

Manual de usuario – Greenway

Elaborado por:

Sergio Gabriel Chaves Mosquera
Samuel Esteban Reyes Uribe
María José Sánchez Portillo

Presentado a

Néstor German Bolívar Pulgarin

Asignatura:

Programación Orientada a Objetos

Universiada Nacional de Colombia
2025 – 2s
Bogotá, DC
2025

Ficha Técnica

A continuación, se presentan los elementos usados y necesarios para el proyecto:

A. Backend

- Lenguaje de Programación: Python 3.13.9
- Framework Web: Flask (Microframework para el manejo de rutas y servidor).
- Servidor WSGI: Gunicorn (Para producción en Render).
- Manejo de Paquetes: pip y setuptools.

B. Base de Datos y Nube

- Plataforma de Backend: Google Firebase.



- Base de Datos: Firebase Realtime Database (NoSQL, estructura de árbol JSON).

The screenshot shows the Firebase Realtime Database interface. On the left, there's a tree view of the database structure under the URL `https://greenway-450aa-default.firebaseio.com/`. The structure includes nodes for `admins`, `chats`, `experiencias`, `propietarios`, and `usuarios`. On the right, a specific node under `propietarios` is expanded, showing a single child node with the key `t7xLmDC2SSdedKLzOOr5Vd60SS3`. This child node contains several fields: `email` (set to "Carmen.gmz123@gmail.com"), `foto_url` (set to "`https://i.postimg.cc/wT441BTX/618e08710a6617e2d6c5a4f7a97596f.jpg`"), `nombre` (set to "Carmensita gomez"), `rol` (set to "proprietaria"), and `telefono` (set to "3002645113").

The screenshot shows the Firebase Realtime Database interface. On the left, there's a tree view of the database structure under the URL `https://greenway-450aa-default.firebaseio.com/`. The structure includes nodes for `admins`, `chats`, `experiencias`, `propietarios`, and `usuarios`. Under the `experiencias` node, a specific child node with the key `-OdyDV5af27w4PKzk1vz` is expanded. This child node contains fields: `descripcion` (set to "Las abejas son organismos fundamentales para la vida en el planeta. ¿Sabías que en el mundo existen más de 20.000 especies de abejas?"), `imagenes_url` (set to "`https://i.postimg.cc/XvXtdRWX/Imagen-de-Whats-App-2025-11-03-a-las-09-08-37-282a80d8.jpg`"), and `imagenes` (which is itself a child node containing four items labeled 0, 1, 2, and 3, each pointing to a URL like "`https://i.postimg.cc/pLvPQtgd/Apiario.jpg`"). Other fields in this node include `maps_embed_url` (set to "`<iframe src='https://www.google.com/maps/embed?pb=!m18!1m12!1m3!1d53500.65525916311!2d-74.49037703287533!3d4.68525916311!4d-74.49037703287533!3m2!1e1!2e1'`") and `nombre` (set to "AYNI").

- Autenticación: Firebase Authentication (Manejo de usuarios, correos y contraseñas).



- Librería de Conexión: Pyrebase.

```

data > firebase_config.py
1 import pyrebase
2
3 firebaseConfig = {
4   "apiKey": "AIzaSyCvNgRDVftYYY004NntYY6yjo9huOrH_CU",
5   "authDomain": "greenway-450aa.firebaseio.com",
6   "databaseURL": "https://greenway-450aa-default-rtdb.firebaseio.com",
7   "projectId": "greenway-450aa",
8   "storageBucket": "greenway-450aa.firebaseio.storage.app",
9   "messagingSenderId": "107049448583",
10  "appId": "1:107049448583:web:2f8a6eece8d2e21d913422"
11 }
12
13 firebase = pyrebase.initialize_app(firebaseConfig)
14 auth = firebase.auth()
15 db = firebase.database()
  
