**Instituto Tecnológico y de Estudios**

**Superiores de Occidente – ITESO**

**Departamento de Electrónica, Sistemas e Informática**



**Proyecto Final – “Villa Kebo”**

**Primera Entrega: Documentación**

**Materia:** Tecnologías de Desarrollo en el Servidor

**Maestro:** Sevilla Medina Francisco Javier

**Autores:**

González García Cinthya Lorena

Johannsen Miquelajauregui Iker

Rodríguez Von Hauske Daniel

**Fecha:** 27/02/2024

**1. Título:** Villa Kebo

**2. Descripción**

El proyecto consiste en desarrollar la página web del hotel para mascotas “Villa Kebo”, en donde los dueños pueden reservar una estancia para su mascota (inicialmente sólo animales domésticos como perros y gatos). Asimismo, los cuidadores proveerán actualizaciones acerca del cuidado de la mascota y le llegarán notificaciones al respectivo dueño.

**3. Integrantes**

* Iker Johannsen Miquelajauregui
* Cinthya Lorena González García
* Daniel Rodríguez Von Hauske

**4. Stack de tecnologías a utilizar**

**Librerías:**

* Bcrypt y Jwt
* DotEnv
* Mongoose
* Swagger

**Frameworks**

* Express
* Bootstrap o Vue.js (dependiendo de la complejidad del segundo)

**Base de datos**

* MongoDB
* Servicio S3 de AWS

**Servicios de terceros**

* Google Passport (auth)

**5. Listar y describir los roles de usuarios**

**Guest:** Un usuario que tenga el rol de *guest*, es aquel que no tiene una cuenta, este usuario puede navegar dentro de la página web, ver información acerca de Villa Kebo, puede iniciar sesión o crear una cuenta. Es decir, sólo podrá ver la vista del index. No será un rol determinado en el backend del proyecto, sino un usuario sin autenticar.

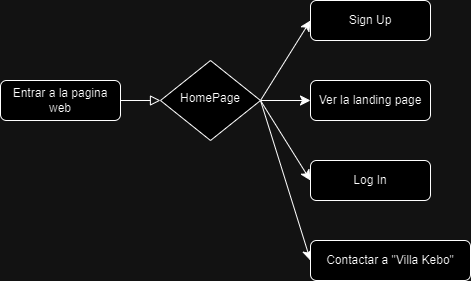
**Owner:** Un usuario que tenga el rol de owner, es aquel que ya está registrado dentro de Villa Kebo. Este usuario ya creó su cuenta y ha llenado la información necesaria para que pueda hacer uso de los servicios de Villa Kebo. Este usuario tiene las mismas capacidades que un guest, pero también puede agendar una estancia para llevar a su perro al hotel, así como gestionar los archivos (como su expediente o cuidados especiales) que ha subido para su mascota. En caso de que tenga una mascota en el hotel, puede interactuar con los caretakers para saber cómo esta su mascota, entre otras cosas.

**Caretaker:** Un caretaker es un usuario el cual tiene un rol de empleado dentro de Villa Kebo. Los caretakers tienen la funcionalidad básica de la aplicación, pero encima de esto, ellos tienen acceso a la información de todas las mascotas registradas en Villa Kebo que pertenezcan a su “grupo de cuidado” y pueden interactuar con los clientes al enviarles notificaciones del cuidado de sus mascotas (al completar tareas que correspondan a su rutina, por ejemplo).

**Admin:** El administrador tiene acceso a toda la información almacenada en la base de datos, puede editar la información y aprueba o deniega las nuevas cuentas de Caretakers creadas en el Hotel.

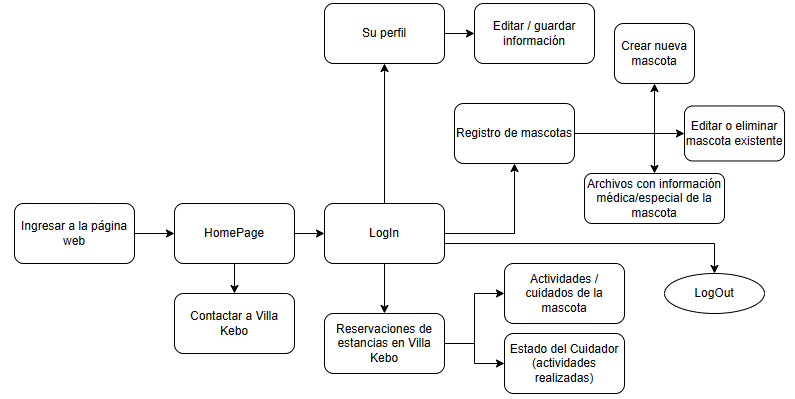
**6. Tareas o flujos principales de cada usuario**

