**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

-------🙞🕮🙜-------



**BÁO CÁO ĐỒ ÁN**

**Môn :**

**PHÁT TRIỄN ỨNG DỤNG CHO CÁC THIẾT BỊ DI ĐỘNG**

**Đề tài :**

**ỨNG DỤNG HỆ THÔNG BÁN HÀNG ONLINE**

Giáo Viên Giảng Dạy: **HUỲNH TRUNG TRỤ**

Họ Tên Sinh Viên Mã Sinh Viên

**VÕ TÙNG NGHĨA N15DCCN136**

**HỒ QUANG VIỆT N15DCCN141**

**PHẠM DUY THƯƠNG N15DCCN187**

**NGUYÊN PHƯƠNG NAM N15DCCN181**

**ĐẬU ANH TUẤN N15DCCN188**

**Thành Phố Hồ Chí Minh – 2019**

**PHẦN I : GIỚI THIỆU**

1. **Giới thiệu**

Hiện nay, Thương Mại Điện Tử (TMĐT) đang được nhiều quốc gia quan tâm, coi là một trong những động lực phát triển chủ yếu của nền kinh tế. TMĐT đem lại những lợi ích tiềm tàng, giúp doanh nghiệp thu được thông tin phong phú về thị trường và đối tác, giảm chi phí tiếp thị và giao dịch, tạo dựng và củng cố quan hệ bạn hàng.

- Việc mua hàng qua mạng chỉ với thủ tục đăng ký mua sắm đơn giản nhưng đem lại nhiều lợi ích: tiết kiệm và chủ động về thời gian, tránh khỏi những phiền phức khó chịu. Vậy nên việc mua bán hàng qua mạng đang rất được mọi người quan tâm. Trên cơ sở các kiến thức được học trong nhà trường và quá trình tìm hiểu các website trong thực tế , em đã quyết định chọn đề tài “Xây dựng ứng dụng hệ thống bán hàng online”.

- Khách hàng chỉ cần các thao tác đơn giản trên điện thoại di động là có thể đặt hàng được ngay

1. **Ưu điểm**

- Dễ dàng quản lý

- Tăng năng suất làm việc

- Tiết kiệm thời gian

1. **Các tính năng**

**3.1 : Công nghệ sử dụng**

* Web service: Dùng RESTful API service

- Sử dụng LinQ to SQL Classes để kết nối CSDL.

* Client:

- Mobile app bán hàng được viết bằng Android.

- Website admin được viết bằng thư viện ReactJs.

**3.2 : Tính năng chính**

- Xem thông tin chi tiết các mặt hàng cần mua

- Đặt mua hàng một cách dễ dàng

- Có thể xem thông tin đơn hàng đã đặt, sản phẩm yêu thích

- Web admin dùng để quản lý việc thêm xóa sửa dữ liệu, thực hiện thông báo những thông tin khuyến mãi, thông tin mới nhất về sản phẩm cho người dùng thông qua Notification

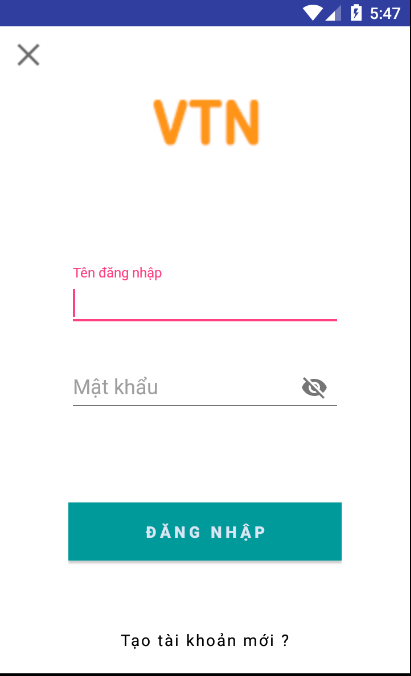
**PHẦN II : ANDROID**

1. **Giao diện**
   1. **Màn hình khởi động (Splash)**

****

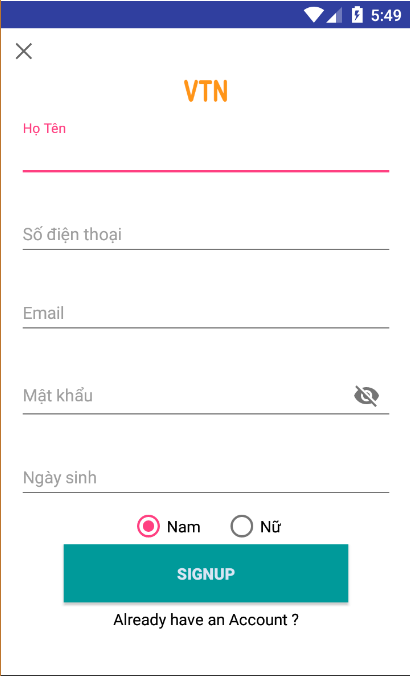
Màn hình khởi động (Splash) hiển thị khi người sử dụng click vào app. **(Võ Tùng Nghĩa)**

* 1. **Màn hình đăng nhập**

****

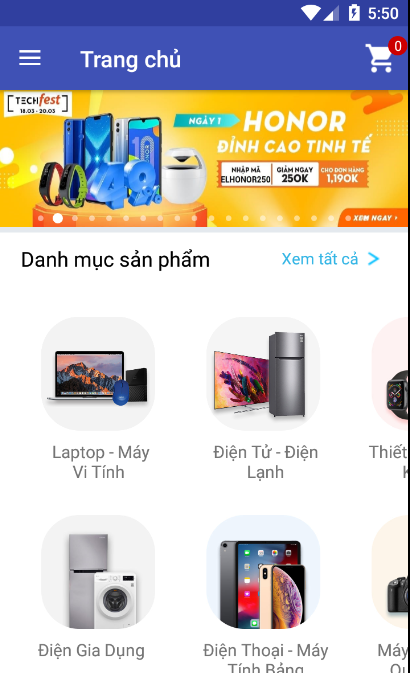
Màn hình đăng nhập khi cần mua hàng hay cập nhập thông tin cá nhân **(Hồ Quang Việt)**

* 1. **Màn hình đăng ký**

****

Màn hình đăng ký giúp khách hàng tạo tài khoản mới nếu muốn mua hàng **(Hồ Quang Việt)**

* 1. **Màn hình trang chủ**

****

Màn hình trang chủ bao gồm :

- Thanh toolbar chưa giỏ hàng và nút mở menu

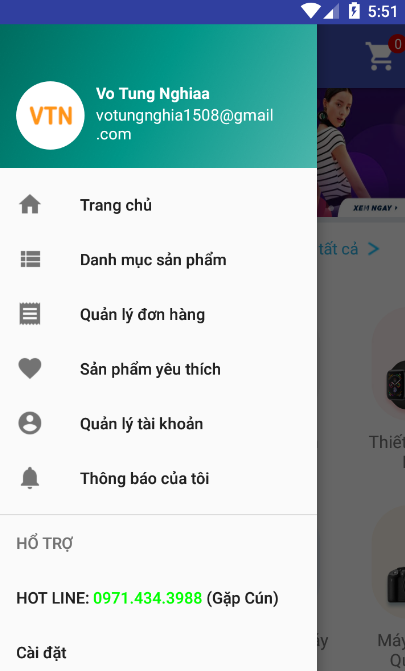
- Slide show dùng để quảng cáo các khuyến mãi

- List danh mục sản phẩm

- List danh sách sản phẩm

**(Võ Tùng Nghĩa)**

* 1. **Màn hình Menu**

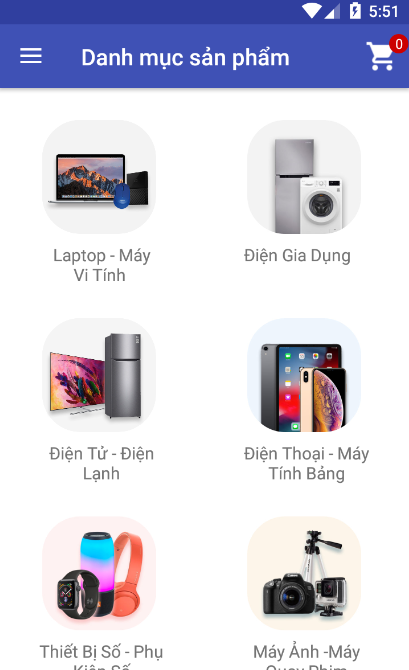
****

Màn hình menu hiện thị thông tin tài khoản đang đăng nhập.

Các danh mục, các màn hình khác .

**(Võ Tùng Nghĩa)**

* 1. **Màn hình danh mục sản phẩm**

****

Màn hình hiển thị tất cả các loại sản phẩm có.

