



Universidad de Guayaquil

Procesos de Software

FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS

TELEGRAM BOT

KAMUI

SOF-S-MA-3-1

Integrantes:

Bryan Vera Lenis

Joaquin Matamoros Jalca

Fernando Marcillo Falcone

Ciclo I 2020-2021

Índice general

| | |
|--|----|
| 0.1. Requerimientos del proyecto | 2 |
| 0.2. Diseño Rápido Y Modelado | 3 |
| 0.3. Construcción Del Prototipo | 3 |
| 0.3.1. Creación del Token con BotFather | 3 |
| 0.3.2. Uso de Token y comienzo de codificación | 4 |
| 0.3.3. Mejora del prototipo | 5 |
| 0.4. Diseño de Prototipo Final | 5 |
| 0.5. Desarrollo, Entrega Y Retroalimentación | 8 |
| 0.6. Estimación de costos | 11 |
| 0.7. Enlaces | 12 |

0.1. Requerimientos del proyecto

Bot Agencia de Viajes Telegram

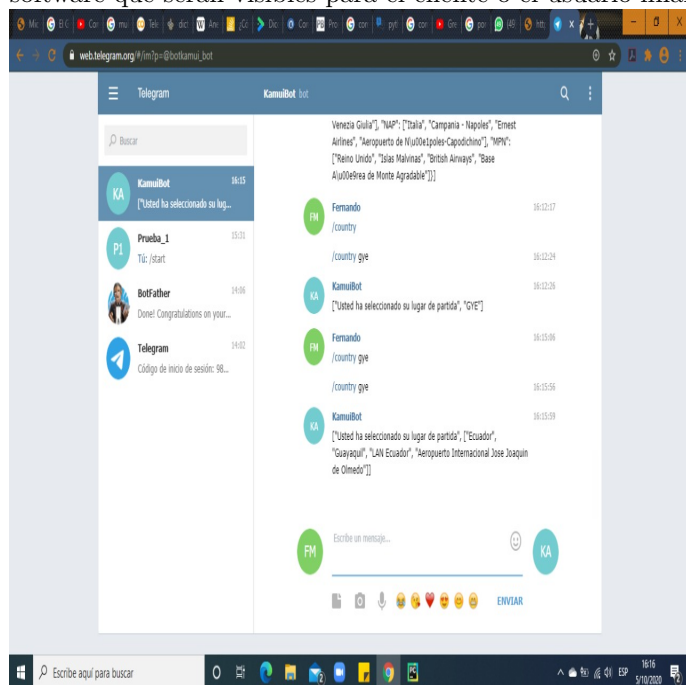
Este proyecto tiene como objetivo la creación de un Bot para una Agencia de Viajes mediante la aplicación Telegram.

La aplicación debe permitir a los Usuarios de la Agencia de Viajes (no clientes) finales realizar lo siguiente:

- **LENGUAJE DE PROGRAMACION:** Python.
- **LIST:** Listar todos los vuelos disponibles.
- **SEARCHD o SEARCHO:** Buscar destino u origen; esto puede ser realizado usando código IATA, Nombre de Aeropuerto, Ciudad o País. Está opción deberá mostrar todos los vuelos disponibles a ese destino o desde ese origen.
- **BUY_TICKET:** Reservar vuelos de solo ida. Considerar todos los datos necesarios para este proceso (ORIGEN-DESTINO, NUMERO DE ASIENTOS)
- **BUYRT_TICKET:** Reservar vuelos de ida-vuelta. Considerar todos los datos necesarios para este proceso (ORIGEN-DESTINO, NUMERO DE ASIENTOS)
- **TIEMPO:** 2 Semanas.

0.2. Diseño Rápido Y Modelado

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final.



Libería (Telegram Bot):

Para Python también existen varias librerías con las que desarrollar bots de Telegram, esta librería nos permite hacer esto simplemente llamando a un procedimiento que la otra persona ya ha hecho y probado, de modo que sólo nos tendremos que enfocar en hacer lo que nos interesa en lugar de tener que hacer también toda la parte de interactuar con los servidores de Telegram.

