UNIVERIDAD DA VINCI DE GUATEMALA
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
DESARROLLO MULTIPLATAFORMA
BRANDON ANTONY CHITAY COUTIÑO

PROYECTO FASE FINAL

WILSON ANDRÉ AGUIRRE CARDONA
202000194
16/05/2024

Documentación Fase 1: Propuesta

Para poder definir la arquitectura del sistema es importante identificar antes los requerimientos funcionales y no funcionales que va a tener el sistema, estos se presentan a continuación.

Requerimientos Funcionales

- El sistema debe permitir el registro de nuevos dueños y mascotas.
- Se debe tener la información relevante de los dueños de las mascotas (nombre, teléfono, correo).
- Se debe tener la información relevante de las mascotas (nombre, especie, raza, edad, dueño).
- Se debe tener programación de citas, en las cuales los administradores puedan programar dichas citas sin ningún problema, así mismo se debe registrar la fecha y la hora de la cita.
- Debe existir un registro de desparasitaciones realizadas a las mascotas.

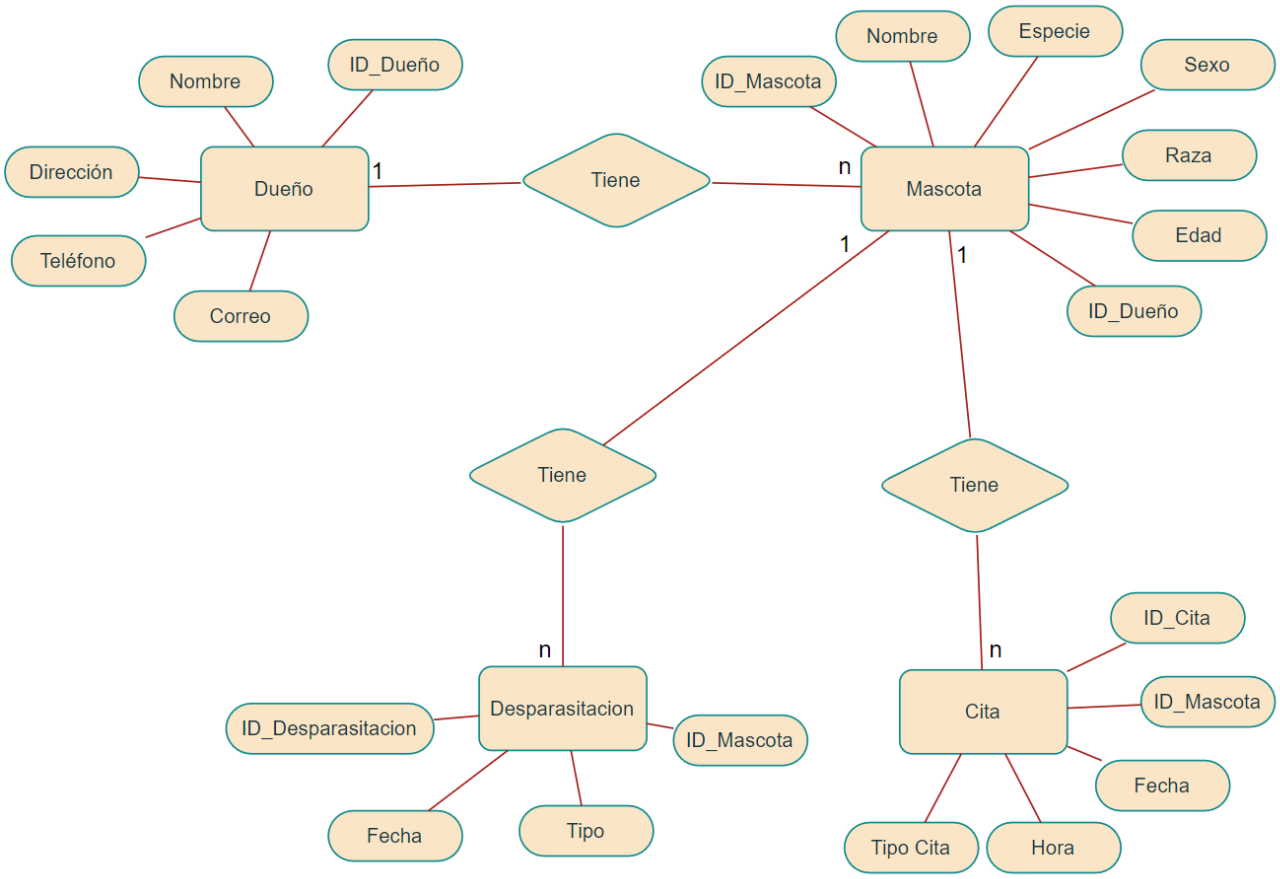
Requerimientos no Funcionales

- El sistema debe ser accesible desde cualquier lugar (móviles, tabletas, computadoras, etc.)
- El sistema debe tener una interfaz intuitiva en la cual sobresalga la facilidad de uso para brindar una experiencia de usuario agradable.
- El sistema debe garantizar la seguridad de los datos confidenciales de los clientes y las mascotas.
- El sistema debe ser escalable para manejar un crecimiento futuro en el número de clientes y mascotas y tener la capacidad de manejar una gran cantidad de datos de manera eficiente.
- El sistema debe tener alta disponibilidad para garantizar el acceso en cualquier momento que sea necesario.

Modelación Conceptual

Para esta parte es importante identificar las principales entidades del sistema, de esta manera se puede iniciar la modelación de los datos de una manera conceptual.

Diagrama de Entidad-Relación sobre las principales entidades del sistema



Con este diagrama de entidad-relación, se visualizan las entidades principales del sistema con sus respectivos atributos y las relaciones entre cada una de ellas, esto proporciona una comprensión clara de cómo podría estructurarse la lógica de la base de datos para la clínica.

Modelación Lógica

Tabla de Dueños

ID_DUEÑO	NOMBRE	DIRECCIÓN	TELÉFONO	CORREO
1	Felipe	Villa Nueva	54972315	Felipe@gmail.com
2	Rodrigo	San Miguel Petapa	42308718	Rodrigo@gmail.com
3	Heidy	Villa Canales	34782201	Heidy@gmail.com

Tabla de Mascotas

ID_MASCOTA	ID_DUEÑO	NOMBRE	ESPECIE	SEXO	RAZA	EDAD
1	1	Lupe	Perro	Macho	Bulldog	7
2	2	Max	Perro	Macho	Labrador	10
3	3	Brownie	Gato	Macho	Angora	12

Tabla de Citas

ID_CITA	ID_MASCOTA	FECHA	HORA	TIPO_CITA
1	1	2024-02-19	Perro	Vacunación
2	2	2024-02-22	Perro	Consulta
3	3	2024-02-28	Gato	Corte de pelo

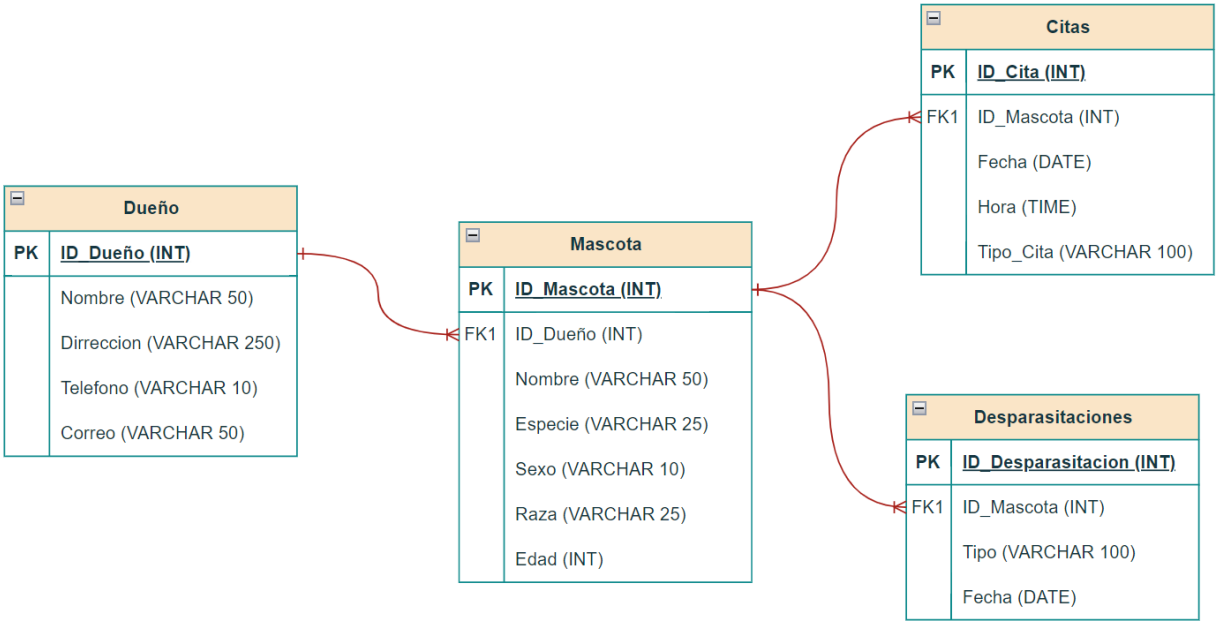
Tabla de Desparasitaciones

ID_DESPARASITACIÓN	ID_MASCOTA	TIPO	FECHA
1	1	Antiparasitario	2024-02-19
2	2	Antiparasitario	2024-03-25
3	3	Antihelmíntico	2024-04-05

Con estas tablas se tiene la estructura lógica de los datos, cada tabla tiene datos relevantes sobre los dueños, las mascotas, las citas y las desparasitaciones que se han realizado. De esta manera es más fácil comprender el tipo de flujo de datos que el sistema va a tener.

