Universidad de Nariño

Ingenieria de Sistemas

Diplomado de actualización en nuevas tecnologías para el desarrollo de Software

Taller Unidad 2 Backend

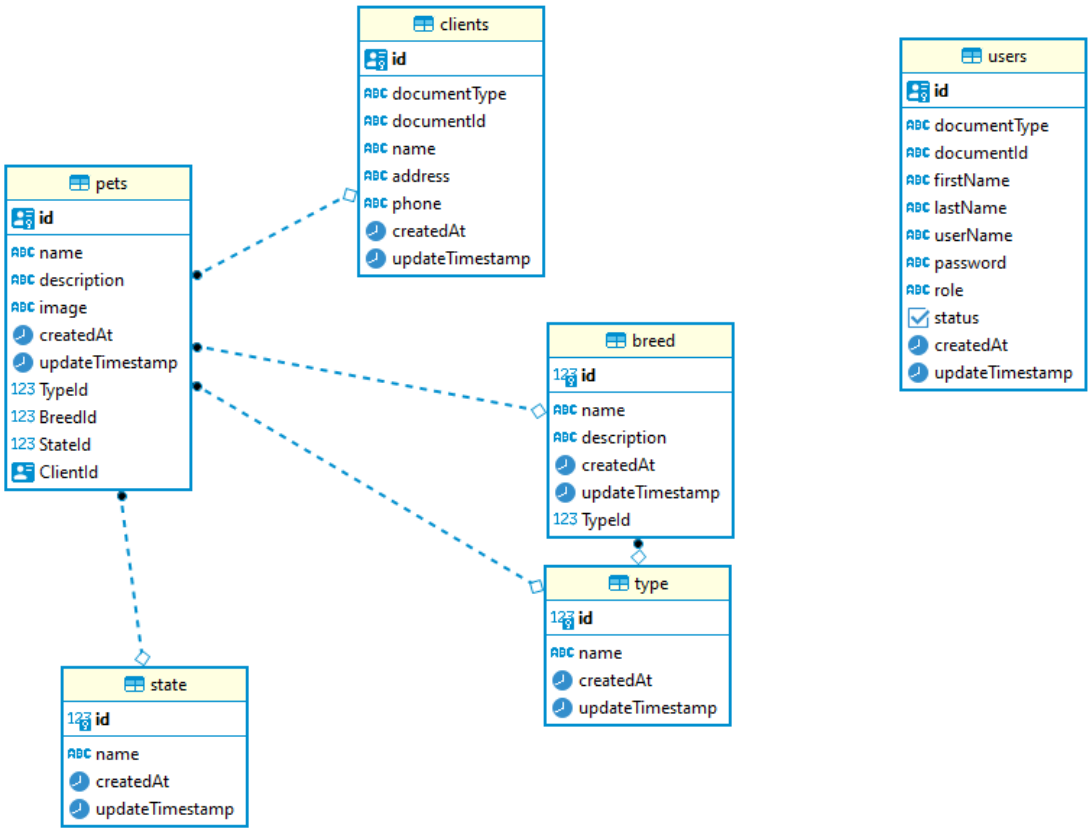
Juan Pablo Chicaiza Muñoz

1085330531

Paul Armando Frias Martinez

1085327469

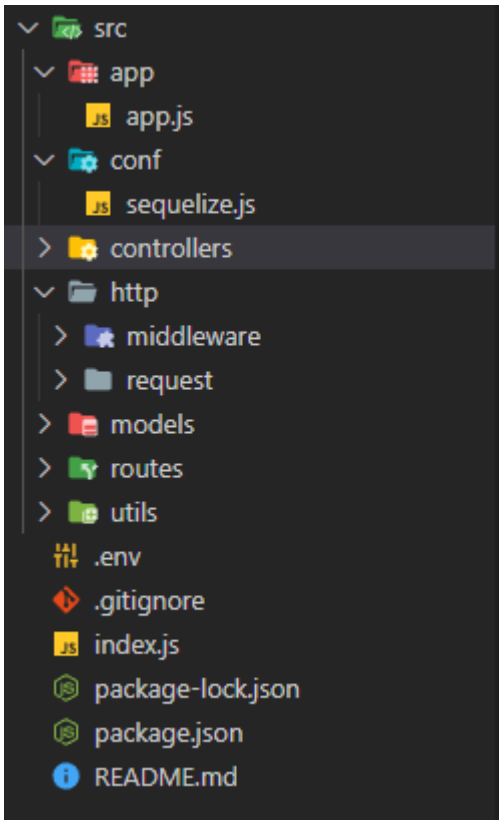
1- Diagrama Entidad Relacion Base de Datos



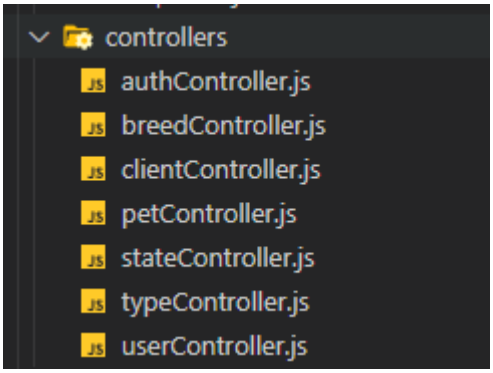
Donde

Pets: Tabla donde se registran las mascotar a ser adoptadas
State: Tabla donde se puede parametrizar el estado de las mascotas (Disponible,Adoptado,etc)
Type: Tabla donde se registran los tipos de mascotas que pueden existir en la tienda (perros,gatos,tortugas,etc)
Breed: Tabla donde se registran las Razas de las mascotas
Clients: Tabla donde se registran los datos basicos de los clientes que adoptan a las mascotas
Users: Tabla donde se registran los usuarios que utilizan la aplicación

