Universidad De Guayaquil

Facultad de Ciencias Matemáticas y Física

PROYECTO FINAL

Proyecto Final del Parcial

Autores:
-Vanessa Ronquillo
-Melissa Plaza
-Emily González

Octubre 2020

1 INTRODUCCIÓN

El presente proyecto tiene como propósito realizar una aplicación (bot) para una Agencia de Viajes, la cuál debe cumplir con los siguientes requerimientos, debe buscar destino u origen y en el caso de este proyecto se realizará buscando por Ciudad y está mostrará los vuelos que estén disponibles a ese destino o desde ese origen. La aplicación (bot) debe poder reservar vuelos de solo ida y considerar todos los datos necesarios para este proceso (ORIGEN-DESTINO, NUMERO DE ASIENTOS). Y por último debe reservar vuelos de ida-vuelta.

de viaje debe cumplir dicho sistema que se creará el cuál debe ser de uso fácil pero que contenga toda la información que se requiera. Para esto se hará uso de la aplicación TELEGRAM ya que en esta app se podrá visualizar y ejecutar el funcionamiento del programa.

El objetivo de esta investigación es poder llevar un control en el desarrollo de software a través de actividades que son partes de los procesos y así se garantice el éxito del proyecto y entregar un software netamente de calidad.

2 FASE DE ANÁLISIS

En el proyecto se usara principalmente la aplicación TELEGRAM ya que este sera el medio por el cual el cliente va poder comprar el vuelo que desee y este necesariamente lo hará buscando por ciudad para así verificar todos los vuelos que se disponen en ese momento. Esto estará realizado por un bot en la cuál se ejecutan dentro de la aplicación de mensajería. No necesitas instalarlos ni hacer nada diferente para poder utilizarlos, ya que se integran de manera que se utilizan como si fueran una persona real con la que interactúas. El programa necesariamente debe poder reservar vuelos de ida-vuelta y esté debe considerar todos los datos necesarios para el proceso.

DESCRIPCIÓN GENERAL

El desarrollo del proyecto se realiza a través de una secuencia simple de fases, cada fase tiene un conjunto de metas y actividades bien definidas.
Este modelo de software es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.
De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.[**danieltapia2016**]
Esos requisitos son la base sobre la que se construye el éxito. Por lo tanto, se lleva a cabo una descripción de los requisitos del software, y se acuerda entre el cliente y la empresa desarrolladora lo que el producto deberá hacer. Disponer de una especificación de los requisitos permite estimar de forma rigurosa las necesidades del software antes de su diseño. Además, permite tener una base a partir de la cual estimar el coste del producto, los riesgos y los plazos.

ANÁLISIS DE REQUERIMIENTOS

Se hace un breve análisis de los requerimientos que necesitara el usuario para desarrollar el sistema solicitado ya que esto permitirá que los Analistas determinen los requerimientos mas relevantes que facilitarán el correcto desarrollo de software.

- El software debe ser codificado únicamente en el lenguaje de programación Python.
- El programa debe ser capaz de leer cada una de las ciudades solicitadas por el usuario.
- El sistema debe identificar cuando se compra un boleto de solo ida e ida-vuelta,
- El sistema mostrara cuando el usuario haya comprado el boleto.

3 DISEÑO

Se utilizó los elementos y modelos obtenidos en la etapa de análisis para transformarlos en mecanismos que puedan ser utilizados en el entorno de desarrollo que se uso para realizar este proyecto.
La etapa de diseño y la de análisis son la mejor estructura para producir un software de calidad.
Se proyecta implementar un sistema que permita al usuario poder comprar un boleto por medio de un bot en la aplicación TELEGRAM, en la cual el sistema permitirá buscar los vuelos disponibles únicamente buscando por ciudad y asi poder realizar una compra efectiva al lugar de destino o desde ese origen.

4 CODIFICACIÓN

La codificación se la realiza en el lenguaje de programación Python, la cuál se ejecutará con un bot que se creo en la aplicación de telegram.

```
1  #LIBRERIAS
2  from bs4 import BeautifulSoup
3  import requests
4  import pandas as pd
5  import logging
6  from random import randrange, choice
7  import random
8  from telegram import (InlineKeyboardMarkup, InlineKeyboardButton)
9  from telegram.ext import (Updater, CommandHandler, MessageHandler, Filters,
10                            ConversationHandler, CallbackQueryHandler)
```

- **BeautifulSoup:** Beautiful Soup es una biblioteca de Python para analizar documentos HTML. Esta biblioteca crea un árbol con todos los elementos del documento y puede ser utilizado para extraer información. Se utiliza para el web scraping.

- **Requests:** La librería requests nos permite enviar solicitudes HTTP con Python sin necesidad de tanta labor manual, haciendo que la integración con los servicios web sea mucho más fácil.

- **Pandas:** Es una librería para el análisis de datos que cuenta con las estructuras de datos que necesitamos para limpiar los datos en bruto y que sean aptos para el análisis (por ejemplo, tablas).

- **Logging:**Es una herramienta útil para prevenir errores, controlar los ataques de piratas informáticos o, simplemente, llevar a cabo análisis .

- **Randrange:**La función randrange() devuelve enteros que van desde un valor inicial a otro final separados entre sí un número de valores determinados.

