OPENGIS

Giới thiệu
Angular và NgModules

Đỗ Thành Long

dtlong@opengis.vn

OPENGIS

Discover the world, Learn with maps

https://opengis.vn

OPENGIS

1. Mục tiêu

Angular là nền tảng chính của Ionic, giúp tổ chức ứng dụng hiệu quả

- Hiểu vai trò của Angular trong Ionic.
- Nắm khái niệm Component, Module, Service.
- Hiểu cấu trúc và cách dùng NgModules trong Ionic.

OPENGIS

2. Lộ trình

Angular là nền tảng chính của Ionic, giúp tổ chức ứng dụng hiệu quả

- Buổi 3: JavaScript, Overview Ionic.
- Buổi 4: Giới thiệu TypeScript (hôm nay).
- **Buổi 5: Giới thiệu Angular và NgModules.**

OPENGIS

3. Angular là gì ?

ANGULAR

- **Angular** là **framework JavaScript** mã nguồn mở, do **Google** phát triển (2016).
- Dùng để xây dựng ứng dụng web động, đơn trang (**SPA**).

Google

open source

OPENGIS

3. Angular là gì ?

ANGULAR

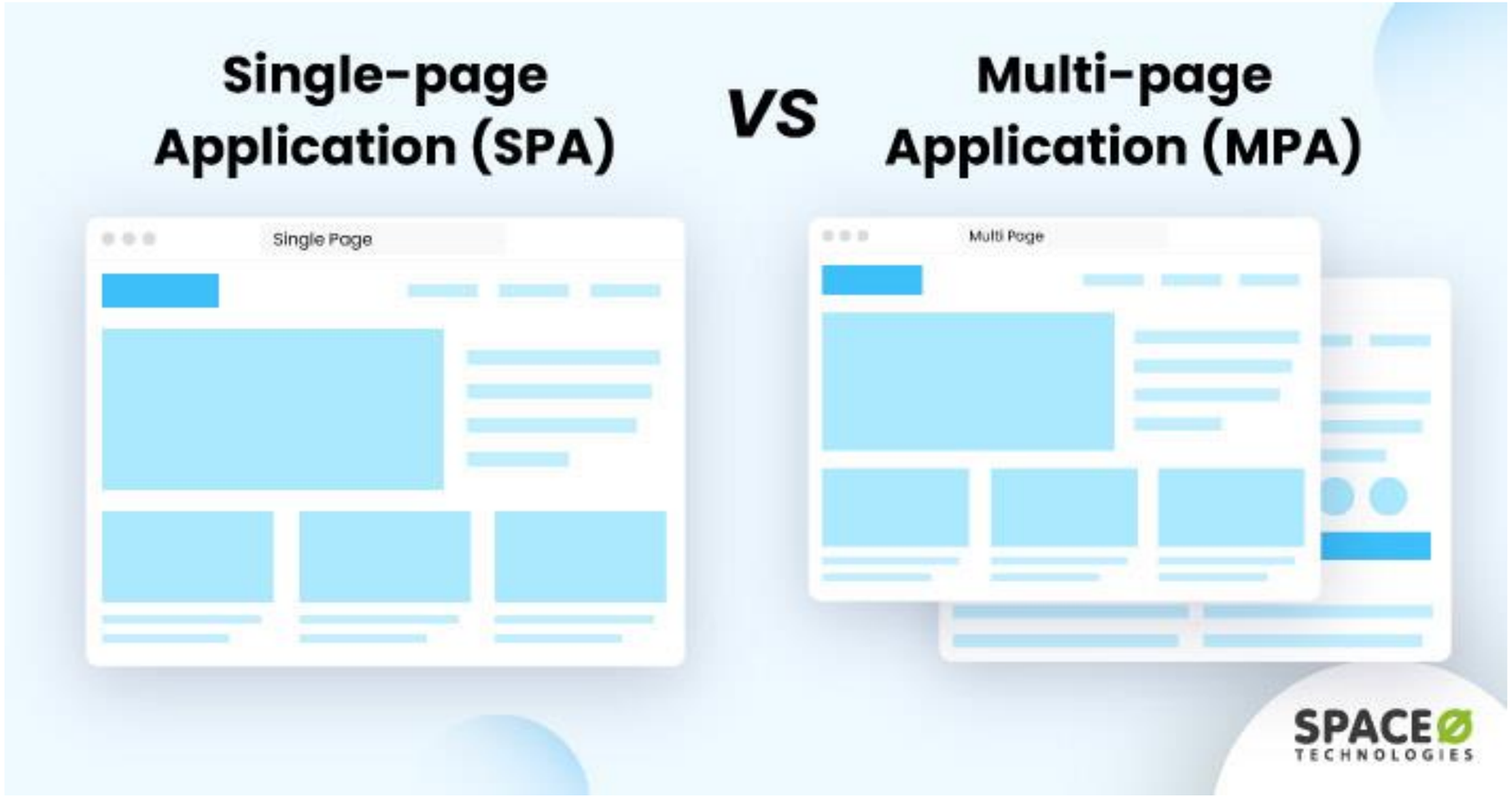
- Dựa trên **Component-based** architecture: chia ứng dụng thành các thành phần nhỏ.
- Hỗ trợ **two-way data binding**: đồng bộ dữ liệu giữa giao diện và logic.

