**PV ANGULAR**

# 1.Các thành phần angular?

Angular là một framework phát triển ứng dụng web bằng ngôn ngữ JavaScript. Angular bao gồm các thành phần chính sau:

1. **Modules (\*)**: một project gồm nhiều module.Được sử dụng để tổ chức và quản lý các thành phần của ứng dụng.Mỗi module là 1 component
2. **Components (\*)**: là thành phần quan trọng nhất của ứng dụng Angular được sử dụng để tạo các phần tử giao diện của ứng dụng. Mỗi chức năng sẽ là 1 component.Các component được hỗ trợ nhờ vào Directive.
3. **Directives (\*)**: Là các lệnh có thể được thêm vào các phần tử HTML để mở rộng chức năng của chúng.(\*ngIf, \*ngFor…)
4. **Templates (\*)**: Là các file HTML được sử dụng để hiển thị nội dung của các components, dữ liệu hiển thị ở template được lấy từ service.
5. **Services (\*)**: Là các thành phần cung cấp các chức năng cho các thành phần khác trong ứng dụng.

* Kết nối angular vs BE, lấy dữ liệu từ BE lên để xử lý nghiệp vụ.
* Được sử dụng trong các Component thông qua DI

1. **Dependency Injection**: Là một cơ chế được sử dụng để cung cấp các đối tượng phụ thuộc cho các thành phần khác trong ứng dụng.
2. **Routing**: Là cơ chế điều hướng giúp chuyển hướng trang trong ứng dụng.
3. **Pipes**: Là các thành phần giúp chuyển đổi hoặc định dạng dữ liệu hiển thị trong các templates.
4. **Forms**: Là các thành phần để tạo và quản lý các form trong ứng dụng.

Tất cả các thành phần này đóng vai trò quan trọng trong việc phát triển ứng dụng web bằng Angular.

# 2. Khái niệm Component trong Angular?

* Trong Angular, Component là một đối tượng JavaScript dùng để xây dựng giao diện người dùng (UI). Là thành phần cơ bản và quan trọng nhất trong angular (mỗi componetnt là 1 chức năng)
* Component là một phần tử độc lập, có thể tái sử dụng và có chứa HTML, CSS và mã JavaScript để xử lý sự kiện và thao tác với dữ liệu => tạo ra một phần giao diện hoàn chỉnh
* Một Component có thể chứa các thành phần khác như Directive, Pipe, Service, hoặc nó có thể được sử dụng bởi các Component khác để xây dựng một ứng dụng hoàn chỉnh.
* Mỗi Component có một tên định danh duy nhất và được gọi bởi các Component khác hoặc bởi router của ứng dụng để hiển thị trên trình duyệt.
* Component bao gồm:
* Template (.html): hiển thị layout đến với người dùng
* Class (.ts): nhận dữ liệu từ service => xử lý nghiệp vụ,logic => đưa kết quả lên cho người dùng qua template.
* Metadata (siêu dữ liệu): dùng cấu hình 1 component (seletor, templateUrl…),
* =>Định nghĩa các thuộc tính, sự kiện và các khai báo khác.

Việc sử dụng Component trong Angular giúp tách biệt các phần của ứng dụng và giúp quản lý code dễ dàng hơn, đồng thời cũng giúp tăng tính tái sử dụng và bảo trì ứng dụng.

# 3. (Trending) So sánh directive với component?

1. Directive và Component là hai khái niệm quan trọng trong Angular Framework. Chúng đều được sử dụng để tạo ra các thành phần UI trong ứng dụng Angular.Chúng có liên quan đến nhau.
2. Directive được sử dụng để thay đổi cách thức hiển thị và hoạt động của các thành phần HTML trong ứng dụng, trong khi Component là một directive đặc biệt với một template, một controller và một scope được sử dụng để tạo các phần tử giao diện độc lập và tái sử dụng trong ứng dụng.
3. Một số điểm khác nhau giữa directive và component trong Angular bao gồm:

* Template: Component có thể có một template, trong khi directive không có.
* Selector: Component được định nghĩa với một selector, còn directive không nhất thiết phải có.
* Scope: Component có một scope riêng, trong khi directive không.
* Lifecycle hooks: Component có thể sử dụng các lifecycle hooks như ngOnInit(), ngOnDestroy(), vv, trong khi directive không.
* View encapsulation: Component có khả năng bảo vệ view encapsulation, trong khi directive không.

