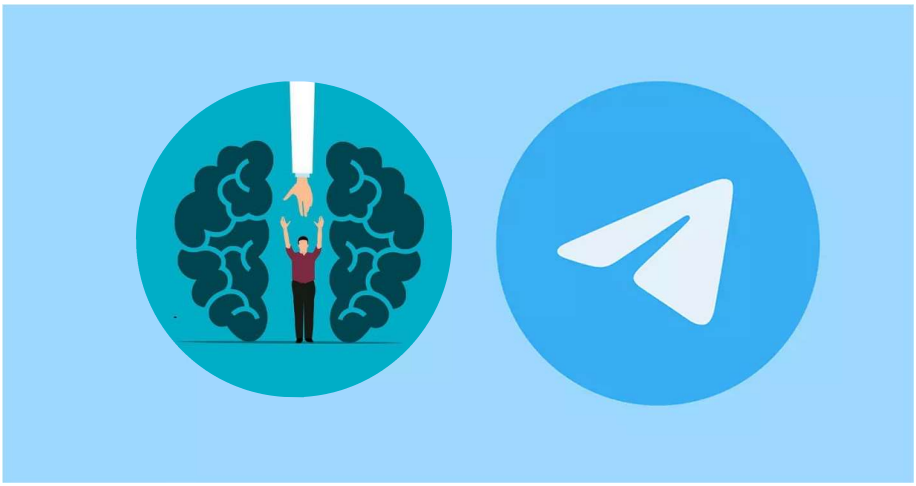
UNIVERSIDAD NACIONAL DE SALTA

FACULTAD DE CS. EXACTAS

DPTO DE INFORMÁTICA

Salta 25-06-2025

PROYECTO FINAL DE INTRODUCCION AL DESARROLLO DE SOFTWARE LIBRE



Bot de Telegram para gestión de Hábitos Inteligentes y Tareas

Autores: Martín Elías ROLDÁN

Carlos Andrés ARANCIBIA DÍAZ

ÍNDICE:

INTRODUCCIÓN:

El presente trabajo describe el desarrollo de un bot de Telegram diseñado para asistir a los usuarios en la adopción y mantenimiento de hábitos saludables e inteligentes, así como en la gestión de recordatorios de tareas. Este proyecto se concibe como una herramienta accesible y de apoyo para fomentar el bienestar personal. La concepción de este bot surge de la necesidad de proporcionar una solución tecnológica que integre funcionalidades de seguimiento de hábitos y recordatorios de manera intuitiva y amigable.

El enfoque metodológico del desarrollo se ha centrado en la utilización de recursos de código abierto, buscando maximizar la transparencia, la flexibilidad y la reproducibilidad del proyecto. Para ello, se ha empleado Python como lenguaje de programación principal, en conjunto con la librería " python-telegram-bot " para la interacción con la plataforma de mensajería. La persistencia de los datos se gestiona mediante una base de datos PostgreSQL, desplegada a través de contenedores Docker, lo que asegura un entorno de ejecución consistente. Adicionalmente, se ha integrado la API de OpenWeather para ofrecer información climática relevante a los usuarios.

El desarrollo del bot se ha guiado por principios de modularidad y escalabilidad, procurando que cada componente sea robusto y bien documentado. Este informe detallará los requerimientos genéricos, los componentes libres analizados, las tecnologías utilizadas, y el proceso de instalación del software desarrollado, así como las conclusiones derivadas del proyecto. El principal objetivo es presentar un análisis exhaustivo del sistema, destacando la justificación de las elecciones tecnológicas y la forma en que el bot aborda las necesidades planteadas.

Cuerpo del trabajo:

1. Requerimientos Genéricos Considerados para el Desarrollo de la Aplicación

Para el desarrollo del bot de Telegram, denominado "Focus Helper Bot" con el usuario @HabitsPilotBot y creado a través de BotFather, se consideraron los siguientes requerimientos genéricos:

- Registro y Bienvenida del Usuario: Al iniciar el bot mediante el comando /start, el sistema debe registrar al nuevo usuario en la base de datos y solicitarle un nombre con el que desea ser referido.
- Gestión de Hábitos (/habits): El bot debe permitir a los usuarios registrar nuevos hábitos y tiene la capacidad de enviar recordatorios diarios sobre estos hábitos a una hora específica definida por el usuario.
- Gestión de Tareas (/task): La funcionalidad de tareas debe permitir a los usuarios registrar nuevas tareas, solicitando una fecha y hora específicas para su realización, y enviando recordatorios en el momento oportuno.
- Información Climática (/clima): El bot debe ser capaz de proporcionar información actualizada sobre el clima, basándose en la ciudad que el usuario solicite.
- Finalización del Servicio (/off): El bot debe permitir a los usuarios finalizar su interacción y eliminar sus datos de la base de datos mediante el comando /off.
- Persistencia de Datos: Toda la información de los usuarios, sus hábitos y tareas debe ser almacenada de forma persistente en una base de datos.
- Uso de Recursos Abiertos: El proyecto debe priorizar la utilización de herramientas y tecnologías de código abierto en todas sus funcionalidades principales.

