🤖 بوت إدارة المجموعات المتقدم

بوت تليجرام احترافي لإدارة المجموعات مع ميزات متقدمة، مطور باستخدام Python و aiogram 3.x مع قاعدة بيانات Supabase.

✨ الميزات الرئيسية

🛡️ الحماية والأمان

- **مكافحة السبام التلقائي** - كتم المستخدمين عند تجاوز حد الرسائل
- **منع الروابط** - حذف الروابط غير المرغوب فيها مع إمكانية إضافة استثناءات
- **منع الكلمات المحظورة** - فلترة الكلمات غير المناسبة
- **نظام التحذيرات** - تحذيرات تراكمية مع عقوبات تلقائية

👮 الأوامر الإدارية

- **حظر/إلغاء حظر** - إدارة شاملة للعضويات
- **كتم/إلغاء كتم** - مع دعم المدة الزمنية
- **طرد الأعضاء** - إزالة مؤقتة من المجموعة
- **تثبيت الرسائل** - إدارة الرسائل المثبتة
- **نظام الرتب** - إدارة صلاحيات مخصصة

👋 الترحيب والتفاعل

- **رسائل ترحيب قابلة للتخصيص** - مع متغيرات ديناميكية
- **أزرار تفاعلية** - واجهة سهلة الاستخدام
- **معاينة الرسائل** - اختبار الرسائل قبل التطبيق

📊 الإحصائيات والتقارير

- **إحصائيات شاملة** - أعضاء، أنشطة، إجراءات
- **تتبع الأنشطة** - مراقبة تفاعل الأعضاء
- **تقارير دورية** - ملخصات الأداء

🚀 التثبيت والإعداد

1. متطلبات النظام

```
Python 3.8+
PostgreSQL (من خلال Supabase)
Redis (للكاش اختياري)
```

2. تحميل المشروع

```
git clone <repository-url>
cd telegram_bot
```

3. تثبيت المتطلبات

```
pip install -r requirements.txt
```

4. إعداد متغيرات البيئة

انسخ ملف `.env.example` إلى `.env` واملأ البيانات:

```
cp .env.example .env
```

5. إعداد قاعدة البيانات Supabase

أ. إنشاء مشروع Supabase

1. اذهب إلى supabase.com
2. أنشئ حساب جديد أو ادخل لحسابك
3. أنشئ مشروع جديد
4. احصل على URL و API Key من إعدادات المشروع

ب. إنشاء الجداول

انسخ والصق SQL التالي في SQL Editor في Supabase:

```
-- جدول المستخدمين
CREATE TABLE IF NOT EXISTS users (
    user_id BIGINT PRIMARY KEY,
    username TEXT,
    first_name TEXT,
    last_name TEXT,
    language_code TEXT DEFAULT 'ar',
    is_bot BOOLEAN DEFAULT FALSE,
    is_premium BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

-- جدول المجموعات
CREATE TABLE IF NOT EXISTS chats (
    chat_id BIGINT PRIMARY KEY,
    title TEXT,
    chat_type TEXT DEFAULT 'group',
    username TEXT,
    description TEXT,
    welcome_enabled BOOLEAN DEFAULT TRUE,
    welcome_message TEXT,
    antiflood_enabled BOOLEAN DEFAULT TRUE,
    antilink_enabled BOOLEAN DEFAULT FALSE,
    antiword_enabled BOOLEAN DEFAULT FALSE,
    warns_enabled BOOLEAN DEFAULT TRUE,
    max_warns INTEGER DEFAULT 3,
    banned_words JSONB DEFAULT '[]',
    allowed_links JSONB DEFAULT '[]',
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

-- جدول المشرفين
CREATE TABLE IF NOT EXISTS admins (
```

```
    id SERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL,
    chat_id BIGINT NOT NULL,
    rank TEXT DEFAULT 'admin',
    title TEXT,
    can_delete_messages BOOLEAN DEFAULT TRUE,
    can_restrict_members BOOLEAN DEFAULT TRUE,
    can_promote_members BOOLEAN DEFAULT FALSE,
    can_change_info BOOLEAN DEFAULT FALSE,
    can_invite_users BOOLEAN DEFAULT TRUE,
    can_pin_messages BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT NOW(),
    UNIQUE(user_id, chat_id)
);

-- جدول التحذيرات
CREATE TABLE IF NOT EXISTS warnings (
    id SERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL,
    chat_id BIGINT NOT NULL,
    admin_id BIGINT NOT NULL,
    reason TEXT,
    warn_count INTEGER DEFAULT 1,
    created_at TIMESTAMP DEFAULT NOW()
);

-- جدول الحظر
CREATE TABLE IF NOT EXISTS bans (
    id SERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL,
    chat_id BIGINT NOT NULL,
    admin_id BIGINT NOT NULL,
    reason TEXT,
    ban_type TEXT DEFAULT 'ban',
    duration INTEGER,
```

```
    expires_at TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT NOW()
);

-- جدول الفلاتر
CREATE TABLE IF NOT EXISTS filters (
    id SERIAL PRIMARY KEY,
    chat_id BIGINT NOT NULL,
    keyword TEXT NOT NULL,
    response TEXT NOT NULL,
    filter_type TEXT DEFAULT 'text',
    media_file_id TEXT,
    created_by BIGINT NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);

-- جدول الملاحظات
CREATE TABLE IF NOT EXISTS notes (
    id SERIAL PRIMARY KEY,
    chat_id BIGINT NOT NULL,
    name TEXT NOT NULL,
    content TEXT NOT NULL,
    note_type TEXT DEFAULT 'text',
    media_file_id TEXT,
    created_by BIGINT NOT NULL,
    created_at TIMESTAMP DEFAULT NOW(),
    UNIQUE(chat_id, name)
);

-- جدول الإعدادات
CREATE TABLE IF NOT EXISTS settings (
    chat_id BIGINT PRIMARY KEY,
    settings_data JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT NOW(),
```

```
    updated_at TIMESTAMP DEFAULT NOW()
);
```