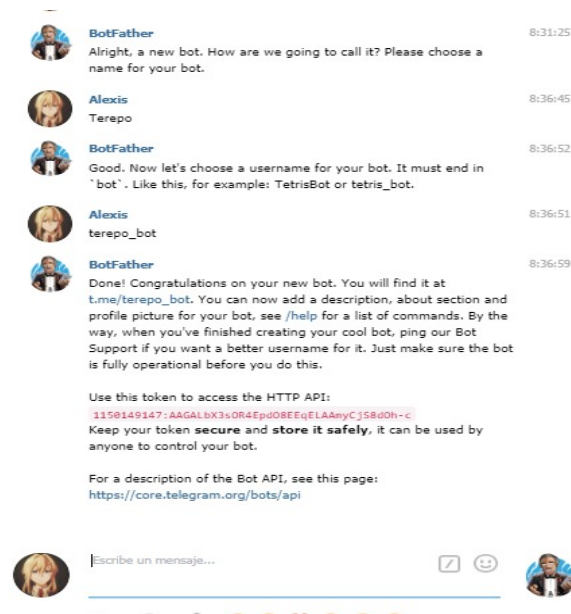
Diccionarios:

Un Diccionario es una estructura de datos y un tipo de dato en Python con características especiales que nos permite almacenar cualquier tipo de valor como enteros, cadenas, listas e incluso otras funciones. Estos diccionarios nos permiten además identificar cada elemento por una clave (Key).

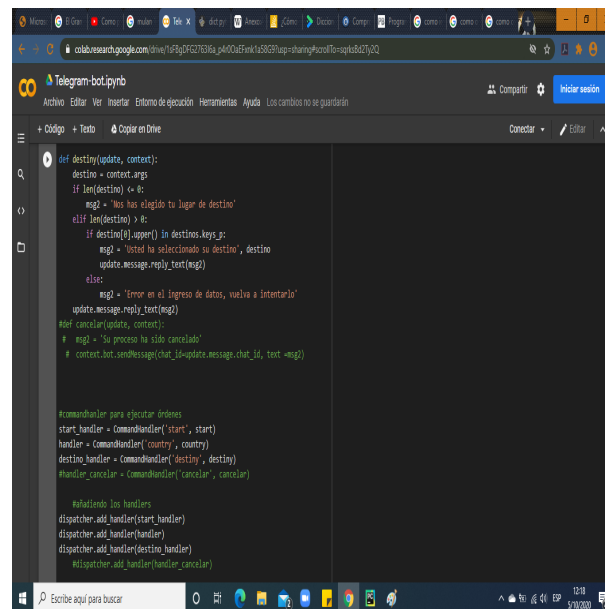
0.3. Construcción Del Prototipo

0.3.1. Creación del Token con BotFather

Antes de empezar a codificar, necesitamos crear un bot mediante el BotFather que nos ofrece Telegram; así, podremos desde Python, gestionar todas funciones del bot.



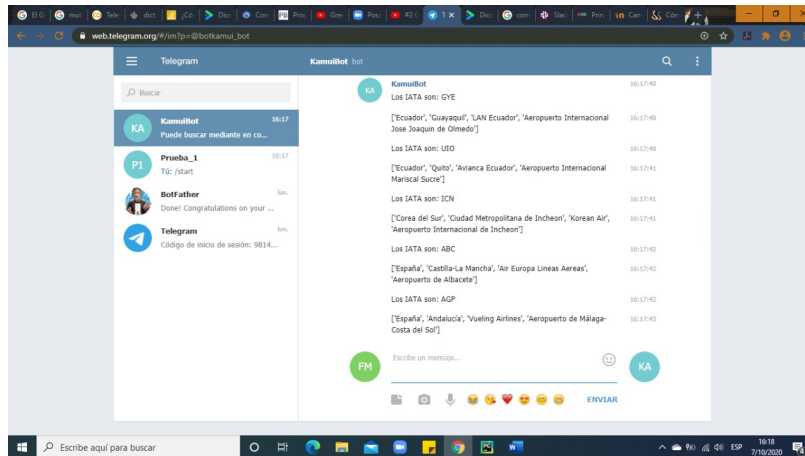
0.3.2. Uso de Token y comienzo de codificación



En la imagen se puede apreciar el uso de handlers para que el bot responda a

los comandos que el usuario haga uso. Este fue el primer prototipo del uso de comandos junto con argumentos.

