# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

**1. Thông tin chung**

|  |  |
| --- | --- |
| Tên đề tài | Xây dựng hệ thống quản lý nhà trọ |
| Họ và tên sinh viên: | Phạm Trọng Nhân |
| Điện thoại liên lạc: | 0775.331.777 |
| Email: | [ptnhan109@gmail.com](mailto:ptnhan109@gmail.com) |
| Lớp: | Công Nghệ Phần Mềm K60 |
| Hệ đào tạo: | Đại học chính quy |
| Đồ án tốt nghiệp được thực hiện tại: | Hà Nội |
| Thời gian làm ĐATN: | 2023 |

**2. Mục tiêu của ĐATN**

Nghiên cứu tổng quan về mô hình kỹ thuật công nghệ

1. ASP.NET API
2. Angular 13

Ứng dụng để xây dựng hệ thống quản lý nhà trọ

**3. Các nhiệm vụ cụ thể của ĐATN**

- Nghiên cứu các kỹ thuật xây dựng ASP.NET API, Angular 12

- Ứng dụng xây dựng Hệ thống quản lý nhà trọ

**4. Lời cam đoan của sinh viên:**

Tôi – Phạm Trọng Nhân – cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của ThS. Hoàng Anh Đức

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Tác giả ĐATN  Phạm Trọng Nhân |

**5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:**

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Cán bộ hướng dẫn  Ký, ghi rõ họ tên |

# TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Phần tóm tắt nội dung đồ án tốt nghiệp được biên soạn sau khi bản đồ án được hoàn chỉnh, tóm tắt đồ án phản ánh trung thực nội dung chủ yếu của đồ án. Yêu cầu của tóm tắt là ngắn gọn, cô đọng, nêu được cấu trúc của đề tài, nêu bật được những nội dung chính của đồ án, nhấn mạnh được những nội dung cần thiết, những kết luận rút ra sau khi nghiên cứu đề tài cùng với các giải pháp, đề xuất, kiến nghị (nếu có). Phần tóm tắt nội dung đồ án được viết thành 01 đoạn văn, dài không quá 300 từ.

# LỜI CẢM ƠN

Trong thời gian làm đồ án tốt nghiệp, em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy hướng dẫn và các thầy cô trong khoa - bộ môn Công nghệ phần mềm.

Em xin gửi lời cảm ơn chân thành đến thầy ThS.Hoàng Anh Đức, giảng viên Bộ môn Công nghệ phần mềm - trường ĐH Mỏ - Địa Chất Hà Nội người đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình làm khoá luận.

Em cũng xin chân thành cảm ơn các thầy cô giáo trong Bộ môn Công nghệ phần mềm Trường Đại học Mỏ - Địa Chất đã dạy dỗ cho em kiến thức các môn chuyên ngành, giúp em có được cơ sở lý thuyết vững vàng và có nhiều góp ý qua những lần bảo vệ tiến độ để đồ án của em được tốt hơn.

Em xin chân thành cảm ơn.

# MỤC LỤC

[PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP I](#_Toc150179013)

[TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP III](#_Toc150179014)

[LỜI CẢM ƠN IV](#_Toc150179015)

[MỤC LỤC V](#_Toc150179016)

[DANH MỤC CÁC HÌNH VẼ VIII](#_Toc150179017)

[DANH MỤC CÁC BẢNG BIỂU IX](#_Toc150179018)

[Danh mục các từ viết tắt X](#_Toc150179019)

[MỞ ĐẦU 1](#_Toc150179020)

[CHƯƠNG 1 TỔNG QUAN 2](#_Toc150179021)

[1.1 Khảo sát tình hình thực tế 2](#_Toc150179022)

[1.2 Phân tích ưu, nhược điểm các hệ thống đã có 2](#_Toc150179023)

[1.3 Mục tiêu xây dựng 4](#_Toc150179024)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 5](#_Toc150179025)

[2.1 Tổng quan ASP NET CORE 5](#_Toc150179026)

[2.1.1 Tổng quan ASP NET 5](#_Toc150179027)

[2.1.2 Tổng quan ASP NET Core 6](#_Toc150179028)

[2.1.3 Tổng quan về Web API 7](#_Toc150179029)

[2.1.4 Kiểu dữ liệu JSON 8](#_Toc150179030)

[2.1.5 Dữ liệu XML 9](#_Toc150179031)

[2.2 Tổng quan về Angular Framework 10](#_Toc150179032)

[2.2.1 Định nghĩa Angular Framework 10](#_Toc150179033)

[2.2.2 Lịch sử phát triển 11](#_Toc150179034)

[2.2.3 Các thành phần cơ bản 12](#_Toc150179035)

[2.2.4 Ưu điểm của Angular 13](#_Toc150179036)

[2.3 Kiến trúc Onion Architecture 15](#_Toc150179037)

[2.3.1 Ý tưởng 15](#_Toc150179038)

[2.3.2 Kiến trúc chung 16](#_Toc150179039)

[2.3.3 Ưu điểm 17](#_Toc150179040)

[2.4 Thư viện Bootstrap 18](#_Toc150179041)

[2.4.1 Giới thiệu thư viện Bootstrap 18](#_Toc150179042)

[2.4.2 Các tính năng chính của Bootstrap 18](#_Toc150179043)

[2.4.3 Ưu điểm của Bootstrap 19](#_Toc150179044)

[CHƯƠNG 3 PHÂN TÍCH VÀ THIẾT HỆ THỐNG 21](#_Toc150179045)

[3.1 Biểu đồ Use Case tổng quát 21](#_Toc150179046)

[3.2 Use case chi tiết 22](#_Toc150179047)

[3.2.1 Use case quản lý khu trọ 22](#_Toc150179048)

[3.2.2 Use case quản lý phòng trọ 23](#_Toc150179049)

[3.2.3 Use case quản lý dịch vụ 23](#_Toc150179050)

[3.2.4 Use case quản lý thiết bị 24](#_Toc150179051)

[3.2.5 Use case quản lý hóa đơn 25](#_Toc150179052)

[3.2.6 Use case quản lý khách hàng 26](#_Toc150179053)

[3.2.7 Use case báo cáo 27](#_Toc150179054)

[3.2.8 Use case quản lý hợp đồng 27](#_Toc150179055)

[3.3 Mô tả chi tiết các use case 28](#_Toc150179056)

[3.3.1 Đăng nhập 28](#_Toc150179057)

[3.3.2 Đăng ký 28](#_Toc150179058)

[3.3.3 Quản lý khu trọ 29](#_Toc150179059)

[3.3.4 Quản lý phòng trọ 30](#_Toc150179060)

[3.3.5 Quản lý thiết bị 31](#_Toc150179061)

[3.3.6 Quản lý dịch vụ 32](#_Toc150179062)

[3.3.7 Quản lý khách hàng 33](#_Toc150179063)

[3.3.8 Quản lý phương tiện 34](#_Toc150179064)

[3.3.9 Quản lý hợp đồng 35](#_Toc150179065)

[3.3.10 Quản lý hóa đơn 37](#_Toc150179066)

[3.3.11 Báo cáo 38](#_Toc150179067)

[CHƯƠNG 4 XÂY DỰNG CƠ SỞ DỮ LIỆU 38](#_Toc150179068)

[4.1 Xây dựng quan hệ giữa các bảng 38](#_Toc150179069)

[4.2 Danh sách các bảng 40](#_Toc150179070)

[4.2.1 dbo.Cities 40](#_Toc150179071)

[4.2.2 Dbo.AppContracts 40](#_Toc150179072)

[4.2.3 dbo.BoardingHouses 41](#_Toc150179073)

[4.2.4 dbo.ContractTerms 42](#_Toc150179074)

[4.2.5 dbo.Customers 43](#_Toc150179075)

[4.2.6 dbo.FitmentInRooms 44](#_Toc150179076)

[4.2.7 dbo.FitmentInRooms 45](#_Toc150179077)

[4.2.8 dbo.Invoices 45](#_Toc150179078)

[4.2.9 dbo.Provides 46](#_Toc150179079)

[4.2.10 dbo.Rooms 46](#_Toc150179080)

[4.2.11 dbo.ServiceInBoardingHouses 47](#_Toc150179081)

[4.2.12 dbo.StagePayments 48](#_Toc150179082)

[4.2.13 dbo.StageRooms 49](#_Toc150179083)

[4.2.14 dbo.SystemFiles 50](#_Toc150179084)

[4.2.15 dbo.Systems 50](#_Toc150179085)

[4.2.16 dbo.UserInfos 51](#_Toc150179086)

[4.2.17 dbo.Users 52](#_Toc150179087)

[4.2.18 dbo.Vehicles 53](#_Toc150179088)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 54](#_Toc150179089)

[PHỤ LỤC 56](#_Toc150179090)

# DANH MỤC CÁC HÌNH VẼ

[Hình 2‑1 Công nghệ ASP NET của Microsoft 5](#_Toc150179091)

[Hình 2‑2 Application Programming Interface 7](#_Toc150179092)

[Hình 2‑3 Angular framework 11](#_Toc150179093)

[Hình 2‑4 Mô hình Onion Architechture 16](#_Toc150179094)

[Hình 2‑5 Thư viện giao diện Bootstrap 18](#_Toc150179095)

[Hình 3‑1 Biểu đồ Use case tổng quát hệ thống 21](#_Toc150179096)

[Hình 3‑2 Biểu đổ Use case quản lý khu trọ 22](#_Toc150179097)

[Hình 3‑3 Biểu đồ use case quản lý phòng trọ 23](file:///D:\projects\moma\docs\Lastest.docx#_Toc150179098)

[Hình 3‑4 Biểu đồ use case quản lý dịch vụ 24](file:///D:\projects\moma\docs\Lastest.docx#_Toc150179099)

[Hình 3‑5 Biểu đồ use case quản lý thiết bị 24](#_Toc150179100)

[Hình 3‑6 Quản lý hóa đơn thuê phòng 25](#_Toc150179101)

[Hình 3‑7 Biểu đồ use case quản lý khách hàng 26](#_Toc150179102)

[Hình 3‑8 Biểu đồ use case báo cáo 27](#_Toc150179103)

[Hình 3‑9 Biểu đồ use case quản lý hợp đồng 27](#_Toc150179104)

# DANH MỤC CÁC BẢNG BIỂU

**No table of figures entries found.**

# Danh mục các từ viết tắt

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Từ viết tắt | Từ tiếng Anh | Từ và nghĩa tiếng Việt |
| 1 | CSDL |  | Cơ sở dữ liệu |
| 2 | DB | Database | Cơ sở dữ liệu |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# MỞ ĐẦU

1. Tổng quan tình hình nghiên cứu thuộc lĩnh vực của đề tài

Hiện tại, hình thức kinh doanh cho thuê trọ, cho thuê kí túc xá không còn quá mới lạ tại các khu đô thị lớn, hay gần các khu công nghiệp lớn nơi tập trung dân cư đông đúc. Bất chấp chi phí đầu tư cho hình thức này là cực lớn nhưng bù lại chi phí duy trì thấp, và nguồn thu ổn định nên việc kinh doanh cho thuê trọ vẫn nở rộ trong những năm gần đây. Hình thức kinh doanh lớn là thế nhưng việc vận hành của việc kinh doanh nhà trọ vẫn còn rất lỏng lẻo về các vấn đề như giá thành, hợp đồng, doanh thu thập chí an toàn PCCC,… kéo theo vô số bất cập cả người cho thuê và người thuê.

**2. Tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài**

Từ những phân phân tích trên, em xuất phát ý tưởng cho một hệ thống tập trung vào việc quản lý nhà trọ theo một cách chuyên nghiệp, với những ưu điểm lớn như hạn chế tối đa những thất thoát về tiền bạc, thời gian cũng như công sức của chủ cho thuê. Đồng thời cũng làm minh bạch, rõ ràng các vấn đề về hợp đồng, giá thuê, chi phí dịch vụ giữa chủ thuê và người thuê trọ.

Trong báo cáo đồ án này sẽ tập trung trình bày những nghiên cứu cơ bản về lập trình Web, API và triển khai các bước phân tích và thiết kế để xây dựng được một hệ thống hoàn chỉnh, thân thiện, an toàn với người sử dụng. Tuy không phải là người mới nhưng kiến thức còn hạn chế nên sẽ không tránh được những sai sót trong quá trình xây dựng, em kính mong thầy cô, cũng như các bán đóng góp ý kiến để đề tài được hoàn thiện hơn.