Google

open source

OPENGIS

3. Web SPA vs MPA



OPENGIS

3.1. Multi Page Application là gì?

Multi page application

- Multi Page Application (**MPA**) là:
  - **Web truyền thống**, trong đó mỗi lần người dùng thực hiện thao tác như truy cập liên kết, gửi biểu mẫu hoặc yêu cầu dữ liệu mới, trình duyệt sẽ **tải lại toàn bộ trang** từ máy chủ và hiển thị nội dung mới

OPENGIS

3.1. Ưu điểm của Multi Page Application

Multi page application

- **Tính linh hoạt**: nhiều trang riêng biệt, dễ dàng quản lý và phát triển mỗi trang một cách độc lập.
- **Tốt cho SEO**: tối ưu hóa riêng để cải thiện hiệu suất SEO, dễ index.
- **Tính tương thích**: Hoạt động tốt trên mọi loại trình duyệt và thiết bị.
- **Bảo mật**: giảm nguy cơ tấn công XSS (Cross-Site Scripting).
- **Hiệu suất**: Có thể tối ưu hóa tốc độ tải trang bằng cách áp dụng Ajax.

OPENGIS

3.1. Hạn chế của Multi Page Application

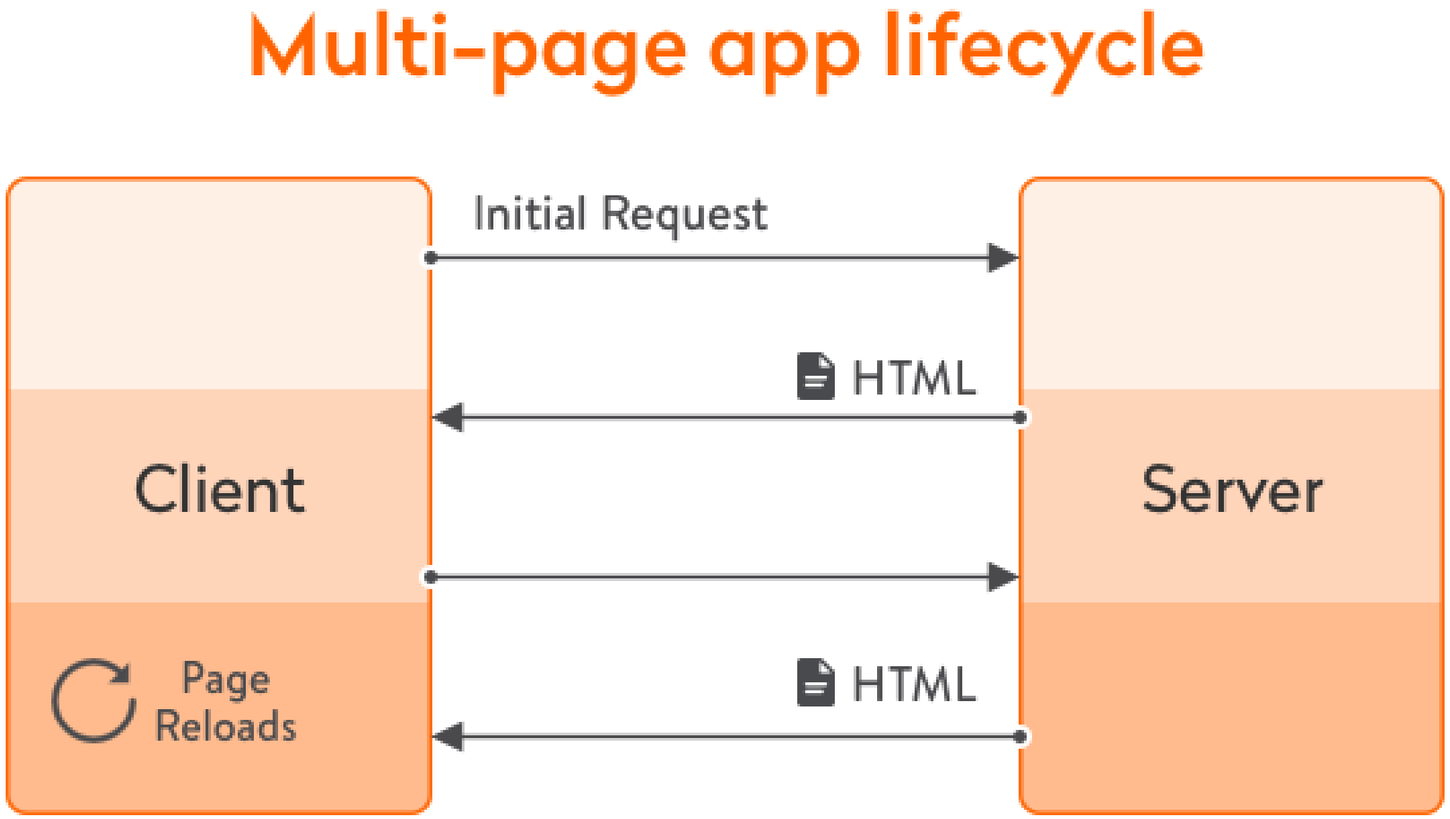
Multi page application

- **Tốc độ chuyển đổi trang chậm**: cần tải lại toàn bộ trang khi chuyển đổi giữa các trang
- **Tương tác người dung thấp**: Do mỗi trang hoạt động độc lập, việc tương tác người dùng giữa các trang có thể không linh hoạt.
- **Quản lý trạng thái**: Việc quản lý trạng thái ứng dụng trong **MPA** có thể phức tạp hơn so với **SPA** vì cần theo dõi và duy trì trạng thái của từng trang riêng lẻ.