```

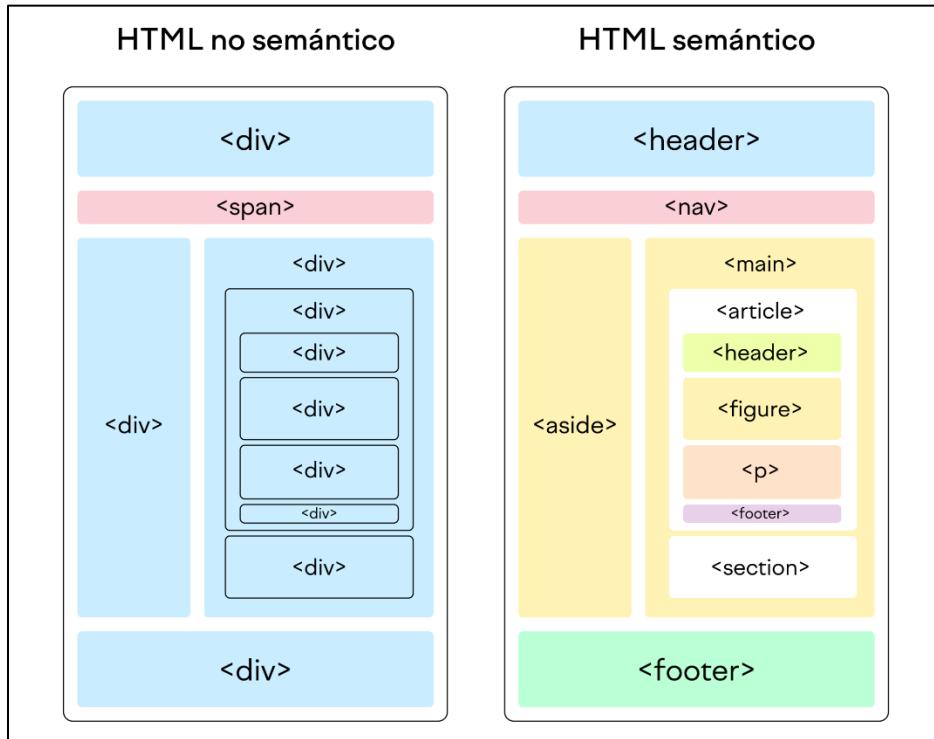
- Gestión de Admin: firebase-admin (SDK oficial para tareas como borrar usuarios o gestionar roles).

```

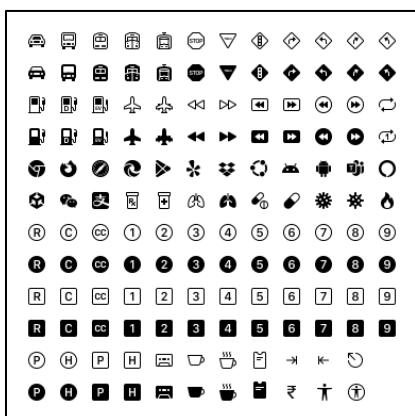
data > firebase_admin.py
1 import firebase_admin
2 from firebase_admin import credentials, auth
3 import os
4 DATABASE_URL = 'https://greenway-450aa-default-rtdb.firebaseio.com'
5 BASE_DIR = os.path.dirname(os.path.abspath(__file__))
6 CRED_PATH = os.path.join(BASE_DIR, '..', 'serviceAccountKey.json')
7
8
9 try:
10   cred = credentials.Certificate(CRED_PATH)
11   firebase_admin.initialize_app(cred, {
12     'databaseURL': DATABASE_URL
13   })
14   admin_auth = auth
15   print("SDK de Admin de Firebase inicializado con éxito.")
16
17 except FileNotFoundError:
18   print("*"*50)
19   print(f"ERROR: No se encontró el archivo '{CRED_PATH}'")
20   print("Por favor, descarga tu 'serviceAccountKey.json' desde la Consola de Firebase")
21   print("y colócalo en tu carpeta 'data/'.")
22   print("*"*50)
23   admin_auth = None
24
25 except Exception as e:
26   print(f"Error inicializando el SDK de Admin: {e}")
27   admin_auth = None
  
```

C. Frontend

- Lenguaje de Marcado: HTML5 Semántico.



- Motor de Plantillas: Jinja2
- Estilos (CSS): CSS3 Nativo, Bootstrap 5.3, Bootstrap Icons.



- Tipografía: Google Fonts ("Montserrat" para todo el sitio).



- Scripting: JavaScript (Vanilla JS ES6+) para la lógica del lado del cliente (chat en tiempo real, buscador, mapas).

D. APIs y Servicios Externos

- Chatbot IA: OpenAI API (Modelo GPT-3.5-turbo/GPT-4) integrado mediante la librería openai.



- Mapas: Google Maps Embed API.



- Alojamiento de Imágenes: Postimages.org (Hosting externo para URLs directas).



- Avatares: UI Avatars API (Generación automática de fotos de perfil con iniciales).



