

CDN: Telegram chatbot

Pablo Carrera Flórez de Quiñones
José Llanes Jurado

2019-05-30

Contenidos

1	Introducción	2
2	Entorno de desarrollo	2
3	Desarrollo	4
3.1	Registro de usuarios	4
3.1.1	Implementación	5
3.2	Creación de tareas	5
3.2.1	Implementación	5
4	Comentarios adicionales	6
5	Recursos	7
6	Código	8



1 Introducción

Como parte de la asignatura de Analítica Web del Máster en Ciencia de Datos que cursamos desarrollamos el proyecto *Helper*, una plataforma virtual que busca poner en contacto a la gente que necesita ayuda con aquellos que se ofrezcan a ayudar. Nuestra propuesta es que la aplicación conecte a los usuarios a través de la oferta y demanda de favores, y sean ellos quienes se pongan de acuerdo en el método de pago, ya sea mediante una cantidad de dinero u otro favor.

La forma de generar una oferta debe ser sencilla, no puede costar más trabajo colgar el anuncio y recibir una respuesta que el que podría costarle a un usuario realizar una tarea con sus propios medios. Para ello se desarrollará un interfaz intuitiva tanto para el registro de usuarios como para la publicación de ofertas. Nuestra idea es evitar los tediosos formularios de registro que deben rellenarse por lo general en cualquier aplicación, de forma que proponemos realizar todas estas gestiones mediante un chatbot en Telegram. Esto esperamos que genere una interfaz más amigable con el usuario, sobretodo con aquellos que aún no se han adaptado totalmente a las nuevas tecnologías. En este trabajo vamos a exponer pues, las diferentes funcionalidades de este chatbot.

2 Entorno de desarrollo

Para desarrollar este chatbot vamos a hacer uso de la plataforma Telegram. Esta plataforma además de tener mayor seguridad y privacidad para los usuarios, proporciona muchas más facilidades para el desarrollo de aplicaciones que el resto de aplicaciones normales de mensajería, lo cual hace que sea un candidato ganador para nuestro proyecto. Además, al ser una plataforma de código abierto, existe una comunidad muy numerosa de usuarios y desarrolladores alrededor de ella, lo que hace muy fácil el acceso a ejemplos de bots y la resolución de problemas con los mismos.

El desarrollo de chatbots en esta plataforma es bastante sencillo, basta con empezar una conversación con *@BotFather*, que como su nombre indica es el padre de todos los bots, y seguir las instrucciones que este nos marca. La interfaz de *@BotFather* va a inspirar mucho el desarrollo de nuestro bot, ya que es justo el tipo de sencillez que queremos para nuestro usuario: menús simples, interacción directa, presencia nula de formularios...

Los pasos a seguir son:

- Buscar a *@BotFather* en la lista de usuarios de Telegram e iniciar una conversación con él.
- Elegir un nombre para el bot: vamos a optar por la sencillez y vamos a llamar a nuestro bot *Helper*, igual que la aplicación que vamos a desarrollar. Esto también aporta un toque de humor sutil, ya que la función del bot también es ayudar.
- Elegir un nombre de usuario para el bot: dado que es una versión de prueba y está aun muy lejos de ser el producto final hemos elegido llamarlo *@Helper_28052019_Bot*, la fecha en la que ha sido desarrollado, y ya buscaremos un nombre más adecuado conforme vayamos avanzando con el proyecto.
- Añadir una descripción del funcionamiento del bot, esto puede facilitar la comprensión de su funcionamiento por parte de los usuarios.
- Añadir información sobre los creadores del bot. Por temas de seguridad creemos que es importante hacer notar la propiedad intelectual del bot de momento.