***Tareas Guest***



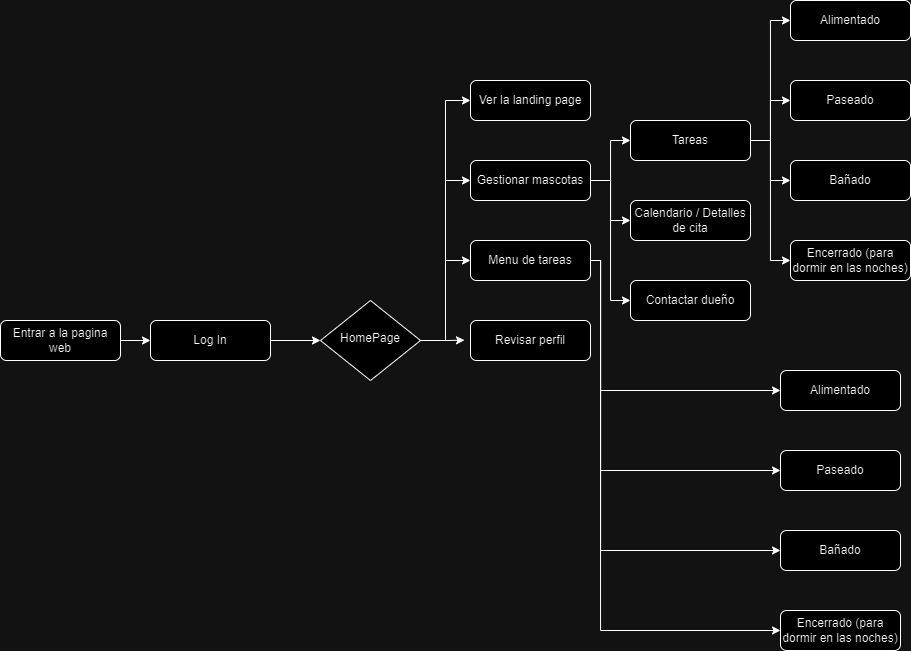
**1.** Diagrama de tareas principales de un usuario de tipo "Guest".

***Tareas Owner***



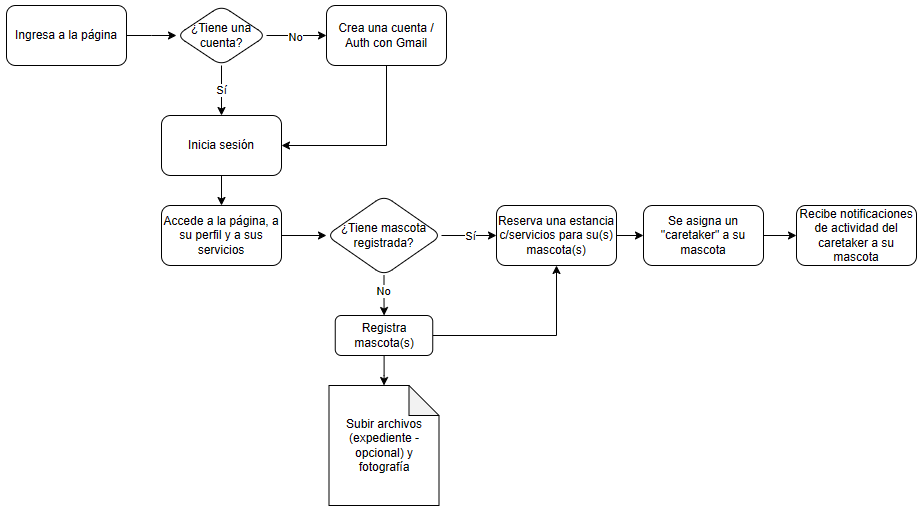
**2.** Diagrama de tareas principales de un usuario "Owner".

