

Diplomado de actualización en nuevas tecnologías para el desarrollo de Software.

Taller Unidad 3 Backend y Frontend.

Estudiante: Yeison Oswaldo Ruano Ortega

Código: 220036075

Universidad de Nariño.

Ingeniería de Sistemas.

Semestre X

Ipiales – Nariño

2024

## 1. Creación de componentes, uso de ngModel, RouterLink, Servicios. (3 Ptos).

se pueden identificar claramente los conceptos de creación de componentes, uso de ngModel, RouterLink, y servicios de la siguiente manera:

### Creación de Componentes:

Aunque la creación de componentes específicos no se detalla directamente en el código, se deduce por el uso de plantillas HTML y el comportamiento que muestra la estructura modular, que los componentes como los formularios y las tablas son parte de los componentes creados en Angular.

### Uso de ngModel:

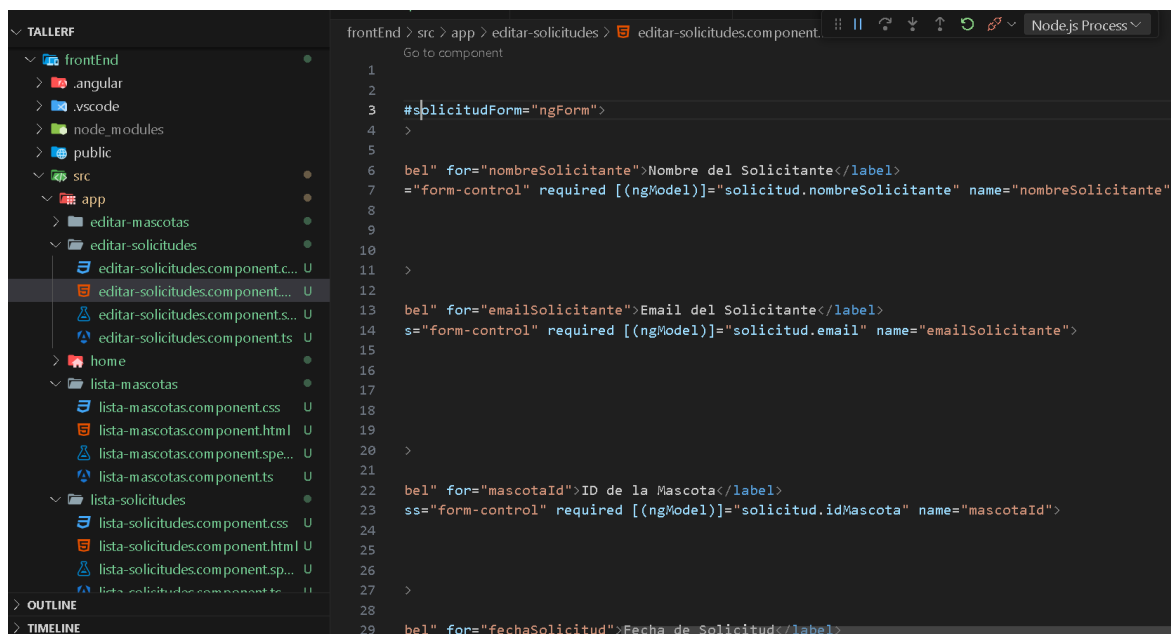
ngModel se utiliza para enlazar los datos del formulario con el modelo de datos en la lógica del componente. Ejemplos en el código:

html

```
<input type="text" class="form-control" required [(ngModel)]="mascota.nombre"
name="nombre">
```

```
<input type="number" class="form-control" required [(ngModel)]="mascota.edad"
name="edad">
```

