Documentación de la Prueba Práctica – Backend

1. Descripción del Servicio

El servicio REST solicitado permite consultar la información básica de un cliente. A continuación se detallan los requisitos y condiciones del servicio:

1.1. Requisitos del Servicio

- Manejo de Códigos HTTP:
  - 200 OK: Respuesta exitosa.
  - 400 Bad Request: Solicitud inválida (por ejemplo, tipo de documento no permitido o datos faltantes).
  - 404 Not Found: Cliente no encontrado (por ejemplo, documento no existe en la base de datos).
  - 500 Internal Server Error: Error en el servidor.

- Datos de Entrada:
  - Tipo de Documento (obligatorio): Solo se aceptan 'C' (Cédula de ciudadanía) y 'P' (Pasaporte).
  - Número de Documento (obligatorio): Identificador único del cliente.

- Datos de Salida:

  Los datos deben estar "quemados" en la aplicación, con la siguiente estructura de respuesta para el cliente con Cédula de ciudadanía 23445322:
  - Primer Nombre: Juan
  - Segundo Nombre: Pablo
  - Primer Apellido: Pérez
  - Segundo Apellido: Gómez
  - Teléfono: 123456789

- Dirección: Calle Falsa 123
- Ciudad de Residencia: Bogotá

- Puerto de la Aplicación:

  La aplicación debe iniciar en el puerto 8090.

- Tecnologías Requeridas:
  - Spring Boot para el desarrollo del servicio REST.
  - Maven para la gestión de dependencias y construcción del proyecto.

1.2. Entregables

- JAR de la Aplicación: El archivo ejecutable de la aplicación.
- Código Fuente: Todo el código necesario para ejecutar y mantener la aplicación.

1.3. Notas Adicionales

- Los datos de respuesta están "mockeados" y solo se retornará información para el cliente con Cédula de ciudadanía 23445322.
- Aunque no es obligatorio, se recomienda implementar:
  - Manejo de Logs: Para registrar la actividad y posibles errores de la aplicación.
  - Pruebas Unitarias: Para asegurar la funcionalidad del servicio.
  - Calidad de Código: Asegurarse de que el código sea limpio y mantenible.

2. Diseño del Servicio

2.1. Endpoints

- Endpoint para Consultar Información del Cliente:

- Método: GET
- URL: `/api/clientes`
- Parámetros de Consulta:
  - `tipo_documento`: Tipo de documento (C o P).
  - `numero_documento`: Número del documento.
- Ejemplo de Solicitud:

  GET /api/clientes?tipo_documento=C&numero_documento=23445322

- Ejemplo de Respuesta Exitosa (200 OK):
  json
  {
    "primerNombre": "Juan",
    "segundoNombre": "Pablo",
    "primerApellido": "Pérez",
    "segundoApellido": "Gómez",
    "telefono": "123456789",
    "direccion": "Calle Falsa 123",
    "ciudadResidencia": "Bogotá"
  }

- Ejemplo de Respuesta de Error (400 Bad Request):
  json
  {
    "error": "Tipo de documento inválido o datos faltantes."
  }

- Ejemplo de Respuesta de Error (404 Not Found):

json

```
{
  "error": "Cliente no encontrado."
}
```

- Ejemplo de Respuesta de Error (500 Internal Server Error):

json

```
{
  "error": "Error en el servidor."
}
```

2.2. Implementación

- Controlador: `ClienteController` - Maneja las solicitudes HTTP y delega la lógica al servicio.

- Servicio: `ClienteService` - Contiene la lógica de negocio para obtener los datos del cliente.

- Configuración: `application.properties` - Configuración de puertos y otras propiedades.

Documentación de la Prueba Práctica – Frontend

1. Descripción de la Aplicación

Se requiere desarrollar una aplicación en Angular que permita a los usuarios ingresar el tipo y número de documento de un cliente para consultar y visualizar su información básica. La información se almacenará en un archivo JSON.

1.1. Requisitos de la Aplicación

- **Pantalla de Ingreso de Información:**
  - **Campos:**
    - **Tipo de Documento:** Lista desplegable con opciones 'Cédula de ciudadanía' y 'Pasaporte'. Por defecto, no debe estar seleccionado ningún valor.

