



# Documentación Técnica

## Mental Health Telegram Bot

**Proyecto:** Bot de Telegram para Salud Mental

**Autor:** Michelle Flamenco

**Tecnologías:** Python, LangChain, Gemini AI, Telegram Bot API

**Repositorio:** [github.com/Michh8/mental-health-bot](https://github.com/Michh8/mental-health-bot)



### Tabla de Contenidos

1. Descripción General
2. Arquitectura del Sistema
3. Módulos Principales
4. Herramientas (Tools)
5. Comandos del Bot
6. Configuración
7. Deployment
8. Flujo de Ejecución

## 1. Descripción General

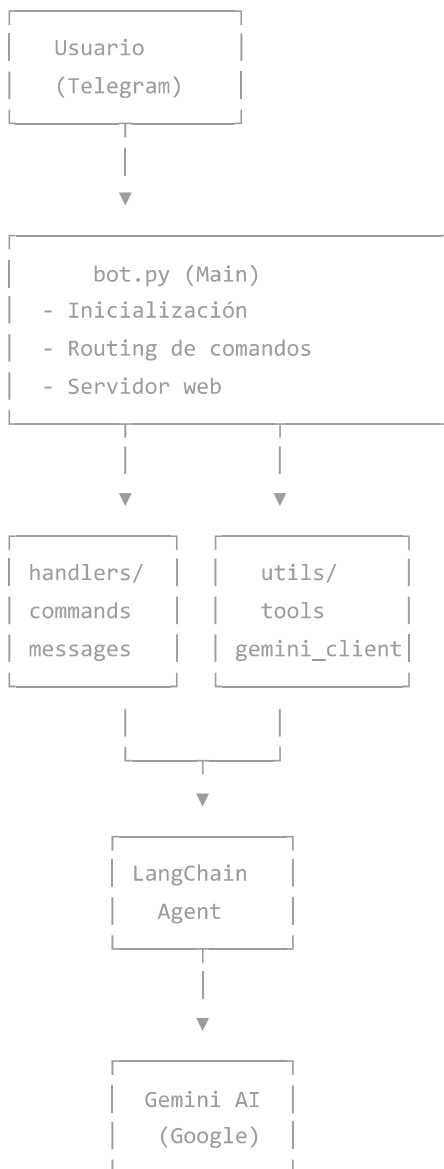
Este proyecto implementa un **bot de Telegram** orientado al bienestar emocional y salud mental. Integra **Google Gemini AI** a través de **LangChain** para proporcionar respuestas inteligentes y contextuales, además de ofrecer herramientas específicas como:

- Búsqueda de centros psicológicos y hospitales (OpenStreetMap)
- Mensajes motivacionales personalizados

- Análisis de estado de ánimo
- Información meteorológica
- Chat conversacional libre con IA

## 2. Arquitectura del Sistema

### 2.1 Diagrama de Componentes



### 2.2 Flujo de Datos

1. Usuario envía mensaje a través de Telegram

- 2. Bot recibe el mensaje y lo procesa según el tipo (comando o texto libre)
- 3. Si es comando: ejecuta el handler correspondiente
- 4. Si es texto libre: pasa por el LangChain Agent
- 5. El Agent decide si usar una Tool o responder directamente con Gemini
- 6. Se envía la respuesta al usuario

## 3. Módulos Principales

### 3.1 bot.py

Archivo Principal

**Descripción:** Punto de entrada principal del bot. Inicializa todos los componentes y mantiene el bot activo.

Componentes Clave:

Componente	Función
Application	Gestiona la aplicación de Telegram
ChatGoogleGenerativeAI	Cliente para comunicarse con Gemini
initialize_agent	Crea el agente de LangChain con las tools
run_webserver	Servidor HTTP para mantener activo el bot en Render/Railway

Verificación de Dependencias:

Al iniciar, el bot verifica las versiones de paquetes críticos:

```
required = {
    "python-telegram-bot": ">=20.0",
    "langchain-google-genai": ">=0.0.9",
    "python-dotenv": ">=1.0.0",
    "google-generativeai": ">=0.8.3"
}
```

## Configuración del Agente:

```
agent = initialize_agent(  
    tools_list, # Herramientas personalizadas  
    llm,        # Modelo Gemini  
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,  
    verbose=True  
)
```

**Nota:** El agente usa `ZERO_SHOT_REACT_DESCRIPTION`, lo que significa que puede razonar sobre qué tool usar basándose solo en las descripciones, sin ejemplos previos.

## Función de Chat:

```
async def chat(update, context):  
    user_message = update.message.text  
    response = agent.run(user_message)  
    await update.message.reply_text(response)
```

## 3.2 config.py

### Configuración

**Descripción:** Centraliza la carga y validación de variables de entorno.

### Variables Requeridas:

- `TELEGRAM_TOKEN` - Token del bot de Telegram
- `GEMINI_API_KEY` - API Key de Google Gemini
- `WEATHER_API_KEY` - API Key de OpenWeatherMap

 **Validación Estricta:** El módulo lanza excepciones si falta alguna variable crítica.

## 3.3 handlers/commands.py

### Comandos

**Descripción:** Define todos los comandos disponibles para el usuario.

### Comandos Implementados:

Comando	Función	Descripción
/start	start()	Mensaje de bienvenida inicial
/help	help_command()	Lista todas las funcionalidades disponibles
/fecha	fecha()	Muestra fecha y hora actual (zona El Salvador)
/clima [ciudad]	clima()	Obtiene información meteorológica
/motivacion	motivacion()	Genera mensaje motivacional con IA
/mood [estado]	mood()	Analiza estado de ánimo y da consejos
/centros [ciudad]	centros()	Busca psicólogos y hospitales cercanos

## Función Auxiliar - Mensajes Largos:

Maneja el límite de 4096 caracteres de Telegram dividiendo mensajes largos:

```
async def enviar_texto_largo(update: Update, texto: str):
    MAX_CHARS = 4096
    if len(texto) <= MAX_CHARS:
        await update.message.reply_text(texto)
    else:
        for i in range(0, len(texto), MAX_CHARS):
            await update.message.reply_text(texto[i:i+MAX_CHARS])
```

## Manejo de Zona Horaria:

Usa `zoneinfo` para mostrar la hora correcta de El Salvador:

```
from zoneinfo import ZoneInfo
now = datetime.now(ZoneInfo("America/El_Salvador"))
```

## 3.4 handlers/messages.py

### Chat Libre

**Descripción:** Maneja mensajes de texto que no son comandos, usando el agente de LangChain.

## Características:

- Usa `ConversationBufferMemory` para mantener contexto conversacional
- Integra todas las tools disponibles
- El agente decide automáticamente si usar una tool o responder directamente

```
memory = ConversationBufferMemory(
    memory_key="chat_history",
    return_messages=True
)

agent = initialize_agent(
    tools=tools_list,
    llm=llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    memory=memory,
    verbose=True
)
```

## 4. Herramientas (Tools)

---

Las tools son funciones especializadas que el agente puede invocar cuando sea necesario.

### 4.1 PsychCentersTool

**Búsqueda Geográfica**      **Overpass API**

**Función:** Busca centros psicológicos, clínicas y hospitales en una ciudad específica.

#### Implementación:

```
def find_psych_centers(location: str) -> str:
    query = f"""
    [out:json][timeout:25];
    area["name"="{location}"]->.searchArea;
    (
        node(area.searchArea)["amenity"~"clinic|hospital"];
        way(area.searchArea)["amenity"~"clinic|hospital"];
        relation(area.searchArea)["amenity"~"clinic|hospital"];
    );
    out center;
    """
    # Envía query a Overpass API
```

```
# Filtra resultados con palabras clave de psicología
# Devuelve lista con nombres y coordenadas
```