- **Hiệu suất**: tốn nhiều băng thông hơn do cần tải lại toàn bộ trang mỗi khi người dùng chuyển đổi, đặc biệt trong trường hợp có nhiều trang và nhiều dữ liệu cần tải.

OPENGIS

3.1. Kiến trúc của Multi Page Application

Multi page application



OPENGIS

3.2. Single Page Application là gì?

Single page application

- Single Page Application (**SPA**) là loại ứng dụng web:
  - Hỗ trợ tương tác và **không yêu cầu người dùng tải lại trang** mỗi khi truy cập.
  - Nền tảng sẽ **tải ứng dụng** thông qua một quá trình duy nhất khi người dùng mới **truy cập lần đầu**.
  - **SPA** thường sử dụng các kỹ thuật như **AJAX** để tải dữ liệu và cập nhật giao diện người dùng mà **không cần tải lại** toàn bộ trang web.

OPENGIS

3.2. Ưu điểm của Single Page Application

Single page application

- **Trải nghiệm người dùng cao**
- **Hiệu suất cao**
- **Quản lý trạng thái ứng dụng hiệu quả**: SPA sử dụng các framework như Angular, React, Vue.js để quản lý trạng thái ứng dụng.
- **Tích hợp tốt với API**
- **Phát triển dễ dàng**: Tổ chức theo module, linh hoạt và tái sử dụng code dễ dàng.
- **Khả năng tùy chỉnh cao**: Do tính module và tái sử dụng code, SPA cho phép cấu hình và tùy chỉnh ứng dụng theo nhu cầu cụ thể của dự án.

