Informe de Angular 2

Tema: Angular 2

Nota

Estudiante	Escuela	Asignatura
Victor Gonzalo Maldonado Vilca vmaldonadov@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Tarea	Tema	Duración
10	Angular 2	2 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 23 de mayo de 2024	Al 8 de julio de 2024

1. Introducción

Angular es un framework desarrollado por Google para crear aplicaciones web y móviles robustas y escalables. Se destaca por su arquitectura basada en componentes, enlace de datos bidireccional, y herramientas integradas para manejar formularios, enrutamiento y optimización de rendimiento. Es ampliamente utilizado por su capacidad de desarrollar aplicaciones de una sola página (SPAs) de manera eficiente y estructurada.

2. Objetivos

- Manejar el sistema de ruteos que ofrece Angular.
- Visualizar que otras aplicaciones tiene Angular.
- Entender conceptos de BackEnd y FrontEnd.
- Manejar Módulos, Componentes y Servicios.

3. Tarea

- Volver a implementar las clases teóricas en un proyecto en github realizando commits de cada avance.
- Compartirlo con el profesor (usuario CarloCorrales010)

4. Entregables

- Informe hecho en Latex.
- URL: Repositorio GitHub.

5. Equipos, materiales y temas utilizados

- Angular
- Componentes
- Servicios
- Formularios y Referencias
- Boostrap
- Django

6. URL de Repositorio Github

- Link Repositorio GitHub.
- `https://github.com/Victor-Gonzalo-Maldonado-Vilca/Angular.git`