- **Choice:**La función choice() se utiliza para seleccionar elementos al azar de una lista. En el siguiente ejemplo se obtiene un elemento de los cinco existentes en una lista.

```
# Habilitar el registro
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)
# Definiciones de los estados para la conversación de alto nivel
SELECTING_ACTION, GUAYAQUIL, QUITO, DESCRIBING_G = map(chr, range(4))
# Definiciones para un subrango de la conversacion
SELECTING_LEVEL, SELECTING_LEVELW = map(chr, range(4, 6))
# Deficiciones para la descripion
SELECTING_FEATURE, TYPING = map(chr, range(6, 8))
# Definicones de metodos
STOPPING, SHOWING = map(chr, range(8, 10))
S = map(chr, range(1,5))
# Un alto de las acciones
END = ConversationHandler.END
# Different constants for this example
(QUITUS, CHILDREN, GUAYAQUILS, GENDER, A1, A2, TIPO, LLOS, START_OVER, FEATURES,
 CURRENT_FEATURE, CURRENT_LEVEL) = map(chr, range(10, 22))
de= list()

#scrapin de una página con ciertos parametros
def button(llos,Tvuelo):
    if llos == "salidas":
        page = requests.get("https://www.aeropuertoquito.aero/es/vuelos-"+Tvuelo+"/salidas-"+Tvuelo+".html")
        soup = BeautifulSoup(page.content, 'html.parser')
    else:
        page = requests.get("https://www.aeropuertoquito.aero/es/vuelos-" + Tvuelo + "/llegadas-" + Tvuelo + ".html")
        soup = BeautifulSoup(page.content, 'html.parser')

    hora = soup.find_all('td', class_='td1 hora')
    ho = list()
    for i in hora:
        ho.append(i.text)
    print(ho)
```

```
    print(ho)
    print(len(ho))
    aerolinea = soup.find_all('td', attrs={'class': 'td2'})
    aer = list()
    for i in aerolinea:
              aer.append(i.text)
    print(aer)
    print(len(aer))
    vuelo = soup.find_all('td', attrs={'class': 'td3'})
    vue = list()
    for i in vuelo:
           vue.append(i.text)
    print(vue)
    print(len(vue))
    destino = soup.find_all('td', class_='td4')
    dest = list()
    for i in destino:
        dest.append(i.text)
    print(dest)
    print(len(dest))
    dia = soup.find_all('td', class_='td5')
    dias = list()
    for i in dia:
        dias.append(i.text)
    print(dias)
    print(len(dias))
    df = pd.DataFrame({"Hora": ho, "Aerolinea": aer, "Vuelo": vue, "Destino": dest, "Disponibilidad": dias})
    print(df)
    for i in range(len(dias)):
        de.extend([[aer[i], dest[i], vue[i] , dias[i], ho[i] ]])
    print(de)

# Helper
def _name_switcher(level):
    if level == GUAYAQUILS:
        return (' ', ' ')
```

```
# CONVERSACION INICIADA
def start(update, context):
    text = 'PRESIONE EL BOTÓN DE MOSTRAR VUELOS. ' \
           '\n\n Si usted desea para esta acción -> /stop.'
    buttons = [[InlineKeyboardButton(text='Aeropuerto José Joaquín de Olmedo', callback_data=str(GUAYAQUIL))], [
        InlineKeyboardButton(text='Aeropuerto Mariscal Sucre', callback_data=str(QUITO))],
        [InlineKeyboardButton(text='MOSTRAR VUELOS', callback_data=str(SHOWING)), InlineKeyboardButton(text='FIN', callback_data=str(END))] ]
    keyboard = InlineKeyboardMarkup(buttons)
    if context.user_data.get(START_OVER):
        update.callback_query.answer()
        update.callback_query.edit_message_text(text=text, reply_markup=keyboard)
    else:
        update.message.reply_text('Hola, bienvenido/a a  Agencia de viajes VEM, este bot es una nueva modalidad a implementar en la agencia con el fin de interactuar más con
                                  '\n\n SIGA LAS INTRUCCIONES PARA ASEGURAR LA EFICIENCIA DEL PROGRAMA')
        update.message.reply_text(text=text, reply_markup=keyboard)

    context.user_data[START_OVER] = False
    return SELECTING_ACTION
#QUITO
def adding_self(update, context):
    context.user_data[CURRENT_LEVEL] = QUITUS
    text = 'Okay, presione el boton.'
    button = InlineKeyboardButton(text='Ingreso de datos', callback_data=str(A1))
    keyboard = InlineKeyboardMarkup.from_button(button)
    update.callback_query.answer()
    update.callback_query.edit_message_text(text=text, reply_markup=keyboard)
    return DESCRIBING_G

#mostrar los datos sacados
def show_data(update, context):
    def prettyprint(user_data, level):
        people = user_data.get(level)
        if not people:
            return '\nNo se ha ingresado datos, se  mostrara todos los vuelos disponibles'
               return '\nNo se ha ingresado datos, se  mostrara todos los vuelos disponibles'
          text = ''
          if level == QUITUS:
              for person in user_data[level]:
                  text += '\nTIPO: {}, \n VUELO: {}'.format(person.get(LLOS, '-'), person.get(TIPO, '-'))
                  text2 = "{}".format(person.get(LLOS))
                  text3= "{}".format(person.get(TIPO))

                  if (text2 == "nacionales" or  text2 == "Nacionales"):
                      if (text3=="llegada" or text3 == "Llegada"):
                          button("llegadas","nacionales")
                      else:
                          button("salidas", "nacionales")
                  else:
                      if text3=="Llegada" or text3 == "llegada":
                          button("llegadas", "internacionales")
                      else:
                          button("salidas", "internacionales")
          else:

              male, female = _name_switcher(level)

              for person in user_data[level]:
                  gender = female if person[GENDER] == A2 else male
                  text += '\n{}: TIPO: {}, VUELO DE: {}'.format(gender, person.get(LLOS, '-'),
                                                          person.get(TIPO, '-'))

          return text
      ud = context.user_data
      text = 'Aeropuerto Mariscal Sucre: ' + prettyprint(ud, QUITUS)
      text += '\n\n Aeropuerto José Joaquín de Olmedo: ' + prettyprint(ud, GUAYAQUILS)
      buttons = [[
          InlineKeyboardButton(text='Confirmar Datos', callback_data=str(S)),
          InlineKeyboardButton(text='Regresar', callback_data=str(END))]]
      keyboard = InlineKeyboardMarkup(buttons)
      update.callback_query.answer()
```

```
    update.callback_query.answer()
    update.callback_query.edit_message_text(text=text, reply_markup=keyboard)
    ud[START_OVER] = True
    return SHOWING

#SE PRESENTA UN MENSAJE PARA COMENZAR POR COMANDOS
def listavie(update, context):
    update.callback_query.edit_message_text(text="Para reservar un vuelo en el areopuerto escogido -> /vuelo \n\n PARA VISUALIZAR TODOS-> /tvuelos")
    return S

#ES COGE EL NUMERO DE VUELO
def l (update, context):
    c = 0
    for i in range(len(de)):
       update.message.reply_text(text=de[i], callback_data=i)
       c = i
    update.message.reply_text("ESCRIBA UN NÚMERO DEL 0 AL {} PARA ESCOGER EL VUELO ".format(c))

#TODOS LOS VUELOS DEL SISTEMA
def todosvuelos(update,context):
    page = requests.get("https://www.aeropuertoquito.aero/es/vuelos-nacionales/llegadas-nacionales.html")
    page2 = requests.get("https://www.aeropuertoquito.aero/es/vuelos-nacionales/salidas-nacionales.html")
    page3 = requests.get("https://www.aeropuertoquito.aero/es/vuelos-internacionales/llegadas-internacionales.html")
    page4 = requests.get("https://www.aeropuertoquito.aero/es/vuelos-internacionales/salidas-internacionales.html")
    soup = BeautifulSoup(page.content, 'html.parser')
    soup2 = BeautifulSoup(page2.content, 'html.parser')
    soup3 = BeautifulSoup(page3.content, 'html.parser')
    soup4 = BeautifulSoup(page4.content, 'html.parser')
    hora = soup.find_all('td', class_='td1 hora') +  soup2.find_all('td', class_='td1 hora') + soup3.find_all('td', class_='td1 hora') + soup4.find_all('td', class_='td1 hora'
    ho = list()
    for i in hora:
        ho.append(i.text)
    print(ho)
    print(len(ho))
    aerolinea = soup.find_all('td', attrs={'class': 'td2'}) + soup2.find_all('td',
                                                                             attrs={'class': 'td2'}) + soup3.find_all(
```

```
                                                                                                attrs={'class': 'td2'}) + soup3.find_all(
        'td', attrs={'class': 'td2'}) + soup4.find_all('td', attrs={'class': 'td2'})

    aer = list()
    for i in aerolinea:
        if (len(aer) < len(ho)):
            aer.append(i.text)
    print(aer)
    print(len(aer))
    vuelo = soup.find_all('td', attrs={'class': 'td3'}) + soup2.find_all('td', attrs={'class': 'td3'}) + soup3.find_all(
        'td', attrs={'class': 'td3'}) + soup4.find_all('td', attrs={'class': 'td3'})
    vue = list()
    for i in vuelo:
        if (len(vue) < len(aer)):
            vue.append(i.text)
    print(vue)
    print(len(vue))
    destino = soup.find_all('td', class_='td4') + soup2.find_all('td', class_='td4') + soup3.find_all('td',
                                                                                                       class_='td4') + soup4.find_all(
        'td', class_='td4')
    dest = list()
    for i in destino:
        dest.append(i.text)
    print(dest)
    print(len(dest))
    dias = list()
    dia = soup.find_all('td', class_='td5') + soup2.find_all('td', class_='td5') + soup3.find_all('td',
                                                                                                  class_='td5') + soup4.find_all(
        'td', class_='td5')
    for i in dia:
        dias.append(i.text)
    print(dias)
    print(len(dias))
    df = pd.DataFrame({"Hora": ho, "Aerolinea": aer, "Vuelo": vue, "Destino": dest, "Disponibilidad": dias})
    print(df)
    for i in range(len(dias)):
```

```

def stop(update, context):
    update.message.reply_text('Okay, bye.')
    return END

#METODO PARA TERMINAR
def end(update, context):
    update.callback_query.answer()
    text = 'Nos vemos'
    update.callback_query.edit_message_text(text=text)
    return END

#PARA SELCCIONAR LAS CARACTERISTICAS
def select_feature(update, context):
    buttons = [[InlineKeyboardButton(text='¿Nacional o Internacional?', callback_data=str(LLOS))],
         [InlineKeyboardButton(text='LISTO', callback_data=str(END))]]
    keyboard = InlineKeyboardMarkup(buttons)
    if not context.user_data.get(START_OVER):
        context.user_data[FEATURES] = {GENDER: update.callback_query.data}
        text = 'Siga las instruciones: \n\n Presione el botón'\
            '\n\n Escriba: "Nacionales"  o "Internacionales" respectivamente '
        update.callback_query.answer()
        update.callback_query.edit_message_text(text=text, reply_markup=keyboard)
    # But after we do that, we need to send a new message
    else:

        text = '¡Lo tengo! Presione el botón \n\n Escriba: "Llegada" en caso que desee viajar hacia el lugar escogido o "Salida" en caso que desee viajar desde el lugar escogi
        buttons = [[InlineKeyboardButton(text='¿Llegada o salida?', callback_data=str(TIPO))],
                   [InlineKeyboardButton(text='LISTO', callback_data=str(END))]]
        keyboard = InlineKeyboardMarkup(buttons)
        update.message.reply_text(text=text, reply_markup=keyboard)

    context.user_data[START_OVER] = False
    return SELECTING_FEATURE
```

```
255
256 def stop(update, context):
257     update.message.reply_text('Okay, bye.')
258     return END
259
260 #METODO PARA TERMINAR
261 def end(update, context):
262     update.callback_query.answer()
263     text = 'Nos vemos'
264     update.callback_query.edit_message_text(text=text)
265     return END
266
267 #PARA SELCCIONAR LAS CARACTERISTICAS
268 def select_feature(update, context):
269     buttons = [[InlineKeyboardButton(text='¿Nacional o Internacional?', callback_data=str(LLOS))],
270          [InlineKeyboardButton(text='LISTO', callback_data=str(END))]]
271     keyboard = InlineKeyboardMarkup(buttons)
272     if not context.user_data.get(START_OVER):
273         context.user_data[FEATURES] = {GENDER: update.callback_query.data}
274         text = 'Siga las instruciones: \n\n Presione el botón'\
275             '\n\n Escriba: "Nacionales"  o "Internacionales" respectivamente '
276         update.callback_query.answer()
277         update.callback_query.edit_message_text(text=text, reply_markup=keyboard)
278     # But after we do that, we need to send a new message
279     else:
280
281         text = '¡Lo tengo! Presione el botón \n\n Escriba: "Llegada" en caso que desee viajar hacia el lugar escogido o "Salida" en caso que desee viajar desde el lugar escogi
282         buttons = [[InlineKeyboardButton(text='¿Llegada o salida?', callback_data=str(TIPO))],
283                    [InlineKeyboardButton(text='LISTO', callback_data=str(END))]]
284         keyboard = InlineKeyboardMarkup(buttons)
285         update.message.reply_text(text=text, reply_markup=keyboard)
286
287     context.user_data[START_OVER] = False
288     return SELECTING_FEATURE
```

```python
#preguntapra repetir la accion
def ask_for_input(update, context):
    context.user_data[CURRENT_FEATURE] = update.callback_query.data
    text = 'Escriba según lo explicado anteriormente:'
    update.callback_query.answer()
    update.callback_query.edit_message_text(text=text)
    return TYPING

def save_input(update, context):
    #GUARDA LOS DATOS
    ud = context.user_data
    ud[FEATURES][ud[CURRENT_FEATURE]] = update.message.text
    ud[START_OVER] = True
    return select_feature(update, context)

#termina la accion de pra volver
def end_describing(update, context):
    ud = context.user_data
    level = ud[CURRENT_LEVEL]
    if not ud.get(level):
        ud[level] = []
    ud[level].append(ud[FEATURES])
    if level == QUITUS:
        ud[START_OVER] = True
        start(update, context)
    else:
        context.user_data[START_OVER] = True
        start(update, context)
    return END

#corta la accion directament
def stop_nested(update, context):
    update.message.reply_text('Okay, Cuidate.')
    return STOPPING

#llama a todos lo metodos
```