6. إعداد ملف .env

```
# إعدادات البوت
BOT_TOKEN=your_bot_token_from_botfather
BOT_USERNAME=your_bot_username

# إعدادات Supabase
SUPABASE_URL=your_supabase_project_url
SUPABASE_KEY=your_supabase_anon_key

# إعدادات المطورين (ضع معرف تليجرام الخاص بك)
SUDO_USERS=123456789

# إعدادات اختيارية
REDIS_URL=redis://localhost:6379/0
WEBHOOK_URL=https://yourdomain.com
```

7. تشغيل البوت

```
python main.py
```

🌐 النشر على PythonAnywhere

1. رفع الملفات

```
# رفع الملفات إلى PythonAnywhere
scp -r telegram_bot/ username@ssh.pythonanywhere.com:/home/
username/
```

2. تثبيت المتطلبات

```
pip3.10 install --user -r requirements.txt
```

3. إعداد متغيرات البيئة

```
# إنشاء ملف .env في المجلد الرئيسي
nano .env
```

4. إعداد Task المجدولة

في لوحة تحكم PythonAnywhere، اذهب إلى "Tasks" وأنشئ task جديدة:

```
cd /home/username/telegram_bot && python3.10 main.py
```

5. إعداد الويب هوك (اختياري)

لاستخدام webhooks بدلاً من polling:

```
# في web tab، أنشئ Flask app
from main import bot, dp
from aiohttp import web
from aiohttp.web import Application

app = Application()
# إعداد webhook
```

📋 دليل الاستخدام

الأوامر الأساسية

أوامر عامة

- `/start` - بدء البوت
- `/help` - عرض المساعدة
- `/admin` - لوحة الإدارة الرئيسية
- `/info` - معلومات المجموعة
- `/ping` - اختبار البوت

أوامر الإدارة

```
/ban @username الحظر سبب          # حظر مستخدم
/unban @username                  # إلغاء حظر
/kick @username الطرد سبب         # طرد مستخدم
/mute @username 30m الكتم سبب      # كتم لمدة 30 دقيقة
/unmute @username                 # إلغاء كتم
/warn @username التحذير سبب        # تحذير مستخدم
/unwarn @username                 # مسح التحذيرات
/pin                              # تثبيت رسالة (رد على الرسالة)
/unpin                            # إلغاء تثبيت
```

أوامر الحماية

```
/protection                       # لوحة إعدادات الحماية
/antiflood                        # تشغيل/إيقاف مكافحة السبام
/antilink                         # تشغيل/إيقاف منع الروابط
/antiword                         # تشغيل/إيقاف منع الكلمات
/addword كلمة_محظورة               # إضافة كلمة محظورة
/removeword كلمة_محظورة            # إزالة كلمة محظورة
/addlink example.com              # إضافة رابط مسموح
```

أوامر الترحيب

```
/welcome                          # إعدادات الترحيب
/setwelcome مرحباً {first_name}!  # تعيين رسالة ترحيب
/resetwelcome                     # إعادة تعيين للافتراضي
/togglewelcome                    # تشغيل/إيقاف الترحيب
/testwelcome                      # اختبار رسالة الترحيب
```