OPENGIS

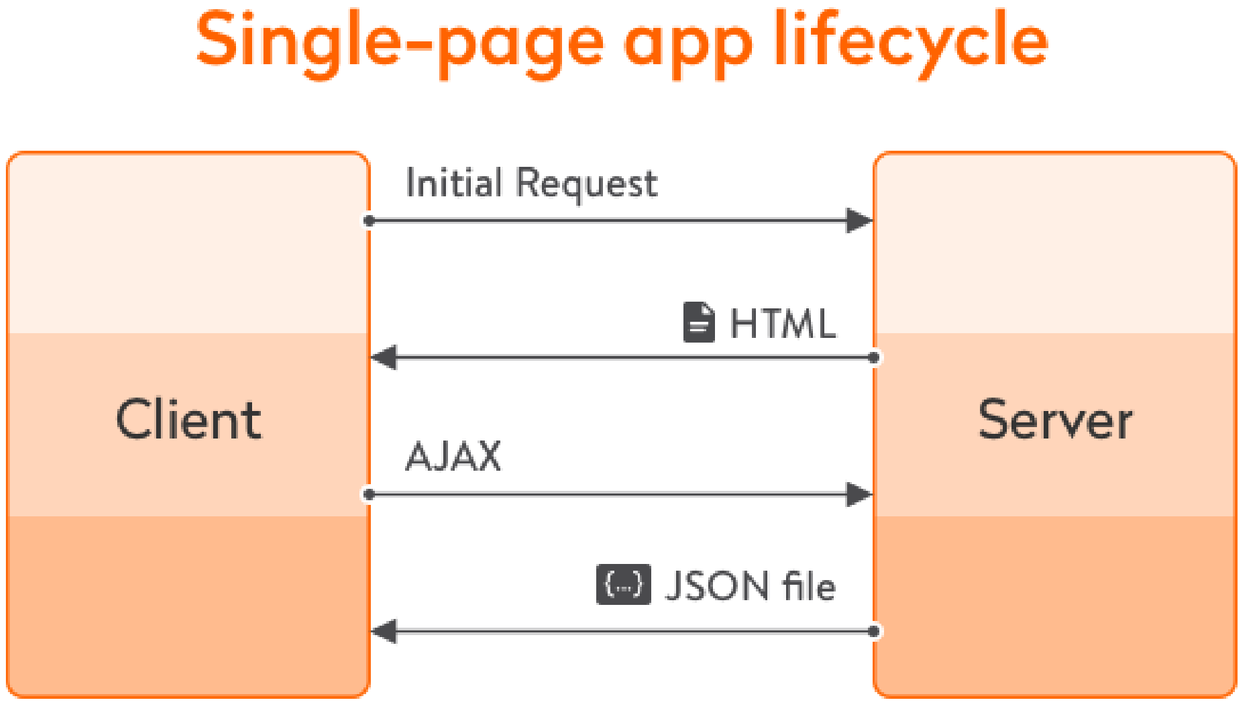
3.2. Hạn chế của Single Page Application

Single page application

- **Tải lần đầu chậm**: Do tải hết nội dung 1 lần
- **Tăng tải cho máy chủ**
- **SEO**: Hạn chế khả năng index.
- **Bảo mật:** Dễ bị Cross-Site Scripting (XSS)
- **Quản lý bộ nhớ:** Có thể tiêu tốn nhiều bộ nhớ trên trình duyệt.
- **Hạn chế các thiết bị cũ.**
- **Nguy cơ lọt lộ dữ diệu** qua API
- **Trình duyệt phải bật Javascript**

3.2. Kiến trúc của Single Page Application

Single page application



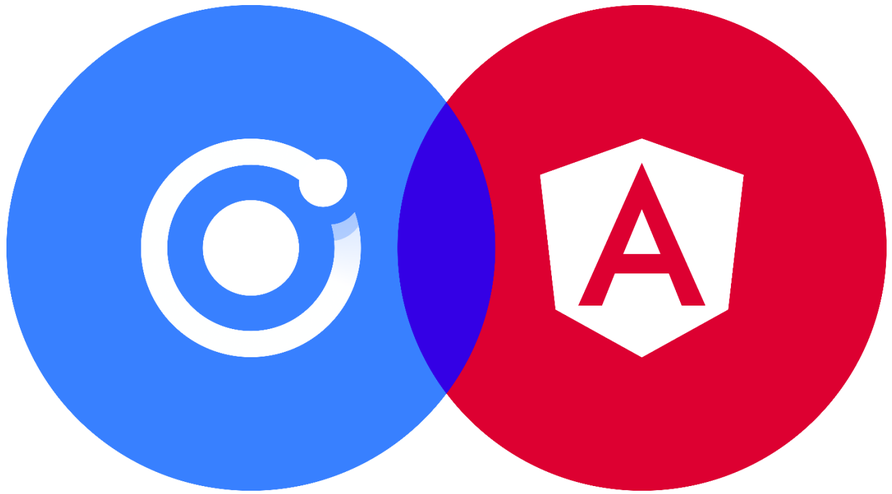
3.3. Các trạng thái chính trong ứng dụng

Single page application

- **Trạng thái UI (User Interface):** Đây là trạng thái liên quan đến giao diện người dùng, bao gồm các thành phần UI như các thành phần hiển thị, form nhập liệu, thông báo, v.v. Quản lý trạng thái UI giúp ứng dụng hiển thị đúng thông tin và tương tác một cách mượt mà.
- **Trạng thái Dữ liệu**: Đây là trạng thái liên quan đến dữ liệu được hiển thị và xử lý trong ứng dụng. Quản lý trạng thái dữ liệu giúp đảm bảo rằng dữ liệu được đồng bộ và cập nhật đúng cách trên toàn bộ ứng dụng.
- **Trạng thái Hành vi (Behavior):** Đây là trạng thái liên quan đến cách mà ứng dụng phản ứng với hành động của người dùng, sự kiện và tương tác khác. Quản lý trạng thái hành vi giúp ứng dụng đáp ứng nhanh chóng và chính xác với hành động của người dùng.