Estos inputs están enlazados a las propiedades nombre y edad del objeto mascota, demostrando la vinculación bidireccional de datos.



```
frontEnd > src > app > editar-solicitudes > editar-solicitudes.component.html
Go to component
1
2
3 #solicitudForm="ngForm">
4 >
5
6 bel" for="nombreSolicitante">Nombre del Solicitante</label>
7 s="form-control" required [(ngModel)]="solicitud.nombreSolicitante" name="nombreSolicitante">
8
9
10
11 >
12
13 bel" for="emailSolicitante">Email del Solicitante</label>
14 s="form-control" required [(ngModel)]="solicitud.email" name="emailSolicitante">
15
16
17
18
19
20 >
21
22 bel" for="mascotaId">ID de la Mascota</label>
23 ss="form-control" required [(ngModel)]="solicitud.idMascota" name="mascotaId">
24
25
26
27 >
28
29 bel" for="fechaSolicitud">Fecha de Solicitud</label>
```

### Uso de RouterLink:

RouterLink se utiliza para la navegación dentro de la aplicación sin recargar la página.  
Ejemplos en el código:

html

```
<a class="btn btn-primary" [routerLink]="['/solicitudes/agregar']">Nueva Solicitud</a>
```

```
<a class="btn btn-info" [routerLink]="['/solicitudes/editar/', solicitud.id]">Editar</a>
```

Estos enlaces dirigen al usuario a rutas específicas (/solicitudes/agregar y /solicitudes/editar) para agregar o editar una solicitud.

```
<form class="form-inline">
  <fieldset class="form-group col-sm-11"></fieldset>
  <fieldset class="form-group col-sm-1">
    <a class="btn btn-primary" [routerLink]="['/solicitudes/agregar']">
  </fieldset>
</form>
<br>
```

```
<fieldset class="form-group col-sm-1">
  <a class="btn btn-primary" [routerLink]="['/mascotas/agregar']">Nueva Mascota</a>
</fieldset>
</form>
```

## Servicios:

Aunque no se ve directamente en el fragmento proporcionado, los servicios suelen estar implicados en la lógica de manipulación de datos y llamadas HTTP. la interacción con el backend y la actualización de la solicitud, como en la función `actualizarSolicitud`, normalmente se gestionan mediante servicios inyectados en los componentes.

```
onSubmit() {
  console.log("On Submit");
  console.log("Solicitud a actualizar:", this.solicitud);

  // Viene de Editar
  if (this.solicitud.id) {
    this.solicitudService.actualizarSolicitud(this.solicitud).subscribe({
      next: data => {
        console.log(data);
        this.router.navigate(['/solicitudes']); // Redirige a la lista de solicitudes
      },
      error: err => {
        console.log(`Error al actualizar ${err}`);
      }
    });
  } else {
    // Viene de Nueva Solicitud
    this.solicitudService.agregarSolicitud(this.solicitud).subscribe({
      next: data => {
        console.log(data);
        this.router.navigate(['/solicitudes']); // Redirige a la lista de solicitudes
      },
      error: err => {
        console.log(`Error al Agregar ${err}`);
      }
    });
  }
}
```

The screenshot displays two windows from a development environment. On the left is the Thunder Client, which lists several REST API requests. The selected request is a POST to `127.0.0.1:4000/solicitudes/crear`, made 36 minutes ago. Other requests include PUT and GET calls to `/mascotas/actualizar/1`, `/mascotas/buscarid/1`, `/solicitudes/actualizar/1`, `/solicitudes/buscarid/1`, `/solicitudes/eliminar/1`, `/mascotas/eliminar/2`, `/solicitudes/buscar`, and `/mascotas/crearMascota`. On the right is a VS Code editor showing the `editar-solicitudes.component.ts` file. The code defines the `onSubmit()` method, which checks if a request ID exists. If it does, it calls `actualizarSolicitud`; otherwise, it calls `agregarSolicitud`. Both calls use RxJS `subscribe` to handle success (logging data and navigating to `/solicitudes`) and error (logging the error message) scenarios.

## **2. Uso de HTML 5 y JavaScript (Se debe desarrollar una estructura ordenada, con código legible y documentado). (1 Pto.).**

Estructura Semántica: Utiliza etiquetas semánticas de HTML5 como <header>, <footer>, <section>, <article>, y <aside> para mejorar la claridad y accesibilidad del código.

