**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU**

**BÙI THANH HẢI**

**KHÓA LUẬN TỐT NGHIỆP**

**PHÁT TRIỂN GIẢI PHÁP GIÁM SÁT**

**NHÀ THÔNG MINH THỜI GIAN THỰC DỰA TRÊN**

**NỀN TẢNG ĐIỆN TOÁN ĐÁM MÂY**

**DEVELOPMENT OF A SURVEILLANCE SYSTEM FOR SMART HOUSES INTEGRATED OVER CLOUD PLATFORM**

**KỸ SƯ/ CỬ NHÂN NGÀNH TRUYỀN THÔNG VÀ MẠNG MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2018**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU**

**BÙI THANH HẢI – 14520236**

**KHÓA LUẬN TỐT NGHIỆP**

**PHÁT TRIỂN GIẢI PHÁP GIÁM SÁT**

**NHÀ THÔNG MINH THỜI GIAN THỰC DỰA TRÊN**

**NỀN TẢNG ĐIỆN TOÁN ĐÁM MÂY**

**DEVELOPMENT OF A SURVEILLANCE SYSTEM FOR SMART HOUSES INTEGRATED OVER CLOUD PLATFORM**

**KỸ SƯ/ CỬ NHÂN NGÀNH TRUYỀN THÔNG VÀ MẠNG MÁY TÍNH**

**GIẢNG VIÊN HƯỚNG DẪN**

**PGS.TS LÊ TRUNG QUÂN**

**TP. HỒ CHÍ MINH, 2018**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

NHẬN XÉT CỦA CÁN BỘ HƯỚNG DẪN

NHẬN XÉT CỦA CÁN BỘ PHẢN BIỆN

LỜI CẢM ƠN

MỤC LỤC

[Chương 1. TỔNG QUAN 17](#_Toc527486961)

[1.1. Lí do chọn đề tài 17](#_Toc527486962)

[1.2. Mục tiêu 17](#_Toc527486963)

[1.3. Đối tượng và phạm vi nghiên cứu 17](#_Toc527486964)

[Chương 2. MỘT SỐ PLATFORM DÙNG ĐỂ PHÁT TRIỂN ỨNG DỤNG WEB/API 18](#_Toc527486965)

[2.1. ASP.NET 18](#_Toc527486966)

[2.1.1. Tổng quan 18](#_Toc527486967)

[2.1.2. ASP.NET Framework 18](#_Toc527486968)

[2.1.2.1. Tổng quan 18](#_Toc527486969)

[2.1.2.2. Lịch sử phát triển 19](#_Toc527486970)

[2.1.3. ASP.NET Core 20](#_Toc527486971)

[2.1.3.1. Tổng quan 20](#_Toc527486972)

[2.1.3.2. Lịch sử phát triển 20](#_Toc527486973)

[2.1.3.3. Những cải tiến của ASP.NET Core 21](#_Toc527486974)

[2.1.4. So sánh ASP.NET Framework và ASP.NET Core 21](#_Toc527486975)

[2.2. NodeJS 22](#_Toc527486976)

[2.2.1. Tổng quan 22](#_Toc527486977)

[2.2.2. Lịch sử phát triển của NodeJS 22](#_Toc527486978)

[2.2.3. Một số đặc điểm của NodeJS 23](#_Toc527486979)

[2.2.3.1. Không đồng bộ và phát sinh sự kiện 23](#_Toc527486980)

[2.2.3.2. Chạy nhanh 23](#_Toc527486981)

[2.2.3.3. Các tiến trình đơn giản nhưng hiệu năng cao 23](#_Toc527486982)

[2.3. So sánh ASP.NET Core và NodeJS 23](#_Toc527486983)

[Chương 3. XÂY DỰNG HỆ THỐNG GỬI NHẬN DỮ LIỆU THÔNG QUA API BẰNG ASP.NET CORE 24](#_Toc527486984)

[3.1. Tổng quan 24](#_Toc527486985)

[3.2. Mục tiêu 24](#_Toc527486986)

[3.3. Các bước thực hiện 24](#_Toc527486987)

[3.3.1. Bước 1: Tạo một project mới bằng Visual Studio 24](#_Toc527486988)

[3.3.2. Bước 2: Cấu trúc thư mục 26](#_Toc527486989)

[3.3.3. Bước 3: Install các package cần thiết 27](#_Toc527486990)

[3.3.4. Bước 4: Tạo Model để định dạng dữ liệu và các phương thức truyền nhận 28](#_Toc527486991)

[3.3.4.1. PinModel 28](#_Toc527486992)

[3.3.4.2. UserModel 32](#_Toc527486993)

[3.3.4.3. DHTModel 38](#_Toc527486994)

[3.3.4.4. KitKey 42](#_Toc527486995)

[3.3.4.5. UserRole 42](#_Toc527486996)

[3.3.4.6. Gender 42](#_Toc527486997)

[3.3.5. Bước 5: Tạo DBContext để tương tác với CSDL 43](#_Toc527486998)

[3.3.6. Bước 6: Chỉnh sửa file “Startup.cs” 43](#_Toc527486999)

[3.3.7. Bước 7: Tạo API 44](#_Toc527487000)

[3.3.7.1. PinController 44](#_Toc527487001)

[3.3.7.2. UserController 46](#_Toc527487002)

[3.3.7.3. DHTController 50](#_Toc527487003)

[3.3.8. Bước 8: Tạo dữ liệu mặc định 52](#_Toc527487004)

[3.3.9. Bước 9: Migration 54](#_Toc527487005)

[3.3.10. Bước 10: Chạy chương trình 55](#_Toc527487006)

[Chương 4. XÂY DỰNG HỆ THỐNG THU THẬP DỮ LIỆU 57](#_Toc527487007)

[Chương 5. XÂY DỰNG HỆ THỐNG GIÁM SÁT VÀ ĐIỀU KHIỂN 57](#_Toc527487008)

[5.1. Tổng quan 57](#_Toc527487009)

[5.2. Mục tiêu 57](#_Toc527487010)

[Chương 6. TRIỂN KHAI HỆ THỐNG LÊN OPENSTACK 57](#_Toc527487011)

[6.1. Tổng quan 57](#_Toc527487012)

[6.2. Mục tiêu 57](#_Toc527487013)

[Chương 7. KẾT LUẬN 58](#_Toc527487014)

[Chương 8. HƯỚNG PHÁT TRIỂN 58](#_Toc527487015)

DANH MỤC HÌNH VẼ

[Hình 2.1: Tạo một project (1) 24](#_Toc527486891)

[Hình 2.2: Tạo một project (2) 25](#_Toc527486892)

[Hình 2.3: Cấu trúc thư mục của Project 26](#_Toc527486893)

[Hình 2.4: Cài đặt các packages cần thiết 27](#_Toc527486894)

[Hình 2.5: Cấu trúc thư mục của PinModel 27](#_Toc527486895)

[Hình 2.6: Định nghĩa thuộc tính của PinModel 28](#_Toc527486896)

[Hình 2.7: Định nghĩa phương thức của PinModel 28](#_Toc527486897)

[Hình 2.8: Triển khai phương thức của PinModel (1) 29](#_Toc527486898)

[Hình 2.9: Triển khai phương thức của PinModel (2) 30](#_Toc527486899)

[Hình 2.10: Triển khai phương thức của PinModel (3) 31](#_Toc527486900)

[Hình 2.11: Cấu trúc thư mục của UserModel 32](#_Toc527486901)

[Hình 2.12: Định nghĩa thuộc tính của UserModel 32](#_Toc527486902)

