Proyecto Programación

Calculadora de Nómina

Documentación

2024

Roles de los Integrantes

- **Thomas Ayala**: Gerente.
- **Felipe Sierra**: Diseño.
- **Valentina Mesa**: Documentación y pruebas.

Contents

1 Introducción

El proyecto **Calculadora de Nómina** es una herramienta tecnológica backend diseñada para emprendimientos que necesiten gestionar nóminas de manera eficiente. Este sistema permite:

- Ingresar datos de empleados y calcular sus salarios.
- Manejar la gestión de contraseñas para garantizar la seguridad.
- Registrar datos en un archivo CSV para futuras consultas.
- Ofrecer una interfaz gráfica basada en `pygame`.

El código del proyecto está alojado en el repositorio oficial: https://github.com/Vale0911/proyecto_final.

---

2 Requisitos

- **Sistema operativo**: Windows, MacOS o Linux.
- **Librerías necesarias**:
  - `pygame`
  - `flask`
- **Resolución recomendada para Pygame**: 800x600 px.
- **Instalación**: Asegúrate de ejecutar:

```
pip install -r requirements.txt
```

---

3 Manual de Usuario

1. Se debe tener instalado Python
2. Copiar el link de la terminal.
3. Abre un navegador y accede a http://localhost:5000. (Ejemplo de como se ve.)
4. Ingresa los datos requeridos: (Pygame)
   - **ID del empleado**.
   - **Horas trabajadas**.
   - **Horas extras**.
5. Haz clic en "Calcular" para obtener el salario total.
6. Los datos serán almacenados en formato CSV para futuras consultas.

---

4 Guía Interna del Código

4.1 Resumen de las funciones principales

- **Versión 2:**Mejora la interfaz utilizando Flask para conectar el backend con el frontend de HTML y CSS.
- **Versión 1:**Utiliza Pygame para la interfaz gráfica.

4.2 Estructura del código

Arquitectura general: Cliente-servidor.

Organización de los Archivos y Carpetas del Proyecto:

- **proyecto/version1/:**Contiene la implementación basada en Pygame.
- **proyecto/version2/:**Contiene la implementación mejorada con Flask.

Archivos principales:

- **proyecto/version1/Backend/calc.py:** Código de la versión 1.
- **proyecto/version2/main.py:** Código de la versión 2.
- **proyecto/version2/templates/inicio.html:**Página de inicio.
- **proyecto/version2/templates/login.html:** Página de login.

4.3 Tecnologías utilizadas

1. **Lenguajes de Programación:**
   - `Python`
   - `HTML`
   - `CSS`
2. **Frameworks y Librerías:**
   - **Versión 1:**`Pygame`
   - **Versión 2:**`Flask`
3. **Requisitos del Entorno:**
   - `Python 3.x`
   - `Flask`
   - `Pygame`

4.4 Componentes clave

BaseDeDatos: Clase para manejar la base de datos de usuarios. Este módulo gestiona las operaciones básicas de almacenamiento, recuperación y validación de usuarios mediante un archivo JSON. Permite agregar nuevos usuarios y validar credenciales existentes.

```
class BaseDeDatos:
    def __init__(self):
        self.archivo = 'usuarios.json'
        self.usuarios = self.cargar_usuarios()

    def cargar_usuarios(self):
        if os.path.exists(self.archivo):
            with open(self.archivo, 'r') as f:
                return json.load(f)
        return {}

    def guardar_usuarios(self):
        with open(self.archivo, 'w') as f:
            json.dump(self.usuarios, f)

    def agregar_usuario(self, usuario, contrasena):
        if usuario in self.usuarios:
            return False
        self.usuarios[usuario] = contrasena
        self.guardar_usuarios()
        return True

    def validar_usuario(self, usuario, contrasena):
        return self.usuarios.get(usuario) == contrasena
```

Listing 1: Clase BaseDeDatos

main.py: Archivo principal que inicializa y ejecuta la aplicación Flask. Este archivo gestiona las rutas y las vistas principales de la aplicación, sirviendo como punto de entrada para el servidor Flask. Permite renderizar las páginas de inicio y login.

```
from flask import Flask, render_template, request, redirect,
   url_for, session

app = Flask(__name__)

@app.route("/")
def inicio():
    return render_template("inicio.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    # L gica de autenticaci n
    pass

if __name__ == '__main__':
    app.run(debug=True)
```

Listing 2: Archivo main.py

4.5 Interacciones Entre Módulos

- En la versión 2, **Flask** actúa como intermediario entre el usuario y los componentes del backend.
- La clase `BaseDeDatos` se utiliza para manejar la lógica de persistencia de datos, mientras que las plantillas HTML renderizan el contenido dinámico para el usuario.