### **HTML5**

#### **Estructura del Formulario:**

El formulario está diseñado para capturar datos relacionados con una solicitud, utilizando Angular Forms con ngModel para la vinculación bidireccional de datos.

El formulario se envía a través de un evento (ngSubmit) que llama a la función onSubmit().

#### **Campos del Formulario:**

Nombre del Solicitante: Un campo de texto que requiere que el usuario ingrese su nombre. Se usa ngModel para enlazar el valor a solicitud.nombreSolicitante.

Email del Solicitante: Un campo de correo electrónico que valida automáticamente el formato del email y está vinculado a solicitud.email.

ID de la Mascota: Un campo numérico para ingresar el ID de la mascota. Este campo también está vinculado a solicitud.idMascota.

Fecha de Solicitud: Un campo de fecha que permite al usuario seleccionar la fecha de la solicitud, vinculado a solicitud.fechaSolicitud.

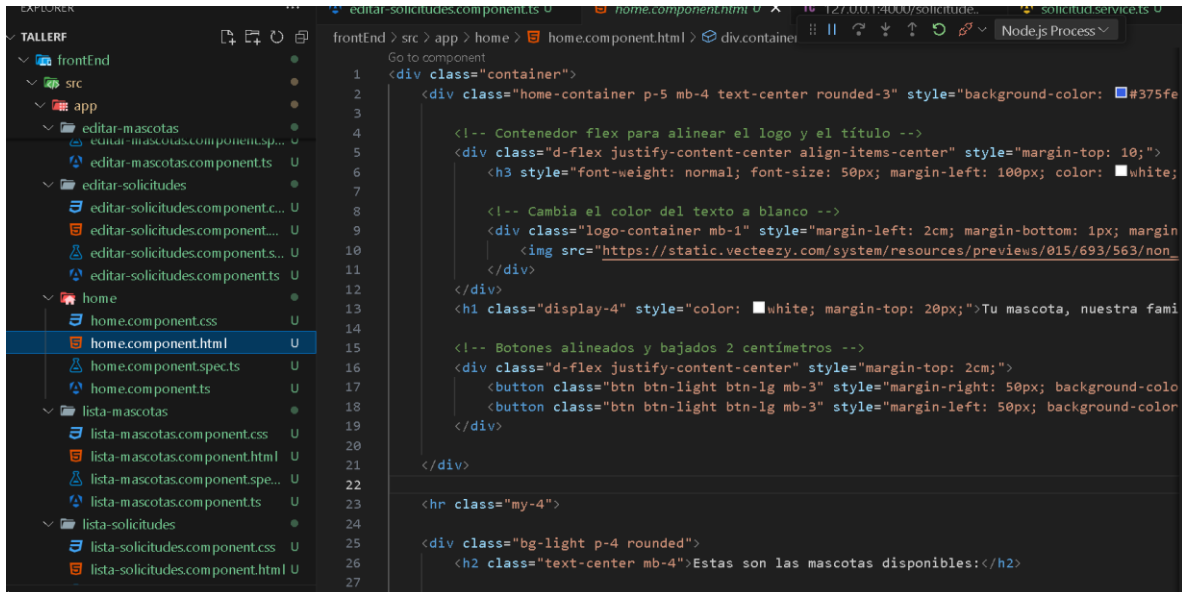
Estado: Un campo de texto para ingresar el estado de la solicitud, que está vinculado a solicitud.estado.

#### **Botón de Envío:**

Hay un botón de envío que solo está habilitado si el formulario es válido ([disabled]="!solicitudForm.form.valid"). Este botón ejecuta la acción de envío del formulario.

#### **Estilo y Diseño:**

El formulario utiliza clases de Bootstrap (form-control, btn, etc.) para estilos y diseño responsivo, asegurando que se vea bien en diferentes tamaños de pantalla.



