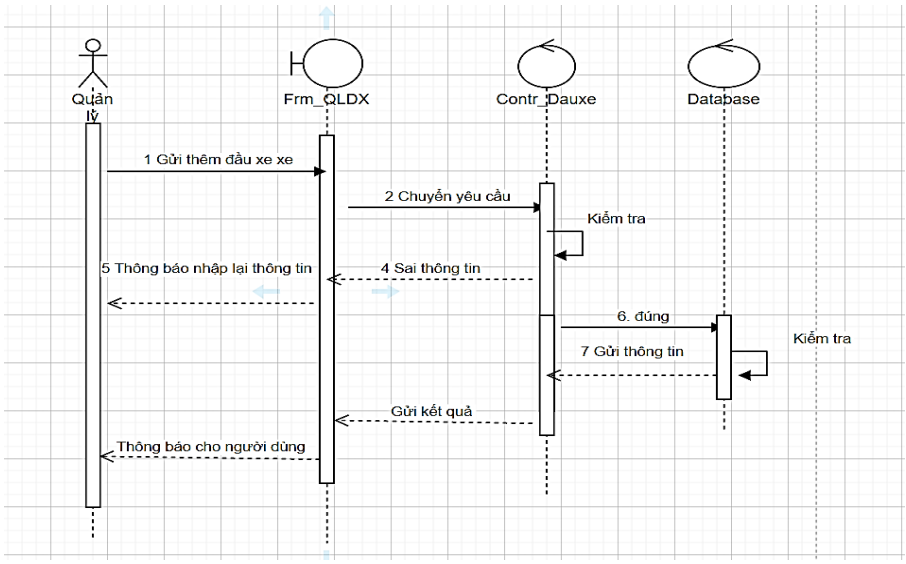


Hình 3. 14: Biểu đồ tuần tự chức năng xóa tuyến đường

Mô tả tóm tắt:

- Khách hàng chọn chức năng xóa tuyến đường
- Hệ thống sẽ thông báo cho người dùng là muốn xóa hay không
- Nếu khách hàng đông ý thì hệ thống sẽ kiểm tra và xóa thông tin
- Còn nếu người dùng không đồng ý thì sẽ không thực hiện xóa

3.5.8. Biểu đồ tuần tự cho chức năng thêm đầu xe

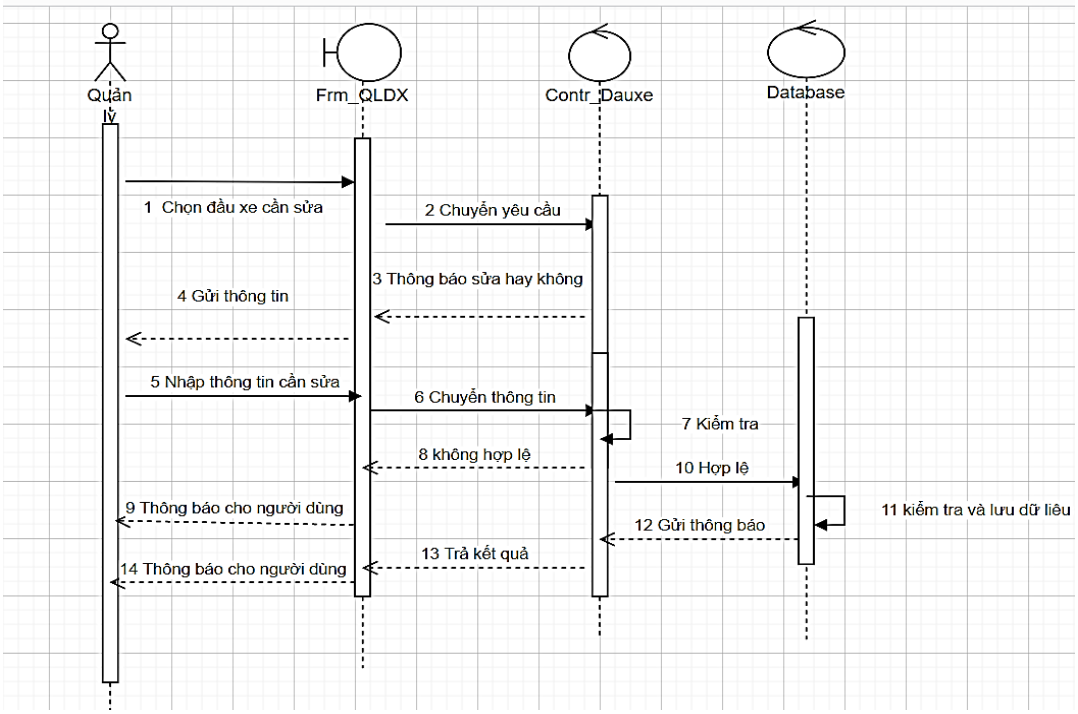


Hình 3. 15: Biểu đồ tuần tự chức năng thêm đầu xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng thêm đầu xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo thêm thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại

3.5.9. Biểu đồ tuần tự cho chức năng sửa đầu xe

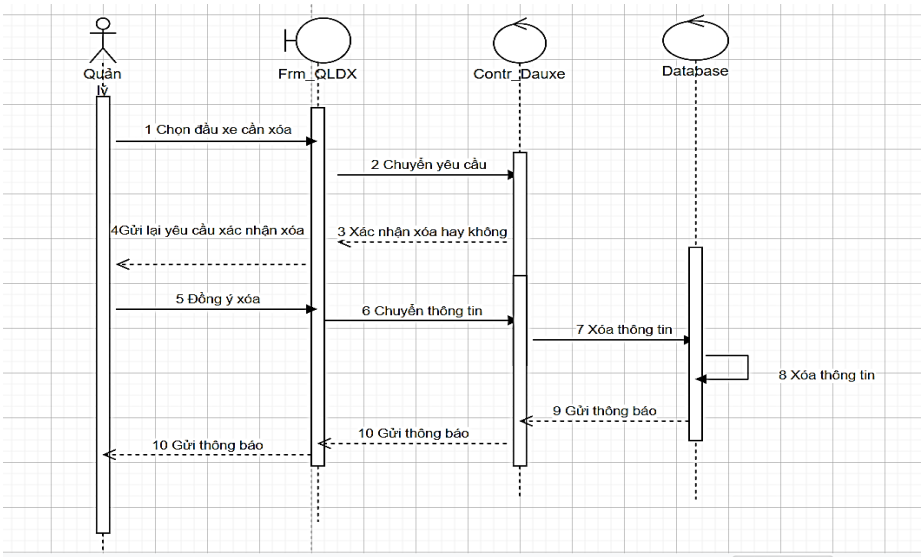


Hình 3. 16: Biểu đồ tuần tự chức năng sửa đầu xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng sửa đầu xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.10. Biểu đồ tuần tự cho chức năng xóa đầu xe

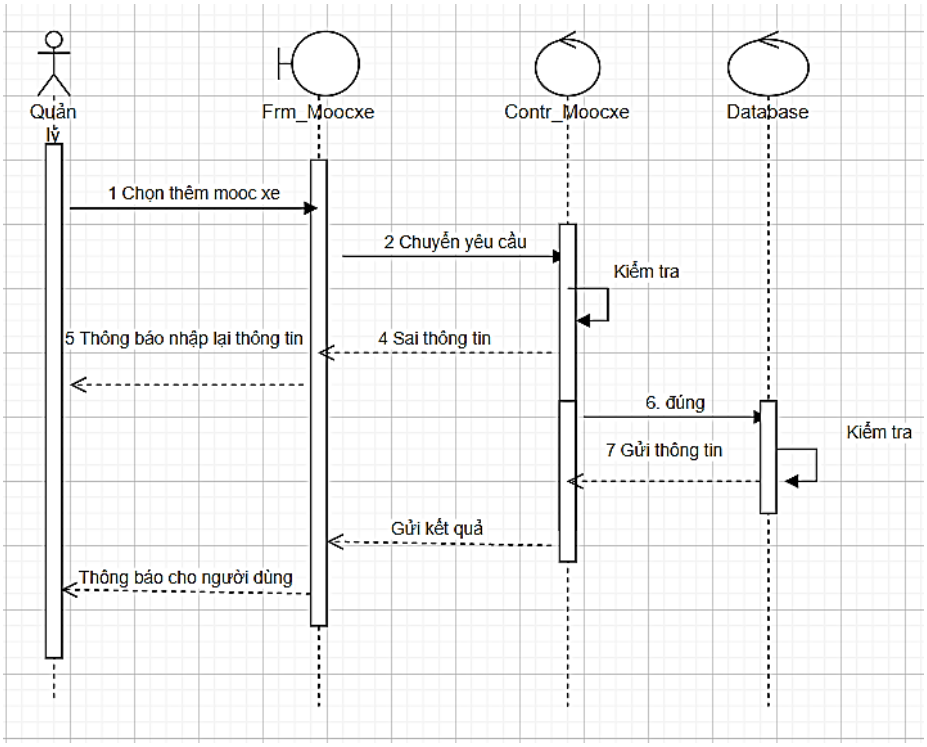


Hình 3. 17: Biểu đồ tuần tự chức năng xóa đầu xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng xóa đầu xe
- Hệ thống sẽ thông báo cho người dùng là muốn xóa hay không
- Nếu khách hàng đông ý thì hệ thống sẽ kiểm tra và xóa thông tin
- Còn nếu người dùng không đồng ý thì sẽ không thực hiện xóa

