Manual Técnico - Technical Manual

André Joaquin Ortega De Paz 201900597, Erick Enrique González Chávez 201900621, Julio Alfredo Fernández Rodríguez 201902416 y Albertt Wosveli Itzep Raymundo 201908658

Resumen – Se ha desarrollo un software para el manejo de información que se encuentra en archivos locales de la empresa Desktop Records que le permita a los usuarios manejarla de la manera más fácil.

Índice de Términos – Django, API, Flask, Backend

I. INTRODUCCIÓN

La empresa Desktop Records, decidió contratar a unos estudiantes de la facultad de Ingenieria pertenecientes a la carrera de Ciencias y Sistemas, los cuales cursan el curso de IPC 2, para el desarrollo de web site el cual ayude a presentar sus datos de manera más cómoda para los usuarios, el cual es capaz de procesar, manipular y presentar la información de la empresa.

El sistema estará compuesto por dos aplicaciones principales, un servidor de Python el cual funciona como Backend, el cual ayuda con el manejo de datos, y una plataforma Web que servirá como Frontend, el cual ayuda para presentar la información de manera más resumida y completa posible.

Dado que actualmente el mercado de desarrollo de software tiene bastante competencia, nuestro equipo de desarrollo desarrolló una aplicación original, siendo muy personalizada para que el usuario en cuestión pueda manejarse entre ventanas de una manera rápida y sencilla, y tener una interfaz capaz para cualquier persona que opte por primera vez al puesto de trabajo, siendo esta muy intuitiva para cualquier persona que requiera realizar una solicitud, teniendo como puntos fuertes las opciones de búsqueda, agregar, eliminar y generar un reporte de los datos de la empresa.

II. FUNCIONES

A. *Backend*

1) ***/discos/:*** se toma los datos de un archivo, se envía la información en formato json.

2) ***/discosTitulo/:*** se recibe un título en una estructura json, se carga un archivo, se busca el título en la información del archivo y se envía la información en una estructura json de no encontrar se envía un mensaje.

3) ***/discoYear/:*** se recibe un año en una estructura json, se carga un archivo, se busca el año en la información del archivo y se envía la información en una estructura json con los discos que salieron en el año de no encontrar se envía un mensaje.

4) ***/agregarDisco/:*** se recibe la información de un disco en una estructura en una estructura json y se agrega al contenido de un archivo y se sobrescribe.

5) ***/modificarDisco/:*** se recibe la información de un disco, se carga un archivo, su busca si existe el disco con el título, de no encontrarse se enviará un mensaje, de lo contrario se cambiará la información y se sobrescribirá el archivo.

6) ***/eliminarDisco/:*** se recibe un título en una estructura json, se carga un archivo, se busca el título en la información del archivo y se eliminará de la información y se sobrescribirá el archivo de no encontrar se envía un mensaje.

7) ***/reporteDiscos/:*** se carga un archivo, se usa el archivo para generar un archivo con comando para realizar un reporte con graphviz.

8) ***/empleados/:*** se carga un archivo, se envía la información en formato json.

9) ***/empleadoNombre/:*** se recibe un nombre en una estructura json, se carga un archivo, se busca el nombre en la información del archivo y se envía la información en una estructura json de no encontrar se envía un mensaje.

10) ***/empleadosDepartamento/:*** se recibe un departamento en una estructura json, se carga un archivo, se busca el departamento en la información del archivo y se envía la información en una estructura json con los empleados que estén en el departamento de no encontrar se envía un mensaje.

11) ***/empleadosSueldo/:*** se recibe un sueldo en una estructura json, se carga un archivo, se busca el sueldo en la información del archivo y se envía la información en una estructura json con los empleados que tienen ese sueldo de no encontrar se envía un mensaje.

12) ***/agregarEmpleado/:*** se recibe la información de un empleado en una estructura en una estructura json y se agrega al contenido de un archivo y se sobrescribe.

13) ***/modificarEmpleado/:*** se recibe la información de un empleado, se carga un archivo, su busca si existe con el id, de no encontrarse se enviará un mensaje, de lo contrario se cambiará la información y se sobrescribirá el archivo.