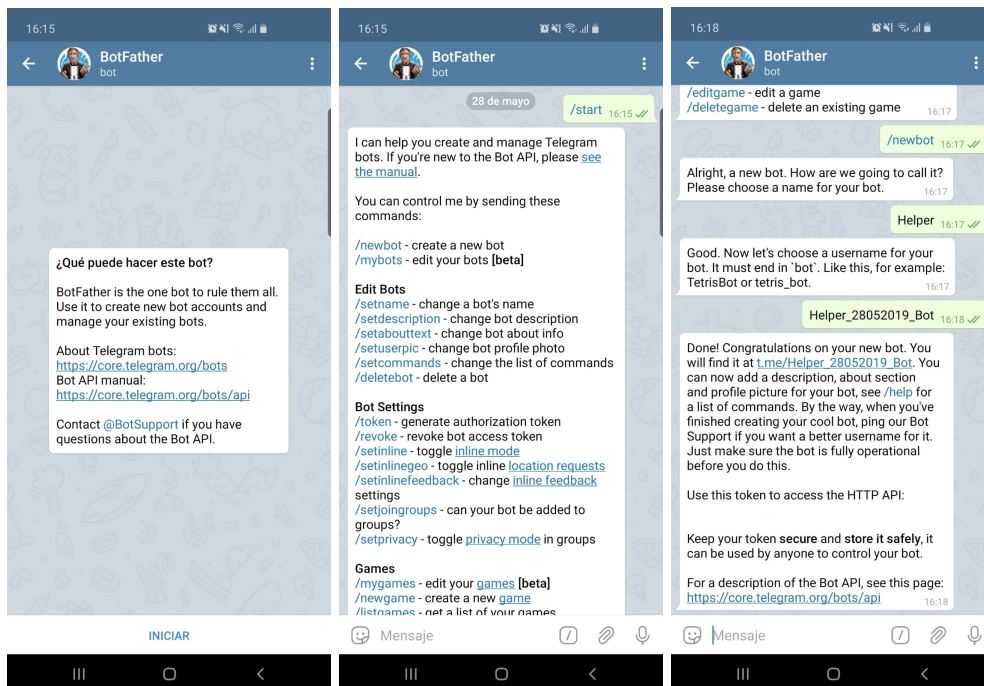


Figura 1: Capturas de pantalla ilustrando el proceso de creación de un bot en Telegram. Como se puede ver el proceso es rápido y sencillo, uno de los principales motivos por los que se ha elegido esta plataforma.

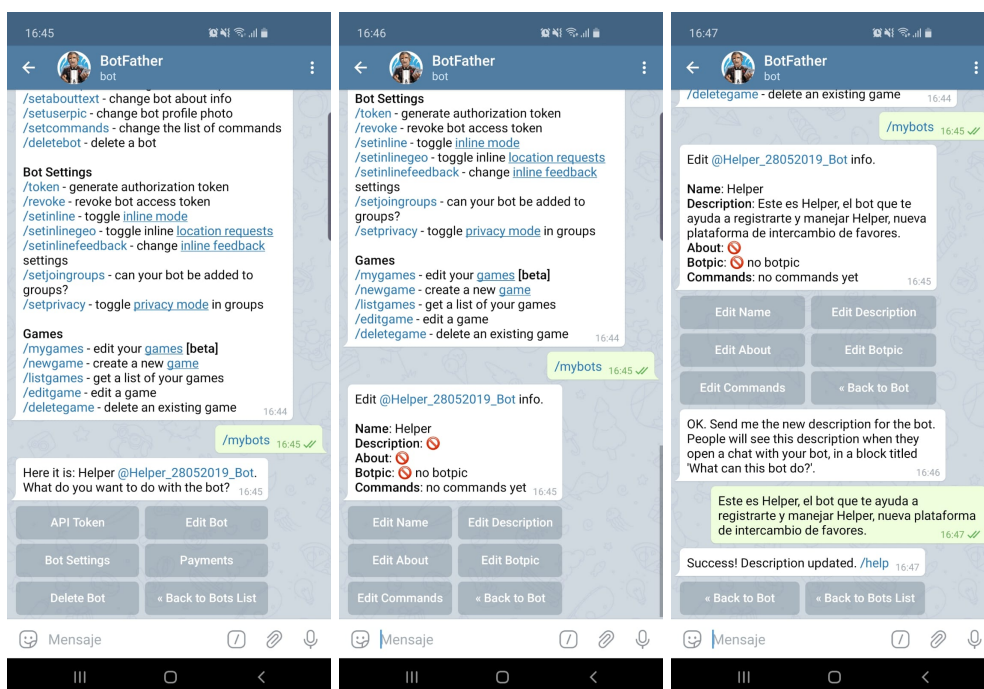


Figura 2: Capturas de pantalla ilustrando el proceso de creación de un bot en Telegram. La personalización del bot es un aspecto importante, una descripción clara puede ayudar a los usuarios.