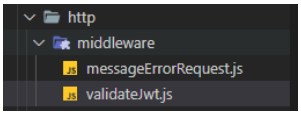
2- Estructura General del Proyecto



- Src: Paquete donde se Encuentra todo el codigo Fuente del Proyecto
    - App: Contiene el archivo app.js, donde se encuentra la configuracion de express
    - Conf: Contiene el archivo sequelize.js, donde se encuentra la configuracion de sequelize, ORM utitlizado para el proyecto
    - Controllers: Contiene todos los controladores del proyecto



    - http: Contiene 2 carpetas.
        - Middleware:

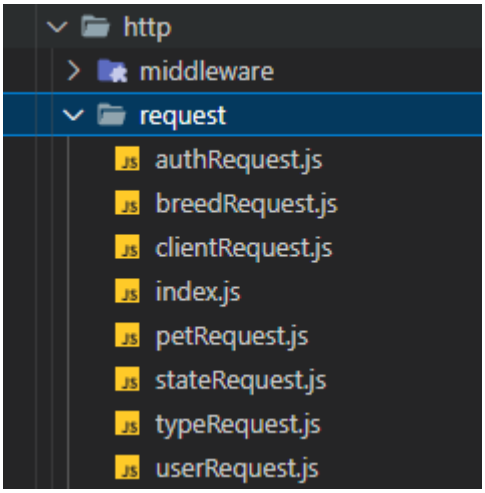


Donde MessageErrorRequest.js utiliza express-validator para realizar validaciones de las peticiones hacia las rutas. Si hay un error este arroja una respuesta con status 400(BAD REQUEST) y un array con los errores encontrados por express-validator

Donde validateJwt.js se encarga de verificar si las peticiones que se realizan cuentan con un token JWT, si el token es valido continua con las peticiones. Si el token es invalido o no existe arroja una respuesta 401(Unauthorized) informando del error

```
{
    "status": 401,
    "type": "error",
    "error": {
        "title": "Unauthorized",
        "msg": "Token Invalido",
        "param": "Authorization",
        "location": "header"
    }
}
```

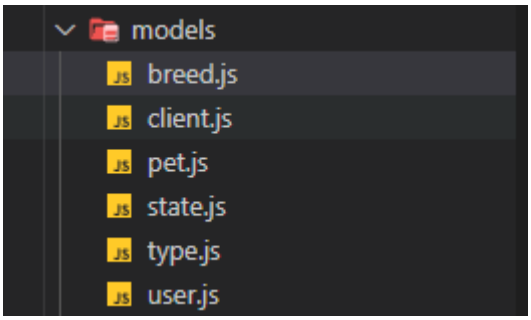
- Request:



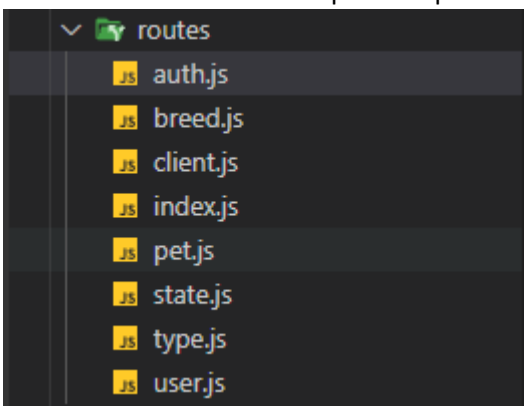
Contiene todas las validaciones correspondientes de cada una de las peticiones entrantes, estas validaciones se realizan con express-validator para validar el requestBody de las peticiones y tambien aquí se manda a llamar la funcion de evaluar el Token JWT
Ejemplo:

```
import { check, param } from "express-validator";
import MessageErrorRequest from "../middleware/messageErrorRequest.js";
import { validateJwt } from "../middleware/validateJwt.js";

const createBreedRequest = [
    validateJwt,
    check('name','Nombre Requerido').not().isEmpty(),
    check('description','Descripcion Requerida').not().isEmpty(),
    check('typeId','Tipo de Mascota Requerido').not().isEmpty(),

    MessageErrorRequest
]
```

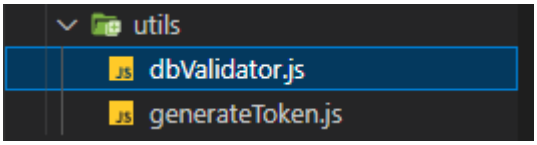
- Models: Contiene los modelos utilizados en la aplicación. Tiene relacion con el diagrama entidad-relacion del punto 1