[Hình 2.13: Định nghĩa phương thức của UserModel 33](#_Toc527486903)

[Hình 2.14: Triển khai phương thức của UserModel (1) 33](#_Toc527486904)

[Hình 2.15: Triển khai phương thức của UserModel (2) 34](#_Toc527486905)

[Hình 2.16: Triển khai phương thức của UserModel (3) 35](#_Toc527486906)

[Hình 2.17: Triển khai phương thức của UserModel (4) 35](#_Toc527486907)

[Hình 2.18: Triển khai phương thức của UserModel (5) 36](#_Toc527486908)

[Hình 2.19: Triển khai phương thức của UserModel (6) 36](#_Toc527486909)

[Hình 2.20: Triển khai phương thức của UserModel (7) 37](#_Toc527486910)

[Hình 2.21: Cấu trúc thư mục của DHTModel 37](#_Toc527486911)

[Hình 2.22: Định nghĩa thuộc tính của DHTModel 38](#_Toc527486912)

[Hình 2.23: Định nghĩa phương thức của DHTModel 38](#_Toc527486913)

[Hình 2.24: Triển khai phương thức của DHTModel (1) 39](#_Toc527486914)

[Hình 2.25: Triển khai phương thức của DHTModel (2) 40](#_Toc527486915)

[Hình 2.26: Triển khai phương thức của DHTModel (3) 40](#_Toc527486916)

[Hình 2.27: Nội dung file “KitKey.cs” 41](#_Toc527486917)

[Hình 2.28: Nội dung file “UserRole.cs” 41](#_Toc527486918)

[Hình 2.29: Nội dung file “Gender.cs” 41](#_Toc527486919)

[Hình 2.30: Nội dung file “ThesisAPIContext.cs” 42](#_Toc527486920)

[Hình 2.31: Thiết lập kết nối với SQLite 42](#_Toc527486921)

[Hình 2.32: Thiết lập Cors 42](#_Toc527486922)

[Hình 2.33: Tạo các Controller 43](#_Toc527486923)

[Hình 2.34: Nội dung file “PinController.cs” (1) 43](#_Toc527486924)

[Hình 2.35: Nội dung file “PinController.cs” (2) 44](#_Toc527486925)

[Hình 2.36: Nội dung file “PinController.cs” (3) 44](#_Toc527486926)

[Hình 2.37: Nội dung file “PinController.cs” (4) 45](#_Toc527486927)

[Hình 2.38: Nội dung file “UserController.cs” (1) 45](#_Toc527486928)

[Hình 2.39: Nội dung file “UserController.cs” (2) 46](#_Toc527486929)

[Hình 2.40: Nội dung file “UserController.cs” (3) 46](#_Toc527486930)

[Hình 2.41: Nội dung file “UserController.cs” (4) 47](#_Toc527486931)

[Hình 2.42: Nội dung file “UserController.cs” (5) 47](#_Toc527486932)

[Hình 2.43: Nội dung file “UserController.cs” (6) 48](#_Toc527486933)

[Hình 2.44: Nội dung file “UserController.cs” (7) 48](#_Toc527486934)

[Hình 2.45: Nội dung file “UserController.cs” (8) 49](#_Toc527486935)

[Hình 2.46: Nội dung file “DHTController.cs” (1) 49](#_Toc527486936)

[Hình 2.47: Nội dung file “DHTController.cs” (2) 50](#_Toc527486937)

[Hình 2.48: Nội dung file “DHTController.cs” (3) 50](#_Toc527486938)

[Hình 2.49: Nội dung file “DHTController.cs” (4) 51](#_Toc527486939)

[Hình 2.50: Tạo SeedData 51](#_Toc527486940)

[Hình 2.51: Nội dung file “SeedData.cs” (1) 52](#_Toc527486941)

[Hình 2.52: Nội dung file “SeedData.cs” (2) 52](#_Toc527486942)

[Hình 2.53: Khai báo SeedData 53](#_Toc527486943)

[Hình 2.54: Chạy Migration 54](#_Toc527486944)

[Hình 2.55: Kết quả chạy chương trình 54](#_Toc527486945)

DANH MỤC BẢNG

[Bảng 2.1: Lịch sử phát triển của ASP.NET Framework 20](#_Toc527487016)

[Bảng 2.2: Lịch sử phát triển của ASP.NET Core 21](#_Toc527487017)

[Bảng 2.3: So sánh ASP.NET Framework và ASP.NET Core 22](#_Toc527487018)

[Bảng 2.4: Lịch sử phát triển của NodeJS 23](#_Toc527487019)

[Bảng 2.5: So sánh ASP.NET Core và NodeJS 24](#_Toc527487020)

DANH MỤC TỪ VIẾT TẮT

IoT: Internet of Things

API: Application Programming Interface

CSDL: Cơ sở dữ liệu

CLR: Common Language Runtime

IDE: Integrated Developement Environment

TÓM TẮT KHÓA LUẬN

MỞ ĐẦU

Internet of Thing là một hệ thống các thiết bị đồ dùng được kết nối với nhau thông qua mạng Internet.

# TỔNG QUAN

## Lí do chọn đề tài

## Mục tiêu

## Đối tượng và phạm vi nghiên cứu

Nội dung …………………

# MỘT SỐ PLATFORM DÙNG ĐỂ PHÁT TRIỂN ỨNG DỤNG WEB/API

## ASP.NET

### Tổng quan

ASP.NET là nền tảng phát triển web (web application framework), cung cấp một mô hình lập trình, cơ sở hạ tầng phần mềm toàn diện và các dịch vụ cần thiết để xây dựng các ứng dụng web động mạnh mẽ cho máy tính cũng như trên các thiết bị di động.

ASP.NET là một phần của nền tảng Microsoft.NET. Ứng dụng ASP.NET được biên dịch mã, được viết bằng cách sử dụng mở rộng và tái sử dụng các thành phần hoặc đối tượng trong nền tảng NET. Các mã này được sử dụng cho toàn bộ hệ thống phân cấp của các class trong .NET.

Các ứng dụng ASP.NET có thể được viết bằng bất kỳ ngôn ngữ nào sao đây:

* C#
* Visual Basic.Net
* Jscript
* J#

ASP.NET được phát hành đầu tiên vào 12/2/2012 với tên gọi ASP.NET Framework v1.0 cùng với Visual Studio.NET 2002.

### ASP.NET Framework

#### Tổng quan

.NET Framework là một nền tảng lập trình và cũng là một nền tảng thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows được phát triển bởi Microsoft. Các chương trình được viết trên nền.NET Framework sẽ được triển khai trong môi trường phần mềm (ngược lại với môi trường phần cứng) được biết đến với tên Common Language Runtime (CLR). Môi trường phần mềm này là một máy ảo trong đó cung cấp các dịch vụ như an ninh phần mềm (security), quản lý bộ nhớ (memory management), và các xử lý lỗi ngoại lệ (exception handling).

.NET framework bao gồm tập các thư viện lập trình lớn, và những thư viện này hỗ trợ việc xây dựng các chương trình phần mềm như lập trình giao diện; truy cập, kết nối cơ sở dữ liệu; ứng dụng web; các giải thuật, cấu trúc dữ liệu; giao tiếp mạng... CLR cùng với bộ thư viện này là 2 thành phần chính của.NET framework.