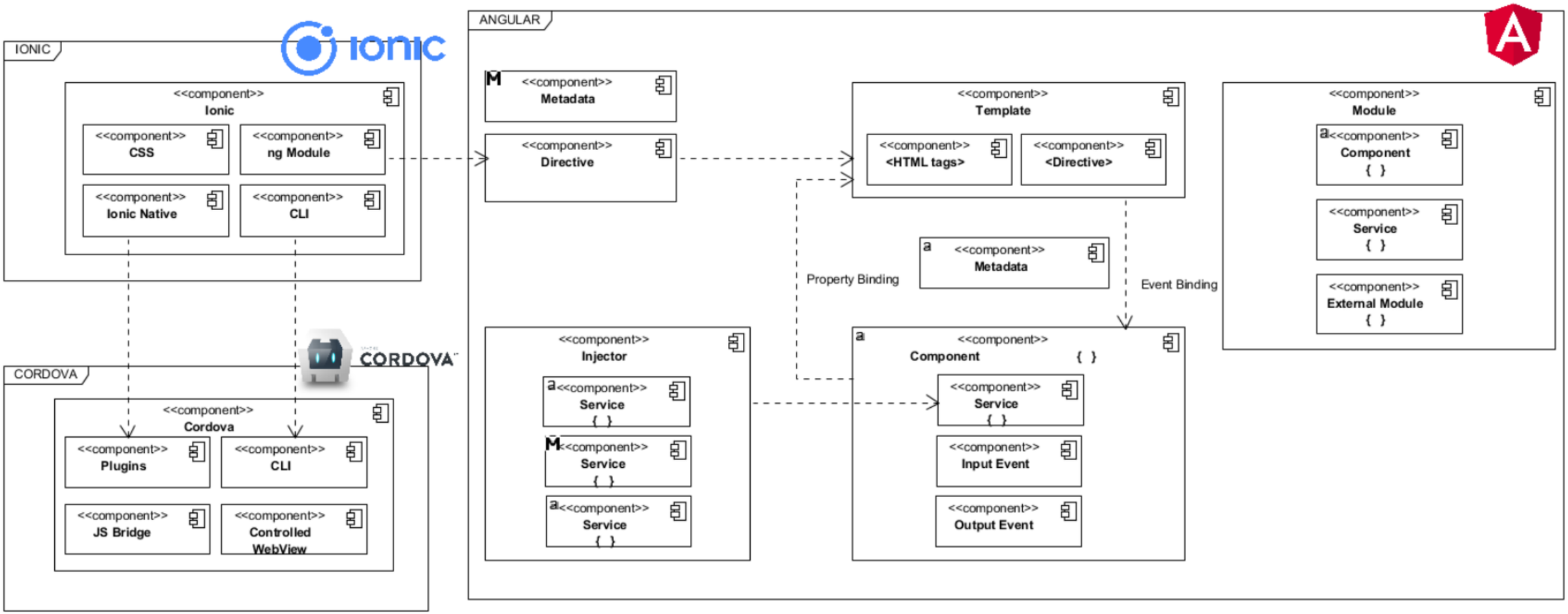
OPENGIS

4. Vai trò của Angular trong Ionic



- **Cấu trúc và tổ chức ứng dụng**: Angular giúp xác định cấu trúc của ứng dụng, quản lý các thành phần, modules và services một cách có tổ chức.
- **Quản lý trạng thái ứng dụng**: Angular hỗ trợ quản lý trạng thái ứng dụng thông qua các công cụ như RxJS, nơi bạn có thể theo dõi và xử lý trạng thái ứng dụng một cách hiệu quả.
- **Xử lý logic và dữ liệu**: Angular cung cấp các công cụ mạnh mẽ để xử lý logic ứng dụng và quản lý dữ liệu, giúp ứng dụng hoạt động mượt mà và nhất quán.
- **Tích hợp với Ionic components**: Angular là framework chính thức được Ionic hỗ trợ, giúp kết hợp dễ dàng với các thành phần giao diện được cung cấp sẵn trong Ionic để tạo ra giao diện đẹp và tính tương tác cao.