3.5.11. Biểu đồ tuần tự cho chức năng thêm mooc xe

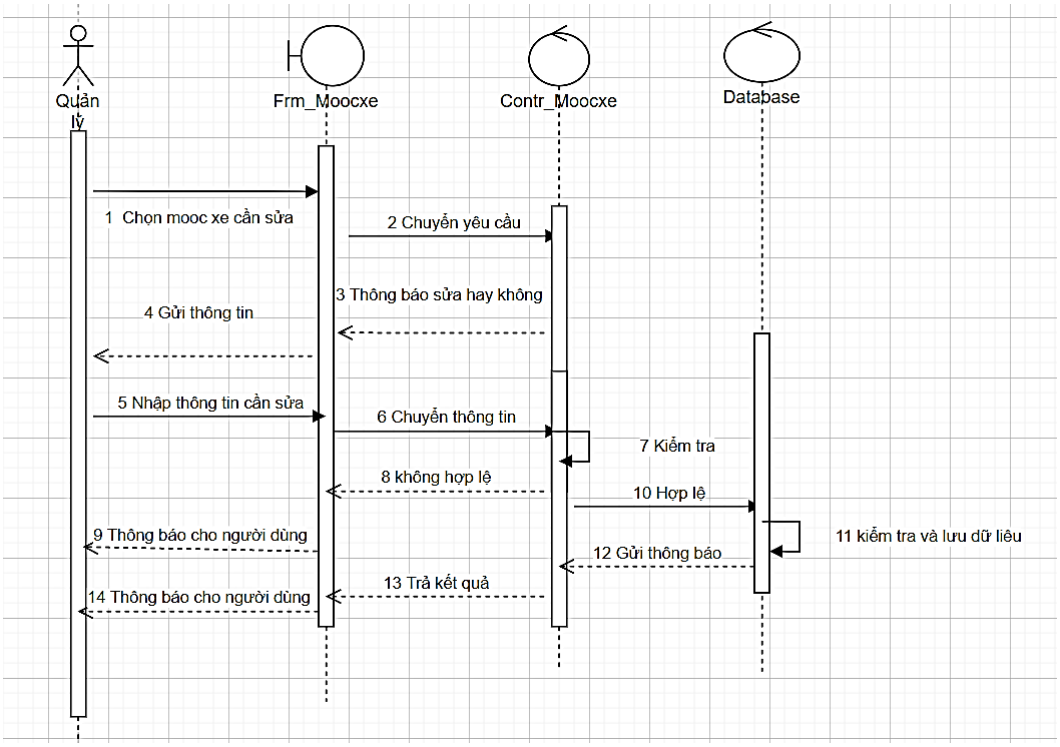


Hình 3. 18: Biểu đồ tuần tự chức năng thêm mooc xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng thêm mooc xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.12. Biểu đồ tuần tự cho chức năng sửa mooc xe

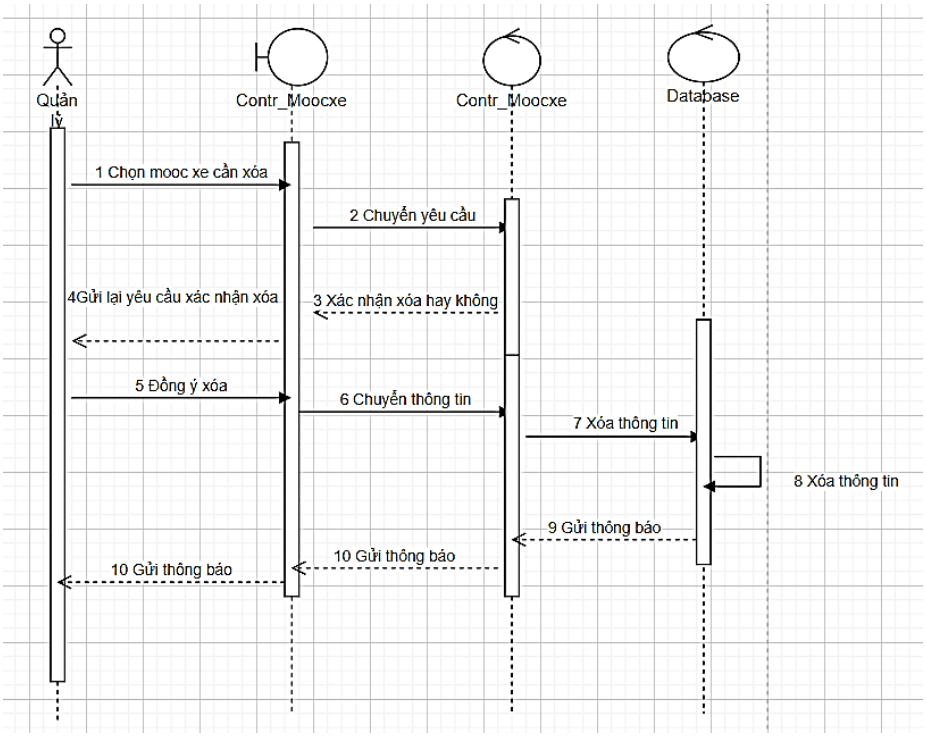


Hình 3. 19: Biểu đồ tuần tự chức năng sửa mooc xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng sửa tuyến xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.13. Biểu đồ tuần tự cho chức năng xóa mooc xe

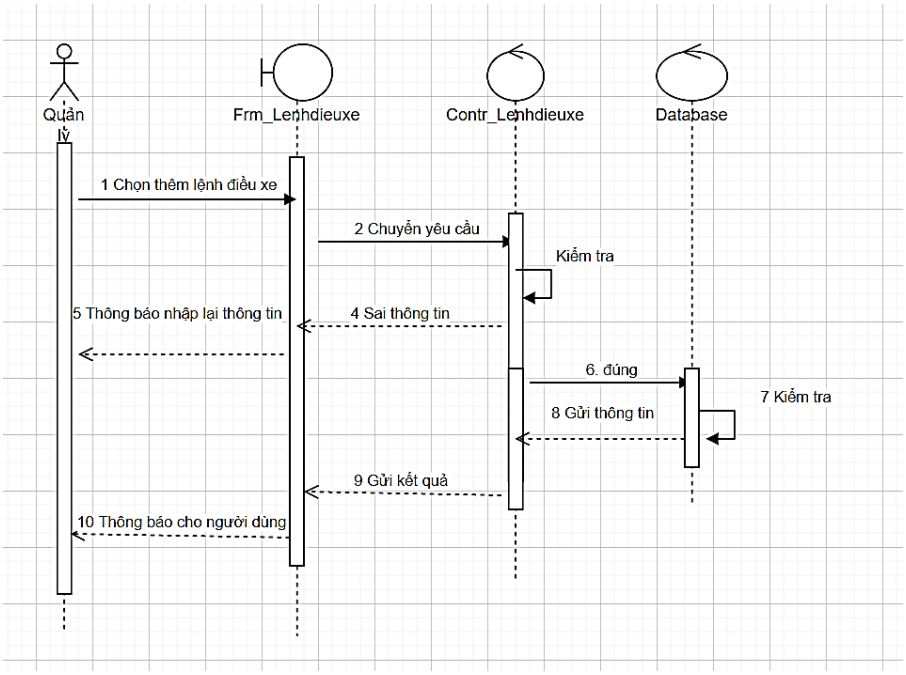


Hình 3. 20: Biểu đồ tuần tự chức năng xóa mooc xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng xóa mooc xe
- Hệ thống sẽ thông báo cho người dùng là muốn xóa hay không
- Nếu khách hàng đông ý thì hệ thống sẽ kiểm tra và xóa thông tin
- Còn nếu người dùng không đồng ý thì sẽ không thực hiện xóa