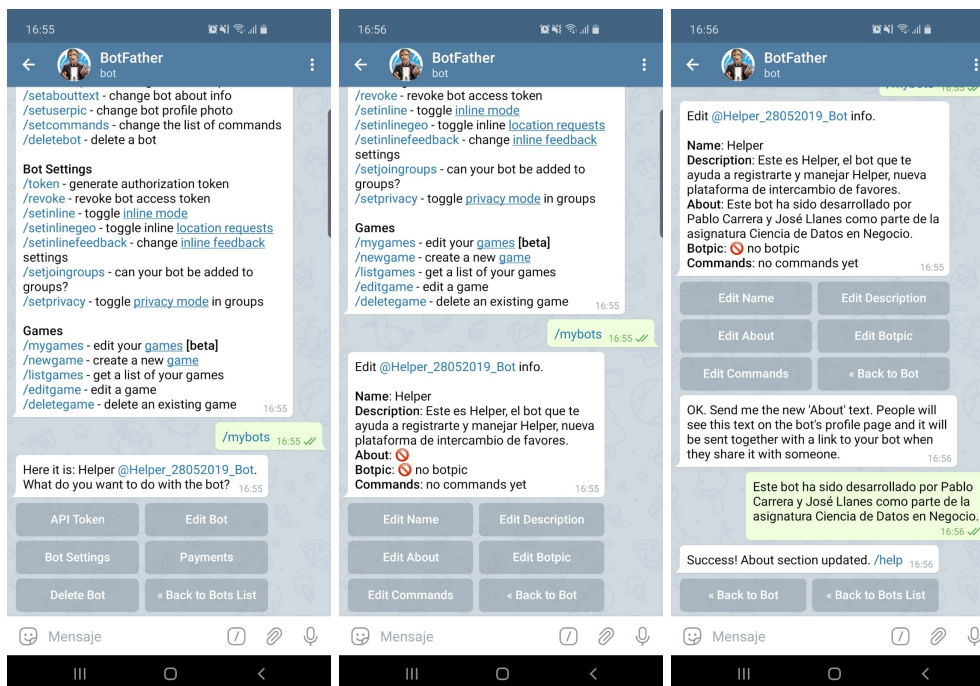


Figura 3: Capturas de pantalla ilustrando el proceso de creación de un bot en Telegram. La propiedad intelectual en las primeras etapas de desarrollo también es un aspecto importante.

3 Desarrollo

Una vez creado el bot podemos proceder a incluir en este las funcionalidades necesarias para convertirlo en un chatbot propiamente dicho. Para ello haremos uso del lenguaje de programación Python, también de código abierto y uso intuitivo. En concreto haremos uso de la librería python-telegram-bot, que incluye una gran cantidad de funciones implementadas para desarrollar un chatbot de manera sencilla.

3.1 Registro de usuarios

El primer paso sería el registro en la aplicación, para ello bastaría con introducir el correo electrónico o el teléfono móvil, esto generaría un identificador único para cada usuario que se mantendría privado y solo se usaría para gestiones internas de la base de datos de la aplicación. Una vez hecho esto se introducirían los datos públicos de cada usuario:

- Nombre y apellidos: no olvidemos que los usuarios de esta aplicación van a ser personas, de forma que parece adecuado tratar al resto de la comunidad como personas que son, por su nombre y no como completos desconocidos.
- Foto de perfil: realmente la mayoría de la gente con la que interacciones serán vecinos más o menos cercanos, de forma que una foto puede ayudarte a reconocer a esas personas y facilitar así un lazo de confianza rápido con el resto de usuarios.
- Ocupación y aficiones: la idea de la aplicación no es cumplir una tarea puntual, sino formar una comunidad de personas, por ello consideramos relevante introducir cierta información

personal que ayude a los usuarios a conocerse entre sí y facilitar la identificación de usuarios afines con los que poder contactar directamente dado el caso.

3.1.1 Implementación

Hemos incluido todo ese proceso dentro del código del chatbot para que simule una conversación natural entre dos personas que acaban de conocerse. Esto hace que el proceso no se haga pesado y que el usuario adquiriera algo de confianza con nosotros al ser tratado de una manera más natural.