2. Componentes Libres Analizados y Utilizados

El desarrollo del bot se ha fundamentado en el uso de diversos componentes de software de código abierto. A continuación, se detallan los principales:

2.1. python-telegram-bot

La librería python-telegram-bot ofrece una documentación exhaustiva que incluye una referencia técnica completa de clases, módulos, métodos y argumentos[1]. Además, cuenta con una wiki con introducciones elaboradas a sus diferentes funcionalidades y una sección de ejemplos que demuestran el uso de la API de Bot de Telegram y la propia librería.

Esta libreria está licenciada bajo LGPL-3.0-only. Esto permite copiar, distribuir y modificar el software, siempre que las modificaciones sean descritas y licenciadas gratuitamente bajo LGPL-3. Las obras derivadas (incluyendo modificaciones o cualquier cosa vinculada estáticamente a la librería) solo pueden ser redistribuidas bajo LGPL-3, pero las aplicaciones que usan la librería no tienen por qué serlo.

No se realizaron pruebas específicas sobre la librería python-telegram-bot en el marco de este proyecto. Se confía en la estabilidad y pruebas internas de la librería.

2.2. PostgreSQL

Implementado principalmente en C, PostgreSQL cuenta con una extensa y bien mantenida documentación oficial disponible públicamente, cubriendo desde la instalación y configuración hasta el uso avanzado y la administración.

PostgreSQL se distribuye bajo la PostgreSQL License, una licencia permisiva de código abierto similar a la licencia BSD.

No se realizaron pruebas específicas sobre el motor de base de datos PostgreSQL. Se confía en la robustez y estabilidad del sistema.

2.3. SQLAlchemy

Proporciona un conjunto de herramientas y funcionalidades para interactuar con bases de datos utilizando SQL. Permite construir consultas SQL de manera programática en Python, lo que te da un gran control sobre las sentencias SQL generadas.

SQLAlchemy posee una documentación muy completa, que abarca desde tutoriales para principiantes hasta guías detalladas para su uso avanzado como ORM y constructor de consultas SQL.

SQLAlchemy está licenciado bajo la licencia MIT.

2.4. psycopg2-binary

La documentación de psycopg2 y psycopg2-binary está disponible en su sitio web oficial, proporcionando guías sobre cómo conectar Python con bases de datos PostgreSQL y realizar operaciones.

Se distribuye bajo la GNU Lesser General Public License (LGPL) versión 3 o posterior.

2.5. requests

La librería requests es ampliamente conocida por su documentación clara y ejemplos sencillos que facilitan el manejo de solicitudes HTTP. Está licenciado bajo la licencia Apache 2.0.

2.6. python-dotenv

La documentación de python-dotenv es concisa y explica claramente cómo cargar variables de entorno desde un archivo .env.

Dicha libreria está licenciada bajo la licencia BSD 3-Clause "New" or "Revised".

2.7. hupper

Hupper es una herramienta para la recarga automática de procesos Python, con documentación que describe su configuración y uso para el desarrollo. Hupper está licenciado bajo la licencia MIT.

2.9. Asyncpg

Asyncpg para python ofrece documentación en su sitio web oficial. Asyncpg está licenciado bajo la licencia Apache 2.0.

2.10. APScheduler

La documentación de APScheduler está disponible en su sitio web oficial. APScheduler está licenciado bajo la licencia MIT.

2.11. dateparser

Ideal para trabajar conpython, la documentación de dateparser está disponible en su repositorio de GitHub. Además está licenciada bajo la licencia Apache 2.0.

2.12. wait-for-it.sh

La documentación de wait-for-it.sh está disponible en su repositorio de GitHub. wait-for-it.sh está licenciado bajo la licencia MIT.

3. Justificación de la Elección Realizada

La elección de cada una de las tecnologías mencionadas se basó en criterios de robustez, flexibilidad, soporte de la comunidad, madurez del proyecto, y la adhesión al principio de código abierto.