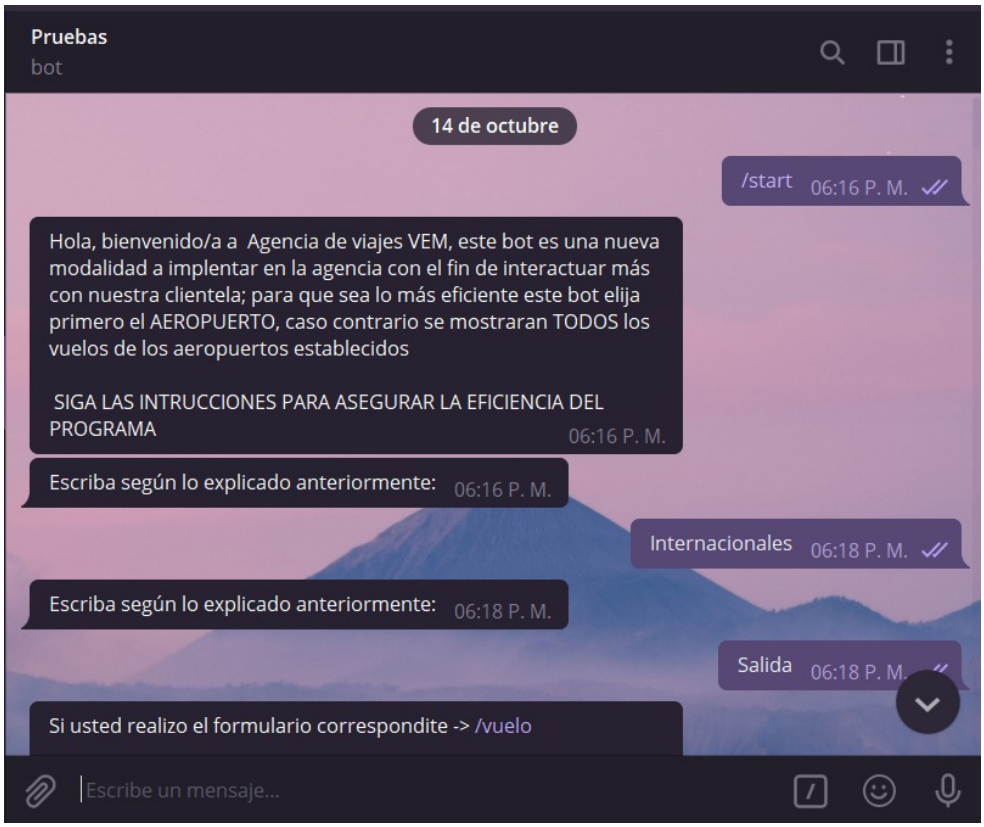
```
#llama a todos lo metodos
def main():
    updater = Updater("1331458519:AAHV1B1XmZ71Dfj0gUDFry2UQnMor_xp34M", use_context=True)
    dp = updater.dispatcher
    description_conv = ConversationHandler(
        entry_points=[CallbackQueryHandler(select_feature,
                                           pattern='^' + str(A1) + '$|^' + str(A2) + '$')],
        states={
            SELECTING_FEATURE: [CallbackQueryHandler(ask_for_input,
                                                     pattern='^(?!' + str(END) + ').*$')],
            S: [CallbackQueryHandler(end_describing, pattern='^' + str(S))],

            TYPING: [MessageHandler(Filters.text & ~Filters.command, save_input)],
        },

        fallbacks=[
            CallbackQueryHandler(end_describing, pattern='^' + str(END) + '$'),
            CommandHandler('stop', stop_nested), CommandHandler("vuelo",l)
        ],

        map_to_parent={
            END: SELECTING_LEVEL,
            STOPPING: STOPPING,
        }
    )
    add_conv = ConversationHandler(
        entry_points=[CallbackQueryHandler(select_q,
                                           pattern='^' + str(GUAYAQUIL) + '$')],

        states={
                        SELECTING_LEVELW: [description_conv]
        },

        fallbacks=[
            CallbackQueryHandler(show_data, pattern='^' + str(SHOWING) + '$'),
            CallbackQueryHandler(end_second_level, pattern='^' + str(END) + '$'),
```