## JavaScript

### Importaciones:

Se importan los módulos necesarios de Angular, el modelo de datos SolicitudModel, y el servicio SolicitudService para gestionar las solicitudes.

Se importan ActivatedRoute y Router para manejar la navegación y el acceso a parámetros de la ruta.

### Definición del Componente:

El componente EditarSolicitudesComponent se define con un selector, una plantilla y un archivo de estilos.

### Variables

idSolicitud: Un número que representa el ID de la solicitud que se va a editar. Se inicializa en 0.

solicitud: Instancia del modelo SolicitudModel, inicializada en el constructor con valores predeterminados.

### Constructor:

Se inyectan SolicitudService, ActivatedRoute, y Router.

Se inicializa solicitud con un nuevo objeto SolicitudModel.

### Método ngOnInit:

Se obtiene el idSolicitud de los parámetros de la ruta y se convierte a número.

Si idSolicitud es válido, se llama al servicio para obtener la solicitud a editar, y se asigna a la variable solicitud. Se convierte la fecha de la solicitud a un objeto Date.

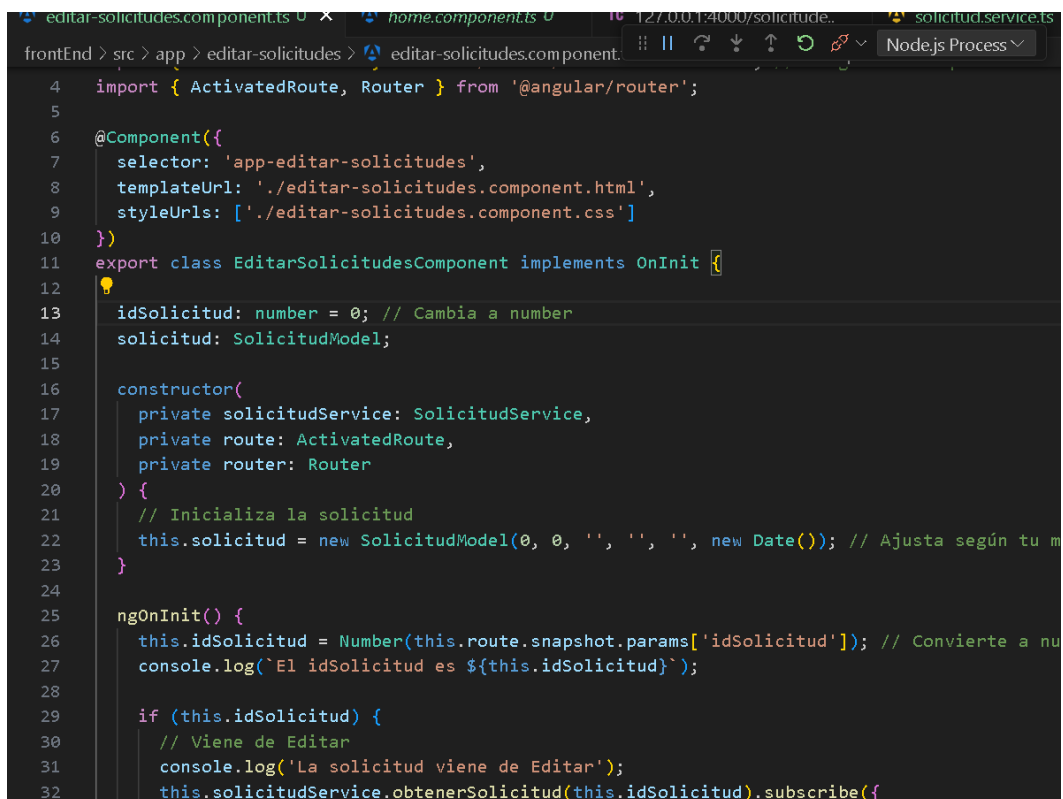
Si idSolicitud no es válido, se indica que se viene de una nueva solicitud.

### Método onSubmit:

Al enviar el formulario, se verifica si hay un id en la solicitud:

Si hay un ID, se llama al método de actualización del servicio y se redirige a la lista de solicitudes al completar.