3.2 Creación de tareas

A continuación vamos a detallar la creación de tareas para demandar un favor. El procedimiento de publicación de una oferta también será igual de sencillo que el de registro, se rellenará un pequeño formulario interactivo y se publicará automáticamente en la base de datos de tareas. Los campos que consideramos necesarios para una tarea serán los siguientes:

- Título: un pequeño resumen de lo que se necesita en menos de 50 caracteres. El tipo de títulos que se esperan son sencillos y claros, como por ejemplo "¿Alguien puede hacer la compra por mi?", "¡Necesito ayuda para cambiar la rueda del coche!" o "¿Alguien puede ayudarme con la mudanza?". Incluir algún ejemplo en el formulario puede ayudar a los usuarios a rellenar este campo de forma efectiva.
- Descripción: aquí se espera una descripción mas larga de la tarea, incluyendo los detalles de que favor necesitas y de por qué no puedes realizarlo tu mismo, esto ayudará a los posibles *helpers* a entender mejor tus necesidades y a ti a seleccionar quién puede ser más adecuado para ayudarte dado el caso.
- Prioridad: es importante también dejar notar si una tarea es urgente, como por ejemplo cambiar una rueda, o si puede hacerse a lo largo del día, como pasear al perro o hacer la compra, o incluso si es de sentido más amplio, como por ejemplo dar clases particulares o elaborar un plan nutricional.
- Retorno preferido: igual que los usuarios tiene la posibilidad de elegir de que forma prefieren el retorno de sus favores, los usuarios que propongan ofertas también tienen la libertad de que ofrecer como quieren ser correspondidos en cada oferta. La idea es que esto sea negociable y una vez exista un contacto y un intercambio de mensajes puede decidirse esta de forma conveniente y sin presiones.

3.2.1 Implementación

Estos procesos también los hemos incluido dentro del código del chatbot para que simulen una conversación natural entre dos personas. Algunas funcionalidades aún no están disponibles a faltar de una base de datos que las respalde, pero la interfaz de cara al usuario si que estaría completa.