## Filtrado Inteligente:

Prioriza resultados que contengan palabras clave relacionadas con salud mental:

```
keywords = ["psico", "mental", "salud", "psychiatry", "psychology"]
```

## Ejemplo de Salida:

- Centro de Psicología San Salvador - 📍 Lat: 13.6929, Lon: -89.2182
- Hospital Psiquiátrico Nacional - 📍 Lat: 13.7130, Lon: -89.2069
- Clínica de Salud Mental - 📍 Lat: 13.6988, Lon: -89.2253

## 4.2 MotivationTool

IA Generativa    Gemini AI

**Función:** Genera mensajes motivacionales personalizados usando Gemini AI.

### Prompt del Sistema:

```
prompt = f"""
Eres un asistente de apoyo emocional breve.
Tu tarea es:
1. Dar un mensaje de motivación cálido y comprensivo.
2. Sugerir una acción práctica para mejorar el bienestar emocional.
3. Si detectas señales de desesperanza extrema, responde de forma
   empática y sugiere buscar ayuda profesional.

Usuario: "{query}"
"""
```

### Fallback de Seguridad:

Si falla la conexión con Gemini, usa mensajes predefinidos:

```
frases = [
    "💖 Recuerda que no estás solo/a...",
    "🧘 Respira profundo tres veces...",
    "🌟 Cada día trae una nueva oportunidad..."
]
```

## 4.3 MoodCheckTool

**Análisis Emocional**    **Gemini AI**

**Función:** Analiza el estado de ánimo descrito por el usuario y proporciona consejos.

**Proceso:**

1. Recibe descripción del estado emocional
2. Envía a Gemini para análisis contextual
3. Devuelve consejo personalizado
4. Si falla IA, usa análisis de palabras clave básico

**Fallback Basado en Palabras Clave:**

```
if "triste" in description or "deprimido" in description:
    return "😞 Parece que te sientes triste. Da un pequeño paseo."
if "estresado" in description or "ansioso" in description:
    return "😓 Estás estresado. Medita o escucha música relajante."
if "feliz" in description or "bien" in description:
    return "😊 Me alegra que te sientas bien. Mantén esa energía positiva."
```

## 4.4 WeatherTool

**API Externa**    **OpenWeatherMap**

**Función:** Obtiene información meteorológica actual de cualquier ciudad.

**Implementación:**

```
url = f"http://api.openweathermap.org/data/2.5/weather"
params = {
    "q": ciudad,
    "appid": WEATHER_API_KEY,
    "units": "metric",
    "lang": "es"
}
```

**Datos Devueltos:**

- Temperatura actual (°C)
- Condición meteorológica
- Humedad relativa



- País y nombre de la ciudad

## 4.5 Registro de Tools

Todas las tools se registran en una lista para el agente:

```
tools_list = [  
    psych_tool,  
    motivation_tool,  
    mood_tool,  
    weather_tool  
]
```

## 5. Comandos del Bot

---

### Ejemplos de Uso

#### Comando /fecha

Usuario: /fecha

Bot: 📅 Hoy es Viernes, 03 de octubre de 2025  
🕒 Hora: 14:35:22

#### Comando /clima

Usuario: /clima San Salvador

Bot: ☀️ Clima en San Salvador, SV:  
🌡️ Temperatura: 28°C  
💧 Humedad: 65%  
📖 Condición: Parcialmente nublado

#### Comando /motivacion

Usuario: /motivacion

Bot: 💙 Hoy es un nuevo comienzo. Recuerda que cada pequeño paso cuenta, no importa qué tan grande sea el desafío.

💡 Acción práctica: Escribe 3 cosas por las que estés agradecido hoy. Este simple ejercicio puede cambiar tu perspectiva.