- **Python:** Seleccionado por su facilidad de aprendizaje, amplia comunidad de desarrollo, versatilidad y la disponibilidad de una vasta colección de librerías, lo que acelera el desarrollo de aplicaciones robustas.
- **python-telegram-bot (v20.7):** Se eligió esta librería por su reconocida robustez y su naturaleza de código abierto. Además, ofrece una interfaz asíncrona pura para la API de Bot de Telegram y cuenta con clases de alto nivel en el submódulo *telegram.ext* que simplifican enormemente el desarrollo de bots. Su soporte nativo para todas las tipos y métodos de la Bot API 9.0 y su integración fluida con webhooks y polling son ventajas significativas. La inclusión de job-queue es crucial para la funcionalidad de recordatorios programados, ya que instala APScheduler, permitiendo la ejecución de tareas en momentos específicos para los recordatorios de hábitos y tareas.
- **PostgreSQL:** La elección de PostgreSQL se fundamentó principalmente en su naturaleza de código abierto y su reputación como un potente y confiable sistema de gestión de bases de datos relacionales. Adicionalmente, fue el motor de base de datos utilizado y estudiado en la asignatura de Bases de Datos 2025 de la carrera, lo que facilitó su integración en el proyecto.
- **Docker:** Aunque la justificación inicial fue para un futuro despliegue en la nube facil y rapido, la decisión de utilizar Docker para la contenerización del bot y la base de datos ofrece beneficios inherentes en términos de consistencia del entorno, aislamiento de dependencias y facilidad de despliegue, sentando una base sólida para una posible migración a entornos de nube en el futuro.
- **OpenWeather API:** Se seleccionó esta API debido a su disponibilidad de una capa de uso gratuita, lo cual es coherente con la filosofía de un proyecto de código abierto y sin fines de lucro, permitiendo acceder a datos meteorológicos sin incurrir en costos.

4. Tecnologías Utilizadas

Las principales tecnologías utilizadas en el desarrollo del bot "Focus Helper Bot" incluyen:

- **Lenguaje de Programación:** Python (versión 3.9+).
- **Framework/Librería de Bot:** python-telegram-bot (versión 20.7.0).
- **Sistema de Gestión de Base de Datos:** PostgreSQL.
- **ORM/Toolkit de Base de Datos:** SQLAlchemy.
- **Contenerización:** Docker.
- **Librerías Adicionales de Python:** python-dotenv, requests, hupper, psycopg2-binary, pytz.
- **API Externa:** OpenWeather API.

5. Dependencias y Versiones

Las dependencias del proyecto y sus versiones específicas, tal como se definen en el archivo requirements.txt, son las siguientes:

- python-telegram-bot[job-queue]==20.7.0
- SQLAlchemy
- python-dotenv
- requests
- hupper
- psycopg2-binary
- pytz

Link del sistema de control de versiones: https://github.com/Andres25ar/ProductivityHabitsTlgBot

6. Instalación del Software Desarrollado

Para la instalación y puesta en marcha del bot "Focus Helper Bot", se siguen los siguientes pasos, asumiendo que Docker y Docker Compose están previamente instalados en el sistema:

1. **Clonar el Repositorio:**

Bash

git clone https://github.com/Andres25ar/ProductivityHabitsTlgBot

cd [nombre_del_directorio_donde_se_guardará_el_bot]

2. **Configuración de Variables de Entorno:**

Crear un archivo .env en el directorio raíz del proyecto con las siguientes variables:

TELEGRAM_BOT_TOKEN=”Token del bot de telegram proporcionado por Bot_Father”

OPENWEATHER_API_KEY=”API de openweather para actualizaciones del clima”

POSTGRES_DB=nombre_de_db

POSTGRES_USER=usuario_de_db

POSTGRES_PASSWORD=contraseña_de_db

DB_HOST=db

DB_PORT=5432

3. **Construir y Levantar los Contenedores Docker:**

Desde el directorio raíz del proyecto, ejecutar:

Bash

docker-compose up --build -d

Este comando construirá las imágenes de Docker para el bot y la base de datos PostgreSQL, y luego levantará ambos servicios en segundo plano. La base de datos se inicializará automáticamente con los parámetros definidos en el archivo .env.

4. **Verificar el Estado de los Contenedores:**

Bash

docker-compose ps

Se espera ver los contenedores bot y db en estado Up.