Modelación Física

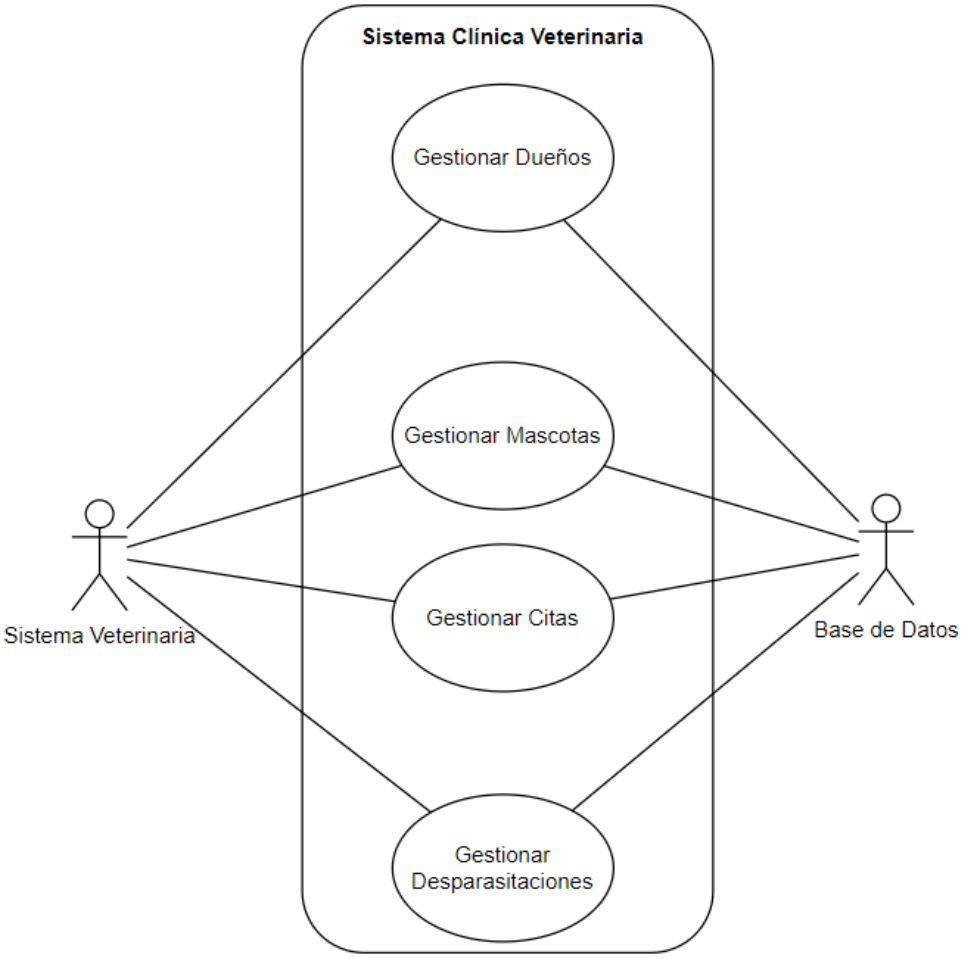
Diagrama de modelación física



En este diagrama se muestra la relación de las llaves primarias de cada entidad, la entidad de dueño no contiene ninguna llave foránea a diferencia de las demás entidades que cuentan con una llave foránea. La entidad mascota hace referencia a la entidad de dueño, las entidades de citas y desparasitaciones hacen referencia a la entidad de mascota.

Diagrama de caso de uso General

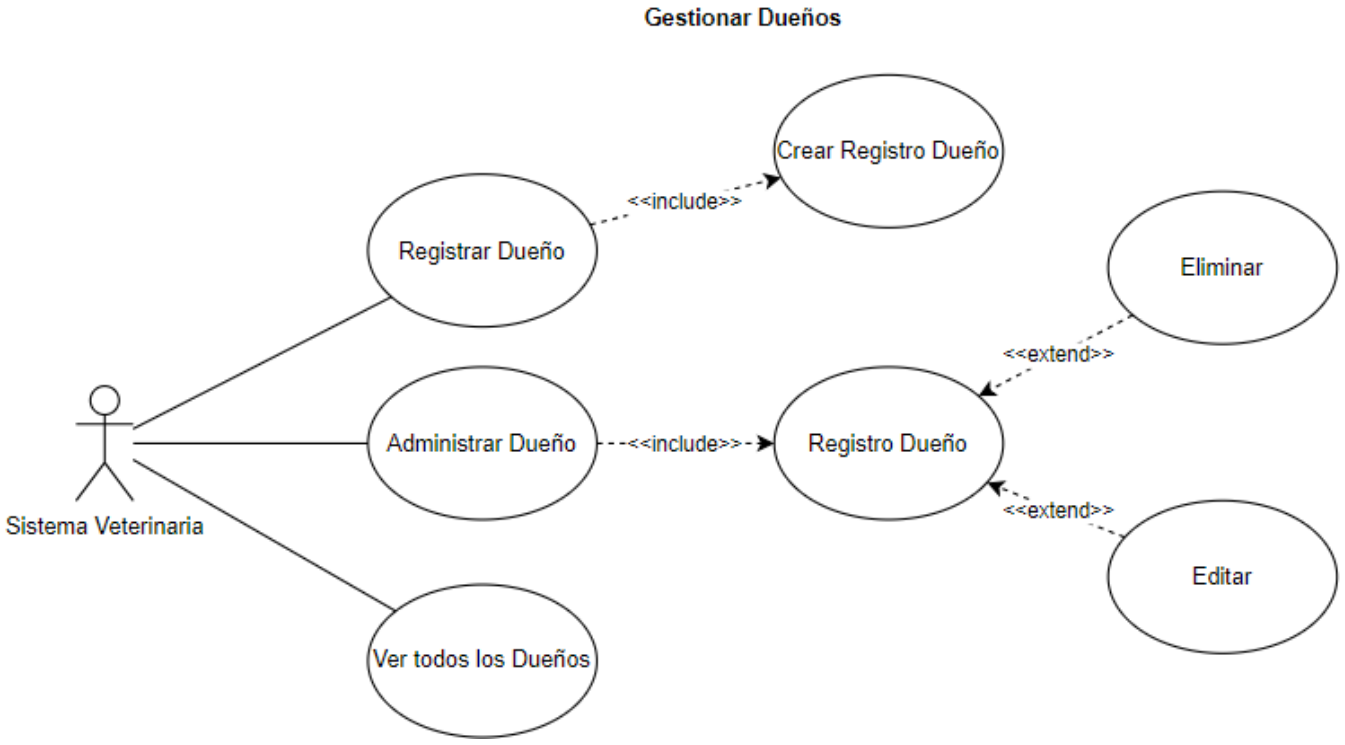
Diagrama de caso de uso del sistema de la clínica veterinaria



En este diagrama se muestra que el sistema contará con 4 módulos principales. La veterinaria podrá interactuar con cada uno de los módulos por separado, Dichos módulos se comunican con una Base de Datos donde se guardará toda la información importante.