14) ***/eliminarEmpleado/:*** se recibe un id en una estructura json, se carga un archivo, se busca el id en la información del archivo y se eliminará de la información y se sobrescribirá el archivo de no encontrar se envía un mensaje.

15) ***/reporteEmpleados/:*** se carga un archivo, se usa el archivo para generar un archivo con comando para realizar un reporte con graphviz.

16) ***/paises/:*** se toma los datos de un archivo, se envía la información en formato json.

17) ***/paisMoneda/:*** se recibe una moneda en una estructura json, se carga un archivo, se busca la moneda en la información del archivo y se envía la información en una estructura json de los países que usen esa moneda de no encontrar se envía un mensaje.

18) ***/paisIdioma/:*** se recibe un idioma en una estructura json, se carga un archivo, se busca el idioma en la información del archivo y se envía la información en una estructura json de los países que usen el idioma de no encontrar se envía un mensaje.

19) ***/paisContinente/:*** se recibe un continente en una estructura json, se carga un archivo, se busca el continente en la información del archivo y se envía la información en una estructura json de los países que estén en el continente de no encontrar se envía un mensaje.

20) ***/agregarPais/:*** se recibe la información de un pais en una estructura en una estructura json y se agrega al contenido de un archivo y se sobrescribe.

21) ***/modificarPais/:*** se recibe la información de un país, se carga un archivo, su busca si existe con el continente y país, de no encontrarse se enviará un mensaje, de lo contrario se cambiará la información y se sobrescribirá el archivo.

22) ***/reporteRegiones/:*** se carga un archivo, se usa el archivo para generar un archivo con comando para realizar un reporte con graphviz.

24) ***/discoArtista/:*** se recibe un artista en una estructura json, se carga un archivo, se busca el artista en la información del archivo y se envía la información en una estructura json con los discos que tiene el artista de no encontrar se envía un mensaje.

III. Herramientas de Software

A. *Visual Studio Code*

Es un editor de código fuente liviano que puede ser usado en una en una computadora de escritorio y está disponible para Windows, macOS y Linux. Contiene un soporte para JavaScript, TypeScript y Node.js, y tiene un ecosistema perfecto lleno de extensiones para otros lenguajes.

B. *Django*

Es un software gratis y de código abierto, que se encarga de la mayor parte del desarrollo web y permite que el desarrollador se concentre en la creación de la aplicación. Django maneja autenticación de usuario, administración de contenido, mapa del sitio, y otras tareas. Permite tener una escala rápida y flexible.

C. *Flask*

Es la decisión de diseño de hacer tareas simples de manera simple, no debe de tener mucho código y se usan request válidos.

D. *Github*

Es un software de desarrollo usado por desarrolladores y compañías para construir, enviar y mantener su software.

References

[1] Getting Started Available: https://code.visualstudio.com/docs.

[2] Flask web development one drop at a time Available: https://flask.palletsprojects.com/en/2.1.x/

[3] Overvier Available: https://www.djangoproject.com/start/overview/

[4] https://github.com/about

IV. ANEXOS

```python
#Get /discos/
#Devuelve todos los discos en formato Json
@app.route('/discos/')
def discos():
    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()
    #print(raizDiscos)
    lista=[]

    for cd in raizDiscos:
            diccionario={
                "title":cd.findall("title")[0].text,
                "artist":cd.findall("artist")[0].text,
                "country":cd.findall("country")[0].text,
                "company":cd.findall("company")[0].text,
                "price":cd.findall("price")[0].text,
                "year":cd.findall("year")[0].text
                }
            lista.append(diccionario)

    return jsonify({"discos":lista})
```

