

Segundo Parcial - Proyectos Aplicados a la I.E , Sección N

Victor Daniel Tepáz Morente, 202202119, Alejandra Magali Lopéz Miranda 201600085

Resumen—Se desarrolló una plataforma web para una clínica veterinaria utilizando el framework Django con Python y una base de datos PostgreSQL. El sistema permite la gestión integral de servicios veterinarios como consultas, vacunaciones, castraciones, lavado y agendamiento de citas en línea. La plataforma está diseñada para ofrecer una experiencia segura y eficiente tanto para los clientes como para el personal administrativo, incorporando módulos de autenticación, validación de datos y una interfaz intuitiva para la solicitud y confirmación de servicios.

I. INTRODUCCIÓN

Este proyecto desarrolla una plataforma web integral para una clínica veterinaria utilizando Django, Python y PostgreSQL, con el objetivo de digitalizar y optimizar la gestión de servicios médicos veterinarios. La solución implementada permite el registro seguro de usuarios, agendamiento de citas, gestión de mascotas y administración de servicios como consultas, vacunaciones y esterilizaciones, incorporando validaciones de datos, confirmaciones vía correo electrónico y una interfaz intuitiva que garantiza una experiencia de usuario fluida y eficiente, representando una modernización significativa en la interacción entre la clínica y sus clientes.

II. OBJETIVOS

II-A. General

Desarrollar una plataforma web robusta y funcional para una clínica veterinaria que optimice la gestión de servicios y mejore la interacción entre clientes y personal, mediante el uso de tecnologías modernas como Django, Python y PostgreSQL.

II-B. Específicos

- * Implementar un sistema seguro de registro y autenticación de usuarios.
- * Integrar una base de datos relacional que garantice la consistencia y disponibilidad de la información de usuarios, mascotas, servicios y citas.
- * Facilitar la administración centralizada de citas y servicios a través de una interfaz accesible y responsive.

III. ALGORITMO

Algoritmo del programa

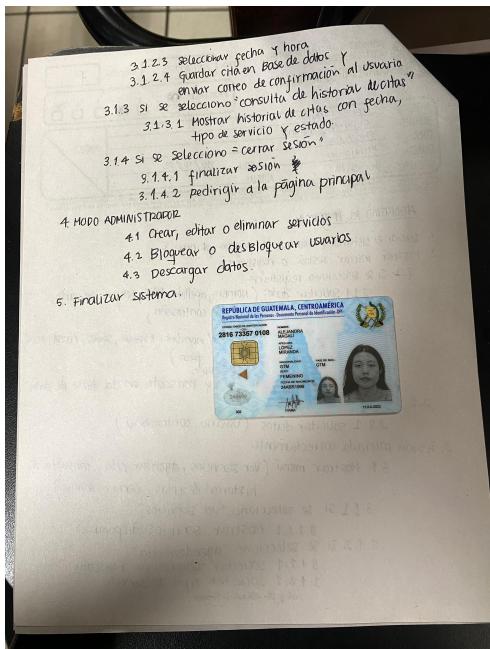
- 1) Iniciar y mostrar la página principal
Muestra la información de la clínica, los servicios ofrecidos y el contacto
- 2) Verifica si el usuario posee cuenta.
Si no tiene, procede a enviarlo a crear una
Solicita datos
Solicita datos de la mascota
Solicita el historial médico
Registro al usuario y a la mascota
Si tiene cuenta reenvia al inicio de sesión.
- 3.) Sesión iniciada correctamente
Mostrar menú
 - Servicios
 - Agendar citas
 - Consultar el historial
 - Cerrar sesión
- 4.) Finaliza el sistema