***Flujo Caretaker***



**3.** Diagrama de tareas principales del "Caretaker".

**7. Diagramas de flujo de la aplicación**

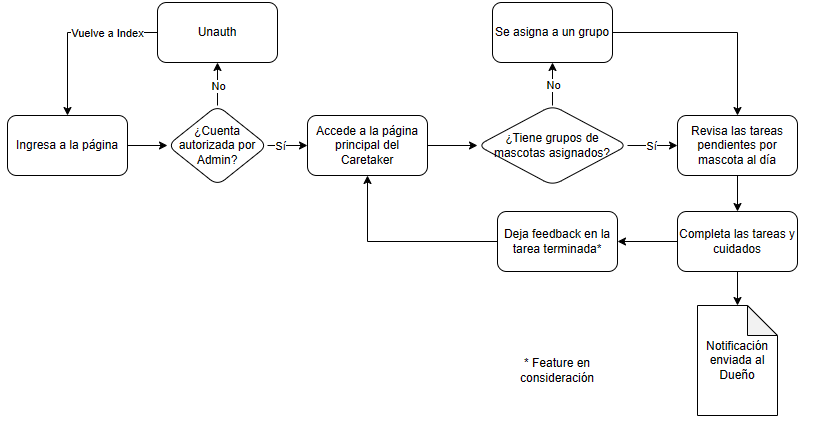


**4.** Diagrama de flujo básico de un "Owner".

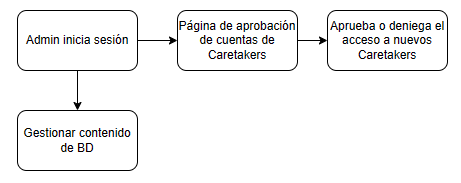
El diagrama 4 muestra, a grandes rasgos, cómo es que un Dueño sigue un proceso de inicio de sesión (auth) o registro en la página. Después, accede a su perfil y a sus servicios disponibles. Si tiene mascotas registradas, entonces puede realizar una reservación en el Hotel. Si no, deberá registrar al menos una para poder realizar esa acción. Al registrar una mascota, también deberá de subir ciertos archivos necesarios, como la fotografía de ésta. De forma opcional, podrá subir archivos con el expediente médico de la mascota si necesita cuidados especiales u otras indicaciones.

Una vez que se tiene una mascota registrada, se puede crear una reservación y ésta contendrá las actividades y cuidados que deberán ser otorgados por el personal del Hotel (Cuidador que se asigna) y a qué grupo se asignó la mascota. Cuando el Cuidador lleve a cabo tareas asignadas, se notificará al Dueño.

Antes de que un Cuidador pueda iniciar sesión, debe haber creado una cuenta (consideramos que este tipo de cuenta no necesariamente deberá estar asociada a un Auth de Google, pero puede cambiar) que debe ser aprobada por un Admin. Si se tiene, entonces accede a su página principal donde puede ver los grupos de mascotas que se le han asignado bajo su cuidado. Dentro de cada grupo, deberá completar las tareas y/o cuidados de cada mascota. Cuando las complete, las marcará como “terminadas” y tendrá la opción de dejar “feedback” en esa tarea. Ambos se le notificarán al Dueño.



**5.** Diagrama de flujo básico de un "Caretaker".

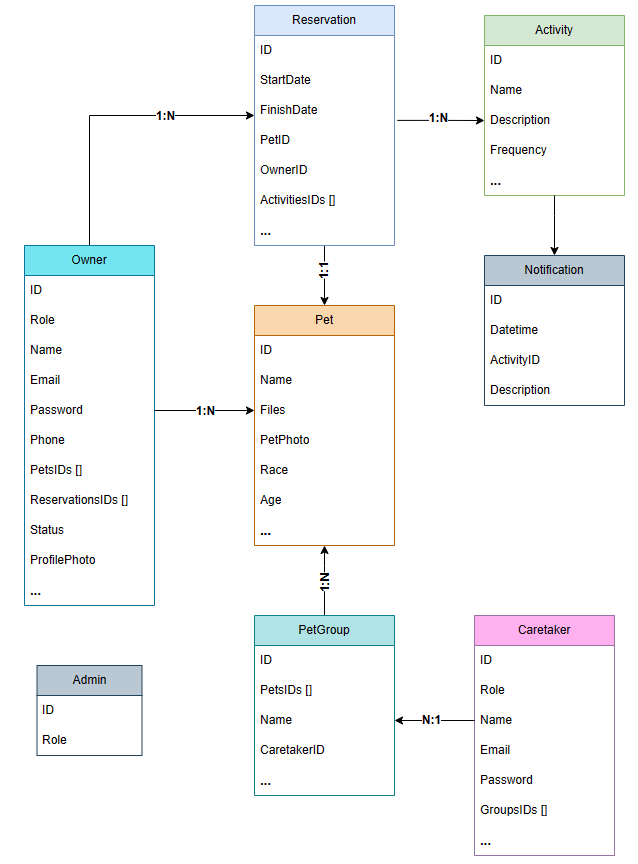


**6.** Diagrama de flujo básico de "Admin".

Un Administrador es aquel que puede acceder al contenido de la base de datos. Además de gestionarla, su rol principal es aprobar o no las cuentas para nuevos Caretakers.