Fig. 1. Función de /discos/.

```python
#Post /discoTitulo/
#Recibe un json con el titulo del disco y devule su informacion
@app.route('/discoTitulo/',methods=["POST"])
def discoTitulo():
    jsonres=request.get_json()
    try:
        title=jsonres["title"]
    except:
        return jsonify({"error":"Error en el json"})

    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()
    for cd in raizDiscos:
            if(cd.findall("title")[0].text.lower()==title.lower()):
                diccionario={
                "artist":cd.findall("artist")[0].text,
                "country":cd.findall("country")[0].text,
                "company":cd.findall("company")[0].text,
                "price":cd.findall("price")[0].text,
                "year":cd.findall("year")[0].text
                }
                return jsonify({"disco":diccionario})
    return jsonify({"mensaje":"No existe un disco con este titulo"})
```

Fig. 2. Función de /discosTitulo /.

```python
#Post /discoYear/
#Recibe un json con el año un año y devuelve tods los discos con esa fecha
@app.route('/discoYear/',methods=["POST"])
def discoYear():
    jsonres=request.get_json()
    try:
        year=str(jsonres["year"])
    except:
        return jsonify({"error":"Error en el json"})
    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()
    lista=[]
    rs=False
    for cd in raizDiscos:
            if(cd.findall("year")[0].text==year):...
    if rs:
        return jsonify({"disco":lista})
    else:
        return jsonify({"mensaje":"No se encontraron discos con ese año"})
```

Fig. 3. Función de /discoYear /.

```python
#Post /agregarDisco/
#Recibe un Json con toda la informacion de un disco y lo agrega
@app.route('/agregarDisco/',methods=["POST"])
def agregarDisco():
    doc = ET.parse("archivos/discos.xml")
    raiz = doc.getroot()

    jsonres=request.get_json()

    try:···
    except:···

    nuevoDisco = ET.SubElement(raiz,"cd")
    ET.SubElement(nuevoDisco, "title").text = title.lower()
    ET.SubElement(nuevoDisco, "artist").text = artist.lower()
    ET.SubElement(nuevoDisco, "country").text = country.lower()
    ET.SubElement(nuevoDisco, "company").text = company.lower()
    ET.SubElement(nuevoDisco, "price").text = str(price)
    ET.SubElement(nuevoDisco, "year").text = str(year)
    doc.write("archivos/discos.xml", xml_declaration=True,encoding="utf-8")
    return jsonify({"mensaje":"Listo,disco agregado con exito"})
```

Fig. 4. Función de /agregarDisco /.

```python
#Post /modificarDisco/
#Recibe un Json con todos los datos de un disco, pero primero verifica si exite el titulo
@app.route('/modificarDisco/',methods=["POST"])
def modificarDisco():
    jsonres=request.get_json()

    try:···
    except:···

    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()

    for cd in raizDiscos:
        if(cd.findall("title")[0].text.lower()==title.lower()):···
    return jsonify({"error":"No existe un disto con este titulo"})
```

Fig. 5. Función de / modificarDisco /.

```python
#Post /eliminarDisco/
#Recibe un Json con el titulo a eliminar
@app.route('/eliminarDisco/',methods=["POST"])
def eliminarDisco():

    jsonres=request.get_json()

    try:
        title=jsonres["title"]
    except:
        return jsonify({"error":"Error en el json"})

    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()

    for cd in raizDiscos:
        if(cd.findall("title")[0].text.lower()==title.lower()):
            raizDiscos.remove(cd)
            doc.write("archivos/discos.xml", xml_declaration=True,encoding="utf-8")
            return jsonify({"mensaje":"Disco eliminado con exito"})

    return jsonify({"mensaje":"No existe un disto con este titulo"})
```

Fig. 6. Función de /eliminarDisco /.

```python
#Get /reporteDiscos/
#Genera un reporte con graphviz de todos los discos
@app.route('/reporteDiscos/')
def reporteDiscos():
    contador = 0
    manf = open('Comandos/ComandosDiscos.dot', 'w')

    comando = 'digraph G {\ncharset="latin1"\nnode [shape=square,style="rounded"];\nrankdir="LR";\n'

    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()

    for cd in raizDiscos:···
    comando += "}"

    #print(comando)
    manf.write(comando)
    manf.close()
    comand = "dot -Tjpg Comandos/ComandosDiscos.dot -o Reportes/ReporteDiscos.jpg"
    check_output(comand, shell=True)
    return jsonify({"mensaje":"Reporte generado con exito"})
```