4. Vai trò của Angular trong Ionic



https://www.researchgate.net/publication/337326821_Does_the_Migration_of_Cross-Platform_Apps_Towards_the_Android_Platform_Matter_An_Approach_and_a_User_Study

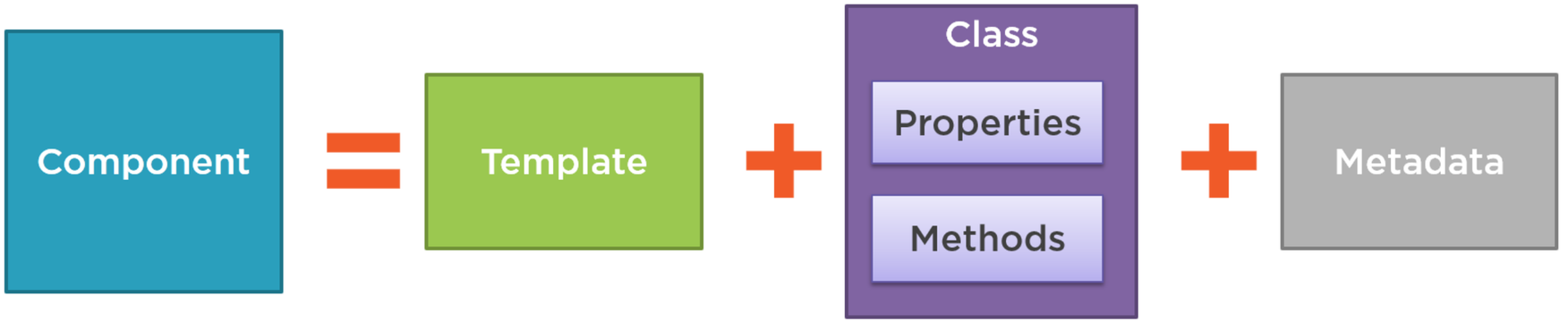
OPENGIS

5. Khái niệm Component

- Là khối xây dựng cơ bản của **Angular**, **đại diện cho một phần giao diện** (VD: header, footer, map). Bao gồm:
  - **Selector**: Định danh duy nhất của Component
  - **Template** (HTML): Giao diện.
  - **Class** (TypeScript): Logic xử lý dữ liệu và tương tác với Component.
  - **Styles** (CSS): Kiểu dáng cho giao diện.

OPENGIS

5. Khái niệm Component



- View layout
- Created with HTML
- Includes binding and directives

- Code supporting the view
- Created with TypeScript
- Properties: data
- Methods: logic

- Extra data for Angular
- Defined with a decorator

OPENGIS

5. Khái niệm Component

- **Cú pháp khai báo:**

typescript

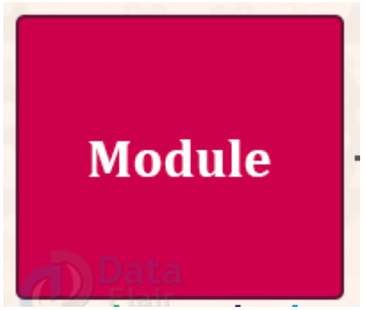
```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-map',
  templateUrl: './map.component.html',
  styleUrls: ['./map.component.css']
})
export class MapComponent {
  title: string = 'Bản đồ GIS';
}
```

Sử dụng: `<app-map></app-map>` trong HTML.