**8. Acciones que se realizarán para cada entidad**

|  |  |  |
| --- | --- | --- |
| **Owner** | **Caretaker** | **Activity** |
| * Iniciar sesión con Google * Ver y editar perfil (datos personales) * Agregar, editar o eliminar mascotas * Agregar, editar o eliminar archivos de mascotas * Crear reservaciones * Asignar mascotas a reservaciones creadas * Crear actividades en reservaciones | * Auth con email * Ver y editar perfil (datos personales) * Tomar (asignarse) un grupo de mascotas * Ver detalles de las mascotas asignadas en cada grupo * Completar tareas o cuidados * Dejar comentarios en la tarea completada | * Owner y Caretaker pueden ver sus detalles y a qué mascota pertenece * Completada o no * Comentario del cuidador |
| **Pet** | **Reservation** | **Notification** |
| * Caretaker puede ver datos y archivos; Owner puede verlos y editarlos | * Owner puede crear, editar y eliminar reservaciones * Caretaker y Owner pueden ver detalles * Contiene tareas asignadas por mascota | * Owner puede verla cuando una tarea se completa o se deja un comentario del cuidador |
| **Pet Group** | **Admin** |  |
| * Owner puede ver los grupos a los que pertenece su mascota * Caretaker puede ver los grupos existentes * Contiene hasta tres mascotas con características similares | * Aprobar o eliminar/denegar cuentas de Caretakers |  |

**9. Diagrama Entidad-Relación**

**10. Alcance en la guía de aprendizaje**

* **Passport:** se manejará, al menos, la sesión de los Owners por medio de autenticación con Google.
* **Imágenes:** se implementará un bucket de S3 donde se guardarán tras haberlas subido por un Owner.
* **Encriptación:** generación de tokens encriptados con herramientas como Bcrypt.
* **Hash:** las contraseñas de los usuarios serán guardadas en la base de datos sólo tras ser hasheadas.
* **Archivos:** uso de Multer para serializar y procesar la subida de archivos con Express y JS.

**SEGUNDA ENTREGA**

**11. Diagrama de Secuencia**

El siguiente diagrama de secuencia muestra el flujo que un usuario del tipo “Owner” sigue para poder registrar una mascota (Pet) a su nombre y, posteriormente, poderla eliminar. Considera solamente el “happy path”, o flujo en donde no hay excepciones o errores de validación por parte del usuario. Estas son funcionalidades que ya se encuentran implementadas en la API.

Diagrama

Descripción generada automáticamente**Repositorio:** <https://github.com/cinthya-g/VillaKebo/tree/dev>

**12. Base de datos**

Una de las cadenas utilizadas es la siguiente (.env):

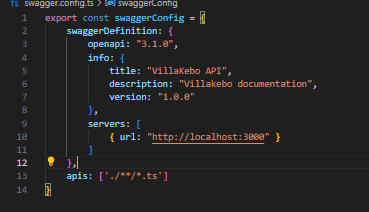
DB = mongodb+srv://villa\_kebo\_user:kebo123@myapp.2j6ykj4.mongodb.net/villa\_kebo?retryWrites=true&w=majority&appName=MyApp