Fig. 7. Función de /reporteDiscos/.

```python
#Get /empleados/
#Devuelve todos los empleados en formato json
@app.route('/empleados/')
def empleados():
    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()
    departamentos={}

    for departamento in raizEmpleados:
        depto=departamento.attrib['departamento']
        empleadosDepto=[]
        for empleado in departamento:
            dic={
                'id': empleado.attrib['id'],
                "nombre":empleado.findall("nombre")[0].text ,
                "puesto":empleado.findall("puesto")[0].text ,
                "salario":empleado.findall("salario")[0].text
            }
            empleadosDepto.append(dict(dic))
        departamentos[depto]=empleadosDepto
    return jsonify({"empresa":departamentos})
```

Fig. 8. Función de /empleados/.

```python
#Post /empleadonombre/
#Recibe el nombre del empleado y muestra su informacion
@app.route('/empleadoNombre/',methods=["POST"])
def empleadoNombre():
    jsonres=request.get_json()

    try:
        nombre=jsonres["nombre"]
    except:
        return jsonify({"error":"Error en el json"})

    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados= doc.getroot()

    for departamento in raizEmpleados:
            for empleado in departamento:
                depto=departamento.attrib['departamento']
                if (empleado.findall("nombre")[0].text.lower() == nombre.lower()):···
    return jsonify({"mensaje":"No hay un empleado con ese nombre"})
```

Fig. 9. Función de /empleadoNombre/.

```python
#Post /empleadosDepartamento/
#Recibe un json con el nombre del departamento
@app.route('/empleadosDepartamento/',methods=["POST"])
def empleadosDepartamento():
    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()
    departamentos={}
    jsonres=request.get_json()
    try:
        dept=jsonres["departamento"]
    except:
        return jsonify({"Error":"Error en el json"})

    for departamento in raizEmpleados:
        depto=departamento.attrib['departamento']
        empleadosDepto=[]
        for empleado in departamento:
            dic={···
            }
            empleadosDepto.append(dic)
        if depto==dept:
                return jsonify({"deparmento":empleadosDepto})
    return jsonify({"Mensaje":"Este departamento no existe dentro de la empresa"})
```

Fig. 10. Función de /empleadosDepartamento/.

```python
#Post /empleadoSueldo/
#Recibe un json con el sueldo a buscar
@app.route('/empleadosSueldo/',methods=["POST"])
def empleadosSueldo():
    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()
    jsonres=request.get_json()

    try:
        sueldo=jsonres["sueldo"]
    except:
        return jsonify({"error":"Error en el json"})
    empleados=[]
    rs=False

    for departamento in raizEmpleados:
        depto=departamento.attrib['departamento']
        for empleado in departamento:
            if float(empleado.findall("salario")[0].text)==float(sueldo):···
    if rs:
        return jsonify({f"sueldo":empleados})
    else:
        return jsonify({"mensaje":"No hay ningun empleado con ese salario"})
```

Fig. 11. Función de /empleadosSueldo/.

```python
#Post agregarEmpleado/
#Recibe un json con el departamento a agregar y toda la info. del empleado
@app.route('/agregarEmpleado/',methods=["POST"])
def agregarEmpleado():
    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()
    rs=False
    contador=0

    raizEmpleados = doc.getroot()
    jsonres=request.get_json()
    try:
        depto=jsonres["departamento"]
        id=jsonres["id"]
        nombre=jsonres["nombre"]
        puesto=jsonres["puesto"]
        salario=jsonres["salario"]
    except:
        return jsonify({"mensaje":"Error en el json"})

    for departamento in raizEmpleados:
            if departamento.attrib['departamento'].lower()==depto.lower() and rs==False:···

    return jsonify({"mensaje":"No existe un departamento con este nombre"})
```

