UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LENGUAJES FORMALES Y DE PROGRAMACIÓN

MANUAL TÉCNICO

PRACTICA 1

Allan Josué Rafael Morales 201709196

AGOSTO/2022

REQUISITOS MINIMOS DEL SISTEMA

- SO: Windows Vista/7/8/8.1/10.
- Procesador: 2.0 GHz o mejor.
- Memoria: 1 GB de RAM.
- Gráficos: 1280×720 de resolución mínima, 512 MB video RAM recomendada.
- DirectX: Versión 11.
- Almacenamiento: 250 MB de espacio disponible.

APLICACIÓN PROGRAMADA EN PYTHON

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

IDE UTILIZADA

Visual Studio Code proporciona una finalización del código inteligente, inspecciones del código, indicación de errores sobre la marcha y arreglos rápidos, así como refactorización de código automática y completas funcionalidades de navegación.

LIBRERIAS IMPORTADAS

```
from tkinter import *
from tkinter import filedialog, ttk
import sys
```

LÓGICA DEL PROGRAMA

La creación base del programa se realizó con la librería tkinter, para hacer una aplicación visible, sencilla e intuitiva para los usuarios, las mayormente herramientas utilizadas fueron 'Label' y 'Botton', entre algunas otras para crear una presentable interfaz.

```
def __init__(self):
    self.window = Tk()
    self.window.title('Practica 1 LFP B+')
    self.center(self.window, 600, 500)
    self.window.resizable(False,False)
    self.lista_global = []
    self.Window()
```

Al presentarse el menú, lo que sigue es abrir un filedialog, en la que el usuario seleccionará el archivo. lfp de manera exclusiva con los datos que harán funcionar de manera correcta las demás opciones. Aquí es cuando utilizamos las librerías importadas:

```
def cargarDoc(self):
    global listaGlobal
    try:
        archivo = filedialog.askopenfilename(title = "Seleccionar archivo LFP", filetypes=[("LFP files", "*.lfp")])
        lista = analizador.selecDoc(archivo)
```

Al instante de abrir dicho archivo, se procederá a leer el contenido de manera conjunta con el método selecDoc (archivo):

```
def selecDoc(archivo):
    datos = open(archivo, encoding='utf-8')
    lineas = datos.readlines()
    datos.close()

    curs = []
    for linea in lineas:
        data = linea.split(',')
        curso = cursos(data[0], data[1], data[2], data[3], data[4], data[5], data[6])
        curs.append(curso)
    return curs
```

DESPUES DE LEER EL ARCHIVO…

Se crearon varias opciones en las cuales se maneja una base de datos que contiene objetos cuyo contenido son listas de datos, estos se importan y se manejaron con la variable listaGlobal[]:

```
def viewCurso():
    temporal = False
    for i in range(len(listaGlobal)):
        if codigoEntrada.get() == listaGlobal[i].getCodigo():

            nombre = StringVar()
            prereq = StringVar()
            semestre = StringVar()
            opcional = StringVar()
            creditos = StringVar()
            estado = StringVar()
```

Se crearon las ventanas necesarias para conseguir una estética personalmente agradable y dentro de cada clase las funciones de cada una de estas:

```
def editCurso():

    temporal = False
    for i in range(len(listaGlobal)):

        if codigo_entry.get() == listaGlobal[i].getCodigo():

            listaGlobal[i].nombre = nombre_entry.get()
            listaGlobal[i].prerequisito = prereq_entry.get()
            listaGlobal[i].semestre = semestre_entry.get()
            listaGlobal[i].creditos = credito_entry.get()
```

CARGA MASIVA

La clase que funciona como almacenamiento de los datos que se manejaron esta en otro archivo Python, por medio de esta se crearon los métodos Set y Get:

```
class cursos():

    def __init__(self, codigo, nombre, prerequisito, obligatorio, semestre, creditos, estado):
        self.codigo = codigo
        self.nombre = nombre
        self.prerequisito = prerequisito
        self.obligatorio = obligatorio
        self.semestre = semestre
        self.creditos = creditos
        self.estado = estado
```

PARADIGMAS

De los paradigmas utilizados para realizar la aplicación esta principalmente el paradigma orientado a objetos, una manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones. La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.