E. Infraestructura y Despliegue

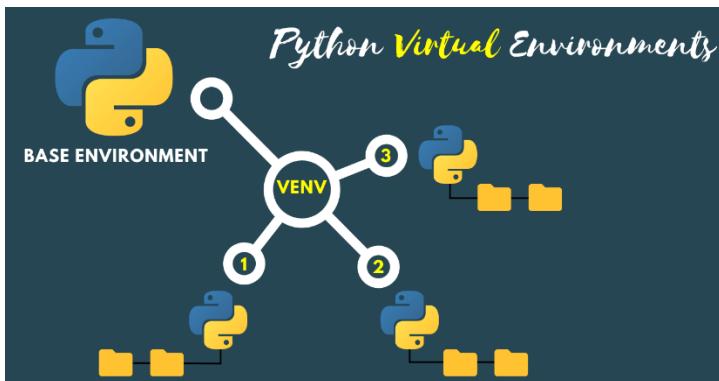
- Control de Versiones: Git y GitHub.



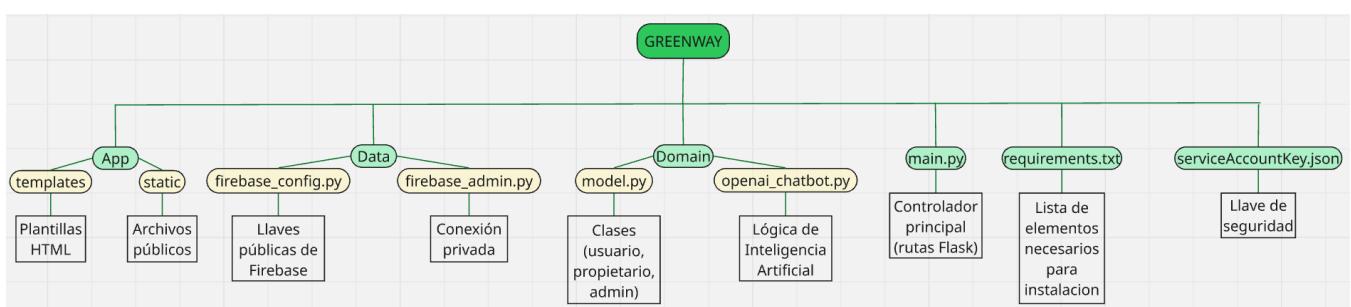
- Hosting (Despliegue): Render (Plataforma PaaS).



- Entorno Virtual: venv

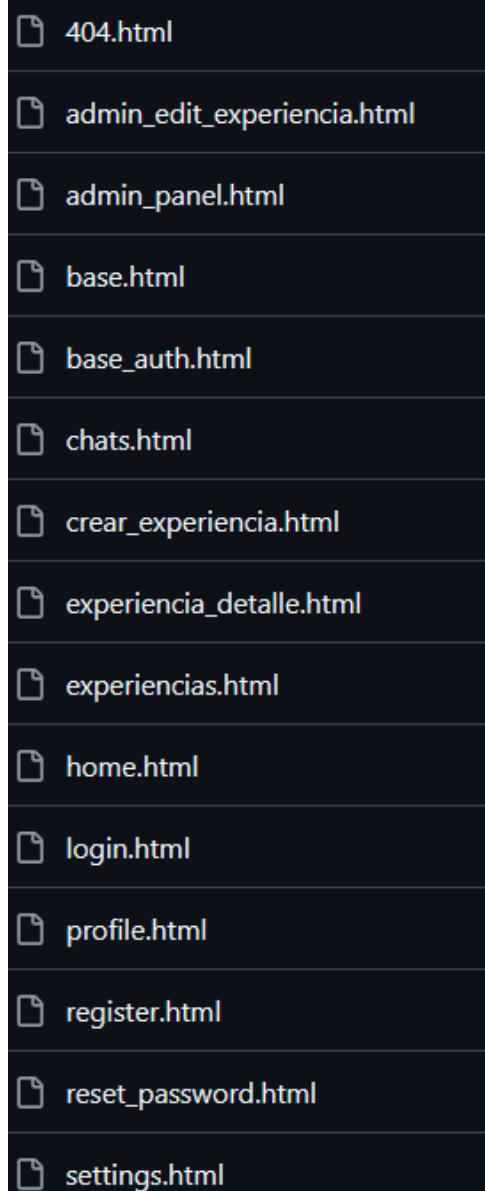


Estructura del Proyecto



- app:

◆ Templates



```

1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>[% block title %]Bienvenido[% endblock %] - GREENWAY</title>
7
8      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
9      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.1.3/font/bootstrap-icons.min.css">
10     <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400,600,700,900&display=swap" rel="stylesheet">
11
12     <style>
13         /* ---- ESTILOS COMUNES PARA TODO EL SITIO ---- */
14
15     :root {
16         --greenway-primary: #4682B4; /* Verde principal */
17         --greenway-accent: #0066FF; /* Naranja (para el chat) */
18         --greenway-yellow-light: #FFEE33; /* Amarillo alerta */
19         --greenway-blue: #6680BD; /* Azul Info */
20         --greenway-text: #222;
21     }
22
23     body {
24         font-family: 'Montserrat', sans-serif;
25         background-color: #fafafa;
26         /* El padding-top se hace con 35 para que sea dinámico */
27     }
28

```

```

29     /* NAVBAR (Común) */
30     .navbar-greenway {
31         background-color: var(--greenway-primary) !important;
32         box-shadow: 0 2px 6px rgba(0,0,0,.3);
33         padding: 10px 40px;
34         position: fixed;
35         top: 0;
36         width: 100%;
37         z-index: 1030;
38         transition: transform 0.3s ease-in-out;
39     }
40     .navbar-greenway .navbar-brand, .navbar-greenway .nav-link { color: white !important; font-weight: 600; transition: color 0.3s; }
41     .navbar-greenway .nav-link:hover { color: #222 !important; }
42     .navbar-greenway .btn-light { background-color: white !important; color: var(--greenway-primary) !important; font-weight: 700; border: none; transition: all 0.3s; }
43     .navbar-greenway .btn-light:hover { background-color: #f8f8f8 !important; transform: scale(1.05); }
44     .navbar-hidden { transform: translateY(-100%); }
45
46     /* FOOTER (Común) */
47     .footer-greenway {
48         background-color: #222;
49         color: #ccc;
50     }
51     .footer-greenway h5 { font-weight: 700; color: var(--greenway-primary); font-size: 1.5rem; }
52     .footer-greenway p { font-size: 0.9rem; margin-bottom: 5px; }
53     .footer-greenway strong { color: #fff; }
54     .footer-greenway .copyright { background-color: rgba(0, 0, 0, 0.2); font-size: 0.8rem; }
55

```

◆ Static

Apario.jpg
Apicultores.jpg
Bicho1.jpg
Bicho2.jpg
Bicho3.jpg
Bicho4(con miel).jpg
Gente Random 1.jpg
Gusano1.jpg
JardinTomas.jpg
LaMesaCundinamarca.jpg
Loro.jpg
Mariposa.jpg
Mariposa2.jpg
Mariposa3.jpg
MariposaCool.png
Microscopio niña1.jpg

- data:

◆ firebase config.py

```
1      import pyrebase
2
3  firebaseConfig = {
4      "apiKey": "AIzaSyCvNgRDVftYYYY004NntYy6yjo9hUOrH_CU",
5      "authDomain": "greenway-450aa.firebaseio.com",
6      "databaseURL": "https://greenway-450aa-default-rtdb.firebaseio.com",
7      "projectId": "greenway-450aa",
8      "storageBucket": "greenway-450aa.firebaseio.storage.app",
9      "messagingSenderId": "107049448583",
10     "appId": "1:107049448583:web:2f8a6eece8d2e21d913422"
11 }
12
13 # 1. Llamamos a la app "firebase"
14 firebase = pyrebase.initialize_app(firebaseConfig)
15
16 # 2. Usamos "firebase" para crear los servicios
17 auth = firebase.auth()
18 db = firebase.database()
```