7. Desarrollo del trabajo

Continuamos con el Desarrollo a partir de Angular1

7.1. Capturas de la Actividad

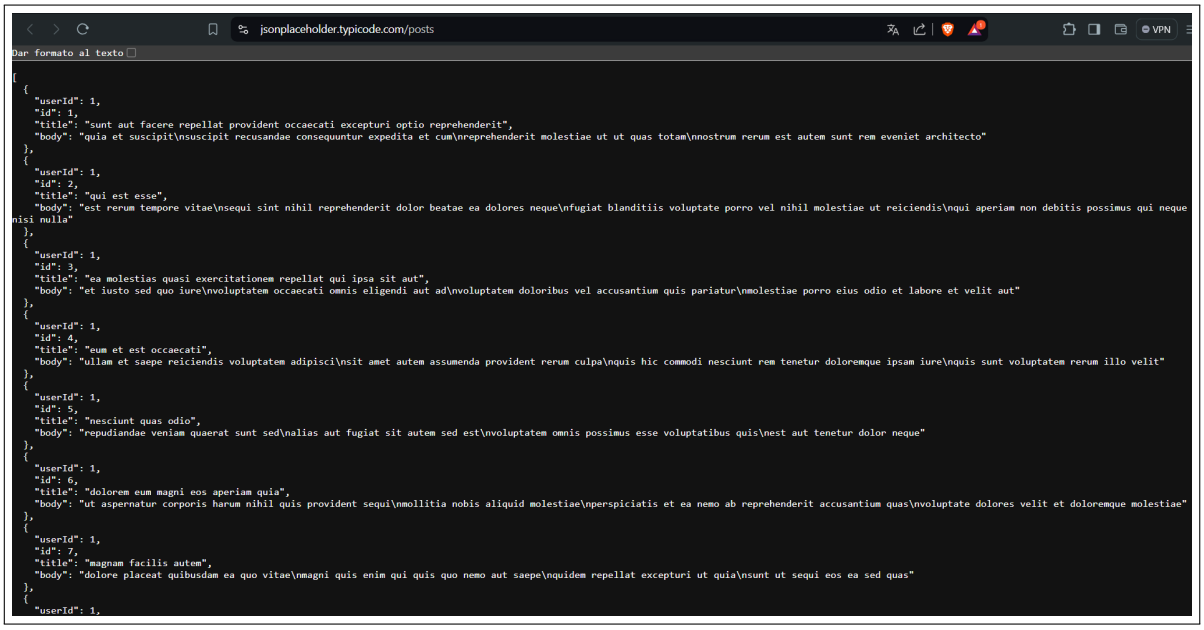