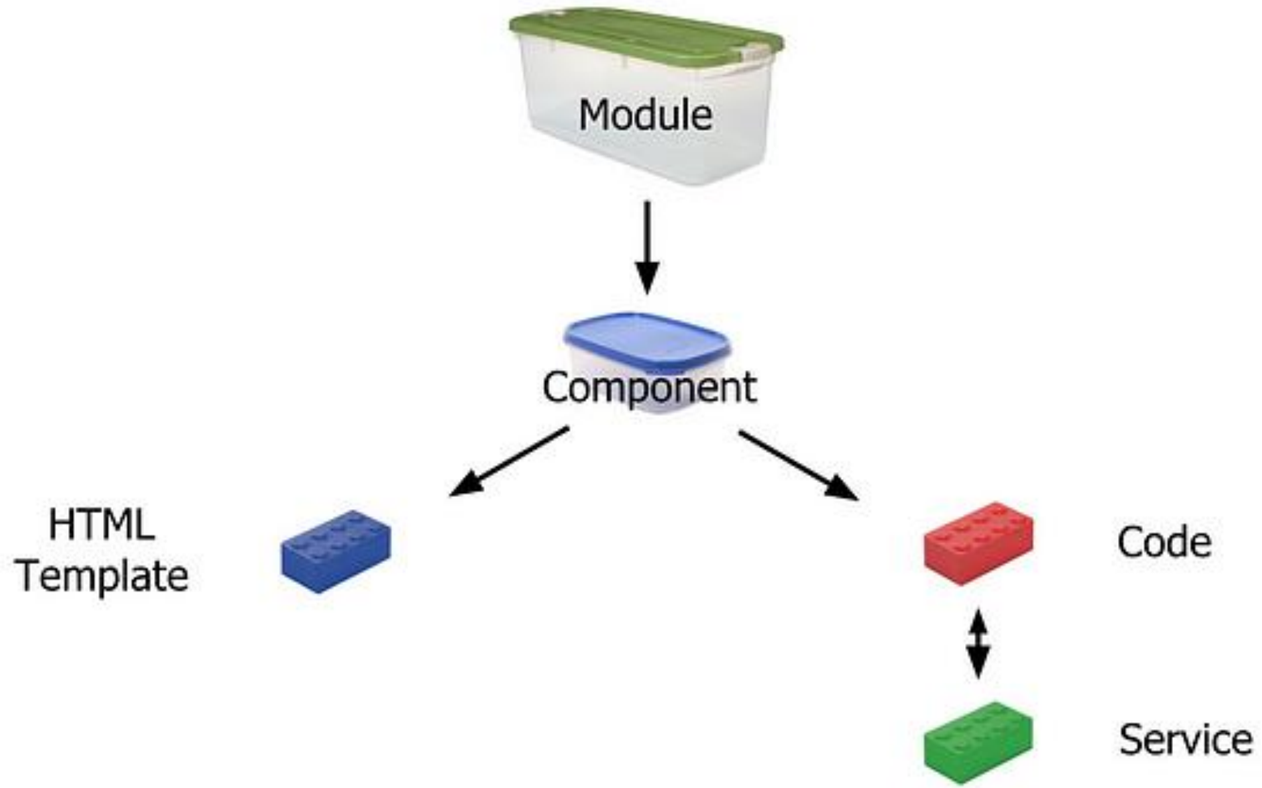
OPENGIS

6. Khái niệm Module



- Phần quan trọng của cấu trúc ứng dụng, tổ chức và quản lý các thành phần của ứng dụng. Tách biệt các phần cốt lõi (core modules) và các phần chức năng (feature modules), bao gồm:
  - **Components**
  - **Directives**: để thay đổi cách các thành phần giao diện hoạt động
  - **Pipes**: xử lý và hiển thị dữ liệu
  - **Services**: xử lý logic hoặc tương tác với dữ liệu
  - **Configuration**: thiết lập cấu hình: routing configuration, providers, imports, exports, etc.

6. Khái niệm Module



https://medium.com/@dulanjayasandaruwan1998/angular-modules-and-components-7c3371198b99