3.5.14. Biểu đồ tuần tự cho chức năng thêm lệnh điều xe

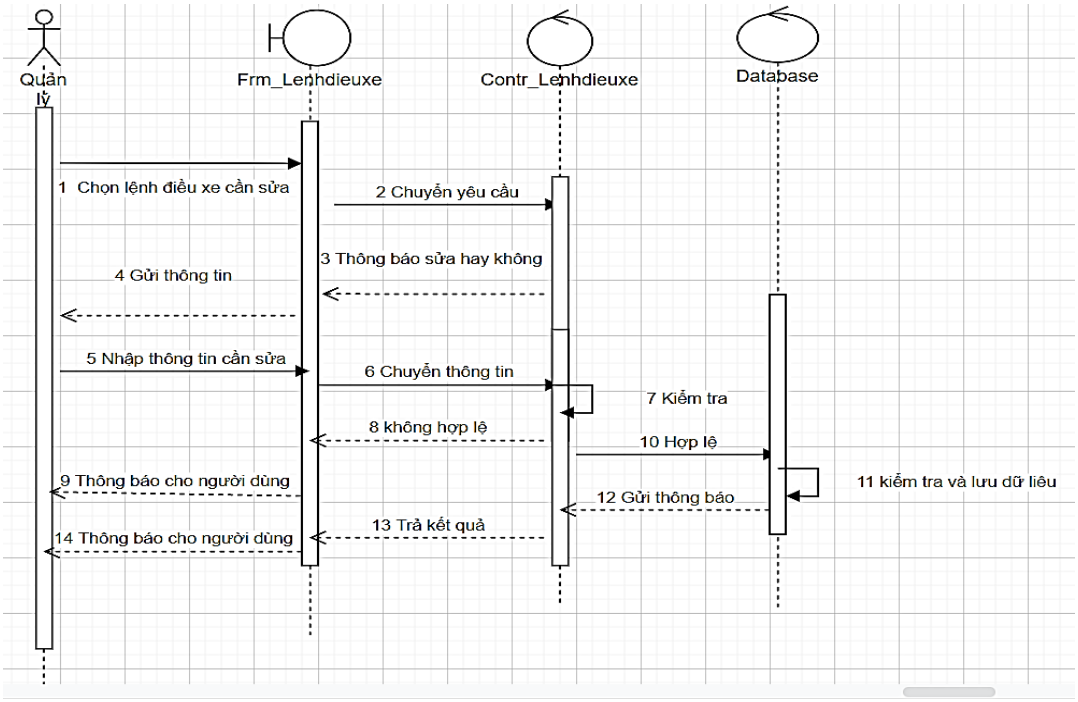


Hình 3. 21: Biểu đồ tuần tự chức năng thêm lệnh điều xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng thêm lệnh điều xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.15. Biểu đồ tuần tự cho chức năng sửa lệnh điều xe

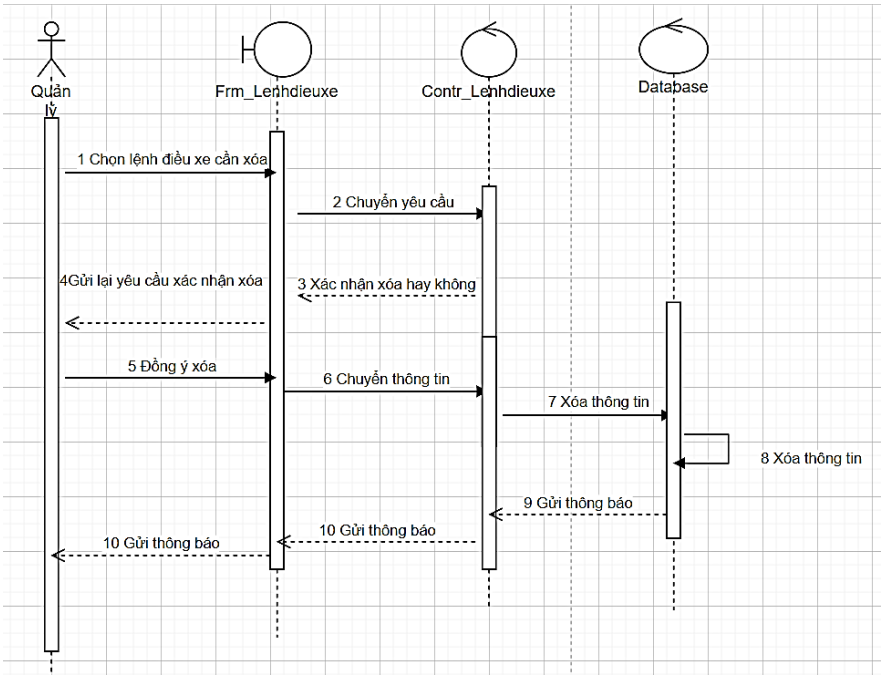


Hình 3. 22: Biểu đồ tuần tự chức năng sửa lệnh điều xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng sửa lệnh xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.16. Biểu đồ tuần tự cho chức năng xóa lệnh điều xe