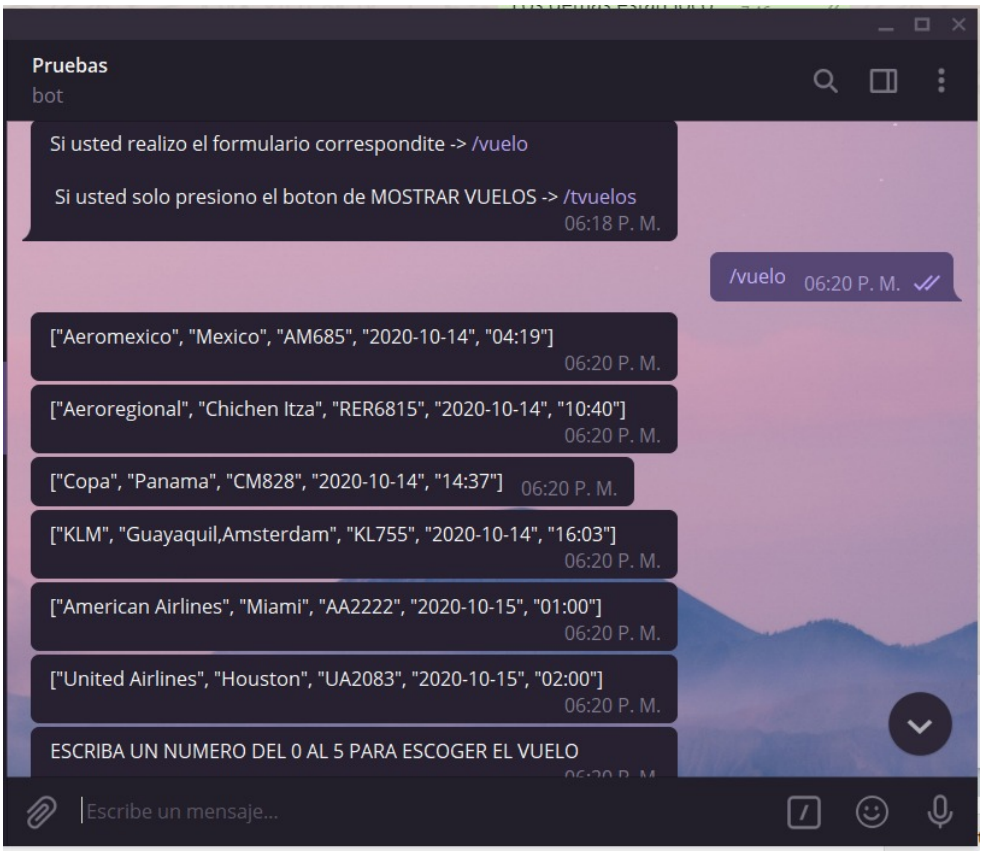
```
363
364             map_to_parent={
365                 SHOWING: SHOWING,
366                 END: SELECTING_ACTION,
367                 STOPPING: END,
368             }
369         )
370
371         selection_bac = [
372             CallbackQueryHandler(listavie, pattern='^' + str(S)),
373             CallbackQueryHandler(start, pattern='^' + str(END) + '$'),
374         ]
375         selection_handlers = [
376             add_conv,
377             CallbackQueryHandler(show_data, pattern='^' + str(SHOWING) + '$'),
378             CallbackQueryHandler(adding_self, pattern='^' + str(QUITO) + '$'),
379             CallbackQueryHandler(end, pattern='^' + str(END) + '$'),
380         ]
381         conv_handler = ConversationHandler(
382             entry_points=[CommandHandler('start', start)],
383
384             states={
385                 SHOWING: selection_bac,
386                 SELECTING_ACTION: selection_handlers,
387                 SELECTING_LEVEL: selection_handlers,
388                 DESCRIBING_G: [description_conv],
389                 STOPPING: [CommandHandler('start', start)],
390             },
391
392             fallbacks=[CommandHandler('stop', stop)  ],
393
394         )
395         #comandos
```

