

MANUAL DE PROGRAMADOR

Tabla de Contenido

Introducción..... 3

Información
destacada..... 3

Objetivos..... 3

1.Requerimientos..... 4

2.Instalación y
Configuración..... 5

Introducción

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Información destacada

El manual técnico hace referencia a la información necesaria con el fin de orientar al personal en la concepción, planteamiento análisis programación e instalación del sistema. Es de notar que la redacción propia del manual técnico está orientada a personal con conocimientos en sistemas y tecnologías de información, conocimientos de programación avanzada sobre entorno web, administración de bases de datos, responsables del mantenimiento e instalación del sistema en los servidores.

Objetivos

Instruir el uso adecuado del Sistema de Información, para el acceso oportuno y adecuado en la instalación del mismo, mostrando los pasos a seguir en el proceso de instalación, así como la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración y soporte del mismo.

1.Requerimientos

El sistema puede ser instalado en cualquier sistema operativo que cumpla con los siguientes requerimientos:

- MS Windows 10
- Python 3.10 o más reciente
- Navegador Web
- MS Windows 10
- Visual Studio Code
- .Net Framework

Tecnica

Treeview

El widget `ttk.Treeview` muestra una colección en árbol de elementos. Cada elemento tiene una etiqueta textual, una imagen opcional y una lista opcional de valores de datos. Los valores de datos se muestran en columnas sucesivas después de la etiqueta de árbol.

El orden en que se muestran los valores de datos se puede controlar estableciendo la opción de widget `displaycolumns`. El Treeview también puede mostrar encabezados. Se puede acceder a las columnas por número o nombres simbólicos enumerados en las columnas de opciones del widget. Consulte [Column Identifiers](#).

Cada elemento se identifica con un nombre único. El widget genera IDs para los elementos si no se proporcionan en la declaración. Hay un elemento raíz distinguido, denominado `{}`. El elemento raíz en sí no se muestra; sus hijos aparecen en el nivel superior de la jerarquía.

Cada elemento también tiene una lista de etiquetas que se pueden usar para asociar enlaces de eventos con elementos individuales y controlar la apariencia del elemento.

El widget Treeview admite el deslizamiento horizontal y vertical, según las opciones descritas en [Scrollable Widget Options](#) y los métodos `Treeview.xview()` y `Treeview.yview()`. (*Tkinter.ttk — Tk Widgets Temáticos — Documentación De Python - 3.10.6, n.d.*)