تنسيقات الوقت

- `s` = ثواني
- `m` = دقائق
- `h` = ساعات
- `d` = أيام
- `w` = أسابيع

أمثلة:
- `30s` = 30 ثانية
- `5m` = 5 دقائق
- `2h` = ساعتان
- `1d` = يوم واحد

متغيرات رسائل الترحيب

- `{first_name}` - الاسم الأول
- `{last_name}` - الاسم الأخير
- `{full_name}` - الاسم الكامل
- `{username}` - اسم المستخدم
- `{user_id}` - معرف المستخدم
- `{chat_title}` - اسم المجموعة
- `{chat_id}` - معرف المجموعة

مثال على رسالة ترحيب:

```
🌟 أهلاً وسهلاً {first_name}!

مرحباً بك في {chat_title}
نتمنى لك وقتاً ممتعاً معنا 🎉

معرفك: {user_id}
```

التخصيص والتطوير🔧

هيكل المشروع

```
telegram_bot/
├── main.py                 # ملف البوت الرئيسي
├── config/                 # إعدادات البوت
│   ├── __init__.py
│   └── config.py
├── database/               # قاعدة البيانات
│   ├── __init__.py
│   ├── database.py
│   └── models.py
├── handlers/               # معالجات الأوامر
│   ├── __init__.py
│   ├── admin_handlers.py
│   ├── protection_handlers.py
│   ├── welcome_handlers.py
│   ├── general_handlers.py
│   └── callback_handlers.py
├── filters/                # فلاتر مخصصة
│   ├── __init__.py
│   └── admin_filter.py
├── keyboards/              # لوحات المفاتيح
│   ├── __init__.py
│   └── admin_keyboards.py
├── utils/                  # وظائف مساعدة
│   ├── __init__.py
│   └── helpers.py
├── requirements.txt        # متطلبات Python
├── .env.example           # مثال متغيرات البيئة
└── README.md              # هذا الملف
```

إضافة ميزات جديدة

1. إنشاء معالج جديد

```
# في handlers/new_feature.py
from aiogram import Router
from aiogram.types import Message
from aiogram.filters import Command

new_feature_router = Router()

@new_feature_router.message(Command("newcommand"))
async def new_command(message: Message):
    await message.reply("ميزة جديدة!")
```

2. تسجيل المعالج في main.py

```
from handlers.new_feature import new_feature_router

# في main.py
dp.include_router(new_feature_router)
```

تخصيص الرسائل

يمكنك تعديل الرسائل في `config/config.py`:

```
EMOJI = {
    "check": "✅",
    "cross": "❌",
    # ... إضافة رموز جديدة
}
```

🐛 استكشاف الأخطاء

مشاكل شائعة وحلولها

1. خطأ في الاتصال بقاعدة البيانات

```
خطأ: connection refused
```

الحل: تأكد من صحة `SUPABASE_URL` و `SUPABASE_KEY`

2. البوت لا يرد على الأوامر

```
البوت غير نشط
```

الحل:
- تأكد من صحة `BOT_TOKEN`
- تأكد من إضافة البوت للمجموعة كمشرف
- تحقق من السجلات (logs)

3. صلاحيات غير كافية

```
ليس لديك صلاحية
```

الحل:
- أضف معرفك إلى `SUDO_USERS`
- تأكد من كونك مشرف في المجموعة

تفعيل السجلات التفصيلية

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

📞 الدعم والمساعدة

طرق التواصل

- **GitHub Issues**: لتقارير الأخطاء والاقتراحات
- **Telegram**: [@your_support_username]
- **Email**: support@yourproject.com

المساهمة في المشروع

1. Fork المشروع
2. أنشئ فرع جديد للميزة
3. اكتب الكود واختبره
4. أرسل Pull Request

ترخيص المشروع

هذا المشروع مرخص تحت MIT License

🔄 التحديثات المستقبلية

ميزات قادمة

- [ ] نظام النقاط والمكافآت
- [ ] تكامل مع APIs خارجية

- لوحة تحكم ويب [ ]
- نظام الإشعارات المتقدم [ ]
- دعم قواعد بيانات إضافية [ ]
- تحليلات متقدمة [ ]
- دعم متعدد اللغات [ ]
- نظام البلاغات [ ]

سجل التغييرات

v1.0.0 (2024-01-XX)

- إطلاق النسخة الأولى
- ميزات الإدارة الأساسية
- نظام الحماية
- رسائل الترحيب
- واجهة إدارية تفاعلية

تم التطوير بواسطة MiniMax Agent 🤖

للحصول على أحدث التحديثات، تابع المشروع على GitHub.