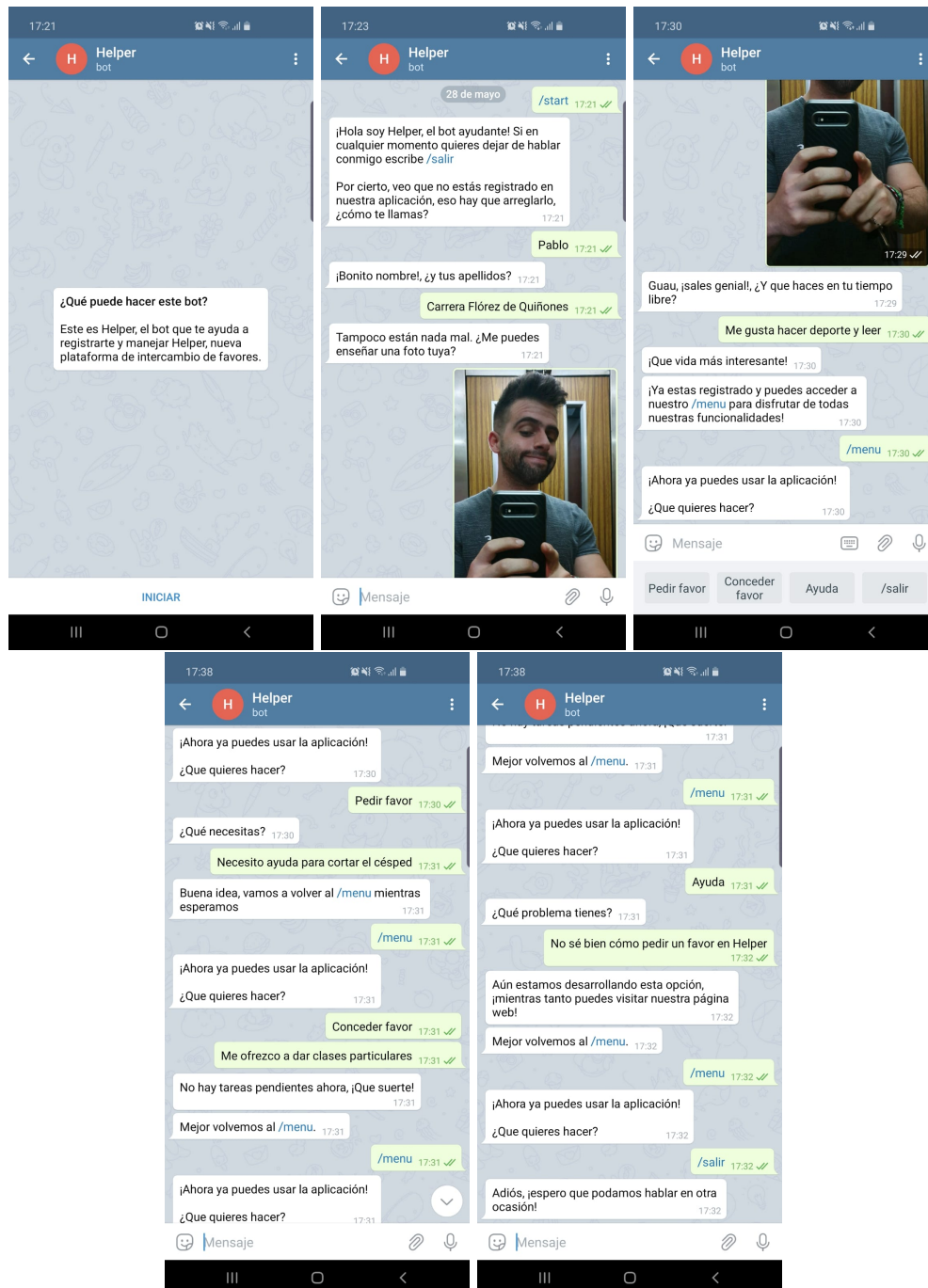


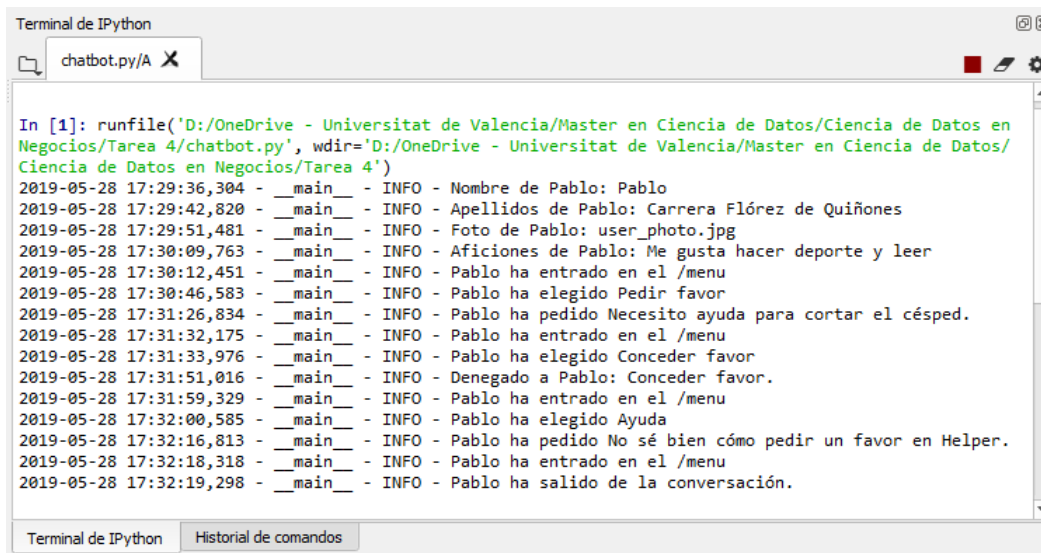
Figura 4: Capturas de pantalla ilustrando el proceso de registro y la navegación por los menús.

4 Comentarios adicionales

En este proyecto hemos desarrollado un chatbot en Telegram, *@Helper_28052019_Bot*, usando Python para dar soporte al registro y gestión de una aplicación que estamos desarrollando, Helper. Hemos incluido todas las funcionalidades de este para que simulen una conversación natural entre dos personas, de manera que el proceso se hace ameno y sencillo. Hemos incluido imágenes tanto de su creación, como de su uso por parte de uno de los desarrolladores para demostrar su

uso.