- Routes: Contiene las rutas que se exponen al consumidor final

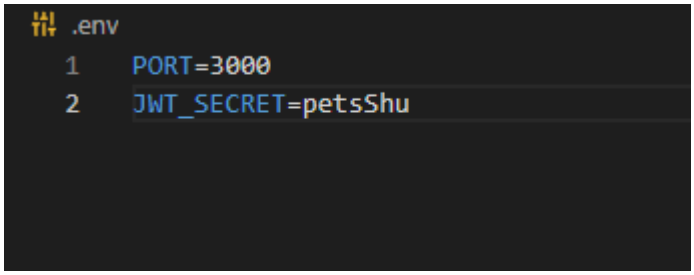


- Utils: Contiene funciones que apoyan a la logica del proyecto



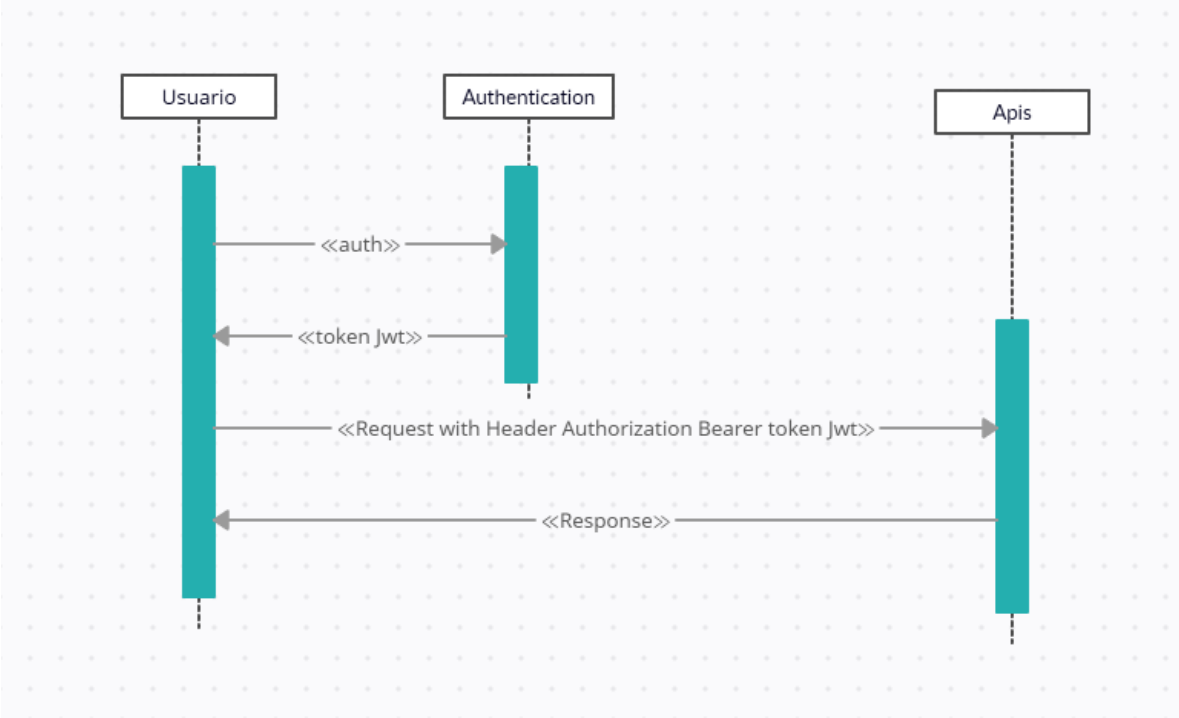
dbValidator: Verifica si existe clientes,usuarios, etc
generateToken: Se encarga de Generar el Token cuando un usuario se autentica. Utiliza jsonwebtoken

- .env: Archivo donde se guardan las variables de configuracion del proyecto



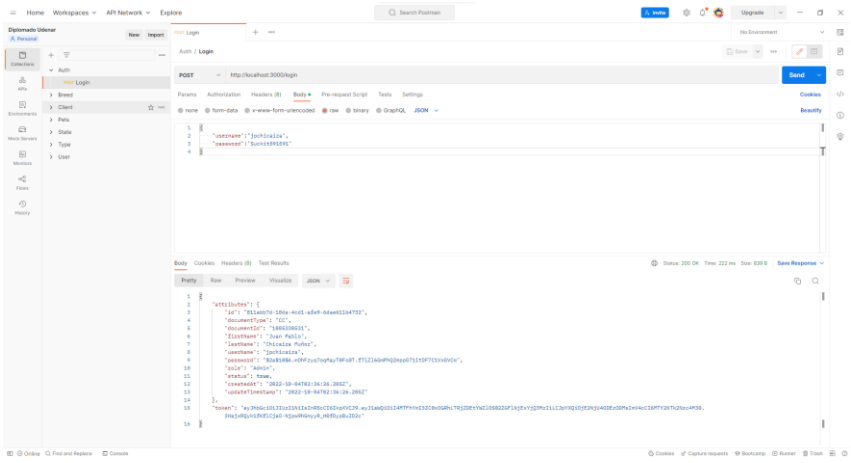
- index.js: Archivo donde se Inicializa la Aplicación

Flujo para consumir las funcionalidades del backend



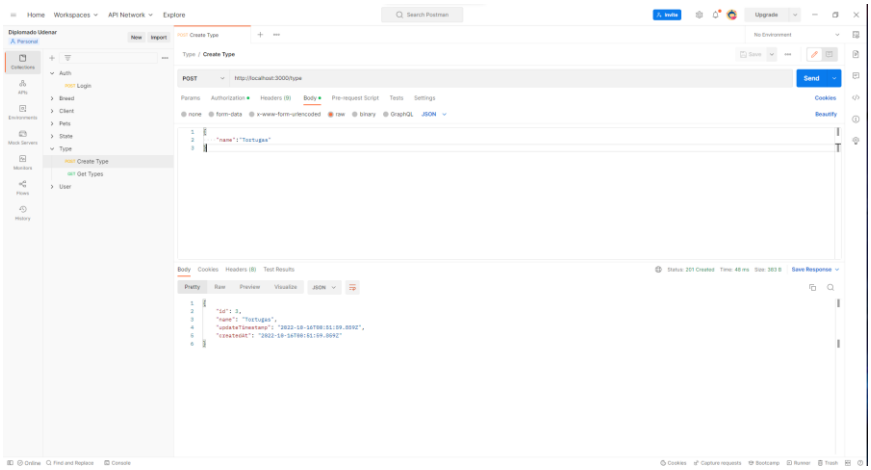
3-

- Login

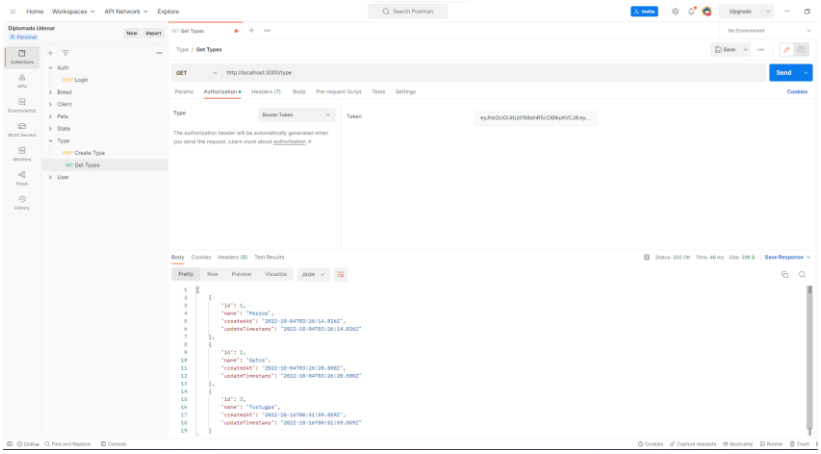


- Type (Tipo de Mascota)