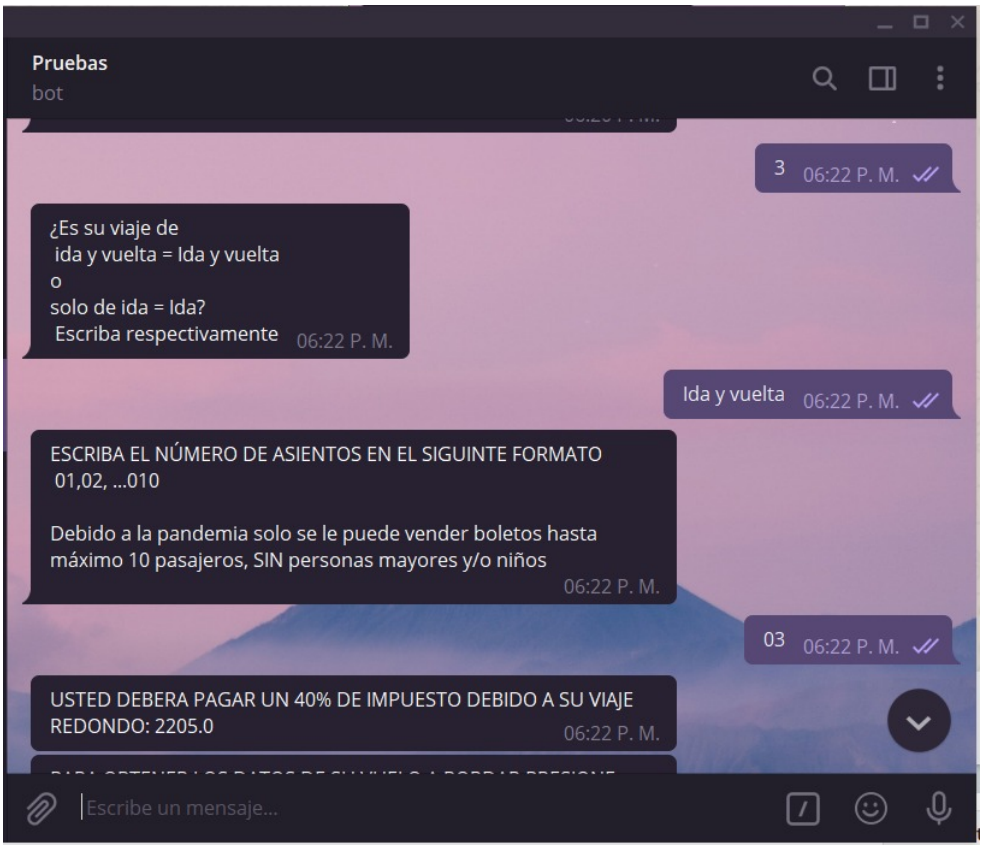
```
383
384            states={
385                SHOWING: selection_bac,
386                SELECTING_ACTION: selection_handlers,
387                SELECTING_LEVEL: selection_handlers,
388                DESCRIBING_G: [description_conv],
389                STOPPING: [CommandHandler('start', start)],
390            },
391
392            fallbacks=[CommandHandler('stop', stop)  ],
393
394        )
395        #comandos
396        dp.add_handler(CommandHandler("vuelo", l))
397        dp.add_handler(MessageHandler(Filters.regex("^(1|0|2|3|4|5|6|7|8|9|10|11|12|11|13|14|15|17|18|19|20|21|22|21|23|24|25|27|28|29|30|31|32|31|33|34|
398        dp.add_handler(conv_handler)
399        updater.start_polling()
400        updater.idle()
401
402
403    if __name__ == '__main__':
404        main()
```