Finalmente hemos incluido también un log que nos permite ver en tiempo real todas las interacciones que se van teniendo con el chatbot. Esto nos a permitido solucionar errores de forma rápida y nos deja la puerta abierta para la introducción de dichos datos en una base de datos a medida que un mayor número de usuarios haga uso del bot.



```
Terminal de IPython
chatbot.py/A X

In [1]: runfile('D:/OneDrive - Universitat de Valencia/Master en Ciencia de Datos/Ciencia de Datos en
Negocios/Tarea 4/chatbot.py', wdir='D:/OneDrive - Universitat de Valencia/Master en Ciencia de Datos/
Ciencia de Datos en Negocios/Tarea 4')
2019-05-28 17:29:36,304 - __main__ - INFO - Nombre de Pablo: Pablo
2019-05-28 17:29:42,820 - __main__ - INFO - Apellidos de Pablo: Carrera Flórez de Quiñones
2019-05-28 17:29:51,481 - __main__ - INFO - Foto de Pablo: user_photo.jpg
2019-05-28 17:30:09,763 - __main__ - INFO - Aficiones de Pablo: Me gusta hacer deporte y leer
2019-05-28 17:30:12,451 - __main__ - INFO - Pablo ha entrado en el /menu
2019-05-28 17:30:46,583 - __main__ - INFO - Pablo ha elegido Pedir favor
2019-05-28 17:31:26,834 - __main__ - INFO - Pablo ha pedido Necesito ayuda para cortar el césped.
2019-05-28 17:31:32,175 - __main__ - INFO - Pablo ha entrado en el /menu
2019-05-28 17:31:33,976 - __main__ - INFO - Pablo ha elegido Conceder favor
2019-05-28 17:31:51,016 - __main__ - INFO - Denegado a Pablo: Conceder favor.
2019-05-28 17:31:59,329 - __main__ - INFO - Pablo ha entrado en el /menu
2019-05-28 17:32:00,585 - __main__ - INFO - Pablo ha elegido Ayuda
2019-05-28 17:32:16,813 - __main__ - INFO - Pablo ha pedido No sé bien cómo pedir un favor en Helper.
2019-05-28 17:32:18,318 - __main__ - INFO - Pablo ha entrado en el /menu
2019-05-28 17:32:19,298 - __main__ - INFO - Pablo ha salido de la conversación.
```

Figura 5: Ejemplo del log provocado por la interacción de uno de nuestros desarrolladores con el chatbot.

5 Recursos

Para la realización de este trabajo se ha hecho un gran uso de la documentación proporcionada por las diferentes herramientas que hemos utilizado:

- Telegram:
 - Telegram: <https://core.telegram.org/bots>
 - Bots API: <https://core.telegram.org/bots/api>
 - Samples: <https://core.telegram.org/bots/samples>
 - Ejemplos: <https://github.com/python-telegram-bot/python-telegram-bot>
- Python: <https://docs.python.org/3/>
- Python libraries:
 - python-telegram-bot: <https://python-telegram-bot.readthedocs.io/en/stable/>

6 Código

```
#####
# Librerías #
#####

# Telegram
from telegram import ReplyKeyboardMarkup, ReplyKeyboardRemove
from telegram.ext import Updater, Filters, CallbackQueryHandler
from telegram.ext import CommandHandler, MessageHandler, ConversationHandler, RegexHandler

# Others
import logging
import config

#####
# Preparación #
#####

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

logger = logging.getLogger(__name__)

#####
# Sección de registro #
#####

NOMBRE, APELLIDOS, FOTO, AFICIONES = range(4)

def start(bot, update):
    update.message.reply_text(
        "¡Hola soy Helper, el bot ayudante! "
        "Si en cualquier momento quieres dejar de hablar conmigo escribe /salir \n \n"
        "Por cierto, veo que no estás registrado en nuestra aplicación, "
        "eso hay que arreglarlo, ¿cómo te llamas?"
    )
    return NOMBRE

def nombre(bot, update):
    user = update.message.from_user
    logger.info("Nombre de %s: %s", user.first_name, update.message.text)
    update.message.reply_text("¡Bonito nombre!, ¿y tus apellidos?")
    return APELLIDOS

def apellidos(bot, update):
    user = update.message.from_user
```



```

logger.info("Apellidos de %s: %s", user.first_name, update.message.text)
update.message.reply_text("Tampoco están nada mal. ¿Me puedes enseñar una foto tuya?")
return FOTO

def foto(bot, update):
    user = update.message.from_user
    photo_file = bot.get_file(update.message.photo[-1].file_id)
    photo_file.download('user_photo.jpg')
    logger.info("Foto de %s: %s", user.first_name, 'user_photo.jpg')
    update.message.reply_text("Guau, ¡sales genial!, ¿Y que haces en tu tiempo libre?")
    return AFICIONES

def aficiones(bot, update):
    user = update.message.from_user
    logger.info("Aficiones de %s: %s", user.first_name, update.message.text)
    update.message.reply_text("¡Que vida más interesante!")
    update.message.reply_text("!Ya estas registrado y puedes acceder a nuestro /menu "
                                "para disfrutar de todas nuestras funcionalidades!")
    return MENU

#####
# Sección de menú #
#####

MENU, ELEGIR, PEDIR, CONCEDER, AYUDA, SALIR = range(4,10)

def menu(bot, update):
    user = update.message.from_user
    reply_keyboard = ["Pedir favor", "Conceder favor", "Ayuda", "/salir"]
    update.message.reply_text(
        "¡Ahora ya puedes usar la aplicación! \n \n"
        "¿Que quieres hacer?",
        reply_markup = ReplyKeyboardMarkup(reply_keyboard,
                                            one_time_keyboard = True,
                                            resize_keyboard = True)
    )
    logger.info("%s ha entrado en el %s", user.first_name, update.message.text)

    return ELEGIR

def elegir(bot, update):
    user = update.message.from_user
    logger.info("%s ha elegido %s", user.first_name, update.message.text)
    if update.message.text == "Pedir favor" :
        update.message.reply_text("¿Qué necesitas?")
        return PEDIR
    if update.message.text == "Conceder favor" :
        return CONCEDER
    if update.message.text == "Ayuda" :
        update.message.reply_text("¿Qué problema tienes?")
        return AYUDA
    if update.message.text == "Salir" :