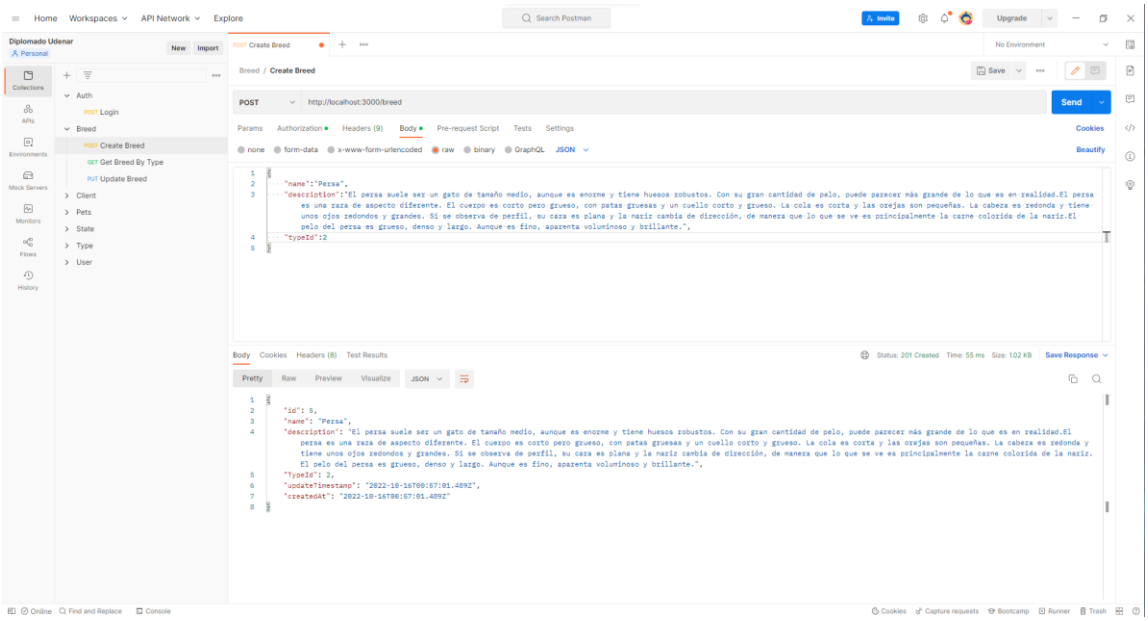
  - Create Type



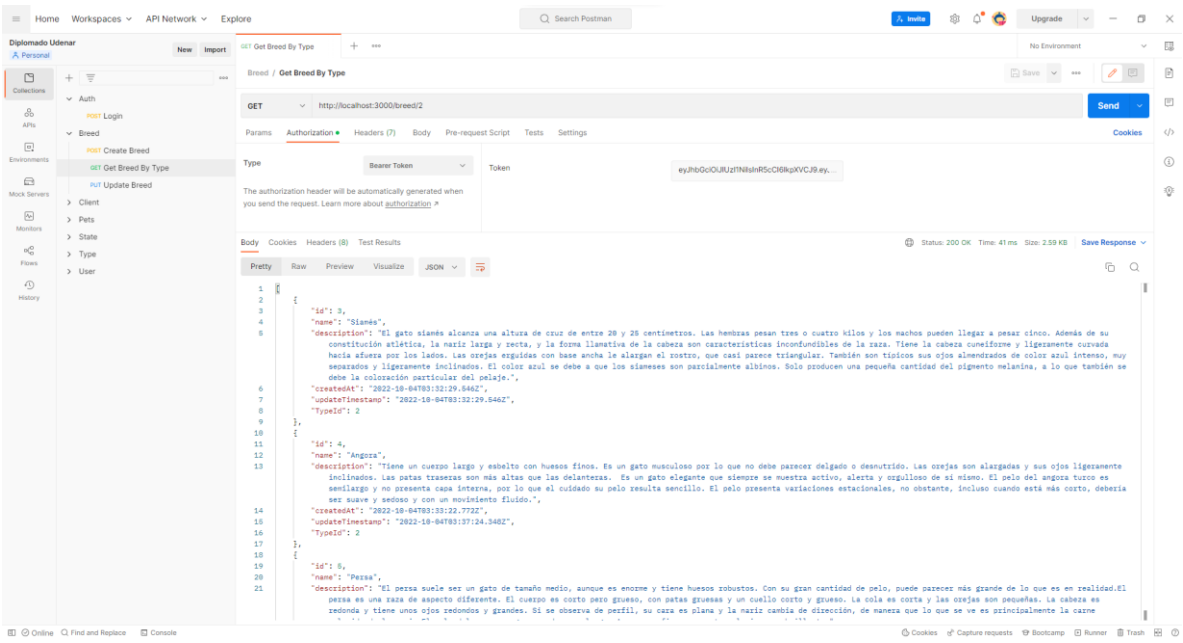
  - Get All Types



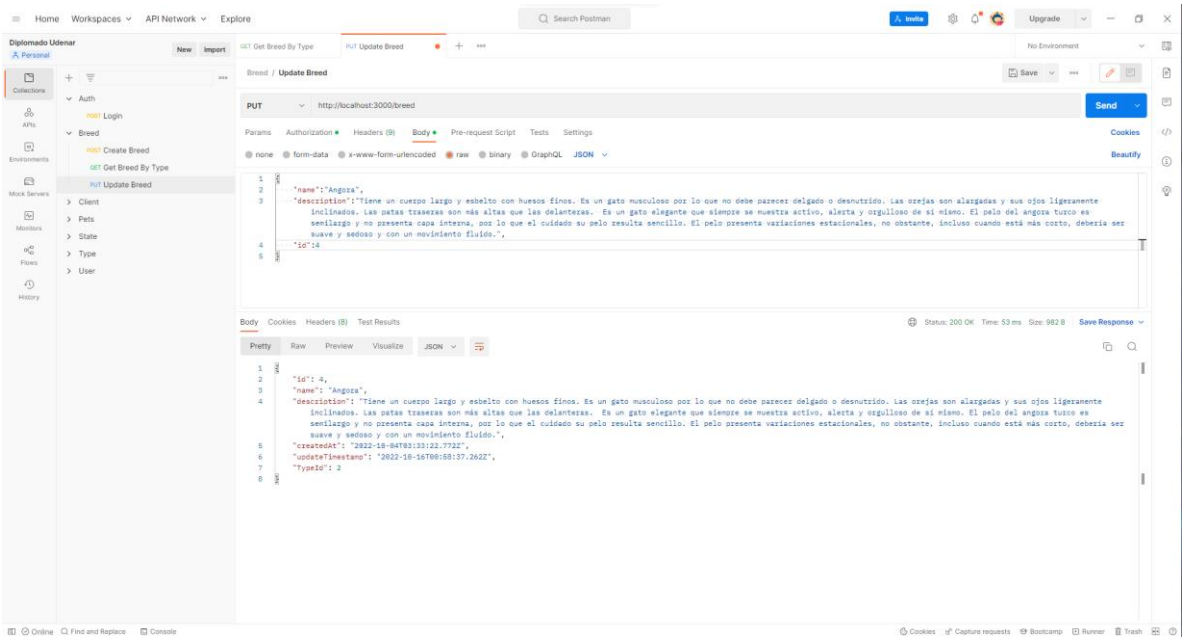
- Breed (Raza)
