PROYECTO #3 IPC2

202004810 – Henry Ronely Mendoza Aguilar

Resumen

Empleando la programación orientada a objetos, los fundamentos de la programación web se desarrolló un software que permite la autorización de documentos tributarios electrónicos (DTE)

Para la lectura y registro de las autorizaciones y documentos electrónicos se empleó el formato de archivo xml, además para la creación de un servicio web se utilizó el lenguaje de programación Python usando la librería Flask para realizar el BackEnd, además se utilizó el framework Django para el desarrollo del FrontEnd asimismo de la implementación del uso de JavaScript, HTML y CSS para la maquetación de la página web y las solicitudes al servidor. No se emplearon bases de datos en su reemplazo se utilizaron archivos auxiliares en formateo xml.

Palabras clave
Backend, Frontend, web, POO, Django

Abstract

Using object-oriented programming, the fundamentals of web programming, a software was developed that allows the authorization of electronic tax documents (DTE)

For the reading and registration of authorizations and electronic documents, the xml file format was used, in addition to the creation of the web service, the Python programming language was used using the Flask library to perform the BackEnd, in addition the Django framework was used for the Development of the FrontEnd also of the implementation of the use of JavaScript, HTML and CSS for the layout of the web page and requests to the server. No databases were used to replace them, auxiliary files in xml format were used.

Keywords
Backend, Frontend, web, OOP, Django

Introducción

En los últimos años las aplicaciones y paginas web han cobrado una mayor repercusión para las empresas ya que permite una reducción en el uso de recursos y además de la portabilidad y compatibilidad entre dispositivos ya que únicamente se necesita de un navegador.

Por tal motivo fue solicitado el desarrollo de un software que permita la gestión de autorización de documentos tributarios.

Desarrollo

Para el desarrollo de la aplicación solicitada por la Superintendencia de administración Tributaría (SAT), comencé realizando un diagrama sobre las clases (*ver figura 1*) que implementaría para el desarrollo del software y el diagrama de despliegue entre el Frontend y Backend. (*ver figura 2*).

Dentro de las clases para el almacenamiento de los datos de entrada a través del archivo xml, plantee dos clases, solicitud_dte y apronadas, dentro de la clase solicitud_dte (*ver figura 3*). se plantearon seis atributos para la instancia de la clase de los cuales fueron obtenidos del archivo xml de entrada, añadiendo a esto se crearon dos atributos más un booleano que determina si la solicitud está autorizada y una lista de los errores que posea el DTE. La clase aprobadas se creó para la gestión de las solicitudes que no tuvieron errores. (*ver figura 4*)

Añadiendo a las clases anteriores se creó una clase llamada error_dte (*ver figura 5*). La cual funciona como gestor de los errores que posean los DTE almacenados dentro de la clase solicitud_dte en estas se añadieron contadores y métodos para validar el DTE entrante y determinar si esta puede ser autorizada o no.

Posterior a la creación de clases para la gestión de las solicitudes se procedió a comenzar con la maquetación de la aplicación web. Cabe destacar que el proceso no fue lineal entre capas, es decir, no se realizó el Frontend completo y luego el Backend sino que se iban creando paralelamente mediante el avance del proyecto.

FrontEnd

Front End es la parte de una aplicación que interactúa con los usuarios, es conocida como el lado del cliente. Básicamente es todo lo que vemos en la pantalla cuando accedemos a un sitio web o aplicación: tipos de letra, colores, adaptación para distintas pantallas (RWD), los efectos del ratón, teclado, movimientos, desplazamientos, efectos visuales… y otros elementos que permiten navegar dentro de una página web. Este conjunto crea la experiencia del usuario.

En este caso se empleó un framework para facilitar la gestión del frontend en este caso se utilizón Django. Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador (MVC). Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt.

En junio de 2008 fue anunciado que la recién formada Django Software Foundation se haría cargo de Django en el futuro.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Los orígenes de Django en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan

el desarrollo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores escriban controladores y vistas para las áreas de administración de la página, Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con Django y que puede administrar varias páginas a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).

Ya que Django funciona con vistas dentro del proyecto se implementaron 2 vistas llamadas index y peticiones (*ver figura 6*). Además del uso de Django se utilizó JavaScript que es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web. Fue inventado por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla. Principalmente el uso de JavaScript se utilizó para la interacción de "pestañas" en ciertas partes de la aplicación web y para realizar peticiones múltiples en una misma vista ya que django unicmente permite dos, un post y un get por vista. Asimismo, javascript fue utilizado para la generación de reportes pdf. (*ver figura 7*).