**(Đậu Anh Tuấn)**

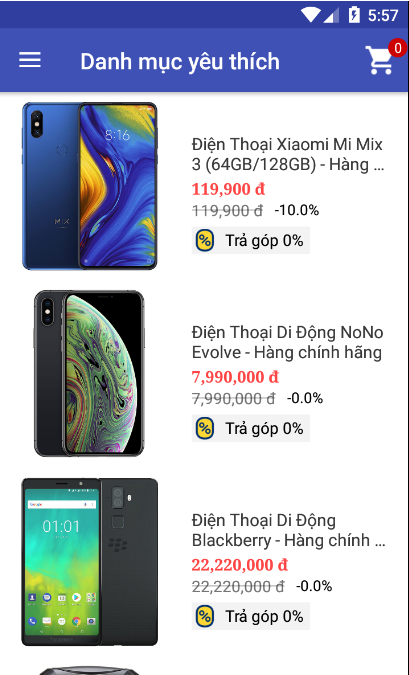
* 1. **Màn hình quản lý đơn hàng**

****

Màn hình hiển thị tất cả các đơn hàng đã đặt của khách hàng.

**(Võ Tùng Nghĩa)**

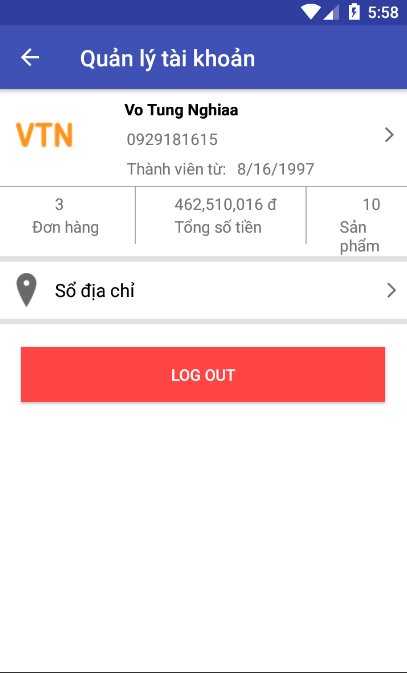
* 1. **Màn hình sản phẩm yêu thích**

****

Màn hình hiển thị tất cả các sản phẩm được thích bởi khách hàng.

**(Võ Tùng Nghĩa)**

* 1. **Màn hình quản lý tài khoản**

****

Màn hình hiển thị tất cả các thông tin về tài khoản của khách hàng khách hàng.

**(Phạm Duy Thương)**

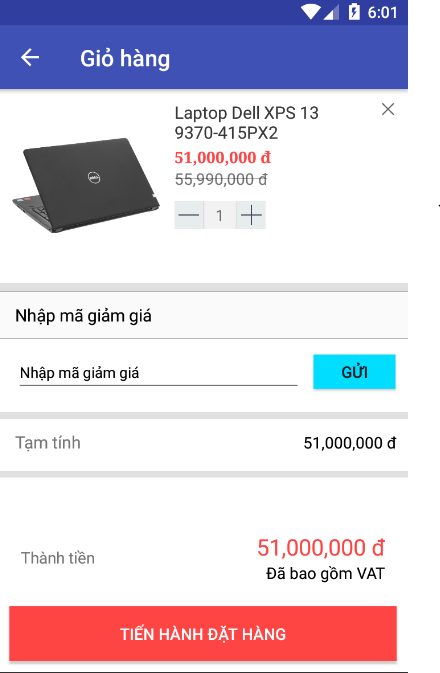
* 1. **Màn hình chi tiết sản phẩm**

****

Màn hình hiển thị chi tiết một sản phẩm có thể xử lý yêu thích và mua hàng trong này

**(Đậu Anh Tuấn)**

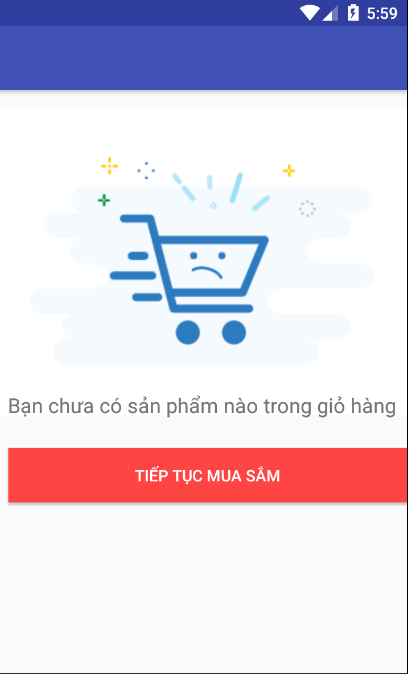
* 1. **Màn hình giỏ hàng**

****

Màn hình hiển thị tất cả các sản phẩm đã chọn mua

**(Nguyễn Phương Nam)**

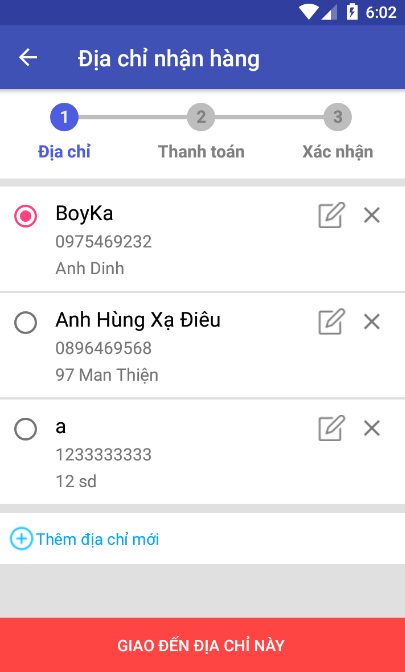
* 1. **Màn hình giỏ hàng trống**

****

Màn hình hiển thị thông báo là hiện tại không có sản phẩm nào được chọn mua

**(Nguyễn Phương Nam)**

* 1. **Màn hình thanh toán**

****

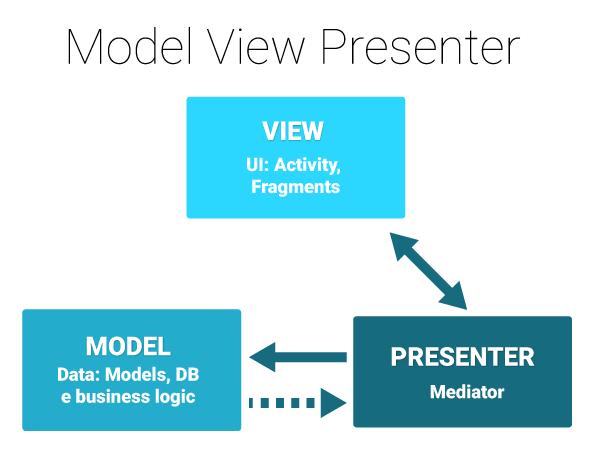
Màn hình hiển thị các bước thanh toán : địa chỉ, phương thức thanh toán và xác nhận.

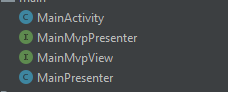
**(Nguyễn Phương Nam)**

1. **Cơ sở lý thuyết**
   1. **Mô hình MVP**

**Công dụng :** MVP là một mô hình kiến trúc hướng giao diện người dùng, được thiết kế để tạo thuận lợi cho việc kiểm thử đơn vị (unit testing) và tăng tính tách biệt giữa tầng dữ liệu và tầng hiển thị dữ liệu trong mô hình MVC.

Mô hình MVP cho phép tách tầng trình diễn (Presenter) ra khỏi tầng dữ liệu (Model), vì vậy tương tác với giao diện được tách biệt với cách chúng ta biểu diễn nó trên màn hình (View), hay nói cách khác, tất cả logic khi người dùng tương tác được tách ra và đưa vào tầng trình diễn. Thiết kế lý tưởng nhất là với cùng một logic được áp dụng cho nhiều View khác nhau và hoán đổi được cho nhau.



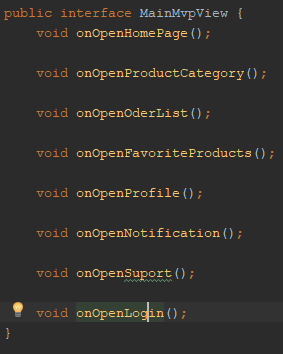
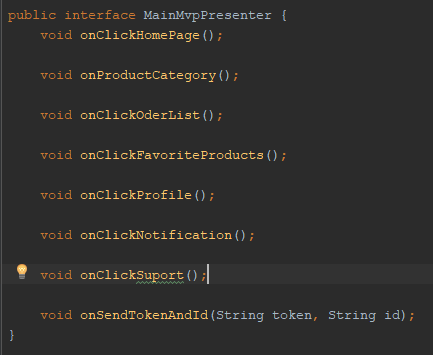
 **Cách dùng :**

- Cơ bản chúng ta sẽ chia các thành phần như bên gồm 2 interface và 2 class

- Ta tiếp tục thiết lập các phương thức cho 2 interface MainMvpPresenter và MainMvpView.

+ MainMvpView : Chúng ta dùng các phương thức này ở MainActivity để thay đổi giao diện .