Si no hay un ID, se considera una nueva solicitud y se llama al método para agregar la solicitud, redirigiendo también a la lista de solicitudes al finalizar.



```
4 import { ActivatedRoute, Router } from '@angular/router';
5
6 @Component({
7   selector: 'app-editar-solicitudes',
8   templateUrl: './editar-solicitudes.component.html',
9   styleUrls: ['./editar-solicitudes.component.css']
10 })
11 export class EditarSolicitudesComponent implements OnInit {
12
13   idSolicitud: number = 0; // Cambia a number
14   solicitud: SolicitudModel;
15
16   constructor(
17     private solicitudService: SolicitudService,
18     private route: ActivatedRoute,
19     private router: Router
20   ) {
21     // Inicializa la solicitud
22     this.solicitud = new SolicitudModel(0, 0, '', '', '', new Date()); // Ajusta según tu m
23   }
24
25   ngOnInit() {
26     this.idSolicitud = Number(this.route.snapshot.params['idSolicitud']); // Convierte a nu
27     console.log(`El idSolicitud es ${this.idSolicitud}`);
28
29     if (this.idSolicitud) {
30       // Viene de Editar
31       console.log('La solicitud viene de Editar');
32       this.solicitudService.obtenerSolicitud(this.idSolicitud).subscribe({
```

**3. Estilos CSS (Uso de Bootstrap), se debe generar una interface ordenada estructurada y agradable para el usuario final. (1 Pto.) Elaborar un informe que detalle el proceso de construcción y la implementación del aplicativo.**

El uso de Bootstrap en el desarrollo de la interfaz de usuario de la aplicación ha permitido crear un diseño ordenado, estructurado y visualmente agradable para el usuario final.

### **Estructura y Organización**

Contenedores y Filas: Se utilizan las clases `.container`, `.row`, y `.col` para definir la estructura general de la página. Esto garantiza un diseño responsivo que se adapta a diferentes tamaños de pantalla.

Secciones Claras: Cada sección de la aplicación se encuentra claramente delimitada, lo que mejora la navegabilidad y la experiencia del usuario. Por ejemplo, la sección de bienvenida, los botones de navegación y la lista de mascotas están organizadas en contenedores separados.

### **Estilos Consistentes**

Colores y Tipografía: Se han aplicado colores coherentes que se alinean con la temática de la aplicación. Por ejemplo, se utilizan colores como el azul para el fondo de la sección de bienvenida y verde para el botón de "Mascotas", creando un ambiente acogedor y atractivo.

### **Uso de Componentes de Bootstrap**

Cards para Mascotas: Se implementan tarjetas (`.card`) para mostrar información de cada mascota, proporcionando una visualización clara y organizada de los datos. Las tarjetas incluyen imágenes, títulos y descripciones, lo que permite a los usuarios obtener información de manera rápida y efectiva.

Flexbox: Se utiliza el sistema de flexbox de Bootstrap para alinear y distribuir los elementos de manera efectiva, asegurando que se vean bien en diferentes dispositivos. Por ejemplo, las clases `.d-flex`, `.justify-content-center` y `.align-items-center` permiten una alineación adecuada de los elementos.



## Botones

Los botones son grandes y claros, utilizando estilos de Bootstrap que permiten mantener una apariencia uniforme. Esto facilita su identificación y uso por parte del usuario. Se usaron clases como `.btn`, `.btn-lg`, `.btn-light` para el tamaño y el estilo.

## Estilos Utilizados:

`.container` para la estructura principal.

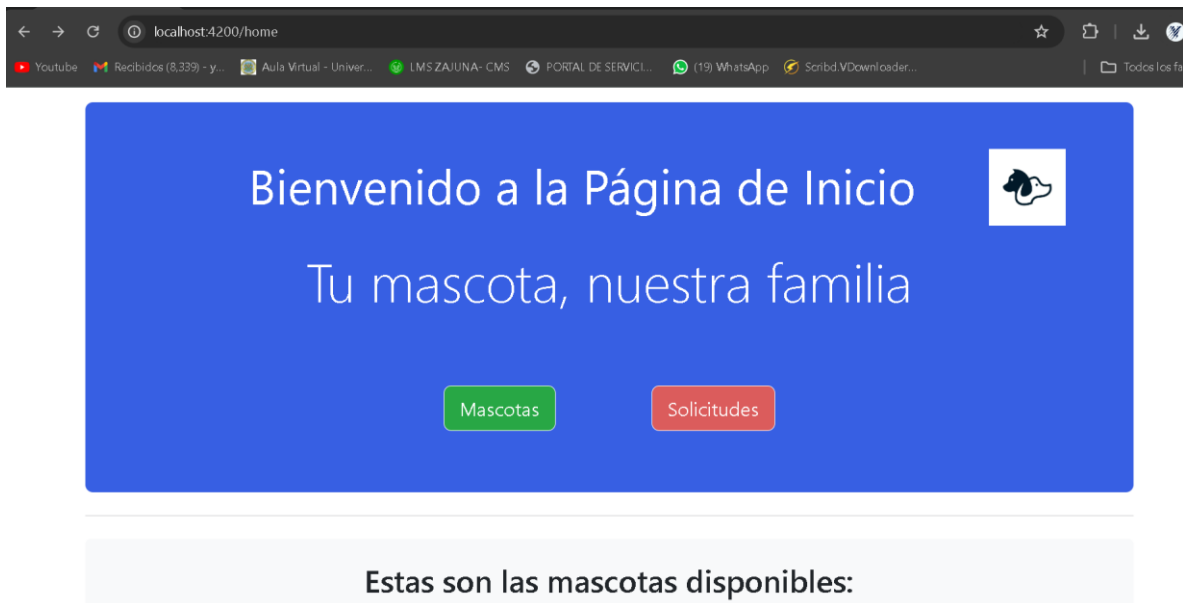
`.row` y `.col` para el diseño de rejilla.

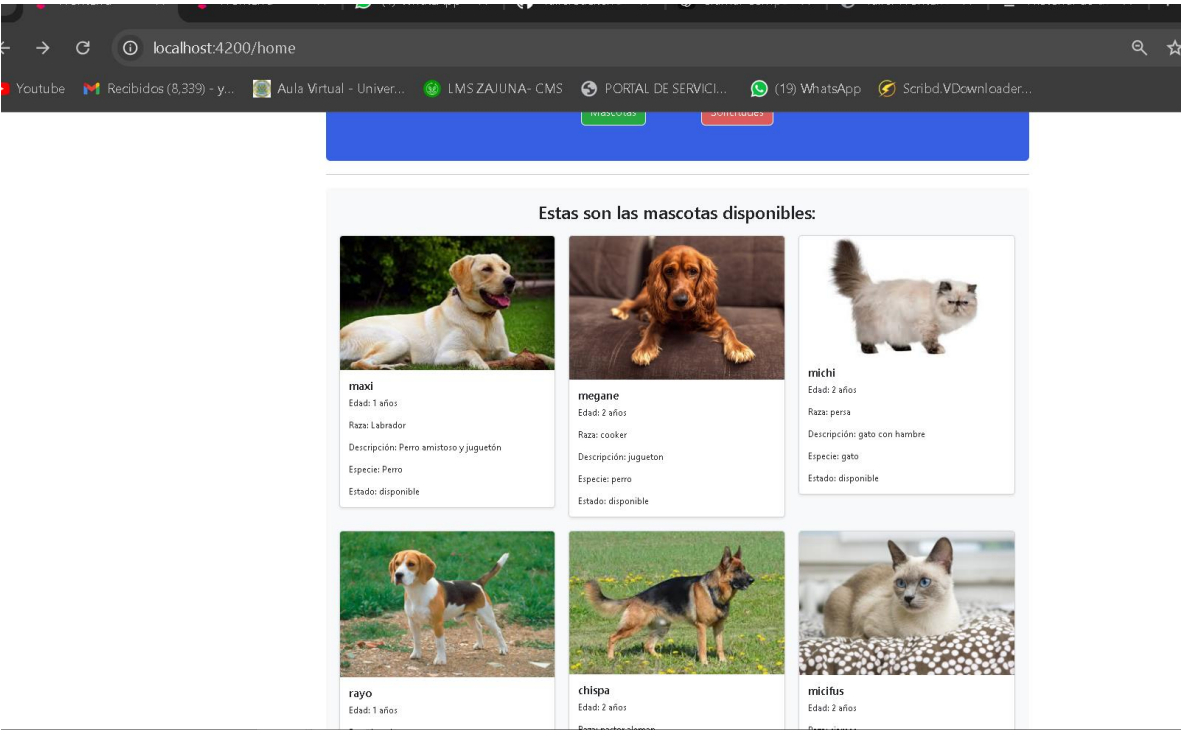
`.btn`, `.btn-lg`, y `.btn-light` para los botones.

`.card` para mostrar las mascotas.

Clases de utilidades como `text-center`, `mb-3`, `p-4`, y `rounded` para márgenes, padding y bordes redondeados.

HOME:





Mascotas:

Nueva Mascota

ID	NOMBRE	EDAD	ESPECIE	RAZA	DESCRIPCIÓN	ESTADO	FECHA DE REGISTRO	ACCIONES	
2	maxi	1	Perro	Labrador	Perro amistoso y juguetón	disponible	2024-09-24T00:00:00.0	Editar	Borrar
7	megane	2	perro	cooker	jugueton	disponible	2024-09-23T17:30:25.0	Editar	Borrar
10	michi	2	gato	persa	gato con hambre	disponible	2024-09-23T18:40:46.0	Editar	Borrar
12	rayo	1	Perro	beagle	alegre	disponible	2024-09-23T19:34:20.0	Editar	Borrar
14	chispa	2	Perro	pastor aleman	entrenado	disponible	2024-09-22T00:00:00.0	Editar	Borrar
22	micifus	2	gato	siames	molesto	disponible	2024-09-25T19:48:08.0	Editar	Borrar

## AGREGAR MASCOTA:

localhost:4200/mascotas/agregar

os (8,339) - y... Aula Virtual - Univer... LMS ZAJUNA- CMS PORTAL DE SERVICI... (19) WhatsApp Scribd.VDownloader...

Nombre

Edad

Especie

Raza

Descripción

Estado

Fecha de Registro

dd/mm/aaaa

Imagen

Enviar

## EDITAR MASCOTA:

Nombre

maxi

Edad

1

Especie

Perro

Raza

Labrador

Descripción

Perro amistoso y juguetón

Estado

disponible

Fecha de Registro

dd/mm/aaaa

Imagen

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9Gc5ClclLBPXgLatnRgGjoxXnXAc9lcSyoGpyA&ks

Enviar

SOLICITUDES:

Nueva Solicitud

ID	Nombre Solicitante	Email Solicitante	Fecha de Solicitud	Estado	Acciones
2	Juan sebastian	juan.perez@example.com	9/20/24	cerrado	<div>EditarBorrar</div>
3	Juan hernandez	juan.perez@example.com	9/20/24	pendiente	<div>EditarBorrar</div>
4	miguel ortegaw	miguelortega@example.com	9/20/24	pendiente	<div>EditarBorrar</div>

NUEVA SOLICITUD:

Nombre del Solicitante

Email del Solicitante

ID de la Mascota

0

Fecha de Solicitud

dd/mm/aaaa

Estado

pendiente

Enviar

EDITAR SOLICITUD:

Nombre del Solicitante

Juan sebastian

Email del Solicitante

juan.perez@example.com

ID de la Mascota

1

Fecha de Solicitud

dd/mm/aaaa

Estado

cerrado

Enviar