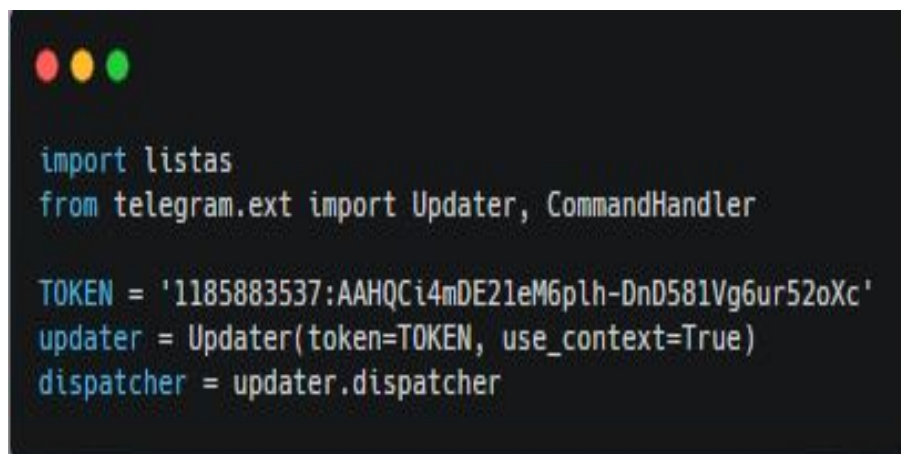
0.3.3. Mejora del prototipo



En esta etapa todos los cambios identificados en el diseño técnico son implementadas y probadas para asegurar su viabilidad, corrección y completitud con respecto a los requerimientos.

Se hizo de diccionarios anidados, para una mayor facilidad de organización y acceso a la información requerida.

0.4. Diseño de Prototipo Final



Estructura inicial del proyecto, donde se incluyen las librerías a usar y se establece el Token del bot, además de un updater y dispatcher para los comandos.

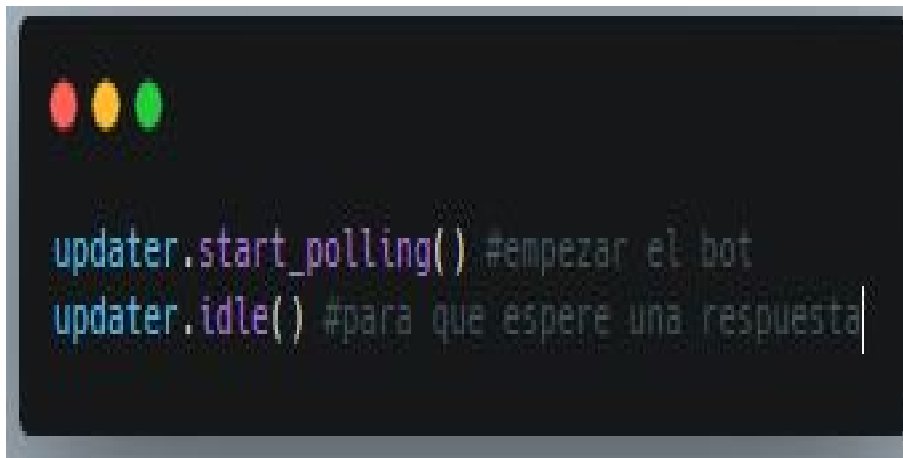
```
def start(update, context):
    first_name = update.message.from_user.first_name
    message = 'Hola {}, ingrese un comando para iniciar el trámite ' \
              'o ingrese /list para mostrar todos los codigos IATA disponibles'.format(first_name)
    message2= 'ingrese /origen + IATA Ciudad para seleccionar pais de origen ' \
              'ingrese /destiny + IATA Ciudad para seleccionar pais de destino' \
              'ingrese /registro_nombre + Nombre para registrar nombre de Comprador ' \
              'ingrese /registro_pasajeros + Cantidad de boletos' \
              'ingrese /registro_fecha + dd/mm/aa - dd/mm/aa para saber fecha de viaje' \
              'ingrese /total_a_pagar para saber Valor total a cancelar' \
              'ingrese /reserve Imprimir Ticket' \
              'ingrese /exit para salir'
    update.message.reply_text(message)
    update.message.reply_text(message2)
```

Definición del comando /start que le da la bienvenida al usuario y envía al usuario información necesaria para el correcto uso del bot.

```
#commandhanler para ejecutar órdenes
start_handler = CommandHandler('start', start)
handler = CommandHandler('origen', origen)
destino_handler = CommandHandler('destiny', destiny)
help_handler = CommandHandler("list", list)
handler_exit = CommandHandler('exit', exit)
reserve_handler = CommandHandler('reserve', reserve)
nombre_handler = CommandHandler('registro_nombre', registro_nombre)
pasajeros_handler = CommandHandler('registro_pasajeros', registro_pasajeros)
fecha_handler = CommandHandler('registro_fecha', registro_fecha)
total_handler = CommandHandler('total_a_pagar', total_a_pagar)

#añadiendo los handlers
dispatcher.add_handler(start_handler)
dispatcher.add_handler(handler)
dispatcher.add_handler(destino_handler)
dispatcher.add_handler(help_handler)
dispatcher.add_handler(handler_exit)
dispatcher.add_handler(reserve_handler)
dispatcher.add_handler(fecha_handler)
dispatcher.add_handler(nombre_handler)
dispatcher.add_handler(pasajeros_handler)
dispatcher.add_handler(total_handler)]
```