+ MainMvpPresenter : Chúng ta dung các phương thức này ở MainPresenter để kích hoạt những phương thức của MainMvpView cần thiết.



- Cuối cùng chúng ta implements các interface tương ứng để xử lý các phương thức được định nghĩa sẵn.

C:\Users\votun\Desktop\New folder\Caspture.PNGC:\Users\votun\Desktop\New folder\a.PNG

* 1. **Retrofit 2.0 Library**

- Việc phát triển thư viện HTTP type-safe để giao tiếp với Rest API có thể thực sự là một điều khó khăn vì bạn phải xử lí rất nhiều các chức năng chẳng hạn như : tạo kết nối, lưu trữ cache, thử lại các yêu cầu thất bại, phân tích luồng phản hồi, xử lí lỗi và nhiều hơn nữa. Nhưng mặt khác Retrofit được sinh ra để giải quyết một cách dễ dàng những công việc này và từ đó sẽ giúp bạn tiết kiệm thời gian hơn rất nhiều

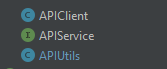
**Công dụng :** Retrofitlà một thư viện vô cùng hữu ích cho việc kết nối internet và nhận dữ liệu từ server một cách dễ dàng và viết code theo mô hình chuẩn RESTFul Webservices**.**Giống như hầu hết các phần mềm mã nguồn mở khác, Retrofit được xây dựng dựa trên một số thư viện mạnh mẽ và công cụ khác. Retrofit không tích hợp bất kỳ một bộ chuyển đổi JSON nào để phân tích từ JSON thành các đối tượng Java. Thay vào đó nó đi kèm với các thư viện chuyển đổi JSON để xử lý điều đó



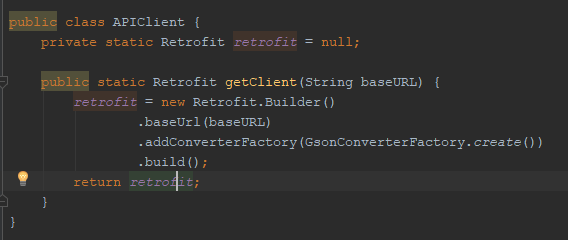
**Cài đặt :**

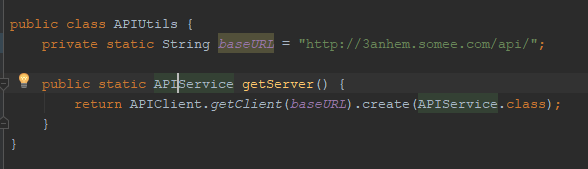
implementation 'com.squareup.retrofit2:retrofit:2.5.0'  
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'  
implementation 'com.google.code.gson:gson:2.8.5'

**Cách dùng :**

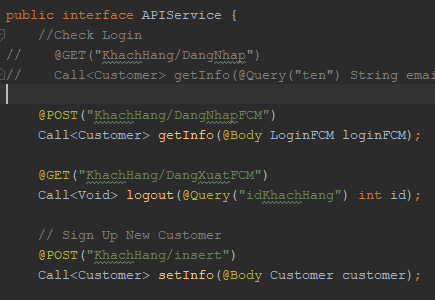
**-** Đầu tiên chúng ta sẽ cài đặt trước ta dùng 2 class và 1 interface như hình dưới :

- Tiếp theo chúng ta sẽ setup cho 2 class trên





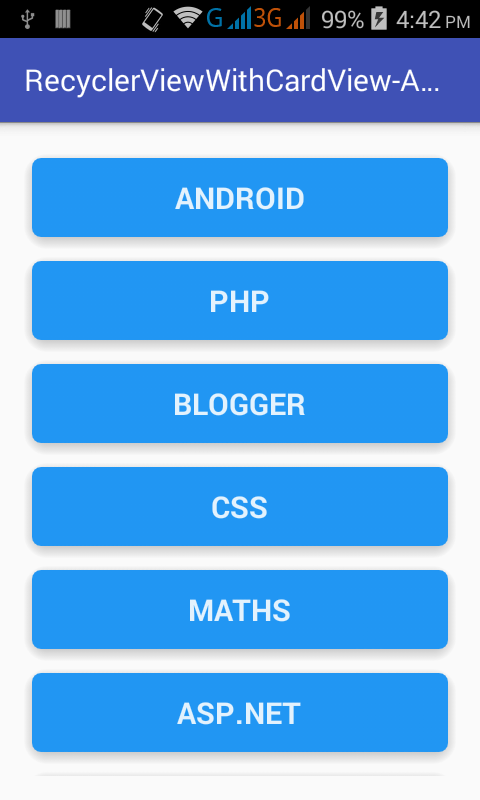
- Về phí Interface ApiService chúng ta sẽ chứa những phương thức Get, Post,Delete để thực hiện request lên server

****

* 1. **RecyclerView Library**

**- Công dụng:**  RecyclerView là một ViewGroup mới, nó là sự kế thừa và nâng cao của ListView và GridView được hỗ trợ trong support-v7 version. Một trong những ưu điểm nổi trội của Recycler là nó có khả năng mở rộng tốt hơn, nó có thể hiển thị dữ liệu theo cả chiều ngang và chiều dọc.

\*\* **So sánh khác biệt cơ bản với ListView** \*\*

* \*Required ViewHolder in Adapters \* ListView không yêu cầu bắt buộc sử dụng ViewHolder để cải thiện hiệu suất. Ngược lại khi bạn sử dụng \* RecyclerView thì điều đó là bắt buộc.
* \* Customizable Item Layouts \* Để hiển thị dữ liệu theo chiều ngang thì bạn không thể sử dụng ListView mà bạn cần đến HorizontalScrollView. Nhưng với Recycler thì chúng ta có tùy chọn RecyclerView.LayoutManager để chỉ định cách hiển thị dữ liệu.
* \* Manual Data Source \* ListView có 2 Adapter để bạn có thể tùy chỉnh điều hướng dữ liệu là \*ArrayAdapter \* và CursorAdapter, còn với RecyclerView thì bạn bắt buộc phải sử dụng RecyclerView.Adapter. 

- **Cài dặt** :

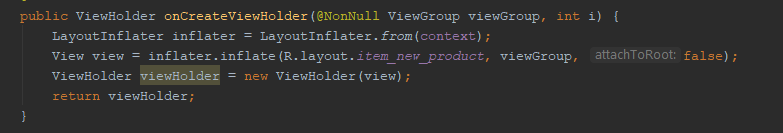
* implementation 'com.android.support:recyclerview-v7:28.0.0'

- **Cách dùng** :

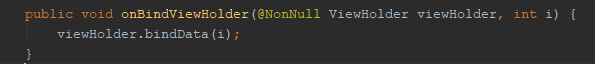
+ Đầu tiên bạn cần setup 1 adapter cho 1 recyclerview

C:\Users\votun\Desktop\New folder\Captưure.PNG

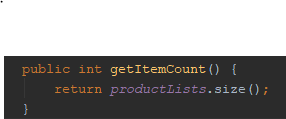
Dùng để gán giao diện cho một phần tử của RecyclerView



Dùng để gán dữ liệu từ listData vào viewHolder

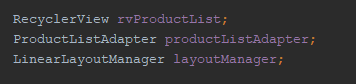


Trả về số lượng phần tử

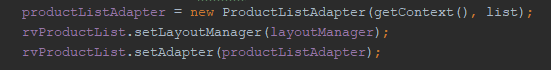


- Ở Activity ta cần

Khởi tạo các phần tử cần thiết



Khi ta đã có dữ liệu ta DB ta sẽ truyền 1 Context, 1 listData vào adapter, set layout cho recycelerview cuối cùng là set adapter cho nó



* 1. **Glide Library**

- **Công dụng** : Glide là một thư viện open source hỗ trợ load ảnh trên Android. Dùng Glide sẽ đơn giản hóa các công việc mà bạn cần làm khi sử dụng một bức ảnh trong Android đi rất nhiều. Chúng ta không cần quan tâm đến việc decoding, memory and disk caching mà thay vào đó chỉ cần sử dụng interface rất đơn giản từ Glide.

- Glide hỗ trợ fetching, decoding và hiển thị cả ảnh tĩnh hoặc ảnh động đó là điểm nổi bật của nó so với các thư viện load ảnh khác cho android. Glide có thể được dễ dàng đưa vào bất kỳ mô hình mạng nào. Mặc định Glide sử dụng mô hình custom của HttpUrlConnection tuy nhiên chúng ta có thể dễ dàng cho nó hoạt động với Volley hoặc OkHttp

- Glide tính đến hai khía cạnh chính của hiệu suất tải hình ảnh trên Android.

