Manual de Usuario: Tinder Canino

Versión del Sistema: 1.0
Última actualización: Octubre 2024

Índice

Introducción

Tinder Canino es una aplicación web diseñada para conectar mascotas con otras mascotas cercanas para que puedan socializar. Los dueños de las mascotas pueden registrar a sus mascotas, explorar posibles amistades cercanas, hacer match y comunicarse a través de la plataforma.

Requisitos Previos

Para ejecutar la aplicación, asegúrate de tener los siguientes requisitos:

- Python 3.8 o superior instalado en tu máquina: Lenguaje de programación en el cual está desarrollado el proyecto.
- Instalación: Descarga desde python.org y sigue las instrucciones de instalación para tu sistema operativo.
- MongoDB en ejecución en localhost (puerto 27017 por defecto): Base de datos NoSQL para almacenar la información de los usuarios, mascotas y coincidencias.
- Instalación: Descarga desde mongodb.com y sigue las instrucciones para configurar una instancia local en el puerto 27017.
- Una cuenta de Google para configurar la autenticación OAuth y acceso a la API de Google Maps.
- Instalación de Cloudinary para almacenamiento multimedia: Plataforma para la gestión de archivos multimedia (imágenes y videos) en la nube.
- Configuración: Crea una cuenta en Cloudinary y obtén tu URL de API para integrarla en el archivo de configuración.

- Entorno virtual configurado para gestionar dependencias de Python: Aisla las dependencias de Python del proyecto.
- Instalación: Una vez instalado Python, ejecuta python -m venv env en el directorio del proyecto.

Instalación y configuración

Clona el repositorio de Tinder Canino en tu máquina local:

git clone “URL del repositorio”

Instala las dependencias necesarias:

cd TinderCanino

python -m venv env

source env/bin/activate # En Windows: env\Scripts\activate

pip install -r requirements.txt

Configura tus credenciales en los archivos necesarios:

Autenticación de Google OAuth: Debes tener el archivo autorizacion.txt con las credenciales de cliente.

Configurar Cloudinary: Establece la URL en el archivo de configuración para subir imágenes y videos.

Iniciar MongoDB si no está ejecutándose:

Mongod

Ejecución de la aplicación

1. Para ejecutar la aplicación Flask, usa:

   python app.py

2. Esto levantará el servidor en http://127.0.0.1:5000. Puedes acceder a la aplicación desde tu navegador usando esa dirección.

Uso de la aplicación

Inicio de sesión y registro

Inicio de sesión: Puedes iniciar sesión con tu correo electrónico y contraseña o usando tu cuenta de Google.

Registro: Completa el formulario de registro proporcionando tu nombre, correo, contraseña y fecha de registro. Al finalizar, recibirás un correo de confirmación.

Registro de mascotas

Llena el formulario para incluir detalles de tu mascota como nombre, edad, raza, descripción, género e imágenes/videos.

Visualización de mascotas cercanas en el mapa

Haz clic en "Ver Mapa de Mascotas Cercanas" para ver ubicaciones usando la API de Google Maps.

Sistema de coincidencias (match)

Selecciona “Me gusta” o “Descartar” para dar un positivo o negativo en mascotas. Si hay un match, ambos dueños serán notificados.

Mensajería entre usuarios

Si se ha hecho match con una mascota, podrás iniciar una conversación en la sección de chat.

Mantenimiento y gestión

Gestión de usuarios

Los usuarios están en la colección usuarios en MongoDB, accesible desde la interfaz de administración o usando MongoDB shell.

Gestión de mascotas

Las mascotas se gestionan en la colección mascotas, y pueden actualizarse desde un formulario de edición.

Backups y restauración

Es importante hacer respaldos de la base de datos de MongoDB regularmente usando:

mongodump --db tinder_canino --out /ruta/de/respaldo

Solución de problemas comunes

Error de conexión a MongoDB: Asegúrate de que el servicio mongodb está en ejecución.

Error de autenticación OAuth con Google: Revisa autorizacion.txt y las APIs de Google.

Problemas con la carga de imágenes: Asegúrate de tener las credenciales correctas para Cloudinary y revisa la configuración en el archivo app.py.

Librerías utilizadas

Flask: Para el desarrollo de la aplicación web.

pymongo: Conexión a la base de datos MongoDB.

flask-jwt-extended: Gestión de autenticación con JWT.

werkzeug.security: Para el hash y verificación de contraseñas.

google-auth: Para la autenticación de usuarios con cuentas de Google.

geopy: Cálculo de distancias entre ubicaciones.

cloudinary: Subida y gestión de archivos multimedia.

flask-mail: Para enviar correos electrónicos.

Estructura de archivos del proyecto

```
├── env/                # Entorno virtual
├── templates/
│   └── index.html      # Archivo HTML principal
├── app.py              # Archivo principal de la aplicación Flask
├── autorizacion.txt    # Credenciales de Google OAuth
├── db.py               # Configuración y conexión a MongoDB
├── requirements.txt    # Dependencias de Python
└── tindercanino.py     # Scripts adicionales y funciones del proyecto
```

Consideraciones adicionales

Mantén tus claves y credenciales seguras y no las compartas en repositorios públicos.

Revisa la documentación de las librerías utilizadas para entender configuraciones avanzadas y posibles errores.

Explicación de partes esenciales del código

Configuración de la Aplicación Flask

-from flask import Flask

-from flask_jwt_extended import JWTManager

-import pymongo

Se importan las librerías esenciales: Flask para la estructura web, JWTManager para manejar la autenticación basada en tokens y pymongo para la conexión a la base de datos MongoDB.

Inicialización y Configuración

-app = Flask(__name__)

-app.config['JWT_SECRET_KEY'] = 'your_jwt_secret_key'

Se inicializa la instancia de Flask y se establece una clave secreta para la autenticación JWT.

Autenticación de Usuarios con Google OAuth

-from google_auth_oauthlib.flow import Flow

-flow = Flow.from_client_secrets_file('autorizacion.txt', scopes=['openid', 'https://www.googleapis.com/auth/userinfo.email'])

Esta sección configura la autenticación de usuarios con OAuth utilizando el archivo autorizacion.txt, que contiene las credenciales generadas en Google Cloud.

Rutas Principales del Proyecto

Registro de Usuario

@app.route('/register', methods=['POST'])

def register():

La ruta /register permite a los usuarios crear una cuenta proporcionando su información.

Inicio de sesión

@app.route('/login', methods=['POST'])

def login():

La ruta /login autentica a los usuarios y, si es correcto, devuelve un token JWT para futuras solicitudes.

Registro de mascotas

@app.route('/register_pet', methods=['POST'])

def register_pet():

Al visitar /nearby_pets, se carga un mapa con ubicaciones de mascotas cercanas utilizando la API de Google Maps.

Coincidencias y mensajes

@app.route('/match', methods=['POST'])

def match():

Integración con Cloudinary para Gestión de Imágenes

import cloudinary

cloudinary.config(

    cloud_name = 'your_cloud_name',

    api_key = 'your_api_key',

    api_secret = 'your_api_secret'

)

Recomendaciones

- Asegúrate de que los archivos estáticos se carguen correctamente verificando la configuración de `STATIC_URL` y `STATICFILES_DIRS`.
- Usa el comando `python manage.py runserver` para iniciar el servidor en modo de desarrollo.