WhatsApp Bot Python Project Structure

Python WhatsApp bot using Meta Cloud API. Webhook-based message handling with FastAPI and modular command processors.

#whatsapp #python #bot #meta-api #webhooks #fastapi

PNG PDF Copy Prompt

Project Directory

```
whatsapp-bot/
  src/ Bot source code
    __init__.py
    main.py FastAPI app, we…
    handlers/ Message handlers
      __init__.py
      text.py Text message ha…
      media.py Image/video/aud…
      interactive.py Buttons, lists
      location.py
    commands/ Bot commands
      __init__.py
      start.py /start command
      help.py
      menu.py Interactive menu
    services/ Business logic
      __init__.py
      whatsapp.py API client wrap…
      message_queue.py Rate limiting
    utils/
      __init__.py
      validators.py Webhook signatu…
      templates.py Message templat…
    config.py Settings from e…
  tests/
    __init__.py
    conftest.py
    test_handlers.py
    test_webhook.py
  pyproject.toml
  .env.example API tokens here
  .gitignore
  README.md
```

Why This Structure?

This structure uses Meta's Cloud API with webhooks—the official way to build WhatsApp bots. FastAPI handles incoming webhooks, handlers process different message types, and services abstract the WhatsApp API. Commands are separated for easy extension.

Key Directories

src/main.py - FastAPI app with webhook verification and message routing

src/handlers/ - Process text, media, interactive messages separately

src/commands/ - Bot commands triggered by keywords or menus

src/services/whatsapp.py - Wrapper for sending messages via Cloud API

Getting Started

1. `uv init whatsapp-bot && cd whatsapp-bot`
2. `uv add fastapi uvicorn httpx python-dotenv`
3. `cp .env.example .env` and add your Meta API credentials
4. `uv run uvicorn src.main:app --reload`
5. Configure webhook URL in Meta Developer Console

Message Handler

```python
# src/handlers/text.py
from src.services.whatsapp import WhatsAppClient

async def handle_text(client: WhatsAppClient, message: dic
    text = message["text"]["body"].lower()
    sender = message["from"]

    if text.startswith("/"):
        # Route to command handler
        return await handle_command(client, sender, text)

    # Echo or process with AI
    await client.send_text(sender, f"You said: {text}")
```

Webhook Verification

Meta requires webhook verification via GET request with `hub.verify_token`. All incoming messages arrive as POST with signature in `X-Hub-Signature-256` header. Always verify signatures before processing.

When To Use This

- Building official WhatsApp Business bots
- Customer support or notification systems
- Projects needing message templates and interactive elements
- Teams already using Python async/FastAPI

Best Practices

- Use message templates for outbound notifications (required by Meta)
- Implement rate limiting—WhatsApp has strict limits
- Store conversation state in Redis for multi-message flows
- Handle webhook retries gracefully (idempotency)
- Use `WHATSAPP_VERIFY_TOKEN` for webhook setup

Trade-offs

Business API required - Need Meta Business verification for production

Template approval - Outbound messages need pre-approved templates

24-hour window - Free-form replies only within 24h of user message