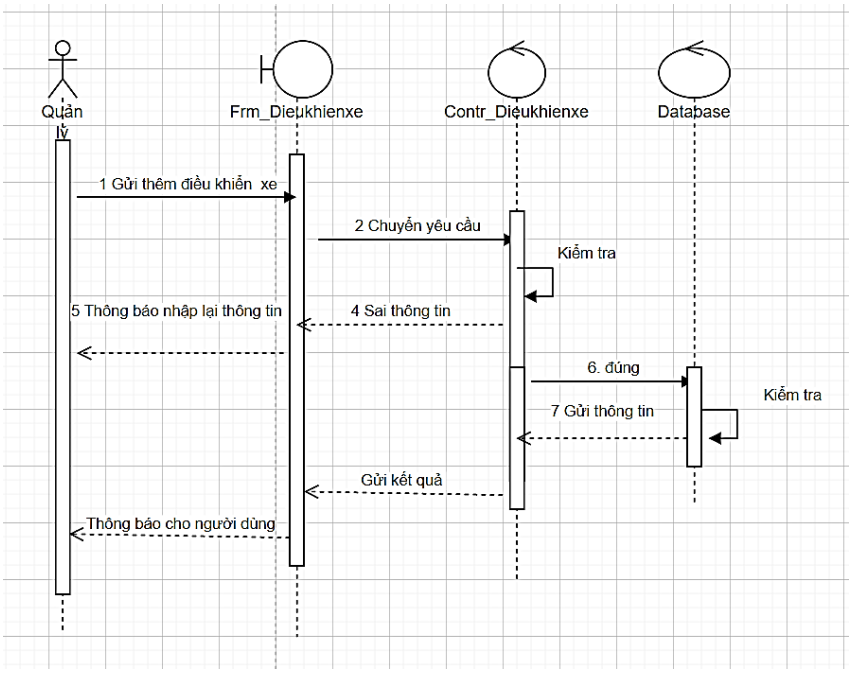


Hình 3. 23: Biểu đồ tuần tự chức năng xóa lệnh điều xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng xóa lệnh xe
- Hệ thống sẽ thông báo cho người dùng là muốn xóa hay không
- Nếu khách hàng đông ý thì hệ thống sẽ kiểm tra và xóa thông tin
- Còn nếu người dùng không đồng ý thì sẽ không thực hiện xóa

3.5.17. Biểu đồ tuần tự cho chức năng thêm điều khiển xe

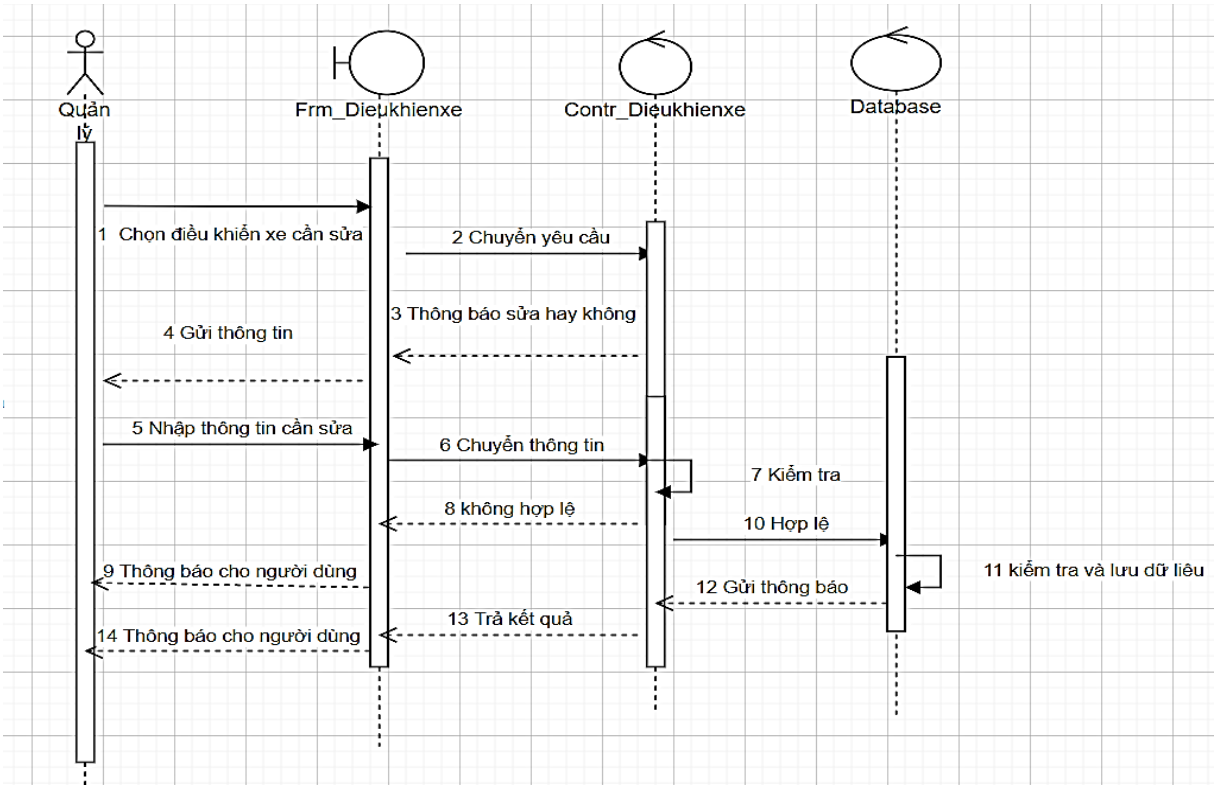


Hình 3. 24: Biểu đồ tuần tự chức năng thêm điều khiển xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng thêm điều khiển xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.18. Biểu đồ tuần tự cho chức năng sửa điều khiển xe