**TERCERA ENTREGA**

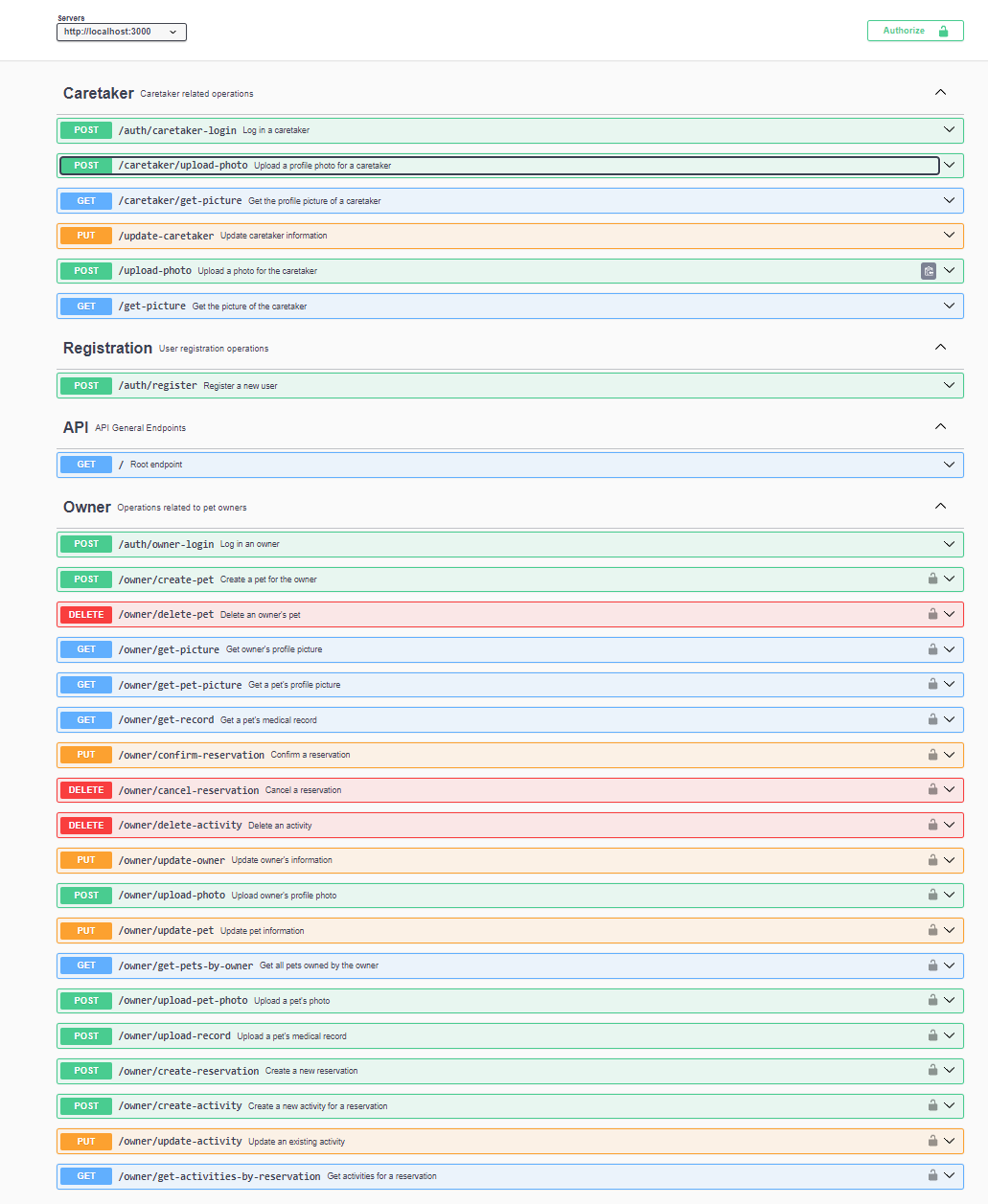
**13. Swagger (Documentacion)**

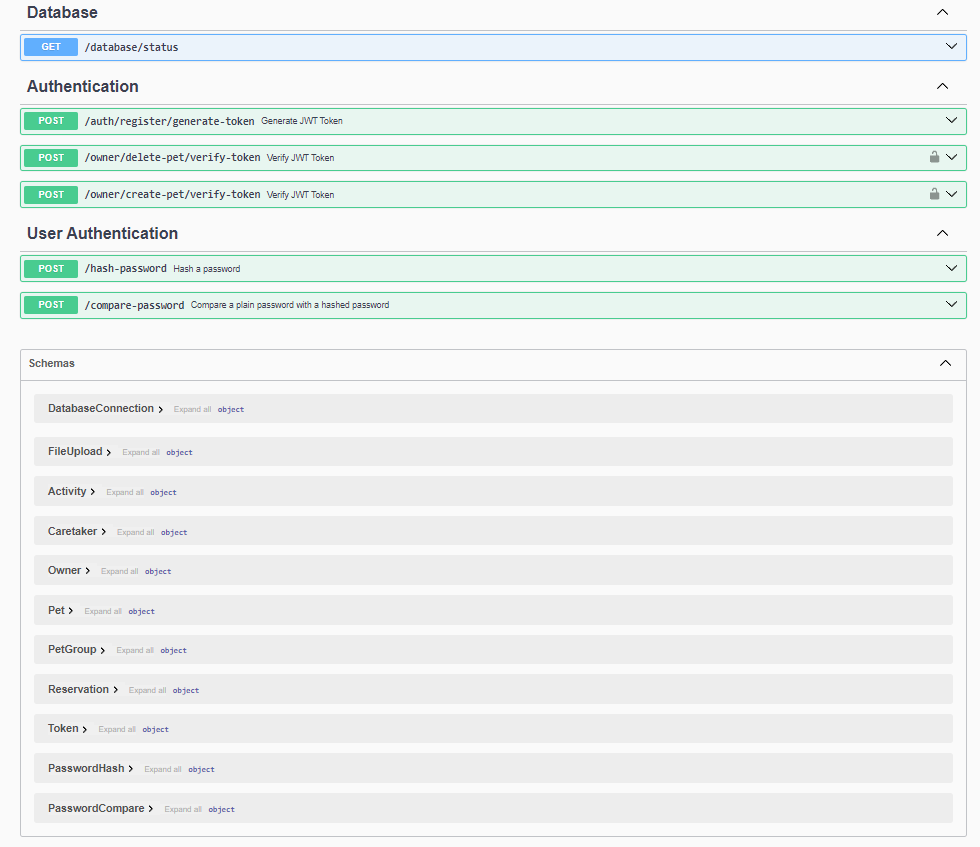
Para facilitar la documentación de nuestra aplicación utilizaremos Swagger el cual nos proporciona con una estructura organizada para la generación de comentarios, así como una pagina web que nos despliega la información de nuestra api de manera organizada y estilizada lo cual hace mas fácil su visualización. Encima de esto dentro de dicha pagina podemos realizar pruebas para ver el funcionamiento de la api.