5. **Utilizar el bot en Telegram**

En el buscador de la aplicacion se debe buscar @HabitsPilotBot, seleccionar la primera aparicion y una vez dentro del chat, precionar el commando /start el cual dará la bienvenida al Usuario y le solicitará un nombre con el cual será almacenado en la base de datos.

Una vez recibido el mensaje de bienvenida, el Usuario tendrá al costado del teclado de la app un menú desplegable con los demás comandos que puede utilizar. Estos comandos son:

- **/habits:** brinda al usuario una serie de hábitos cargados por defecto con un id numerico, con los cuales se solicita al usuario enviar dichos id como respuesta para seleccionar los habitos que se desea mejorar o poner en practica
- **/task:** permite al usuario generar una tarea ingresando una descripcion de la tarea a realizer y seleccionando una fecha y hora en la que dicha tarea será realizada
- **/clima:** solicita al usuario que escriba la ciudad de la cual quiere conocer el clima actual y se le devuelve informacion de dicha ciudad
- **/off:** se borra al usuario de la base de datos y se muestra un mensaje de despedida, dejando ademas saber al usuario que puede volver a utilizar el bot cuando lo desee con /start

7. Establecer Licencia al Software Desarrollado

El software desarrollado para el "Focus Helper Bot" se distribuirá bajo la **GNU General Public License v3.0 (GPLv3)**. Esta licencia se incluirá en un archivo LICENSE en el directorio raíz del repositorio del proyecto. Bajo los términos de la GPLv3, los usuarios tienen la libertad de ejecutar, estudiar, compartir y modificar el software, siempre que las obras derivadas también se distribuyan bajo la misma licencia.

Conclusiones:

Como resultado del desarrollo del "Focus Helper Bot", hemos logrado un sistema funcional que cumple con los requerimientos esenciales para la gestión de hábitos y tareas de los usuarios, así como la consulta del clima.

No obstante, identificamos que la configuración inicial de la zona horaria para el usuario es un punto débil, haciéndola poco intuitiva y complicada de entender, lo cual representa un aspecto crítico a mejorar para optimizar la experiencia del usuario.

Las principales causas de las limitaciones y las ausencias de prácticas fluidas se atribuyen al poco tiempo y a nuestra inexperiencia inicial con el lenguaje Python, lo que ralentizó el proceso de toma de decisiones y la implementación.

Un problema inesperado significativo fue la dificultad inicial para lograr que el bot enviara notificaciones en los horarios programados por defecto o seleccionador por el usuario, lo cual requirió investigación adicional y ajustes.

De cara al futuro, aspiramos a resolver estas debilidades, específicamente mejorando la facilidad de configuración. Además, buscamos llevar el bot a un entorno de nube para asegurar su disponibilidad continua y fomentar su adopción por una cantidad usuarios más amplia en la comunidad de telegram, consolidando así el impacto positivo del "Focus Helper Bot" en el bienestar de las personas que desean utilizarlo para mejorar su enfoque y proactividad.

Bibliografía:

- Documentación de *Python-Telegram-Bot: python-telegram-bot (s.f.). Documentation*. En python-telegram-bot. Recuperado de https://python-telegram-bot.org/ el 24 de junio de 2025.

- Documentación de *APScheduler: APScheduler (s.f.). APScheduler Documentation*. En APScheduler. Recuperado de https://apscheduler.readthedocs.io/en/master/ el 24 de junio de 2025.

- Documentación de *SQLAlchemy: SQLAlchemy (s.f.). SQLAlchemy Documentation*. En SQLAlchemy. Recuperado de https://www.sqlalchemy.org/ el 24 de junio de 2025.

- Documentación de *python-dateutil: python-dateutil (s.f.). Dateutil Documentation*. En dateutil. Recuperado de https://dateutil.readthedocs.io/en/stable/ el 24 de junio de 2025.

- Documentación de *Requests: Requests (s.f.). Requests: HTTP for Humans*. En Requests. Recuperado de https://requests.readthedocs.io/en/latest/ el 24 de junio de 2025.

- Documentación de *python-dotenv: pypi.org (s.f.). python-dotenv*. En PyPI. Recuperado de https://pypi.org/project/python-dotenv/ el 24 de junio de 2025.

- Documentacion de *PostgreSQL: The PostgreSQL Global Development Group*. En PostgreSQL. Recuperado de https://www.postgresql.org/docs/ el 24 de junio de 2025

- Lista de *Zonas Horarias IANA: Wikipedia (s.f.). List of tz database time zones*. En Wikipedia. Recuperado de https://en.wikipedia.org/wiki/List_of_tz_database_time_zones el 24 de junio de 2025.