```

```

        return SALIR

def pedir(bot, update):
    user = update.message.from_user
    logger.info("%s ha pedido $s.", user.first_name, update.message.text)
    update.message.reply_text("Buena idea, vamos a volver al /menu mientras esperamos")
    return MENU

def conceder(bot, update):
    user = update.message.from_user
    logger.info("User %s try $s.", user.first_name, "Conceder favor")
    update.message.reply_text("No hay tareas pendientes ahora, ¡Que suerte!")
    update.message.reply_text("Mejor volvemos al /menu.")
    return MENU

def ayuda(bot, update):
    user = update.message.from_user
    logger.info("%s ha pedido $s.", user.first_name, update.message.text)
    update.message.reply_text("Aún estamos desarrollando esta opción, ¡mientras "
                              "tanto puedes visitar nuestra página web!")
    update.message.reply_text("Mejor volvemos al /menu.")
    return MENU

#####
# Sección de otros #
#####

def salir(bot, update):
    user = update.message.from_user
    logger.info("%s ha salido de la conversación.", user.first_name)
    update.message.reply_text("Adiós, ¡espero que podamos hablar en otra ocasión!",
                              reply_markup = ReplyKeyboardRemove()
                              )
    return ConversationHandler.END

def error(bot, update, error):
    logger.warning('Update "%s" caused error "%s"', update, error)

#####
# Implementación #
#####

def main():
    # Creamos las instancias necesarias
    updater = Updater(token = config.TOKEN)
    dispatcher = updater.dispatcher

```

```

# Sección de conversación para realizar el registro
conv_handler = ConversationHandler(
    entry_points = [CommandHandler('start', start)],

    states = {
        # Sección de conversación para realizar el registro
        NOMBRE: [MessageHandler(Filters.text, nombre)],
        APELLIDOS: [MessageHandler(Filters.text, apellidos)],
        FOTO: [MessageHandler(Filters.photo, foto)],
        AFICIONES: [MessageHandler(Filters.text, aficiones)],

        # Sección de conversación para realizar el registro
        MENU: [CommandHandler("menu", menu)],
        ELEGIR: [RegexHandler("(Pedir favor|Conceder favor|Ayuda|Salir)", elegir)],
        PEDIR: [MessageHandler(Filters.text, pedir)],
        CONCEDER: [MessageHandler(Filters.text, conceder)],
        AYUDA: [MessageHandler(Filters.text, ayuda)]
    },

    fallbacks=[CommandHandler('salir', salir)]
)

dispatcher.add_handler(conv_handler)

# Muestra los errores
dispatcher.add_error_handler(error)

# Iniciar el bot
updater.start_polling()

# Mantener el bot hasta pulsar Ctrl-C
updater.idle()

if __name__ == '__main__':
    main()

```