◆ firebase_admin.py

```
 1  import firebase_admin
 2  from firebase_admin import credentials, auth
 3  import os
 4
 5  # --- Configuración ---
 6
 7  # 1. Esta es la URL de tu base de datos (de tu archivo firebase_config.py)
 8  DATABASE_URL = 'https://greenway-450aa-default.firebaseio.com'
 9
10 # 2. Construye la ruta al archivo de clave que debe estar en esta misma carpeta ('data/')
11 #   os.path.abspath(__file__) -> Obtiene la ruta de este archivo (firebase_admin.py)
12 #   os.path.dirname(...)          -> Obtiene la carpeta que lo contiene ('data/')
13 #   os.path.join(..., ...)        -> Une 'data/' + 'serviceAccountKey.json'
14 BASE_DIR = os.path.dirname(os.path.abspath(__file__))
15 CRED_PATH = os.path.join(BASE_DIR, '..', 'serviceAccountKey.json')
16
17 # --- Fin Configuración ---
18
19 try:
20     # Intenta cargar la "llave maestra" desde la ruta
21     cred = credentials.Certificate(CRED_PATH)
22
23     # Inicializa la aplicación de Admin
24     firebase_admin.initialize_app(cred, {
25         'databaseURL': DATABASE_URL
26     })
27
28     # Exportamos solo el módulo 'auth' del SDK de Admin
29     # Lo llamamos 'admin_auth' para no confundirlo con el 'auth' de pyrebase
30     admin_auth = auth
31     print("SDK de Admin de Firebase inicializado con éxito.")
32
33 except FileNotFoundError:
34     print("=**")
35     print("ERROR: No se encontró el archivo '{CRED_PATH}'")
36     print("Por favor, descarga tu 'serviceAccountKey.json' desde la Consola de Firebase")
37     print("y colócalo en tu carpeta 'data/'")
38     print("=**")
39     admin_auth = None # Asigna None para que la app falle y sepas qué pasó
40
41 except Exception as e:
42     print(f"Error inicializando el SDK de Admin: {e}")
43     admin_auth = None
```

● domain:

◆ Models.py

```
 1  v  class Persona:
 2  v      def __init__(self, user_id: str, nombre: str, email: str, telefono: str = ""):
 3          # Validación básica (El teléfono puede ser opcional al inicio, por eso el default "")
 4          if not all([user_id, nombre, email]):
 5              raise ValueError("ID, nombre y email no pueden estar vacíos")
 6
 7          self.user_id = user_id
 8          self.nombre = nombre
 9          self.email = email
10          self.telefono = telefono
11          self.foto_url = "" # Campo nuevo para la foto de perfil
12
13          # Valores por defecto (se sobreescriben en las clases hijas)
14          self.rol = "persona"
15          self.db_node = "personas"
16
17  v      def to_dict(self) -> dict:
18          return {
19              "nombre": self.nombre,
20              "email": self.email,
21              "telefono": self.telefono,
22              "rol": self.rol,
23              "foto_url": self.foto_url # Guardamos la foto también
24          }
25
26  v      def save_to_db(self, db):
27          """
28              Guarda la instancia en el nodo de la base de datos que le corresponde.
29          """
30          try:
31              # print(f"Guardando en: {self.db_node}/{self.user_id}") # Debug opcional
32              db.child(self.db_node).child(self.user_id).set(self.to_dict())
33          return True
34      except Exception as e:
```

```
35             print(f"Error al guardar en la DB: {e}")
36             return False # Retornamos False si falla
37
38     @staticmethod
39     def get_db_node_by_role(rol: str) -> str:
40         """Método estático para saber en qué nodo buscar según el rol."""
41         if rol == 'admin':
42             return 'admins'
43         elif rol == 'propietaria':
44             return 'propietarios'
45         else:
46             return 'usuarios'
47
48     @staticmethod
49     def get_user_data_by_role(db, rol: str, user_id: str) -> dict:
50         """
51             Método estático para obtener los datos de un usuario
52             buscando en el nodo correcto.
53         """
54         nodo = Persona.get_db_node_by_role(rol)
55         try:
56             return db.child(nodo).child(user_id).get().val()
57         except Exception as e:
58             print(f"Error al obtener datos del usuario {user_id} en {nodo}: {e}")
59             return None
60
61
62     # --- CLASES HIJAS ---
63
64     class Usuario(Persona):
65         def __init__(self, user_id: str, nombre: str, email: str, telefono: str = ""):
66             super().__init__(user_id, nombre, email, telefono)
67             self.rol = "usuario"
68             self.db_node = "usuarios"
69
70     # --- CLASES HIJAS ---
71
72     class Propietaria(Persona):
73         def __init__(self, user_id: str, nombre: str, email: str, telefono: str = ""):
74             super().__init__(user_id, nombre, email, telefono)
75             self.rol = "propietaria"
76             self.db_node = "propietarios"
77
78     class Admin(Persona):
79         def __init__(self, user_id: str, nombre: str, email: str, telefono: str = ""):
80             super().__init__(user_id, nombre, email, telefono)
81             self.rol = "admin"
82             self.db_node = "admins"
```