La configuración se realizo con los mismos pasos con los que se configuro en clase, el archivo de configuración quedo de la siguiente manera:



Finalmente una vez que redactamos los comentarios con el formato que nos indica swagger, obtuvimos la siguiente pagina de api-docs donde podemos visualizar nuestros endpoints.





**14. Authenticacion y Permisos**

Flujo de autenticacion

La autenticación inicia cuando un usuario se registra, proporcionando datos personales y una contraseña que se hashea para su almacenamiento. Posteriormente, al iniciar sesión, se verifica la coincidencia de la contraseña y se genera un token JWT con genToken que encapsula la identidad del usuario. Este token se valida en cada solicitud a través del middleware para autenticar al usuario y se verifica su rol con otro middleware que valida el rol para determinar sus permisos. Diferentes tipos de usuarios, como propietarios o cuidadores, pueden tener rutas y permisos específicos en el sistema, y sus acciones están reguladas por la validación de sus roles y la autenticación a través de sus tokens, asegurando que solo los usuarios autorizados accedan a recursos específicos.

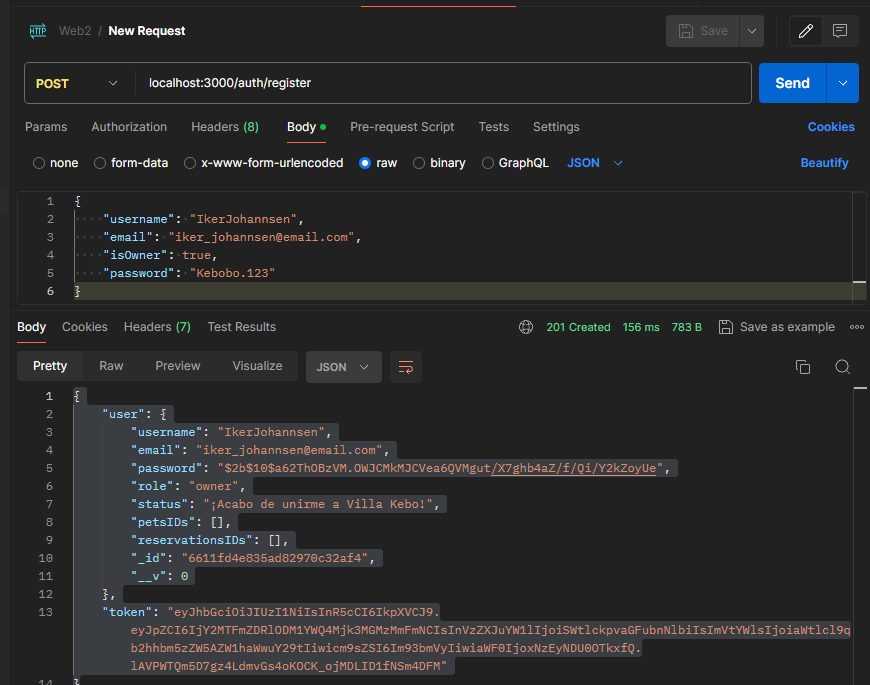
Tenemos dos tipos de usuario con registro y login diferente, sin embargo, en cuestiones de funcionamiento es importante mencionar que al momento de logear un usuario o un caretaker la única diferencia es el rol que se busca validar.