# TỔNG QUAN

## Khảo sát tình hình thực tế

Trước khi thực hiện xây dựng hệ thống em đã khảo sát một số khu trọ, chủ trọ trên địa bàn phường Cổ Nhuế

1. Chủ trọ Vũ Quang Hải – 0364109554 - Số nhà 401/33, Cổ Nhuế 2

Số lượng phòng: 14 phòng trọ được chia thành 2 khu trọ

Hình thức quản lý: Quản lý thủ công thông qua sổ sách

Các khó khăn gặp phải

* Hay nhầm lẫn về tiền thuê, chi phí dịch vụ
* Phải đi lại khu trọ để ghi số điện, số nước và thu tiền người thuê trọ
* Khó khăn trong quản lý khách thuê, và tài sản do không ở gần khu trọ

1. Chủ trọ Nguyễn Thế Khánh – 0364109554 - Số nhà 521/33/36, Cổ Nhuế 2

Số lượng phòng: 8 phòng trọ xây cạnh nhà chính

Hình thức quản lý: Quản lý thông qua hệ thống KhuTro.vn

Các khó khăn gặp phải

* Hệ thống khó sử dụng, nhầm lẫn trong thao tác
* Chi phí duy trì cao

## Phân tích ưu, nhược điểm các hệ thống đã có

**Mona House**

Mona House được xem là phần mềm quản lý nhà trọ được các chủ nhà trọ tin dùng nhiều nhất hiện nay. Đây là phần mềm cực kỳ dễ sử dụng với bất kỳ thiết bị nào. Công ty có 11 năm kinh nghiệm trong việc phát triển phần mềm, Mona House có hơn 800 dự án phần mềm quản lý bất động sản, căn hộ cho thuê, nhà trọ, chung cư mini,...đáp ứng mọi nhu cầu kinh doanh ở khắp mọi nơi.

Ưu điểm

Mona House là một hệ thống lớn, và cực kì chuyên nghiệp. Hệ thống đã xây dựng trên nhiều nền tảng khác nhau như Web, AppMobile. Không những quản lý nhà trọ ứng dụng còn tập trung sang các mảng liên quan như Bất động sản, cho thuê ngắn ngày, …

Tích hợp sẵn nhiều tính năng nâng cao, và phân quyền người dùng rõ ràng trong hệ thống với các vai trò như Chủ trọ, kế toán, quản lý nhà, tổng quản lý ..

Nhược điểm

Chính vì việc ứng dụng lớn và tích hợp các mảng khác nên phát sinh việc ứng dụng tương đối khó sử dụng với người không có nhiều kinh nghiệm về máy tính, đây là một nhược điểm lớn, vì đa phần các chủ trọ cho thuê đều là những người lớn tuổi

Chi phí duy trì còn tương đối cao gói thấp nhất với ứng dụng có phí 290.000 đ / tháng, chi phí này không thực sự phù hợp với các chủ trọ có số lượng phòng cho thuê ít, doanh thu không nhiều

**Quản lý nhà trọ Lam Nguyen**

Ứng dụng quản lý nhà trọ - Lam Nguyen là ứng dụng tính phí khá quen thuộc với những người từng dùng ứng dụng này. Ứng dụng này cũng có các tính năng tương tự các ứng dụng quản lý phần mềm khác như quản lý thông tin khách thuê, số liệu điện nước tiêu thụ, báo cáo chi tiết các hoạt động kinh doanh, có thể xuất hóa đơn qua máy in Bluetooth.

Ưu điểm

Chi phí tương đối thấp phù hợp với các khu trọ nhỏ, là một ứng dụng trên điện thoại nên mang lại tính cơ động cao phù hợp với đối tượng mà hệ thống đang hướng tới là các nhà trọ có quy mô từ nhỏ đến vừa ở các khu công nghiệp

Nhược điểm

Là một ứng dụng trên điện thoại nên đôi khi các tính năng không được hiển thị một cách tối ưu, dẫn đến việc khó khăn trong quá trình sử dụng, ví dụ các bảng biểu khó để hiển thị trên thiết bị di động một cách đầy đủ

Hệ thống còn hạn chế tính năng về việc báo cáo, thu chi

**PosApp**

Kinh doanh khách sạn, resort không chỉ có dịch vụ cho thuê phòng mà còn có nhiều các dịch vụ kèm theo nhằm tăng trải nghiệm dịch vụ của khách hàng như quán cafe, nhà hàng, spa, massage… Công việc quản lý các quán phục vụ tiện ích mở rộng này không phải đơn giản. Bạn có thể tham khảo qua phần mềm PosApp chuyên quản lý các dịch vụ này trong khách sạn.

Ưu điểm

Tương tự như Mona House đây là một ứng dụng lớn, với hệ thống chức năng nhiều không thuần túy là quản lý nhà trọ mà cả nhiều lĩnh vực khác như bán hàng, cho thuê nhà nghỉ, khách sạn của ngành F&B

Nhược điểm

Ứng dụng không tâp trung chuyên sâu vào quản lý riêng nhà trọ

Ứng dụng lớn, giao diện không quá thân thiện

Chi phí còn tương đối cao

## Mục tiêu xây dựng

Từ những nghiên cứu về tình hình thực tế, và ưu nhược điểm các hệ thống sẵn có trên thị trường em định hướng xây dựng hệ thống quản lý nhà trọ với mong muốn đảm bảo được các yêu cầu cần thiết về việc quản lý nhà trọ như sau

Với chủ trọ

* Quản lý các khu trọ, phòng trọ và các dịch vụ hiện có
* Quản lý khách thuê trọ, phương tiện của khách thuê
* Quản lý hợp đồng thuê trọ, hợp đồng đặt cọc phòng trọ
* Quản lý tiền thuê, các đợt thanh toán tiền trọ
* Báo cáo tổng quan, báo cáo chi tiết hệ thống
* Quản lý các vấn đề phát sinh của khu trọ như hỏng thiết bị, mất cắp
* Gửi thông báo đến người thuê
* …

Với người thuê

* Nhận thông báo và thanh toán trực tuyến tiền trọ
* Tra cứu các thông tin thanh toán
* Tìm kiếm nhà trọ
* Báo hỏng các thiết bị hoặc các vấn đề phát sinh trong quá trình thuê
* …

Đồng thời hệ thống cũng cần đảm bảo các yêu cầu khác như

* Dễ sử dụng, với các những người sử dụng có kĩ năng về công nghệ thông tin chưa tốt như người lớn tuổi, người ít tiếp xúc với công nghệ
* Tốc độ hệ thống tốt
* An toàn về bảo mật thông tin

# CƠ SỞ LÝ THUYẾT

## Tổng quan ASP NET CORE

### Tổng quan ASP NET

ASP.NET là một nền tảng web được sử dụng để tạo các trang web, ứng dụng và dịch vụ web. Đây là sản phẩm tích hợp của các ngôn ngữ HTML, CSS và JavaScript. ASP.NET được ra đời vào năm 2002 và đến nay đã được nâng cấp thành nhiều phiên bản khác nhau.



Hình 2‑1 Công nghệ ASP NET của Microsoft

ASP.NET hoạt động trên HTTP, sử dụng các lệnh và chính sách của HTTP để trình duyệt có thể trở thành giao tiếp song phương của máy chủ.

ASP.NET cung cấp những công nghệ để tạo ứng dụng Web

1. **Webform**

ASP.NET cung cấp tương tác hướng sự kiện cho các ứng dụng web, nó được sử dụng để phát triển ứng dụng có quyền truy cập dữ liệu. Đồng thời, kiểu phát triển web này cũng sẽ cung cấp dữ liệu cho máy chủ và sự kiện để tạo nên ứng dụng phù hợp. Web Forms được phát triển một cách nhanh chóng nhờ vào sử dụng thư viện điều khiển phong phú đánh dấu HTML

1. **ASP.NET MVC**

Cung cấp các mẫu MVC (Model View Controller) để xây dựng lên trang web động. Những mẫu này được sử dụng cho dữ liệu (Model), giao diện người dùng (View) và logic ứng dụng (Controller). Mô hình này sẽ đảm nhận nhiệm vụ duy trì dữ liệu và logic của ứng dụng. Chế độ xem sẽ là giao diện người dùng và hiển thị dữ liệu. Bộ điều khiển sẽ xử lý yêu cầu của người dùng về phần View và Model này

1. **ASP.NET web Pages**

Được sừ dụng để tạo ra các trang web động kết hợp máy chủ với HTML một cách nhanh chóng. ASP.NET web Pages sẽ đánh dấu mã và HTML cùng nhau trong cùng một tệp.

Các thành phần chính của nền tảng ASP.NET cơ bản gồm:

1. Ngôn ngữ: ASP.NET sử dụng nhiều ngôn ngữ lập trình khác nhau như VB.NET và C#.
2. Thư viện: ASP.NET có bộ thư viện chuẩn bao gồm các giao diện, các lớp và kiểu giá trị. Bộ thư viện này có thể sử dụng lại cho quá trình phát triển ASP.NET và xây dựng các chức năng của hệ thống.
3. Thời gian chạy ngôn ngữ chung (CLR): CLR - Common Language Runtime được sử dụng để thực hiện các hoạt động mã. Các hoạt động này sẽ thực hiện xử lý các ngoại lệ và thu gom rác.

### Tổng quan ASP NET Core

ASP.NET Core là một web framework mã nguồn và được tối ưu hóa cho cloud để phát triển các ứng dụng web chạy trên nhiều nền tảng như Windows, Linux và Mac. Hiện tại, nó bao gồm MVC framework được kết hợp các tính năng của MVC và Web API thành một web framework duy nhất.

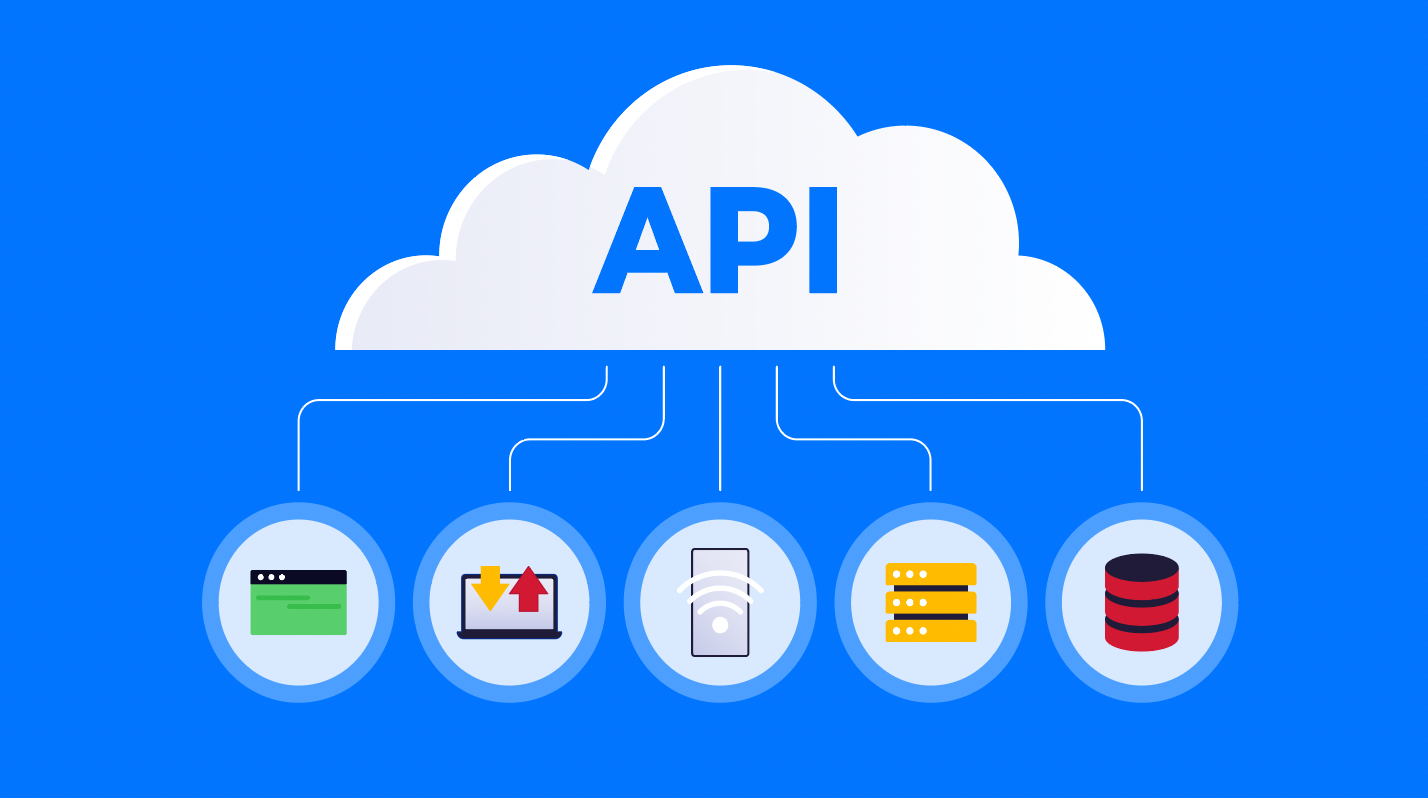
Các ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên .NET Framework hoàn chỉnh. Nó đã được thiết kế để cung cấp một framework tối ưu cho các ứng dụng để triển khai tới cloud hoặc chạy on-premises. Nó bao gồm những modular với các thành phần tối thiểu, do đó các lập trình viên có thể giữ được tính linh hoạt trong quá trình xây dựng các giải pháp của mình. Có thể phát triển và chạy các ứng dụng đa nền tảng từ ASP.NET Core trên Windows, Mac và Linux.