        - **Número de Documento:** Campo de texto que debe permitir entre 8 y 11 caracteres numéricos. Debe mostrar los números con separadores de miles.
    - **Botón Buscar:**
        - Debe estar inactivo inicialmente.
        - Se activará solo cuando ambos campos (tipo de documento y número de documento) estén completados y sean válidos.
- **Pantalla de Resumen:**
    - **Datos a Mostrar:**
        - Información básica del cliente obtenida de un archivo JSON.
    - **Botón Volver:** Permite regresar a la pantalla de ingreso de información.

1.2. Componentes de la Aplicación

- **Componente de Ingreso (IngresoComponent):**
    - **Template (ingreso.component.html):** Contiene el formulario con los campos para el tipo y número de documento, y el botón buscar.
    - **Styles (ingreso.component.css):** Estilo del formulario y del botón.
    - **Clase (ingreso.component.ts):** Lógica para la validación de campos y habilitación del botón buscar.
- **Componente de Resumen (ResumenComponent):**
    - **Template (resumen.component.html):** Muestra la información básica del cliente y el botón volver.
    - **Styles (resumen.component.css):** Estilo de la pantalla de resumen.
    - **Clase (resumen.component.ts):** Lógica para mostrar la información del cliente y manejar el botón volver.

1.3. Requisitos de Diseño

- **Bootstrap:** La aplicación debe utilizar Bootstrap para el diseño de los componentes y la interfaz de usuario.
- **Estructura del Proyecto:** Cada pantalla debe ser un componente Angular independiente.
- **Archivo JSON:** La información del cliente se almacenará en un archivo JSON.

1.4. Criterios de Aceptación

- **Componentización:** Cada pantalla debe estar implementada como un componente Angular independiente.
- **Entrega:** El entregable es el código fuente completo de la aplicación.
- **Similitud con Diseños:** El desarrollo debe seguir los diseños proporcionados o especificados.
- **Uso de Bootstrap:** La aplicación debe usar Bootstrap para la interfaz de usuario.
- **Opcional:**
    - **Consumo del Servicio:** Aunque no es obligatorio, se recomienda consumir el servicio REST para obtener la información del cliente.
    - **Uso de Grillas:** Para la presentación de datos en la pantalla de resumen.
    - **Pruebas Unitarias:** Se recomienda implementar pruebas unitarias para asegurar la funcionalidad del componente.

2. Implementación

2.1. Pantalla de Ingreso de Información

Template (ingreso.component.html):

html

Copiar código

```
<div class="container">
  <form>
    <div class="form-group">
      <label for="tipoDocumento">Tipo de Documento</label>
      <select id="tipoDocumento" class="form-control" [(ngModel)]="tipoDocumento" name="tipoDocumento">
        <option value="">Seleccionar</option>
        <option value="C">Cédula de ciudadanía</option>
        <option value="P">Pasaporte</option>
      </select>
    </div>
```

```
    <div class="form-group">
      <label for="numeroDocumento">Número de Documento</label>
      <input id="numeroDocumento" type="text" class="form-control"
[(ngModel)]="numeroDocumento" name="numeroDocumento" maxlength="11"
[ngModel]="formatNumeroDocumento()">
    </div>
    <button type="button" class="btn btn-primary" [disabled]="!isFormValid()"
(click)="buscar()">Buscar</button>
  </form>
</div>
```

Clase (ingreso.component.ts):

typescript

Copiar código

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

@Component({
  selector: 'app-ingreso',
  templateUrl: './ingreso.component.html',
  styleUrls: ['./ingreso.component.css']
})
export class IngresoComponent {
  tipoDocumento: string = '';
  numeroDocumento: string = '';

  isFormValid(): boolean {
    return this.tipoDocumento && this.numeroDocumento.length >= 8 &&
this.numeroDocumento.length <= 11;
  }

  formatNumeroDocumento(): string {
```

```
    return this.numeroDocumento.replace(/\B(?=(\d{3})+(?!\d))/g, ',');
  }

  buscar(): void {
    // Implementar lógica para buscar cliente
  }
}
```

2.2. Pantalla de Resumen

Template (resumen.component.html):

```
html
Copiar código
<div class="container">
  <h2>Información del Cliente</h2>
  <!-- Mostrar información básica aquí -->
  <button class="btn btn-secondary" (click)="volver()">Volver</button>
</div>
```

Clase (resumen.component.ts):

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

@Component({
  selector: 'app-resumen',
  templateUrl: './resumen.component.html',
  styleUrls: ['./resumen.component.css']
})
export class ResumenComponent {
  // Aquí se mostraría la información del cliente
```

```
  volver(): void {
    // Implementar lógica para volver a la pantalla de ingreso
  }
}
```