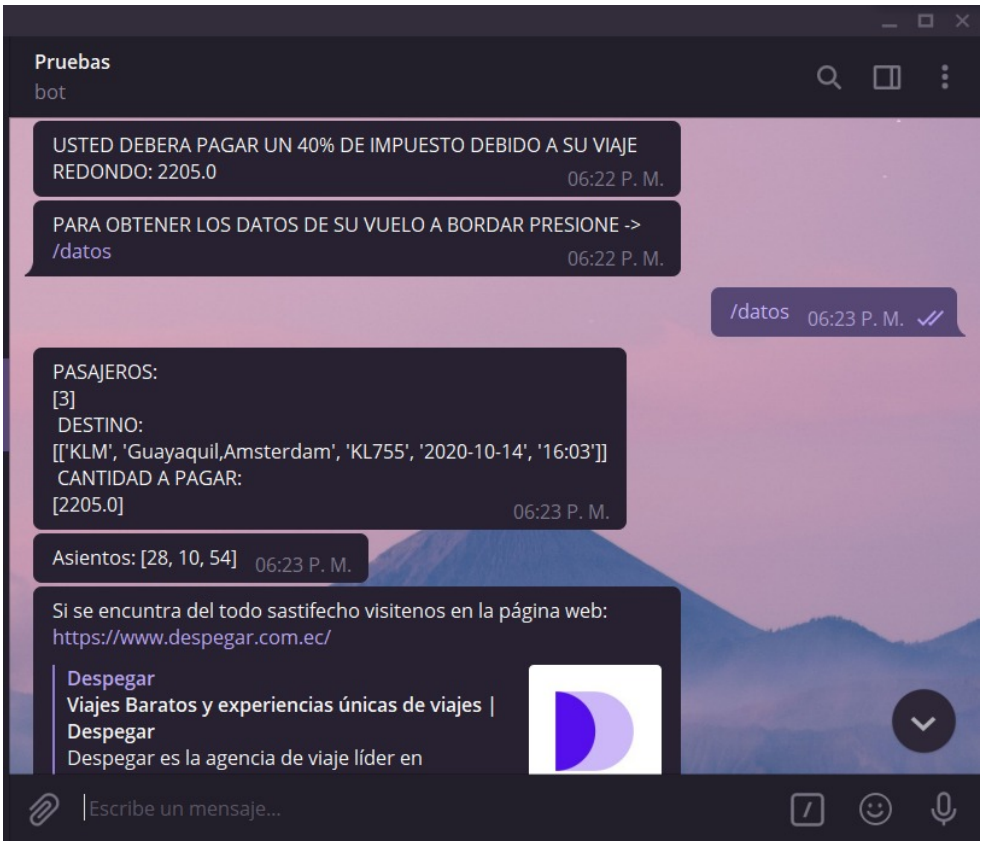
5 PRUEBAS

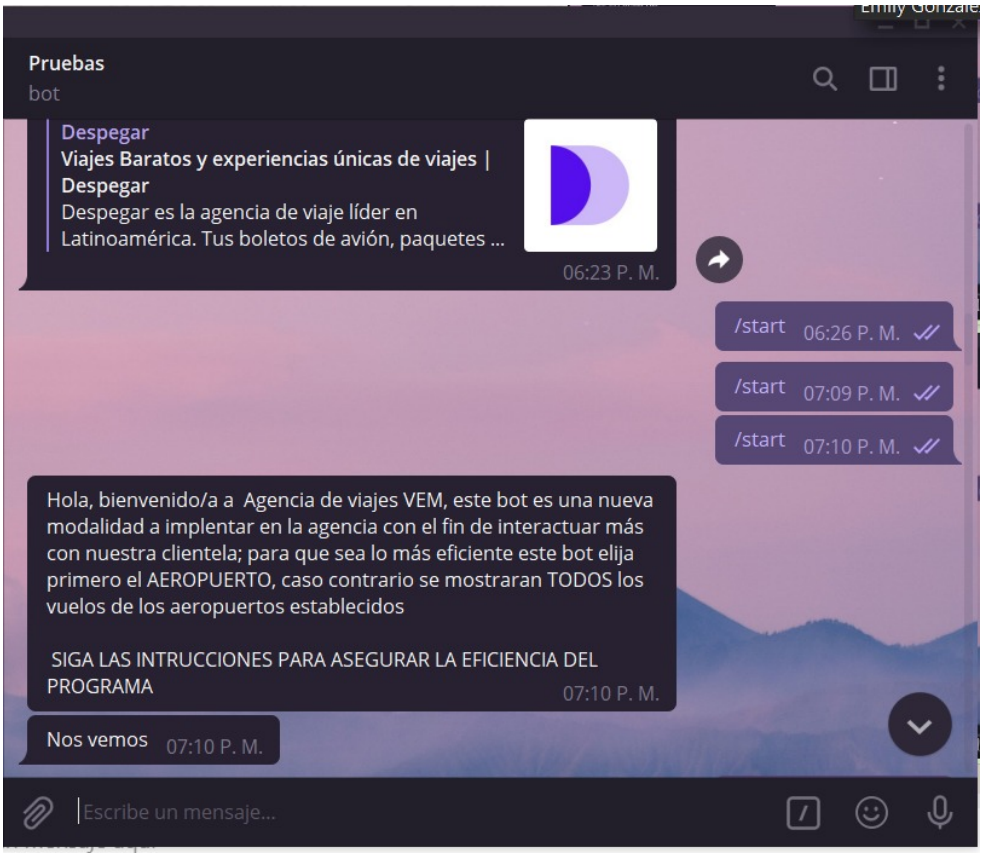
Conforme se implementaban funciones al software se procedía a probar, para comprobar la correcta implementación de las mismas, lo cual llevo a colocar condiciones o excepciones para evitar el mayor numero de errores, comprobando asi que todos los datos sean presentados de manera correcta.