4.6 Funciones Representativas de la Interfaz Pygame

En la **versión 1**, se emplean funciones para manejar eventos y presentar botones en pantalla. Ejemplo de la página de inicio:

```
def manejar_pagina_inicio():
    pantalla.fill(BLANCO)
    mostrar_texto("Gesti n de Usuarios y N mina", 240, 50,
       NEGRO)
    boton_crear_usuario = crear_boton("Crear Usuario", 300, 200,
       200, 50, AZUL, BLANCO)
    boton_iniciar_sesion = crear_boton("Iniciar Sesi n", 300,
       300, 200, 50, VERDE, BLANCO)
    return {"crear_usuario": boton_crear_usuario,
       "iniciar_sesion": boton_iniciar_sesion}
```

Listing 3: Función manejar_pagina_inicio

4.7 Funciones Importantes para la Nómina

1. Agregar Usuario: Esta función permite registrar nuevos usuarios en un archivo CSV, asegurando la persistencia de datos.

```
def agregar_usuario(email, password_hash):
    with open('usuarios.csv', mode='a', newline='') as archivo:
        escritor = csv.writer(archivo)
        escritor.writerow([email, password_hash])
```

Listing 4: Función agregar_usuario

2. Validación de Usuarios: Método en la clase `BaseDeDatos` que compara credenciales con los datos existentes.

```
def validar_usuario(self, usuario, contrasena):
    return self.usuarios.get(usuario) == contrasena
```

Listing 5: Método validar_usuario

3. Cálculo de Nómina: Aunque solo es un esquema inicial, este método prevé realizar cálculos automáticos de salarios y deducciones.

```
def calculador_nomina():
    # Configuraci n inicial
    # Captura interactiva
    # Validar entrada de datos
    # Buscar al empleado
    # Mostrar resultados
    pass
```

Listing 6: Esquema inicial para el cálculo de nómina

5 Comparación de Avances Semanales

Semana 1: Configuración inicial

- Configuración básica de Pygame y diseño HTML/CSS inicial.
- Problema: Cuadros de texto no alineados correctamente.

```

import pygame
pygame.init()

c_base = (0,0,0) #negro
c_rectangulo = (169, 169, 169)  # gris
c_texto = (255, 255, 255) # blanco
Largo = 1000  #lagro y acnho de la venta
Ancho = 700

screen = pygame.display.set_mode((Largo, Ancho)) # define screen y ademas define el largo y anho de la ventana
pygame.display.set_caption('Calculador de nomina') # literalmente solo nombra la ventaa xd\
fuente = pygame.font.SysFont(None, 40) #fuente para e texto creo que es arial segun reddit no lo es pero no onfio en esos locos]


def texto_id():

    rect_x = 100 #posicion x del rectangulo
    rect_y = 100 #posicion y del rectangulo
    rect_ancho = 300 #tamano del rectangulo
    rect_alto = 50 #ancho del rectangulo

    pygame.draw.rect(screen, c_rectangulo, (rect_x, rect_y, rect_ancho, rect_alto)) #literalmente solo dibuje

    texto = fuente.render("Id del Empleado", True, c_texto) #que texto quiero que salga en el recatangulo

```

```
27      screen.blit(texto, (rect_x + 50, rect_y +10)) #margenes
            del texto
28
29  def texto_horas():
30
31      rect_x = 100
32      rect_y = 200
33      rect_ancho = 300
34      rect_alto = 50 #
35
36      pygame.draw.rect(screen, c_rectangulo, (rect_x, rect_y,
            rect_ancho, rect_alto))
37
38      texto = fuente.render("horas trabajdas " , True, c_texto)
39
40      screen.blit(texto, (rect_x + 40, rect_y +10))
41
42  def texto_extas():
43
44      rect_x = 100
45      rect_y = 300
46      rect_ancho = 300
47      rect_alto = 50 #
48
49      pygame.draw.rect(screen, c_rectangulo, (rect_x, rect_y,
            rect_ancho, rect_alto))
50
51      texto = fuente.render("horas extras " , True, c_texto)
52
53      screen.blit(texto, (rect_x + 70, rect_y +10))
54
55  def main():
56      running = True # para que no se cierre la ventana
57      while running:
58          screen.fill(c_base) #darle un color a la ventana
59          texto_id()
60          texto_horas()
61          texto_extas()
62          dibujar_input_rects()
63          pygame.display.flip() #no se ccierre la ventana
64
65      pygame.quit() #cerrar la ventana
66
67  if __name__ == "__main__": #correr el codigo NO TOCAR (no se
        como funciona)
68      main()
```

Semana 2: Avances en interactividad

- Implementación de cuadros de entrada interactivos.