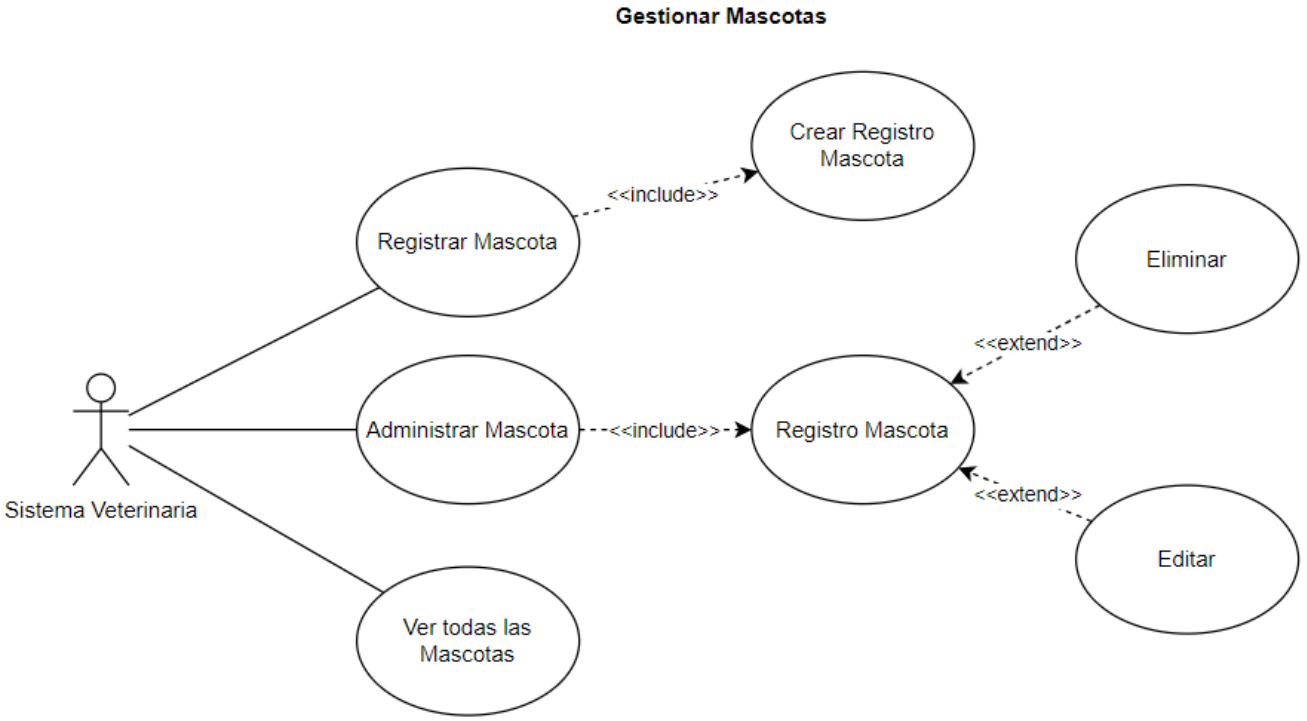
Diagramas de casos de uso Específicos

Diagrama de caso de uso para gestionar dueños



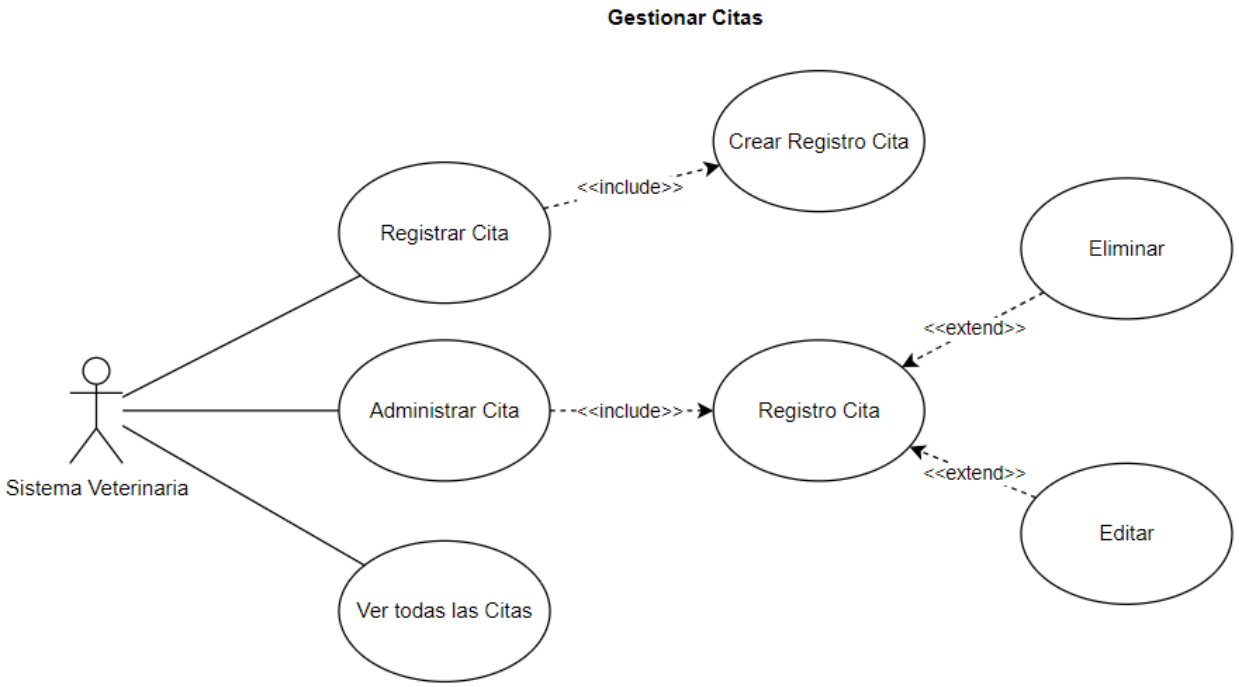
En el módulo Gestionar dueños se encuentran 3 sub-módulos. El primero consiste en registrar un nuevo dueño, este permite crear el registro del dueño. El segundo permite administrar los dueños, aquí se selecciona el registro del dueño, en este caso se puede eliminar o editar dicho registro. Por último, se pueden ver todos los registros existentes de dueños.

Diagrama de caso de uso para gestionar mascotas



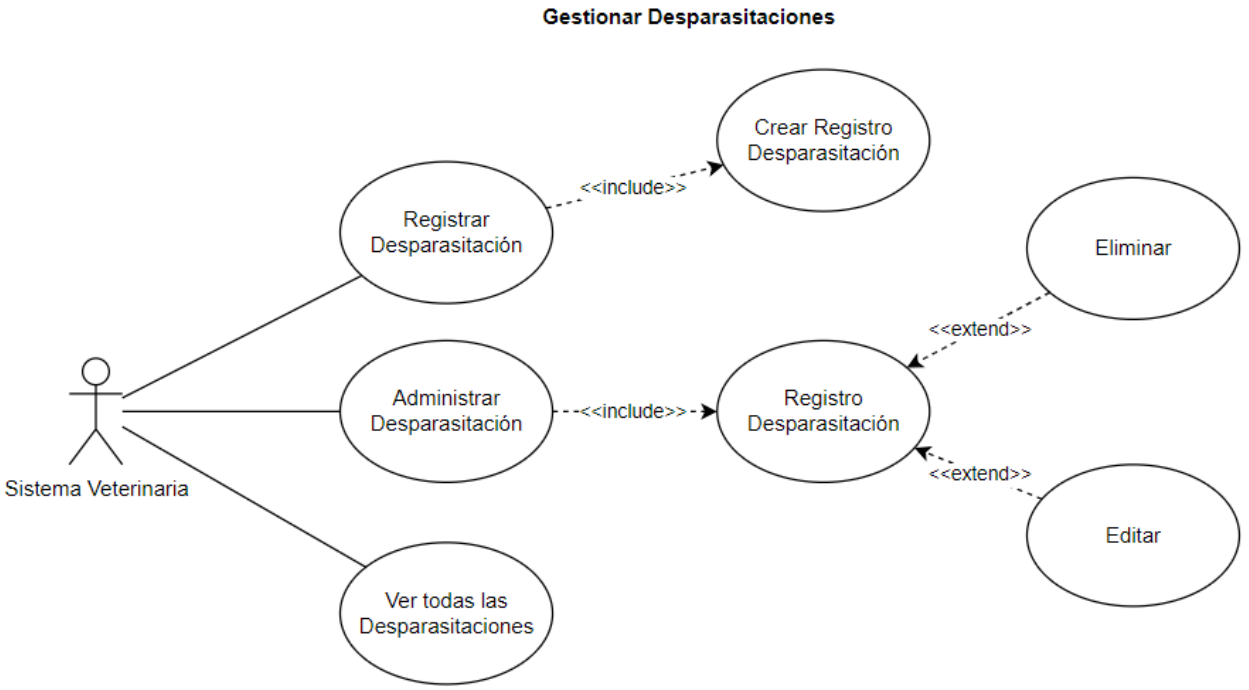
En el módulo Gestionar mascotas se encuentran 3 sub-módulos. El primero consiste en registrar una nueva mascota, este permite crear el registro de la mascota. El segundo permite administrar las mascotas, aquí se selecciona el registro de la mascota, en este caso se puede eliminar o editar dicho registro. Por último, se pueden ver todos los registros existentes de mascotas.

Diagrama de caso de uso para gestionar las citas



En el módulo Gestionar citas se encuentran 3 sub-módulos. El primero consiste en registrar una nueva cita, este permite crear el registro de la cita. El segundo permite administrar las citas, aquí se selecciona el registro de la cita, en este caso se puede eliminar o editar dicho registro. Por último, se pueden ver todos los registros existentes de citas.

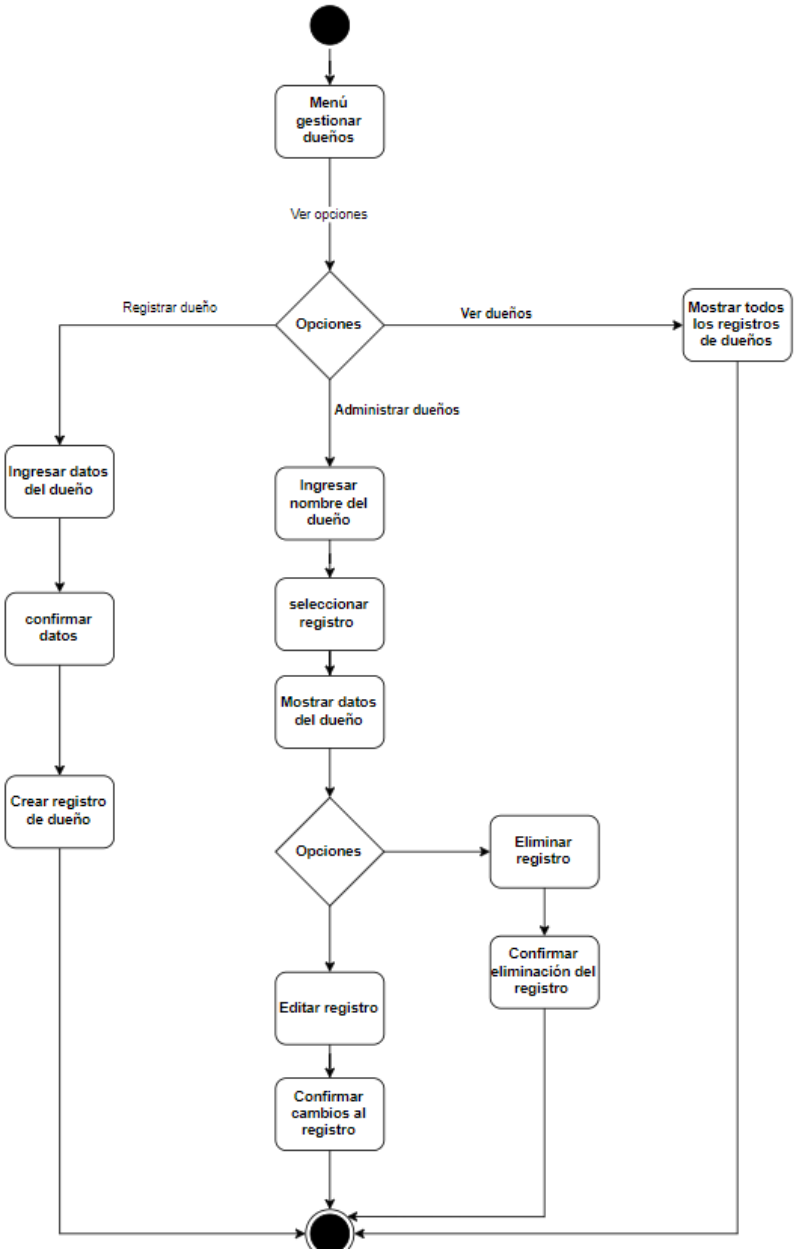
Diagrama de caso de uso para gestionar desparasitaciones



En el módulo Gestionar desparasitaciones se encuentran 3 sub-módulos. El primero consiste en registrar una nueva desparasitación, este permite crear el registro de la desparasitación. El segundo permite administrar las desparasitaciones, aquí se selecciona el registro de la desparasitación, en este caso se puede eliminar o editar dicho registro. Por último, se pueden ver todos los registros existentes de desparasitaciones.