.NET framework đơn giản hóa việc viết ứng dụng bằng cách cung cấp nhiều thành phần được thiết kế sẵn, người lập trình chỉ cần học cách sử dụng và tùy theo sự sáng tạo mà gắn kết các thành phần đó lại với nhau. Nhiều công cụ được tạo ra để hỗ trợ xây dựng ứng dụng.NET, và IDE (Integrated Developement Environment) được phát triển và hỗ trợ bởi chính Microsoft là Visual Studio.

#### Lịch sử phát triển

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phiên bản** | **Số hiệu phiên bản** | **Ngày phát hành** | **Visual Studio** | **Được phát hành kèm theo** |
| 1.0 | 1.0.3705.0 | 13 tháng 2 năm 2002 | [Visual Studio](https://vi.wikipedia.org/wiki/Visual_Studio).NET | Windows XP Tablet and Media Center Editions |
| 1.1 | 1.1.4322.573 | 24 tháng 4 năm 2003 | Visual Studio.NET 2003 | Windows Server 2003 |
| 2.0 | 2.0.50727.42 | 7 tháng 11 năm 2005 | Visual Studio 2005 | Windows Server 2003 R2 |
| 3.0 | 3.0.4506.30 | 6 tháng 11 năm 2006 |  | Windows Vista, Windows Server 2008 |
| 3.5 | 3.5.21022.8 | 19 tháng 11 năm 2007 | Visual Studio 2008 | Windows 7, Windows Server 2008 R2 |
| 4.0 | 4.0.30319.1 | 12 tháng 4 năm 2010 | Visual Studio 2010 |  |
| 4.5 | 4.5.50709 | 15 tháng 8 năm 2012 | Visual Studio 2012 | Windows 8, Windows Server 2012 |
| 4.6 |  | 20 tháng 7 năm 2015 | Visual Studio 2015 | Windows 10 |

Bảng 2.1: Lịch sử phát triển của ASP.NET Framework

### ASP.NET Core

#### Tổng quan

ASP.NET Core là một OpenSource mới và framework đa nền tảng (cross-platform) cho việc xây dựng những ứng dụng hiện tại dựa trên kết nối điện toán đám mây, giống như Web apps, IoT và Backend cho mobile. Ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework.

ASP.NET Core được thiết kế để cung cấp và tối ưu Development framework cho những ứng dụng được triển khai trên đám mây (cloud) hoặc chạy on-promise. Điểm đặc biệt của ASP.NET Core là ta có thể phát triển và chạy những ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux . Điều mà ASP.NET Framework không làm được.

#### Lịch sử phát triển

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Release Date** | **Support Ended** | **Development Tool** |
| 1.0 | 2016-06-27 | 2019-06-27 | Visual Studio 2015, 2017 |
| 1.1 | 2016-11-18 | 2019-06-27 | Visual Studio 2015, 2017 |
| 2.0 | 2017-08-14 | 2018-10-01 | Visual Studio 2017 |
| 2.1 | 2018-05-30 |  | Visual Studio 2017 |

Bảng 2.2: Lịch sử phát triển của ASP.NET Core

#### Những cải tiến của ASP.NET Core

Một số cải tiến của ASP.NET Core bao gồm:

* Web UI và Web API được hợp nhất.
* Tích hợp các frameworks hiện đại cho khách hàng và nhà phát triển.
* A cloud-ready environment-based configuration system.
* Built-in dependency injection.
* New light-weight and modular HTTP request pipeline.
* Khả năng lưu trữ trên IIS hoặc tự chủ trong tiến trình riêng.
* Built on .NET Core, which supports true side-by-side app versioning.
* Ships entirely as NuGet packages.
* Công cụ vừa mới và đơn giản để phát triển các web hiện đại.
* Xây dựng và chạy nền tảng ứng dụng ASP.NET không chỉ trên Windows mà còn trên MAC và Linux.

### So sánh ASP.NET Framework và ASP.NET Core

|  |  |
| --- | --- |
| **ASP.NET Framework** | **ASP.NET Core** |
| Chạy trên Window | Đa nền tảng : Windows , Mac , Linux |
| Mã nguồn đóng | Mã nguồn mở |
| Chạy trên phiên bản đầy đủ của ASP.Net Framework | Có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework |
| Nặng nhưng đầy đủ | Gọn nhẹ tích hợp các Framwork khác nhau lại trong một bản duy nhất là ASP.NET Core |
| Build được thực hiện trên giao diện Visual Studio | Build có thể thực hiện bằng Command line hoặc giao diện |
| Hỗ trợ đầy đủ | Không hỗ trợ Web Form |

Bảng 2.3: So sánh ASP.NET Framework và ASP.NET Core

## NodeJS

### Tổng quan

Node js được viết bằng ngôn ngữ JavaScript, nó là một trình biên đóng gói của Google’s V8 JavaScript engine, libuv platform abstraction layer, và một thư viện lõi được viết bằng JavaScript. Mục tiêu của NodeJS là làm cho web có khả năng push như trong một số ứng dụng gmail. Node js cung cấp công cụ giúp lập trình viên có thể làm việc trong non-blocking, mô hình I/O. Sau hơn 20 năm nghiên cứu, xây dựng và phát triển, nhóm kĩ sư đã cho ra đời sản phẩm ứng dụng web NodeJS chạy thời gian thực và kết nối 2 chiều client và server, cho phép trao đổi dữ liệu một cách tự do.

### Lịch sử phát triển của NodeJS

|  |  |  |
| --- | --- | --- |
| **Version** | **Name** | **Release Date** |
| 0.10.x |  | 2013-03-11 |
| 0.12.x |  | 2015-02-06 |
| 4.x | Argon | 2015-08-09 |
| 5.x |  | 2015-10-29 |
| 6.x | Boron | 2016-04-26 |
| 7.x |  | 2016-10-25 |
| 8.x | Carbon | 2017-05-30 |
| 9.x |  | 2017-10-01 |
| 10.x | Dubnium | 2018-04-24 |
| 11.x |  | 2018-10-23 |

Bảng 2.4: Lịch sử phát triển của NodeJS

### Một số đặc điểm của NodeJS

#### Không đồng bộ và phát sinh sự kiện

Tất các các APIs của thư viện Node.js đều không đồng bộ, nghĩa là không blocking (khóa). Nó rất cần thiết vì NodeJS không bao giờ đợi một API trả về dữ liệu. Server sẽ chuyển sang một API khác sau khi gọi nó và cơ chế thông báo về sự kiện của NodeJS giúp Server nhận được phản hồi từ các API đã được gọi trước đó.

#### Chạy nhanh

Dựa trên V8 Javascript Engine của Google Chrome, thư viện NodeJS rất nhanh trong các quá trình thực hiện code.

#### Các tiến trình đơn giản nhưng hiệu năng cao

NodeJS sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa (non-blocking) và tạo cho Server hiệu quả cao so với các cách truyền thống là tạo ra một số lượng luồng hữu hạn để quản lý request. NodeJS sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.

## So sánh ASP.NET Core và NodeJS

|  |  |
| --- | --- |
| **ASP.NET Core** | **NodeJS** |
| Đa nền tảng | Đa nền tảng |
| Chủ yếu phát triển Server Side | Phát triển cả Server Side và Client Side |
| Ngôn ngữ chính là C# | Ngôn ngữ chính là JavaScript |
| Hosting hỗ trợ hiếm | Hosting hỗ trợ phong phú |
| Chạy đồng bộ hoặc bất đồng bộ | Chạy bất đồng bộ |

Bảng 2.5: So sánh ASP.NET Core và NodeJS

# XÂY DỰNG HỆ THỐNG GỬI NHẬN DỮ LIỆU THÔNG QUA API BẰNG ASP.NET CORE

## Tổng quan

Trong phạm vi nghiên cứu này sẽ sử dụng ASP.NET Core 2.0 để xây dựng một số API nhằm cung cấp môi trường truyền nhận dữ liệu cho các thiết bị Arduino, Raspberry Pi và ứng dụng Web.