Fuente: Elaboración propia, 2025

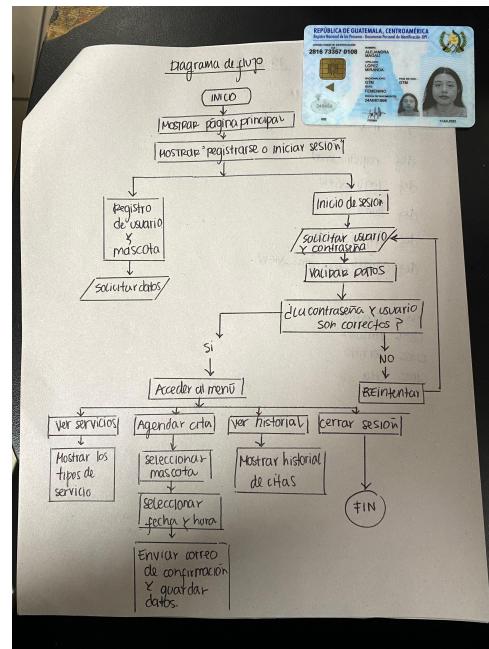
Algoritmo del programa

1. Iniciar el sistema y mostrar la página principal
2. Mostrar iniciar sesión o registrar
 - 2.1 Si se selecciona registrar
 - 2.1.1 Solicitar datos (Nombre, apellido, DPI, teléfono, usuario, contraseña)
 - 2.1.2 Confirmar contraseña
 - 2.1.3 Solicitar datos mascota (nombre, especie, sexo, raza, edad, peso)
 - 2.1.4 Solicitar historial médico
 - 2.1.5 Registrar al usuario y mascota en la base de datos
 - 2.2 Si se selecciona iniciar sesión
 - 2.2.1 solicitar datos (Usuario, contraseña)
3. Sesión iniciada correctamente
 - 3.1 Mostrar menú (Ver servicios, agendar cita, consulta de historial de citas, cerrar sesión)
 - 3.1.1 Si se selecciona "ver servicios"
 - 3.1.1.1 Mostrar servicios disponibles
 - 3.1.2 Si se selecciona "agendar citas"
 - 3.1.2.1 Solicitar seleccionar mascota
 - 3.1.2.2 Solicitar tipo de servicio
 - 3.1.2.3 Enviar a Todos

Fuente: Elaboración propia, 2025

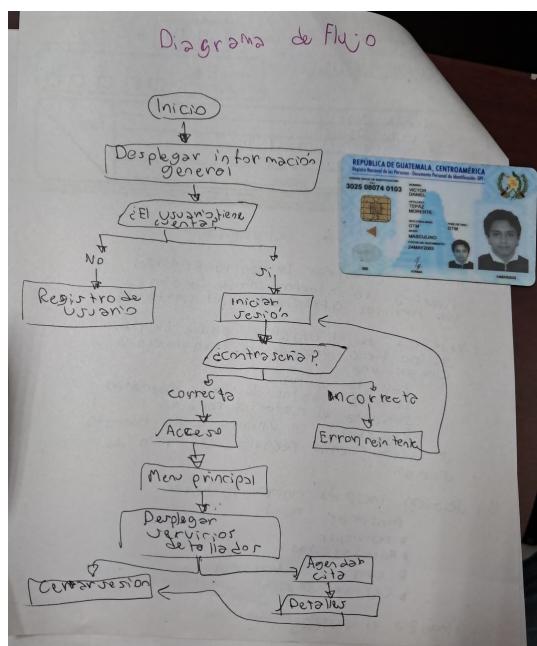


Fuente: Elaboración propia, 2025



Fuente: Elaboración propia, 2025

IV. DIAGRAMA DE FLUJO



Fuente: Elaboración propia, 2025

V. RESULTADOS

V-A. Activación del entorno

```
C:\Users\Alejandra\Desktop\980-Proyectos\Proyecto\Django\proyecto\Scripts>activate
(proyecto) C:\Users\Alejandra\Desktop\980-Proyectos\Proyecto\Django\proyecto\Scripts>
```

Fuente: Elaboración propia, 2025

V-B. Inicialización del servidor

```
(proyecto) C:\Users\Alejandra\Desktop\980-Proyectos\Proyecto\Django\veterinaria>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until
you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 06, 2025 - 22:17:01
Django version 5.2.7, using settings 'veterinaria.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fuente: Elaboración propia, 2025

V-C. Estructura de los views utilizado

```

1  from django.contrib import admin
2  from django.urls import path, include
3  from django.conf import settings
4  from django.conf.urls.static import static
5  from usuarios.views import editar_usuario
6  from django.contrib.auth import views as auth_views
7  from usuarios.forms import CustomPasswordResetForm
8  from usuarios.forms import CustomSetPasswordForm
9  from django.contrib import admin
10 from django.urls import path, include
11 from usuarios.views import CustomPasswordResetConfirmView
12 from usuarios.views import CustomPasswordResetView, CustomPasswordResetConfirmView
13
14 from citas.views import (
15     agendar_cita,
16     historial_citas,
17     editar_cita,
18     eliminar_cita,
19     marcar_completada, # <- aqui
20 )
21
22 from usuarios.views import (
23     inicio, custom_login, registro, redirect_by_role,
24     panel_administracion, perfil_usuario, cerrar_sesion,
25     editar_perfil, eliminar_usuario, detalle_cita,
26     gestion_citas, cambiar_estado_cita
27 )
28 from mascotas.views import (
29     agregar_mascota, editar_mascota, eliminar_mascota
30 )
31 from citas.views import (
32     agendar_cita, historial_citas, editar_cita, eliminar_cita
33 )
34 from servicios.views import (
35     gestion_servicios, crear_servicio, editar_servicio, eliminar_servicio

```

Fuente: Elaboración propia, 2025

V-D. Estructura de los modelos utilizado