Ưu điểm của ASP NET Core

1. ASP.NET Core có một số thay đổi kiến trúc dẫn đến modular framework nhỏ hơn.
2. ASP.NET Core không còn dựa trên System.Web.dll. Nó dựa trên một tập hợp nhiều yếu tố của Nuget packages.
3. Điều này cho phép tối ưu ứng dụng của mình chỉ cần những NuGet packages cần thiết.
4. Lợi ích của diện tích bề mặt ứng dụng nhỏ hơn thì bảo mật chặt chẽ hơn, giảm dịch vụ, cải thiện hiệu suất và giảm chi phí.
5. Xây dựng và chạy các ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux.
6. Công cụ mới giúp đơn giản hóa việc phát triển web hiện đại.
7. Liên kết đơn các web stack như Web UI và API Web.
8. Cấu hình dựa trên môi trường đám mây sẵn có.
9. Được xây dựng dựa trên cho DI (Dependency Injection).
10. Tag Helpers làm cho các Razor makup trở nên tự nhiên hơn với HTML.
11. Có khả năng host trên IIS hoặc self-host.

### Tổng quan về Web API

API là viết tắt của Application Programming Interface (Giao diện lập trình ứng dụng), một công nghệ trung gian cho phép hai ứng dụng nói chuyện với nhau. API được sử dụng rất phổ biến trong các ứng dụng hiện đại

******

Hình 2‑2 Application Programming Interface

Mục đích chính của một API là cung cấp khả năng truy xuất đến một tập các hàm hay dùng, ví dụ, hàm để vẽ các cửa sổ hay các icon trên màn hình. Các API, cũng như hầu hết các interfaces, đều có tính trừu tượng (abstract). Phần mềm muốn cung cấp truy xuất đến chính nó thông qua các API cho sẵn, phải hiện thực API đó. Trong nhiều trường hợp, một API thường là một phần của bộ SDK (Software Development Kit). Một bộ SDK có thể bao gồm một API cũng như các công cụ/ phần cứng, vì thế hai thuật ngữ này không thay thế cho nhau được.

*Tầm quan trọng của WebAPI*

API là khớp nối giữa các thành phần phần mềm.

Giả sử hệ thống có một tính năng cần cung cấp cho module khác phần mềm khác thì sẽ mở một API để tác giả của module/ phần mềm truy cập vào.

Ví dụ, trên các thiết bị điện toán thì hệ điều hành là phần mềm duy nhất có khả năng truy cập tới các thiết bị phần cứng. Do đó, hệ điều hành sẽ phải cung cấp API để ghi file, đọc file, đọc dữ liệu…. Mỗi ứng dụng khi hoạt động sẽ gọi tới API tương ứng của hệ điều hành.

Khi một phần mềm gọi tới API, phần mềm gọi có thể cung cấp dữ liệu đầu vào và đòi hỏi dữ liệu đầu ra từ API hoặc không, nhưng trong mọi trường hợp, phần mềm gọi để có thể tiếp tục hoạt động thì nó cần phần mềm cung cấp phải thực hiện những gì đã cam kết qua API.

### Kiểu dữ liệu JSON

*JSON*

JSON là viết tắt của cụm từ JavaScript Object Notation. Có thể hiểu JSON giống như một kiểu định dạng dữ liệu theo một quy định. Mà quy định này tất cả hầu hết các ngôn ngữ lập trình đều có thể đọc và hiểu được. Cũng chính vì điều đó mà Json được coi như một tiêu chuẩn mở rộng để có thể trao đổi giữ những thông tin, dữ liệu trên web.

Để tạo ra một file Json, người dùng cần phải tuân thủ đúng cú pháp. Đây chính là điều mà tất cả những người dùng đều phải tìm hiểu. Cú pháp chính là điều cơ bản thiết yếu để các lập trình viên sử dụng Json. Có hai yếu tố cốt lõi trong mỗi Object chính là Keys và Values.

**Key**: Chắc chắn đã không còn xa lạ đối với người dùng. Nó phải là chuỗi các ký tự được bao quanh bởi dấu ngoặc kép.

**Values**: Chính là một kiểu dữ liệu Json hợp lệ. Nó có thể tồn tại ở các dạng khác nhau như array, object, chuỗi hoặc boolean,… Đôi khi Values còn có thể là số hoặc rỗng.

**Các loại values**

Array: Đây chính là bộ sưu tập values có thứ tự nhất định. Trong Json, nó được bao quanh bởi dấu ngoặc vuông. Mỗi value bên trong đều được phân tách bằng dấu phẩy. Một dạng Values Array có thể chứa Object của Json. Nói một cách dễ hiểu, nó có thể sử dụng cùng khái niệm cặp key/value. Values phổ biến nhất trong Json là gì chúng ta có thể khẳng định ngay là Array.

Object: Một Object có thể chứa cả value lẫn key. Người dùng có thể để dấu hai chấm sau mỗi key và dấu phẩy sau mỗi value. Điều này được dùng để phân biệt giữa mỗi Object. Trong Json, object được coi như một value. Nó phải tuân theo quy tắc tương tự như một Object.

Chuỗi: Chuỗi cũng được biết đến khá nhiều trong các values của Json. Nó là một bộ tập hợp gồm các số 0 và nhiều ký tự Unicode. Chuỗi được bao quanh bởi hai dấu ngoặc kép.

Số - Bên cạnh chuỗi thì số cũng có thể coi như một value. Số trong Json có thể là số nguyên hoặc số thực.

Boolean - giá trị True hoặc False

Rỗng - Rỗng thể hiện không có thông tin

### Dữ liệu XML

*Định nghĩa*

Cùng giống như JSON, XML là một ngôn ngữ đánh dấu. XML được viết tắt từ tên eXtensible Markup Language, có nghĩa là ngôn ngữ đánh dấu mở rộng. Ngôn ngữ này được World Wide Web Consortium (W3C) kiến nghị tạo nên để góp phần xây dựng các API Service. XML có khả năng truyền dữ liệu và đọc dịch nhiều loại dữ liệu khác nhau. API sẽ trả kết quả về dạng XML để các hệ thống khác nhau có thể nói chuyện với nhau được.

Nếu mô tả một cách đơn giản hơn thì XML giống như một máy phiên dịch ngôn ngữ giữa các hệ thống với nhau vậy. Ngoài chức năng thông dịch, nó còn giúp đơn giản hóa dữ liệu giữa các platform và hệ thống khác nhau. XML cũng có thể được dùng như một kho lưu trữ các dữ liệu đã trao đổi.

*Đặc điểm*

XML được dùng cho những loại dữ liệu có cấu trúc. XML có cấu tạo trực quan khá giống với HTML, nhưng vẫn khác nhau ở nhiều điểm. Có thể nói, XML là cầu nối để đưa HTML đến với XHTML.

Tuy XML tồn tại như một dạng văn bản nhưng nó không dùng để đọc, nó chỉ giúp các hệ thống khác nhau có thể thấu hiểu lẫn nhau.

Vậy định dạng XML là gì? XML có nhiều cách để định dạng khác nhau, dưới đây là một số định dạng phổ biến

RSS và ATOM: Chúng đều dùng để mô tả cách làm thế nào mà ứng dụng đọc xử lý được nguồn cấp web.

Microsoft .NET: Hệ thống này sẽ sử dụng XML cho các file cấu hình của nó.

Các phiên bản kể từ Microsoft Office 2007: Chúng sử dụng XML làm cơ sở cho cấu trúc tài liệu. Ký hiệu “X” trong định dạng tài liệu Word .DOCX cũng xuất phát từ đây.

*Khả năng đọc*

Tuy file XML không được dùng để đọc trực tiếp nhưng nó vẫn vô cùng hữu ích đối với các nhà phân tích dữ liệu. Trước mỗi thông tin quan trọng trong XML đều có tag name đầy đủ với ký tự text thân thiện, phù hợp với khả năng đọc hiểu của con người chứ không chỉ là ngôn ngữ máy tính khô khan. Các thông tin được sắp xếp gọn gàng và có tổ chức nên chúng ta có thể nhanh chóng tra cứu khi cần thiết. Ngoài ra, ngôn ngữ XML tương thích với hầu hết các loại dữ liệu, tạo nên sự trao đổi nhanh chóng giữa máy tính và file cần xử lý.

*Khả năng tương thích*

Các file XML mang tính độc lập. Điều này có nghĩa là chúng ta có thể tùy ý di chuyển chúng và sử dụng ở bất cứ vị trí, điều kiện nào. Điều kiện cần thiết để sử dụng XML là phải có phần mềm xử lý. Khi đó chúng ta có thể lưu trữ và truyền đi dữ liệu của mình.

*Khả năng tùy chỉnh*

XML là một loại ngôn ngữ đánh dấu mở rộng. Vì thế nó cho phép người dùng thoải mái tạo các tag riêng hoặc sử dụng các tag có sẵn từ người khác. Điều kiện để sử dụng tag của người khác cũng rất đơn giản. Chúng ta chỉ cần đảm bảo có ngôn ngữ tự nhiên của domain và chúng sở hữu đầy đủ các tính năng cần thiết. Số lượng tag được tạo cũng sẽ không bị giới hạn trong XML

## Tổng quan về Angular Framework

### Định nghĩa Angular Framework

Angular là một mã nguồn mở (open source) hay Javascript framework hoàn toàn miễn phí chuyên dụng dành cho công việc viết giao diện web. Đây là sản phẩm được viết bởi Misko Hevery và Adam Abrons (một người chúng ta của Misko Hevery). Sau đó, Angular chính thức được phát triển và duy trì bởi Google từ năm 2009.



Hình 2‑3 Angular framework

Hiểu theo cách đơn giản, Angular là khung làm việc của Javascript MVC phía client (máy khách) với mục đích phát triển ứng dụng web động. Angular được xem là framework frontend mạnh mẽ và vô cùng chuyên dụng bởi các lập trình viên thực hiện việc cắt HTML cao cấp.

Angular là một nền tảng phát triển được xây dựng dựa trên Javascript, có khả năng mở rộng quy mô không chỉ các dự án của một nhà phát triển mà còn các ứng dụng cấp doanh nghiệp, nó bao gồm:

1. Một khuôn khổ xây dựng ứng dụng web có khả năng mở rộng dựa trên các thành phần.
2. Một bộ sưu tập thư viện tốt sẽ được tích hợp nhiều tính năng như quản lý biểu mẫu, định tuyến, giao tiếp máy khách-máy chủ.
3. Một bộ công cụ dành riêng cho nhà phát triển để chúng ta có thể phát triển, xây dựng và cập nhật nhanh chóng các bộ mã.

AngularJS được viết bằng Javascript và là từ dùng để nói đến Angular 1 (ra đời vào năm 2009). Angular 2 ( ra đời năm 2016) trở nên sẽ được gọi chung là Angular và được viết bởi phiên bản nâng cao của Javascript. Angular có sự thay đổi nhiều từ AngularJS, do đó cấu trúc của chúng khác nhau hoàn toàn.

### Lịch sử phát triển

14/09/2016: Angular 2 – Initial Version of Angular

23/03/2017: Angular 4 – Version 4: giảm thiểu code được tạo ra, giảm xuống 60% kích thước tệp được đóng gói, đẩy nhanh quá trình phát triển ứng dụng.

11/11/2017: Angular 5 – Version 5: sử dụng HTTPClient thay HTTP, công cụ build optimizer được tích hợp sẵn vào trong CLI, …

03/05/2018: Angular 6 – Version 6: Cập nhật CLI, command line interface, Multiple Validators, …

18/10/2018: Angular 7 – Version 7: ScrollingModule scroll load dữ liệu, dùng Drag and Drop, cập nhật RxJS 6.3, …

25/08/2019: Angular 8 – Version 8: CLI workflow improvements, Dynamic imports for lazy routes ….

06/02/2020: Angular 9.0: di chuyển tất cả các ứng dụng để sử dụng trình biên dịch Ivy và thời gian chạy theo mặc định, dùng TypeScript 3.6 và 3.7

25/03/2020: Angular 9.1

08/04/2020: Angular 10 – Beta Version

### Các thành phần cơ bản

*Thành phần (Components)*

Một thành phần Angular là một class Typescript tương tác với một view. Một view bao gồm một thành phần và một template. Một template là HTML kết hợp vơi cú pháp liên kết dữ liệu của Angular. Một thành phần có các trường (fields) và các phương thức (methods) hỗ trợ cho một view. View sau đó sẽ được cập nhật bằng các đặc tính và phương thức này. Angular tạo và hủy tất cả các thành phần khi người dùng di chuyển trong ứng dụng.