Fig. 12. Función de /agregarEmpleado/.

```python
#Post /modificarEmpleado/
#Recibe la id del empleado y todos sus datos a modificar
@app.route('/modificarEmpleado/',methods=["POST"])
def modificarEmpleado():

    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()

    jsonres=request.get_json()

    try:
        id=jsonres["id"]
        nombre=jsonres["nombre"]
        puesto=jsonres["puesto"]
        salario=jsonres["salario"]
    except:
        return jsonify({"Error":"Error en el json"})

    for departamento in raizEmpleados:···
    return jsonify({"Mensaje":"No existe un empleado con esa id"})
```

Fig. 13. Función de /modificarEmpleado/.

```python
#Post /eliminarEmpleado/
#Recibe un json con el id del empleado a eliminar
@app.route('/eliminarEmpleado/',methods=["POST"])
def eliminarEmpleado():

    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()

    jsonres=request.get_json()
    try:
        id=jsonres["id"]
    except:
        return jsonify({"error":"Error en el json"})


    for departamento in raizEmpleados:
            for empleado in departamento:
                if (int(empleado.attrib['id']) == int(id)):
                    departamento.remove(empleado)
                    doc.write("archivos/empleados.xml", xml_declaration=True,encoding="utf-8")
                    return jsonify({"mensaje":"Empleado eliminado con exito"})
    return jsonify({"mensaje":"No existe un empleado con esa id"})
```

Fig. 14. Función de /eliminarEmpleado/.

```python
#Post /reporteEmpleados/
#Genera un reporte en graphviz con todos los departamentos y empleados de la empresa
@app.route('/reporteEmpleados/')
def reporteEmpleados():
    #obteniendo ruta para realizar el renderizado en carpeta static/ Django
    jsonres=request.get_json()

    try:···
    except:···

    doc = ET.parse("archivos/empleados.xml")
    raizEmpleados = doc.getroot()

    contador = 0
    manf = open('Comandos/ComandosEmpleados.dot', 'w')
    comando = 'digraph G {\ncharset="latin1"\nnode [shape=square,style="rounded"];\nrankdir="LR";\n'

    for departamento in raizEmpleados:···

    comando += "}"

    #print(comando)
    manf.write(comando)
    manf.close()
    comand = "dot -Tjpg Comandos/ComandosEmpleados.dot -o {}/ReporteEmpleados.jpg".format(ruta)
    check_output(comand, shell=True)
    return jsonify({"mensaje":"Reporte generado con exito"})
```

Fig. 15. Función de /reporteEmpleados/.

```python
#Get /paises/
@app.route('/paises/')
def paises():
    doc = ET.parse("archivos/paises.xml")
    raizPaises = doc.getroot()
    continentes={}

    for continente in raizPaises:
        cont=continente.attrib['name']
        paises=[]
        for pais in continente:
            diccionario={
                "pais":pais.findall('nombre')[0].text,
                "capital":pais.findall('capital')[0].text,
                "idioma":pais.findall('idioma')[0].text,
                "moneda":pais.attrib['moneda']
            }
            paises.append(diccionario)
        continentes[cont]=paises
    return jsonify({"mundo":continentes})
```

Fig. 16. Función de /paises/.

```python
#Post /paisMoneda/
@app.route('/paisMoneda/',methods=["POST"])
def paisMoneda():
    doc = ET.parse("archivos/paises.xml")
    raizPaises = doc.getroot()

    jsonres=request.get_json()

    try:
        moneda=jsonres["moneda"]
    except:
        return jsonify({"error":"Error en el json"})

    continentes={}
    rs=False

    lista=[]

    for continente in raizPaises:···
    if rs:
        return jsonify({"moneda":lista})
    else:
        return jsonify({"mensaje":"No hay un pais con esa moneda"})
```

Fig. 17. Función de /paisMoneda/.

```python
#Post /paisIdioma/
@app.route('/paisIdioma/',methods=["POST"])
def paisIdioma():
    doc = ET.parse("archivos/paises.xml")
    raizPaises = doc.getroot()

    jsonres=request.get_json()

    try:
        idioma=jsonres["idioma"]
    except:
        return jsonify({"error":"Error en el json"})

    continentes={}
    rs=False
    lista=[]

    for continente in raizPaises:···
    if rs:
        return jsonify({"idioma":lista})
    else:
        return jsonify({"mensaje":"No hay un pais con esa moneda"})
```