De igual forma para la creación de cada vista se empleó html y css ya que django trabaja con ello para la creación de las vistas dentro del framework.

La aplicación era dinámica y compleja por ello fue necesario crear una API empelando Python como Backend así tener un mejor control de los datos que se manejan entre el servidor y la capa cliente.

BackEnd

El backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor.

Algunas de las funciones que se gestionan en la parte del backend son:

- El Desarrollo de funciones que simplifiquen el proceso de desarrollo.
- Acciones de lógica.
- Conexión con bases de datos.
- Uso de librerías del servidor web (por ejemplo, para implementar temas de caché o para comprimir las imágenes de la web).

Como se mencionó se utilizó el lenguaje Python sin embargo, este lenguaje no esta hecho directamente para la recepción de peticiones y respuestas tipo http por ello fue necesaria la implementación de una librería llamada flask, Flask es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2 y tiene una licencia BSD.

La implementación de flask se dio mediante métodos GET, POST y DELETE los cuales fueron necesarios para las diferentes peticiones y procesos que necesitaba la aplicación web. (*ver figura 8*)

Descripción de la interfaz

La interfaz se basa en dos vistas la principal o index cuenta con dos cuadros de texto con unas etiquetas de entrada y salida. Cuenta con una barra de navegación la cual redirige a peticiones y un menú desplegable con la documentación de la aplicación y la información de soporte, a su vez tiene un botón para cargar un archivo xml, un botón para enviar dicho archivo al servidor y realizar el procesamiento y un botón que resetea la base de datos para empezar de nuevo cualquier acción. (*ver figura 9*).

La vista de peticiones cuenta con la misma barra de navegación que el index, sin embargo, cambia el botón de cambiar archivo por el de inicio que regresa a la vista index. Cuenta con una sección con varias pestañas que funcionan como un menú lo que permite visualizar múltiples peticiones (*ver figura 10*) las cuales son:

- Autorizaciones: muestra las facturas autorizadas detalladas dentro de una tabla
- Rango de fechas: filtra las facturas autorizadas en el rango de fechas seleccionado además de permitir generar un reporte pdf con los datos en pantalla, al filtrar se muestra una tabla y una grafica de el rango seleccionado.
- Movimientos: Realiza una búsqueda por día de los movimientos de Nits efectuados de igual forma permite generar un pdf y al filtrar muestra una tabla y una gráfica con los datos solicitados en la búsqueda.

Anexos

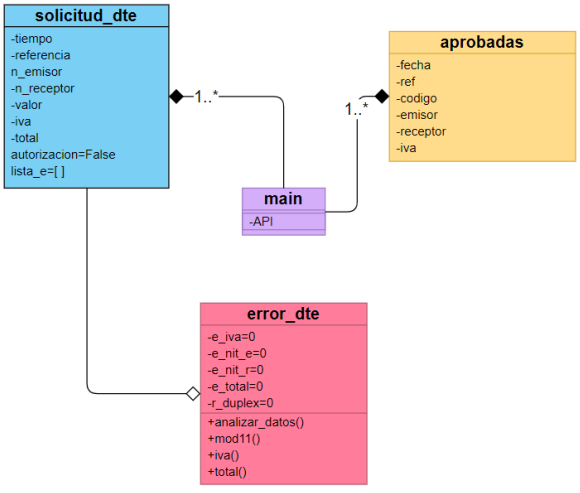

Figura 1. Diagrama de Clases
Fuente: Elaboración propia, 2021

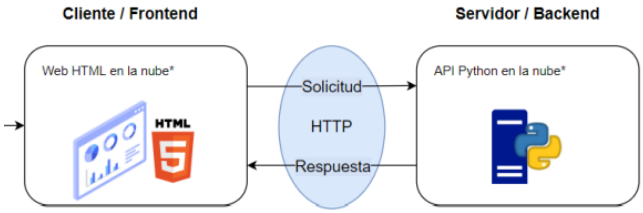

Figura 2. Diagrama de despliegue
Fuente: Elaboración propia, 2021

```
class solicitud_dte:
    def __init__(self,tiempo,referencia,n_emisor,n_receptor,valor,iva,total):
        self.tiempo=tiempo
        self.referencia = referencia
        self.n_emisor = n_emisor
        self.n_receptor = n_receptor
        self.valor = valor
        self.iva = iva
        self.total = total
        self.autorizacion=False
        self.list_e=[]
```

Figura 3. Clase solicitud_dte
Fuente: Elaboración propia, 2021

```
class aprobadas:
    def __init__(self,fecha,ref,codigo,emisor,receptor,iva):
        self.fecha = fecha
        self.ref = ref
        self.codigo = codigo
        self.emisor = emisor
        self.receptor = receptor
        self.iva = iva
```