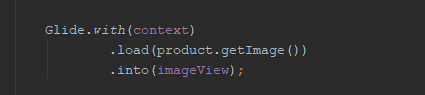
* Tốc độ hình ảnh có thể được decoded
* Số lượng jank phát sinh khi trong khi decoding hình ảnh. Để người dùng có trải nghiệm tuyệt vời với ứng dụng, hình ảnh không chỉ xuất hiện nhanh mà còn phải gây ra nhiều jank và lặp từ Main Thread hoặc gây ra nhiều rác quá mức. Glide thực hiện một số bước để đảm bảo tải hình ảnh nhanh và mượt mà nhất có thể trên Android:
* Tự động downsampling và giảm thiểu chi phí lưu trữ cộng với thời gian decode.
* Tích cực tái sử dụng resource như mảng byte và bitmap giảm thiểu garbage collections và phân mảnh heap.
* Tích hợp sâu vào vòng đời và đảm bảo chỉ yêu cầu đối với Activity và Fragment được ưu tiên, và các ứng dụng giải phóng tài nguyên khi cần thiết



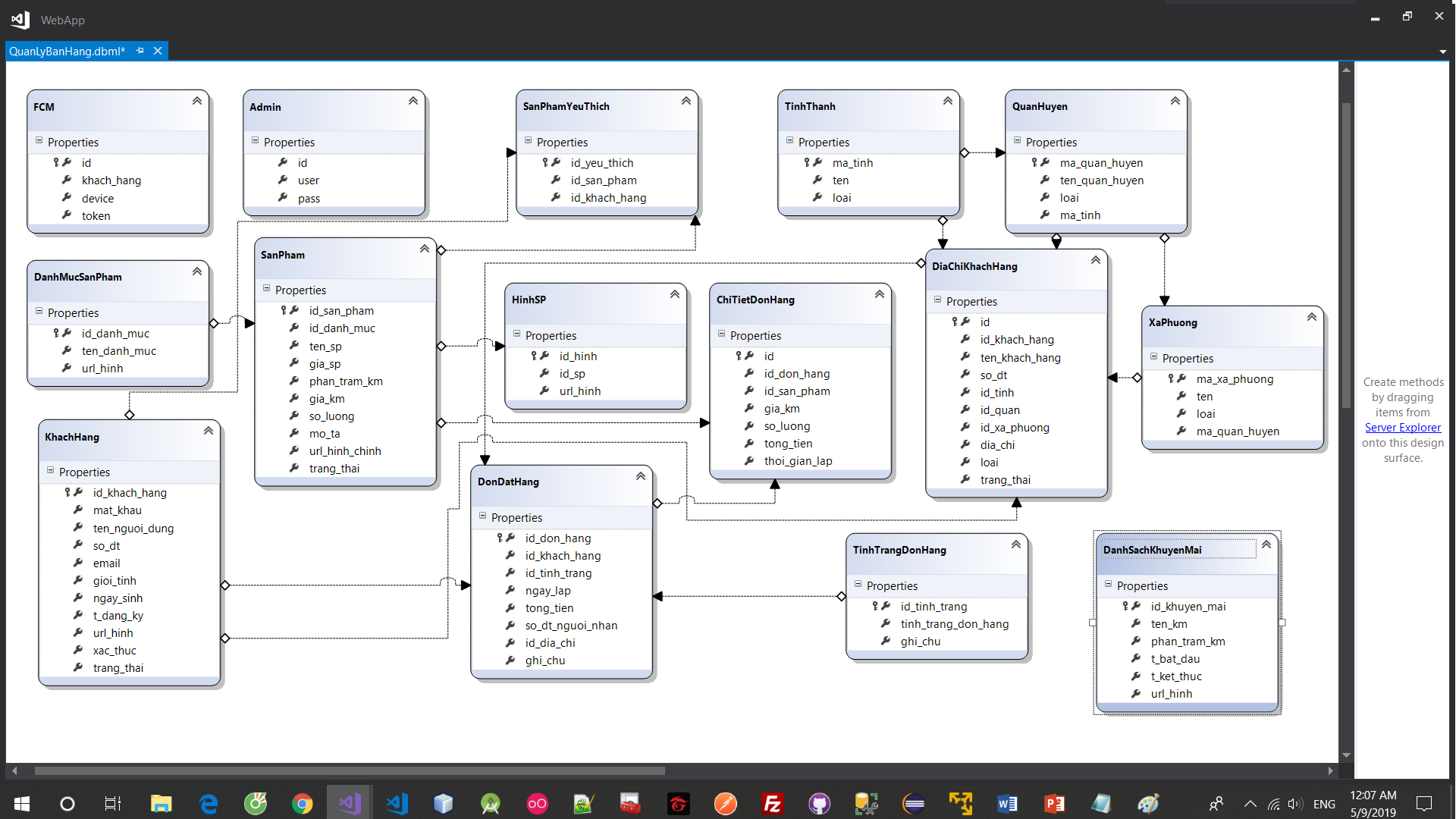
- **Cài đặt** :

implementation 'com.github.bumptech.glide:glide:4.8.0'  
annotationProcessor 'com.github.bumptech.glide:compiler:4.8.0'

- **Cách dùng** :

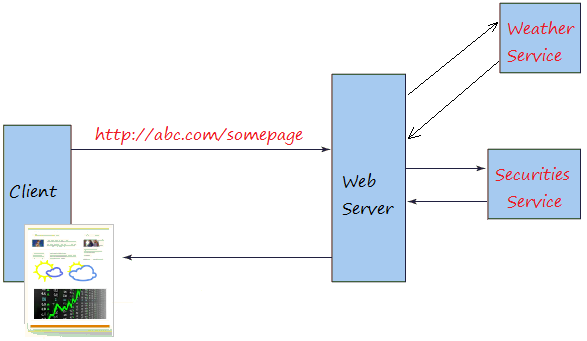


**PHẦN III: DATABASE**

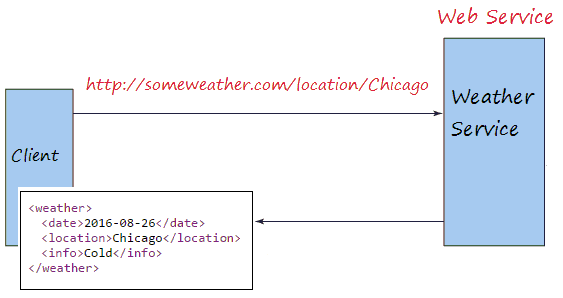
****

**PHẦN IV : WEB SERVICE**

1. **GIỚI THIỆU**

Web Service là một dịch vụ web, nó là một khái niệm rộng hơn so với khái niệm web thông thường, nó cung cấp các thông tin thô, và khó hiểu với đa số người dùng (chúng thường được trả về dưới dạng XML hoặc JSON.), chính vì vậy nó được sử dụng bởi các ứng dụng. Các ứng dụng này sẽ chế biến các dữ liệu thô trước khi trả về cho người dùng cuối cùng. 

RESTful Web Service là các Web Service được viết dựa trên kiến trúc REST. REST đã được sử dụng rộng rãi thay thế cho cácWeb Service dựa trên SOAP và WSDL. RESTful Web Service nhẹ (lightweigh), dễ dàng mở rộng và bảo trì.

RESTFul giống như 1 kiểu kiến trúc gửi/ nhận (client gửi request/dữ liệu tới, server trả dữ liệu tương ứng về)

1. **NGHUYÊN TẮC**
   1. **Sử dụng các phương thức HTTP một cách rõ ràng**

REST đặt ra một quy tắc đòi hỏi lập trình viên xác định rõ ý định của mình thông qua các phương thức của HTTP. Thông thường ý định đó bao gồm lấy dữ liệu, trèn dữ liệu, cập nhập dữ liệu hoặc xóa dữ liệu. Vậy khi bạn muốn thực hiện một trong các ý định trên hãy lưu ý các quy tắc sau:

* Để tạo một tài nguyên trên máy chủ, bạn cần sử dụng phương thức POST.
* Để truy xuất một tài nguyên, sử dụng GET.
* Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng PUT.
* Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng DELETE.

các nguyên tắc ở trên là không bắt buộc, thực tế bạn có thể sử dụng phương thức GET để yêu cầu lấy dữ liệu, trèn, sửa hoặc xóa dữ liệu trên Server. Tuy nhiên REST đưa ra các nguyên tắc ở trên mục đích đưa mọi thứ trở lên rõ ràng và dễ hiểu.

* 1. **Phi trạng thái**

Một đặc điểm của REST là phi trạng thái (stateless), có nghĩa là nó không lưu giữ thông tin của client. Chẳng hạn bạn vừa gửi yêu cầu để xem trang thứ 2 của một tài liệu, và bây giờ bạn muốn xem trang tiếp theo (sẽ là trang 3). REST không lưu trữ lại thông tin rằng trước đó nó đã phục vụ bạn trang số 2. Điều đó có nghĩa là REST không quản lý phiên làm việc (Session).

* 1. **Hiển thị cấu trúc thư mục như các Urls**

REST đưa ra một cấu trúc để người dùng có thể truy cập vào tài nguyên của nó thông qua các URL, tài nguyên ở đây là tất cả những cái mà bạn có thể gọi tên được (Video, ảnh, báo cáo thời tiết,..)

* 1. **Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai**

Khi Client gửi một yêu cầu tới web service nó thường được truyền tải dưới dạng XML hoặc JSON và thông thường nhận về với hình thức tương tự.

