WhatsApp Bot TypeScript Project Structure

TypeScript WhatsApp bot using Baileys. Multi-device protocol, session persistence, and modular command architecture.

#whatsapp #typescript #bot #baileys #nodejs #multi-device

PNG PDF Copy Prompt

Project Directory

```
whatsapp-bot/
  src/ Bot source code
    index.ts Entry point, co…
    client/ WhatsApp client
      connection.ts Baileys setup
      events.ts Message events
      auth.ts Session handling
    commands/ Bot commands
      index.ts Command loader
      ping.ts
      help.ts
      sticker.ts Media to sticker
    handlers/ Message handlers
      message.ts Route messages
      group.ts Group events
      media.ts
    utils/
      logger.ts
      helpers.ts
      media.ts Download/proces…
    types/
      index.ts
      commands.ts
    config.ts Bot configurati…
  auth/ Session data
    .gitkeep
  package.json
  tsconfig.json
  .env.example
  .gitignore
  README.md
```

Why This Structure?

This structure uses Baileys—a reverse-engineered WhatsApp Web API supporting multi-device. No Business API needed. The client/ folder handles connection and auth, commands/ holds bot features, and auth/ persists session across restarts.

Key Directories

src/client/ - Baileys connection, auth state, and event binding

src/commands/ - Command files loaded dynamically at startup

src/handlers/ - Route incoming messages to appropriate handlers

auth/ - Session files (creds.json) for persistent login

Getting Started

1. `npm init -y && npm install @whiskeysockets/baileys pino`
2. `npm install -D typescript @types/node tsx`
3. Scan QR code on first run to authenticate
4. `npx tsx src/index.ts`

Command Structure

```typescript
// src/commands/ping.ts
import { WASocket, proto } from "@whiskeysockets/baileys";

export const name = "ping";
export const description = "Check if bot is alive";

export async function execute(
  sock: WASocket,
  msg: proto.IWebMessageInfo
) {
  const jid = msg.key.remoteJid!;
  await sock.sendMessage(jid, { text: "Pong!" });
}
```

Session Persistence

Baileys uses `useMultiFileAuthState()` to persist session in the auth/ folder. On first run, scan the QR code with WhatsApp mobile. Session survives restarts—no need to re-scan unless you clear auth/.

When To Use This

- Personal or small-scale WhatsApp bots
- Projects not needing Business API approval
- Bots requiring full message access (not just templates)
- Quick prototyping without Meta verification

Trade-offs

Unofficial API - Risk of account ban if abused—follow WhatsApp ToS

No templates - Cannot use official message templates or buttons

Session management - Need to handle reconnection and session expiry

Best Practices

- Never spam—respect rate limits and user consent
- Store auth/ folder securely, never commit to git
- Implement reconnection logic for dropped connections
- Use `pino` logger with `level: 'silent'` to reduce noise
- Handle `connection.update` events for connection state