◆ openai_chatbot.py

```
1 import os
2 from openai import OpenAI
3 from dotenv import load_dotenv
4
5 class GreenwayChatbot:
6     """
7         Clase orientada a objetos para manejar la lógica del chatbot
8         con la API de OpenAI.
9         """
10
11     def __init__(self):
12         """
13             Constructor. Carga la API Key y define el "cerebro" (prompt) del bot.
14             """
15         # 1. Cargar la API Key desde el archivo .env que acabas de crear
16         load_dotenv()
17         api_key = os.getenv("OPENAI_API_KEY")
18
19         if not api_key:
20             # Si no encuentra la clave, lanza un error claro
21             raise ValueError("No se encontró la API Key de OpenAI."
22                             "Asegúrate de crear un archivo .env en la raíz del proyecto"
23                             "con la línea: OPENAI_API_KEY=sk-... ")
24
25         # 2. Inicializar el cliente de OpenAI
26         self.client = OpenAI(api_key=api_key)
27
28         # 3. ¡EL PROMPT CORRECTO! (El "cerebro" y "personalidad" del bot)
29         # Aquí definimos las reglas de Maio y el santuario.
30         self.system_prompt = """
31 Eres "Greenway-bot", un asistente experto en ecoturismo y el anfitrón virtual de Greenway que es un proyecto para un santuario de mariposas llamado "EcoParque Paraiso Mariposa".
32
33 SOBRE GREENWAY:
34 - Greenway es un santuario de abejas nativas (Melipona) y un proyecto de ecoturismo (una finca) en Colombia ubicada en RNT 204618, Inspección La Esperanza, La Mesa, Cundinamarca.
35 - El proyecto es una iniciativa familiar.
36 - Tu objetivo es fomentar guiar a los clientes de la página, ayudarlos y convencerlos de conocer Paraiso Mariposa y el amor por la naturaleza, especialmente las abejas, también así, aumentando nuestra
37
38 TUS REGLAS:
39 1. **Personalidad:** Eres amable, entusiasta, positivo y un poco "eco-consciente". Te apasiona la naturaleza y tratas de que siempre los usuarios reserven una experiencia, ayudándolos en el proceso.
40 """
41
42
43
44
45
46
47
48
49
50
51     def ask(self, pregunta_usuario: str) -> str:
52         """
53             Recibe una pregunta del usuario y devuelve la respuesta de la IA.
54             """
55
56         if not pregunta_usuario:
57             return "Parece que no has escrito nada. ¡Inténtalo de nuevo!"
58
59         try:
60             # Creamos la conversación
61             completion = self.client.chat.completions.create(
62                 model="gpt-4o-mini", # El modelo más nuevo, rápido y barato
63                 messages=[
64                     {"role": "system", "content": self.system_prompt},
65                     {"role": "user", "content": pregunta_usuario}
66                 ],
67                 max_tokens=150 # Límite para que no escriba demasiado
68             )
69
70             # Devolvemos solo el texto de la respuesta
71             return completion.choices[0].message.content
72
73         except Exception as e:
74             print(f"Error en la API de OpenAI: {e}")
75             # Esta es la respuesta de error que verá el usuario
76             return ";Oh, no! Parece que estoy teniendo problemas de conexión con mi cerebro de IA. Inténtalo de nuevo en un momento."
```

• main.py:

```
1 import sys
2 import os
3 from functools import wraps
4 from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify
5 from data.firebaseio_config import auth, db
6 from data.firebaseio_admin import admin_auth
7 from domain.models import Usuario, Propietaria, Admin, Persona
```