Đôi khi Client cũng có thể chỉ định kiểu dữ liệu nhận về mà nó mong muốn (JSON, hoặc XML,..), các chỉ định này được gọi là các kiểu MINE, nó được gửi kèm trên phần HEADER của request.

Dưới đây là các kiểu MINE phổ biến thường sử dụng với REST service.

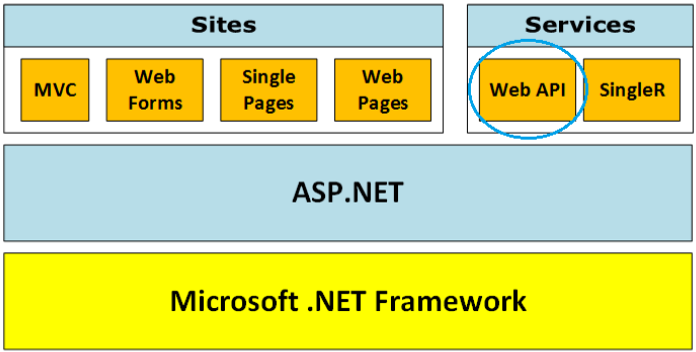
| **MIME-Type** | **Content-Type** |
| --- | --- |
| **JSON** | application/json |
| **XML** | application/xml |
| **XHTML** | application/xhtml+xml |

1. **API RESTful Service** 
   1. **Giới thiêu về ASP.NET Web API**

* **ASP.NET Web API là gì?**

Là framework giúp chúng ta tạo ra các Web API – API trên nền web (HTTP).

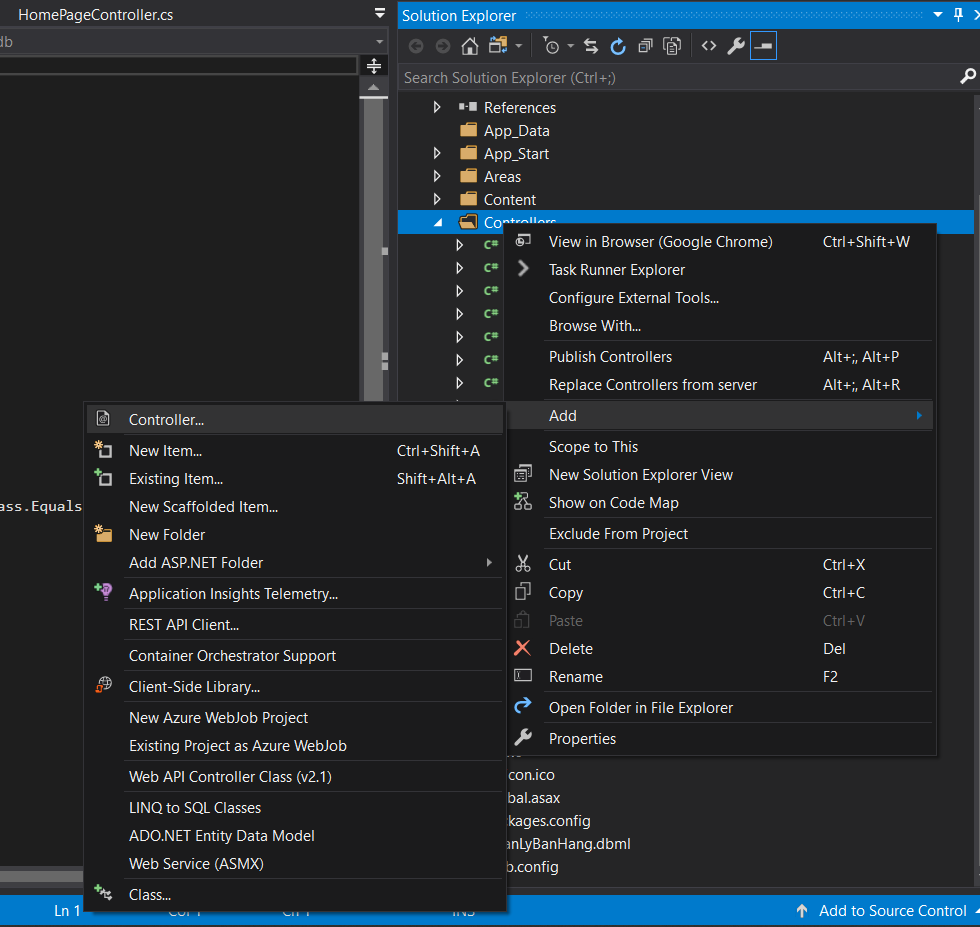
Web API là các dịch vụ Web (Web service) được xây dựng dựa trên HTTP sử dụng mô hình lập trình convention (như ASP.NET MVC).

****

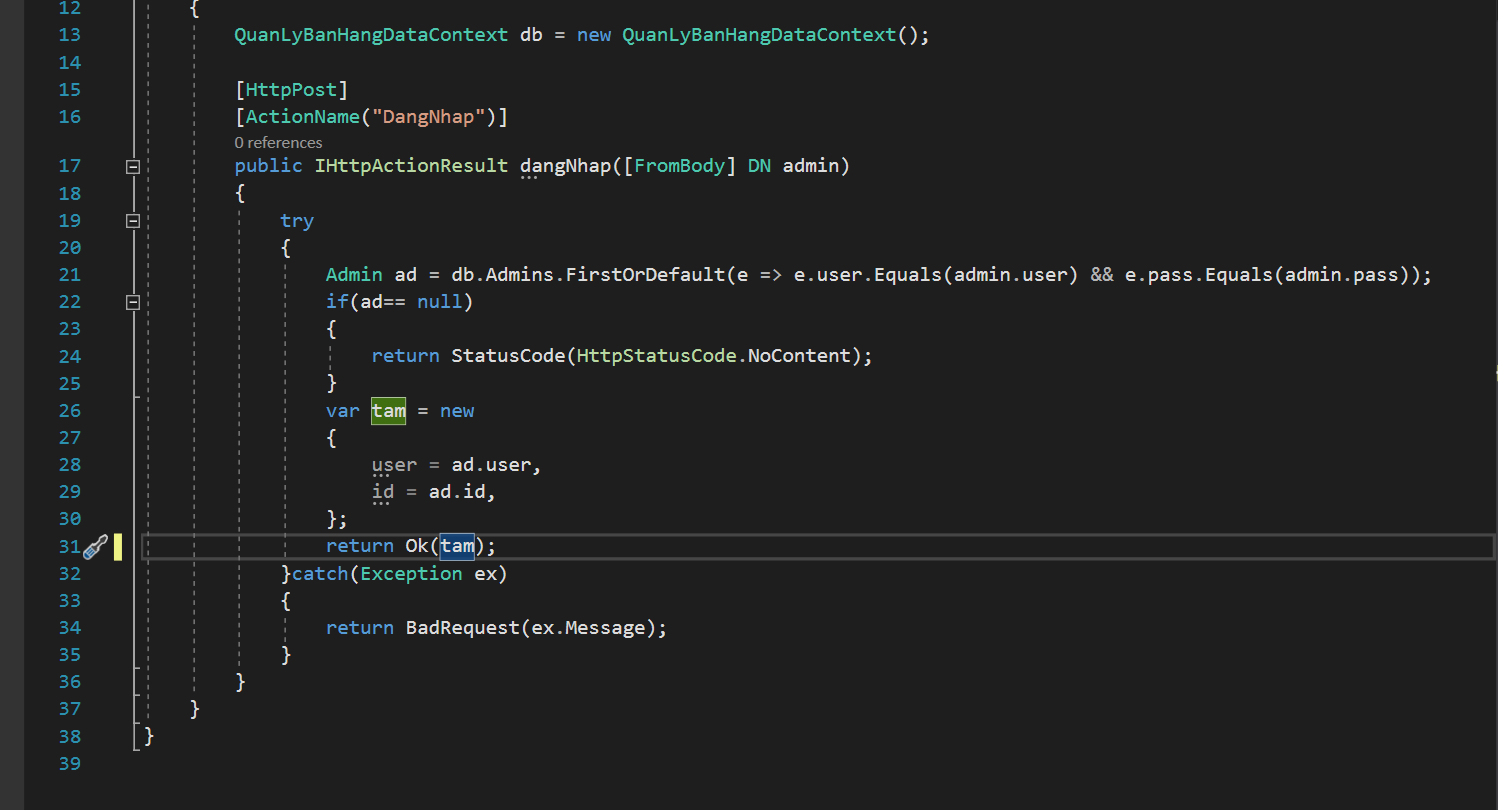
* **Đặc điểm Web API(.NET 4.0 trở lên)**
* Giúp cho việc xây dựng các HTTP service rất đơn giản, nhanh chóng
* Mã nguồn mở (Open Source) và có thể được sử dụng bởi bất kì client nào hỗ trợ XML, JSON.
* Hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content formats.
* Có thể host trong ứng dụng hoặc trên IIS.
* Kiến trúc lý tưởng cho các thiết bị có băng thông giới hạn như các thiết bị di động.
* Định dạng dữ liệu có thể là JSON, XML hoặc một kiểu dữ liệu bất kỳ.
* Làm mới và hiện đại hóa các mẫu dự án mặc định
* Mẫu dự án trên điện thoại di động
* Nhiều tính năng mới để hỗ trợ các ứng dụng di động
* Tùy chỉnh sinh mã(code).
* Tăng cường hỗ trợ cho các phương pháp bất đồng bộ
* Đọc danh sách đầy đủ tính năng trong các ghi chú phát hành
* **Ưu điểm của Web API**
* Cấu hình đơn giản hơn nhiều so với WCF.
* Hiệu suất(performance) cao.
* Hỗ trợ RESTfull đầy đủ.
* Hỗ trợ đầy đủ các thành phần MVC như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test, …
* Mã nguồn mở (Open source).
  1. **Các thành phần chính**