*Liên kết dữ liệu (Data binding)*

Liên kết dữ liệu là một cải thiện lớn của Angular. Nó cho phép dữ liệu truyền qua giữa các thành phần và template của chúng ta. Nếu không có liên kết dữ liệu, các lập trình viên Angular sẽ phải viết code để đẩy dữ liệu vào giữa các thành phần và template. Liên kết dữ liệu không giới hạn ở giao tiếp giữa một template cha và thành phần, nó còn có thể chuyển dữ liệu giữa một thành phần cha với một thành phần con.

*Modules*

Angular có các module được dựng sẵn để hỗ trợ các lập trình viên, các module này được gọi là NgModules. Các NgModule luôn luôn được đánh dấu bằng chú thích @NgModule trong Angular. Một số module thường gặp là Forms Module, RouterModule, HttpClientModule, và module thiết kế tư liệu Angular.

Ý tưởng về module đến từ nguyên tắc trách nhiệm đơn. Trong khi một module có thể làm nhiều việc thì ở cấp độ cao hơn, nó sẽ chỉ tập trung vào một công việc cụ thể như forms hoặc định tuyến. Điều này ngăn việc phình ứng dụng (application bloating) và sắp xếp các ứng dụng thành các khối chức năng ngắn gọn. Lazy loading có thể giúp cải thiện thời gian tải và tốc độ

*Services*

Là từ chỉ chung cho bất kỳ đoạn code nào phục vụ một mục đích cụ thể cho ứng dụng. Đây là một khái niệm khá rộng nhưng sẽ có vài chỉ dẫn mà một service nên tuân theo. Một Angular service phải tái sử dụng được và tuân theo nguyên tắc trách nhiệm đơn đó là làm một công việc và phải làm tốt công việc đó. Khi một thành phần làm việc với view thì một service sẽ xử lý tất cả những gì không liên quan đến view đó.

Chẳng hạn như những công việc lặp lại bao gồm logic nghiệp vụ, lấy dữ liệu hay xác thực input, những công việc thiết yếu cho ứng dụng nhưng không liên quan đến thị giác sẽ thuộc về phần service. Bằng cách định nghĩa logic này trong service, có thể sẵn sàng inject nó vào bất cứ đâu trong ứng dụng, cung cấp cho chúng ta khả năng tái sử dụng vô hạn. Tuy nhiên Angular cũng không bắt buộc chúng ta phải thực hiện bất kỳ điều gì kể trên. Một service phức tạp không phải là hiếm thấy, nhưng một service tốt sẽ tuân theo các quy tắc trên.

### Ưu điểm của Angular

*Liên nền tảng (Cross-platform)*

Angular có thể phát triển các ứng dụng web cấp tiến (Progressive Web Applications – PWA). PWA cũng cung cấp trải nghiệm như ứng dụng (app-like) cho người dùng sử dụng các công nghệ web tân tiến. Tùy theo nhu cầu có thể triển khai một ứng dụng gốc cũng như cấp tiến. Bộ công cụ phát triển phần mềm kết hợp được gọi là Ionic có thể đưa ứng dụng lên app store hoặc triển khai nó lên mobile web như một PWA.

*Tốc độ cao và tối ưu hiệu năng*

Angular định nghĩa lại máy ảo JavaScipt hiện đại bằng cách chuyển các template thành code. Bởi vây, code viểt tay của chúng ta cũng có thể được tận dụng trong framework này. Điều tuyệt vời nhất của Angular đó là có thể render code thành HTML & CSS và có cái nhìn đầu tiên về ứng dụng trên các nền tảng khác như Node.js, .NET, PHP,..v..v..

Thời gian tải của Angular nhanh hơn bất kỳ front-end framework nào ở thời điểm hiện tại. Angular tải cực kì nhanh với bất kỳ router thành phần mới nào. Hơn nữa, code sẽ tự động được chia ra đối với mỗi khi người dùng load và yêu cầu render.

*Kiến trúc MVC*

 Kiến trúc này đã được đề cập đến trong phần đầu của bài viết. Dữ liệu của ứng dụng sẽ được quản lý bởi Model, View quản lý phần hiển thị dữ liệu. Controller sẽ đóng vai trò là bộ nối giữa lớp View và lớp Model. Thông thường trong kiến trúc MVC, có thể chia nhỏ ứng dụng một cách tùy ý và viết code để nối nó lại với nhau. Tuy nhiên trong Angular, lập trình viên chỉ cần chia nhỏ ứng dụng thành các phần của MVC và framework sẽ tự làm những công việc còn lại. Điều này sẽ tiết kiệm một cách đáng kể thời gian code.

*Liên kết dữ liệu hai chiều*

Một trong những tính năng hiệu quả nhất của Angular đó là hệ thống liên kết dữ liệu hai chiều. Lớp View thể hiện chính xác lớp Model và hai lớp này sẽ phối hợp đồng bộ với nhau một cách hoàn hảo. Nếu thay đổi bất kỳ cái gì trong Model, người dùng có thể thấy rõ thay đổi đó trong View.

*Framework ít code*

Angular là một framework ít code so với các công nghệ front-end khác. Lập trình viên không cần phải viết code riêng để  kết nối các lớp MVC. Và cũng không cần đến các code cụ thể để xem một cách thủ công. Các chỉ thị được tách biệt với code ứng dụng. Tất cả sẽ tiết kiệm công sức và thời gian cần thiết để phát triển ứng dụng.

*Angular CLI (Giao diện dòng lệnh)*

Angular CLI tuân theo các nguyên tắc làm việc tổt nhất của lập trình front-end với các tính năng hỗ trợ SASS/SCSS (Chương trình tiền xử lý CSS - CSS preprocessor) và định tuyến. Hơn nữa Angular CLI thông thường như ng-new hay ng-add hỗ trợ các lập trình viên dễ dàng khám phá các tính năng có sẵn.

*CDK và Angular Material*

Angular nâng cấp Bộ công cụ phát triển thành phần (Component Development Kit – CDK) theo mỗi phiên bản mới. Làm mới (Refreshing) và cuộn ảo (virtual scrolling) là các tính năng tiêu biểu nhất của Angular CDK. Nó cũng hỗ trợ cho việc tải động cũng như dỡ tải DOM để tạo ra một danh sách lớn các dữ liệu hiệu năng cao. Chúng ta có thể import ScrollingModule hay DragDropModule vào ứng dụng.

Một số chức năng chính của DragDropModule bao gồm sắp xếp trong một danh sách, kéo thả tự do, xem trước, và các phần giữ chỗ. Thêm nữa, nó còn có thể sắp xếp lại thứ tự danh sách (moveItemInArray) và chuyển các vật thể qua lại giữa các danh sách (transferArrayItem).  
  *Cuộn ảo*

Tính năng cuộn ảo của Angular giúp code phản ứng với các sự kiện cuộn. Ngoài việc tải và dỡ tải các phần tử DOM có thể thấy được, cuộn ảo còn cung cấp chức năng giả lập vật thể.

*TypeScript*

Với TypeScript, Angular cung cấp một trải nghiệm liền mạch cho các lập trình viên. Nó rất hiệu quả trong việc phát hiện lỗi. TypeScript cũng tự động gắn các thiết lập của tệp root dể biên dịch dễ dàng hơn. So với JavaScript, TypeScript có các kiểu liệt kê, giao diện, các lớp generic, các kiểu kết hợp, các kiểu union/intersection, các bộ điều chỉnh truy cập đặc sắc hơn.

*Inject các phần phụ thuộc*

Chức năng inject phần phụ thuộc có sẵn sẽ giảm tải công việc lập trình ứng dụng đi rất nhiều. Nó giúp chúng ta có thể nhúng service vào các component hoặc các service với nhau tùy theo nhu cầu của lập trình viên và của ứng dụng.

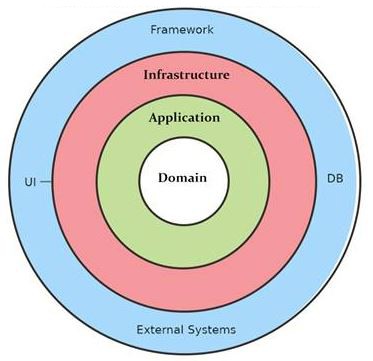
*Các chỉ thị ( Directives)*

Chỉ thị là tính năng mang tính thử thách nhất của Angular. Chúng ta có thể tạo ra các thẻ HTML tùy biến với vai trò là các widget tùy biến với các chỉ thị. Chúng ta cũng có thể tùy ý thao tác với các thuộc tính DOM bằng các chỉ thị.

## Kiến trúc Onion Architecture

### Ý tưởng

Là một thiết kế phần mềm tách các yếu tố thành cấp độ vòng, có quy tắc duy trì xoay quanh phần lõi (Core) đồng thời chia kiến trúc code thành các tầng, lớp riêng biệt phục vụ tốt cho việc phát triển



Hình 2‑4 Mô hình Onion Architechture

### Kiến trúc chung

*Lớp lõi (Core)*

Lớp tập trung các thành phần quan trọng nhất của ứng dụng – Các thực thể

*Lớp miền (Domain)*

Lớp miền trong kiến ​​trúc chứa logic, chúng có vai trò quan trong tương đương các thực thể Lớp này nằm ở trung tâm của kiến ​​trúc nơi chúng ta có các thực thể ứng dụng, là các lớp mô hình ứng dụng hoặc các lớp mô hình cơ sở dữ liệu, sử dụng cách tiếp cận mã đầu tiên trong phát triển ứng dụng bằng ORM khác nhau, các thực thể này được sử dụng để tạo các bảng trong cơ sở dữ liệu .

*Lớp ứng dụng*

Lớp ứng dụng chứa logic nghiệp vụ. Tất cả logic nghiệp vụ sẽ được triển khai trong lớp này. Trong lớp này, các giao diện dịch vụ được giữ, tách biệt với việc triển khai chúng, để kết nối lỏng lẻo và phân tách các mối quan tâm.

Lớp cơ sở hạ tầng:

Trong lớp cơ sở hạ tầng, chúng tôi có các đối tượng mô hình, chúng tôi sẽ duy trì tất cả các đối tượng di chuyển cơ sở dữ liệu và bối cảnh cơ sở dữ liệu trong lớp này. Trong lớp này, chúng ta có kho lưu trữ của tất cả đối tượng mô hình miền. Ngoài ra việc triển khai giao diện được xác định trong lớp ứng dụng.

Lớp trình bày:

Nó có thể là bộ điều khiển API hoặc bất kỳ giao diện người dùng nào. Trong trường hợp lớp Trình bày API hiển thị cho chúng ta dữ liệu đối tượng từ cơ sở dữ liệu bằng cách sử dụng yêu cầu HTTP ở dạng Đối tượng JSON. Nhưng trong trường hợp ứng dụng ngoại vi, chúng tôi trình bày dữ liệu bằng giao diện người dùng bằng cách sử dụng APIS.

### Ưu điểm

1. Tách biệt mối quan tâm, mỗi bộ phận có trách nhiệm duy trì riêng.
2. Luồng phụ thuộc hướng vào trong, điều đó có nghĩa là có thể thay đổi các phần bên ngoài mà không ảnh hưởng gì đến các phần bên trong.
3. Cung cấp một cách rất linh hoạt để thử nghiệm.
4. Cung cấp một cách rõ ràng, ngắn gọn và có hướng dẫn để bắt đầu triển khai bằng bất kỳ ngôn ngữ lập trình nào.
5. Giao diện người dùng và cơ sở dữ liệu có thể được thay đổi bất cứ lúc nào mà không ảnh hưởng đến các phần khác.

## Thư viện Bootstrap

### Giới thiệu thư viện Bootstrap

Bootstrap là một Framework gồm có 3 thành phần cơ bản như sau: HTML, CSS và JavaScript. Responsive là nơi mà chúng dựa theo và sử dụng để phát triển trang web. Bootstrap giúp cho các nhà lập trình thiết kế trong việc thiết kế web tiết kiệm được thời gian



Hình 2‑5 Thư viện giao diện Bootstrap

Bootstrap còn là một collection hoàn toàn miễn phí. Bộ sưu tập này gồm có các mã nguồn mở, hay các công cụ giúp các lập trình viên có thể tạo thành một website đầy đủ với các thành phần.

### Các tính năng chính của Bootstrap

*Cải tiến các biển mẫu*

Chúng có một hệ thống có thể điều khiển biểu mẫu từ đó được cải tiến. Các biểu mẫu này hoàn toàn thao tác được tùy chỉnh, chính vì thế mà các nhà phát triển có thể cung cấp giao diện giống nhau trên tất cả các trình duyệt. Sau cùng thì các biểu mẫu sẽ dễ làm việc và được thống nhất hơn trên các trình duyệt khác nhau.

Các cải tiến đối với biểu mẫu Bootstrap được đơn giản hóa. Sử dụng CSS và hỗ trợ bởi floating label để tạo ra các biểu mẫu mới để nhập văn bản, vùng văn bản và đưa ra lựa chọn.