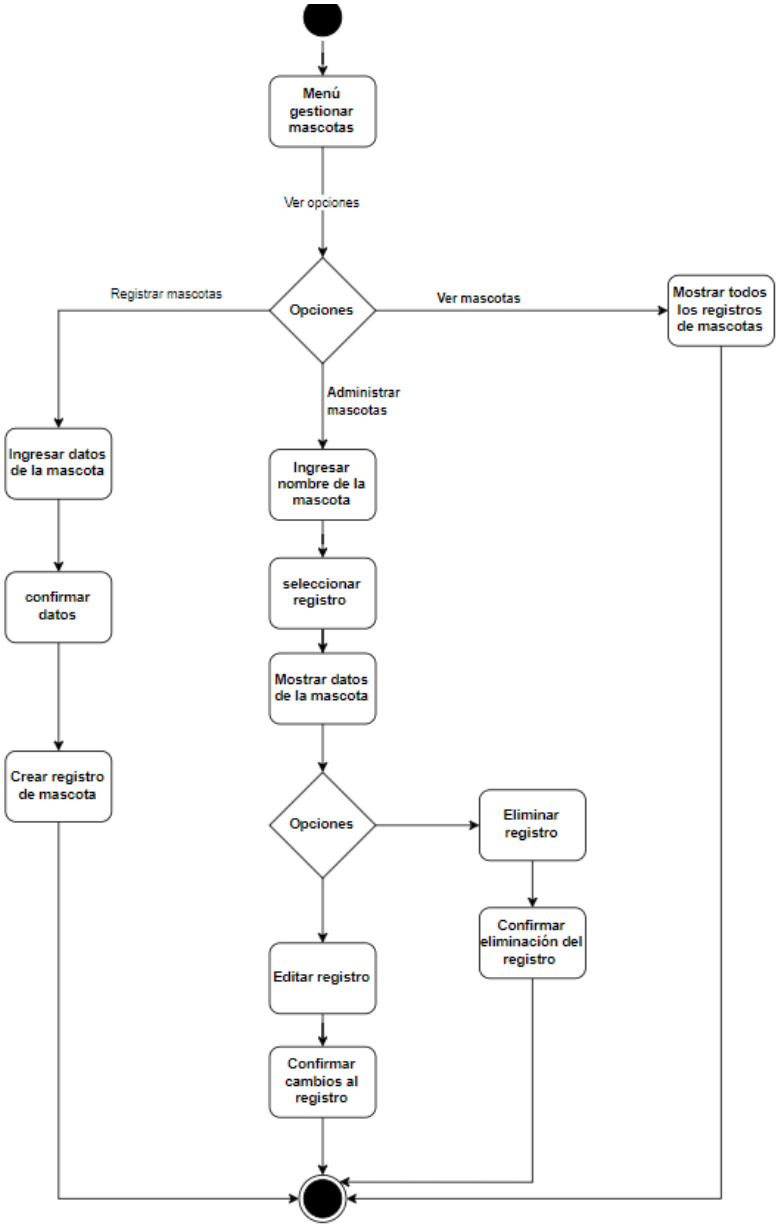
Diagramas de procesos por cada caso de uso

Diagrama de proceso para gestionar dueños



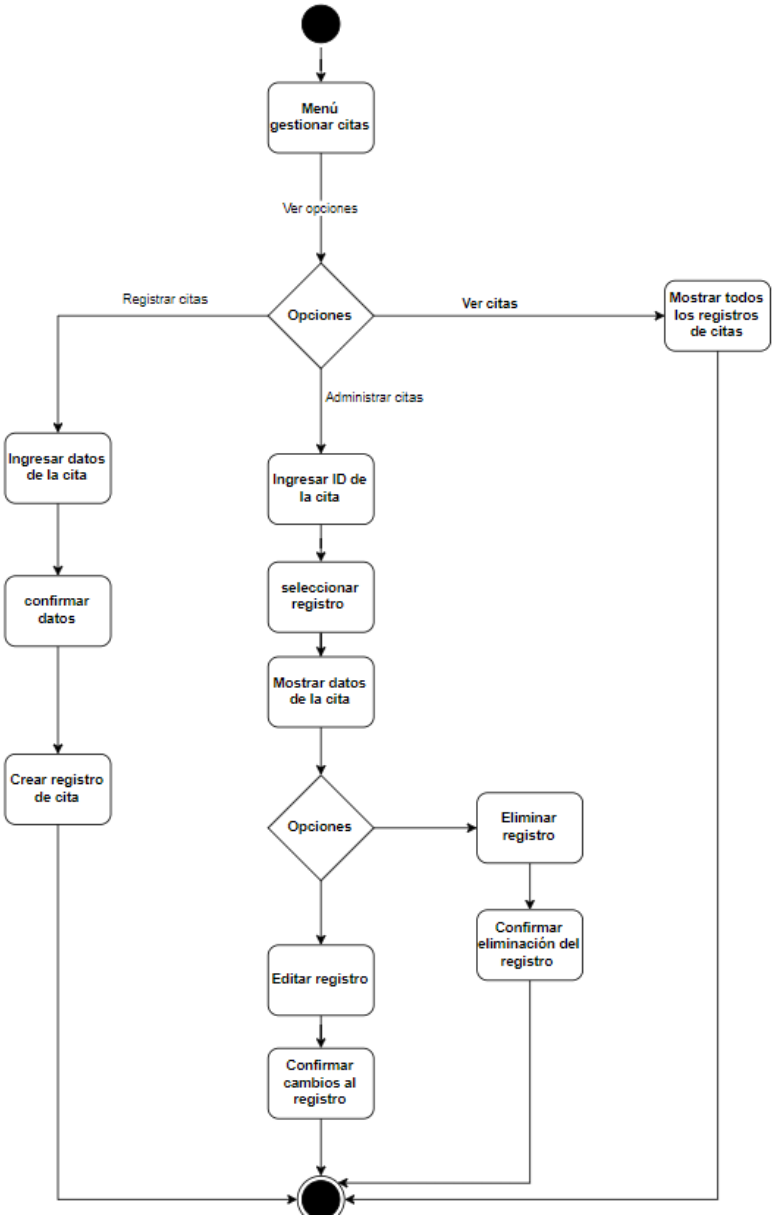
Este diagrama representa el proceso para gestionar a los dueños, en el cual se muestran las opciones disponibles, las cuales tienen procesos específicos, en base a la opción que se elija se recorren los procesos que se deben realizar para llevar a cabo esa opción. Todas finalizan en un estado final.

Diagrama de proceso para gestionar mascotas



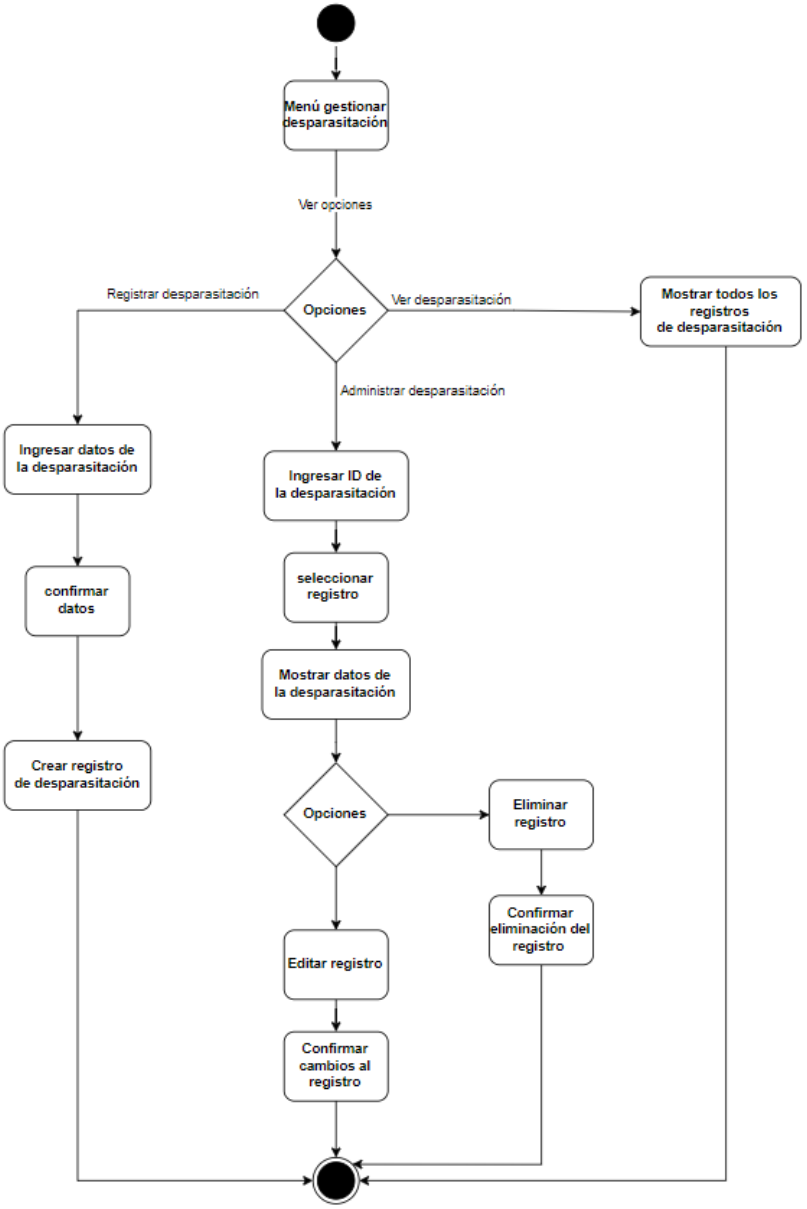
Este diagrama representa el proceso para gestionar a las mascotas, en el cual se muestran las opciones disponibles, las cuales tienen procesos específicos, en base a la opción que se elija se recorren los procesos que se deben realizar para llevar a cabo esa opción. Todas finalizan en un estado final.