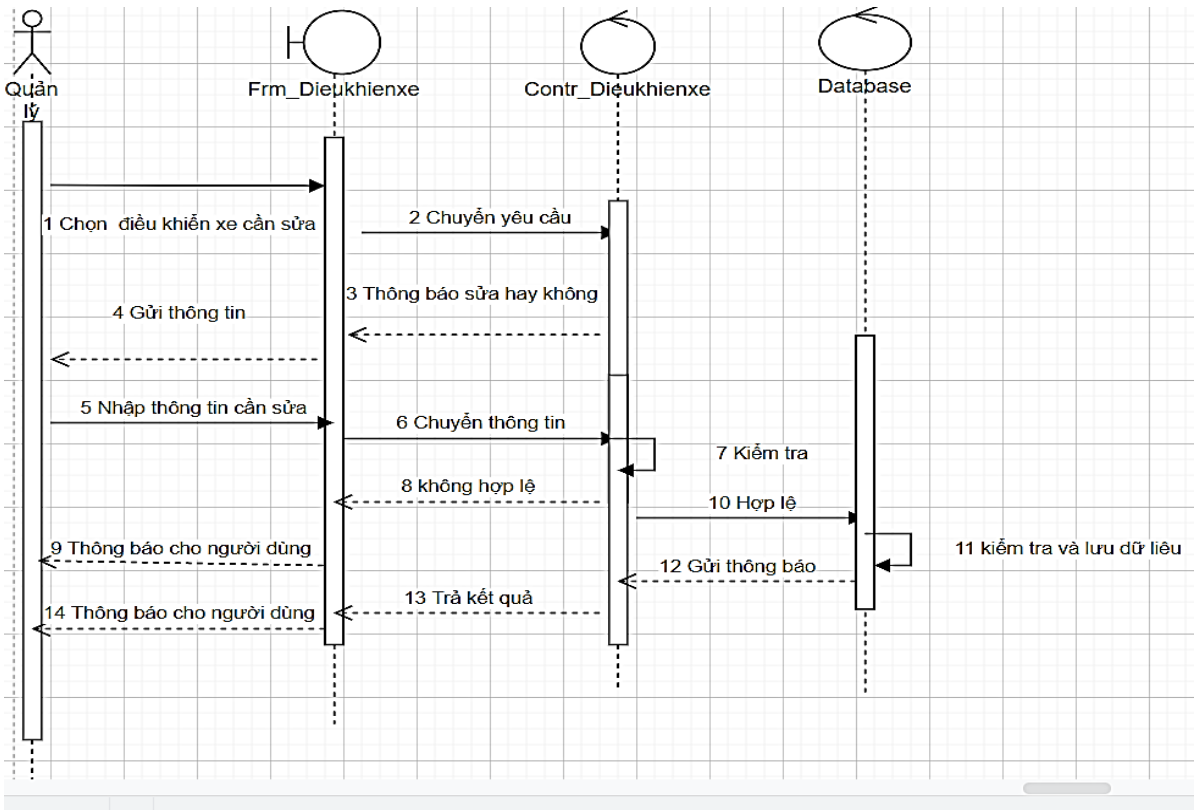


Hình 3. 25: Biểu đồ tuần tự chức năng sửa điều khiển xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng sửa điều khiển xe
- Nếu thông tin đúng hệ thống sẽ gửi thông báo sửa thành công.
- Còn nếu sai hệ thống sẽ yêu cầu nhập lại thông tin