*Hỗ trợ các RTL*

Bootstrap hỗ trợ các RTL cho các ngôn ngữ đọc từ phải sang trái. Việc hỗ trợ RTL là một bước ngoặt lớn, một sự cải tiến lớn về khả năng truy cập trong Bootstrap  mang vai trò loại bỏ các vấn đề mà các developer gặp phải khi cần RTL trên trang web.

*Cải tiến từ jQuery*

Bootstrap sẽ ngưng sử dụng thư viện jQuery, mà thay thế chúng là thư viện JavaScript được cải thiện từ trước. Ngừng việc hỗ trợ jQuery giúp việc cải thiện thời gian tải và giảm được kích thước tệp nguồn, có được sự sẵn sàng hơn trong tương lai. Quá trình phải tải xuống toàn bộ thư viện sẽ không còn tiếp diễn nữa, đây là một dấu hiệu tốt. Vì sao lại như thế, bởi điều này đóng một vai trò quan trọng khi thời gian tải trang Google dựa vào đó để đánh giá thứ hạng của các website tên thiết bị di động.

Bootstrap và jQuery đã gắn bó với nhau trong suốt 8 năm, với trách nhiệm cung cấp cho các nhà phát triển các khả năng chưa từng có tiền lệ trong các chức năng dựa trên JavaScript và khả năng đơn giản hóa nhiều tác vụ.

*Thành phần Offcanvas*

Thành phần offcanvas là một trong số các tính năng của Bootstrap. Trong thành phần này còn bao gồm backdrop có thể định dạng cấu hình, cuộn nội dung và cả vị trí. Biểu tượng chữ V thuộc phần tử trong Bootstrap hiển thị khả năng nhấp và trạng thái. Chúng có thể được đặt ở trên cùng, dưới cùng, bên trái hay bên phải của chế độ xem.

*Cải tiến hệ thống bố cục và lưới*

Giúp cho hệ thống lưới và bố cục được cải tiến hóa và tối ưu các chức năng

### Ưu điểm của Bootstrap

*Tương thích với các trình duyệt*

Khả năng tương thích của framework với nhiều loại phiên bản và nền tảng của các trình duyệt hiện nay luôn là sự đảm bảo tuyệt đối khi lựa chọn Bootstrap. Ngoài ra, các developer cam kết sẽ không hỗ trợ proxy browser hay kể cả trình duyệt cũ.

*Tiết kiệm thời gian*

Khi các nhà phát triển phải vùi đầu trong việc chạy deadline khi xây dựng trang web thì Bootstrap như người chúng ta đồng hành với họ. Vì sao lại nói như thế? Bởi chúng đã có sẵn rất nhiều block được cài sẵn giúp phần nào thời gian lựa chọn của ng làm và đẩy nhanh được tiến độ công việc.

*Khả năng ứng biến tốt*

Khi các nhà phát triển không lựa chọn được các mẫu thiết kế có sẵn của chúng, thì vẫn còn một cách khác. Đó là có thể tùy ý chỉnh lại biến tấu vào tệp CSS của nó. Hay có thể kết hợp với các mã sẵn có và bổ trợ thêm các chức năng cho nhau.

Người dùng có thể tạo một phiên bản tùy chỉnh thông qua trang của chúng. Phân tích sâu hơn nữa thì để thực hiện được điều vừa rồi chúng ta cần loại bỏ toàn bộ các plugin và thành phần trước khi chúng ta tải xuống tệp Bootstrap.

*Dễ sử dụng*

Nghe có vẻ như khó sử dụng nhưng chúng lại rất dễ sử dụng và thao tác. Chỉ cần có kiến thức cơ bản về CSS và HTML thì đều có thể sử dụng được. Ngoài ra còn một điều khác nữa đó chính là chúng rất phổ biến trong giới lập trình web và thiết kế với cấu trúc đơn giản dễ áp dụng.

Chúng ta cũng có thể sử dụng các màu sắc chủ đề cho các CMS phổ biến làm công cụ cho việc học tập. Với mong muốn trang web được tăng thời gian tải trang thì thuật ngữ này có thể giúp giảm các tệp JavaScript và CSS. Không chỉ vậy, nó còn duy trì sự thống nhất về cú pháp giữa các trang web và nhà phát triển, khi thực hiện làm về một dự án nhóm thì rất phù hợp.

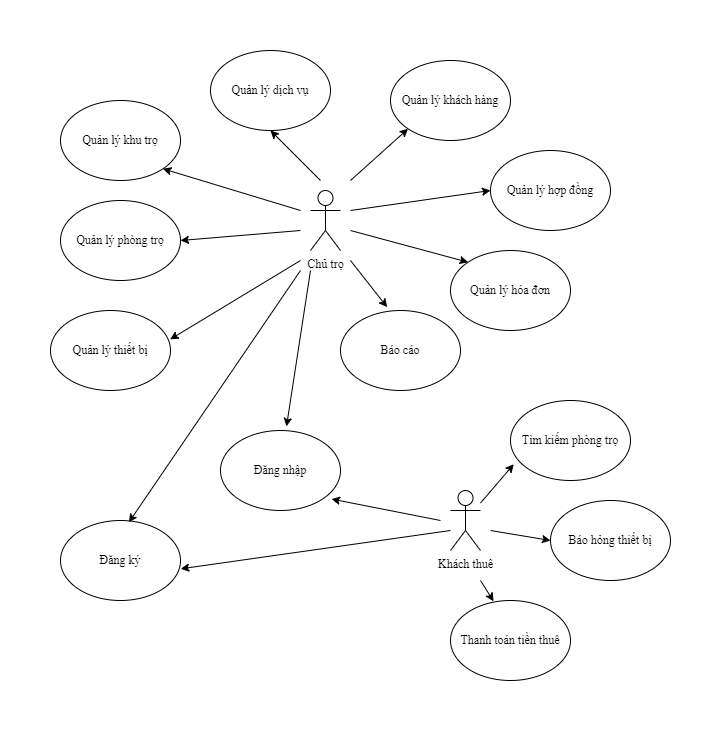
*Có tài liệu hỗ trợ đủ lớn*

Bootstrap sở hữu một cộng đồng lớn các nhà phát triển và nhà thiết kế hùng hậu đằng sau nó. Bởi vì được lưu trữ trên GitHub giúp các developer dễ dàng sáng tạo, sửa đổi và đóng góp vào cơ sở mã của Bootstrap. Không chỉ thế mà còn giúp dễ dàng trong việc cộng tác, đưa ra lời khuyên và  khả năng tương tác với đồng nghiệp được tăng lên…

Ngoài ra, rất nhiều tài nguyên trên trang web và một số trang web khác đều được chúng cung cấp một cách đầy đủ. Thuận tiện cho việc tạo một trang web không còn phức tạp như trước nữa. Tài liệu gồm có các code sample phục vụ cho việc thao tác cơ bản. Chúng ta có thể sao chép và sửa đổi các code template này sao cho các dự án của mình được hoàn hảo và chuyên nghiệp nhất, từ đó thời gian lập trình cũng được rút ngắn lại.

# PHÂN TÍCH VÀ THIẾT HỆ THỐNG

## Biểu đồ Use Case tổng quát



Hình 3‑1 Biểu đồ Use case tổng quát hệ thống

Các use case chính trong hệ thống phân theo vai trò

**Quản trị - chủ trọ**

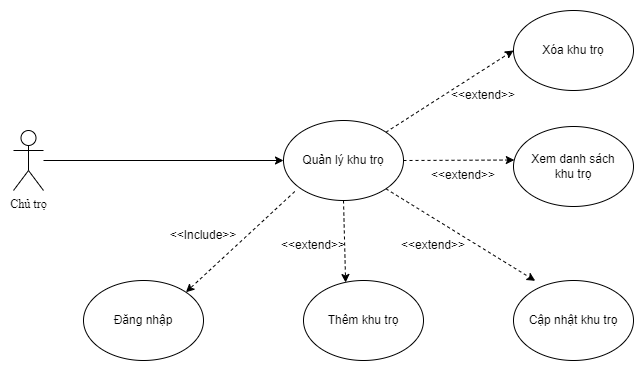
* Quản lý khu trọ
* Quản lý phòng trọ
* Quản lý các thiết bị trong phòng
* Quản lý các dịch vụ trong khu trọ
* Quản lý khách hàng
* Quản lý hợp đồng
* Quản lý hóa đơn
* Xem báo cáo, xuất tệp

**Khách hàng – người thuê**

* Tra cứu phòng trọ
* Báo hỏng thiết bị, tình trạng phòng trọ
* Thanh toán tiền trọ

## Use case chi tiết

### Use case quản lý khu trọ

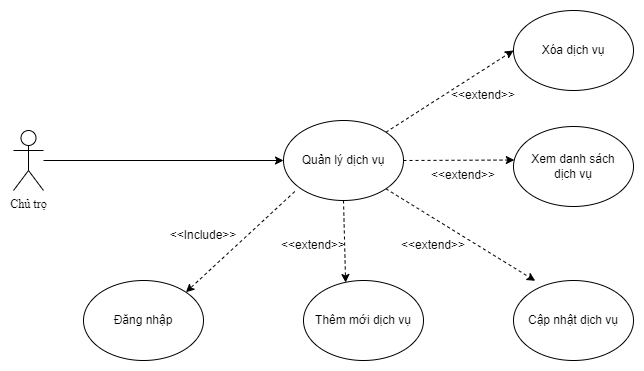


Hình 3‑2 Biểu đổ Use case quản lý khu trọ

### Use case quản lý phòng trọ

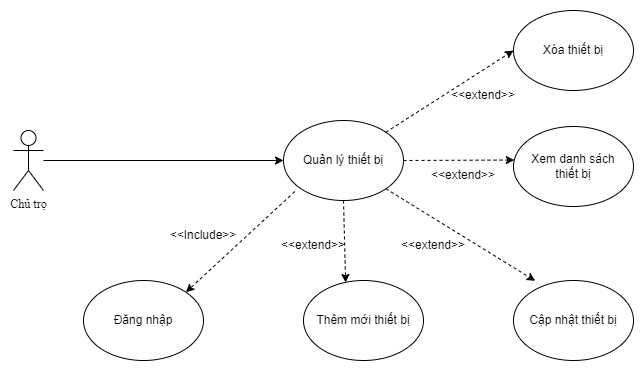
Hình 3‑3 Biểu đồ use case quản lý phòng trọ

### Use case quản lý dịch vụ



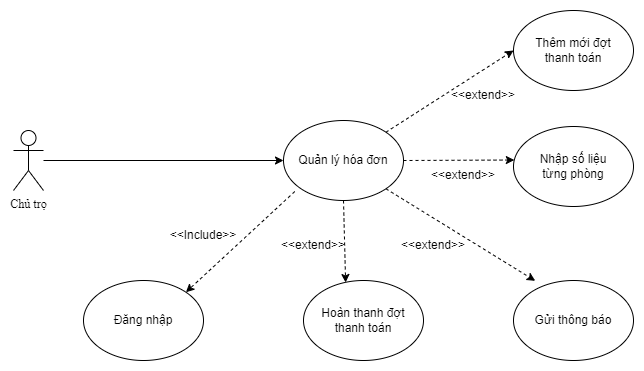
Hình 3‑4 Biểu đồ use case quản lý dịch vụ

### Use case quản lý thiết bị



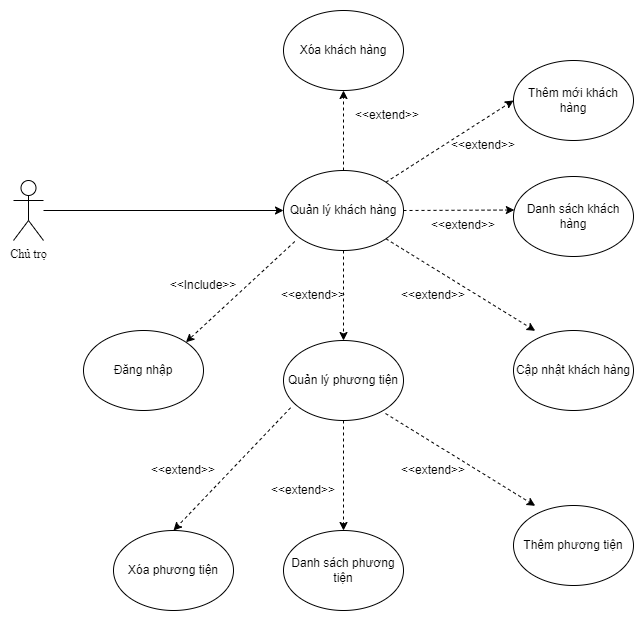
Hình 3‑5 Biểu đồ use case quản lý thiết bị