```

1  from django.contrib.auth.models import AbstractUser
2  from django.db import models
3  from django.conf import settings
4  from django.utils import timezone
5
6  class Cliente(AbstractUser):
7      dpi = models.CharField(max_length=13, unique=True)
8      telefono = models.CharField(max_length=15)
9      direccion = models.TextField()
10
11     def __str__(self):
12         return self.username
13
14     class Mascota(models.Model):
15
16     class Servicio(models.Model):
17
18     class Cita(models.Model):

```

Fuente: Elaboración propia, 2025

El desarrollo del sistema web para la clínica veterinaria demostró la eficacia del entorno Django para la construcción de aplicaciones web robustas, modulares y seguras. Django, al ser un framework de alto nivel basado en Python, permitió una rápida estructuración del proyecto gracias a su arquitectura Modelo-Vista-Template (MVT), facilitando así la separación entre la lógica de negocio, la presentación y el acceso a los datos.

Durante el desarrollo, se organizaron diversas aplicaciones dentro del proyecto principal, siguiendo las buenas prácticas que propone Django. Entre estas aplicaciones, se incluyó una dedicada a la gestión de usuarios y autenticación, otra para la administración de servicios veterinarios (como vacunación, consultas, castraciones, etc.), y una más enfocada en la gestión de mascotas y la programación de citas.

Las vistas desarrolladas en Python permitieron manejar la lógica de las peticiones HTTP, procesar formularios y coordinar la interacción entre el usuario y la base de datos. Estas vistas se integraron con formularios de Django para validar datos como el nombre del cliente, la fecha de la

cita, el tipo de servicio solicitado y la información de las mascotas. Se utilizaron tanto **vistas basadas en funciones (FBV)** como **vistas basadas en clases (CBV)**, dependiendo de la complejidad de cada caso.

En cuanto a la base de datos, se utilizó **PostgreSQL** como motor relacional por su fiabilidad y compatibilidad con Django. Se estableció una conexión fluida entre la aplicación y la base de datos mediante el **ORM (Object-Relational Mapping)** de Django, lo que facilitó la manipulación de datos sin necesidad de escribir consultas SQL directamente. Se crearon modelos personalizados para almacenar información de los clientes (incluyendo usuario y contraseña encriptada), las mascotas registradas, los tipos de servicios solicitados y las fechas programadas para las citas.

La plataforma resultante permite a los usuarios:

- Registrarse e iniciar sesión de forma segura.
- Registrar información básica de sus mascotas.
- Solicitar servicios según disponibilidad y necesidades.
- Almacenar y consultar su historial de citas.
- Recibir confirmación de citas vía correo electrónico.

VI. CONCLUSIONES

- * El desarrollo con Django demostró ser eficiente para crear una plataforma web robusta y segura, facilitando la estructura modular y el manejo de datos mediante ORM con PostgreSQL.
- * Se implementó exitosamente un sistema integral que permite a los usuarios registrar mascotas, solicitar servicios veterinarios y gestionar citas, con validaciones de seguridad y confirmación vía correo electrónico.
- * La arquitectura MVT y la modularidad del proyecto sentaron las bases para futuras expansiones, demostrando que la solución es escalable y adaptable a nuevas funcionalidades.

VII. REPOSITORIO EN GITHUB

El código fuente, ejemplos y materiales utilizados para esta comparación se encuentran disponibles en el siguiente repositorio público de GitHub:

https://github.com/Victepa2417/Tareas-0980

Este repositorio contiene los códigos implementados en MATLAB y Python.

VIII. ANEXOS

REFERENCIAS

- [1] MathWorks. (2023). *MATLAB Documentation*. Disponible en: <https://www.mathworks.com/help/matlab/>
- [2] Eaton, J. W., Bateman, D., Hauberg, S., Wehbring, R. (2022). *GNU Octave Manual Version 7*. GNU Project. Disponible en: <https://www.gnu.org/software/octave/doc/interpreter/>
- [3] Van Rossum, G., Drake, F. L. (2023). *The Python Language Reference*. Python Software Foundation. Disponible en: <https://docs.python.org/3/>

- [4] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). *Array programming with NumPy*. *Nature*, 585(7825), 357–362. Disponible en: <https://numpy.org/>
- [5] Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. *Computing in Science & Engineering*, 9(3), 90–95. Disponible en: <https://matplotlib.org/>