- Problema: Manejo inadecuado de eventos para entrada de texto.

```
1
2  import pygame
3  pygame.init()
4
5  c_base = (0,0,0) #negro
6  c_rectangulo = (169, 169, 169)  # gris
7  c_texto = (255, 255, 255) # blanco
8  Largo = 1000  #lagro y acnho de la venta
9  Ancho = 700
10
11 input_rects = [pygame.Rect(600, 100, 300, 50),
       pygame.Rect(600, 200, 300, 50), pygame.Rect(600, 300,
       300, 50)]
12 input_textos = ["", "", "", ]  # Lista de textos para cada
       cuadro
13 input_activo = [False, False, False,]  # Lista de estados de
       activaci n para cada cuadro
14
15
16 screen = pygame.display.set_mode((Largo, Ancho)) # define
       screen y ademas define el largo y anho de la ventana
17 pygame.display.set_caption('Calculador de nomina') #
       literalmente solo nombra la ventaa xd\
18 fuente = pygame.font.SysFont(None, 40) #fuente para e texto
       creo que es arial segun reddit no lo es pero no onfio en
       esos locos]
19
20
21 def texto_id():
22
23     rect_x = 100 #posicion x del rectangulo
24     rect_y = 100 #posicion y del rectangulo
25     rect_ancho = 300 #tamano del rectangulo
26     rect_alto = 50 #ancho del rectangulo
27
28     pygame.draw.rect(screen, c_rectangulo, (rect_x, rect_y,
           rect_ancho, rect_alto)) #literalmente solo dibuje
29
30     texto = fuente.render("Id del Empleado", True, c_texto)
           #que texto quiero que salga en el recatangulo
31
32     screen.blit(texto, (rect_x + 50, rect_y +10)) #margenes
           del texto
33
34 def texto_horas():
35
36     rect_x = 100
37     rect_y = 200
38     rect_ancho = 300
39     rect_alto = 50 #
```

```
40
41      pygame.draw.rect(screen, c_rectangulo, (rect_x, rect_y,
            rect_ancho, rect_alto))
42
43      texto = fuente.render("horas trabajdas " , True, c_texto)
44
45      screen.blit(texto, (rect_x + 40, rect_y +10))
46
47  def texto_extas():
48
49      rect_x = 100
50      rect_y = 300
51      rect_ancho = 300
52      rect_alto = 50 #
53
54      pygame.draw.rect(screen, c_rectangulo, (rect_x, rect_y,
            rect_ancho, rect_alto))
55
56      texto = fuente.render("horas extras " , True, c_texto)
57
58      screen.blit(texto, (rect_x + 70, rect_y +10))
59
60  def dibujar_input_rects():
61      for i, rect in enumerate(input_rects):
62          pygame.draw.rect(screen, c_texto, rect)
63          texto = fuente.render(input_textos[i], True, c_base)
64          screen.blit(texto, (rect.x + 40, rect.y + 10))
65
66
67  def main():
68      running = True # para que no se cierre la ventana
69      while running:
70          for event in pygame.event.get():
71              if event.type == pygame.QUIT:
72                  running = False
73              if event.type == pygame.MOUSEBUTTONDOWN:
74                  for i, rect in enumerate(input_rects):
75                      if rect.collidepoint(event.pos):
76                          input_activo = [False] *
                                len(input_rects)  # Desactivar
                                todo xd
77                          input_activo[i] = True  # Activar el
                                input seleccionado
78                          break
79
80              if event.type == pygame.KEYDOWN:
81                  for i, activo in enumerate(input_activo):
82                      if activo:  # para que el imput solo se
                            deje cuando uno le de click
83                          if event.key == pygame.K_BACKSPACE:
                                #boton de borado sirve graias a
```

```
                                dios
84                              input_textos[i] =
                                    input_textos[i][:-1]
85                          elif event.key == pygame.K_RETURN:
86                              print(f"Texto ingresado en
                                    cuadro {i+1}:
                                    {input_textos[i]}") # prueba
                                    por que no confio en mi mismo
87                              input_activo[i] = False  # dejar
                                    de esribir cunado se le de
                                    ente al escribri
88                          else:
89                              input_textos[i] += event.unicode
                                    #caracteres especiales
90
91          screen.fill(c_base) #darle un color a la ventana
92          texto_id()
93          texto_horas()
94          texto_extas()
95          dibujar_input_rects()
96          pygame.display.flip() #no se ccierre la ventana
97
98      pygame.quit() #cerrar la ventana
99
100
101 if __name__ == "__main__": #correr el codigo NO TOCAR (no se
    como funciona)
102     main()
```

Semana 3: Código final

- Integración de backend, frontend y base de datos.
- Implementación funcional de manejo de nóminas.

---

6 Conclusión

El proyecto **Calculadora de Nómina** permitió combinar habilidades técnicas en Python y diseño web para construir una solución funcional y robusta. A través de iteraciones semanales, se resolvieron problemas clave y se logró integrar todas las funcionalidades necesarias para una experiencia de usuario fluida.