Metodos Principales

abrirArchivo()

```
Practica.py > ...
1 from cgi import test
2 from cgitb import text
3 from distutils.cmd import Command
4 from distutils.command.build import build
5 import tkinter
6 from tkinter import filedialog
7 from turtle import bgcolor, color
8 from datetime import date, datetime
9 from tkinter import *
10 from Sobrio_2 import vistaPrevia
11 from Creditos import Creditos
12 import shutil
13
14
15 now = datetime.now()
16
17 data = []
18
19 global ruta
20 ruta = []
21
22
23
24 def abrirArchivo():
25     ruta = filedialog.askopenfilename(title = "Open File", filetypes=(("csv files", "*.csv"),("all files", "*.*")))
26     data = open(ruta, 'r', encoding="utf-8")
27     print(data.read())
28     #archivo.close()
29     print(ruta)
30     shutil.copy(ruta, "F:\Python\Desarrollo\Practica 1 (beta)\lista.csv")
31
32
33
34 ventana = tkinter.Tk()
35 ventana.title('USAC')
36 ventana.geometry("500x300")
37 ventana['bg'] = '#74a5d6'
38 ventana.iconbitmap("C:\\Users\\carlo\\Documents\\Usac\\Iconos\\Usac_logo.ico")
39
40
41 # Cours Titles
42 img = tkinter.PhotoImage(file = "C:\\Users\\carlo\\Documents\\Usac\\Iconos\\open-folder2.png")
43 lbl_img = tkinter.Label(ventana, image = img)
44 lbl_img.place(x = 45, y = 74)
45
46 img2 = tkinter.PhotoImage(file = "C:\\Users\\carlo\\Documents\\Usac\\Iconos\\folder2.png")
47 lbl_img2 = tkinter.Label(ventana, image = img2)
48 lbl_img2.place(x = 45, y = 121)
49
50 img3 = tkinter.PhotoImage(file = "C:\\Users\\carlo\\Documents\\Usac\\Iconos\\visits2.png")
51 lbl3_img = tkinter.Label(ventana, image = img3)
52 lbl3_img.place(x = 45, y = 171)
53
54
55 # Bottons
56 boton1 = tkinter.Button(ventana, text = "Open File", padx = 30, pady= 10, command = abrirArchivo)
57 boton1.place(x = 100, y = 70)
58
59 boton2 = tkinter.Button(ventana, text = "Course Management", padx = 30, pady= 10, command = vistaPrevia)
60 boton2.place(x = 100, y = 120)
61
62 boton3 = tkinter.Button(ventana, text = "Credit Count", padx = 30, pady= 10, command = Creditos)
63 boton3.place(x = 100, y = 170)
64
65 boton4 = tkinter.Button(ventana, text = "Exit", padx = 30, pady= 10, command = ventana.destroy)
66 boton4.place(x = 100, y = 220)
67
68
69 etiqueta2 = tkinter.Label(ventana, text = "Lenguajes Formales y Programacion A+", font = "technical 14", bg = "white", relief=RAISED)
70 etiqueta2.place(x = 100, y = 10)
71
72 etiqueta3 = tkinter.Label(ventana, text = "Carlos Hugo Rios Mancilla", font = "technical 10", bg = "white", relief=RAISED )
73 etiqueta3.place(x = 347, y = 230)
74
75 etiqueta4 = tkinter.Label(ventana, text = "9520488", font = "technical 10", bg = "white", relief=RAISED )
76 etiqueta4.place(x = 430, y = 260)
77
78
79 var = StringVar()
80 var.set(now)
81 label = Label( ventana, textvariable = var, bg = '#74a5d6')
82 label.place(x=10,y=280)
83
84 print(type(ruta))
85
86
87 ventana.mainloop()
```