Diagrama de proceso para gestionar citas



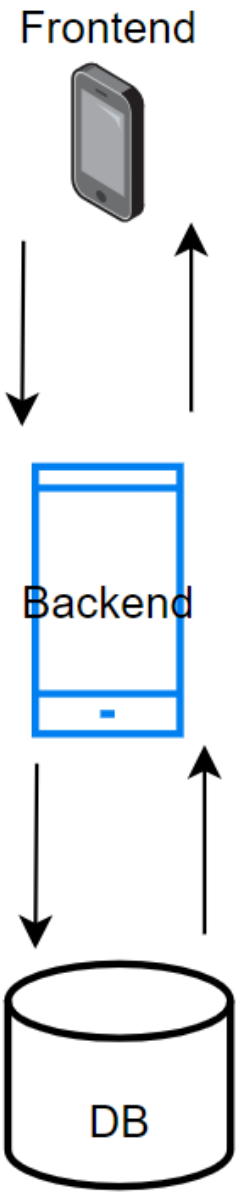
Este diagrama representa el proceso para gestionar a las citas, en el cual se muestran las opciones disponibles, las cuales tienen procesos específicos, en base a la opción que se elija se recorren los procesos que se deben realizar para llevar a cabo esa opción. Todas finalizan en un estado final.

Diagrama de proceso para gestionar desparasitaciones



Este diagrama representa el proceso para gestionar a las desparasitaciones, en el cual se muestran las opciones disponibles, las cuales tienen procesos específicos, en base a la opción que se elija se recorren los procesos que se deben realizar para llevar a cabo esa opción. Todas finalizan en un estado final.

Diagrama de despliegue



Identificación de Herramientas

Docker

Docker será utilizado para crear y gestionar los contenedores que alojarán y ejecutarán las aplicaciones del sistema de la clínica veterinaria "Mascotas Felices". Esto proporcionará un entorno de desarrollo y despliegue consistente y reproducible, facilitando la implementación y la escalabilidad del sistema.

Django

Django será utilizado para desarrollar el backend del sistema de la clínica veterinaria. Proporciona herramientas y funcionalidades integradas para la gestión de bases de datos, el enrutamiento de URLs, la autenticación de usuarios, entre otros, lo que facilitará la implementación de la lógica de negocio y la interacción con la base de datos.

React Native

React Native será utilizado para desarrollar la aplicación móvil del sistema de la clínica veterinaria. Permitirá crear una interfaz de usuario nativa y optimizada para dispositivos móviles, lo que garantizará una experiencia de usuario fluida y receptiva.

PostgreSQL

PostgreSQL será utilizado como el sistema de gestión de bases de datos para almacenar y gestionar la información del sistema de la clínica veterinaria. Proporcionará una estructura de datos sólida y confiable, permitiendo la consulta, inserción, actualización y eliminación de datos de manera eficiente y segura.

Documentación Fase 2: Contratos de Servicio

Dueños

Endpoint para obtener la lista de todos los dueños registrados:

- URL: /api/duenos/
- Método HTTP: GET
- Parámetros de consulta: Ninguno

Respuesta exitosa (código de estado 200 OK):


```
[
    {
        "id_dueno": 1,
        "nombre": "Wilson",
        "direccion": "zona 10",
        "telefono": 45678213,
        "correo": "wilson@gmail.com",
        "status": "Active"
    },
    {
        "id_dueno": 2,
        "nombre": "Wilson",
        "direccion": "zona 10",
        "telefono": 45678213,
        "correo": "wilson@gmail.com",
        "status": "Active"
    },
    {
        "id_dueno": 3,
        "nombre": "Wilson",
        "direccion": "zona 10",
        "telefono": 45678213,
        "correo": "wilson@gmail.com",
        "status": "Active"
    }
]
```

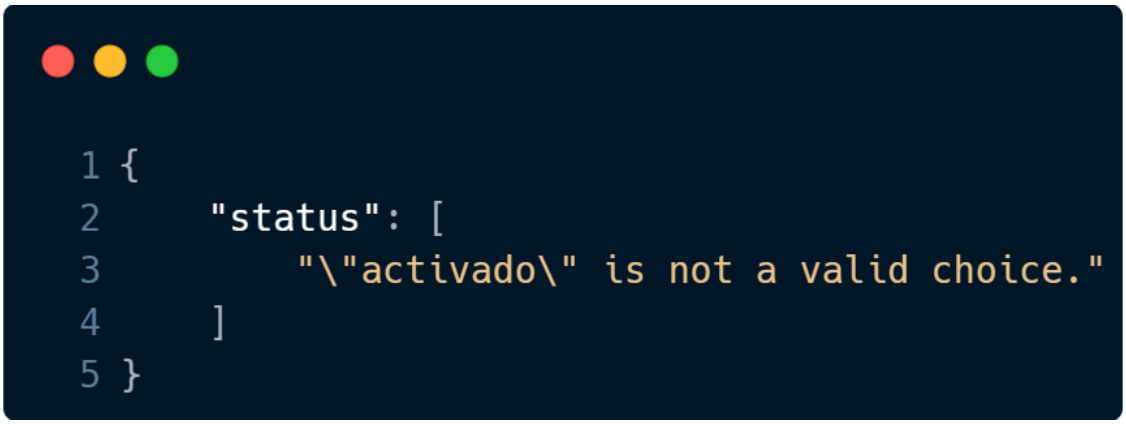
Endpoint para crear un nuevo dueño:

- URL: /api/duenos/
- Método HTTP: POST
- Parámetros de creación: “nombre”, “direccion”, “telefono”, “correo”, “status(Active/Inactive, default=Active)”.

Respuesta exitosa (código de estado 201 Created):

```
{
    "message": "Dueño creado",
    "Dueño": {
        "id_dueno": 7,
        "nombre": "Jaime",
        "direccion": "zona 10",
        "telefono": 45678213,
        "correo": "wilson@gmail.com",
        "status": "Active"
    }
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "status": [
        "\"activado\" is not a valid choice."
    ]
}
```

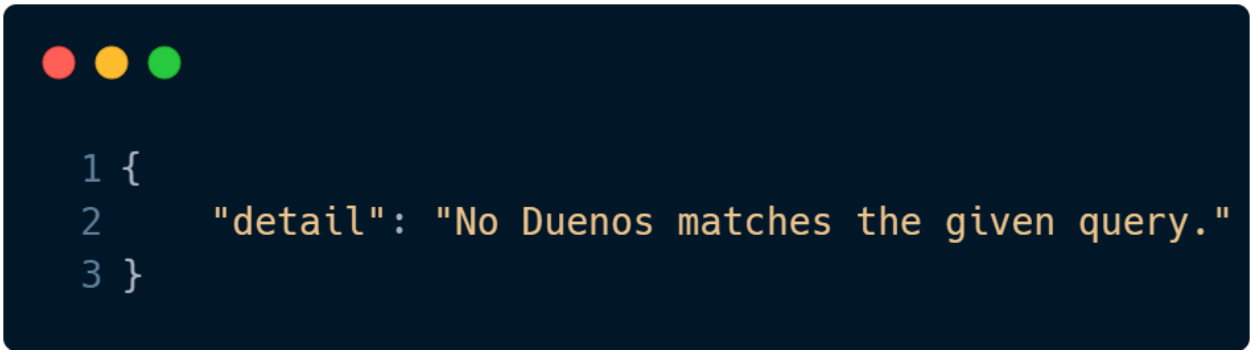
Endpoint para obtener detalles de un dueño específico:

- URL: /api/duenos/{id_dueno}/
- Método HTTP: GET
- Parámetros de consulta: “id_dueno”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_dueno": 1,
    "nombre": "wilson",
    "direccion": "zona 15",
    "telefono": 45678213,
    "correo": "wil@gmail.com",
    "status": "Active"
}
```

Respuesta de código de error 404 Not Found:



```
{
    "detail": "No Duenos matches the given query."
}
```

Endpoint para actualizar información de un dueño:

- URL: /api/duenos/{id_dueno}/
- Método HTTP: PUT
- Parámetros de actualización: “nombre”, “direccion”, “telefono”, “correo”.

Respuesta exitosa (código de estado 200 OK):