- *requirements.txt:*

1 aiohappyeyeballs==2.6.1	45 google-resumable-media==2.7.2	82 PyJWT==2.10.1
2 aiohttp==3.13.2	46 googleapis-common-protos==1.70.0	83 pyparsing==3.2.5
3 aiosignal==1.4.0	47 grpcio==1.75.1	84 Pyrebase4==4.8.0
4 alpaca-trade-api==3.2.0	48 grpcio-status==1.71.2	85 python-dateutil==2.9.0.post0
5 annotated-types==0.7.0	49 gunicorn==23.0.0	86 python-dotenv==1.1.1
6 anyio==4.11.0	50 h11==0.16.0	87 python-jwt==4.1.0
7 APScheduler==3.11.1	51 h2==4.3.0	88 pytz==2025.2
8 attrs==25.4.0	52 hpack==4.1.0	89 PyYAML==6.0.1
9 bcrypt==5.0.0	53 httpcore==1.0.9	90 requests==2.32.5
10 blinker==1.9.0	54 httplib2==0.31.0	91 requests-toolbelt==0.10.1
11 CacheControl==0.14.3	55 httpx==0.28.1	92 rsa==4.9.1
12 cachetools==6.2.1	56 hyperframe==6.1.0	93 six==1.17.0
13 certifi==2025.10.5	57 idna==3.11	94 smmap==5.0.2
14 cffi==2.0.0	58 itsdangerous==2.2.0	95 sniffio==1.3.1
15 charset-normalizer==3.4.4	59 Jinja2==3.1.6	96 sseclient==0.0.27
16 click==8.3.0	60 jiter==0.11.1	97 tqdm==4.67.1
17 colorama==0.4.6	61 jwcrypto==1.5.6	98 typing-inspection==0.4.2
18 cryptography==46.0.3	62 MarkupSafe==3.0.3	99 typing_extensions==4.15.0
19 deprecation==2.1.0	63 msgpack==1.0.3	100 tzdata==2025.2
20 distlib==0.4.0	64 multidict==6.7.0	101 tzlocal==5.3.1
21 distro==1.9.0	65 mysql-connector-python==9.4.0	102 uritemplate==4.2.0
22 filelock==3.20.0	66 mysqlclient==2.2.7	103 urllib3==1.26.20
23 firebase==4.0.1	67 numpy==2.3.4	104 virtualenv==20.35.0
24 firebase_admin==7.1.0	68 oauth2client==4.1.3	105 websocket-client==1.9.0
25 Flask==3.1.2	69 openai==2.6.1	106 websockets==10.4
26 Flask-Bcrypt==1.0.1	70 packaging==25.0	107 Werkzeug==3.1.3
27 Flask-Login==0.6.3	71 pandas==2.3.3	108 WTForms==3.2.1
28 Flask-MySQLdb==2.0.0	72 platformdirs==4.5.0	109 yarl==1.22.0
29 Flask-WTF==1.2.2	73 propcache==0.4.1	110 setuptools
30 frozenlist==1.8.0	74 proto-plus==1.26.1	111 flask-socketio
31 gcloud==0.18.3	75 protobuf==5.29.5	112 python-socketio
32 gh==0.0.4	76 pyasn1==0.6.1	113 python-engineio
33 gitdb==4.0.12	77 pyasn1_modules==0.4.2	114 eventlet
	78 pycparser==2.23	115 flask-socketio
	79 pycryptodome==3.23.0	116 eventlet
	80 pydantic==2.12.3	117 python-socketio
	81 pydantic_core==2.41.4	118 python-engineio
	82 PyJWT==2.10.1	

- *serviceAccountKey.json:*

```

1      .json
2      .env
3      serviceAccountKey.json
4      venv/
5      __pycache__/
6      *.pyc
7      .vscode/

```

Explicación para el Manual

1. Sistema de Chat en Tiempo Real

A diferencia de los sistemas tradicionales que recargan la página, Greenway utiliza una arquitectura de Escucha Activa (Listeners). El Frontend (JavaScript) se conecta directamente a los nodos de Firebase Realtime Database. Cuando un usuario envía un mensaje, este se inyecta como un objeto JSON.

Automáticamente, todos los clientes conectados reciben el evento child_added, actualizando la interfaz en milisegundos sin intervención del servidor Flask.

2. Gestión de Imágenes Híbrida

Para optimizar el almacenamiento, el sistema implementa una lógica de visualización inteligente en home.html. El sistema verifica secuencialmente:

- Si existen correcciones manuales definidas por el administrador.
- Si la experiencia tiene una lista de imágenes (Formato Nuevo).
- Si la experiencia tiene una URL única (Formato Legacy).
- Si no hay datos, aplica un placeholder automático.

3. Seguridad y Roles

El sistema implementa un control de acceso basado en roles (RBAC). Mediante el decorador personalizado @role_required en Flask, se interceptan las peticiones antes de ejecutarlas, verificando si el usuario tiene el token de sesión (admin, propietaria, usuario) adecuado para ver el panel o editar recursos.