vistaPrevia()

```
12 def vistaPrevia():
13
14     with open("F:\\Python\\Desarrollo\\Practica 1 (beta)\\lista.csv", encoding="utf8") as csvfile:
15         data = csv.reader(csvfile, delimiter=",")
16
17
18
19         root = Tk()
20         root.title('USAC')
21         root.iconbitmap('C:\\Users\\carlo\\Documents\\Usac\\Iconos\\Usac_logo.ico')
22         root.geometry("900x800")
23         root['bg'] = '#74a5d6'
24
25         # Add some style
26         style = ttk.Style()
27         style.configure("Treeview",)
28
29         my_tree = ttk.Treeview(root)
30
31         # Define Columns
32         my_tree['columns'] = ("Codigo", "Nombre", "Prerrequisito", "Obligatorio", "Semestre", "Creditos", "Estado")
33
34         # Format Columns
35         my_tree.column("#0", width = 40, minwidth = 10)
36         my_tree.column("Codigo", anchor = CENTER, width = 80)
37         my_tree.column("Nombre", anchor = CENTER, width = 150)
38         my_tree.column("Prerrequisito", anchor = CENTER, width = 80)
39         my_tree.column("Obligatorio", anchor = CENTER, width = 80)
40         my_tree.column("Semestre", anchor = CENTER, width = 80)
41         my_tree.column("Creditos", anchor = CENTER, width = 80)
42         my_tree.column("Estado", anchor = CENTER, width = 80)
43
44         # Create Headings
45         my_tree.heading("#0", text = "No.", anchor = W)
46         my_tree.heading("Codigo", text = "Codigo", anchor = CENTER)
47         my_tree.heading("Nombre", text = "Nombre", anchor = CENTER)
48         my_tree.heading("Prerrequisito", text = "Prerrequisito", anchor = CENTER)
49         my_tree.heading("Obligatorio", text = "Obligatorio", anchor = CENTER)
50         my_tree.heading("Semestre", text = "Semestre", anchor = CENTER)
51         my_tree.heading("Creditos", text = "Creditos", anchor = CENTER)
52         my_tree.heading("Estado", text = "Estado", anchor = CENTER)
53
54         # Add Data
55
56         global count
57         count = 0
58         for record in data:
59             my_tree.insert(parent='', index='end', iid=count, text= count, values=record)
60             count += 1
61
62         # Pack to the screen
63         my_tree.pack(pady=20)
64
65         add_frame = Frame(root)
66         add_frame.pack(pady=20)
67
68         # Labels
69         c1 = Label(add_frame, text = "Codigo")
70         c1.grid(row=0, column=0)
71
72         n1 = Label(add_frame, text = "Nombre")
73         n1.grid(row=0, column=1)
74
75         p1 = Label(add_frame, text = "Prerrequisito")
76         p1.grid(row=0, column=2)
77
78         o1 = Label(add_frame, text = "Obligatorio")
79         o1.grid(row=0, column=3)
80
81         s1 = Label(add_frame, text = "Semestre")
82         s1.grid(row=0, column=4)
83
84         cl = Label(add_frame, text = "Creditos")
85         cl.grid(row=0, column=5)
86
87         el = Label(add_frame, text = "Estado")
88         el.grid(row=0, column=6)
89
90
91         # Entry boxes
92         cod_box = Entry(add_frame)
93         cod_box.grid(row=1, column=0)
94
95         nom_box = Entry(add_frame)
96         nom_box.grid(row=1, column=1)
97
98         pre_box = Entry(add_frame)
99         pre_box.grid(row=1, column=2)
100
101         ob_box = Entry(add_frame)
102         ob_box.grid(row=1, column=3)
103
104         sem_box = Entry(add_frame)
105         sem_box.grid(row=1, column=4)
106
107         cred_box = Entry(add_frame)
108         cred_box.grid(row=1, column=5)
109
110         est_box = Entry(add_frame)
111         est_box.grid(row=1, column=6)
112
113
```

```

114
115 # Add Record
116 def add_record():
117     global count
118
119     my_tree.insert(parent='', index='end', iid=count, text="", values=(cod_box.get(), nom_box.get(), pre_box.get(), ob_box.get(), sem_box.get(), cred_box.get(), est_box.get()))
120
121     count += 1
122
123     # Clear the boxes
124     cod_box.delete(0,END)
125     nom_box.delete(0,END)
126     pre_box.delete(0,END)
127     ob_box.delete(0,END)
128     sem_box.delete(0,END)
129     cred_box.delete(0,END)
130     est_box.delete(0,END)
131
132 # Remove all records
133 def remove_all():
134     my_tree.get_children()
135     for record in my_tree.get_children():
136         my_tree.delete(record)
137
138 # Remove one selected
139 def remove_one():
140     x = my_tree.selection()[0]
141     my_tree.delete(x)
142
143 # Remove many selected
144 def remove_many():
145     x = my_tree.selection()
146     for record in x:
147         my_tree.delete(record)
148
149 # Select Record
150 def select_record():
151     # Clear entry boxes
152     cod_box.delete(0, END)
153     nom_box.delete(0, END)
154     pre_box.delete(0, END)
155     ob_box.delete(0, END)
156     sem_box.delete(0, END)
157     cred_box.delete(0, END)
158     est_box.delete(0, END)
159
160     # Grab Record Number
161     selected = my_tree.focus()
162
163     # Save new data
164
165     # Grab Record Values
166     values = my_tree.item(selected, 'values')
167
168     #temp_label.config(text=values)
169
170     # Output to entry boxes
171     cod_box.insert(0, values[0])
172     nom_box.insert(0, values[1])
173     pre_box.insert(0, values[2])
174     ob_box.insert(0, values[3])
175     sem_box.insert(0, values[4])
176     cred_box.insert(0, values[5])
177     est_box.insert(0, values[6])
178
179 # Save Update Record
180 def update_record():
181     # Grabe record number
182     selected = my_tree.focus()
183
184     # Save new data
185     my_tree.item(selected, text="", values=(cod_box.get(), nom_box.get(), pre_box.get(), ob_box.get(), sem_box.get(), cred_box.get(), est_box.get()))
186
187     cod_box.delete(0, END)
188     nom_box.delete(0, END)
189     pre_box.delete(0, END)
190     ob_box.delete(0, END)
191     sem_box.delete(0, END)
192     cred_box.delete(0, END)
193     est_box.delete(0, END)
194
195 # Buttons
196 select_button = Button(root, text="Select Record", command=select_record)
197 select_button.pack(pady=20)
198
199 update_button = Button(root, text="Save Record", command=update_record)
200 update_button.pack(pady=20)
201
202 add_record = Button(root, text="Add Record", command=add_record)
203 add_record.pack(pady=10)
204
205 # Remove all
206 remove_all = Button(root, text="Remove All Records", command=remove_all)
207 remove_all.pack(pady=10)
208
209 # Remove One
210 remove_one = Button(root, text="Remove One Selected", command=remove_one)
211 remove_one.pack(pady=10)
212
213 # Remove Many
214 remove_many = Button(root, text="Remove Many Selected", command=remove_many)
215 remove_many.pack(pady=10)
216
217 temp_label = Label(root, text="")
218 temp_label.pack(pady=20)
219
220 # Bindings
221 my_tree.bind()
222
223 root.mainloop()