Finalmente un ejemplo de la validación que realizamos seria el siguiente:

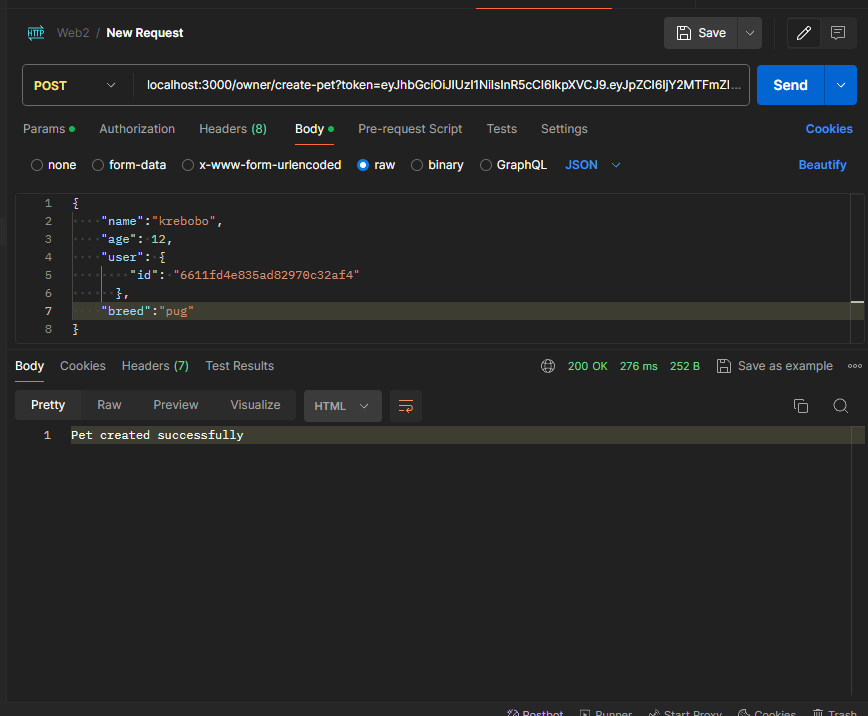
En una operación donde un propietario quiere crear una mascota, primero se verifica su token JWT. Si el token es válido y el rol coincide (por ejemplo, Owner), se le permite acceder a la ruta createPet para registrar una nueva mascota.

Imágenes de evidencia:

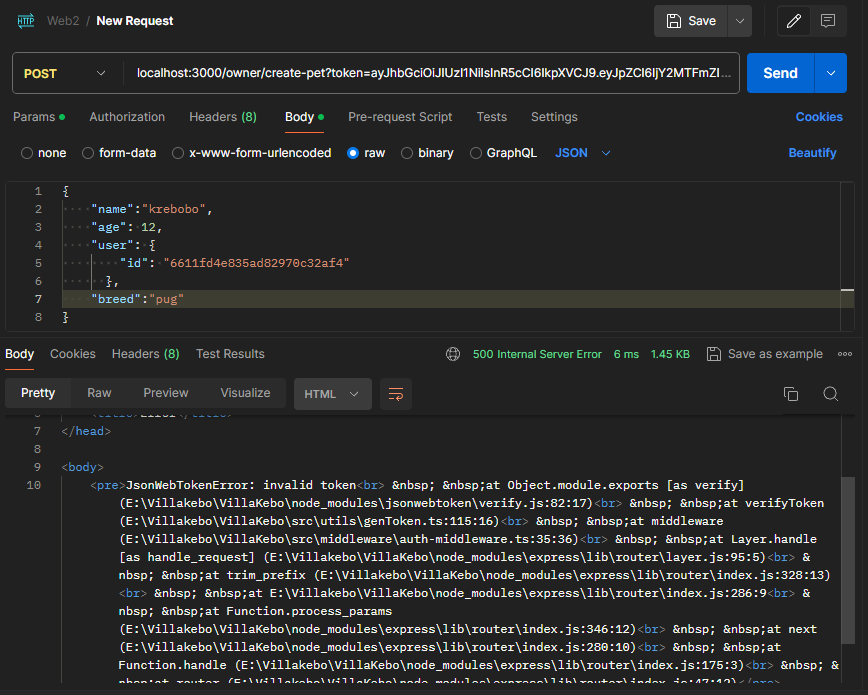
Registro de un usuario



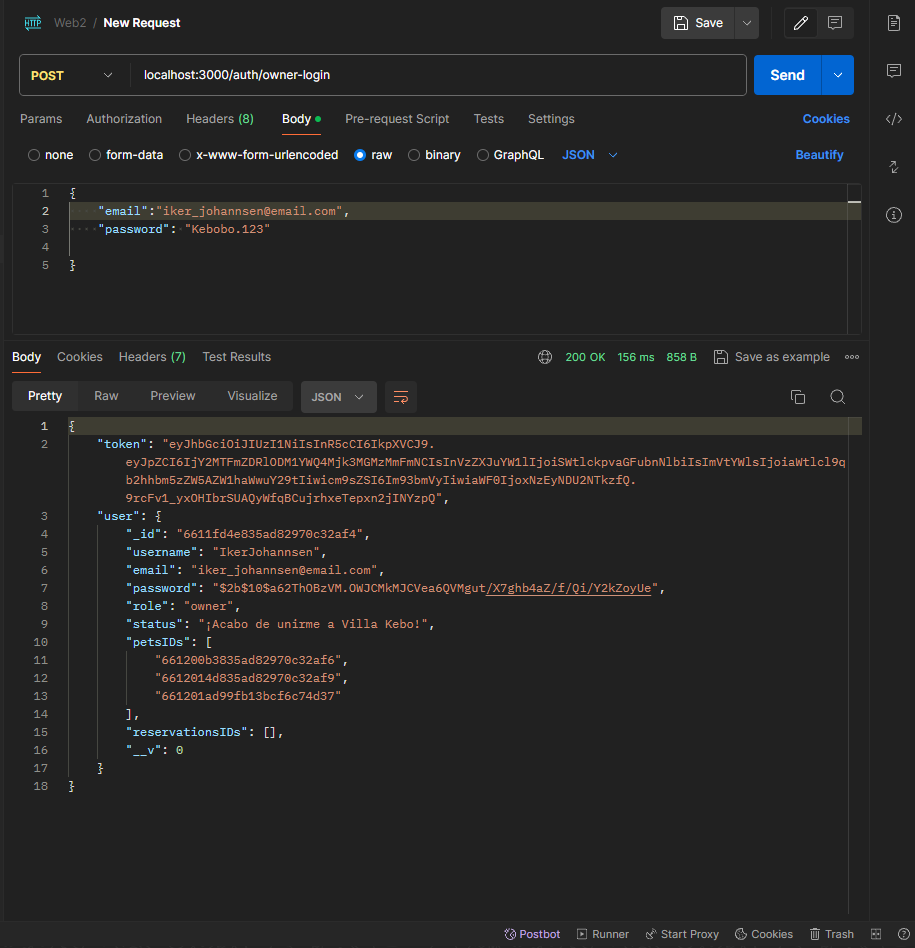
Creación de un pet con token y validación del usuario



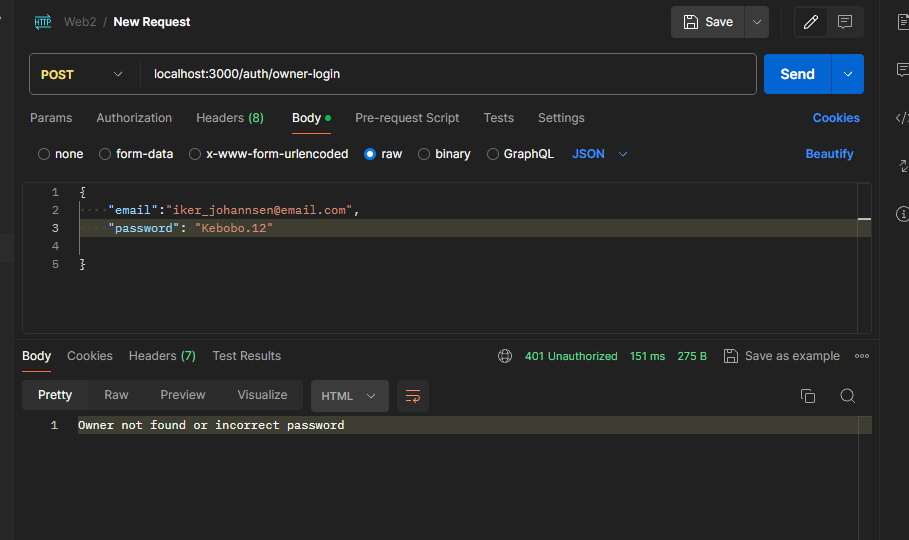
Creación de pet sin un token valido



Log-in con contrasenia correcta:



Log-in con contrasenia incorrecta:



Agrego la información para el .env para que se puedan realizar pruebas:

DB = mongodb+srv://villa\_kebo\_user:kebo123@myapp.2j6ykj4.mongodb.net/villa\_kebo?retryWrites=true&w=majority&appName=MyApp

PORT=3000

SALT=10

TOKEN\_KEY=secret

PUBLIC\_S3\_KEY = AKIA6ODU5RSFMHRO4VWR

SECRET\_S3\_KEY = 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

AWS\_REGION = us-east-1

PHOTOS\_BUCKET\_NAME = vk-profile-photos

FILES\_BUCKET\_NAME = vk-pet-files

ACL\_ACCESS=public-read