    - Create Breed



    - Get Breed By Type

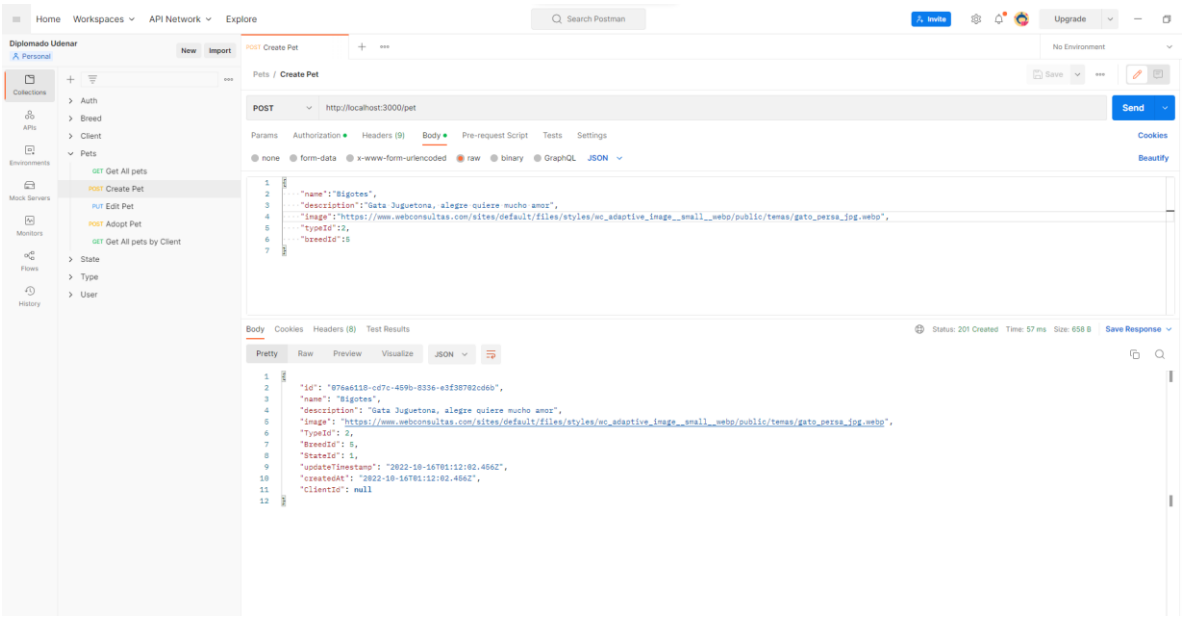


    - Update Breed

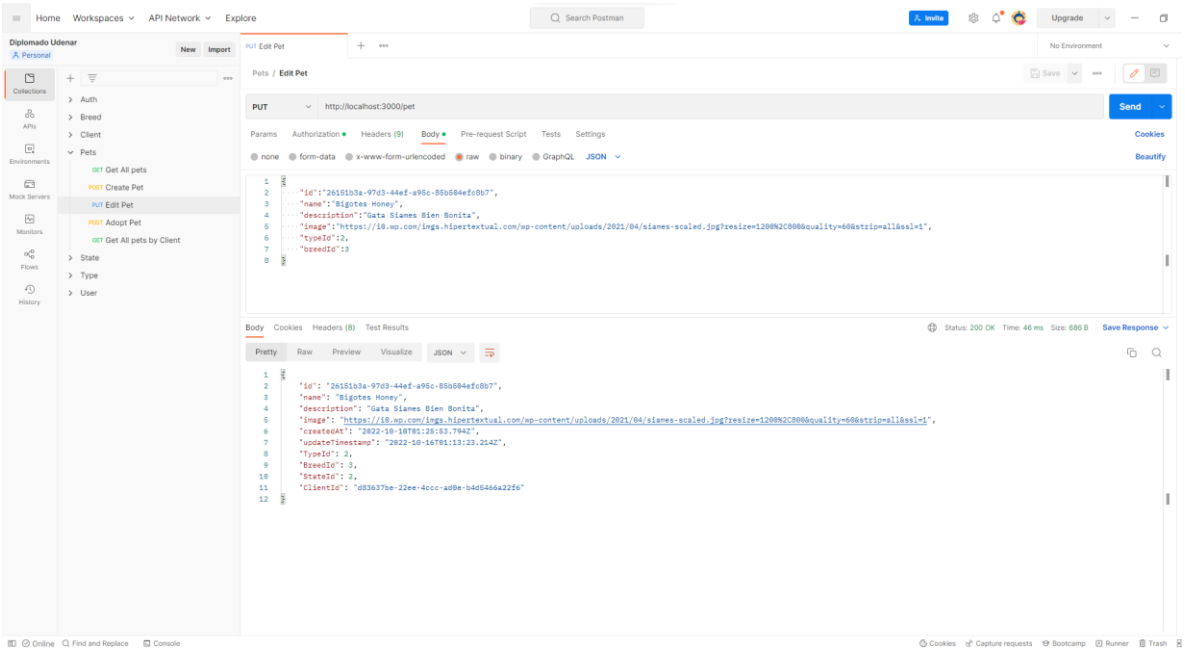


- Pet (Mascotas)