# 4.Rxjs trong angular?

1. RxJS là một thư viện reactive programming được sử dụng trong Angular để xử lý các tác vụ bất đồng bộ, như xử lý các sự kiện người dùng, gọi các API từ server, xử lý các đường dẫn của Router, v.v. RxJS cung cấp các công cụ để xử lý dữ liệu theo kiểu Reactive Streams, giúp cho việc xử lý dữ liệu trở nên dễ dàng và hiệu quả hơn.
2. RxJS cung cấp các khái niệm quan trọng như Observables, Operators và Subscriptions.

* Các Observables là các chuỗi dữ liệu, event có thể sản sinh các giá trị liên tục trong tương lai.
* Operators được sử dụng để xử lý dữ liệu được phát ra từ các Observables, bao gồm các phương thức map, filter, merge, switchMap, debounceTime, throttleTime….
* Subscriptions là cơ chế đăng ký để theo dõi các thay đổi trong Observables và nhận các giá trị mới.

1. RxJS cung cấp cho Angular một số thành phần quan trọng như BehaviorSubject, ReplaySubject, Subject, AsyncSubject, giúp cho việc xử lý sự kiện, dữ liệu trở nên đa dạng và phong phú hơn.
2. Sử dụng RxJS trong Angular giúp cho việc xử lý dữ liệu bất đồng bộ trở nên hiệu quả và dễ dàng hơn, giúp cho việc lập trình trở nên đơn giản hơn, code dễ đọc hơn và dễ bảo trì hơn.

# 5. Khi nào ở phía controller trả về json. Angular liên kết với BackEnd như thế nào?

Angular kết nối với Backend thông qua các API được cung cấp bởi Backend, sử dụng HTTP Service hoặc HttpClient để gửi và nhận dữ liệu và hiển thị dữ liệu này trên các Component và template trong ứng dụng.

* **HTTP Service:** Là một service của Angular để tạo các yêu cầu HTTP đến Backend, nhận và xử lý dữ liệu trả về. Sử dụng HTTP Service, Angular có thể gửi các yêu cầu GET, POST, PUT, DELETE, PATCH đến Backend và xử lý dữ liệu trả về dưới dạng JSON.
* **HttpClient:** Là một module của Angular để xử lý các yêu cầu HTTP đến Backend, có khả năng xử lý các yêu cầu đồng bộ hoặc bất đồng bộ. HttpClient hỗ trợ việc đọc và ghi dữ liệu với nhiều định dạng khác nhau như JSON, XML, Text, ArrayBuffer, Blob.

# 6. Sử dụng Angular 11 để làm những gì?

Angular 11 là một trong những phiên bản của Angular, nó được sử dụng để phát triển các ứng dụng web động và đa nền tảng. Dưới đây là một số công việc mà bạn có thể thực hiện bằng Angular 11:

* Xây dựng ứng dụng web động: Angular 11 là một framework front-end mạnh mẽ, cho phép bạn tạo ra các ứng dụng web động với các tính năng như hiệu ứng hoạt hình, tương tác người dùng và tích hợp các công nghệ mới nhất.
* Phát triển ứng dụng đa nền tảng: Angular 11 cung cấp khả năng phát triển các ứng dụng cho nhiều nền tảng, bao gồm web, mobile và desktop.
* Tích hợp API của Backend: Angular 11 cho phép bạn kết nối và tương tác với các API của Backend thông qua HTTP Service hoặc HttpClient.
* Quản lý dữ liệu: Angular 11 cung cấp nhiều công cụ để quản lý dữ liệu trong ứng dụng, bao gồm Reactive Forms, Reactive Programming và RxJS.
* Phát triển ứng dụng Single-Page Application (SPA): Angular 11 hỗ trợ việc phát triển các ứng dụng Single-Page Application (SPA) với các tính năng như Routing, Lazy Loading và Service Worker.
* Xây dựng ứng dụng real-time: Angular 11 cho phép bạn xây dựng các ứng dụng real-time sử dụng WebSockets hoặc Server-Sent Events.
* Tối ưu hóa hiệu suất: Angular 11 cung cấp nhiều công cụ để tối ưu hóa hiệu suất của ứng dụng, bao gồm Ahead-of-Time (AOT) compilation, Tree Shaking và Code Splitting.

# 7. Cách hoạt động của Angular 11? Angular lưu trữ và xử lý dữ liệu như thế nào?

Angular 11 là một framework front-end, hoạt động dựa trên các khái niệm của MVC (Model-View-Controller) và MVVM (Model-View-ViewModel). Các thành phần chính của Angular 11 bao gồm Component, Service, Directive và Module.

* Khi người dùng tương tác với ứng dụng Angular 11, các sự kiện (events) sẽ được truyền đến Component, là thành phần chịu trách nhiệm cho việc hiển thị dữ liệu trên giao diện. Component sẽ gọi đến các Service, là thành phần chịu trách nhiệm xử lý logic, để truy xuất và xử lý dữ liệu từ các nguồn khác nhau như API của Backend hoặc Local Storage. Sau đó, Service sẽ trả về dữ liệu cho Component và Component sẽ cập nhật giao diện dựa trên dữ liệu đó.
* Angular 11 sử dụng RxJS, một thư viện Reactive Programming, để quản lý các luồng dữ liệu trong ứng dụng. RxJS cung cấp các công cụ để quản lý và xử lý các sự kiện và dữ liệu theo cách có tính tương tác cao. Các Observable trong RxJS là các luồng dữ liệu có thể được theo dõi và xử lý theo thời gian thực.
* Angular 11 cũng hỗ trợ Reactive Forms, là một phần của Angular Forms, để quản lý dữ liệu đầu vào từ người dùng. Reactive Forms cho phép các Form Control (điều khiển biểu mẫu) được kết nối với các Observable, đảm bảo rằng các thay đổi trong Form Control sẽ được theo dõi và xử lý theo thời gian thực.
* Angular 11 cũng hỗ trợ Service Worker, cho phép ứng dụng có thể hoạt động offline và cải thiện tốc độ tải trang. Service Worker là một script chạy ở background của trình duyệt, cho phép ứng dụng cache các tài nguyên và dữ liệu để sử dụng khi không có kết nối mạng.
* Angular 11 sử dụng các ứng dụng Single-Page Application (SPA) để giảm tải trang và cải thiện trải nghiệm người dùng. Với kiến trúc mạnh mẽ và tính linh hoạt

Về cơ bản, Angular 11 lưu trữ và xử lý dữ liệu bằng cách sử dụng các thành phần như Component, Service và Module, và sử dụng RxJS để quản lý các luồng dữ liệu và sự kiện.

# 8. Sự khác nhau MVC (Model-View-Controller) và MVVM (Model-View-ViewModel) trong angular?

MVC (Model-View-Controller) và MVVM (Model-View-ViewModel) là hai mô hình kiến trúc phần mềm phổ biến trong phát triển phần mềm. Tuy nhiên, nếu muốn sử dụng mô hình MVVM trong Angular, có thể sử dụng thư viện hoặc framework hỗ trợ MVVM như KnockoutJS hoặc VueJS.

1. MVC gồm ba thành phần chính:

* Model (M): đại diện cho dữ liệu và logic xử lý dữ liệu. Model được đặt tại các Service để xử lý logic và tương tác với các API hoặc database. Các Service thường được đặt trong các file có đuôi .service.ts.
* View (V): đại diện cho giao diện người dùng. View được định nghĩa trong các Component.Là các file có đuôi .html
* Controller (C): là các file component.ts. Quản lý logic điều khiển và xử lý sự kiện. Các Component cũng có thể sử dụng các Service để tương tác với Model.
* Các thành phần này độc lập với nhau và phối hợp với nhau thông qua các cơ chế truyền thông để tạo ra ứng dụng.

1. MVVM gồm ba thành phần:

* Model (M): Model trong Angular được đặt tại các Service để xử lý logic và tương tác với các API hoặc database. Các Service thường được đặt trong các file có đuôi .service.ts.
* View (V): đại diện cho giao diện người dùng. View được định nghĩa trong các Component.Là các file có đuôi .html
* ViewModel (VM): ViewModel đại diện cho trung gian giữa Model và View. ViewModel sử dụng data binding để liên kết dữ liệu giữa Model và View, giúp tự động cập nhật dữ liệu khi có sự thay đổi. ViewModel trong Angular được triển khai thông qua một kết hợp giữa Component và Service. ViewModel cũng quản lý các thuộc tính dữ liệu được hiển thị trên View.

# 9.Two-way-binding trong Angular?

* Two-way binding là một tính năng quan trọng của Angular, cho phép bạn đồng bộ hóa dữ liệu giữa view và model.
* Khi dữ liệu trong view thay đổi, model sẽ được cập nhật tự động, và ngược lại khi model thay đổi, view cũng sẽ được cập nhật.Top of Form
* Để sử dụng two-way binding, bạn cần import FormsModule vào module và sử dụng [(ngModel)] directive (là sự kết hợp giữa liên kết thuộc tính và sự kiện )với một biến để đồng bộ hóa dữ liệu giữa view và model.

# 10.Trong Angular binding dữ liệu như thế nào ?

* Data binding là một kỹ thuật trong Angular giúp cho dữ liệu giữa component và view được đồng bộ hóa.
* Khi người dùng cập nhật dữ liệu trên view, Angular sẽ cập nhật lại component.
* Khi component có dữ liệu mới, Angular sẽ cập nhật lại view.

**Có 2 loại binding trong angular: one-way-binding và two-way-binding.**

* One-way data binding là kỹ thuật binding dữ liệu một chiều từ component tới view hoặc ngược lại. Khi dữ liệu thay đổi ở component, nó sẽ được cập nhật tới view (Interpolation, Property Binding). Khi dữ liệu thay đổi ở view có thể được cập nhật tới component (Event Binding).
* **Interpolation** **(kiểu nội suy):** Kỹ thuật này cho phép hiển thị giá trị của một biến trong view. Ví dụ: {{variable}}.
* **Property Binding (kiểu kiên kết thuộc tính):** Kỹ thuật này cho phép binding giá trị của một thuộc tính của một phần tử HTML tới một biến trong component. Ví dụ: [property]=“variable”.
* **Event Binding (kiểu liên kết sư kiện):** Kỹ thuật này cho phép binding một sự kiện của một phần tử HTML tới một hàm trong component. Ví dụ: (event)=“function()”.
* Two-way data binding là kỹ thuật binding dữ liệu hai chiều giữa component và view. Khi dữ liệu thay đổi ở component hoặc view, nó sẽ được cập nhật tới phía còn lại.
* Để sử dụng two-way binding, bạn cần **import FormsModule** vào module và sử dụng **[(ngModel)]** directive **(là sự kết hợp giữa liên kết thuộc tính và sự kiện)** với một biến để đồng bộ hóa dữ liệu giữa view và model.

# 11.Nếu không sử dụng [(ngModel)] làm sao để binding 2 chiều ? ( Hỏi về ngShow )

1. Sử dụng cú pháp dấu ngoặc vuông cho one-way binding và dấu ngoặc đơn cho event binding để đạt được two-way binding mà không sử dụng [(ngModel)] trong Angular. Dưới đây là một ví dụ:

<input [value]="myValue" (input)="myValue = $event.target.value">

1. Sử dụng getter và setter để đạt được two-way binding. Dưới đây là một ví dụ:

private \_myValue: string;

get myValue(): string {

return this.\_myValue;}

set myValue(value: string) {

this.\_myValue = value;}

<input [value]="myValue" (input)="myValue = $event.target.value">

Trong ví dụ này, định nghĩa một biến riêng tư \_myValue và tiết lộ nó như một getter và một setter. Ràng buộc thuộc tính giá trị của phần tử input với myValue bằng cú pháp dấu ngoặc vuông và ràng buộc sự kiện input với một event handler để cập nhật \_myValue với giá trị mới của phần tử input.

# 12. Cho component A là cha của component B, component B là cha của component C, làm sao để truyền dữ liệu giữa component A và component C?

1. Sử dụng @Input và @Output decorators để truyền dữ liệu giữa các component con và cha. Bạn có thể truyền dữ liệu từ component A đến component C thông qua component B.
2. Sử dụng một service (VD:Behavior Service, Subject Service…) chia sẻ để truyền dữ liệu giữa các component con và cha. Bạn có thể tạo một service chia sẻ và inject nó vào các component con và cha để truyền dữ liệu giữa chúng.

# 13. Làm sao để kết nối Angular với Spring?

Để kết nối Angular với Spring Boot, có thể sử dụng API RESTful. Dưới đây là một số bước để làm theo:

1. Tạo dự án Spring Boot bằng Spring Initializr.
2. Thêm các phụ thuộc cần thiết cho Spring Web và Spring Data JPA.
3. Tạo API RESTful bằng Spring Web.
4. Tạo một ứng dụng Angular bằng Angular CLI.
5. Sử dụng HttpClient để thực hiện các yêu cầu HTTP tới API RESTful.

# 14.Vòng đời 1 component?

Thứ tự vòng đời 1 component: ngOnChanges – ngOnInit – ngDoCheck – ngAfterContentInit – ngAfterContentChecked – ngAfterViewInit – ngAfterViewChecked - ngOnDestroy

* Trong Angular, constructor HD đầu tiên trong component (không thuộc vòng đời component) là một phương thức khởi tạo của một class. Nó được gọi khi một instance của class được tạo ra. Constructor được sử dụng để khởi tạo các giá trị ban đầu cho các thuộc tính của class và để thực hiện các thao tác khởi tạo khác cần thiết. [Nó cũng có thể được sử dụng để inject các dependency vào class thông qua Dependency Injection (DI) của Angular1](https://ichi.pro/vi/ban-nen-dat-gi-trong-constructor-so-voi-ngoninit-trong-angular-128814076737863).
* **ngOnChanges():** là một lifecycle hook trong Angular2+ được sử dụng để phát hiện khi giá trị đầu vào của một component đã thay đổi và giá trị đầu vào trước đó của nó. Nó được gọi sau khi các thuộc tính đầu vào của component đã được khởi tạo và trước khi component được hiển thị lần đầu tiên trên màn hình.

Khi giá trị đầu vào của một component thay đổi, ngOnChanges() sẽ được gọi và truyền vào một đối tượng SimpleChanges chứa thông tin về các thuộc tính đã thay đổi. Bạn có thể sử dụng đối tượng này để xử lý các thay đổi này.

* **ngOnInit():** là một lifecycle hook trong Angular2+ được sử dụng để khởi tạo các giá trị ban đầu cho một component sau khi các thuộc tính đầu vào của nó đã được khởi tạo. Nó được gọi một lần duy nhất khi component được khởi tạo.
* **ngDoCheck():** là một lifecycle hook trong Angular2+ được sử dụng để kiểm tra các thay đổi trong component và được gọi sau khi ngOnChanges() và ngOnInit() được gọi. Nó được sử dụng để kiểm tra các thay đổi trong component và thực hiện các hành động phù hợp: ngAfterContentInit(), ngAfterContentChecked(), ngAfterViewInit(), ngAfterViewChecked()
* **ngOnDestroy():** là một trong những lifecycle hook được sử dụng để giải phóng tài nguyên không cần thiết của component trước khi component bị hủy. [Khi component bị hủy, ngOnDestroy() sẽ được gọi tự động để giải phóng tài nguyên như các subscription, các event listener hoặc các kết nối đến server](https://angular.io/api/core/OnDestroy). Nó được gọi một lần duy nhất.

# 15. Directive trong Angular là gì? Dùng để làm gì?

* Directive trong Angular là một khái niệm quan trọng, được sử dụng để tạo các thành phần giao diện người dùng tùy chỉnh và tái sử dụng trong các ứng dụng Angular.
* Các Directive có thể được sử dụng để thay đổi cách hiển thị các thành phần giao diện, cung cấp các hành vi tương tác và kiểm soát quyền truy cập vào các phần của ứng dụng
* Là các thành phần mở rộng cho thẻ HTML, dùng bổ trợ các thuộc tính nâng cao cho các thẻ HTML

Gồm 3 loại chính:

1. **Component directive .**

* Component directive trong Angular là một loại directive dùng để tạo ra các component trong Angular.
* Một Component Directive được tạo ra bằng cách sử dụng decorator @Component trong Angular. Decorator @Component chứa các metadata, bao gồm tên của directive (selector), template, style, và các thông tin khác liên quan đến component.
* Nó giúp chúng ta phân chia giao diện thành các phần nhỏ hơn để dễ quản lý và tái sử dụng hơn.
* Component Directive có thể chứa các logic code, xử lý dữ liệu, tương tác với người dùng, gọi service, hoặc tương tác với các Component Directive khác để tạo thành một ứng dụng hoàn chỉnh.

1. **Structural directive**

* Structural directive là một loại directive trong Angular, nó được sử dụng để thay đổi cấu trúc của DOM (Document Object Model) bằng cách thêm hoặc xóa các phần tử HTML.
* Trong Angular, có ba Structural directive được tích hợp sẵn, đó là \*ngIf, \*ngFor, và \*ngSwitch.
* \*ngIf: Structural directive này được sử dụng để hiển thị một phần tử HTML trên trang nếu điều kiện được đưa ra là true, và ẩn phần tử đó nếu điều kiện là false.
* \*ngFor: Structural directive này được sử dụng để lặp lại một phần tử HTML cho mỗi phần tử của một mảng.
* \*ngSwitch: Structural directive này được sử dụng để hiển thị một phần tử HTML khác nhau tùy thuộc vào giá trị của một biểu thức.Gồm \*ngSwitchCase và \*ngSwitchDefault

1. **Attribute directive** là một loại Directive trong Angular được sử dụng để thay đổi thuộc tính của một phần tử HTML hoặc để thêm hoặc xóa các thuộc tính. Attribute directive có thể được sử dụng để thêm các hành vi mới vào phần tử HTML hoặc để tùy chỉnh các thuộc tính của phần tử HTML.

# 16. @Input, @Output trong angular?

* @Input và @Output là các Decorator trong Angular được sử dụng để tạo ra các liên kết đầu vào (Input) và đầu ra (Output) giữa các Component và Directive khác nhau trong ứng dụng.
* @Input: cho phép truyền dữ liệu từ một Component cha vào một Component con thông qua một thuộc tính đầu vào. Khi thuộc tính đầu vào này thay đổi giá trị, Angular sẽ tự động cập nhật lại nội dung của Component con.
* @Output: cho phép tạo ra một sự kiện (Event) để truyền dữ liệu từ Component con vào Component cha. Khi sự kiện này được kích hoạt, Angular sẽ tự động gửi dữ liệu truyền về Component cha để xử lý.

# 17. Tại sao lại sử dụng Angular?

* **Cấu trúc mạnh mẽ:** Angular cung cấp một cấu trúc mạnh mẽ để xây dựng ứng dụng web, đặc biệt là ứng dụng phức tạp. Với các tính năng như Dependency Injection, Observables và Reactive Programming, Angular giúp giải quyết các vấn đề về quản lý trạng thái và điều khiển luồng dữ liệu.
* **Tích hợp dễ dàng:** Angular cung cấp các công cụ để tích hợp với các thư viện và công nghệ khác như Redux, Material Design, Bootstrap và GraphQL. Điều này giúp cho việc phát triển và bảo trì ứng dụng trở nên dễ dàng hơn.
* **Hỗ trợ tốt cho SEO:** Angular cung cấp một cơ chế để tạo ra các trang web có thể được hiển thị tốt trên các công cụ tìm kiếm. Bằng cách sử dụng Server-Side Rendering, Angular giúp cho các trang web được tạo ra bằng framework này có thể được index trên các công cụ tìm kiếm.
* **Cộng đồng lớn:** Với cộng đồng phát triển đông đảo, Angular luôn được cập nhật và cải tiến để đáp ứng nhu cầu của các nhà phát triển. Điều này giúp cho việc tìm kiếm tài liệu, hỗ trợ và giải quyết vấn đề trở nên dễ dàng hơn.
* **Hỗ trợ cho ứng dụng di động:** Angular cung cấp một framework để phát triển ứng dụng di động sử dụng các công nghệ như Cordova và NativeScript. Điều này giúp cho việc phát triển ứng dụng di động trở nên dễ dàng và đồng bộ hơn với ứng dụng web.

# 18. Phân biệt Promise và Observable?

**Giống:** đều là thành phần dùng để xử lý các luồng dữ liệu từ web truyền vào, xử lý bất đồng bộ

**Khác:**

1. Observable: có thể trả về nhiều giá trị theo thời gian và không kết thúc cho đến khi bị hủy hoặc hoàn thành.

Promise: trả về một giá trị duy nhất và kết thúc sau khi giá trị này được trả về.

1. Observable: Có thể cancel request, có thể retry, có thể dùng với event.

Promise: không thể.

1. Observable: chỉ khởi tạo khi có đăng ký đến observable -> xử lý bất đồng bộ tốt hơn.

Promise: Khi khai báo sẽ khởi tạo ngay dù chưa có bất kỳ đăng ký nào.

1. Observable: có các thao tác tiền xử lý DL như fillter, map…

Promise: không có.

1. Observable: có thể xử lý nhiều lần lỗi trong quá trình thực thi và trả về giá trị error cho các đăng ký tiếp theo.

Promise: chỉ xử lý được lỗi một lần trong quá trình thực thi, và khi gặp lỗi, Promise sẽ trả về giá trị reject.

1. Observable: thực thi bất đồng bộ và trả về giá trị theo thời gian.

Promise: thực thi đồng bộ và chỉ trả về giá trị khi hoàn thành.

1. Cú pháp của Promise đơn giản hơn so với Observable. Để sử dụng Observable, bạn cần import Observable và thực hiện một số đăng ký và hủy đăng ký để quản lý trạng thái của Observable.

# 19.Mô hình phát triển của Angular?

Mô hình phát triển của Angular là mô hình Component-based architecture. Điều này có nghĩa là các ứng dụng Angular được phát triển dựa trên một loạt các thành phần (component) độc lập, mỗi thành phần đại diện cho một phần nhỏ của giao diện người dùng và có thể tái sử dụng được.