```


creditos()

```
5
6 def Creditos():
7
8     with open("F:\\Python\\Desarrollo\\Practica 1 (beta)\\lista.csv", encoding="utf8") as csvfile:
9         data = csv.reader(csvfile, delimiter=",")
10
11
12
13     print(data)
14
15     root1 = Tk()
16     root1.title('USAC')
17     root1.iconbitmap('C:\\Users\\carlo\\Documents\\Usac\\Iconos\\Usac_logo.ico')
18     root1.geometry("500x300")
19     root1['bg'] = '#74a5d6'
20
21     # Add some style
22     style = ttk.Style()
23     style.configure("Treeview",)
24
25     my_tree = ttk.Treeview(root1, height=1 )
26
27     # Define Columns
28     my_tree['columns'] = ("Aprobados", "Cursados", "Pendientes")
29
30     # Format Cols
31     my_tree.column("#0", width=20, minwidth=25)
32     my_tree.column("Aprobados", anchor=CENTER, width=120)
33     my_tree.column("Cursados", anchor=CENTER, width=120)
34     my_tree.column("Pendientes", anchor=CENTER, width=120)
35
36     # Create Headings
37     my_tree.heading("#0", text="", anchor=W)
38     my_tree.heading("Aprobados", text="Aprobados", anchor=W)
39     my_tree.heading("Cursados", text="Cursados", anchor=CENTER)
40     my_tree.heading("Pendientes", text="Pendiente", anchor=W)
41
42
43
44     # Add Data
45     my_tree.insert(parent='', index='end', iid=0, text= "", values=(10, 5, 27))
46
47     # Pack to screen
48     my_tree.pack(pady=20)
49
50     print(type(data))
51
52     root1.mainloop()
53
54 #Creditos()
55
56
```