```
{
    "id_dueno": 1,
    "nombre": "Wilson",
    "direccion": "zona 12",
    "telefono": 45678213,
    "correo": "Mau@gmail.com",
    "status": "Active"
}
```


Respuesta de código de error 400 Bad Request:

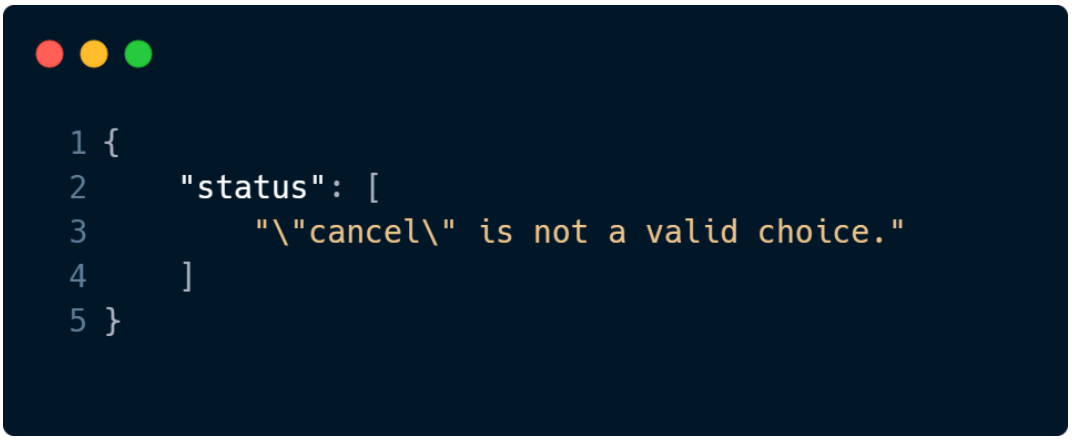

```
{
    "nombre": [
        "This field is required."
    ],
    "direccion": [
        "This field is required."
    ],
    "telefono": [
        "This field is required."
    ],
    "correo": [
        "This field is required."
    ]
}
```

Endpoint para modificar el estado de un dueño:

- URL: /api/duenos/{id_dueno}/
- Método HTTP: PATCH
- Parámetros de consulta: “id_dueno”.
- Parámetros de actualización: “status(Active/Inactive)”.

Respuesta exitosa (código de estado 200 OK):

Respuesta de código de error 400 Bad Request:



Mascotas

Endpoint para obtener la lista de todas las mascotas registradas:

- URL: /api/mascotas/
- Método HTTP: GET
- Parámetros de consulta: Ninguno

Respuesta exitosa (código de estado 200 OK):


```
[
    {
        "id_mascota": 1,
        "nombre": "Rodolfo",
        "sexo": "Masculino",
        "raza": "Pitbull",
        "edad": 15,
        "status": "Active",
        "id_dueno": 9
    },
    {
        "id_mascota": 2,
        "nombre": "Kiko",
        "sexo": "Masculino",
        "raza": "Chitzu",
        "edad": 17,
        "status": "Active",
        "id_dueno": 9
    },
    {
        "id_mascota": 3,
        "nombre": "lupe",
        "sexo": "Femenino",
        "raza": "Chihuahua",
        "edad": 4,
        "status": "Active",
        "id_dueno": 9
    }
]
```

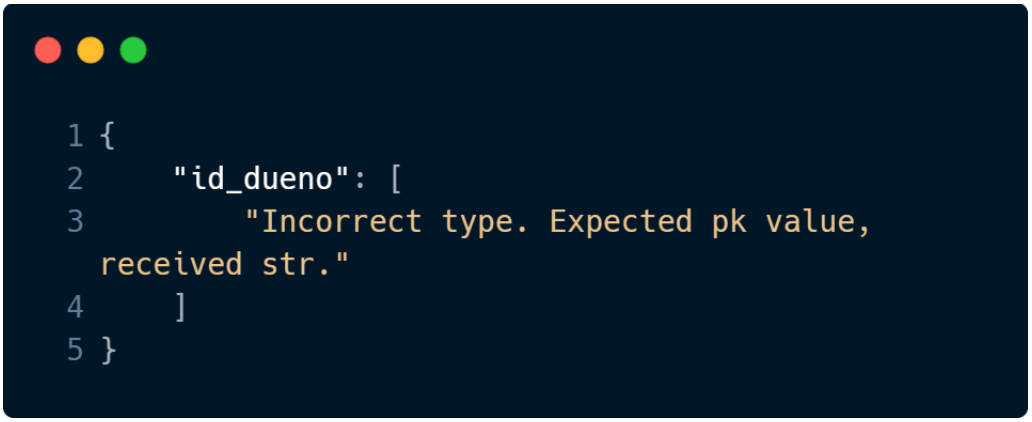
Endpoint para crear una nueva mascota:

- URL: /api/mascotas/
- Método HTTP: POST
- Parámetros de creación: “id_dueno”, “nombre”, “sexo”, “raza”,”edad”, “status(Active/Inactive, default=Active)”.

Respuesta exitosa (código de estado 201 Created):

```
{
    "message": "Mascota creada",
    "Mascota": {
        "id_mascota": 3,
        "nombre": "lupe",
        "sexo": "Femenino",
        "raza": "Chihuahua",
        "edad": 4,
        "status": "Active",
        "id_dueno": 9
    }
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "id_dueno": [
        "Incorrect type. Expected pk value, received str."
    ]
}
```

Endpoint para obtener detalles de una mascota en específico:

- URL: /api/mascotas/{id_mascota}/
- Método HTTP: GET
- Parámetros de consulta: “id_mascota”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_mascota": 3,
    "nombre": "lupe",
    "sexo": "Femenino",
    "raza": "Chihuahua",
    "edad": 4,
    "status": "Active",
    "id_dueno": 9
}
```

Respuesta de código de error 404 Not Found:

```
{
    "detail": "No Mascotas matches the given query."
}
```

Endpoint para actualizar información de una mascota:

- URL: /api/mascotas/{id_mascota}/
- Método HTTP: PUT
- Parámetros de actualización: “id_dueno”, “nombre”, “raza”,”edad”.

Respuesta exitosa (código de estado 200 OK):


```
{
    "id_mascota": 3,
    "nombre": "Patty",
    "sexo": "Femenino",
    "raza": "Chihuahua",
    "edad": 4,
    "status": "Active",
    "id_dueno": 9
}
```


Respuesta de código de error 400 Bad Request:


```
{
    "nombre": [
        "This field is required."
    ],
    "raza": [
        "This field is required."
    ],
    "edad": [
        "This field is required."
    ],
    "id_dueno": [
        "This field is required."
    ]
}
```

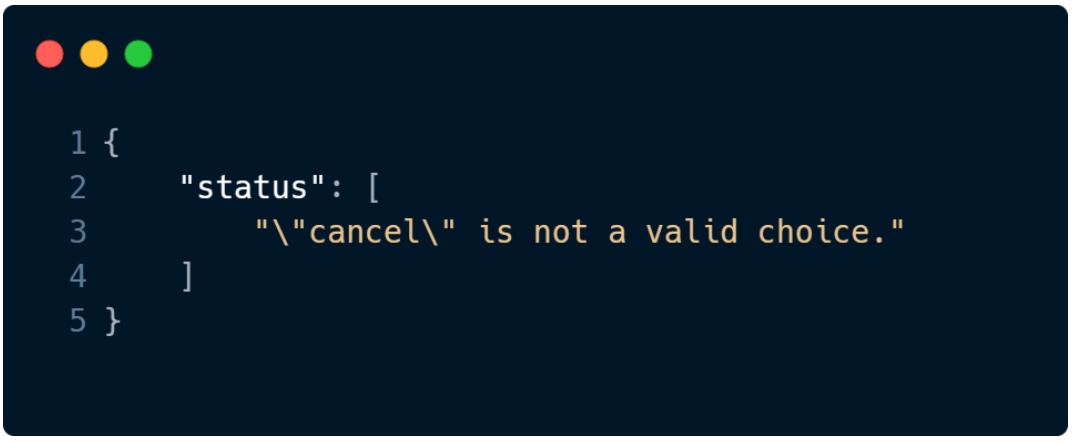
Endpoint para modificar el estado de una mascota:

- URL: /api/mascotas/{id_mascota}/
- Método HTTP: PATCH
- Parámetros de consulta: “id_mascota”.
- Parámetros de actualización: “status(Active/Inactive)”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_mascota": 3,
    "nombre": "Patty",
    "sexo": "Femenino",
    "raza": "Chihuahua",
    "edad": 4,
    "status": "Inactive",
    "id_dueno": 9
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "status": [
        "\"cancel\" is not a valid choice."
    ]
}
```

Citas

Endpoint para obtener la lista de todas las citas:

- URL: /api/citas/
- Método HTTP: GET
- Parámetros de consulta: Ninguno

Respuesta exitosa (código de estado 200 OK):


```
[
    {
        "id_cita": 1,
        "fecha": "2024-04-18",
        "hora": "22:58:00",
        "tipo_cita": "Desparasitación",
        "status": "Done",
        "id_mascota": 3
    },
    {
        "id_cita": 2,
        "fecha": "2024-04-18",
        "hora": "22:58:00",
        "tipo_cita": "Desparasitación",
        "status": "Done",
        "id_mascota": 3
    },
    {
        "id_cita": 3,
        "fecha": "2024-04-18",
        "hora": "22:58:00",
        "tipo_cita": "Desparasitación",
        "status": "Done",
        "id_mascota": 3
    }
]
```

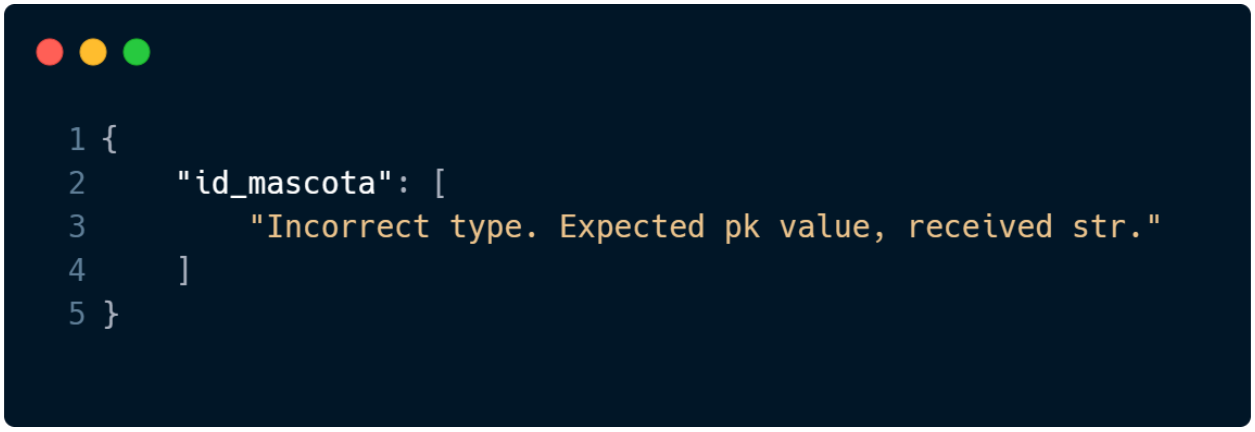
Endpoint para crear una nueva cita:

- URL: /api/citas/
- Método HTTP: POST
- Parámetros de creación: “id_mascota”, “fecha”, “hora”, “tipo_cita”, “status(Done/Canceled, default=Done)”.

Respuesta exitosa (código de estado 201 Created):