## Comando /mood

Usuario: /mood me siento muy estresado por el trabajo

Bot: 😞 Entiendo que el estrés laboral puede ser abrumador.  
Es importante que te tomes pequeños descansos durante el día.

💡 Consejos:

- Practica respiración profunda (4-7-8)
- Sal a caminar 10 minutos
- Habla con alguien de confianza

Si el estrés persiste, considera buscar apoyo profesional.

## Comando /centros

Usuario: /centros San Salvador

Bot: 📍 Centros psicológicos encontrados en San Salvador:

- Centro de Psicología Integral - 📍 Lat: 13.6929, Lon: -89.2182
- Hospital Psiquiátrico Nacional - 📍 Lat: 13.7130, Lon: -89.2069
- Clínica Mental Health Center - 📍 Lat: 13.6988, Lon: -89.2253
- Consultorio Psicológico López - 📍 Lat: 13.7025, Lon: -89.2145

# 6. Configuración

---

## 6.1 Variables de Entorno (.env)

```
TELEGRAM_TOKEN=123456789:ABCdefGHIjklMNOpqrsTUVwxyz  
GEMINI_API_KEY=AiZaSyAaBbCcDdEeFfGgHhIiJjKkLlMm  
WEATHER_API_KEY=1234567890abcdef1234567890abcdef
```

## 6.2 Obtención de API Keys

### Telegram Bot Token:

1. Habla con [@BotFather](#) en Telegram
2. Envía /newbot
3. Sigue las instrucciones
4. Copia el token proporcionado

### Google Gemini API Key:

1. Ve a [Google AI Studio](#)
2. Inicia sesión con tu cuenta de Google
3. Crea una nueva API Key
4. Cópiala al archivo .env

### OpenWeatherMap API Key:

1. Regístrate en [OpenWeatherMap](#)
2. Ve a "API Keys" en tu perfil
3. Genera una nueva key
4. Cópiala al archivo .env

## 6.3 Dependencias (requirements.txt)

```
python-telegram-bot>=20.0
langchain>=0.1.0
langchain-google-genai>=0.0.9
python-dotenv>=1.0.0
google-generativeai>=0.8.3
requests>=2.31.0
aiohttp>=3.9.0
```

## 7. Deployment

---

### 7.1 Servidor Web Integrado

El bot incluye un servidor HTTP para mantenerse activo en plataformas cloud:

```
async def run_webserver():
    app = web.Application()
    app.router.add_get("/", handle)
    port = int(os.environ.get("PORT", 10000))
    runner = web.AppRunner(app)
    await runner.setup()
    site = web.TCPSite(runner, "0.0.0.0", port)
    await site.start()
```

### 7.2 Deployment en Render

1. Crea una cuenta en [Render.com](https://render.com)
2. Conecta tu repositorio de GitHub
3. Crea un nuevo "Web Service"
4. Configura las variables de entorno
5. Build Command: `pip install -r requirements.txt`
6. Start Command: `python bot.py`

## 7.3 Deployment en Railway

1. Crea una cuenta en [Railway.app](https://railway.app)
2. Crea un nuevo proyecto desde GitHub
3. Configura las variables de entorno
4. Railway detectará automáticamente el `runtime.txt`
5. Deploy automático en cada push

## 7.4 Dockerfile

El proyecto incluye un Dockerfile para containerización:

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["python", "bot.py"]
```

# 8. Flujo de Ejecución

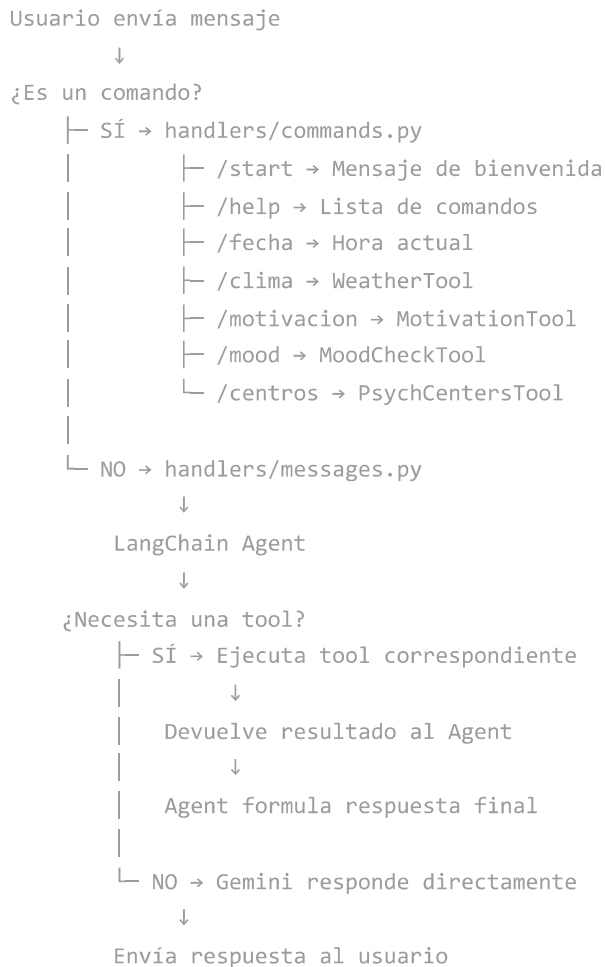
---

## 8.1 Inicialización del Bot

1. Cargar variables de entorno (.env)
2. Verificar dependencias instaladas
3. Conectar con Gemini API
4. Inicializar modelo LangChain
5. Crear Agent con tools
6. Configurar Application de Telegram
7. Registrar handlers (comandos y mensajes)
8. Iniciar polling de Telegram

9. Iniciar servidor web
10. Mantener bot activo

## 8.2 Procesamiento de Mensajes



## 8.3 Ejemplo de Flujo Completo

**Escenario:** Usuario pregunta "¿Dónde puedo encontrar un psicólogo en San Salvador?"

1. Usuario envía: "¿Dónde puedo encontrar un psicólogo en San Salvador?"  
↓
2. handlers/messages.py recibe el mensaje  
↓
3. Se pasa al LangChain Agent  
↓
4. Agent analiza el mensaje y detecta que necesita buscar psicólogos  
↓
5. Agent decide usar PsychCentersTool  
↓

```
6. PsychCentersTool hace query a Overpass API
↓
7. API devuelve resultados de OpenStreetMap
↓
8. Tool filtra y formatea los resultados
↓
9. Agent recibe los resultados y genera respuesta natural
↓
10. Bot envía respuesta al usuario con lista de centros
```

## 9. Consideraciones de Seguridad

 **Importante:** Este bot maneja información sensible relacionada con salud mental.

### 9.1 Protección de Datos

- Nunca compartir el archivo `.env` en repositorios públicos
- Usar `.gitignore` para excluir archivos sensibles
- Rotar API keys periódicamente
- No almacenar conversaciones sin consentimiento

### 9.2 Limitaciones del Bot

**Disclaimer:** Este bot NO reemplaza atención profesional. Es una herramienta de apoyo básico.

- No está capacitado para manejar crisis de salud mental severas
- Debe sugerir ayuda profesional cuando sea necesario
- No debe dar diagnósticos ni tratamientos médicos

### 9.3 Respuesta ante Crisis

El `MotivationTool` está programado para detectar señales de riesgo:

```
Si detectas señales de desesperanza extrema o pensamientos de autolesión,
responde de forma empática y sugiere buscar ayuda profesional o llamar
```

a una línea de emergencia.

## 10. Mantenimiento y Actualizaciones

---

### 10.1 Logs y Debugging

El bot usa `logging` de Python para registrar eventos:

```
logging.basicConfig(
    format="% (asctime)s - %(name)s - %(levelname)s - %(message)s",
    level=logging.INFO
)
```

### 10.2 Monitoreo de Errores

Todos los comandos y tools tienen manejo de excepciones:

```
try:
    # Operación
except Exception as e:
    logging.exception("Error en [componente]")
    await update.message.reply_text("⚠ Error al procesar...")
```

### 10.3 Actualización de Modelos

Para cambiar el modelo de Gemini, modifica en `bot.py`:

```
llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-pro", # ← Cambiar aquí
    google_api_key=GEMINI_API_KEY
)
```

Modelos disponibles:

- `gemini-pro` - Modelo estándar
- `gemini-2.5-pro` - Versión mejorada (actual)
- `gemini-ultra` - Máximo rendimiento (cuando esté disponible)

# 11. Preguntas Frecuentes (FAQ)

---

## ? ¿Por qué el bot no responde?

**Respuesta:** Verifica:

- Que todas las API keys sean válidas
- Que el bot esté ejecutándose ( `python bot.py` )
- Que no haya errores en los logs
- Que el servidor web esté activo (si está en cloud)

## ? ¿Cómo añadir nuevos comandos?

**Respuesta:**

1. Define la función en `handlers/commands.py`
2. Registra el handler en `bot.py` :

```
app.add_handler(CommandHandler("nuevo_comando", commands.nueva_funcion))
```

## ? ¿Cómo añadir nuevas tools?

**Respuesta:**

1. Crea la función en `utils/tools.py`
2. Crea el objeto Tool:

```
nueva_tool = Tool(  
    name="NombreTool",  
    description="Descripción clara de qué hace",  
    func=tu_funcion  
)
```

3. Agrégala a `tools_list`

## ? ¿El bot guarda las conversaciones?

**Respuesta:** No permanentemente. Usa `ConversationBufferMemory` que mantiene el contexto solo durante la sesión activa. Al reiniciar el bot, se pierde el historial.

## ? ¿Puedo cambiar la zona horaria?











**Respuesta:** Sí, en `handlers/commands.py`, modifica:

```
now = datetime.now(ZoneInfo("America/El_Salvador")) # ← Cambiar aquí
```

## 12. Roadmap y Mejoras Futuras

---

### Funcionalidades Planeadas

-  Persistencia de conversaciones con base de datos
-  Dashboard de estadísticas de uso
-  Soporte multiidioma
-  Recordatorios de bienestar programados
-  Integración con otras plataformas (WhatsApp, Discord)
-  Personalización de respuestas según perfil de usuario
-  Tracking de estado de ánimo a largo plazo
-  Conexión directa con profesionales de salud mental

## 13. Contribuciones

---

Las contribuciones son bienvenidas. Para contribuir:

1. Fork el repositorio
2. Crea una rama: `git checkout -b feature/nueva-funcionalidad`
3. Commit tus cambios: `git commit -m 'Añade nueva funcionalidad'`
4. Push a la rama: `git push origin feature/nueva-funcionalidad`
5. Abre un Pull Request

## 14. Licencia

---

Este proyecto está bajo la Licencia MIT. Ver archivo `LICENSE` para más detalles.

## 15. Contacto y Soporte

---

**Desarrolladora:** Michelle Flamenco

**Email:** michelleflamenko@yahoo.com

**GitHub:** [@Michh8](#)

**Bot Demo:** [@mental\\_health\\_guy\\_bot](#)

## 16. Agradecimientos

---

Este proyecto utiliza las siguientes tecnologías y servicios:

- **Google Gemini AI** - Motor de inteligencia artificial
  - **LangChain** - Framework para aplicaciones con LLM
  - **Telegram Bot API** - Plataforma de mensajería
  - **OpenStreetMap** - Datos geográficos abiertos
  - **OpenWeatherMap** - Información meteorológica
- 

**Documentación generada el 03 de octubre de 2025**

Versión 1.0 - Mental Health Telegram Bot