Figura 1: Json de donde se obtendra los datos

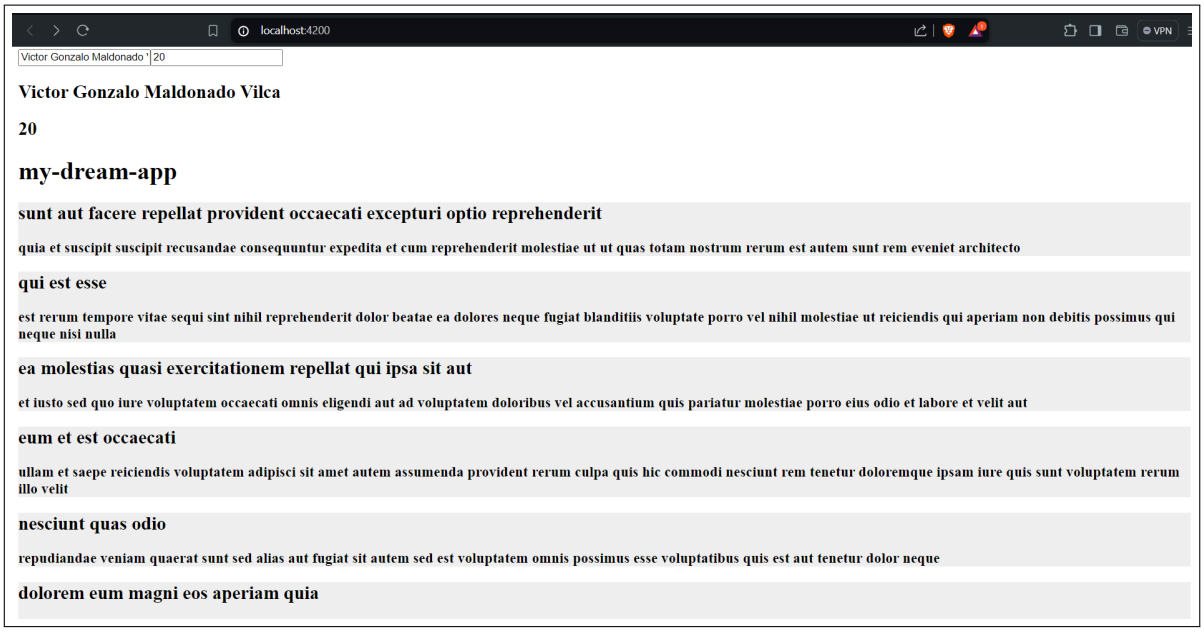
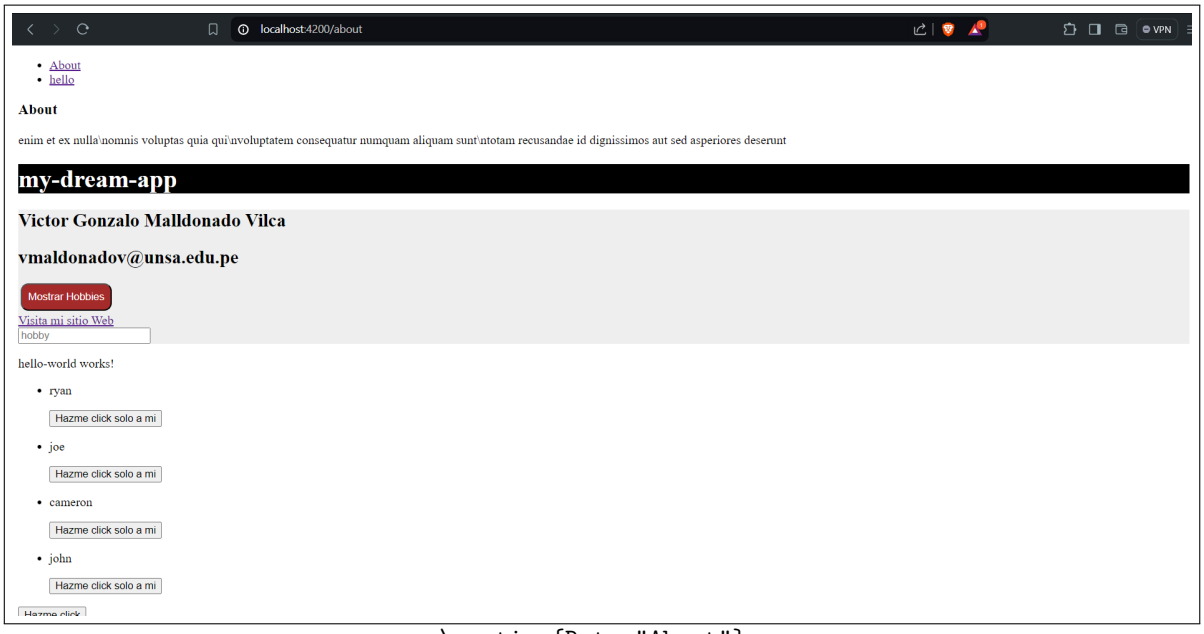
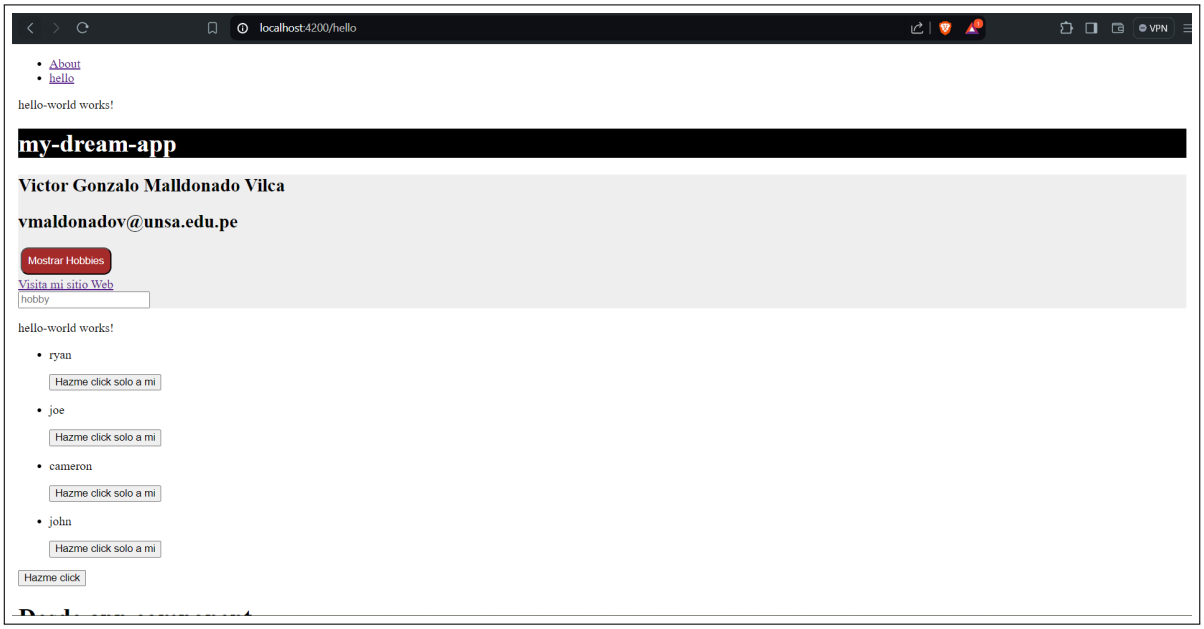


Figura 2: Ejecución 1

Figura 3: Ejecución 2



\caption{Ruta "About"}



\caption{Ruta "Hello"}

7.2. Componente app

7.2.1. Archivo TypeScript

- **Modificación del constructor:** El código ilustra cómo se emplea el servicio DataService para obtener y gestionar datos en un componente. El servicio se integra en el componente mediante la inyección de dependencias (private dataService: DataService). Dentro del constructor del componente, se realiza la inicialización de variables como name, email, webpage, hobbies, y showHobbies. Además, se accede al método getData() del servicio, el cual devuelve un Observa-

ble que se suscribe para obtener los datos (`this.dataService.getData().subscribe(data => { this.posts = data; });`). Este enfoque demuestra cómo Angular maneja la recuperación de datos de forma asincrónica mediante servicios dedicados.

```
17    //Angular 2
18    posts: any = [];
19
20    //Angular 2
21    constructor(private dataService: DataService){
22      console.log('Constructor Working...');
23      this.name = 'Victor Gonzalo Malldonado Vilca';
24      this.email = 'vmaldonadov@unsa.edu.pe';
25      this.webpage = 'http://www.unsa.edu.pe';
26      this.hobbies = ['Futbol', 'Programacion', 'Netflix'];
27      this.showHobbies = false;
28      //Angular 2
29      this.dataService.getData().subscribe(data => {
30        //console.log(data);
31        this.posts = data;
32      });
33    }
```

- **Definir variables:** se define una variable name1 de tipo string que almacena el nombre. `{"Victor Gonzalo Maldonado Vilca"}`, y una variable age de tipo number con el valor inicial de 20.

```
66    //Angular 2
67    name1: string = "Victor Gonzalo Maldonado Vilca";
68    age: number = 20;
```

7.2.2. Archivo html

- **Navegación entre rutas:** Utiliza elementos de lista (`<ul>`) para crear una lista desordenada con dos elementos de lista (`<li>`). Cada elemento de lista contiene un enlace (`<a>`) que utiliza el atributo routerLink para dirigir a diferentes rutas dentro de la aplicación. Uno de los enlaces apunta a la ruta /about y el otro a la ruta /hello.

```
1     <!--Angular 2-->
2     <ul>
3       <li><a routerLink="/about">About</a></li>
4       <li><a routerLink="/hello">hello</a></li>
5     </ul>
```

- **Formulario de entrada:** Esta sección contiene un formulario HTML que permite al usuario introducir un nombre y una edad. Utiliza el binding bidireccional de Angular ([(ngModel)]) para enlazar los valores de los campos de entrada con las propiedades name1 y age en el componente TypeScript. Los valores ingresados se muestran debajo del formulario en etiquetas `<h2>`.

```
53    <!--Angular 2-->
54    <div>
55      <form>
56        <input type='text' name='name' [(ngModel)]='name1'>
```

```
57          <input type='text' name='age' [(ngModel)]='age'>
58        </form>
59        <h2>{{ name1 }}</h2>
60        <h2>{{ age }}</h2>
61      </div>
```

- **Título de la Aplicación:** Esta sección simplemente muestra el valor de la propiedad title del componente en una etiqueta `<h1>`.

```
63      <div>
64        <h1>{{ title }}</h1>
65      </div>
```

- **Lista de Publicaciones:** Esta sección utiliza la directiva *ngFor de Angular para iterar sobre un array de publicaciones (posts). Para cada publicación en el array, se crea un div con la clase ash que contiene el título y el cuerpo de la publicación, mostrados en etiquetas `<h2>` y `<h3>`, respectivamente.

```
66      <div class="ash" *ngFor="let post of posts">
67        <h2>{{ post.title }}</h2>
68        <h3>{{ post.body }}</h3>
69      </div>
```

7.3. Componente About

7.3.1. Archivo html

- **Descripción:** El fragmento HTML representa una sección `"About"` con un título `<h3>` y un párrafo `<p>` que contiene varias líneas de texto descriptivo.
- **Código:**

```
1     <h3>About</h3>
2     <p>enim et ex nulla\nomnis voluptas quia qui\nvoluptatem
3       consequatur numquam aliquam sunt\ntotam recusandae id
4       dignissimos aut sed asperiores deserunt</p>
```

7.3.2. Servicio data

- **Descripción:** La clase DataService está diseñada para gestionar la obtención de datos en una aplicación Angular. El constructor de esta clase toma un parámetro httpClient del tipo HttpClient, que se inyecta automáticamente por Angular cuando se crea una instancia de este servicio. El constructor también incluye una llamada a console.log para confirmar que el servicio está funcionando. La clase define un método getData, que utiliza el httpClient para realizar una solicitud HTTP GET a la URL `https://jsonplaceholder.typicode.com/posts`. Este método devuelve un observable que emite un array de objetos de tipo Post, permitiendo que otros componentes de la aplicación se suscriban a los datos cuando estén disponibles.