```
{
    "message": "Cita creada",
    "Cita": {
        "id_cita": 1,
        "fecha": "2024-04-18",
        "hora": "22:58:00",
        "tipo_cita": "Desparasitación",
        "status": "Done",
        "id_mascota": 3
    }
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "id_mascota": [
        "Incorrect type. Expected pk value, received str."
    ]
}
```

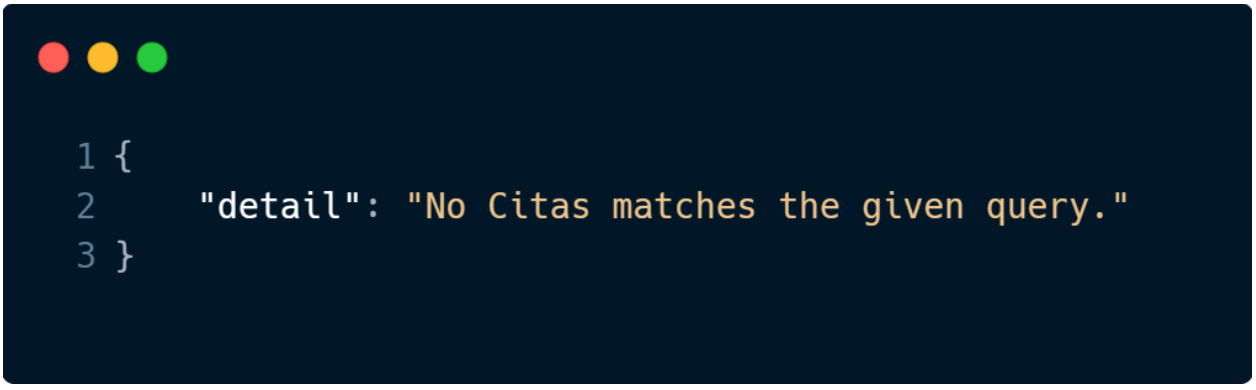
Endpoint para obtener detalles de una cita en específico:

- URL: /api/citas/{id_cita}/
- Método HTTP: GET
- Parámetros de consulta: “id_cita”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_cita": 1,
    "fecha": "2024-04-18",
    "hora": "22:58:00",
    "tipo_cita": "Desparasitación",
    "status": "Done",
    "id_mascota": 3
}
```

Respuesta de código de error 404 Not Found:



```
{
    "detail": "No Citas matches the given query."
}
```

Endpoint para actualizar información de una cita:

- URL: /api/citas/{id_cita}/
- Método HTTP: PUT
- Parámetros de actualización: “id_mascota”, “fecha”, “hora”, “tipo_cita”,

Respuesta exitosa (código de estado 200 OK):


```
{
    "id_cita": 1,
    "fecha": "2024-04-24",
    "hora": "23:58:00",
    "tipo_cita": "Desparacitación de pulgas",
    "status": "Done",
    "id_mascota": 3
}
```


Respuesta de código de error 400 Bad Request:


```
{
    "fecha": [
        "This field is required."
    ],
    "hora": [
        "This field is required."
    ],
    "tipo_cita": [
        "This field is required."
    ],
    "id_mascota": [
        "This field is required."
    ]
}
```

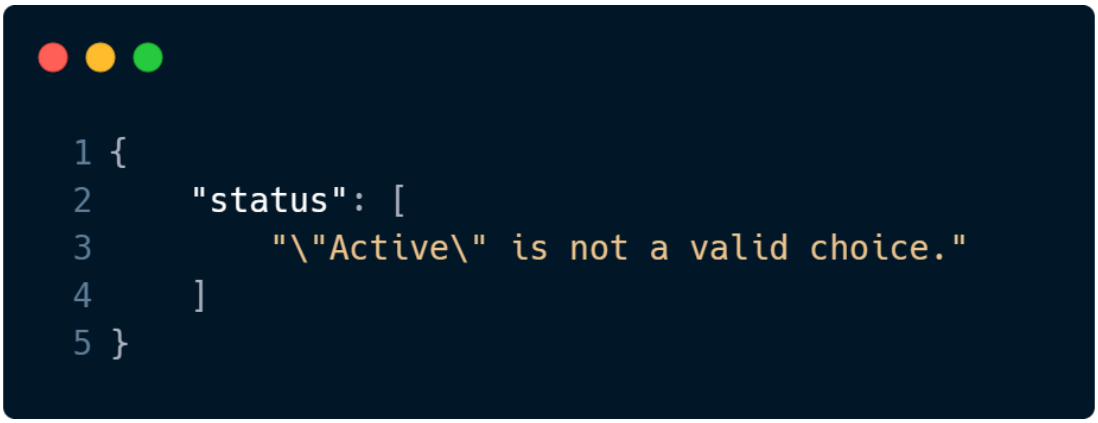
Endpoint para modificar el estado de una cita:

- URL: /api/citas/{id_cita}/
- Método HTTP: PATCH
- Parámetros de consulta: “id_cita”.
- Parámetros de actualización: “status(Done/Canceled)”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_cita": 1,
    "fecha": "2024-04-24",
    "hora": "23:58:00",
    "tipo_cita": "Desparacitación de pulgas",
    "status": "Canceled",
    "id_mascota": 3
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "status": [
        "\"Active\" is not a valid choice."
    ]
}
```

Desparasitaciones

Endpoint para obtener la lista de todas las desparasitaciones:

- URL: /api/desparasitaciones/
- Método HTTP: GET
- Parámetros de consulta: Ninguno

Respuesta exitosa (código de estado 200 OK):


```
[
    {
        "id_desparasitacion": 1,
        "tipo": "Aplicada",
        "fecha": "2024-04-17",
        "status": "Completado",
        "id_mascota": 2
    },
    {
        "id_desparasitacion": 2,
        "tipo": "Aplicada",
        "fecha": "2024-04-17",
        "status": "Completado",
        "id_mascota": 3
    },
    {
        "id_desparasitacion": 3,
        "tipo": "Aplicada",
        "fecha": "2024-04-17",
        "status": "Completado",
        "id_mascota": 1
    }
]
```

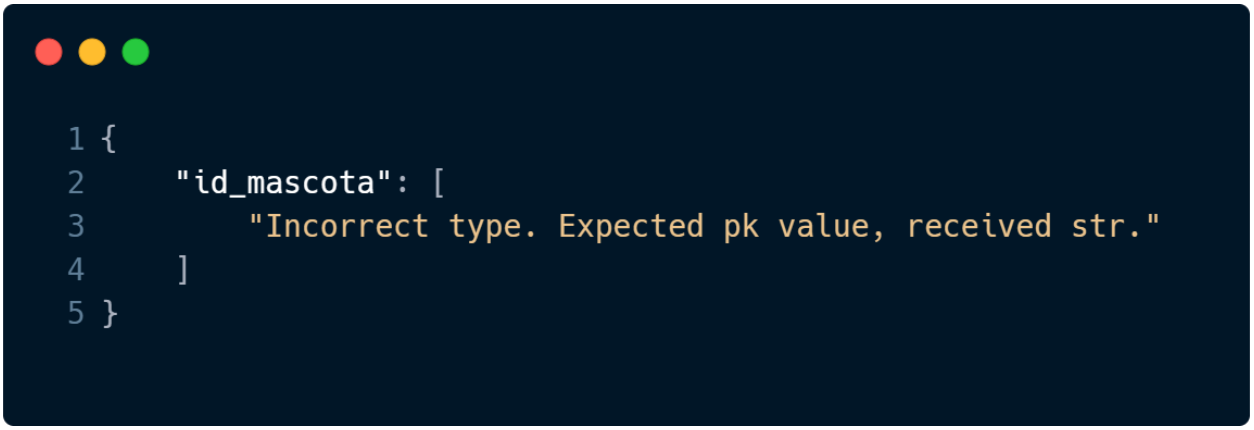
Endpoint para crear una nueva desparasitación:

- URL: /api/desparasitaciones/
- Método HTTP: POST
- Parámetros de creación: “id_desparasitacion”, “tipo”, “fecha”, “status(Completado/No completado, default=Completado)”.

Respuesta exitosa (código de estado 201 Created):