### Use case quản lý hóa đơn



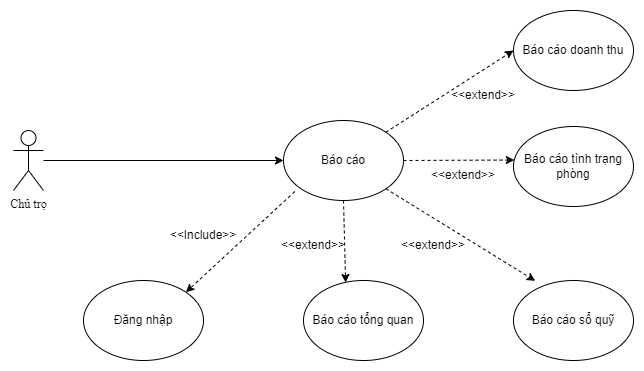
Hình 3‑6 Quản lý hóa đơn thuê phòng

### Use case quản lý khách hàng

******

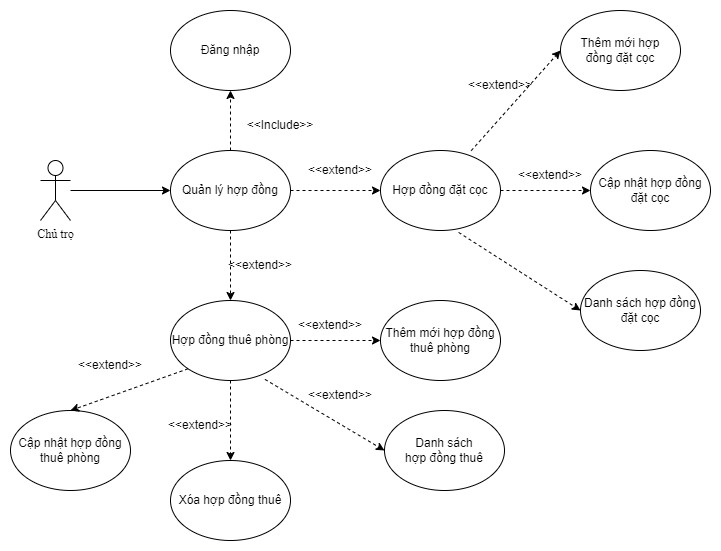
Hình 3‑7 Biểu đồ use case quản lý khách hàng

### Use case báo cáo



Hình 3‑8 Biểu đồ use case báo cáo

### Use case quản lý hợp đồng



Hình 3‑9 Biểu đồ use case quản lý hợp đồng

## Mô tả chi tiết các use case

### Đăng nhập

**Mô tả vắn tắt**

Use case này cho phép người sử dụng ở đây là chủ trọ hoặc người thuê trọ đăng nhập vào hệ thống

**Luồng sự kiện**

* ***Luồng cơ bản***

1. Use case này bắt đầu người dùng truy cập vào hệ thống, hệ thống sẽ hiển thị màn hình đăng nhập yêu cầu nhập tên đăng nhập (Số điện thoại hoặc Email) và mật khẩu
2. Người dùng nhập tên đăng nhập và mật khẩu, sau đó nhấn nút đăng nhập hoặc phím Enter trên bàn phím. Hệ thống sẽ kiểm tra tính xác thực của 2 thông tin trên và đưa ra kết quả đăng nhập

* ***Luồng rẽ nhánh***

Tại luồng cơ bản 1 trong trường hợp người dùng nhập tên đăng nhập hoặc mật khẩu không đúng định dạng về email hoặc số điện thoại hệ thống sẽ hiển thị thông báo yêu cầu nhập lại, người dùng có thể nhập lại thông tin hoặc thoát hệ thống, use case kết thúc

Tại luồng cơ bản 2 trong trường hợp người dùng đã nhập đúng định dạng, hệ thống sẽ kiểm tra thông tin đăng nhập, nếu thông tin đăng nhập là chính xác hệ thống sẽ chuyển sang màn hình quản trị. Ngược lại nếu thông tin đăng nhập không chính xác hệ thống đưa ra thông báo, người dùng có thể tiếp tục sử dụng thông tin khác để đăng nhập hoặc thoát, use case kết thúc

### Đăng ký

**Mô tả vắn tắt**

Use case này cho phép người dùng đăng ký tài khoản để có thể sử dụng hệ thống

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang đăng ký. Hệ thống sẽ hiển thị màn hình đăng ký để người dùng nhập thông tin bao gồm Số điện thoại, email, họ tên.

* ***Luồng rẽ nhánh***

Tại luồng cơ bản, khi người sử dụng nhập đầy đủ thông tin, hệ thống sẽ kiểm tra thông tin đăng ký có trùng với thông tin của người dùng khác đã đăng ký trước đó chưa.

* Nếu thông tin đã trùng, hệ thống đưa ra thông báo, người dùng có thể sử dụng thông tin khác để đăng ký sử dụng, hoặc thoát khỏi màn hình này. Use case kết thúc
* Nếu thông tin không trùng hệ thống sẽ gửi một email chứa mã xác thực (OTP) đến email người dùng đã điền trước đó. Người dùng sử dụng mã OTP này để xác thực trên hệ thống. Mã OTP chính xác hệ thống đưa ra thông báo đăng ký thành công, use case kết thúc. Mã OTP không chính xác hệ thống yêu cầu nhập lại người dùng có thể nhập lại mã OTP hoặc thoát khỏi hệ thống. Use case kết thúc.

### Quản lý khu trọ

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý các khu trọ sở hữu

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý khu trọ, hệ thống sẽ hiển thị danh sách các khu trọ mà người dùng sở hữu và trạng thái các khu trọ đó.

1. **Thêm khu trọ** Khi người dùng nhấn nút thêm khu trọ, hệ thống sẽ xuất hiện một cửa sổ để người dùng nhập thông tin khu trọ như tên, địa chỉ, mô tả, thời gian thanh toán và các dịch vụ khu trọ đang có, khi người dùng nhấn Lưu dữ liệu sẽ được thêm vào bảng BoardingHouses, đồng thời cập nhật lại màn hình danh sách, và đưa ra thông báo đã thêm khu trọ thành công. Use case kết thúc
2. **Chỉnh sửa khu trọ** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa khu trọ, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin về khu trọ, sau khi người dùng nhấn Lưu, dữ liệu sẽ được cập nhật trong bảng BoardingHouses và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Xóa khu trọ** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin. Hệ thống đưa ra cảnh báo về việc xóa khu trọ, nếu người dùng xác nhận xóa, dữ liệu sẽ được xóa khỏi bảng BoardingHouses và các bảng liên quan, đồng thời đưa ra thông báo xóa thành công cho người dùng. Use case kết thúc

* ***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa khu trọ, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc

### Quản lý phòng trọ

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý các phòng trọ thuộc các khu trọ đang sở hữu

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý phòng trọ, hoặc di chuyển từ màn hình quản lý khu trọ. Hệ thống sẽ hiển thị các phòng trọ và trạng thái các phòng trọ theo các khu trọ

1. **Thêm phòng trọ** Khi người dùng nhấn nút *Thêm phòng trọ*, hệ thống sẽ xuất hiện một cửa sổ để người dùng nhập thông tin phòng trọ như khu trọ của phòng trọ, tên phòng trọ, giá tiền, các hình ảnh về phòng trọ, các thiết bị mà phòng trọ đang có, khi người dùng nhấn nút *Thêm mới phòng trọ* dữ liệu sẽ được lưu vào bảng Rooms và các bảng liên quan đồng thời hệ thống cũng dưa ra thông báo thêm mới phòng trọ thành công. Use case kết thúc
2. **Chỉnh sửa phòng trọ** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa phòng trọ, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin về phòng trọ, sau khi người dùng nhấn *Cập nhật*, dữ liệu sẽ được cập nhật trong bảng BoardingHouses và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Chuyển đổi trạng thái** Khi người dùng nhấn vào biểu tượng menu và chọn *tạm dừng cho thuê / tiếp tục* cho thuê tại mỗi dòng thông tin. Hệ thống sẽ đưa ra yêu cầu xác nhận về việc chuyển đổi trạng thái sang *Tạm dừng cho thuê / tiếp tục cho thuê*, sau khi người dùng nhấn xác nhận, dữ liệu bảng Rooms sẽ được cập nhật, đồng thời đưa ra thông báo cho người dùng về việc chuyển đổi trạng thái thành công. Use case kết thúc.
4. **Xóa phòng trọ** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin, hệ thống sẽ đưa ra yêu cầu xác nhận về việc xóa phòng trọ, sau khi người dùng nhấn *xác nhận* dữ liệu sẽ được xóa khỏi bảng Rooms đồng thời đưa ra thông báo về việc xóa phòng trọ thành công. Use case kết thúc.

* ***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 và 4 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận hành động, người dùng có thể nhấn Đồng ý để thực hiện yêu cầu hoặc Nhấn hủy bỏ để không thực hiện yêu cầu. Use case kết thúc

### Quản lý thiết bị

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý danh mục các thiết bị trong các khu trọ đang sở hữu

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý thiết bị, hệ thống sẽ hiển thị danh sách các thiết bị và trạng thái, giá mua các thiết bị đó

1. **Thêm mới thiết bị** Khi người dùng nhấn nút *Thêm mới thiết bị* hệ thống sẽ hiển thị một cửa sổ để người dùng nhập thông tin về thiết bị như tên, giá mua, tình trạng. Sau khi người dùng nhấn *Thêm mới* dữ liệu sẽ được thêm vào bảng Fitments và đưa ra thông báo cho người dùng về việc đã thêm thiết bị thành công. Use case kết thúc
2. **Chỉnh sửa thiết bị** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa thiết bị, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin về thiết bị, sau khi người dùng nhấn Lưu, dữ liệu sẽ được cập nhật trong bảng *Fitments* và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Xóa thiết bị** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin. Hệ thống đưa ra cảnh báo về việc xóa thiết bị, nếu người dùng xác nhận xóa, dữ liệu sẽ được xóa khỏi bảng Fitments và các bảng liên quan, đồng thời đưa ra thông báo xóa thành công cho người dùng. Use case kết thúc
4. **Xuất tệp danh sách** Khi người dùng nhấn vào nút *Xuất tệp* hệ thống sẽ xử lý tạo ra file Excel người dùng chọn vị trí lưu rồi nhấn *OK* để tải file xuống. Use case kết thúc

* ***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa khu trọ, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc
3. Tại luồng cơ bản 4 khi thực hiện thao tác xuất tệp hệ thống sẽ hiển thị vị trí lưu, người dùng chọn vị trí lưu và nhấn OK hoặc nhấn Cancel để hủy bỏ thao tác xuất tệp. Use case kết thúc

### Quản lý dịch vụ

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý các dịch vụ trong khu trọ

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý dịch vụ, hệ thống sẽ hiển thị danh sách các thiết bị và hình thức thu tiền các dịch vụ đó

1. **Thêm mới dịch vụ** Khi người dùng nhấn nút *Thêm mới dịch vụ* hệ thống sẽ hiển thị một cửa sổ để người dùng nhập thông tin về dịch vụ như tên dịch vụ, hình thức thu tiền. Sau khi người dùng nhấn *Thêm mới* dữ liệu sẽ được thêm vào bảng Services và đưa ra thông báo cho người dùng về việc đã thêm dịch vụ thành công. Use case kết thúc
2. **Chỉnh sửa dịch vụ** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa dịch vụ, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin về thiết bị, sau khi người dùng nhấn Lưu, dữ liệu sẽ được cập nhật trong bảng *Services* và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Xóa dịch vụ** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin. Hệ thống đưa ra cảnh báo về việc xóa thiết bị, nếu người dùng xác nhận xóa, dữ liệu sẽ được xóa khỏi bảng *Services* và các bảng liên quan, đồng thời đưa ra thông báo xóa thành công cho người dùng. Use case kết thúc
4. **Xuất tệp danh sách** Khi người dùng nhấn vào nút *Xuất tệp* hệ thống sẽ xử lý tạo ra file Excel người dùng chọn vị trí lưu rồi nhấn *OK* để tải file xuống. Use case kết thúc

* ***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa khu trọ, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc
3. Tại luồng cơ bản 4 khi thực hiện thao tác xuất tệp hệ thống sẽ hiển thị vị trí lưu, người dùng chọn vị trí lưu và nhấn OK hoặc nhấn Cancel để hủy bỏ thao tác xuất tệp. Use case kết thúc

### Quản lý khách hàng

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý các khách đang thuê trọ và phương tiện của khách

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý khách hàng, hoặc di chuyển từ màn quản lý phòng trọ

