MANUAL TECNICO

Pensum Application:

Tabla de Contenido

Introducción

El presente documento describe los describe la seria de pasos que realiza algunos de los métodos o funciones que conforman el programa computacional (PensumApplication) por medio de fragmentos de código, como también se detalla los conocimientos previos que debe tener el lector de este manual para comprender de una mejor manera el funcionamiento de cada uno partes del código que conforman el programa.

Objetivos

Instruir el uso adecuado del programa computacional, describiendo el diseño y la lógica del programa por medio de fragmentos de códigos. Describir al usuario el funcionamiento del programa para el mejor uso de él y demostrar el proceso necesario para su ejecución.

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 Python, responsables del mantenimiento e instalación del programa computacional en el computador.

Conocimientos Previos

Los conocimientos mínimos que deben tener las personas que operarán las páginas y deberán utilizar este manual son:

- Cocimientos y entendimientos en logaritmos
- Conocimientos en Python
- Programación Orientada a Objetos
- Interfaces graficas (Tkinter)
- Conocimiento básico de Windows

1. Requerimientos

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

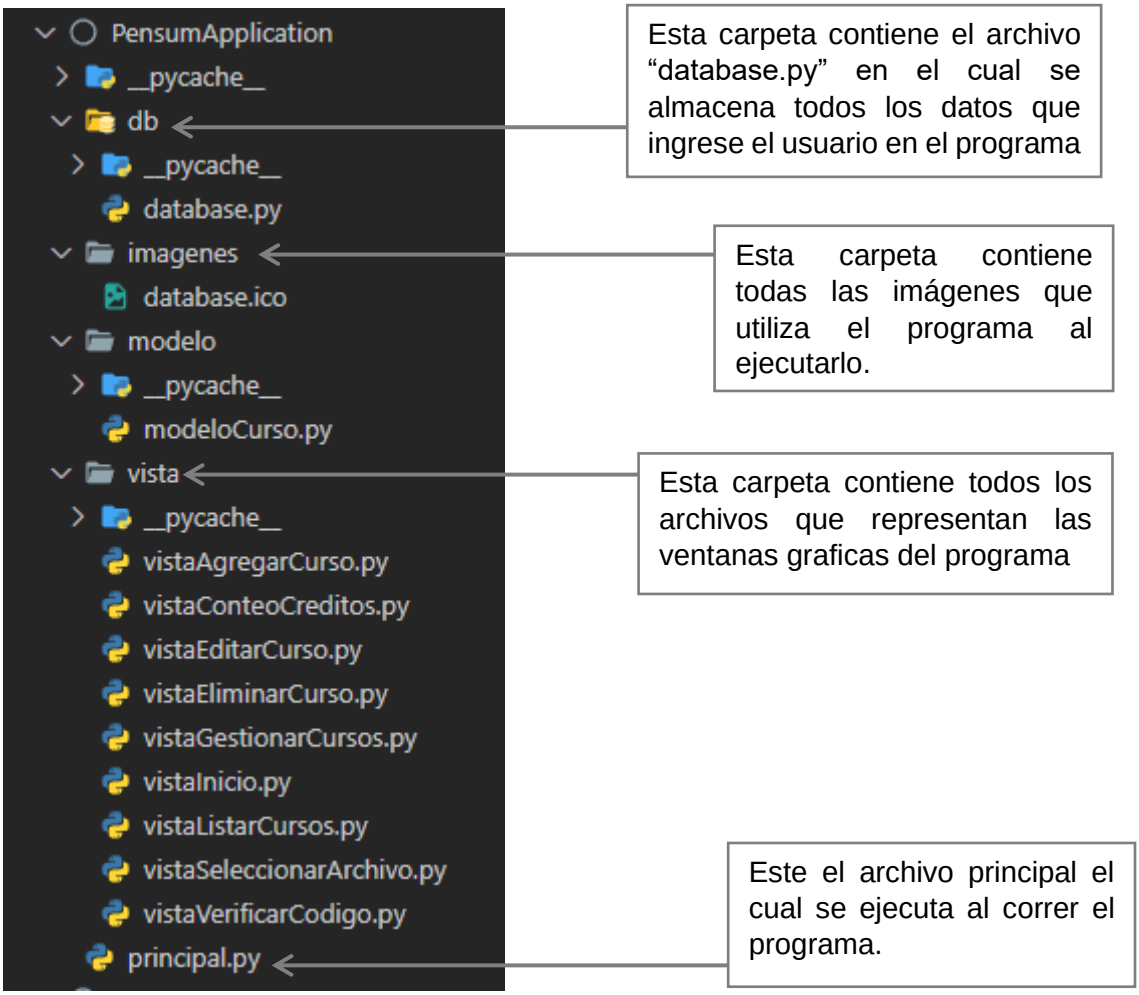
- Sistema Operativo: Windows 7 o superior
- Procesador mínimo Intel Pentium (800MHz Intel Pentium)
- Mínimo 1GB en RAM
- IDE Visual Studio Code, o compatible con Python
- Exploradores: Internet Explorer 9 y superior

2. Instalación y configuración

Para el proceso de instalación de esta aplicación únicamente es necesario tener instalado un IDE que sea compatible con el lenguaje Python para ejecutar la aplicación desde la terminal de este.

No es necesario tener alguna configuración ya que la configuración que trae por determinado el IDE es la necesaria para que el funcionamiento del programa sea posible.

3. Estructura del programa



4. Fragmentos de Códigos

En este apartado se explican detalladamente los método y funciones más importantes que conforman el código del programa. Esto con el objetivo de que la persona a usar el programa necesite dar soporte a la aplicación se le realice una manera más sencilla comprender la lógica del programa.