# 20. Luồng hoạt động của project Angular từ lúc nhận request từ người dùng?

1. Router tiếp nhận request: Router trong Angular sẽ tiếp nhận request từ người dùng và phân tích URL để xác định đường dẫn đến trang cần hiển thị.
2. Tìm kiếm component tương ứng: Sau khi xác định được đường dẫn, Angular sẽ tìm kiếm component tương ứng để hiển thị trên trang. Component là một khối xây dựng cơ bản trong Angular, chịu trách nhiệm xây dựng nội dung của trang.
3. Khởi tạo component: Khi tìm được component phù hợp, Angular sẽ khởi tạo component đó bằng cách gọi constructor và các lifecycle hooks.
4. Tải dữ liệu: Trong quá trình khởi tạo component, Angular có thể tải dữ liệu từ các service hoặc API để hiển thị trên trang.
5. Hiển thị trang: Khi component đã được khởi tạo và dữ liệu được tải, Angular sẽ hiển thị trang cho người dùng.
6. Thực hiện các hoạt động bất đồng bộ: Trong quá trình hiển thị trang, Angular có thể thực hiện các hoạt động bất đồng bộ, chẳng hạn như tải thêm dữ liệu hoặc thực hiện các thao tác khác trên trang.
7. Lắng nghe các sự kiện: Angular có thể lắng nghe các sự kiện từ người dùng, chẳng hạn như các sự kiện click hoặc submit form.
8. Cập nhật trang: Khi có sự kiện xảy ra hoặc dữ liệu thay đổi, Angular sẽ cập nhật trang để hiển thị lại các thông tin mới nhất cho người dùng.
9. Giải phóng tài nguyên: Khi người dùng thoát khỏi trang hoặc chuyển sang trang khác, Angular sẽ giải phóng tài nguyên được sử dụng trong quá trình hiển thị trang đó, giúp tối ưu hóa hiệu suất của ứng dụng.

# 21. Một project Angular gồm những gì (Front End, Back End)

**Phía Front-end:**

Angular framework: để xây dựng các component và quản lý logic của ứng dụng

* TypeScript: để viết mã lập trình
* HTML/CSS: để xây dựng giao diện người dùng
* RxJS: để xử lý dữ liệu bất đồng bộ

**Phía Back-end:**

* Một ngôn ngữ lập trình phù hợp như Node.js, Java, Python, hoặc Ruby để xử lý các yêu cầu từ phía Front-end.
* Một database để lưu trữ dữ liệu như MySQL, MongoDB, hoặc PostgreSQL.
* API: Một RESTful API cung cấp dữ liệu cho phần front-end của ứng dụng. API có thể được xây dựng bằng Node.js, ASP.NET,Java hoặc một nền tảng back-end khác.

Tuy nhiên , nếu dùng Angular xây dựng web tĩnh (không yêu cầu phái server-side) thì k cần backend.

# 22. Ý nghĩa của file package.json và angular.json trong angular?

Trong Angular, hai tệp quan trọng nhất là **package.json** và **angular.json.**

1. **File package.json** chứa thông tin về các phụ thuộc (dependencies) và phiên bản của chúng, cùng với các script để xây dựng, kiểm tra và triển khai ứng dụng.

* Khi một ứng dụng Angular được tạo, npm install sẽ được sử dụng để cài đặt các phụ thuộc được liệt kê trong package.json.
* Điều này cho phép quản lý các phụ thuộc dễ dàng hơn, vì chúng ta chỉ cần thêm hoặc xoá các phụ thuộc trong tệp này và chạy lệnh npm install để cài đặt hoặc gỡ bỏ chúng.

1. **File angular.json** là tệp cấu hình cho một dự án Angular.

* Nó chứa thông tin về các cài đặt cấu hình của ứng dụng như các đường dẫn, tên ứng dụng, kiểu build và các tùy chọn khác cho trình xây dựng.
* Nó được sử dụng để cấu hình các thành phần trong dự án, bao gồm các tệp style và script, cùng với các cài đặt khác liên quan đến trải nghiệm phát triển của ứng dụng.
* File angular.json được sử dụng khi chạy lệnh ng trong CLI để cấu hình các tùy chọn xây dựng và phát triển của ứng dụng.

# 23.Subscrible dùng để làm gì trong angular?

* Trong Angular, **subscribe**() là một phương thức được sử dụng để lắng nghe các sự kiện (events) hoặc giá trị (values) được trả về từ một Observable hoặc một Subject.
* Một Observable là một luồng dữ liệu có thể bị thay đổi theo thời gian. Khi một Observable phát ra một giá trị, phương thức **subscribe** sẽ lắng nghe và thực hiện các hành động được định nghĩa bên trong nó.
* Subscrible trả về 1 Observer là một đối tượng được sử dụng để lắng nghe các giá trị được trả về từ Observable đó.
* Observer là 1 hàm callback() để xử lý giá trị trả về của Observable gồm: next, error, complete

# 24. Phân biệt cách tạo ra trang web bởi Spring và cách tạo ra trang web kết hợp giữa Spring + Angular (nói kĩ về cách lấy dữ liệu + điều hướng giữa các trang)?