## Mục tiêu

Sử dụng ASP.NET Core để tạo ra một số API cơ bản nhằm hỗ trợ giám sát, điều khiển và thu thập dữ liệu.

Xây dựng được Backend có khả năng tùy chỉnh, thay đổi theo mong muốn, dễ dàng triển khai.

Xây dựng một số API cung cấp các chức năng sau:

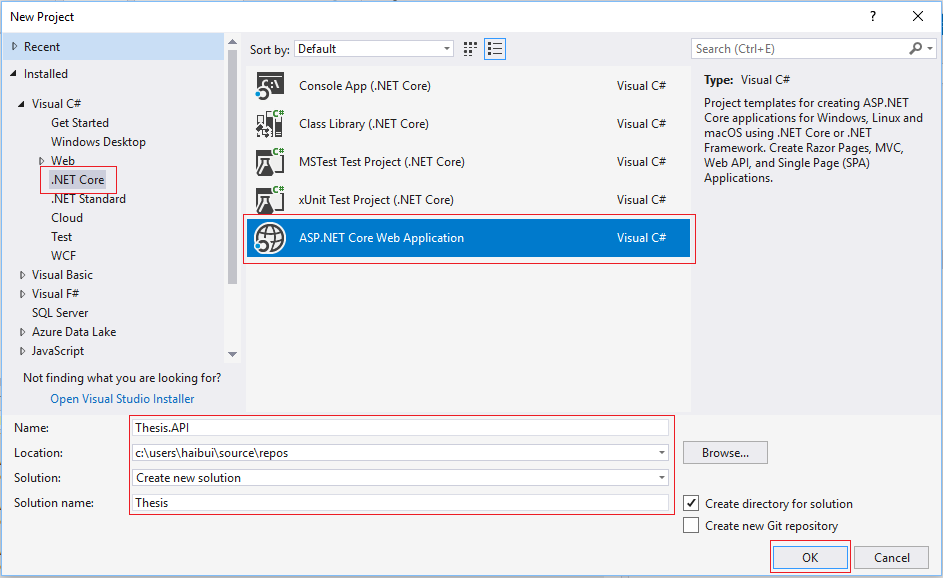
* Đăng nhập, đăng kí.
* Quản lí người dùng : thêm, xóa, cập nhật thông tin người dùng.
* Điều khiển tín hiệu: đèn, cửa ra vào
* Thu thập dữ liệu: nhiệt độ, độ ẩm, ánh sáng, chuyển động, cảnh báo lửa, trạng thái hiện tại của các đèn trong nhà…

## Các bước thực hiện

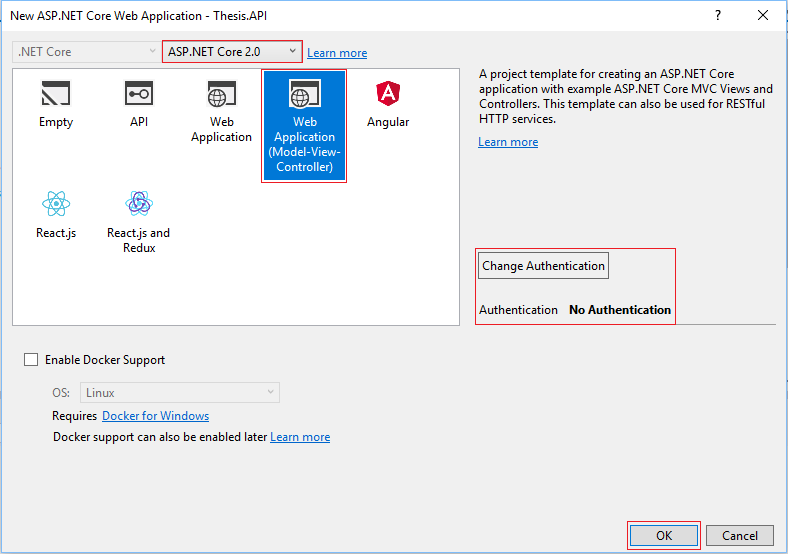
### Bước 1: Tạo một project mới bằng Visual Studio

Mở Visual Studio phiên bản nên lớn hơn 2017. Bấm tổ hợp phím “ctrl + shift + n” để tạo một project mới như hình bên dưới.

Sử dụng ASP.NET core 2.0[[1]](#footnote-1) để xây dựng server.



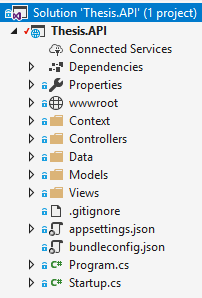
Hình 2.1: Tạo một project (1)



Hình 2.2: Tạo một project (2)

### Bước 2: Cấu trúc thư mục

Tạo cấu trúc thư mục như hình bên dưới.



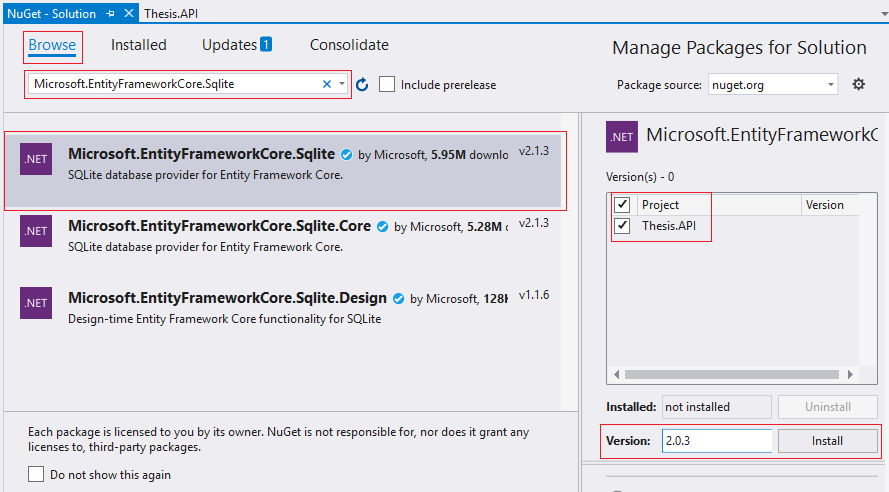
Hình 2.3: Cấu trúc thư mục của Project

### Bước 3: Install các package cần thiết

Vào Tool -> NuGet Package Manager -> Manage Nuget Packages for Solution…

Cần cài một số packages sau:

* Microsoft.EntityFrameworkCore.Tools
* Microsoft.EntityFrameworkCore.Sqlite
* Microsoft.EntityFrameworkCore.SqlServer
* System.Linq
* Microsoft.AspNet.WebApi.Cors



Hình 2.4: Cài đặt các packages cần thiết

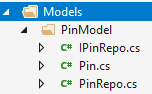
Với ASP.NET core 2.0 nên chọn package version 2.0.x và cài tương tự với các packages còn lại.

### Bước 4: Tạo Model để định dạng dữ liệu và các phương thức truyền nhận

#### PinModel

Mục đích của PinModel nhằm lưu trữ trạng thái của các cổng GPIO của Arduino. Chỉ có hai trạng thái duy nhất đó là 0 và 1.

Tạo cấu trúc thư mục cho PinModel như hình bên dưới.