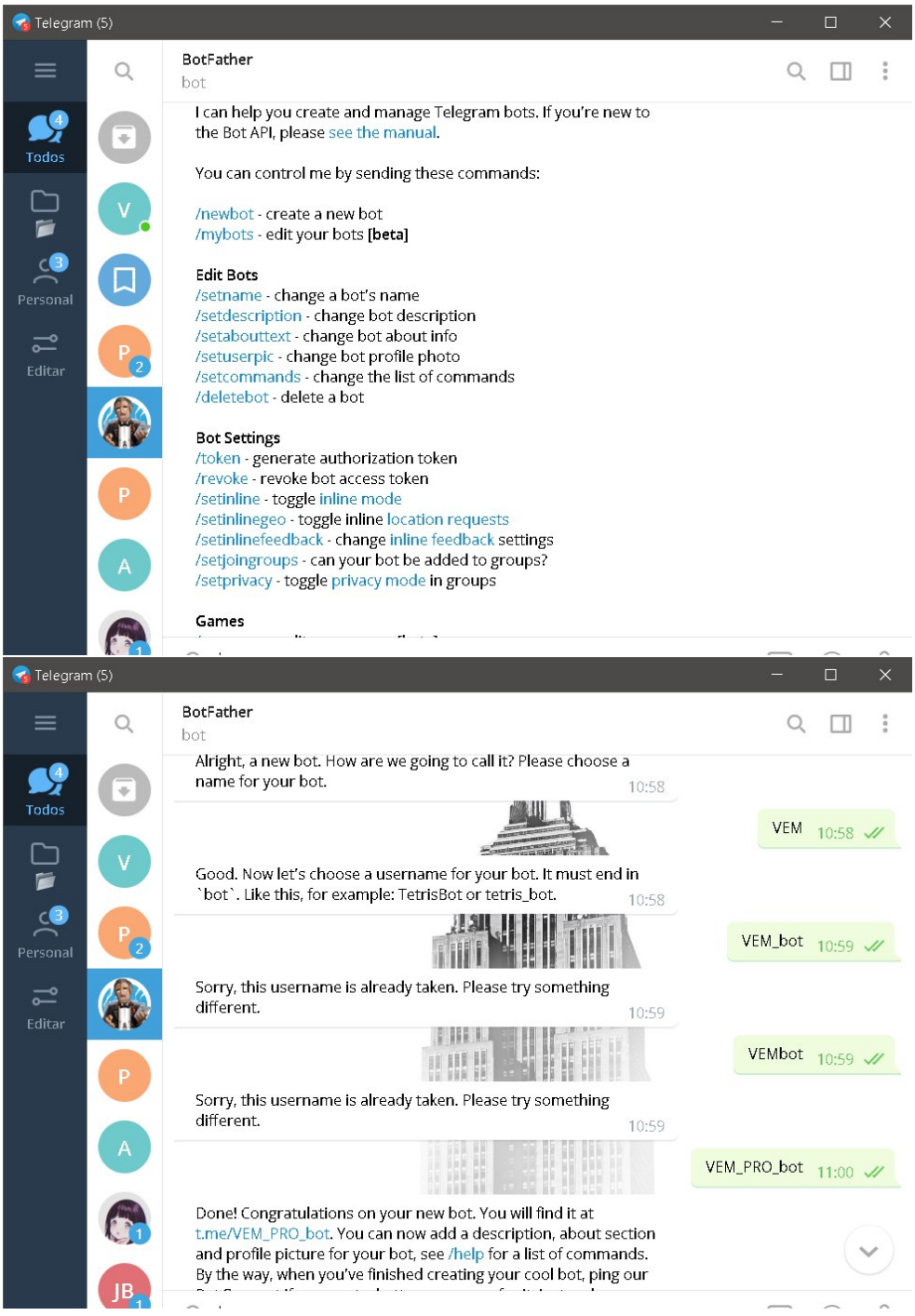


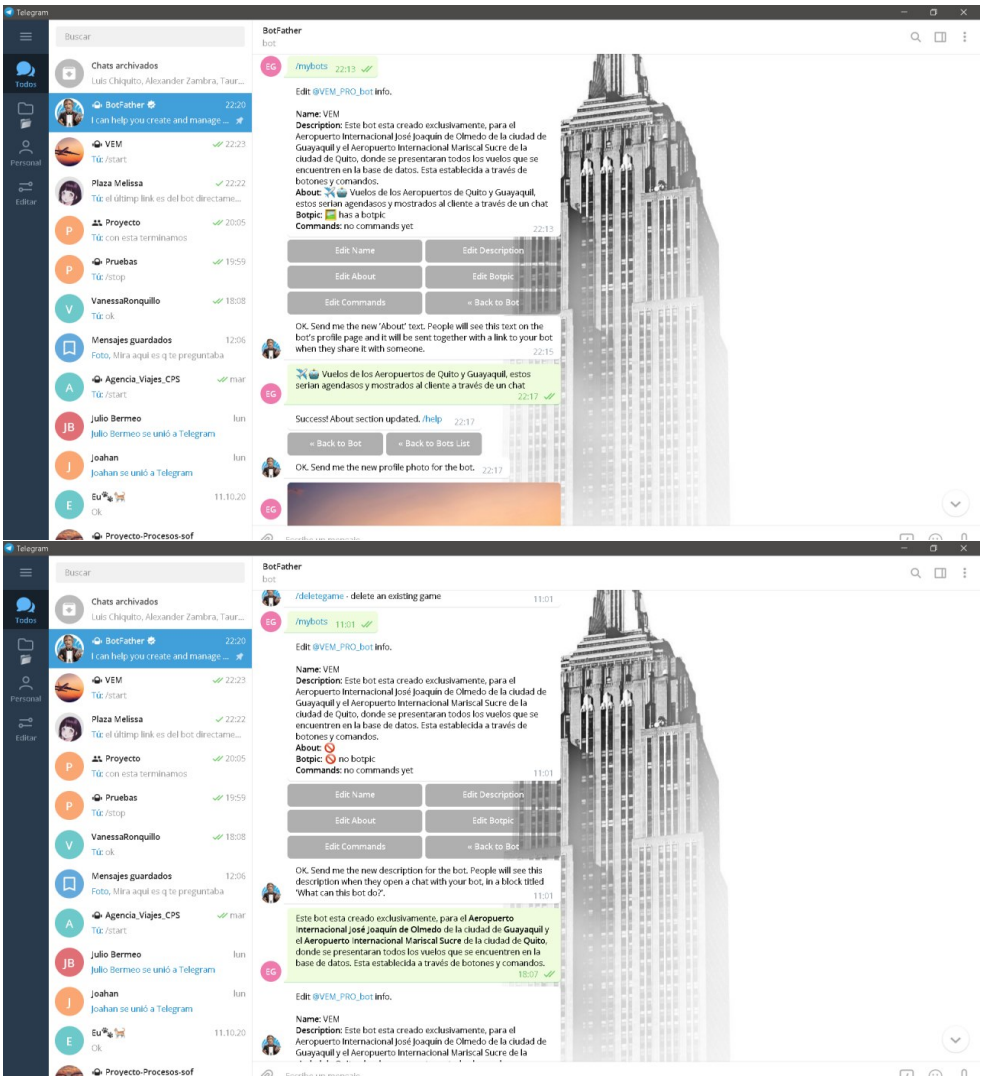

Pruebas
bot
Si usted realizo el formulario correspondite -> /vuelo
Si usted solo presiono el boton de MOSTRAR VUELOS -> /tvuelos
06:18 P. M.
/vuelo 06:20 P. M.
["Aeromexico", "Mexico", "AM685", "2020-10-14", "04:19"]
06:20 P. M.
["Aeroregional", "Chichen Itza", "RER6815", "2020-10-14", "10:40"]
06:20 P. M.
["Copa", "Panama", "CM828", "2020-10-14", "14:37"] 06:20 P. M.
["KLM", "Guayaquil,Amsterdam", "KL755", "2020-10-14", "16:03"]
06:20 P. M.
["American Airlines", "Miami", "AA2222", "2020-10-15", "01:00"]
06:20 P. M.
["United Airlines", "Houston", "UA2083", "2020-10-15", "02:00"]
06:20 P. M.
ESCRIBA UN NUMERO DEL 0 AL 5 PARA ESCOGER EL VUELO
Escribe un mensaje...


Pruebas
bot
3 06:22 P. M.
¿Es su viaje de
ida y vuelta = Ida y vuelta
o
solo de ida = Ida?
Escriba respectivamente 06:22 P. M.
Ida y vuelta 06:22 P. M.
ESCRIBA EL NÚMERO DE ASIENTOS EN EL SIGUINTE FORMATO
01,02, ...010
Debido a la pandemia solo se le puede vender boletos hasta
máximo 10 pasajeros, SIN personas mayores y/o niños
06:22 P. M.
03 06:22 P. M.
USTED DEBERA PAGAR UN 40% DE IMPUESTO DEBIDO A SU VIAJE
REDONDO: 2205.0 06:22 P. M.
Escribe un mensaje...


Pruebas
bot
USTED DEBERA PAGAR UN 40% DE IMPUESTO DEBIDO A SU VIAJE REDONDO: 2205.0
06:22 P. M.
PARA OBTENER LOS DATOS DE SU VUELO A BORDAR PRESIONE ->
/datos
06:22 P. M.
/datos 06:23 P. M.
PASAJEROS:
[3]
DESTINO:
[['KLM', 'Guayaquil,Amsterdam', 'KL755', '2020-10-14', '16:03']]
CANTIDAD A PAGAR:
[2205.0]
06:23 P. M.
Asientos: [28, 10, 54] 06:23 P. M.
Si se encuntra del todo sastifecho visitenos en la página web:
https://www.despegar.com.ec/
Despegar
Viajes Baratos y experiencias únicas de viajes | Despegar
Despegar es la agencia de viaje líder en
Escribe un mensaje...


Pruebas
bot
Despegar
Viajes Baratos y experiencias únicas de viajes | Despegar
Despegar es la agencia de viaje líder en Latinoamérica. Tus boletos de avión, paquetes ...
06:23 P. M.
/start 06:26 P. M.
/start 07:09 P. M.
/start 07:10 P. M.
Hola, bienvenido/a a Agencia de viajes VEM, este bot es una nueva modalidad a implentar en la agencia con el fin de interactuar más con nuestra clientela; para que sea lo más eficiente este bot elija primero el AEROPUERTO, caso contrario se mostraran TODOS los vuelos de los aeropuertos establecidos
SIGA LAS INTRUCCIONES PARA ASEGURAR LA EFICIENCIA DEL PROGRAMA
07:10 P. M.
Nos vemos 07:10 P. M.
Escribe un mensaje...