1. **Thêm mới khách hàng** Khi người dùng nhấn nút *Thêm mới khách hàng* hệ thống sẽ đưa ra một cửa sổ để người dùng có thể nhập thông tin của khách hàng như họ tên, ảnh căn cước công dân trước và sau, số điện thoại, email, số liên hệ khẩn cấp. Sau khi hoàn thành xong thông tin người sử dụng nhân nút *Lưu khách hàng.* Dữ liệu sẽ được lưu vào bảng Customers, đồng thời đưa ra thông báo đã thêm thành công cho khách hàng
2. **Chỉnh sửa dịch vụ** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa dịch vụ, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin về thiết bị, sau khi người dùng nhấn Lưu, dữ liệu sẽ được cập nhật trong bảng *Services* và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Xóa dịch vụ** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin. Hệ thống đưa ra cảnh báo về việc xóa thiết bị, nếu người dùng xác nhận xóa, dữ liệu sẽ được xóa khỏi bảng *Services* và các bảng liên quan, đồng thời đưa ra thông báo xóa thành công cho người dùng. Use case kết thúc
4. **Xuất tệp danh sách** Khi người dùng nhấn vào nút *Xuất tệp* hệ thống sẽ xử lý tạo ra file Excel người dùng chọn vị trí lưu rồi nhấn *OK* để tải file xuống. Use case kết thúc

* ***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa khu trọ, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc
3. Tại luồng cơ bản 4 khi thực hiện thao tác xuất tệp hệ thống sẽ hiển thị vị trí lưu, người dùng chọn vị trí lưu và nhấn OK hoặc nhấn Cancel để hủy bỏ thao tác xuất tệp. Use case kết thúc

### Quản lý phương tiện

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý các phương tiện của khách hàng trong hệ thống

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý phương tiện hoặc di chuyển từ màn hình quản lý khách hàng

1. **Thêm mới phương tiện** Khi người dùng nhấn nút *Thêm mới phương tiện* hệ thống sẽ đưa ra một cửa sổ để người dùng có thể nhập thông tin của phương tiện như biển kiểm soát, loại xe, màu xe, và chọn người thuê đang sở hữu sau khi hoàn thành thông tin người dùng nhấn *Lưu phương tiện.* Nếu thông tin hợp lệ hệ thống sẽ lưu dữ liệu vào bảng *Vehicles* đồng thời đưa ra thông báo lưu dữ liệu thành công. Use case kết thúc
2. **Chỉnh sửa phương tiện** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa phương tiện, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin về phương tiện, sau khi người dùng nhấn Lưu, dữ liệu sẽ được cập nhật trong bảng *Vehicles* và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Xóa phương tiện** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin. Hệ thống đưa ra cảnh báo về việc xóa thiết bị, nếu người dùng xác nhận xóa, dữ liệu sẽ được xóa khỏi bảng *Vehicles* và các bảng liên quan, đồng thời đưa ra thông báo xóa thành công cho người dùng. Use case kết thúc
4. **Xuất tệp danh sách** Khi người dùng nhấn vào nút *Xuất tệp* hệ thống sẽ xử lý tạo ra file Excel người dùng chọn vị trí lưu rồi nhấn *OK* để tải file xuống. Use case kết thúc

***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa phương tiện, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc
3. Tại luồng cơ bản 4 khi thực hiện thao tác xuất tệp hệ thống sẽ hiển thị vị trí lưu, người dùng chọn vị trí lưu và nhấn OK hoặc nhấn Cancel để hủy bỏ thao tác xuất tệp. Use case kết thúc

### Quản lý hợp đồng

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý hợp đồng cho thuê hoặc hợp đồng đặt cọc

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý hợp đồng

1. **Thêm mới hợp đồng** Khi người dùng nhấn nút *Thêm mới hợp đồng* hoặc tạo mới hợp đồng từ màn hình quản lý phòng trọhệ thống sẽ đưa ra một cửa sổ để người dùng có thể nhập thông tin của hợp đồng như loại hợp đồng (hợp đồng thuê, hợp đồng đặt cọc), thông tin bên A, bên B, các điều khoản trong hợp đồng. Sau khi hoàn thành xong thông tin người dùng nhấn nút *Lưu hợp đồng.* Nếu hợp lệ hệ thống sẽ lưu dữ liệu vào bảng *Contracts* đồng thời đưa ra thông báo cho người dùng về việc tạo hợp đồng thành công.
2. **Chỉnh sửa hợp đồng** Khi người dùng nhấn vào biểu tượng menu và chọn chỉnh sửa phương tiện, hệ thống sẽ hiện một cửa sổ để người dùng chỉnh sửa các thông tin hợp đồng, sau khi người dùng nhấn *Lưu hợp đồng*, dữ liệu sẽ được cập nhật trong bảng *Contracts* và đưa ra thông báo cập nhật thành công. Use case kết thúc
3. **Xóa hợp đồng** Khi người dùng nhấn vào biểu tượng xóa tại mỗi dòng thông tin. Hệ thống đưa ra cảnh báo về việc xóa thiết bị, nếu người dùng xác nhận xóa, dữ liệu sẽ được xóa khỏi bảng *Contracts* và các bảng liên quan, đồng thời đưa ra thông báo xóa thành công cho người dùng. Use case kết thúc
4. **Xuất tệp hợp đồng** Khi người dùng nhấn vào biểu tượng *In* tại mỗi dòng thông tin, Hệ thống sẽ cho phép tải xuống hợp đồng dạng file Word (.docx), người dùng chọn vị trí lưu dữ liệu và nhấn *OK* hợp đồng sẽ được lưu vào thiết bị của người dùng. Use case kết thúc
5. **Xuất tệp danh sách** Khi người dùng nhấn vào nút *Xuất tệp* hệ thống sẽ xử lý tạo ra file Excel người dùng chọn vị trí lưu rồi nhấn *OK* để tải file xuống. Use case kết thúc
6. **Đổi trạng thái hợp đồng** Khi đến thời hạn của hợp đồng hệ thống sẽ tự động chuyển đổi hợp đồng sang trạng thái *Đã hết hạn.* Use case kết thúc.

***Luồng rẽ nhánh***

1. Tại luồng cơ bản 1 và 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 3 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa phương tiện, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc
3. Tại luồng cơ bản 4,5 khi thực hiện thao tác xuất tệp hệ thống sẽ hiển thị vị trí lưu, người dùng chọn vị trí lưu và nhấn *OK* hoặc nhấn *Cancel* để hủy bỏ thao tác xuất tệp. Use case kết thúc
4. Tại luồng cơ bản 6 nếu người dùng bật tính năng này tại màn hình *Cài đặt* hệ thống sẽ tự cập nhật trạng thái hợp đồng, nếu người dùng không bật tính năng này trạng thái của hợp đồng sẽ không được cập nhật.

### Quản lý hóa đơn

**Mô tả vắn tắt**

Use case này cho phép người dùng quản lý các đợt thanh toán tiền phòng, thường là vào đầu tháng

**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang quản lý hóa đơn

1. **Thêm mới đợt thanh toán** Khi người dùng nhấn nút *Thêm mới đợt thanh toán* hệ thống sẽ xuất hiện cửa sổ cho phép người dùng điền các thông tin của đợt thanh toán như tên đợt thanh toán (ví dụ: Thanh toán tiền nhà tháng 12), ngày bắt đầu thu tiền, ngày kết thúc, và chọn các phòng trong đợt thanh toán. Khi người dùng nhấn nút *Thêm mới,* nếu dữ liệu hợp lệ hệ thống sẽ thêm dữ liệu vào bảng *StagePayments.* Use case kết thúc
2. **Chi tiết đợt thanh toán** Khi người dùng nhấn vào biểu tượng chi tiết tại mỗi dòng thông tin, hệ thống sẽ chuyển sang màn hình chi tiết đợt thanh toán, tại màn hình này người dùng có thể xem tổng quan thông tin về đợt thanh toán như Tổng tiền đã thu, số phòng đã thu, mức độ hoàn thành của đợt thanh toán, danh sách các phòng trong đợt thanh toán. Use case kết thúc.
3. **Nhập số liệu phòng** Khi người dùng nhấn vào biểu tượng *Nhập số liệu* tại mỗi dòng thông tin màn hình sẽ xuất hiện của sổ để người dùng nhập số liệu như điện, nước, chi phí dịch vụ và các chi phí phát sinh của phòng trong đợt thanh toán đó, người dùng có thể chọn *Tất toán hợp đồng* trong màn hình này. Sau khi nhập xong thông tin hệ thống sẽ tính tiền cần thu trong đợt thanh toán đó. Khi người dùng nhấn *Lưu hóa đơn* nếu hợp lệ dữ liệu sẽ được cập nhật trong bảng *Invoices,*  đồng thời chuyển trạng thái hóa đơn phòng sang *Đã nhập dữ liệu* và đưa ra thông báo cho người dùng về việc cập nhật dữ liệu thành công.
4. **Cập nhật trạng thái thanh toán** Khi người dùng nhấn vào biểu tượng *Đã thanh toán* tại mỗi dòng dữ liệu, hệ thống sẽ cập nhật trạng thái cho phòng thành *Đã thanh toán,* đồng thời cập nhật lại dữ liệu bảng *Invoices* và đưa ra thông báo cho người dùng về việc cập nhật trạng thái thanh toán thành công.
5. **Gửi thông báo thanh toán**  Người dùng có thể chọn các dòng dữ liệu ở trạng thái *Chưa thanh toán* và nhấn nút *Gửi thông báo* hệ thống sẽ gửi thông báo đến khách hàng thuê về việc thanh toán tiền phòng, kèm QR Code thanh toán. Use case kết thúc
6. **Xóa phòng khỏi đợt thanh toán** Khi người dùng ấn vào nút *Xóa khỏi đợt thanh toán.* Hệ thống sẽ xóa dữ liệu khỏi bảng *StageRooms, Invoices* đồng thời đưa ra thông báo cho người dùng về việc xóa phòng thành công.

***Luồng rẽ nhánh***

1. Tại luồng cơ bản 2 nếu các thông tin người sử dụng nhập vào không hợp lệ hệ thống sẽ đưa ra thông báo yêu cầu nhập lại, người dùng có thể nhập lại hoặc thoát hệ thống. Use case kết thúc
2. Tại luồng cơ bản 6 khi thực hiện thao tác xóa hệ thống sẽ hiện thông báo yêu cầu xác nhận xóa phương tiện, người dùng có thể nhấn Đồng ý để thực hiện xóa, hoặc Nhấn hủy bỏ để không thực hiện xóa. Use case kết thúc
3. Tại luồng cơ bản 4 khi người dùng thực hiện thao tác cập nhật trạng thái thanh toán hệ thống sẽ đưa ra xác nhận về việc cập nhật trạng thái thanh toán người dùng có thể nhấn *Xác nhận* để cập nhật trạng thái thanh toán hoặc *Hủy bỏ* . Use case kết thúc

### Báo cáo

**Mô tả vắn tắt**

Use case này cho phép người dùng xem tổng quan tình hình các khu trọ như doanh thu, số người ở, tình trạng các phòng

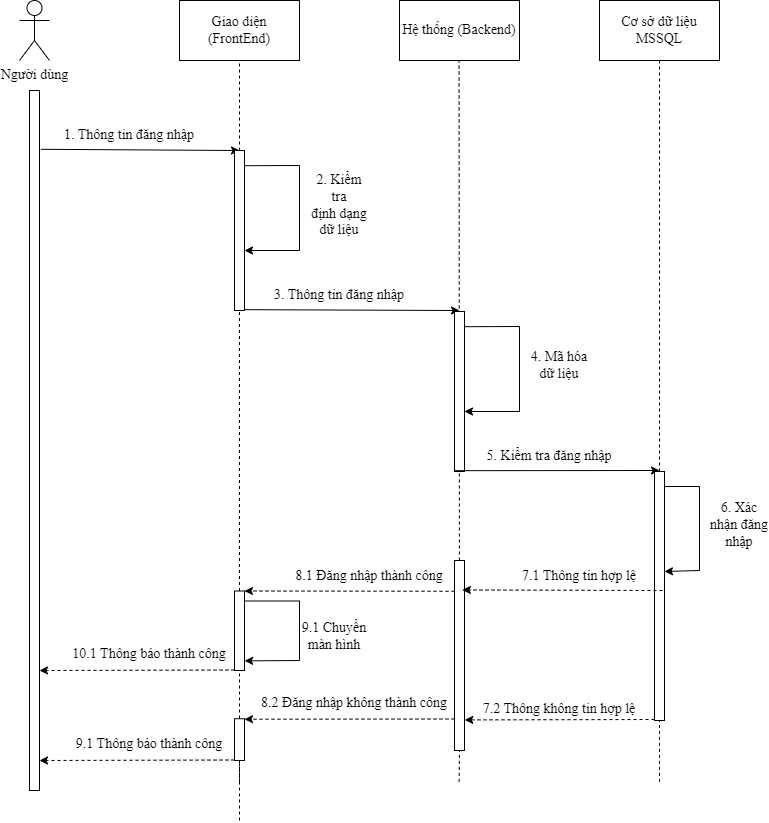
**Luồng sự kiện**

* ***Luồng cơ bản***

Use case này bắt đầu khi người sử dụng truy cập trang báo cáo

1. **Xem báo cáo doanh thu** Khi người dùng chọn *Báo cáo doanh thu* sau đó chọn khoảng thời gian cần xem báo cáo và nhấn nút *Xem báo cáo* hệ thống sẽ liệt kê toàn bộ các đợt thanh toán và tổng tiền thanh toán, nợ trong khoảng thời gian đó