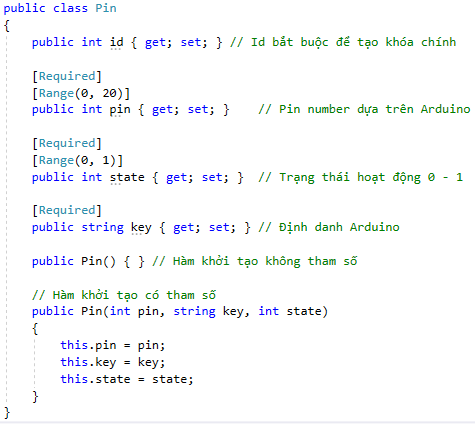


Hình 2.5: Cấu trúc thư mục của PinModel

Trong đó:

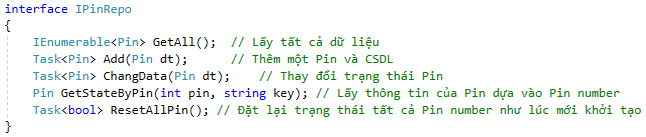
* IPinRepo.cs: Dùng để định nghĩa các phương thức sẽ được sử dụng.
* Pin.cs: Dùng để định nghĩa các thuộc tính cần thiết.
* PinRepo: Dùng để triển khai các phương thức đã được định nghĩa.

Định nghĩa các thuộc tính cần thiết như hình dưới.



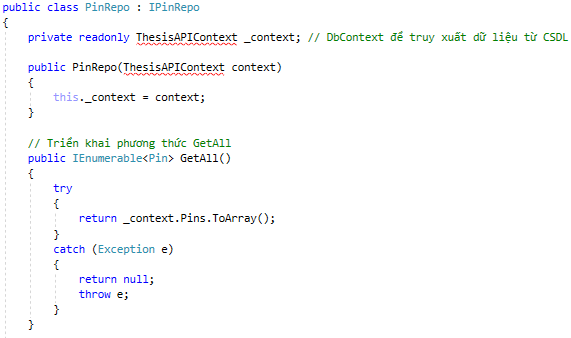
Hình 2.6: Định nghĩa thuộc tính của PinModel

Định nghĩa các phương thức dùng để truyền nhận dữ liệu.



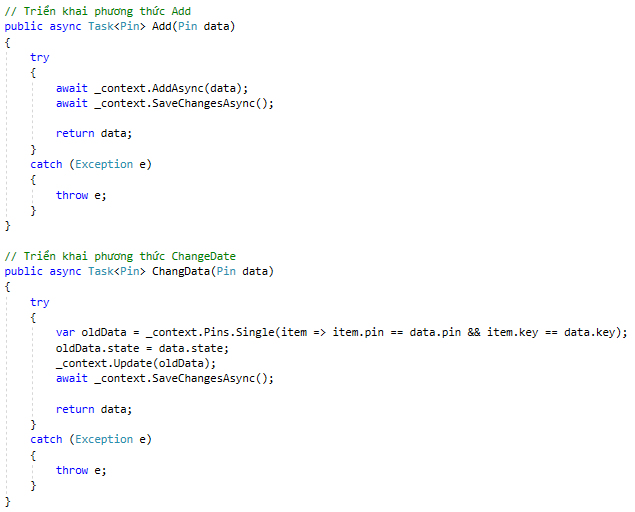
Hình 2.7: Định nghĩa phương thức của PinModel

Sau khi đã khai báo các thuộc tính và phương thức sẽ tiến hành triển khai các phương thức đã định nghĩa ở trên.

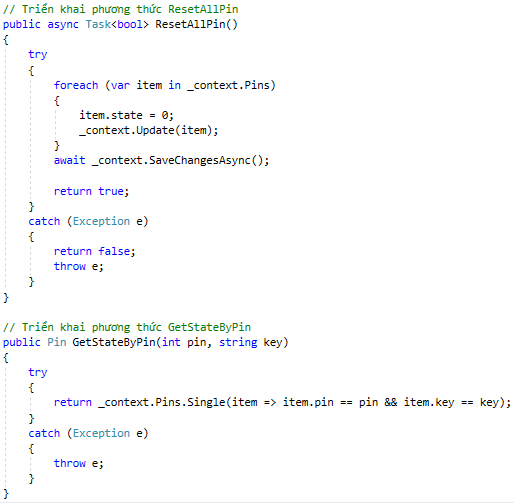


Hình 2.8: Triển khai phương thức của PinModel (1)

Ở bên trên báo lỗi không tìm thấy “ThesisAPIContext”, phần này sẽ khởi tạo bên dưới sau khi đã định nghĩa tất cả Model.



Hình 2.9: Triển khai phương thức của PinModel (2)



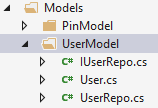
Hình 2.10: Triển khai phương thức của PinModel (3)

#### UserModel

Mục đích của UserModel nhằm lưu thông tin của người dùng bao gồm các thông tin cơ bản như tên người dùng và mã số thẻ RFID tương ứng.

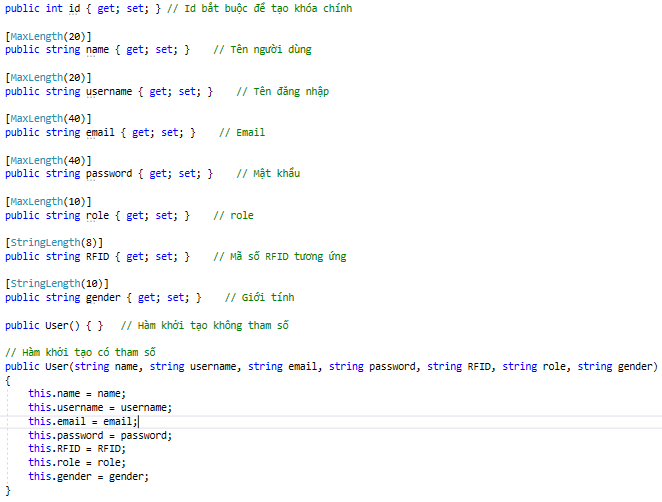
Các bước thực hiện tương tự PinModel nên sẽ không giải thích lại.

Tạo cấu trúc thư mục.



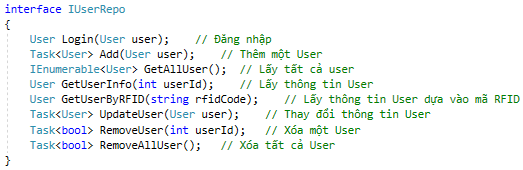
Hình 2.11: Cấu trúc thư mục của UserModel

Định nghĩa các thuộc tính.



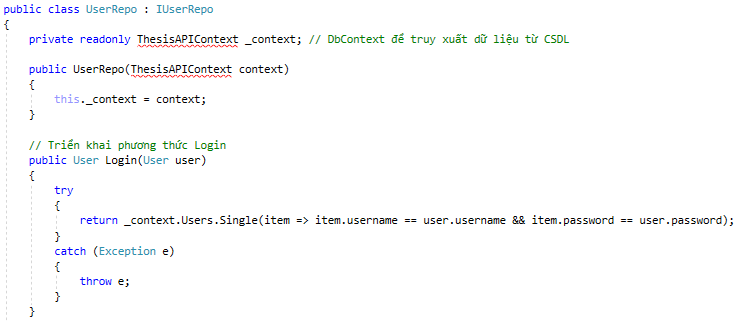
Hình 2.12: Định nghĩa thuộc tính của UserModel

Định nghĩa các phương thức.

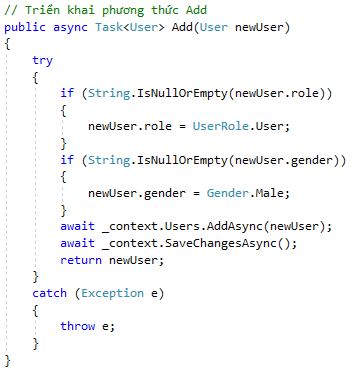


Hình 2.13: Định nghĩa phương thức của UserModel

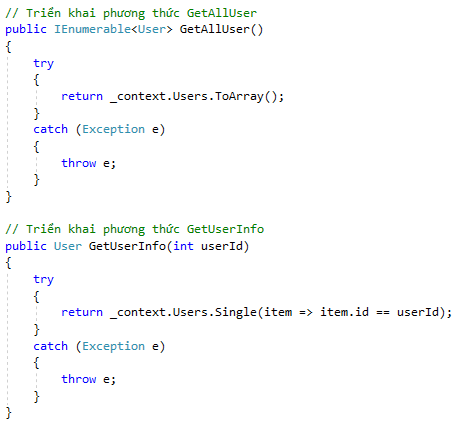
Triển khai các phương thức.



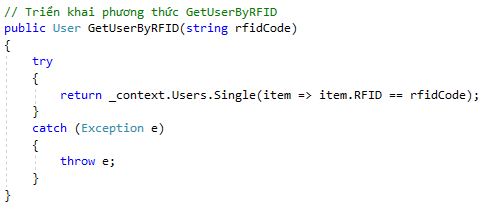
Hình 2.14: Triển khai phương thức của UserModel (1)



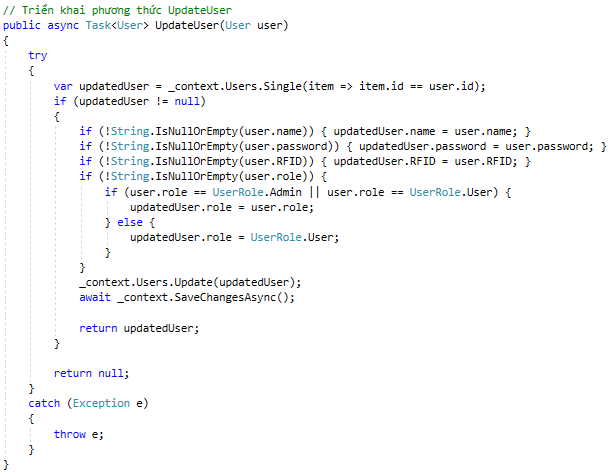
Hình 2.15: Triển khai phương thức của UserModel (2)



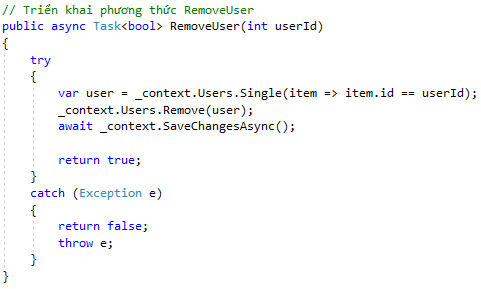
Hình 2.16: Triển khai phương thức của UserModel (3)



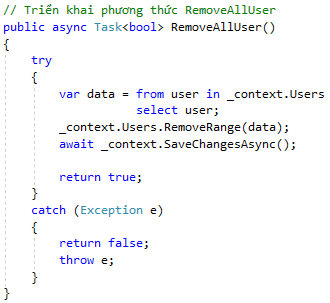
Hình 2.17: Triển khai phương thức của UserModel (4)



Hình 2.18: Triển khai phương thức của UserModel (5)



Hình 2.19: Triển khai phương thức của UserModel (6)



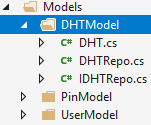
Hình 2.20: Triển khai phương thức của UserModel (7)

#### DHTModel

Mục đích của DHTModel nhằm lưu trữ thông tin về nhiệt độ và độ ẩm thu thập được từ cảm biến DHT11[[2]](#footnote-2).

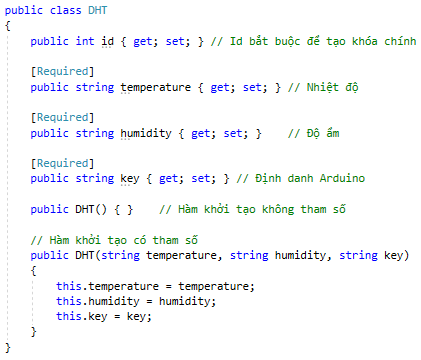
Các bước thực hiện tương tự PinModel nên sẽ không giải thích lại.

Tạo cấu trúc thư mục.



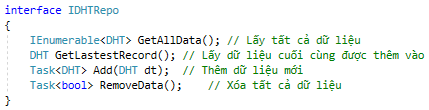
Hình 2.21: Cấu trúc thư mục của DHTModel

Định nghĩa các thuộc tính.



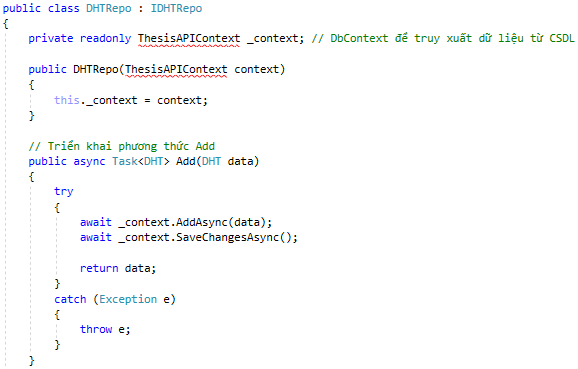
Hình 2.22: Định nghĩa thuộc tính của DHTModel

Định nghĩa các phương thức.

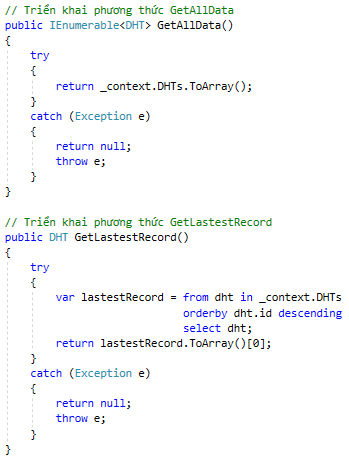


Hình 2.23: Định nghĩa phương thức của DHTModel

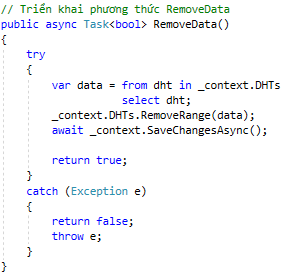
Triển khai các phương thức.



Hình 2.24: Triển khai phương thức của DHTModel (1)



Hình 2.25: Triển khai phương thức của DHTModel (2)

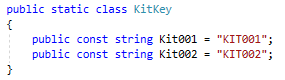


Hình 2.26: Triển khai phương thức của DHTModel (3)

#### KitKey

Định nghĩa “key” sẽ được dùng để định danh cho từng thiết bị Arduino.

Tạo một file “KitKey.cs” bên trong thư mục Models.



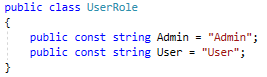
Hình 2.27: Nội dung file “KitKey.cs”

Ở đây khai báo theo số lượng thiết bị hiện tại. Nếu nhiều hơn hai thiết bị thì có thể khai báo thêm.

#### UserRole

Định nghĩa vai trò của từng User.

Tạo một file “UserRole.cs” bên trong thư mục Models.

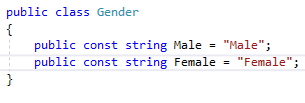


Hình 2.28: Nội dung file “UserRole.cs”

#### Gender

Định nghĩa giới tính cho User

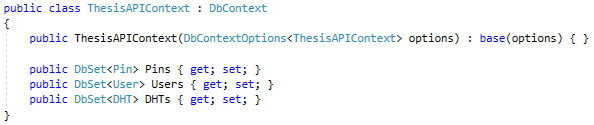
Tạo một file “Gender.cs” bên trong thư mục Models.



Hình 2.29: Nội dung file “Gender.cs”

### Bước 5: Tạo DBContext để tương tác với CSDL

Trong thư mục context, tạo một file “ThesisAPIContext.cs” có nội dung như sau.

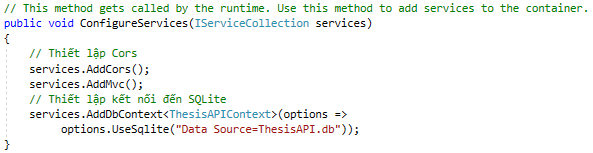


Hình 2.30: Nội dung file “ThesisAPIContext.cs”

Sau khi tạo file “ThesisAPIContext.cs”, tiến hành cập nhật cho các file “PinRepo.cs”, “UserRepo.cs”, “DHTRepo.cs”.

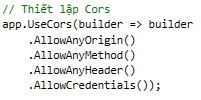
### Bước 6: Chỉnh sửa file “Startup.cs”

Trong file “Startup.cs”, thiết lập kết nối với SQLite[[3]](#footnote-3).



Hình 2.31: Thiết lập kết nối với SQLite

Thêm đoạn code bên dưới vào hàm “Configure”, trước “app.UseMvc”.



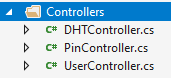
Hình 2.32: Thiết lập Cors

### Bước 7: Tạo API

Tiến hành tạo các Controller để có thể gửi nhận dữ liệu thông qua các API mà Controller cung cấp.

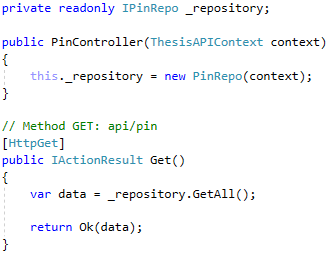
Để tạo Controller, chuột phải vào thư mục Controllers -> Add -> Controller… -> API Controller – Empty -> Add, đặt tên cho Controller và chọn Add.

Tạo Controller tương ứng với các Model như hình.

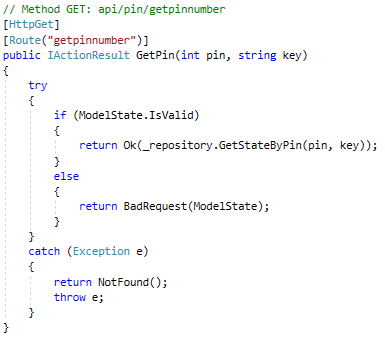


Hình 2.33: Tạo các Controller

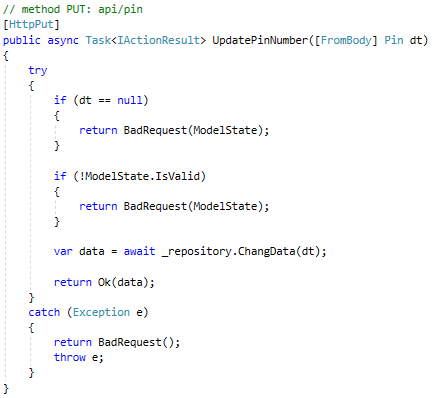
#### PinController



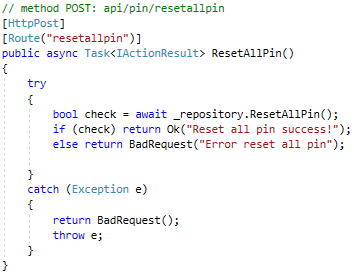
Hình 2.34: Nội dung file “PinController.cs” (1)



Hình 2.35: Nội dung file “PinController.cs” (2)



Hình 2.36: Nội dung file “PinController.cs” (3)

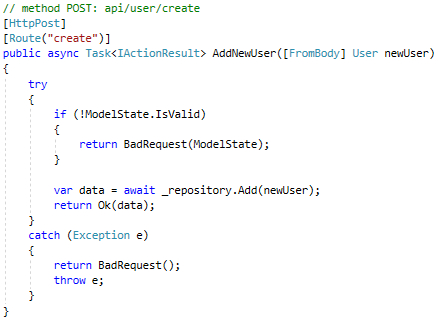


Hình 2.37: Nội dung file “PinController.cs” (4)

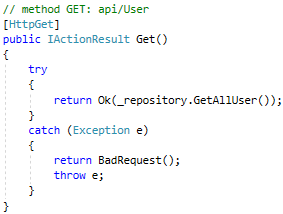
#### UserController



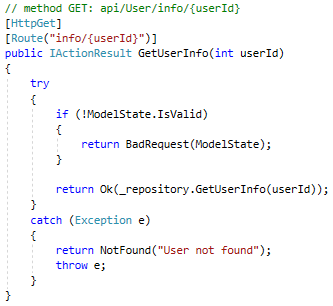
Hình 2.38: Nội dung file “UserController.cs” (1)



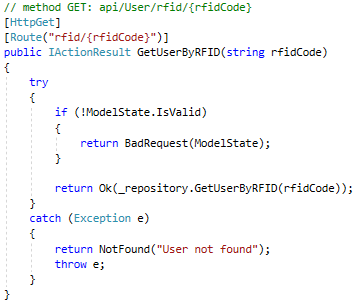
Hình 2.39: Nội dung file “UserController.cs” (2)



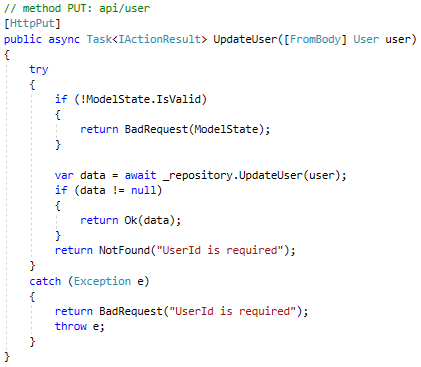
Hình 2.40: Nội dung file “UserController.cs” (3)



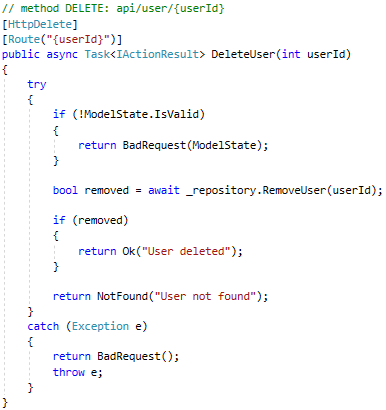
Hình 2.41: Nội dung file “UserController.cs” (4)



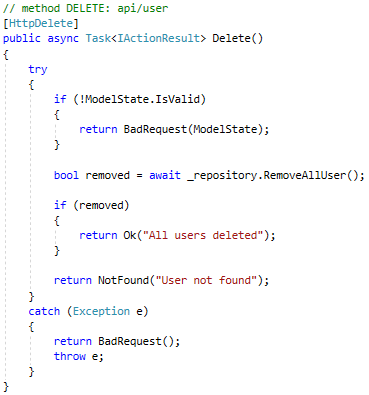
Hình 2.42: Nội dung file “UserController.cs” (5)



Hình 2.43: Nội dung file “UserController.cs” (6)

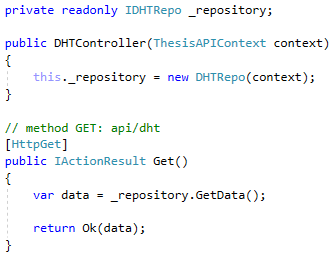


Hình 2.44: Nội dung file “UserController.cs” (7)

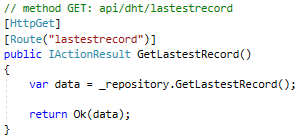


Hình 2.45: Nội dung file “UserController.cs” (8)

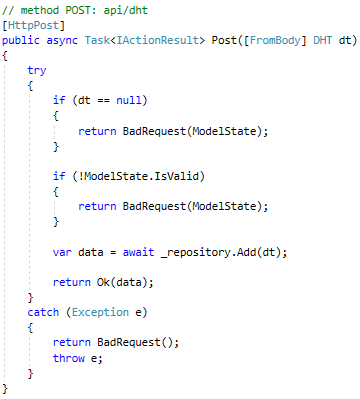
#### DHTController



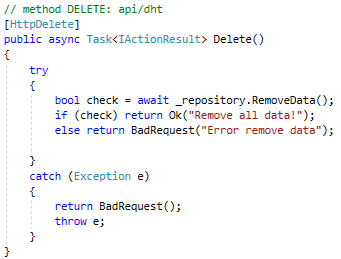
Hình 2.46: Nội dung file “DHTController.cs” (1)



Hình 2.47: Nội dung file “DHTController.cs” (2)



Hình 2.48: Nội dung file “DHTController.cs” (3)



Hình 2.49: Nội dung file “DHTController.cs” (4)

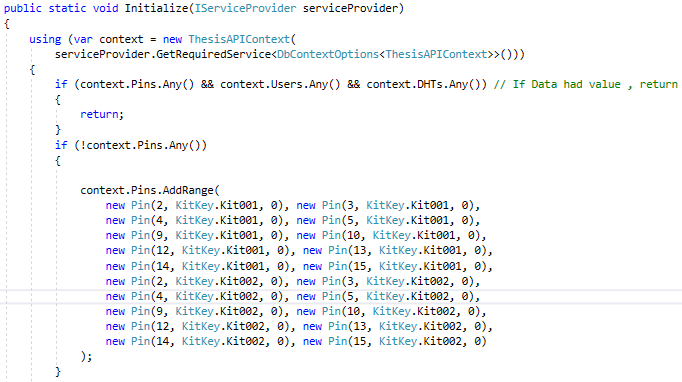
### Bước 8: Tạo dữ liệu mặc định

Trong thư mục Data, tạo một class mới tên SeedData.cs

Mục đích là tạo ra những dữ liệu mặc định trong CSDL khi chạy chương trình lần đầu tiên, hoặc sau khi xóa CSDL và tạo mới CSDL.



Hình 2.50: Tạo SeedData

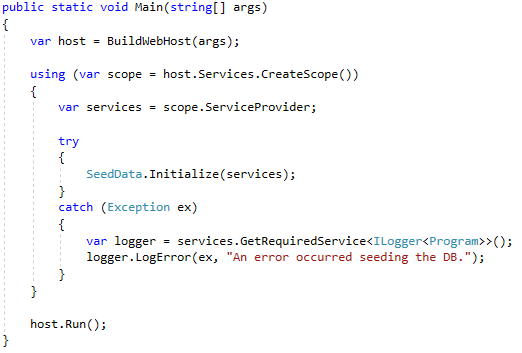


Hình 2.51: Nội dung file “SeedData.cs” (1)



Hình 2.52: Nội dung file “SeedData.cs” (2)

Khai báo dữ liệu vừa tạo vào file Program.cs -> hàm Main()



Hình 2.53: Khai báo SeedData

### Bước 9: Migration

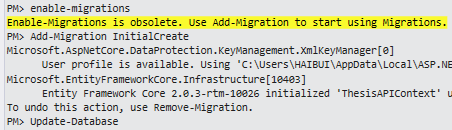
Chuột phải vào Solution -> Properties -> Common Properties -> Startup Project -> Single startup project. Chuyển “docker-compose” thành “Thesis.API”.

Chọn Tool -> Nuget Package Manager -> Package Manager Console

Chạy từng lệnh sau:

* enable-migrations
* Add-Migration InitialCreate
* Update-Database

Sau khi chạy xong ta có kết quả như sau.



Hình 2.54: Chạy Migration

Trong project sẽ thấy một thư mục Migrations được tự động sinh ra. Thư mục này sẽ tạo bảng tự động trong CSDL khi chạy chương trình. Đồng thời xuất hiện file “ThesisAPI.db”. Đây là file SQLite, tự động sinh ra sau khi chạy dòng lệnh cuối cùng.

### Bước 10: Chạy chương trình

Có thể chạy chương trình bằng nhiều cách như : IISServer, console, dotnet run

Sau khi chạy chương trình, kiểm tra đường dẫn [http://localhost:<port>/api/pin](http://localhost:%3cport%3e/api/pin) được kết quả như sau.



Hình 2.55: Kết quả chạy chương trình

Kiểm tra tương tự với các API còn lại.

# XÂY DỰNG HỆ THỐNG THU THẬP DỮ LIỆU

# XÂY DỰNG HỆ THỐNG GIÁM SÁT VÀ ĐIỀU KHIỂN

## Tổng quan

Hệ thống giám sát và điều khiển được xây dựng trên nền Web. Sử dụng thư viện ReactJS làm khuôn cho ứng dụng. Redux để hỗ trợ quản lí State của React tốt hơn. Ngoài ra còn có một số package hỗ trợ cần thiết khác.

## Mục tiêu

Xây dựng một ứng dụng Web có khả năng cho phép:

* Theo dõi dữ liệu thu thập được từ các cảm biến trong nhà.
* Điều khiển được một số thiết bị trong nhà.
* Quản lí người trong nhà.

# TRIỂN KHAI HỆ THỐNG LÊN OPENSTACK

## Tổng quan

Trong nghiên cứu này sẽ sử dụng OpenStack - Mitaka của trường Đai học Công nghệ thông tin để triển khai Server.

Sử dụng Docker để triển khai Backend API và ứng dụng Web lên OpenStack một cách dễ dàng.

## Mục tiêu

Triển khai được hệ thống lên OpenStack – Mitaka bằng Docker.

Có thể truy cập hệ thống bất cứ lúc nào trong phạm vi trường Đại học Công nghệ thông tin.

Tạo nền tảng cho việc triển khai trên các hosting khác sau này.

# KẾT LUẬN

# HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE

1. ASP.NET core 2.0: <https://docs.microsoft.com/en-us/aspnet/core/getting-started/?view=aspnetcore-2.0&tabs=windows> [↑](#footnote-ref-1)
2. DHT11: <https://vngiotlab.github.io/vbluno/vi/mydoc_arduino_tut10_vi.html> [↑](#footnote-ref-2)
3. SQLite: <https://www.sqlite.org/index.html> [↑](#footnote-ref-3)