3.5.19. Biểu đồ tuần tự cho chức năng xóa điều khiển xe

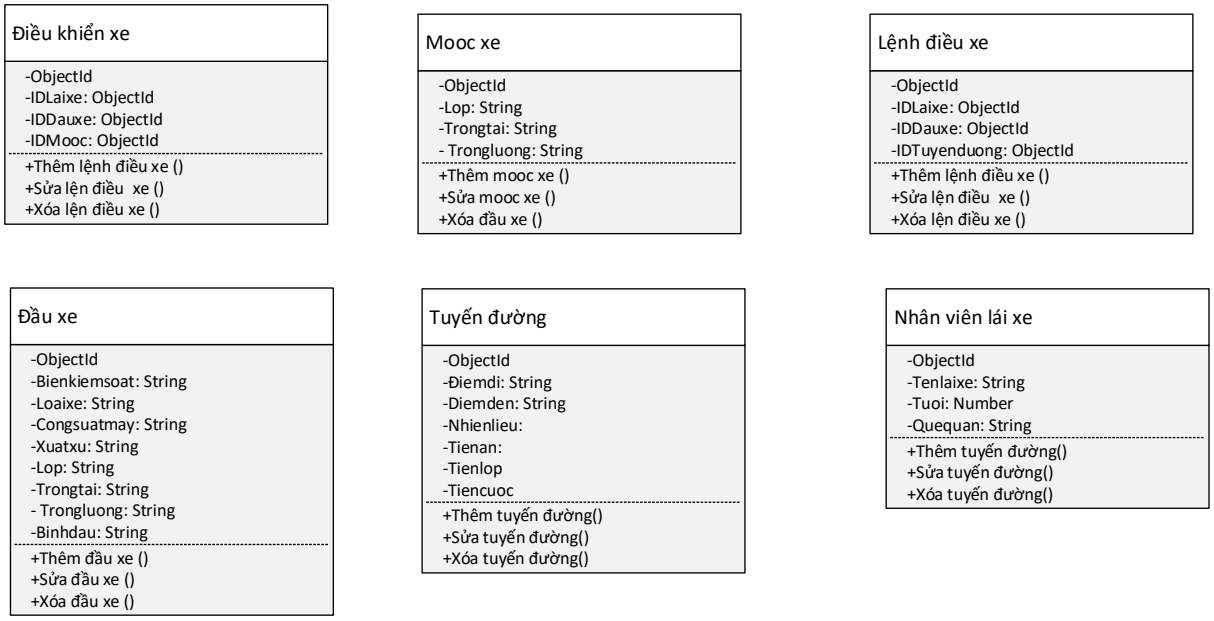


Hình 3. 26: Biểu đồ tuần tự chức năng xóa điều khiển xe

Mô tả tóm tắt:

- Khách hàng chọn chức năng xóa điều khiển xe
- Hệ thống sẽ thông báo cho người dùng là muốn xóa hay không
- Nếu khách hàng đông ý thì hệ thống sẽ kiểm tra và xóa thông tin
- Còn nếu người dùng không đồng ý thì sẽ không thực hiện xóa

3.6. Biểu đồ lớp



Hình 3. 27: Biểu đồ cơ sở dữ liệu

3.7. Xây dựng cơ sở dữ liệu

3.7.1. Bảng "Laixe"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	ObjectId	ObjecId	Mã nhân viên lái xe được sinh tự động
2	Tenlaixe	String	Tên nhân viên lái xe
3	Tuoi	Number	Tuổi của nhân viên lái xe
4	Quequan	String	Quê quản của nhân viên lái xe

Bảng 3. 16: Bảng dữ liệu lái xe

3.7.2. Bảng "Dauxe"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	ObjectId	ObjecId	Mã đầu xe xe được sinh tự động
2	Bienkiemsoat	String	Biển kiểm soát
3	Loaixe	String	Loại xe
4	Congxuatmay	String	Công suất của máy
5	Xuatxu	String	Xuất xử
6	Lop	String	Số lốp
7	Trongtai	String	Trọng tải
8	Trongluong	String	Trọng lượng
9	Binhdau	String	String

Bảng 3. 17: Bảng dữ liệu đầu xe

3.7.2. Bảng “Moocxe”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	ObjectId	ObjecId	Mã mooc xe được sinh tự động
2	Taitrong	String	Tải trong
3	Trongluong	String	Trọng lượng
4	Lop	String	Số lốp

Bảng 1: Bảng dữ liệu mooc xe

3.7.3. Bảng “Tuyenduong”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	ObjectId	ObjecId	Mã mooc xe được sinh tự động
2	Diemdi	String	Điểm đi
3	Diemden	String	Điểm đến
4	Nhienlieu	String	Nhiên liệu
5	Tienan	String	Tiền ăn
6	Tienlop	String	Tiền lốp
7	Tiencuoc	String	Tiền cước

Bảng 3. 18: Bảng dữ liệu tuyến đường

3.7.4. Bảng “Lenhdieuxe”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	ObjectId	ObjecId	Mã của lệnh điều xe được sinh tự động
2	Nguoidieuxe	String	Tên người điều xe
3	IDDauxe	ObjecId	Là ObjectID của bảng Dauxe
3	IDLaixe	ObjecId	Là ObjectID của bảng Laixe
3	IDTuyenduong	ObjecId	Là ObjectID của bảng Tuyenduong

Bảng 3. 19: Bảng dữ liệu lệnh điều xe

3.7.4. Bảng “Dieukhienxe”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	ObjectId	ObjecId	Mã của lệnh điều xe được sinh tự động
2	IDMocxe	ObjecId	Mã của bảng mooc xe
3	IDDauxe	ObjecId	Mã của bảng đầu xe
3	IDLaixe	ObjecId	Là mã của bảng Laixe