**Sử dụng Spring:**chúng ta thường sử dụng Spring Framework để xây dựng ứng dụng web theo mô hình MVC (Model-View-Controller) hoặc RESTful API. Để xây dựng một trang web bằng Spring, có thể thực hiện theo các bước sau:

* Thiết kế cơ sở dữ liệu: Thiết kế cơ sở dữ liệu để lưu trữ dữ liệu của trang web.
* Tạo Model: Tạo các lớp Model để đại diện cho các đối tượng trong ứng dụng. Model sẽ đại diện cho dữ liệu của ứng dụng.
* Tạo Controller: Tạo các lớp Controller để xử lý yêu cầu từ phía người dùng. Controller sẽ gọi đến các phương thức xử lý dữ liệu và trả về các kết quả.
* Tạo View: Tạo các View để hiển thị các thông tin cho người dùng. View sẽ sử dụng các đối tượng Model để hiển thị dữ liệu.
* Xử lý yêu cầu HTTP: Sử dụng các phương thức của Spring để xử lý yêu cầu HTTP từ phía người dùng. Để xử lý các yêu cầu HTTP, chúng ta có thể sử dụng các annotation của Spring như @RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping,...
* Truy vấn cơ sở dữ liệu: Sử dụng các công cụ của Spring để truy vấn cơ sở dữ liệu và trả về kết quả. Các công cụ này bao gồm Spring Data JPA, Spring JDBC, Hibernate,...
* Tạo điều hướng giữa các trang: Sử dụng các chức năng của Spring để quản lý các route và điều hướng cho các trang khác nhau. Để quản lý các route và điều hướng giữa các trang, chúng ta có thể sử dụng các annotation của Spring như @RequestMapping, @PathVariable, @RequestParam,...
* Sử dụng redirect để chuyển hướng

**Spring + Angular**

* Thiết kế cơ sở dữ liệu: Thiết kế cơ sở dữ liệu để lưu trữ dữ liệu của trang web.
* Tạo RESTful API: Sử dụng Spring để xây dựng RESTful API để cung cấp dữ liệu cho Angular. RESTful API sẽ trả về các dữ liệu dưới dạng JSON hoặc XML.
* Tạo Angular Service: Tạo các Angular Service để gọi các RESTful API của Spring. Angular Service sẽ đại diện cho phần Front-end và sẽ giao tiếp với phần Back-end của trang web.
* Tạo Angular Component: Tạo các Angular Component để hiển thị các thông tin cho người dùng. Component sẽ sử dụng các Angular Service để lấy dữ liệu từ phía Back-end và hiển thị dữ liệu.
* Xử lý yêu cầu HTTP: Sử dụng các phương thức của Spring để xử lý yêu cầu HTTP từ phía người dùng và trả về các kết quả dưới dạng JSON hoặc XML. Để xử lý các yêu cầu HTTP, chúng ta có thể sử dụng các annotation của Spring như @RequestMapping, @GetMapping, @PostMapping, @PutMapping, @DeleteMapping,...
* Truy vấn cơ sở dữ liệu: Sử dụng các công cụ của Spring để truy vấn cơ sở dữ liệu và trả về kết quả. Các công cụ này bao gồm Spring Data JPA, Spring JDBC, Hibernate,...
* Tạo điều hướng giữa các trang: Sử dụng các chức năng của Angular để quản lý các route và điều hướng cho các trang khác nhau. Để quản lý các route và điều hướng giữa các trang, chúng ta có thể sử dụng Angular Router.(navigate, navigateUrl)

# 25. Microservice là gì?

* Microservice là một kiến trúc phần mềm phân tán, trong đó các ứng dụng được xây dựng dưới dạng các dịch vụ nhỏ, độc lập với nhau và hoạt động cùng nhau để tạo ra một ứng dụng lớn hơn. Mỗi microservice thực hiện một chức năng cụ thể của ứng dụng và có thể được phát triển, triển khai và quản lý độc lập với các microservice khác.
* Kiến trúc Microservice cho phép các nhà phát triển tập trung vào phát triển từng phần nhỏ của ứng dụng, tăng tính linh hoạt, giảm độ phức tạp của ứng dụng và cho phép quản lý ứng dụng dễ dàng hơn. Ngoài ra, Microservice còn cho phép ứng dụng có khả năng mở rộng tốt hơn, do đó đáp ứng được nhu cầu tăng trưởng và mở rộng của doanh nghiệp.