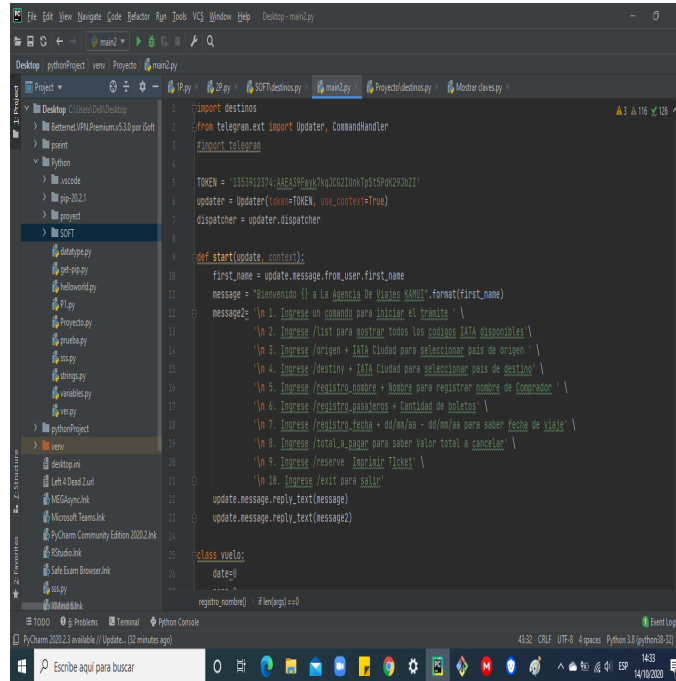
Además del comando /start se definen una serie de comandos más, no sin antes haberlos codificado, todos estos deben ser guardados en el dispatcher para que puedan ser usados.



Estas sentencias son necesarias. La sentencia `updater.start_polling()` inicia el bot, y el `updater.idle()` ayuda a que el bot espere respuesta.

0.5. Desarrollo, Entrega Y Retroalimentación

Se puede desarrollar un prototipo para cada uno de los componentes de la aplicación, se evalúan por todos los participantes con el fin de obtener mejores resultados mediante una retroalimentación.