## Biểu đồ trình tự - Đăng nhập



# XÂY DỰNG CƠ SỞ DỮ LIỆU

## Xây dựng quan hệ giữa các bảng

|  |  |  |
| --- | --- | --- |
| **#** | **Tên quan hệ** | **Mô tả** |
| 1 | FK\_AppContracts\_Rooms\_RoomId | Hợp đồng – phòng trọ |
| 2 | FK\_Customers\_Rooms\_RoomId | Khách hàng – phòng trọ |
| 3 | FK\_Vehicles\_Customers\_CustomerId | Phương tiện – khách hàng |
| 4 | FK\_FitmentInRooms\_Rooms\_RoomId | Thiết bị trong phòng - phòng |
| 5 | FK\_FitmentInRooms\_Fitments\_FitmentId | Thiết bị trong phòng – thiết bị |
| 6 | FK\_Rooms\_BoardingHouses\_BoardingHouseId | Khu trọ - phòng trọ |
| 7 | FK\_ServiceInBoardingHouses\_BoardingHouses\_BoardingHouseId | Dịch vụ - khu trọ |
| 8 | FK\_StageRooms\_Rooms\_RoomId | Đợt thanh toán – phòng trọ |
| 9 | FK\_StageRooms\_StagePayments\_StagePaymentId | Đợt thanh toán – phòng trong đợt |
| 10 | FK\_Invoices\_StageRooms\_StageRoomId | Hóa đơn – đợt thanh toán |
| 11 | FK\_RoomDepositeds\_Rooms\_RoomId | Phòng đặt cọc |
| 12 | FK\_BoardingHouses\_Cities\_CityId | Khu trọ - thành phố |
| 13 | FK\_ServiceInBoardingHouses\_Provides\_ProvideId | Dịch vụ - khu trọ |

## Danh sách các bảng

### dbo.Cities

* Lưu trữ danh sách các thành phố để chọn khu vực khi tạo khu trọ

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | Guid | Primary Key |
| 2 | Name | Nvarchar(500) | Not Null |
| 3 | IsDeleted | Bit | Not Null |
| 4 | CreatedAt | Datetime | Not null |
| 5 | UpdatedAt | Datetime | Not null |

### Dbo.AppContracts

* Lưu trữ hợp đồng khu trọ

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | Guid | Primary Key |
| 2 | Name | Nvarchar(500) | Not Null |
| 3 | ContractNo | Nvarchar(200) | Not null |
| 4 | CustomerId | Guid | Not null |
| 5 | CustomerName | Nvarchar(200) | Null |
| 6 | CustomerAddress | Nvarchar(500) | null |
| 7 | RoomId | Guid | Not null |
| 8 | CreatedDate | Datetime | Not null |
| 9 | ExpiredDate | DateTime | null |
| 10 | DepositedAmount | Decimal | null |
| 11 | Type | Int | Not null |
| 12 | ContractDuration | Int | Not null |
| 13 | IsDeleted | Bit | Not Null |
| 14 | CreatedAt | Datetime | Not null |
| 15 | UpdatedAt | Datetime | Not null |

### dbo.BoardingHouses

* Lưu trữ danh sách khu trọ

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | Guid | Primary Key |
| 2 | Name | Nvarchar(150) | Not Null |
| 3 | Address | Nvarhcar(250) | null |
| 4 | Description | Nvarchar(500) | null |
| 5 | Months | Int | null |
| 6 | StartDatePayment | Int | null |
| 7 | EndDatePayment | Int | null |
| 8 | CityId | Guid | null |
| 9 | IsNotLimitTime | Bit | null |
| 10 | IsSelfPayment | Bit | null |
| 11 | IsDeleted | Bit | Not Null |
| 12 | CreatedAt | Datetime | Not null |
| 13 | UpdatedAt | Datetime | Not null |

### dbo.ContractTerms

* Lưu trữ các điều khoản hợp đồng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | Guid | Primary Key |
| 2 | AppContractId | Guid | Not Null |
| 3 | Type | Int | Not Null |
| 4 | Content | Nvarchar(5000) | null |
| 5 | CreatedAt | Datetime | Not null |
| 6 | UpdatedAt | Datetime | Not null |
| 7 | IsDeleted | Bit | Not null |

### dbo.Customers

* Lưu trữ danh sách các thành phố để chọn khu vực khi tạo khu trọ

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | Name | nvarchar(255) | Not null |
| 5 | Phone | nvarchar(16) | Not null |
| 6 | BirthDay | datetime2(7) | Null |
| 7 | IdentificationCitizen | nvarchar(13) | Not null |
| 8 | IdFactory | nvarchar(255) | Not null |
| 9 | Address | nvarchar(255) | Not null |
| 10 | Email | nvarchar(50) | Not null |
| 11 | Career | int | Not null |
| 12 | Gender | int | Not null |
| 13 | AvatarPath | nvarchar(150) | Not null |
| 14 | FontIdentityPath | nvarchar(150) | Not null |
| 15 | BackIdentityPath | nvarchar(150) | Not null |
| 16 | RoomId | uniqueidentifier | Not null |
| 17 | Password | nvarchar(MAX) | Not null |
| 18 | IsDeleted | bit | Not null |

### dbo.FitmentInRooms

* Lưu trữ danh sách các thiết bị trong phòng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | RoomId | uniqueidentifier | Null |
| 5 | FitmentId | uniqueidentifier | Null |
| 6 | Quantity | int | Null |
| 7 | IsDeleted | bit | Not null |

### dbo.FitmentInRooms

* Lưu trữ danh sách các thiết bị

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | Name | nvarchar(150) | Not null |
| 5 | IsCanUse | bit | Null |
| 6 | RecoupPrice | decimal(18, 2) | Null |
| 7 | Description | nvarchar(255) | Not null |
| 8 | IsDeleted | bit | Null |

### dbo.Invoices

* Lưu trữ hóa đơn chi tiết các phòng theo các đợt thanh toán

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | StageRoomId | uniqueidentifier | Null |
| 5 | ProvideId | uniqueidentifier | Not null |
| 6 | LastValue | decimal(18, 2) | Not null |
| 7 | NewValue | decimal(18, 2) | Not null |
| 8 | Price | decimal(18, 2) | Null |
| 9 | Amount | decimal(18, 2) | Not null |

### dbo.Provides

* Lưu trữ các dịch vụ hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | Name | nvarchar(MAX) | Not null |
| 5 | Type | int | Null |
| 6 | DefaultPrice | decimal(18, 2) | Null |
| 7 | IsDeleted | bit | Not null |

### dbo.Rooms

* Lưu trữ danh sách các phòng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | BoardingHouseId | uniqueidentifier | Null |
| 5 | Name | nvarchar(255) | Not null |
| 6 | Price | decimal(18, 2) | Null |
| 7 | Floor | int | Not null |
| 8 | MaxHuman | int | Not null |
| 9 | Description | nvarchar(4000) | Not null |
| 10 | Status | int | Null |
| 11 | Location | nvarchar(63) | Not null |
| 12 | IsSelfContainer | bit | Not null |
| 13 | Count | int | Not null |
| 14 | Area | int | Null |
| 15 | IsDeleted | bit | Null |

### dbo.ServiceInBoardingHouses

* Lưu trữ danh sách các dịch vụ trong các khu trọ

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | BoardingHouseId | uniqueidentifier | Null |
| 5 | ProvideId | uniqueidentifier | Null |
| 6 | Price | decimal(18, 2) | Null |
| 7 | IsDeleted | bit | Not null |

### dbo.StagePayments

* Lưu trữ danh sách các đợt thanh toán

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | InvoiceNo | nvarchar(MAX) | Not null |
| 5 | Name | nvarchar(MAX) | Not null |
| 6 | StageDate | datetime2(7) | Null |
| 7 | EndDate | datetime2(7) | Null |
| 8 | TotalAmount | decimal(18, 2) | Null |
| 9 | AmountPaid | decimal(18, 2) | Null |
| 10 | RoomData | int | Null |
| 11 | TotalRooms | int | Null |
| 12 | RoomPaid | int | Null |
| 13 | IsComplete | bit | Null |
| 14 | IsDeleted | bit | Not null |

### dbo.StageRooms

* Lưu trữ các phòng trong đợt thanh toán

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | InvoiceNo | nvarchar(MAX) | Not null |
| 5 | Name | nvarchar(MAX) | Not null |
| 6 | StageDate | datetime2(7) | Null |
| 7 | EndDate | datetime2(7) | Null |
| 8 | TotalAmount | decimal(18, 2) | Null |
| 9 | AmountPaid | decimal(18, 2) | Null |
| 10 | RoomData | int | Null |
| 11 | TotalRooms | int | Null |
| 12 | RoomPaid | int | Null |
| 13 | IsComplete | bit | Null |
| 14 | IsDeleted | bit | Not null |

### dbo.SystemFiles

* Lưu trữ các tệp hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | MapId | uniqueidentifier | Null |
| 5 | FileType | int | Null |
| 6 | FileName | nvarchar(MAX) | Not null |
| 7 | Extension | nvarchar(MAX) | Not null |
| 8 | Path | nvarchar(MAX) | Not null |
| 9 | IsDeleted | bit | Not null |

### dbo.Systems

* Lưu trữ các cài đặt hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | GroupKey | nvarchar(MAX) | Not null |
| 5 | [Group] | nvarchar(MAX) | Not null |
| 6 | Name | nvarchar(MAX) | Not null |
| 7 | Value | nvarchar(MAX) | Not null |
| 8 | IsDeleted | bit | Not null |

### dbo.UserInfos

* Lưu trữ thông tin người dùng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | UserId | uniqueidentifier | Null |
| 5 | Name | nvarchar(50) | Not null |
| 6 | Phone | nvarchar(16) | Not null |
| 7 | DayOfBirth | datetime2(7) | Not null |
| 8 | City | nvarchar(MAX) | Not null |
| 9 | IdentityNumber | nvarchar(16) | Not null |
| 10 | IdentityProvider | nvarchar(200) | Not null |
| 11 | IdentityDate | nvarchar(150) | Not null |
| 12 | Address | nvarchar(255) | Not null |
| 13 | Email | nvarchar(255) | Not null |
| 14 | AccountBankNumber | nvarchar(50) | Not null |
| 15 | BankName | nvarchar(50) | Not null |
| 16 | IsDeleted | bit | Not null |

### dbo.Users

* Lưu trữ tài khoản người dùng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | Password | nvarchar(255) | Not null |
| 5 | role | int | Null |
| 6 | Name | nvarchar(255) | Not null |
| 7 | Gender | int | Null |
| 8 | Mail | nvarchar(255) | Not null |
| 9 | Phone | nvarchar(10) | Not null |
| 10 | CityId | uniqueidentifier | Not null |
| 11 | Address | nvarchar(255) | Not null |
| 12 | IsDeleted | bit | Not null |

### dbo.Vehicles

* Lưu trữ phương tiện người dùng

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cột | Kiểu dữ liệu | Ràng buộc |
| 1 | Id | uniqueidentifier | Null |
| 2 | CreatedAt | datetime2(7) | Null |
| 3 | UpdatedAt | datetime2(7) | Null |
| 4 | Type | int | Null |
| 5 | LicensePlate | nvarchar(255) | Not null |
| 6 | Color | nvarchar(255) | Not null |
| 7 | CustomerId | uniqueidentifier | Not null |
| 8 | IsDeleted | bit | Not null |

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Tác giả đã làm đồ án này như thế nào? kết quả ra sao? có giống như kết quả đã dự kiến không?

Phần này sinh viên cần bám vào mục tiêu và nội dung đã nêu ở PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP và phần MỞ ĐẦU, đối chiếu với kết quả đã làm để có kết luận phù hợp.

Còn vướng mắc ở chỗ nào? lý do vướng mắc, hướng khắc phục ... (dựa vào đây để các sinh viên khóa sau tham khảo và phát triển đồ án hoặc tránh lặp lại để đỡ mất thời gian)

Kết luận chung

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | A. Tuner, Introduction to Neogeography, Sebastopol, CA USA: O'Reilly Media, 2006. |
| [2] | Hoàng Anh Đức, Lê Văn Hưng, Thiết kế website, Hanoi: NXB Giáo dục Việt Nam, 2016. |

# PHỤ LỤC

Các mã nguồn, các bảng biểu lớn, các phụ lục cho vào mục này. Trong phần đồ án, phần PHỤ LỤC không được tính vào tổng số trang của đồ án tốt nghiệp.

Các mã nguồn sử dụng Style “Code”, ví dụ như sau:

**<?php**

**echo “Hello world!”;**

**?>**