- **Código:**

```
9   export class DataService {
10
11    constructor(private httpClient:HttpClient) {
12      console.log("Service working...");
13    }
14
15    getData() {
16      return this.httpClient.get<Post[]>("https://jsonplaceholder.typicode.com/posts");
17    }
18  }
```

7.3.3. Interface Post.ts

- **Descripción:** La interfaz Post define la estructura de los objetos que representan publicaciones en la aplicación. Contiene cuatro propiedades: userId, id, title y body, todas obligatorias. userId y id son de tipo number, mientras que title y body son de tipo string.

- **Código:**

```
1   export interface Post{
2     "userId" : number;
3     "id" : number;
4     "title" : string;
5     "body" : string;
6   }
```

7.3.4. Rutas

- **Descripción:** Se define un arreglo de rutas para la aplicación. Cada ruta incluye un path (ruta de la URL) y un component (componente que se carga al acceder a esa ruta). Las rutas especificadas son la ruta raíz, que carga AppComponent, la ruta about, que carga AboutComponent, y la ruta hello, que carga HelloWorldComponent.

- **Código:**

```
7   const routes: Routes = [
8     {path: '', component: AppComponent},
9     {path: 'about', component: AboutComponent},
10    {path: 'hello', component: HelloWorldComponent},
11  ];
```

7.4. Visualización de otras aplicaciones de Angular

- **Incorporación de Videos:** Angular permite la fácil integración de videos en las aplicaciones, ya sea a través de componentes personalizados o utilizando servicios externos. Esto es útil para agregar contenido multimedia enriquecido que mejore la experiencia del usuario.

- **Integración con Bootstrap:** Utilizando Bootstrap con Angular, los desarrolladores pueden crear interfaces de usuario receptivas y atractivas de manera eficiente. Bootstrap proporciona un conjunto de componentes predefinidos y estilos que se pueden personalizar e integrar sin problemas en las aplicaciones Angular.

- **Interoperabilidad con Django:** Angular puede funcionar junto con Django para crear aplicaciones web robustas y escalables. Mientras que Django maneja el backend y la lógica del servidor, Angular se encarga del frontend, proporcionando una experiencia de usuario fluida y dinámica.

8. Conclusiones

- La configuración de rutas en Angular es fundamental para la navegación dentro de una aplicación.
- Definir rutas claras y asignarles componentes específicos permite una mejor organización y accesibilidad de las diferentes partes de la aplicación.
- Utilizar componentes separados para distintas rutas ayuda a mantener el código modular y más fácil de mantener.
- Este enfoque también facilita la implementación de nuevas funcionalidades y mejora la experiencia del usuario al navegar por la aplicación.
- Seguir una estructura bien definida para las rutas asegura que la aplicación sea escalable y manejable a medida que crece en complejidad.
- El uso de servicios como `DataService` permite la obtención de datos de fuentes externas, como una API, y la integración de estos datos en la aplicación.
- Emplear interfaces como `Post` para definir la estructura de los datos recibidos garantiza que los datos se manejen de manera consistente y tipada, lo que mejora la seguridad y la legibilidad del código.
- La interacción con APIs RESTful, como la obtención de datos desde `https://jsonplaceholder.typicode.com/posts`, demuestra la capacidad de Angular para manejar datos dinámicos y actualizaciones en tiempo real.
- Integrar datos JSON en la aplicación permite a los desarrolladores trabajar con datos estructurados de manera eficiente, facilitando tareas como la visualización de listas y la gestión de información en la interfaz de usuario.

8.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

9. Referencias

- https://v17.angular.io/guide/architecture
- https://v17.angular.io/guide/binding-syntax