Código del método “cargarArchivo”:

```
def cargarArchivo(self):
    if (self.txtRuta.get() == ""):
        tkinter.messagebox.showwarning("Advertencia", "¡Complete el campo vacio!")
    else:
        try:
            archivo = open(self.txtRuta.get(), 'r+')
            info = archivo.readlines()
            info = list(map(lambda l: l.rstrip("\n"), info))
            for cadena in info:
                curso = cadena.split(",")
                if int(curso[3]) == 1:
                    curso[3] = "Obligatorio"
                else:
                    curso[3] = "Opcional"
                if int(curso[6]) == 0:
                    curso[6] = "Aprobado"
                elif int(curso[6]) == 1:
                    curso[6] = "Cursando"
                else:
                    curso[6] = "Pendiente"
                self.db.addCurso(curso[0], curso[1], curso[2], curso[3], curso[4], curso[5], curso[6])
            archivo.close()
            tkinter.messagebox.showinfo("Informacion", "¡Curso(s) agregado(s) exitosamente!")
            self.txtRuta.set("")
        except FileNotFoundError:
            tkinter.messagebox.showerror("Error", "¡El archivo no existe!")
```

El método “cargarArchivos” funciona de la siguiente manera:

Intenta abrir el archivo localizado en la ruta ingresada en el campo vacío, luego lee línea por línea e inserta cada línea en un arreglo. Luego recorre el arreglo y convierte a texto algunos de los datos contenidos en él. Continúa instanciando un objeto de la clase “Database” en la cual se almacenarán los datos extraídos del archivo en un arreglo.

Por último, cierra la lectura del archivo y muestra en pantalla un mensaje de realización exitosa.

Si en dado caso el intento al abrir el archivo fallara se muestra en pantalla un mensaje del posible error.

Código del método “addCurso”:

```
def addCurso(self, codigo, nombre, prerrequisito, opcionalidad, semestre, creditos, estado):
    nuevoCurso = Curso(codigo, nombre, prerrequisito, opcionalidad, semestre, creditos, estado)
    if (codigo in self.codigoCursos):
        self.Cursos[self.codigoCursos.index(codigo)] = nuevoCurso
    else:
        self.Cursos.append(nuevoCurso)
        self.codigoCursos.append(codigo)
```

Este método solicita como parámetro el código, nombre, prerrequisito, opcionalidad, semestre, créditos y estado del curso a agregar para luego crear un nuevo objeto de la clase “Curso” y asignarle los parámetros solicitados.

Luego verifica que el código del curso ingresado ya exista en el arreglo “codigoCursos” y de no ser así inserta este nuevo curso en el arreglo “Cursos”.

Si en caso contrario el código del curso ingresado ya existe, reemplaza los datos del curso existente por los del nuevo curso.

Código del método “updateCurso”

```
def updateCurso(self, codigo, nombre, prerrequisito, opcionalidad, semestre, creditos, estado):
    cursoEditado = Curso(codigo, nombre, prerrequisito, opcionalidad, semestre, creditos, estado)
    self.Cursos[self.codigoCursos.index(codigo)] = cursoEditado
    self.codigoCursos[self.codigoCursos.index(codigo)] = codigo
```

Este método solicita como parámetro el código, nombre, prerrequisito, opcionalidad, semestre, créditos y estado del curso a actualizar para luego crear un nuevo objeto de la clase “Curso” y asignarle los parámetros solicitados.

Luego en el arreglo “Cursos” en la posición del curso ya existente asigna el nuevo curso creado, de manera que reemplaza el curso que se encontraba asignado anteriormente.

Al igual que en el arreglo “Cursos” reemplaza el código del curso ya existente por el del nuevo curso en el arreglo “codigoCursos”.

Código del método “deleteCurso”:

```
def deleteCurso(self, codigo):
    self.Cursos.pop(self.codigoCursos.index(codigo))
    self.codigoCursos.remove(codigo)
```

El método “deleteCurso” elimina un curso del arreglo “Cursos” y “codigoCursos” por medio del código del curso ingresado como parámetro de dicho método.

Código del método “sumatoriaCreditos”:

```
def sumatoriaCreditos(self):
    self.creditosAprobados = 0
    self.creditosCursando = 0
    self.creditosPendientes = 0
    for curso in self.Cursos:
        if curso.getEstado() == "Aprobado":
            self.creditosAprobados += int(curso.getCreditos())
        elif curso.getEstado() == "Cursando":
            self.creditosCursando += int(curso.getCreditos())
        elif curso.getEstado() == "Pendiente":
            if curso.getOpcionalidad() == "Obligatorio":
                self.creditosPendientes += int(curso.getCreditos())
```

Este método es el encargado de realizar la suma total de los créditos que pertenecen a los cursos aprobados, cursando y pendientes. La manera de funcionar de este método es la siguiente:

Un ciclo for se encarga de recorrer el arreglo “Cursos” y mientras lo recorre verifica la cantidad de créditos que le pertenece a dicho curso para sumarla a la variable correspondiente.

Código del método “sumatoriaCreditosSemestre”:

```
def sumatoriaCreditosSemestre(self, semestre):
    self.creditosSemestreAprobados = 0
    self.creditosSemestrePendientes = 0
    for curso in self.Cursos:
        if int(curso.getSemestre()) == semestre and (curso.getEstado() == "Aprobado"):
            self.creditosSemestreAprobados += int(curso.getCreditos())
        elif int(curso.getSemestre()) == semestre and (curso.getEstado() == "Pendiente"):
            self.creditosSemestrePendientes += int(curso.getCreditos())
```

Este método solicita como parámetro el número de semestre del curso al que se le realizara la sumatoria de créditos. Luego por medio de un ciclo for recorre el arreglo donde se encuentran almacenados todos los cursos agregados y verifica uno por uno que cumpla con las condiciones puestas.