Gồm có ba thành phần chính là: Contronller, Model, View

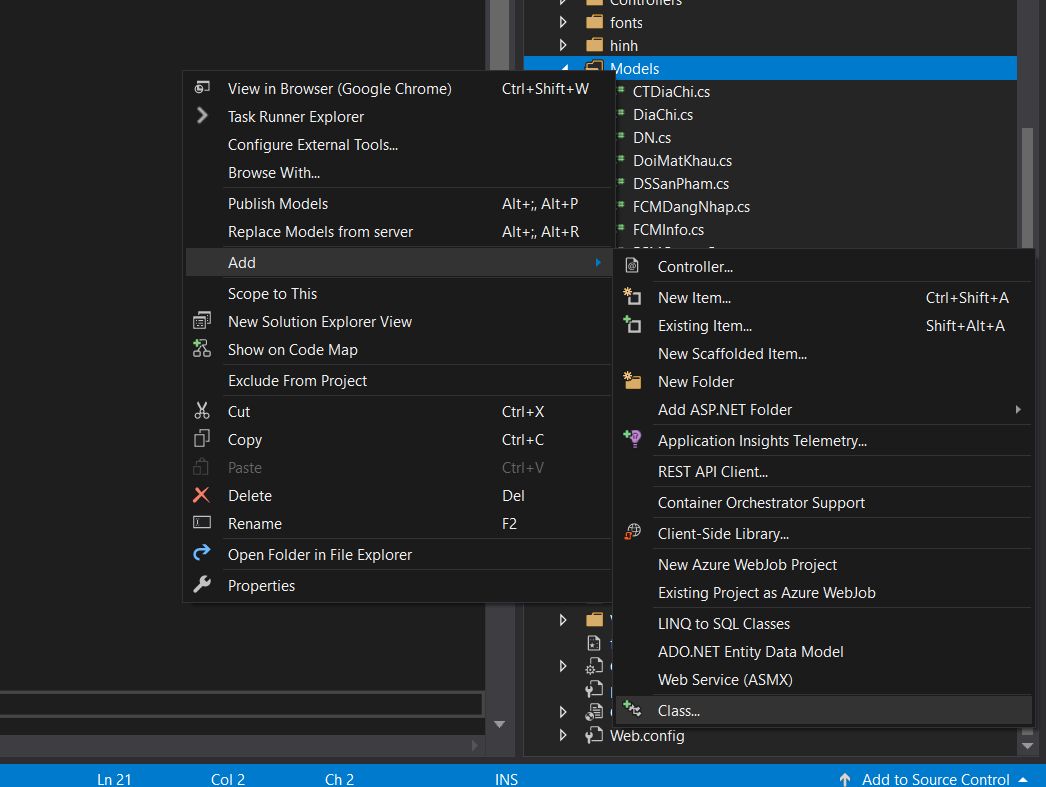
* **Controller** : có chức năng điều khiển, sắp xếp và xử lý các yêu cầu của người dùng.
  + - * Các file contronller được lưu tại thư mục \WebApp\Controllers.
      * Cách tạo các file contronller: bấm chuột phải vào Folder contronller/chọn Add/chọn **Controller**



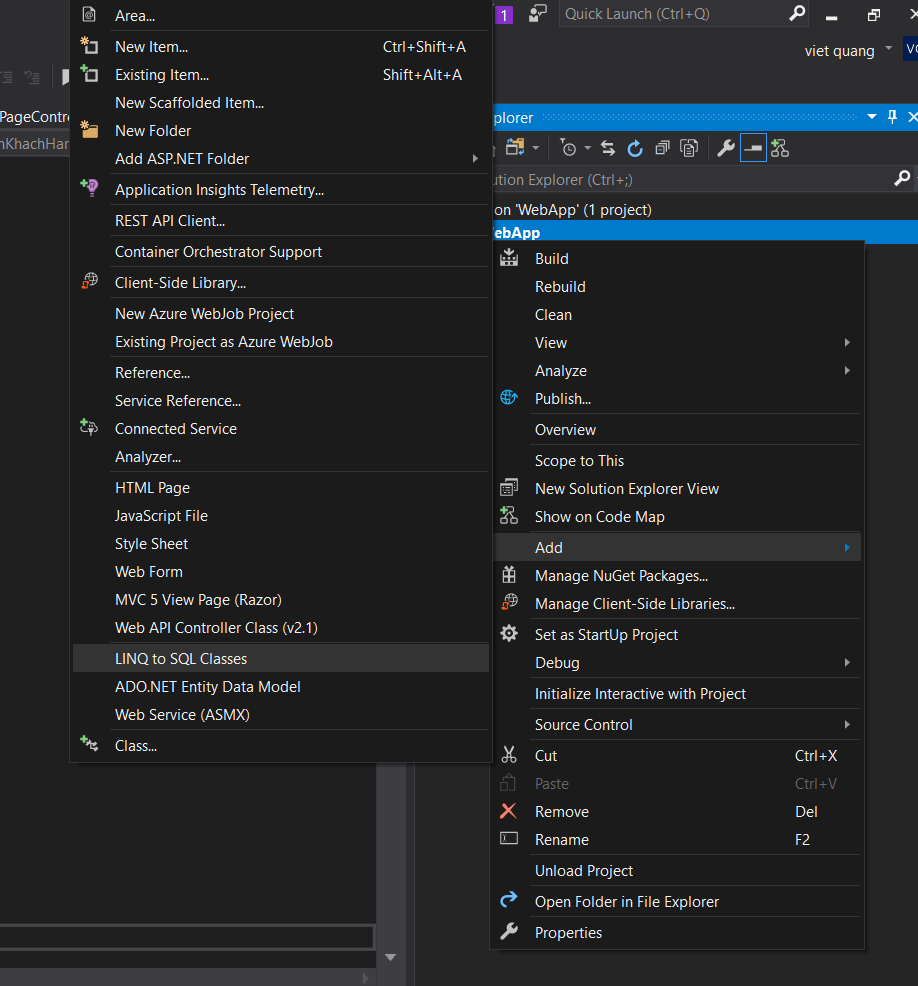
* + - * **Cách viết các phương thức GET, POST, PUT, DELETE**

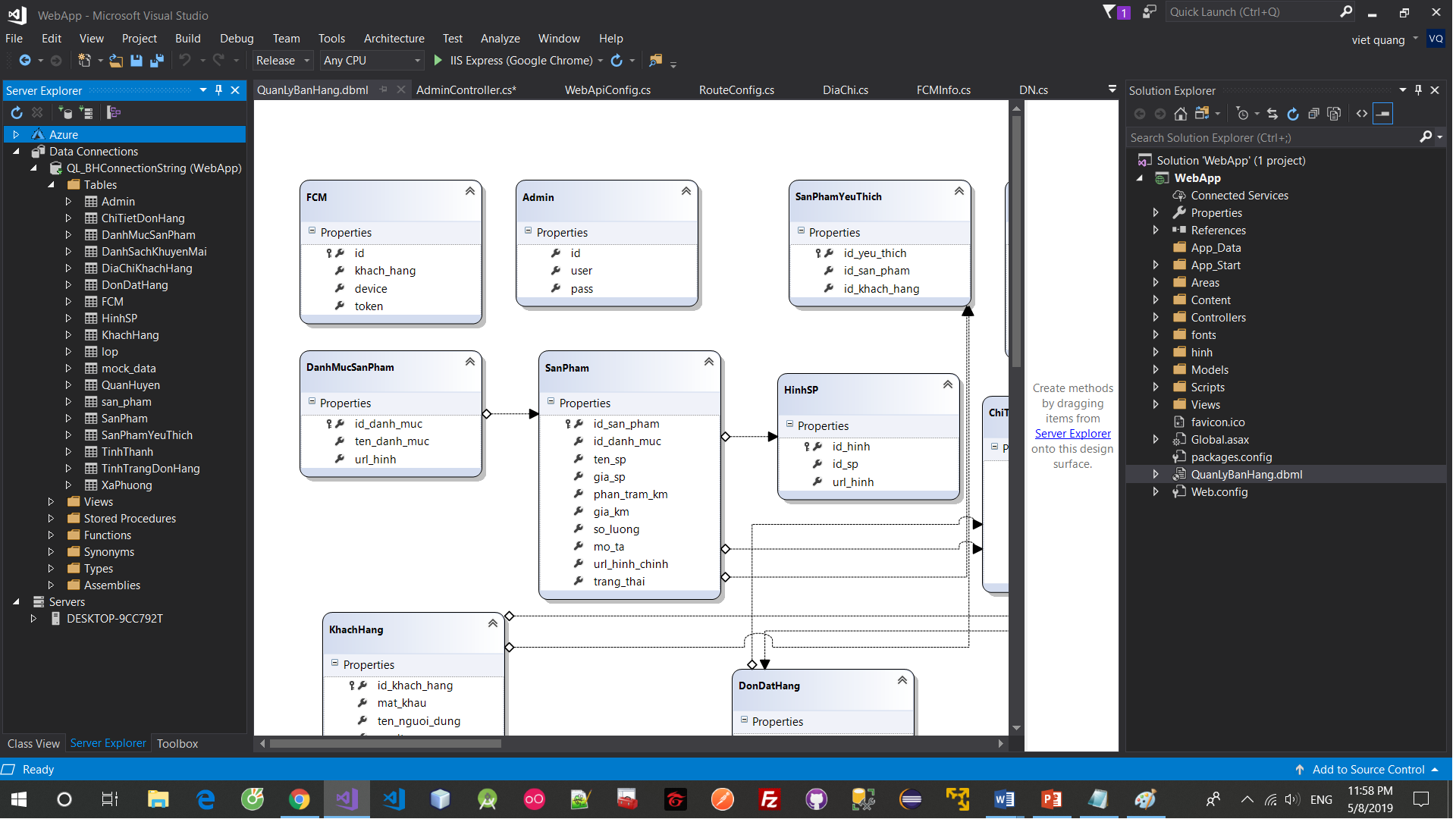


* **Model** : Nơi chưa các đôi tượng.
  + - Các file Model được lưu tại thư mục \WebApp\Models.
    - Cách tạo các file Model: bấm chuột phải vào Folder Models/chọn Add/chọn **Class**



* **View** : là thành phần giao diện, hiển thị dữ liệu cho người dùng.
* **Sử dụng LINQ to SQL Classes:** là thành phần kết nối cơ sở dự liêu.
  + - Cách tao file: bấm chuột phải vào Project /chọn Add/chọn **LINQ to SQL Classes**





* 1. **Các chức năng API**
* Admin
* POST: …/api/Admin/DangNhap : Đăng nhập web admin
* KhachHang
* POST: …/api/KhachHang/DangNhapFCM : Đăng nhập
* GET: …/api/KhachHang/DangXuatFCM : Đăng xuất
* GET: …/api/KhachHang/danhsach : Hiển thị danh sách khách hàng
* GET: …/api/KhachHang/danhsach?page={page}&size={size} : Hiển thị danh sách khách hàng theo trang và số lượng khách hàng trên một trang
* GET: …/api/KhachHang/getMot?id={id} : Lấy thông tin của một khách hàng theo id
* POST: …/api/KhachHang/insert : Thêm mới một khách hàng
* POST: …/api/KhachHang/update : Sửa thông tin
* DELETE: …/api/KhachHang/delete?id={id} : xóa tài khoan
* SanPham
* GET: …/api/SanPham/getMotSanPham?id={id} : Lấy một sản phẩm theo id
* GET: …/api/SanPham/MotSanPham?idSanPham={idSanPham}&idKhachHang={idKhachHang} : Lấy một sản phẩm có thêm thuộc tính yêu thích
* GET: …/api/SanPham/getTheoDanhMuc?id={id} : Lấy danh sách sản phẩm theo danh mục
* GET: …/api/SanPham/TheoDanhMuc?id={id}&locTheo={locTheo} : Lấy danh sach theo danh mục sản phẩm và lọc theo thứ tự( giá cao đến thấp, thấp đến cao,…)
* POST: …/api/SanPham/DSPhanTrang : Lấy danh sách sản phẩm theo phân trang
* GET: …/api/SanPham/DanhSach : Lấy danh sách sản phẩm cho web admin
* GET: …/api/SanPham/DanhSach?page={page}&size={size} : Lấy danh sách sản phẩm cho web admin
* PUT: …/api/SanPham/update : Sửa sản phẩm
* POST: …/api/SanPham/insert : Thêm mới sản phẩm
* DELETE: …/api/SanPham/delete/{id} :Xóa sản phẩm
* SanPhamYeuThich
* GET: .../api/SanPhamYeuThich/layMot?id={id} : Lấy một sản phẩm yêu thích
* POST: .../api/SanPhamYeuThich/kiemTra?idSanPham={idSanPham}&IdKhachHang={IdKhachHang} : Kiểm tra
* POST: .../api/SanPhamYeuThich/DanhSach : Lấy danh sách sản phẩm yêu thích
* POST: .../api/SanPhamYeuThich/themMoi : Thêm mới một sản phẩm yêu thích
* DELETE: .../api/SanPhamYeuThich/Xoa?id={id} :Xóa một sản phẩm yêu thích
* DanhMucSanPham
* GET: .../api/DanhMucSanPham/danhSach : Lấy danh sách danh mục sản phẩm
* GET: .../api/DanhMucSanPham/layMot?id={id} : Lây một loai sản phẩm theo id
* POST: .../api/DanhMucSanPham/them : Thêm một loại sản phẩm
* PUT: .../api/DanhMucSanPham/sua : Sửa loại sản phẩm
* DELETE: .../api/DanhMucSanPham/xoa?id={id} Xóa một loại sản phẩm theo id
* DanhSachKhuyenMai
* GET: .../api/DanhSachKhuyenMai/getListKhuyenMai : Lấy danh sách
* GET: .../api/DanhSachKhuyenMai/getMotKhuyenMai?id={id} : Lấy một loại khuyến mãi
* POST: .../api/DanhSachKhuyenMai/suahinh : Sửa hình ảnh khuyến mãi
* POST: .../api/DanhSachKhuyenMai/them : Thêm khuyến mãi
* PUT: .../api/DanhSachKhuyenMai/updateKhuyenMai : Sửa
* DELETE: .../api/DanhSachKhuyenMai/deleteKhuyenMai?id={id} : Xóa
* DonDatHang
* POST: .../api/DonDatHang/getList : Lấy danh sách hóa đơn
* GET: .../api/DonDatHang/hoaDon?id={id} :Hiển thị một hóa đơn đặt hàng
* POST: .../api/DonDatHang/TaoHoaDon : Tạo một hóa đơn đặt hàng mới
* GET: .../api/DonDatHang/huyHoaDon?id={id} : Hủy một hóa đơn đạt hàng theo id
* GET: .../api/DonDatHang/danhsach : Hiển thị danh sách hóa đơn
* GET: .../api/DonDatHang/danhsach?page={page}&size={size} : Hiển thị danh sách hóa đơn theo phân trang
* POST: .../api/DonDatHang/chuyentrangthai?idDonHang={idDonHang}&idTinhTrang={idTinhTrang} : Thay đổi trạng thái của hóa đơn
* ChiTietDonDatHang
* GET: .../api/ChiTietDonHang/getListMotDon?id={id} : Hiển thị danh sách chi tiết hóa đơn
* POST: .../api/ChiTietDonHang/insert : Thêm một chi tiết hóa đơn
* PUT: .../api/ChiTietDonHang/update : Sưa
* DELETE: .../api/ChiTietDonHang/xoa?idDonHang={idDonHang} :Xóa
* TinhTrangDonHang
* GET: .../api/TinhTrangDonHang/danhsach : Lấy danh sách tình trạng hóa đơn
* GET: .../api/TinhTrangDonHang/LayMot?id={id} :Lấy một tình trạng
* POST: .../api/TinhTrangDonHang/them : Thêm mới
* DELETE: .../api/TinhTrangDonHang/delete?id={id} : Xóa một theo id
* DiaChiKhachHang
* GET: .../api/DiaChiKhachHang/DanhSach?id={id} : Hiển thì danh sách địa chỉ của một khách hàng
* GET: .../api/DiaChiKhachHang/DanhSach?page={page}&size={size} :Hiển thị danh sách địa chỉ theo phân trang
* GET: .../api/DiaChiKhachHang/layMotDiaChi?id={id} : Lấy một địa chỉ theo id
* POST: .../api/DiaChiKhachHang/taoDiaChiMoi : Thêm địa chỉ mới
* POST: .../api/DiaChiKhachHang/update : Sửa
* DELETE: .../api/DiaChiKhachHang/XoaDiaChi?id={id} : Xóa
* Tinh
* GET: .../api/Tinh/lay : Lấy danh sách các tỉnh thành trong nước
* Quan
* GET: .../api/Quan/lay?idTinh={idTinh} : Lấy danh sách các quận huyện thuộc tỉnh có idTinh
* Xa
* GET: .../api/Xa/lay?idQuan={idQuan} : lấy danh sách các xa thuôc quân huyện có idQuan