OPENGIS

7. Khái niệm Service

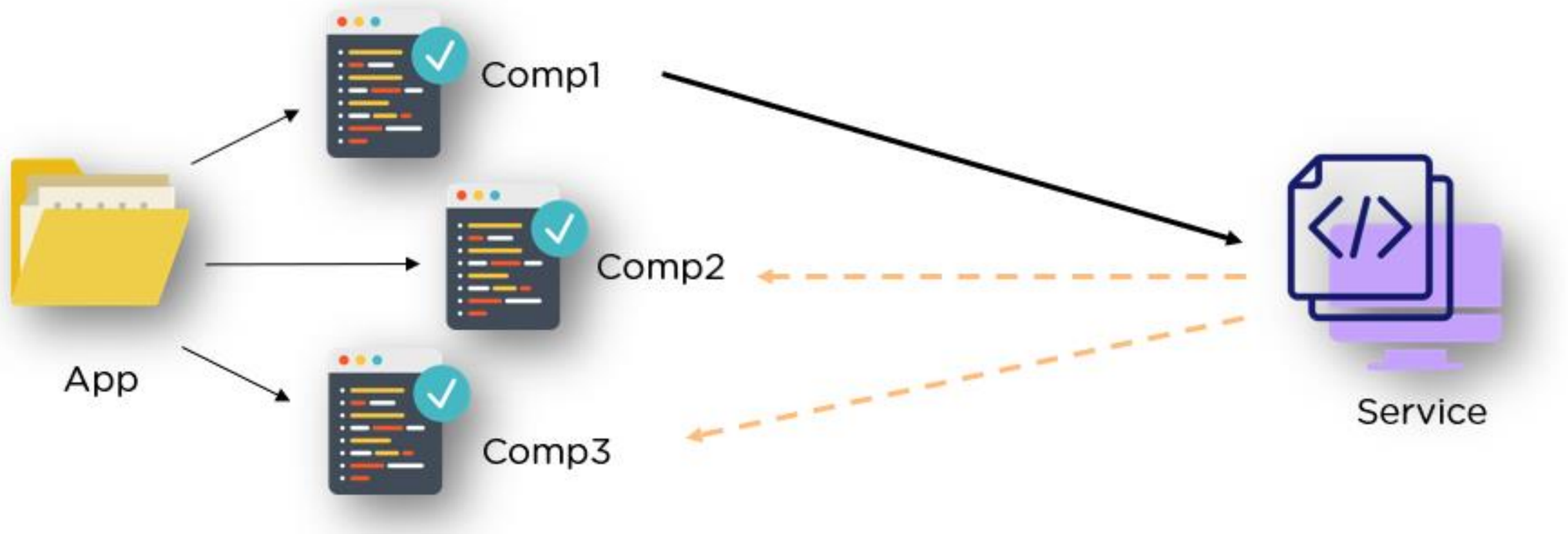
- Là lớp **cung cấp chức năng chung** (VD: gọi API, chia sẻ dữ liệu).
- Được **inject** vào **component** khi cần. Ví dụ:
  - Service "ApiService" gọi API GIS.
  - Service "LocationService" lấy vị trí người dùng.

OPENGIS

7. Khái niệm Service

- **Xử lý logic**: xử lý dữ liệu và logic phức tạp mà không liên quan trực tiếp đến giao diện người dùng.
- **Giao tiếp với server**: thường sử dụng để tương tác với các API để lấy và gửi dữ liệu.
- **Chia sẻ dữ liệu**: sử dụng để lưu trữ và chia sẻ dữ liệu giữa các Component trong ứng dụng.
- **Singleton**: Mỗi Service trong Angular là singleton, có nghĩa rằng chỉ có một phiên bản duy nhất của Service đó trong ứng dụng.

7. Khái niệm Service



https://www.simplilearn.com/tutorials/angular-tutorial/angular-service

OPENGIS

8. NgModules là gì?

@NgModule

- NgModules:
  - Là cách Angular quản lý các module, khai báo bằng **@NgModule**.
- Cấu trúc:
  - **declarations**: Component, directive, pipe.
  - imports: Module khác.
  - **providers**: Service.
  - **bootstrap**: Component khởi động.

OPENGIS

8. NgModules là gì?

@NgModule

```typescript
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [ApiService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

OPENGIS

9. NgModules trong IONIC

@NgModule

- Trong Ionic, cũng giống như trong Angular, NgModule đóng vai trò quan trọng trong việc tổ chức và quản lý các thành phần của ứng dụng.
- Tuy nhiên, trong Ionic, các NgModule được sử dụng để tổ chức các thành phần của ứng dụng dưới dạng **Ionic Module** thay vì Angular Module.

IONIC

9. NgModules trong IONIC

@NgModule

- Mỗi ứng dụng Ionic có ít nhất một NgModule (thường là AppModule).
- **Page trong Ionic là component** trong NgModule.

OPENGIS

9. NgModules trong IONIC

@NgModule

ionic

master Mobile-GIS-Programing-Course / lythuyet / buoi4 / test1 / src / app / home / **home.module.ts**

soiqualang update

Code Blame 19 lines (16 loc) · 457 Bytes

```
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { IonicModule } from '@ionic/angular';
4   import { FormsModule } from '@angular/forms';
5   import { HomePage } from './home.page';
6
7   import { HomePageRoutingModule } from './home-routing.module';
8
9
10  @NgModule({
11    imports: [
12      CommonModule,
13      FormsModule,
14      IonicModule,
15      HomePageRoutingModule
16    ],
17    declarations: [HomePage]
18  })
19  export class HomePageModule {}
```

OPENGIS

9. NgModules trong IONIC

@NgModule

master Mobile-GIS-Programing-Course / lythuyet / buoi4 / test1 / src / app / home / **home.page.ts**

soiqualang update

Code | Blame | 19 lines (15 loc) · 307 Bytes

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  standalone: false,
})
export class HomePage {

  constructor() {}

  a: number = 10;
  b: number = 42;

  calc_dientich() {
    return this.a * this.b;
  }
}
```

OPENGIS

THANK YOU