Descripcion

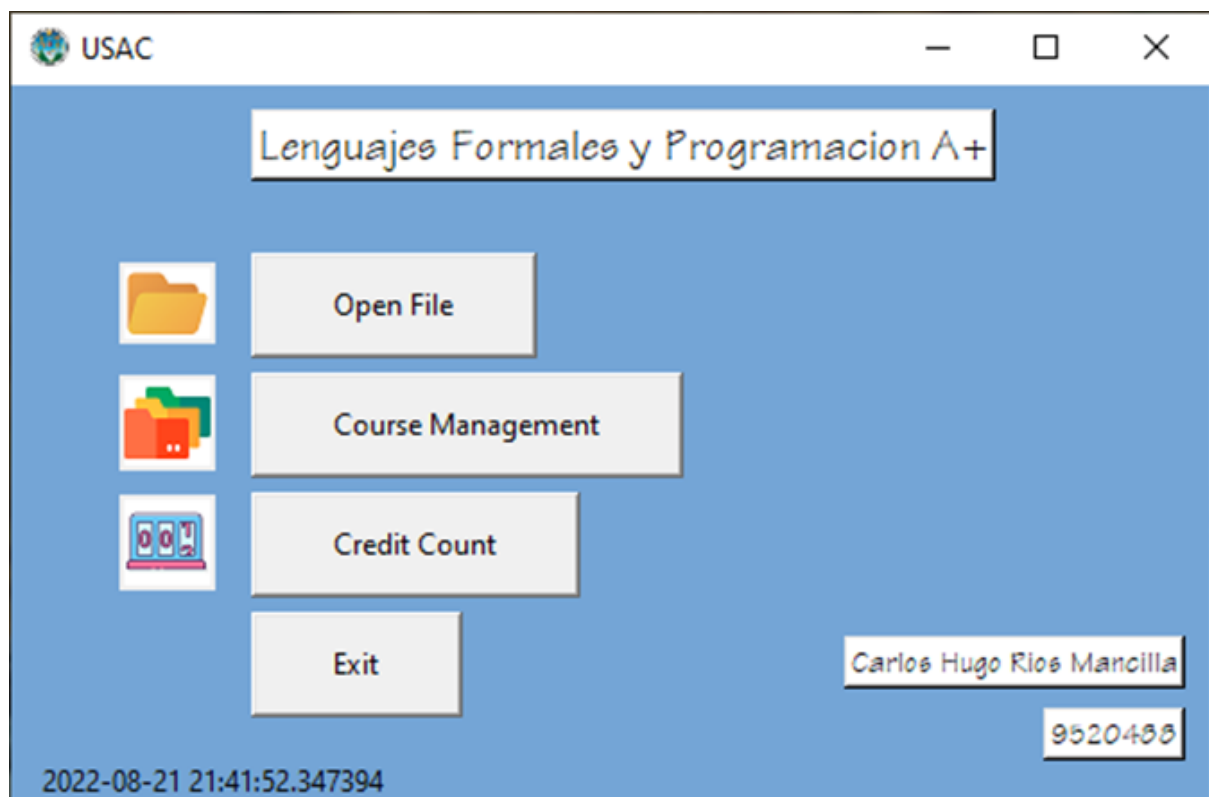
Herramientas:

Ventajas de Visual Studio Code


al Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos podemos destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros. Otra ventaja interesante es la posibilidad de configurar la vista a nuestro gusto.

Interfaces Principales

Ventana Principal



Manejo de Datos

 USAC

No.	Codigo	Nombre	Prerrequisito	Obligatorio	Semestre	Creditos	Estado
0	17	Social Humanística 1		1	1	4	0
1	101	Mate Básica 1		1	1	7	0
2	69	Técnica Complementaria 1		1	1	3	0
3	39	Deportes 1		0	1	1	0
4	348	Química General 1		1	1	3	0
5	6	Idioma Técnico 1		0	1	2	-1
6	19	Social Humanística 2	17	1	2	4	0
7	103	Mate Basica 2	101	1	2	7	0
8	5	Técnicas de estudio		1	2	2	0
9	147	Fisica B	101	1	2	3	0

Codigo	Nombre	Prerrequisito	Obligatorio	Semestre	Creditos	Estado
39	Deportes 1		0	1	1	0

Select Record

Save Record

Add Record

Remove All Records

Remove One Selected

Remove Many Selected

Creditos

 USAC

Aprobados	Cursados	Pendiente
10	5	27

Glosario

atributo

Un valor asociado a un objeto que es referenciado por el nombre usado en expresiones de punto. Por ejemplo, si un objeto *o* tiene un atributo *a* sería referenciado como *o.a*.

clase

Una plantilla para crear objetos definidos por el usuario. Las definiciones de clase normalmente contienen definiciones de métodos que operan una instancia de la clase.

CPython

La implementación canónica del lenguaje de programación Python, como se distribuye en python.org. El término «CPython» es usado cuando es necesario distinguir esta implementación de otras como *Jython* o *IronPython*.

diccionario

Un arreglo asociativo, con claves arbitrarias que son asociadas a valores. Las claves pueden ser cualquier objeto con los métodos `__hash__()` y `__eq__()`. Son llamadas hash en Perl.

Objetos tipo archivo

Un sinónimo de [file object](#).

función

Una serie de sentencias que retornan un valor al que las llama. También se le puede pasar cero o más [argumentos](#) los cuales pueden ser usados en la ejecución de la misma. Vea también [parameter](#), [method](#), y la sección [Definiciones de funciones](#).

importar

El proceso mediante el cual el código Python dentro de un módulo se hace alcanzable desde otro código Python en otro módulo.