Figura 4. Clase aprobadas
Fuente: Elaboración propia, 2021

```
class error_dte:
    def __init__(self):
        self.e_iva=0
        self.e_nitE=0
        self.e_nitR=0
        self.e_total=0
        self.r_duplex=0

    def analizar_datos(self,lista_dte):...

    def mod11(self,nit,type,lista_e):...

    def iva(self,iva,valor,lista_e):...

    def total(self,iva,valor,total,lista_e):...
```

Figura 5. Clase error_dte
Fuente: Elaboración propia, 2021

```python
def index(request):
    if request.method == 'GET':
        url = port.format('/file')  # http://localhost:3000/file
        data = requests.get(url).json()  # consulta a la API
        context = {
            'data': data,
        }
        return render(request, 'index.html', context)

    elif request.method == 'POST':
        docs = str(request.FILES['document'].read())
        docs=docs.replace("b'",'')
        docs=docs.replace("'",'')
        docs=docs.replace("\\r\\n",'\n')
        data = {'URL': str(docs)}
        url = port.format('/file')
        requests.post(url, json=data)
        return redirect('index')

def peticiones(request):
    if request.method == 'GET':
        url = port.format('/peticion')  # http://localhost:3000/file
        data = requests.get(url).json()  # consulta a la API
        context = {
            'data': data,
        }
        return render(request, 'peticiones.html',context)
```

Figura 6. Views en Django
Fuente: Elaboración propia, 2021

```javascript
function exportarpdf(){
    var element = document.getElementById("rango")
    var fecha1 = document.querySelector('#inicio').value
    var fecha2 = document.querySelector('#fin').value
    let date = "Rango: "+format(String(fecha1))+" - "+format(String(fecha2))

    var ePDF = element.outerHTML
    console.log(ePDF)
    ePDF = ePDF.replace('<article id="tab11" style="">','<h6 style="text-align: center; font-size: 24px;">Resume
    ePDF = ePDF.replace('<article id="tab22" style="display: none;">','<h6>Ver sin IVA:</h6><article id="tab22"
    ePDF = ePDF.replace('display: none;','')
    var opt = {
        margin:       [2,2,1,2],
        filename:     'rango_fechas.pdf',
        image:        { type: 'jpeg', quality: 0.99 },
        html2canvas:  { scale: 4 },
        jsPDF:        { unit: 'cm', format: 'letter', orientation: 'portrait' }
      };

    html2pdf().set(opt).from(ePDF).save();
}
```

Figura 7. Exportar a PDF
Fuente: Elaboración propia, 2021

```python
@app.route('/file', methods=['GET'])
def get_file():...


@app.route('/file', methods=['POST'])
def post_data():...
```

Figura 8. API en Flask
Fuente: Elaboración propia, 2021

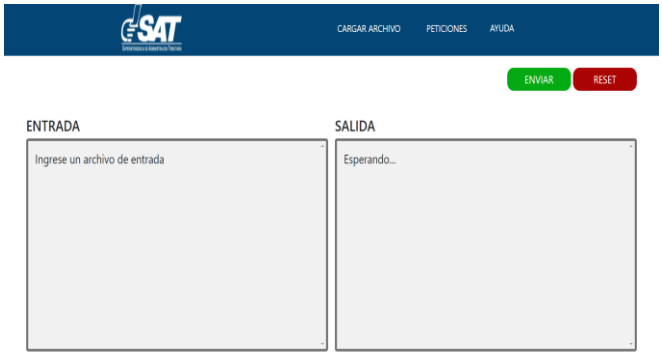

Figura 9. Vista principal
Fuente: Elaboración propia, 2021

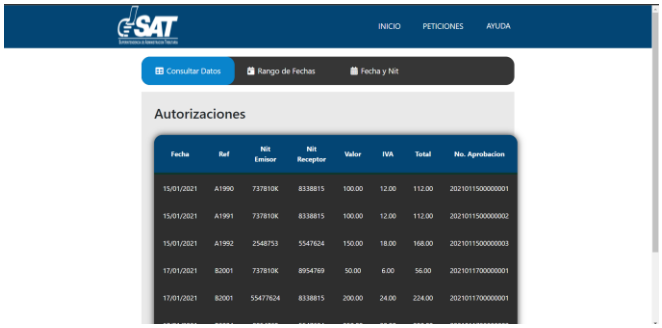

Figura 9. Vista peticiones
Fuente: Elaboración propia, 2021