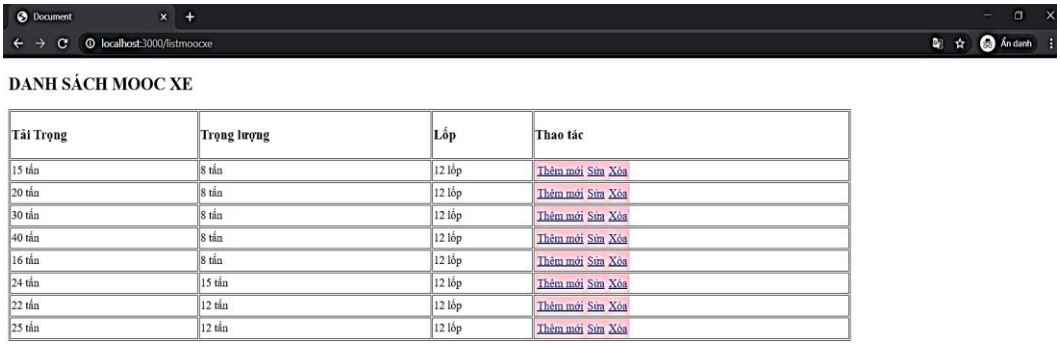
Bảng 3. 20: Bảng dữ liệu điều khiển xe

3.8. Giao diện chương trình

3.8.1. Giao diện danh sách đầu xe

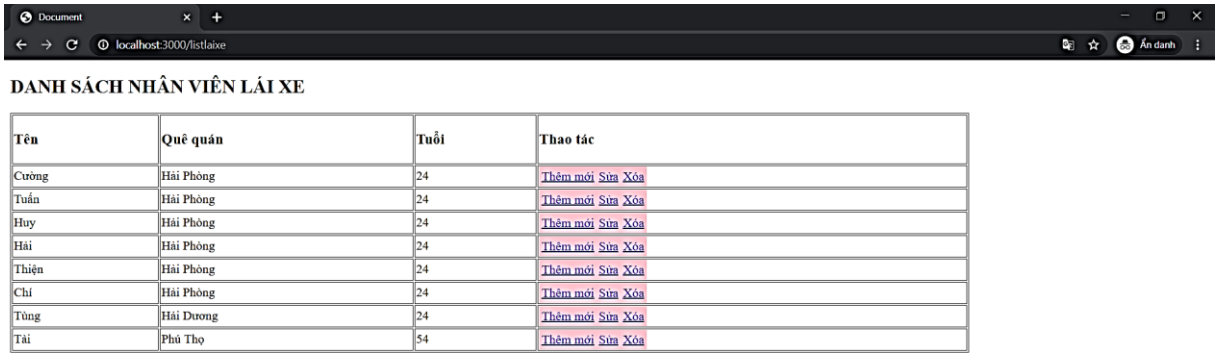
Hình 3. 28: Giao diện danh sách đầu xe

3.8.2. Giao diện danh sách mooc xe



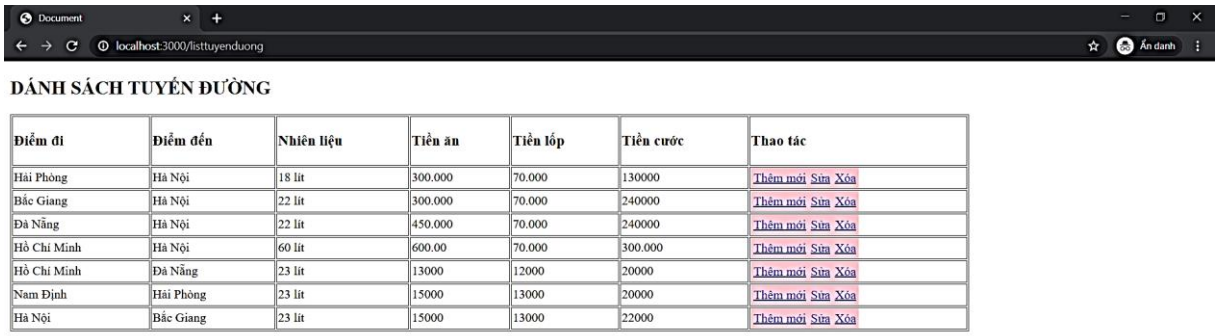
Hình 3. 29: Giao diện danh sách mooc xe

3.8.3. Giao diện danh sách lái xe



Hình 3. 30: Giao diện danh sách lái xe

3.8.4. Giao diện danh sách tuyến đường



Hình 3. 31: Giao diện dánh sách tuyến đường

KẾT LUẬN

Sau 03 tháng thực hiện nghiên cứu đề tài, dưới sự hướng dẫn tận tình của thầy Tiến sỹ Nguyễn Trịnh Đông, đồ án của em đã đạt được những kết quả sau:

1. Kết quả đạt được

- Tìm hiểu và hiểu được các cơ sở dữ liệu MongoDB
- Nắm được nguyên lý hoạt động của WebSocket
- Tìm kiểu, nắm bắt được kĩ thuật sự dụng NodeJS và Socket.IO kết nối với cơ sở dữ liệu MongoDB để tạo ra cách thức xử lí dữ liệu theo thời gian thực
- Xây dựng được website với các chức năng như cập nhật, xóa, sửa với các chức năng như trên phân tích

2. Mặt hạn chế

Trong thời gian qua, em đã cố gắng hết sức để tìm hiểu thực hiện đề tài. Tuy nhiên với kinh nghiệm và thời gian hạn chế nên không thể tránh khỏi những thiếu sót trong đồ án. Cụ thể:

- Phần giao diện vẫn còn chưa được đẹp mắt
- Vì là công nghệ mới, thời gian và trình độ còn hạn chế cho nên hệ thống xây dựng trong đồ án chỉ dừng ở mức thử nghiệm. Do đó nếu muốn áp dụng vào thực tiến thì phải cần có thời gian và công sức để hoàn thiện
- Mới chỉ dừng lại ở những nghiệp vụ cơ bản như: quản lý đầu xe, tuyến đường, mooc xe…

3. Hướng phát triển

- Xây dựng một phần mềm quản lý toàn diện bao gồm việc bảo trì, giám sát xe, giám sát nhiên liệu, quản lý tài xế để đảm bảo sự an toàn cho tài xế và xe, giúp làm giảm đến mức tối thiểu rủi ro về đầu tư cũng như quá trình vận hành xe
- Cung cấp cho nhà quản lý những thông tin cốt yếu trong việc bảo trì xe cũng như can bằng hiệu quả kinh tế: tổng số nhiên liệu đã sử dụng, số km đã đi, mức dầu, mức nước, doanh thu từng xe, từng chuyến, theo các mốc thời gian

TÀI LIỆU THAM KHẢO

[1] Nguyễn Văn Vy. “Phân tích thiết kế các hệ thống thông tin hiện đại theo hướng cấu trúc & hương đối tượng”
[2] https://nodejs.org/en/
[3] https://morioh.com/p/c466eab81513
[4] https://expressjs.com/
[5] https://mongoosejs.com/