**PHẦN V : WEB ADMIN**

**1.Các công nghệ được áp dụng**

* Framework react JS
* Thư viện mdbootstrap giúp trang trí website
* Công nghệ single page application

1. **Cấu trúc thư mục mã nguồn**

* api: folder chứa các file javascript, chứa các hàm thực thi gọi lên API và nhận lại response từ api.  
  Dưới đây là một hàm của một file trong folder api. Hàm sử dụng hàm fetch của js để có thể gọi lên api và trả kết quả. Vì đây là một hàm bất đồng bộ của js nên phải sử dụng async/await cho hàm.
* export async function FindAll() {
* try {
* let response = await fetch(URL\_API + "DanhMucSanPham/danhSach", {
* method: "get",
* })
* if (response.status == 200) {
* return await response.json();
* }
* } catch (error) {
* console.log(error);
* return null;
* }
* }
* Assets: chứa các file tĩnh, là các file như hình ảnh, css, js giúp trang trí trang web
* Components: chứa các component của react. Đây là các file mã nguồn chính. Thực hiện gọi các hàm từ folder api, và nhận lại kết quả sau đó thực hiện hàm render của react để thể hiện dữ liệu nhận được từ server.
* Đầu tiền là phần import các thư viện, file cần dùng trong component
* Tiếp theo là khai báo component được kế thừa từ React.Component
* Trong component sẽ có hàm render giúp thể hiện data. Chúng ta sẽ xử lý để có được dữ liêu sau đó gán dữ liệu vào state để thực hiện render.
* import React, { Component } from 'react';
* import {
* MDBModal,
* MDBModalBody,
* MDBModalFooter,
* MDBContainer,
* MDBBtn,
* toast,
* ToastContainer,
* MDBPagination,
* MDBPageItem,
* MDBPageNav,
* MDBCol,
* MDBRow
* } from 'mdbreact';
* import { NavLink } from 'react-router-dom';
* import Loading from '../loading';
* import \* as CategoryApi from '../../../api/category';
* export default class Category extends Component {
* //#region CONSTRUCTOR
* constructor(props) {
* super(props);
* this.state = {
* isShowLoading: false,
* isShowModalDelete: false,
* categoryId: 0,
* categories: []
* }
* this.init();
* // this.test1();
* }
* //#endregion
* //#region INIT
* async init() {
* this.setState({ isShowLoading: true })
* let categories = await CategoryApi.FindAll();
* if (categories != null) {
* this.setState({ categories: categories });
* }
* }
* //#endregion
* //#region MODAL FUNCTION
* toggleModalDelete(categoryId) {
* this.setState(previousState => ({
* categoryId: categoryId,
* isShowModalDelete: !previousState.isShowModalDelete
* }));
* }
* //#endregion
* //#region DELETE
* async confirmDelete() {
* let result = CategoryApi.Delete(this.state.categoryId);
* if (result == null) {
* toast.error("Hệ thống lỗi. Vui lòng thử lại");
* return;
* }
* if (result.ResponseCode == 1) {
* toast.warn(result.ResponseMessage);
* return;
* }
* this.init();
* this.setState(previousState => ({
* isShowModalDelete: !previousState.isShowModalDelete
* }));
* }
* //#endregion
* //#region RENDER
* render() {
* return (
* <MDBContainer fluid>
* <Loading isShow={this.state.isShowLoading} />
* <MDBContainer>
* <MDBModal isOpen={this.state.isShowModalDelete} toggle={() => this.toggleModalDelete()}>
* <MDBModalBody>
* Bạn có chắc muốn xóa
* </MDBModalBody>
* <MDBModalFooter>
* <MDBBtn color="secondary" onClick={() => { this.confirmDelete() }}>Xóa</MDBBtn>
* <MDBBtn color="danger" onClick={() => this.toggleModalDelete()}>Đóng</MDBBtn>
* </MDBModalFooter>
* </MDBModal>
* </MDBContainer>
* <ToastContainer
* hideProgressBar={true}
* newestOnTop={true}
* autoClose={5000}
* />
* <NavLink to="/add-category" activeClassName="activeClass">
* <MDBBtn color="primary">Thêm mới</MDBBtn>
* </NavLink>
* <MDBRow>
* <table className="table table-striped table-bordered">
* <thead>
* <tr>
* <th scope="col">#</th>
* <th scope="col">Tên danh mục</th>
* <th scope="col">Hình ảnh</th>
* <th scope="col">Xử lý</th>
* </tr>
* </thead>
* <tbody>
* {
* this.state.categories.map(element => {
* return (
* <tr key={element.id\_danh\_muc}>
* <th scope="row">{element.id\_danh\_muc}</th>
* <td>{element.ten\_danh\_muc}</td>
* <td>
* <img src={element.url\_hinh} style={{ width: 200 }} />
* </td>
* <td>
* <NavLink
* to={"/edit-category/" + element.id\_danh\_muc}
* activeClassName="activeClass">
* <MDBBtn color="success" size="sm">Sửa</MDBBtn>
* </NavLink>
* {/\* <MDBBtn
* onClick={() => { this.toggleModalDelete(element.id\_danh\_muc) }}
* color="danger"
* size="sm">Xóa</MDBBtn> \*/}
* </td>
* </tr>
* )
* })
* }
* </tbody>
* </table>
* </MDBRow>
* </MDBContainer >
* );
* }
* //#endregion
* }
* File index.js: Đây là file mặc định của react, sẽ được chạy đầu tiên khi web được gọi. Cấu trúc của cũng gần giống với component. Nhưng thay vì dùng hàm render thì sẽ dùng RenderDOM.render
* ReactDOM.render(<Router><App /></Router>, document.getElementById('root'));
* registerServiceWorker.js: Giúp website có thể sử dụng cookie hoặc session khi trong trình trạng mạng yếu hoặc không mạng.
* Ngoài ra còn có các file hỗ trợ như package.json chứa các cấu hình, thư viện sử dụng.

1. **Giới thiệu về single page application(SPA)**

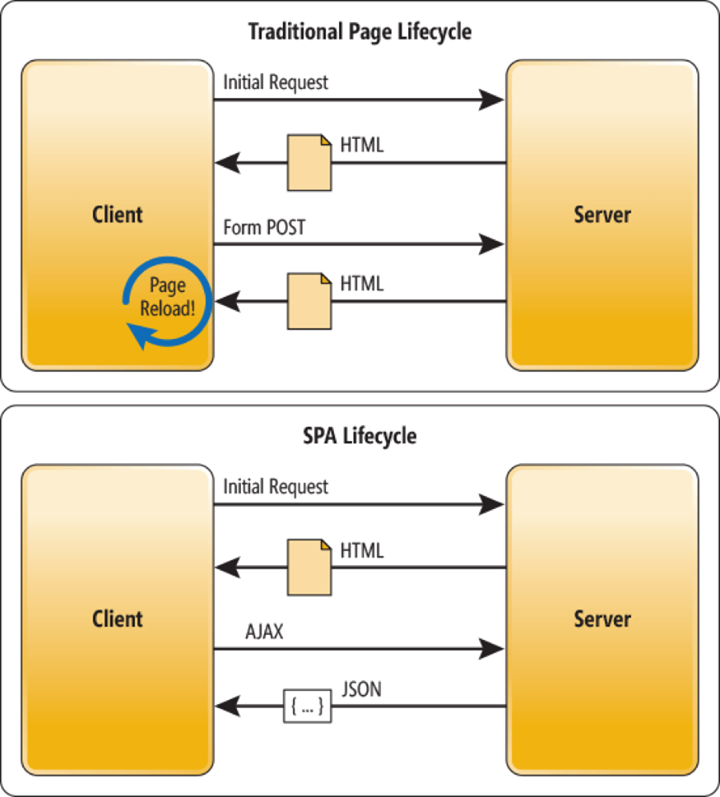


Figure 1Mô hình hoạt động của single page application và website truyển thống

* Như hình minh họa ta có thể thấy: Single page application là một trang web hay một ứng dụng web, mà tất cả những thao tác xử lý của trang web đều được diễn ra trên một trang duy nhất. Các lần sau client sẽ gọi lên và chỉ lấy dữ liệu về để render lại dưới client.
* Việc render html ở server là một điều đáng quan tâm nếu trang web của bạn có nhiều người dùng, cực kì tốn tài nguyên hệ thống
* SPA tách biệt phần front-end và back-end ra, SPA giao tiếp với server chủ yếu qua JSON Rest API giúp cho dữ liệu gửi và trả giữa client và server được giảm đến mức tối thiểu. Việc phát triển, kiểm thử cũng có thể độc lập giữa front-end và back-end.
* SPA rất nhanh, vì các tài nguyên tĩnh (static resources) như HTML, CSS, Script chỉ được tải 1 lần duy nhất. Trong suốt quá trình sử dụng, chỉ có dữ liệu là được chuyển giao qua lại giữa client với server.

**PHẦN VI : TÀI LIỆU THAM KHẢO**