```
{
    "message": "Desparasitación creada",
    "Desparasitación": {
        "id_desparasitacion": 1,
        "tipo": "Aplicada",
        "fecha": "2024-04-17",
        "status": "Completado",
        "id_mascota": 2
    }
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "id_mascota": [
        "Incorrect type. Expected pk value, received str."
    ]
}
```

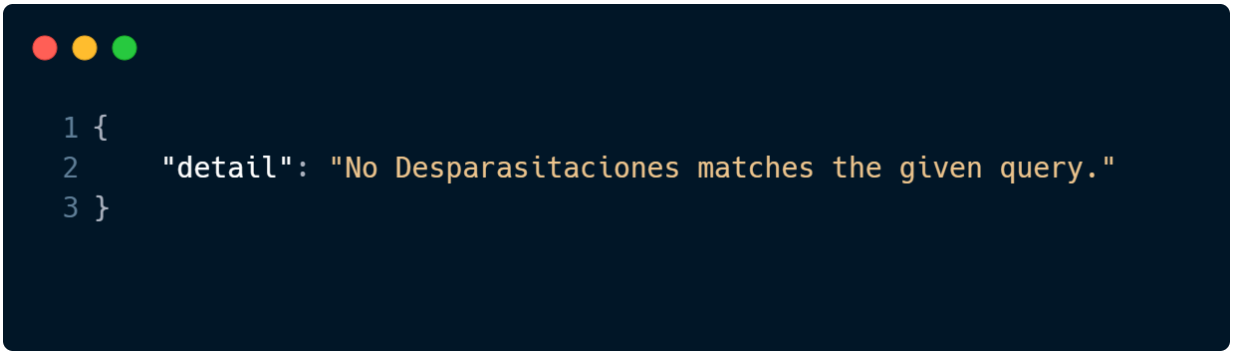
Endpoint para obtener detalles de una desparasitación en específico:

- URL: /api/desparasitaciones/{id_desparasitacion}/
- Método HTTP: GET
- Parámetros de consulta: “id_desparasitacion”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_desparasitacion": 1,
    "tipo": "Aplicada",
    "fecha": "2024-04-17",
    "status": "Completado",
    "id_mascota": 2
}
```

Respuesta de código de error 404 Not Found:



```
{
    "detail": "No Desparasitaciones matches the given query."
}
```

Endpoint para actualizar información de una desparasitación:

- URL: /api/desparasitaciones/{id_desparasitacion}/
- Método HTTP: PUT
- Parámetros de actualización: “id_mascota”, “tipo”, “fecha”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_desparasitacion": 1,
    "tipo": "Inyectada",
    "fecha": "2024-04-22",
    "status": "Completado",
    "id_mascota": 3
}
```

Respuesta de código de error 400 Bad Request:

```
{
    "tipo": [
        "This field is required."
    ],
    "fecha": [
        "This field is required."
    ],
    "id_mascota": [
        "This field is required."
    ]
}
```

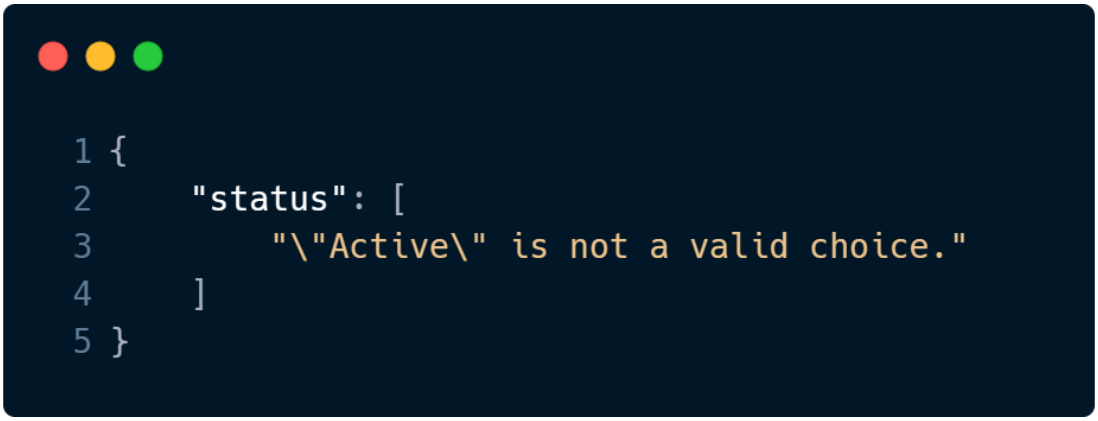
Endpoint para modificar el estado de una desparasitación:

- URL: /api/desparasitaciones/{id_desparasitacion}/
- Método HTTP: PATCH
- Parámetros de consulta: “id_desparasitacion”.
- Parámetros de actualización: “status(Completado/No completado)”.

Respuesta exitosa (código de estado 200 OK):

```
{
    "id_desparasitacion": 1,
    "tipo": "Inyectada",
    "fecha": "2024-04-22",
    "status": "No completado",
    "id_mascota": 3
}
```

Respuesta de código de error 400 Bad Request:



```
{
    "status": [
        "\"Active\" is not a valid choice."
    ]
}
```

Documentación de uso del Sistema

El detalle sobre el uso de cada una de las partes del sistema se muestra a continuación.

Pantalla de Inicio

Esta pantalla funciona como un resumen del estado actual del sistema, en el que se incluye la cantidad de dueños y mascotas registradas, las citas y desparasitaciones registradas. Esto con el fin de que se visualizar un poco el flujo de trabajo que tiene el sistema.

Listado de elementos

Todos los módulos de la aplicación tienen la operación y capacidad de listar, para ello se debe elegir la opción de ver el modulo en específico, dentro del mismo se mostrará una lista de todos los registros disponibles.

Administración de Dueños

Dentro del módulo de Dueños, se tienen las acciones disponibles de crear, leer, editar y eliminar.

Administración de Mascotas

Dentro del módulo de Mascotas, se tienen las acciones disponibles de crear, leer, editar y eliminar.

Administración de Citas

Dentro del módulo de Citas, se tienen las acciones disponibles de crear, leer, editar y eliminar.

Administración de Desparasitaciones

Dentro del módulo de Desparasitaciones, se tienen las acciones disponibles de crear, leer, editar y eliminar.

Requisitos necesarios para publicar aplicación en Google Play

Los pasos a realizar son los siguientes:

1. **Crear una cuenta de desarrollador en Google:** El precio es de 25$, un único pago, sin anualidades, con el que tendrás acceso a publicar tus aplicaciones en Google Play.
2. **Nombre y descripción de la aplicación:** Se debe introducir un nombre y una descripción para la aplicación. El nombre debe ser único y reflejar la esencia de la aplicación.
3. **Logos y Marca:** El logo y marca de la aplicación es una de las tareas más importantes, ya que será el medio por el cual los usuarios reconocerán tu aplicación.
4. **Política de privacidad e información general**: Es importante establecer un limite de edad, para ello se puede iniciar sesión en Google Play Developer Console y completar el cuestionario para cada una de las aplicaciones. Según las características de la aplicación, puede bloquearse para usuarios según su edad o también restringirse en ciertos países. Así mismo se debe agregar un link a una página web para informarle cómo maneja los datos confidenciales de usuarios y dispositivos.
5. **Cargar la aplicación:** Carga el archivo de tu aplicación compilada en formato ABB (.abb). Esto es necesario para ejecutar tu aplicación en un dispositivo Android. El formato APK ha quedado en desuso, ya que ABB es un 15% más pequeño y permite descargar más rápido.
6. **Revisión:** Si la aplicación pasa la revisión, estará disponible en Google Play Store en unas pocas horas o varios días. algunos de los motivos de rechazo por parte de Google Play son: Contenido restringido, Propiedad intelectual robada, Cuestiones de seguridad y privacidad, Anuncios, Errores en la aplicación.

Requisitos necesarios para publicar aplicación en AppStore

Los pasos a realizar son los siguientes:

1. **Crear una cuenta de desarrollador en Apple:** Se debe crear una cuenta de desarrollador de Apple y pagar una tarifa de registro anual de 100$. Además de crear algunos elementos adicionales, como el Certificado de distribución, y configurar un ID de aplicación.
2. **Nombre y descripción de la aplicación:** Se debe introducir un nombre y una descripción para la aplicación. El nombre debe ser único y reflejar la esencia de la aplicación.
3. **Logos y Marca:** El logo y marca de la aplicación debe deslumbrar, aunque hay menos aplicaciones que en Google, los usuarios de Apple són más exigentes.
4. **Información adicional y política de seguridad**: Se debe tener en cuenta que, igual que en Webs tenemos el SEO para posicionar, en las tiendas de aplicaciones debes tener presente el ASO. Para ello se deben emplear palabras clave, URL de soporte y marketing, Precios, categoría y límite de edad.
5. **Publicación en Test Flight:** Enviar la aplicación para la prueba beta en TestFlight para asegurarse de que la aplicación cumpla con todas las pautas de la tienda de aplicaciones de Apple. Esto llevará unas 24 horas a una semana para revisar la aplicación antes de su publicación o rechazo. Algunos de los motivos de rechazo por parte de Apple App Store son: Múltiples errores, Contenido inacabado, Interfaz de usuario pobre, Enlaces rotos, Duplicación de aplicaciones, Baja calidad y poca cantidad de características.

Referencias

- Requisitos para publicar aplicaciones en Google Play y App Store en 2021. (2021, 7 noviembre). AppMaster - Ultimate All-in No-code Platform. https://appmaster.io/es/blog/requisitos-para-publicar-aplicaciones-en-google-play-y-app-store-en-2021
- Vergara, S. (2022, 15 marzo). Guía y requisitos para publicar APPs en Google Play y App Store. Blog ITDO - Agencia de Desarrollo Web, APPs y Marketing En Barcelona. https://www.itdo.com/blog/guia-y-requisitos-para-publicar-apps-en-google-play-y-app-store/

Enlaces

Repositorio GitHub Proyecto Django: https://github.com/wilsoncar/MascotasFelices

Repositorio GitHub Proyecto React Native: https://github.com/wilsoncar/MascotasFelicesApp

Repositorio Docker:https://hub.docker.com/repository/docker/wilsonudv/mascotasfelices/general