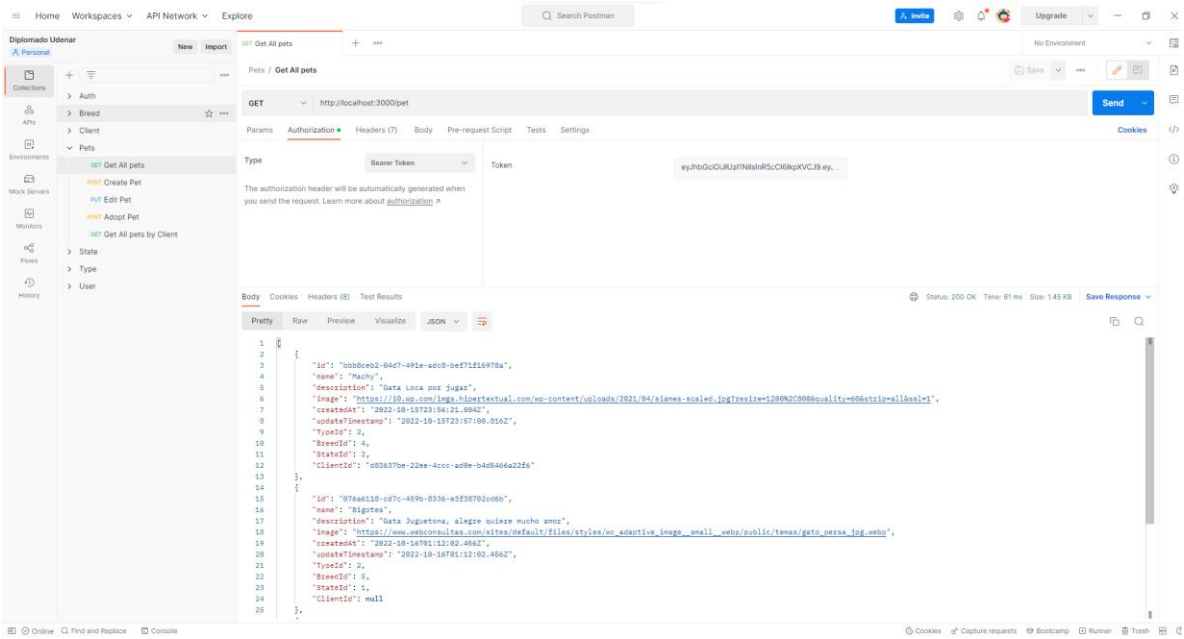
  - Create Pet



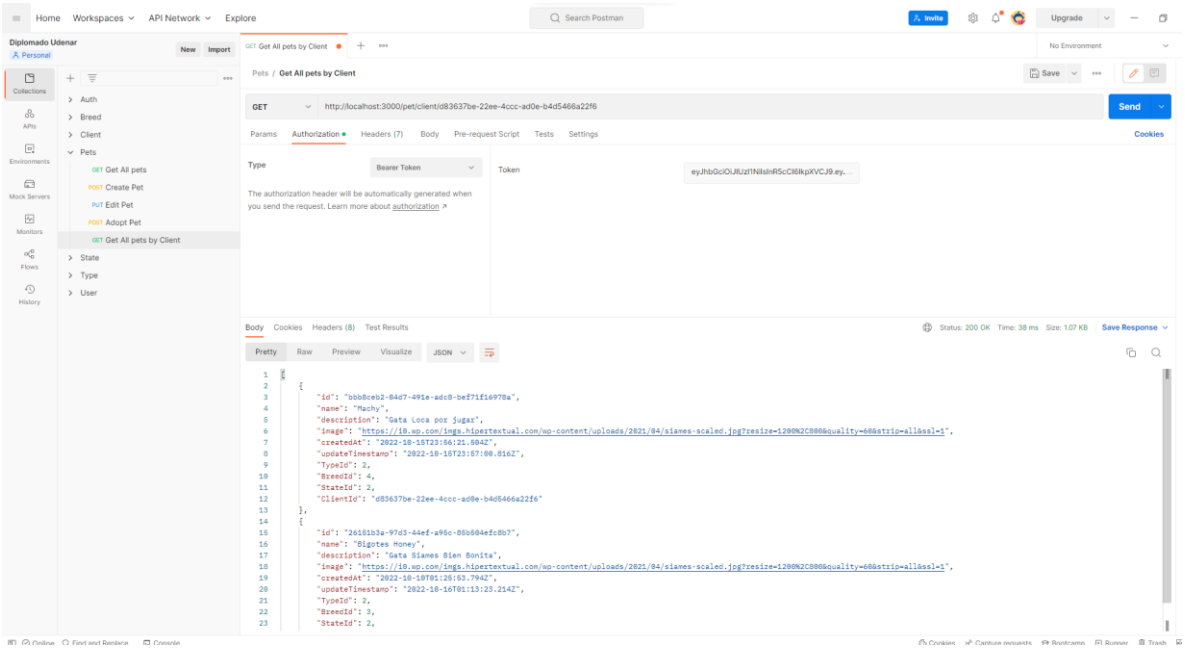
  - Edit Pet



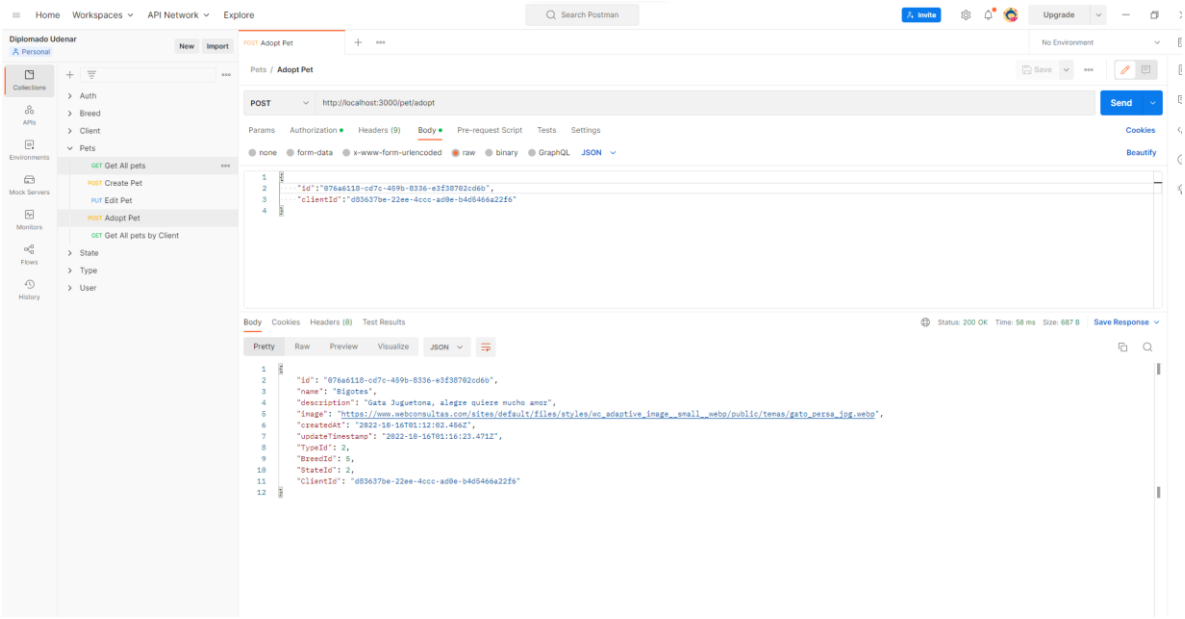
- Get All Pets



- Get All Pets By Client

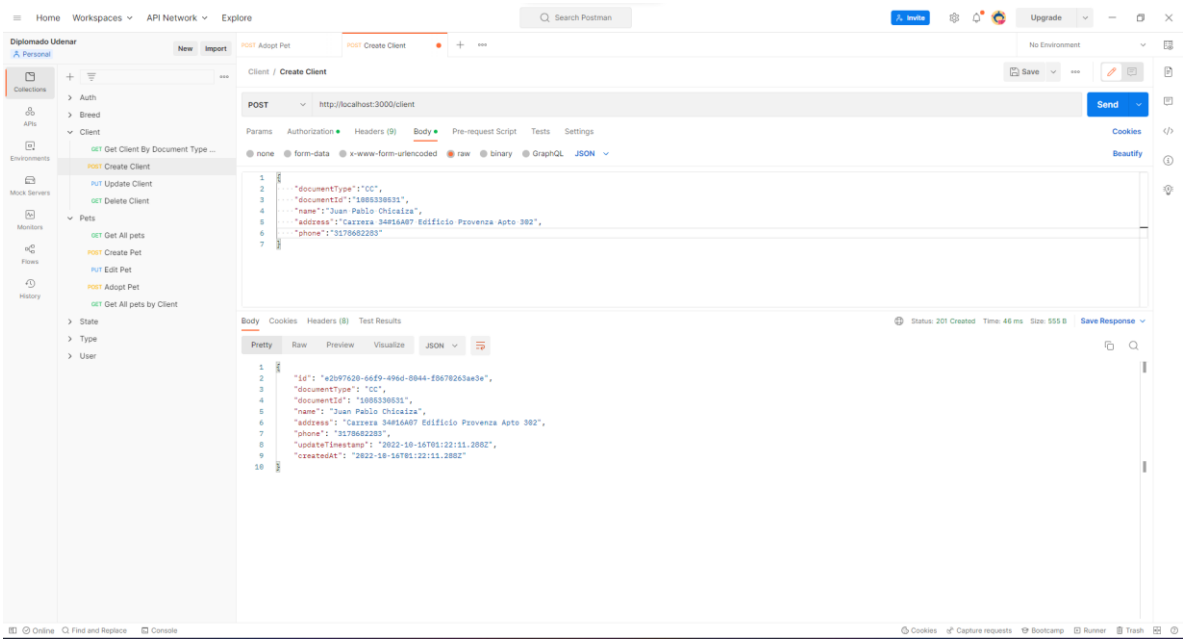


- Adopt Pet

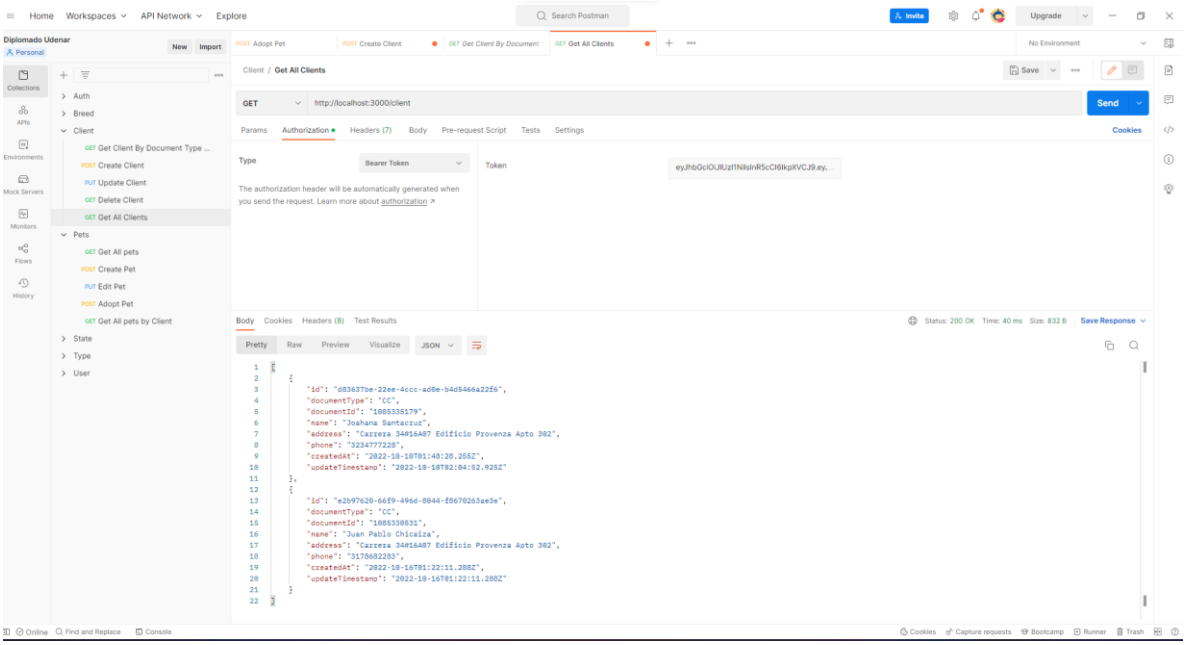


- Client (Clientes)

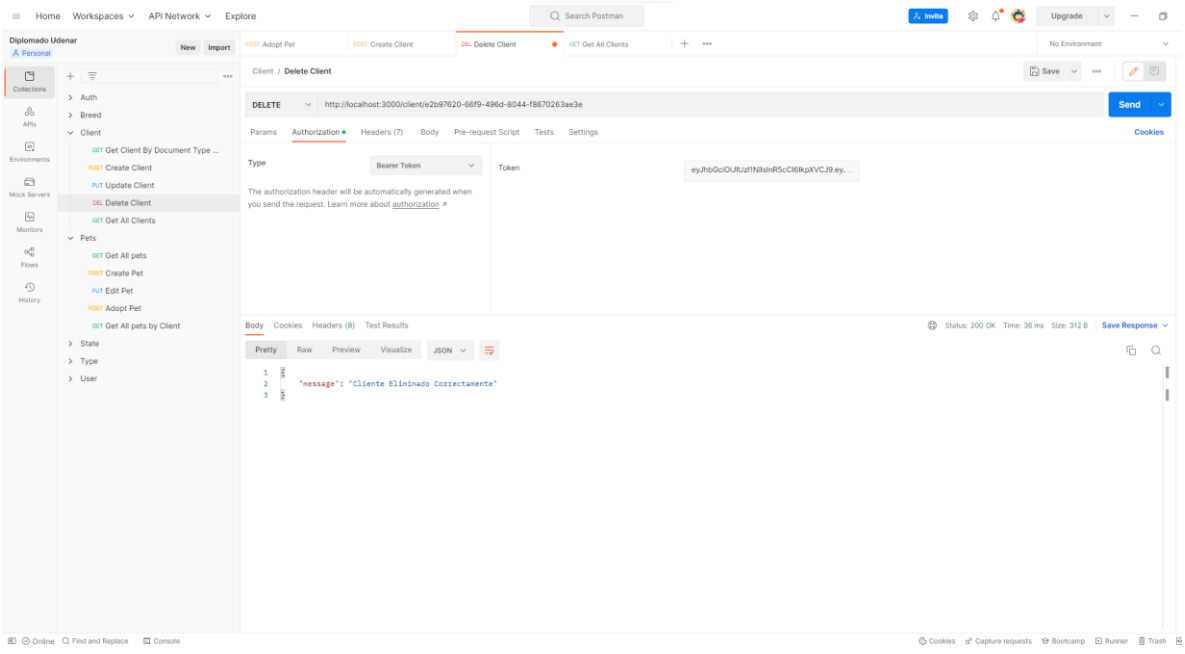
  - Create Client



- Get All Client



- Delete Client


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
Diplomado Udenar
Personal
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
Auth
Breed
Client
GET Get Client By Document Type ...
POST Create Client
PUT Update Client
DEL Delete Client
GET Get All Clients
Pets
GET Get All pets
POST Create Pet
PUT Edit Pet
POST Adopt Pet
GET Get All pets by Client
State
Type
User
POST Adopt Pet
POST Create Client
DEL Delete Client
GET Get All Clients
No Environment
Client / Delete Client
Save
DELETE
http://localhost:3000/client/e2b97620-66f9-496d-8044-f8670263ae3e
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Type
Bearer Token
Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...
The authorization header will be automatically generated when you send the request. Learn more about authorization ↗
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Time: 36 ms
Size: 312 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"message": "Cliente Eliminado Correctamente"
}
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash