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

Manual Técnico

Nombre: Elvis Joseph Vásquez Villatoro
Carné: 202006666

Manual Técnico

Para la elaboración del software fue utilizada la Programación Orientada a Objetos (POO), implementando clases en las cuales se almacenarían los distintos datos de los cursos o bien widgets de las distintas interfaces graficas utilizadas las cuales serian implementadas en la creación de las ventanas del programa. También se utilizaron distintas funciones y métodos que sirven para la complementación de el software.

Clase Usuario

Esta clase es utilizada para almacenar todos los atributos de los cursos que serán almacenados, también se crearon los métodos set y get correspondientes a cada atributo de la clase Curso que nos permitirán obtener o modificar los atributos deseados.

```python
class Curso:

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


    def getCodigo(self):
        return self.codigo

    def getNombre(self):
        return self.nombre

    def getPrerrequisitos(self):
        return self.prerrequisitos

    def getObligatorio(self):
        return self.obligatorio

    def getSemestre(self):
        return self.semestre

    def getCreditos(self):
        return self.creditos

    def getEstado(self):
        return self.estado
```

```python
def setNombre(self, nombre):
    self.nombre = nombre

def setPrerrequisitos(self, prerrequisitos):
    self.prerrequisitos = prerrequisitos

def setObligatorio(self, obligatorio):
    self.obligatorio = obligatorio

def setSemestre(self, semestre):
    self.semestre = semestre

def setCodigo(self, codigo):
    self.codigo = codigo

def setCreditos(self, creditos):
    self.creditos = creditos

def setEstado(self, estado):
    self.estado = estado
```

Función Lectura

Esta función nos permitirá leer el archivo .LFP línea por línea y almacenar todos los datos de los cursos de forma masiva en una lista de objetos creada a partir de la clase Curso con sus atributos correspondientes. Cada línea del archivo contendrá los datos de los cursos separados por comas. Estos datos serán almacenados gracias al método Split que viene por defecto en Python, el cual hará que se separen los atributos de los cursos por medio de la coma.

```python
def Lectura(ruta):
    try:
        objeto = open(ruta,'r+',encoding='utf-8')
        lineas = objeto.readlines()
        objeto.close()
    except:
        titulo = "Error"
        mensaje = 'Ha ocurrido algún error con el archivo.\n Intenta ingresar otro archivo'
        messagebox.showerror(titulo, mensaje)

    for linea in lineas:
        pos=-1
        try:

            data = linea.split(',') # Devuelve una lista
            curso = Curso(data[0], data[1], data[2], data[3], data[4],data[5],data[6].rstrip('\n'))
        except:
            continue
        # cursos.append(curso)
        for dato in cursos:
            if curso.getCodigo()==dato.getCodigo():
                pos = cursos.index(dato)
        if pos!=-1:
            cursos.pop(pos)
        cursos.append(curso)
    titulo = "Agregar Cursos"
    mensaje = 'Se han agregado los cursos'
    messagebox.showinfo(titulo, mensaje)


    return cursos
```

Clases Para la creación de Frames

Para la creación de Frames que contendrán distintos widgets se crearon clases correspondientes a cada ventana del software las cuales serian clases hijas de la clase tk.Frame, en estas se implementarían un método base llamado “ Ventana “ que contendría todo lo que llevaría visualmente el Frame. Cada clase tendría sus propios métodos que serían implementados para la interacción del usuario con los frames. Entre las clases que se crearon se encontrarían las siguientes:

- Clase FirstScreen
- Clase CargarArchivo
- Clase GestionarCursos
- Clase AgregarCurso
- Clase MostrarCurso
- Clase EliminarCurso
- Clase EditarCurso
- Clase ListarCurso
- Clase Conteo Creditos

```python
class FirstScreen(tk.Frame):
    def __init__(self, root = None):
        super().__init__(root, width=700, height=800)
        self.root = root
        self.pack(padx=30,pady=20)
        self.config( bg='#a6bdae')

        self.Ventana()

    def Ventana(self):
        self.label_curso= tk.Label(self,text="Lab. Lenguajes Formales y de Programación B+")
        self.label_curso.config( font=('Times New Roman',13), fg='white', bg='#00251a', width=50)
        self.label_curso.place(x=25,y=50)

        self.label_curso= tk.Label(self,text="Elvis Joseph Vásquez Villatoro")
        self.label_curso.config( font=('Times New Roman',13), fg='white', bg='#00251a', width=50)
        self.label_curso.place(x=25,y=100)

        self.label_curso= tk.Label(self,text="Carné: 202006666")
        self.label_curso.config( font=('Times New Roman',13), fg='white', bg='#00251a', width=50)
        self.label_curso.place(x=25,y=150)




        Button(self, text="Cargar Archivo", command=self.cargar_archivo , font=('Times New Roman',15), fg='#000000', bg='#c7cc00', width=20, cursor='hand2' ).place
        (x=180, y=275)
        Button(self, text="Gestionar Cursos",command=self.ir_gestionar , font=('Times New Roman',15), fg='#000000', bg='#c7cc00', width=20, cursor='hand2' ).place
        (x=180, y=350)
        Button(self, text="Conteo de Créditos" ,command=self.ir_conteo, font=('Times New Roman',15), fg='#000000', bg='#c7cc00', width=20, cursor='hand2' ).place(x=180,
        y=425)
        Button(self, text="Salir", command=self.salir , font=('Times New Roman',15), fg='#000000', bg='#c7cc00', width=20, cursor='hand2' ).place(x=180, y=500)
```

Métodos para la creación de Ventanas a partir de Clases

Se creo un método para cada clase hija de tk.Frame, este método nos permitirá crear la Ventana correspondiente y en ella agregarle distintas configuraciones como el fondo, posición y colocación del Frame correspondiente junto a sus widgets en cada Ventana.

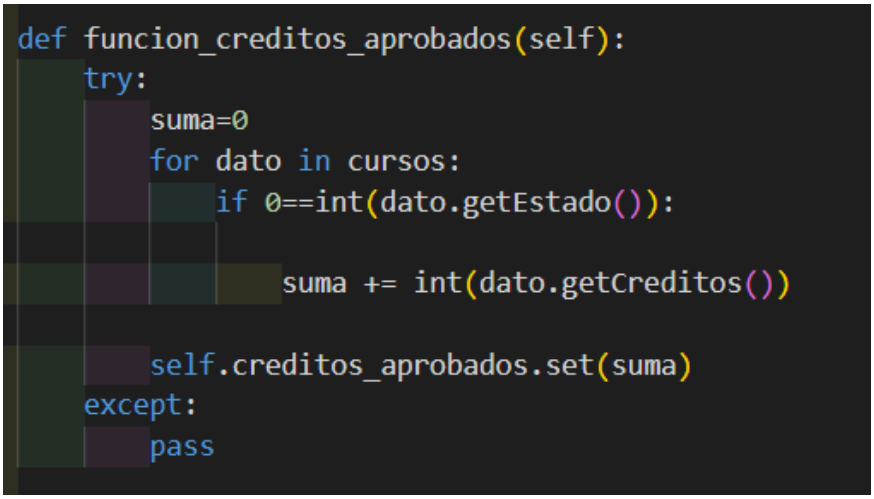
```python
# Main Screen
def ventana_principal():
    root= tk.Tk()
    root.title('Ventana Principal')
    root.geometry('700x800+600+120')
    root.configure(bg='#20001a')
    root.resizable(0,0)
    app = FirstScreen(root = root)

    app.mainloop()
```

Métodos Extras

Dependiendo de cada clase se crearon unos métodos únicos, algunos de ellos son:

- Método función_creditos_aprobados

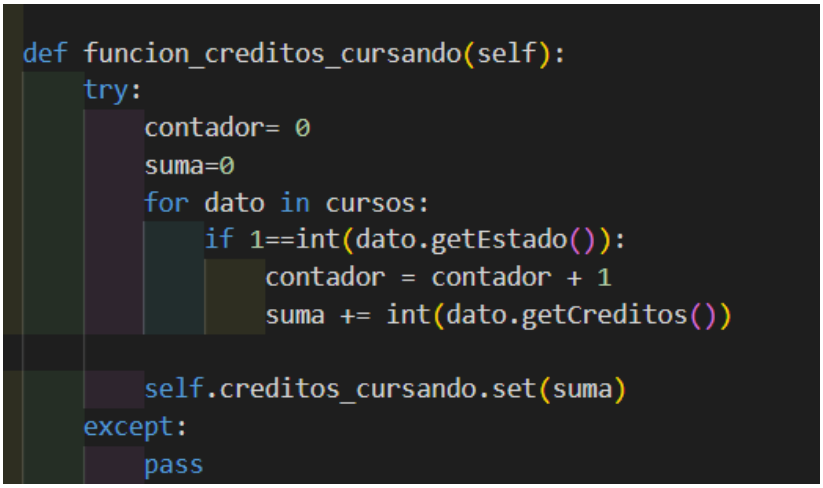
El cual hace una suma de los créditos de todos los cursos aprobados.



```
def funcion_creditos_aprobados(self):
    try:
        suma=0
        for dato in cursos:
            if 0==int(dato.getEstado()):

                suma += int(dato.getCreditos())

        self.creditos_aprobados.set(suma)
    except:
        pass
```

- Método funcion_creditos_cursando

Este método realiza una suma de los créditos de todos los cursos los cuales se encuentran cursando



```
def funcion_creditos_cursando(self):
    try:
        contador= 0
        suma=0
        for dato in cursos:
            if 1==int(dato.getEstado()):
                contador = contador + 1
                suma += int(dato.getCreditos())

        self.creditos_cursando.set(suma)
    except:
        pass
```

- Método función_creditos_hastaN

En este método se realiza una comprobación de todos los cursos y si el el curso es obligatorio y se encuentra dentro del rango del semestre seleccionado por el usuario se sumaran los créditos correspondientes del curso hasta culminar la suma total de los cursos.

```
def funcion_creditos_hastaN(self):

    try:
        n=self.str_creditos_n.get()

        suma=0
        for dato in cursos:
            if  int(dato.getObligatorio())==1 and int(dato.getSemestre())<=int(n):

                suma += int(dato.getCreditos())

        self.Str_creditos_hasta.set(suma)
    except:
        pass
```

- Método funcion_creditos_semestre

En este metodo se ralizara la suma total de los cursos aprobados, pendientes y asignados de un único semestre el cual es elegido por el usuario.

```
def funcion_creditos_semestre(self):

    try:
        semestre=self.str_creditos_del_semestre.get()

        suma_aprobados=0
        suma_pendientes=0
        suma_asignados=0
        for dato in cursos:
            if  int(dato.getEstado())==0 and int(dato.getSemestre())==int(semestre):
                suma_aprobados += int(dato.getCreditos())
            elif  int(dato.getEstado())==-1 and int(dato.getSemestre())==int(semestre):
                suma_pendientes += int(dato.getCreditos())
            elif  int(dato.getEstado())==1 and int(dato.getSemestre())==int(semestre):
                suma_asignados += int(dato.getCreditos())

        self.creditos_aprobados2.set(suma_aprobados)
        self.creditos_cursando2.set(suma_asignados)
        self.creditos_pendientes2.set(suma_pendientes)
    except:
        pass
```