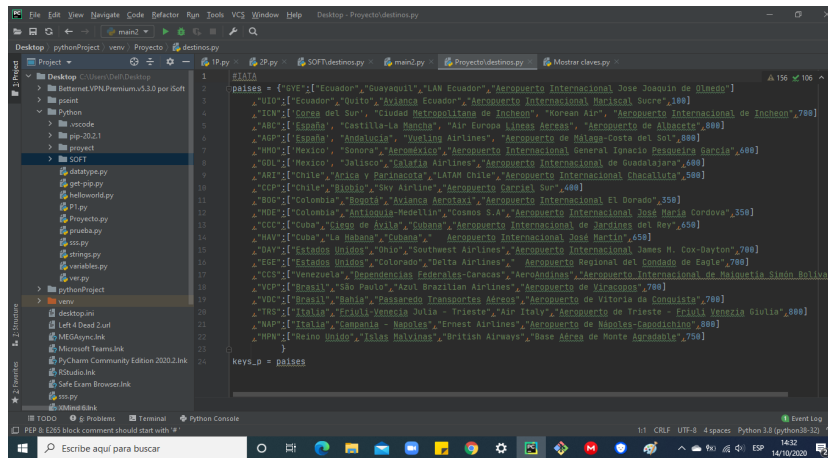


```

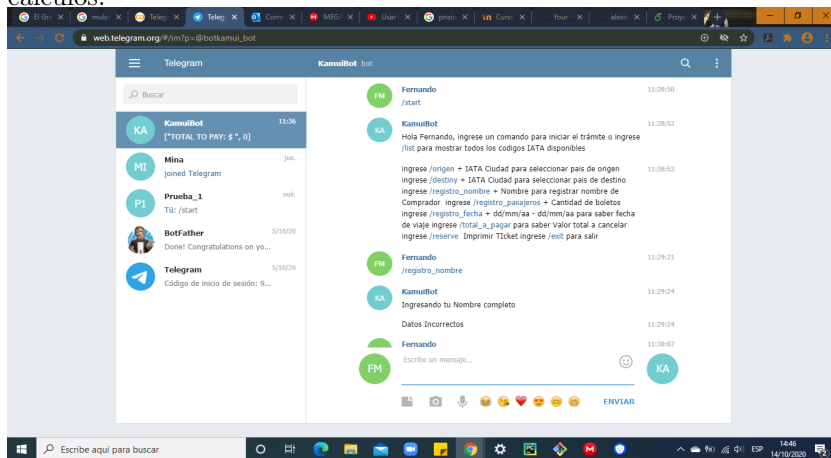
1  import destinos
2  from telegram.ext import Updater, CommandHandler
3  # Import telegram
4
5  TOKEN = '1333912374:AAE49Fmg7KqJ0G2J0nKtp5tSPK29700I'
6  updater = Updater(token=TOKEN, use_context=True)
7  dispatcher = updater.dispatcher
8
9  def start(update, context):
10     first_name = update.message.from_user.first_name
11     message = "Bienvenido {} a la Agencia de Viajes VAMU!".format(first_name)
12     message2 = '\n 1. Ingrese un comando para iniciar el trámite \n'
13     message2 = '\n 2. Ingrese /list para mostrar todos los códigos IATA disponibles \n'
14     message2 = '\n 3. Ingrese /origen + IATA Ciudad para seleccionar país de origen \n'
15     message2 = '\n 4. Ingrese /destino + IATA Ciudad para seleccionar país de destino \n'
16     message2 = '\n 5. Ingrese /registro_nombre + nombre para registrar nombre de pasajero \n'
17     message2 = '\n 6. Ingrese /registro_pasajeros + cantidad de boletos \n'
18     message2 = '\n 7. Ingrese /registro_fecha + dd/mm/aa - dd/mm/aa para saber fecha de viaje \n'
19     message2 = '\n 8. Ingrese /total_a_pagar para saber Valor total a cancelar \n'
20     message2 = '\n 9. Ingrese /reserve Ingrese Ticket \n'
21     message2 = '\n 10. Ingrese /exit para salir \n'
22     update.message.reply_text(message)
23     update.message.reply_text(message2)
24
25     class Vuelo:
26         date=0
27         registro_nombre=0
28         registro_nombre=0

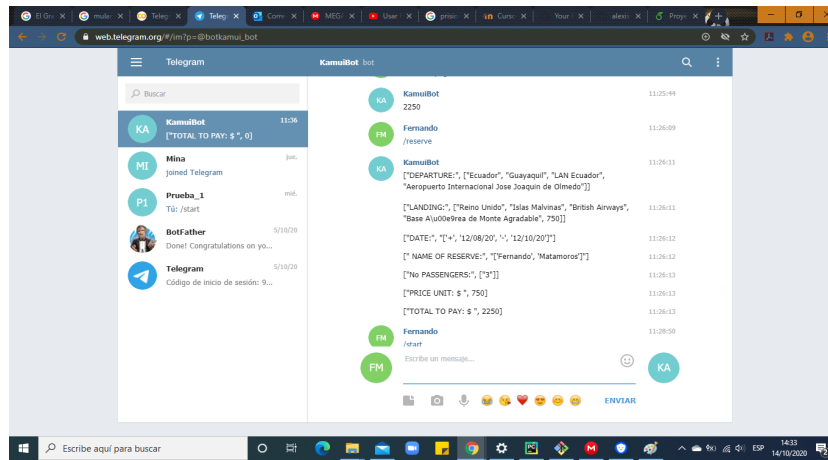
```

Se agregaron una nueva presentación al bot incluyendo nuevos comandos”tales como, Reservación, Registro de pasajeros, entre otros. La clase vuelo hace llamado a los demás módulos para realizar la respectiva presentación de los datos finales en el ticket de vuelo.



En el diccionario se agregó también el precio del vuelo para sus respectivos cálculos.





Presentación final del bot en Telegram y del ticket.

0.6. Estimación de costos

Para el presente proyecto nos basamos exclusivamente en las líneas de código:

Tamaño de líneas de código (sin espacio ni comentarios): $165KDL$

$$Esfuerzo = 3,0 * KLDC^{1,12} = 3,0 * 165^{1,12} = 913,48 \text{ personas-mes}$$

$$Tiempo = 2,5 * Esfuerzo^{0,35} = 2,5 * 913,48^{0,35} = 27 \text{ meses}$$

$$Personas = Esfuerzo/Tiempo = 913,48/27 = 33 \text{ personas}$$

0.7. Enlaces

<https://github.com/alexisevergarden/telegram>

[https://colab.research.google.com/drive/1sFBgDFG2763I6a_p4r00aEFxr1a58G9?
usp=sharing](https://colab.research.google.com/drive/1sFBgDFG2763I6a_p4r00aEFxr1a58G9?usp=sharing)