Telegram (5)
BotFather
bot
I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.
You can control me by sending these commands:
/newbot - create a new bot
/mybots - edit your bots [beta]
Edit Bots
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot
Bot Settings
/token - generate authorization token
/revoke - revoke bot access token
/setinline - toggle inline mode
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added to groups?
/setprivacy - toggle privacy mode in groups
Games
Todos
Personal
Editar
Telegram (5)
BotFather
bot
Alright, a new bot. How are we going to call it? Please choose a name for your bot. 10:58
VEM 10:58
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 10:58
VEM_bot 10:59
Sorry, this username is already taken. Please try something different. 10:59
VEMbot 10:59
Sorry, this username is already taken. Please try something different. 10:59
VEM_PRO_bot 11:00
Done! Congratulations on your new bot. You will find it at t.me/VEM_PRO_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our
Todos
Personal
Editar


Telegram
Buscar
Chats archivados
Luis Chiquito, Alexander Zambra, Taur...
BotFather 22:20
I can help you create and manage ...
VEM 22:23
Tú: /start
Plaza Melissa 22:22
Tú: el último link es del bot directame...
Proyecto 20:05
Tú: con esta terminamos
Pruebas 19:59
Tú: /stop
VanessaRonquillo 18:08
Tú: ok
Mensajes guardados 12:06
Foto, Mira aquí es q te preguntaba
Agencia_Viajes_CPS mar
Tú: /start
Julio Bermeo lun
Julio Bermeo se unió a Telegram
Joahan lun
Joahan se unió a Telegram
11.10.20
Ok
Proyecto-Procesos-sof
Todos
Personal
Editar
BotFather
bot
/mybots 22:13
Edit @VEM_PRO_bot info.
Name: VEM
Description: Este bot esta creado exclusivamente, para el Aeropuerto Internacional José Joaquín de Olmedo de la ciudad de Guayaquil y el Aeropuerto Internacional Mariscal Sucre de la ciudad de Quito, donde se presentaran todos los vuelos que se encuentren en la base de datos. Esta establecida a través de botones y comandos.
About: ✈️🌆 Vuelos de los Aeropuertos de Quito y Guayaquil, estos serían agendasos y mostrados al cliente a través de un chat
Botpic: has a botpic
Commands: no commands yet
22:13
Edit Name
Edit Description
Edit About
Edit Botpic
Edit Commands
« Back to Bot
OK. Send me the new 'About' text. People will see this text on the bot's profile page and it will be sent together with a link to your bot when they share it with someone.
22:15
✈️🌆 Vuelos de los Aeropuertos de Quito y Guayaquil, estos serían agendasos y mostrados al cliente a través de un chat
22:17
Success! About section updated. /help 22:17
« Back to Bot
« Back to Bots List
OK. Send me the new profile photo for the bot. 22:17
Telegram
Buscar
Chats archivados
Luis Chiquito, Alexander Zambra, Taur...
BotFather 22:20
I can help you create and manage ...
VEM 22:23
Tú: /start
Plaza Melissa 22:22
Tú: el último link es del bot directame...
Proyecto 20:05
Tú: con esta terminamos
Pruebas 19:59
Tú: /stop
VanessaRonquillo 18:08
Tú: ok
Mensajes guardados 12:06
Foto, Mira aquí es q te preguntaba
Agencia_Viajes_CPS mar
Tú: /start
Julio Bermeo lun
Julio Bermeo se unió a Telegram
Joahan lun
Joahan se unió a Telegram
11.10.20
Ok
Proyecto-Procesos-sof
Todos
Personal
Editar
BotFather
bot
/deletegame - delete an existing game 11:01
/mybots 11:01
Edit @VEM_PRO_bot info.
Name: VEM
Description: Este bot esta creado exclusivamente, para el Aeropuerto Internacional José Joaquín de Olmedo de la ciudad de Guayaquil y el Aeropuerto Internacional Mariscal Sucre de la ciudad de Quito, donde se presentaran todos los vuelos que se encuentren en la base de datos. Esta establecida a través de botones y comandos.
About: 🚫
Botpic: 🚫 no botpic
Commands: no commands yet
11:01
Edit Name
Edit Description
Edit About
Edit Botpic
Edit Commands
« Back to Bot
OK. Send me the new description for the bot. People will see this description when they open a chat with your bot, in a block titled 'What can this bot do?'.
11:01
Este bot esta creado exclusivamente, para el Aeropuerto Internacional José Joaquín de Olmedo de la ciudad de Guayaquil y el Aeropuerto Internacional Mariscal Sucre de la ciudad de Quito, donde se presentaran todos los vuelos que se encuentren en la base de datos. Esta establecida a través de botones y comandos.
11:01
Edit @VEM_PRO_bot info.
Name: VEM
Description: Este bot esta creado exclusivamente, para el Aeropuerto Internacional José Joaquín de Olmedo de la ciudad de Guayaquil y el Aeropuerto Internacional Mariscal Sucre de la

6 MANTENIMIENTO

A partir de tener el sofware final se ha seguido revisando el código para comprobar su calidad y ha estado funcionando correctamente según los requerimientos planteados en un principio, hasta ahora no se agregado nuevos requerimientos, conforme el uso de la aplicación en un futuro se seguirá planteado posibles modificaciones, mantenimientos y actualizaciones.

7 CONCLUSIÓN

Se puede concluir que la realización e implementación del proyecto ha sido exitosa, se han cumplido todos los objetivos planteados en el principio, los requerimientos se materializaron, se cumple con el diseño propuesto, el código es funcional y las pruebas permitieron la detección de errores y mejoramiento de las diferentes partes del programa, el mantenimiento se seguirá realizado para detectar posibles errores futuros, modificación de requerimientos o adaptación a nuevos entornos

El modelo de ciclo de vida de software escogido ha dado buenos resultados con este tipo de sistema, durante el desarrollo no se presentaron iteraciones o incrementos imprevistos, la planificación y análisis permitió que se realice todo de una forma ordenada, los métodos usados para la extracción de los datos y la generación de la nube también resultaron muy convenientes e interesantes para el cumplimiento del proyecto.

8 ESTIMACIÓN CON EL MÉTODO COCOMO

Para realizar la estimación a nuestro proyecto de la Agencia de Viajes "VEM",se esta usando el modelo básico y el tipo de modo de desarrollo es el empotrado ya que se usa en proyectos bastante complejos, en los que apenas se tiene experiencia y se engloban en un entorno de gran innovación técnica. Además, se trabaja con unos requisitos muy restrictivos y de gran volatilidad.
Para calcular el Esfuerzo, necesitaremos hallar la variable KDLC (Miles de líneas de código), en el código a presentar tenemos 495 KDLC.
FÓRMULAS
E = Esfuerzo = a KLDC $^{b}(personaxmes)$
$T = Tiempo\ \ de\ \ duración\ \ del\ \ desarrollo = c * Esfuerzo^{d}(meses)$
$P = Personal = E/T(personas)$

	Básico	
Modo	a_i	b_i
Orgánico	2.4	1.05
Semiencajado	3.0	1.12
Empotrado	3.6	1.2

LINEAS DE CÓDIGO

KLDC = 495

E = $3.6(495)^{1.20}$

$E = 6164 \ \ PERSONAS/MES$

T = 2.5*$6164^{0.32}$

$T = 22 \ \ MESES$

P= 6164/22

P= 281 PERSONAS

ESTIMACIÓN DE COSTO

- Sueldo de los desarrolladores = 400
- Otros costos del proyecto = 15.000
- Costos = (E*P*sueldo de los desarrolladores)+otros costos
- C= (6164*281*400)+15000
- C= 692.848.600