Fig. 18. Función de /paisIdioma/.

```python
#Post /paisContinente/
@app.route('/paisContinente/',methods=["POST"])
def paisContinente():
    doc = ET.parse("archivos/paises.xml")
    raizPaises = doc.getroot()

    jsonres=request.get_json()

    try:
        conti=jsonres["continente"]
    except:
        return jsonify({"error":"Error en el json"})

    lista=[]
    rs=False

    for continente in raizPaises:···
    if rs:
        return jsonify({"continente":lista})
    else:
        return jsonify({"mensaje":"No hay paises registrados en este continente"})
```

Fig. 19. Función de /paisContinente/.

```python
#Post /agregarPais/
@app.route('/agregarPais/',methods=["POST"])
def agregarPais():
    doc = ET.parse("archivos/paises.xml")
    raizPaises = doc.getroot()

    jsonres=request.get_json()

    try:     ···
    except :
        return jsonify({"error":"Error en el json"})

    rs=False

    for continente in raizPaises:···

    return jsonify({"mensaje":"No existe un continente con este nombre"})
```

Fig. 20. Función de /agregarPais/.

```python
#Post /modificarPais/
@app.route('/modificarPais/',methods=["POST"])
def modificarPais():
    doc = ET.parse("archivos/paises.xml")
    raizPaises = doc.getroot()

    jsonres=request.get_json()

    try:
        continent=jsonres["continente"]
        mon=jsonres["moneda"]
        nombre=jsonres["nombre"]
        capital=jsonres["capital"]
        idioma=jsonres["idioma"]
    except :
        return jsonify({"error":"Error en el json"})

    cambioContinente=True
    rs=False

    for continente in raizPaises:···


    if cambioContinente:···

    if  rs==False:
        return jsonify({"mensaje":"No hay un pais o continente con ese nombre"})
```

Fig. 21. Función de /modificarPais/.

```python
#Get /reporteRegiones/
@app.route('/reporteRegiones/')
def reporteRegiones():
    doc=ET.parse('archivos/paises.xml')
    raizPaises=doc.getroot()

    contador = 0
    manf = open('Comandos/ComandosPaises.dot', 'w')
    comando = 'digraph G {\ncharset="latin1"\nnode [shape=square,style="rounded"];\nrankdir="LR";\n'

    for continente in raizPaises:···


    comando += "}"

    print(comando)
    manf.write(comando)
    manf.close()
    comand = "dot -Tjpg Comandos/ComandosPaises.dot -o Reportes/ReportePaises.jpg"
    check_output(comand, shell=True)
    return jsonify({"mensaje":"Reporte generado con exito"})
```

Fig. 22. Función de /reporteRegiones/.

```python
#Post /discoArtista/
#Recibe un Json con un artista y devuelve todos sus discos
@app.route('/discoArtista/',methods=["POST"])
def discoArtista():
    jsonres=request.get_json()

    try:
        artista=jsonres["artist"]
    except:
        return jsonify({"error":"Error en el json"})

    doc = ET.parse("archivos/discos.xml")
    raizDiscos = doc.getroot()
    lista=[]
    rs=False
    for cd in raizDiscos:
            if(cd.findall("artist")[0].text.lower()==artista.lower()):···
    if rs:
        return jsonify({"artista":lista})
    else:
        return jsonify({"mensaje":"No se encontraron discos de este artista"})
```

Fig. 23. Función de /discoArtista/.

V. Correo para solución de problemas

Andre Joaquin Ortega De Paz – 5538 1782
CORREO – 3191363100501@ingenieria.usac.edu.gt

Erick Enrique González Chávez – 4908-7375
CORREO – gonzalezerickenrique21@gmail.com

Julio Alfredo Fernandes Rodriguez – 4024 0325
CORREO –alberttitzep123@gmail.com

Albertt Wosveli Itzep Raymundo – 4180 4614
CORREO –juliorod374@gmail.com