Invoice Validator Code Challenge

El reto que se describe a continuación consiste en crear una aplicación que aplique las reglas de validación correspondientes a un documento xml.

Se recomienda leer todas las instrucciones antes de comenzar con la codificación. Es importante aclarar todas las dudas, siéntete libre de preguntar tantas veces como sea necesario.

Algunas consideraciones **importantes** previo a compartirnos la solución final:

- Asegurate que tu solución contenga un archivo README y que considere lo siguiente:
  - Instrucciones para compilar y ejecutar el proyecto.
  - Notas importantes que nos ayuden a un mejor entendimiento de la solución que nos presentas.
  - Frameworks , librerías y/o tecnología utilizada en tu proyecto.
- Archivo Zip.
  - El paquete solo debe contener el código fuente y documentación. Evita los archivos innecesarios como librerías , binarios etc.
- Favor de no subir esta solución a repositorios públicos.
- Puedes incluir diagramas y/o documentos de apoyo que ayuden a un mejor entendimiento de la solución que nos presentas.

Es importante que las instrucciones que nos compartes sean suficiente para que un developer experimentado pueda construir y ejecutar la aplicación (build and run) de preferencia en un S.O windows en alguno de los siguientes lenguajes: .dotnet 5.0+ Python 3.7+ , Node v16+.
También **puedes dockerizar tu aplicación.**

¿Cómo calificamos el reto?

- **Código limpio.** Esperamos que el código sea simple y fácil de entender.
- **Estructura del proyecto.** Naming, separation of concern, organización de código.
- **Performance.** Agrega comentarios al código donde creas conveniente para explicarnos las decisiones que impactan directamente en el performance de la aplicación.
- **Open for extension.**
- **Funciona.** La aplicación debe funcionar sin modificación alguna.

- **Documentación.**
- **Pruebas unitarias.** Intentaremos romper el código. Siéntete libre de generar tantas pruebas unitarias como sea posible.

Notas generales

Puedes utilizar librerías , parsers , manejadores de xml etc. Elije cuidadosamente tus librerías y solamente agrega aquellas que sean estrictamente necesarias.
Evita utilizar servicios que agregan complejidad a tu solución, por ejemplo una base de datos no es necesaria.

¿Cómo se espera que funcione la aplicación?

Desde una consola de comandos podremos ejecutar la siguiente instrucción:
[comando aplicación] [archivo de entrada.xml]
Output:

{"cfdi":{"version":"3.3","validation":{"code":"CFDI33101","result":{"isValid":false,"message":"El campo Fecha no cumple con el patrón requerido."}}}}
{"cfdi":{"version":"3.3","validation":{"code":"CFDI33104","result":{"isValid":false,"message":"El campo FormaPago no contiene un valor del catálogo c_FormaPago."}}}}
{"cfdi":{"version":"3.3","validation":{"code":"CFDI33141","result":{"isValid":true,"message":"OK"}}}}

Invoice Validator

Un poco de contexto.
La facturación electrónica en México ha evolucionado desde 2011 a partir de su obligatoriedad. En México nos enfrentamos a diferentes retos con respecto a mantener un documento que se ha vuelto un estandar y que es utilizado para diferentes procesos core de las empresas, desde facturación simple hasta pagos, nómina, transporte de mercancías, comercio exterior etc. Un solo documento , un solo estandar que abarca diferentes procesos de una empresa ha significado un gran reto sobre todo cómo garantizamos que los datos que se capturan en las facturas son correctos.

La solución que desarrollarás como parte de este reto podrá validar 2 tipos de documentos xml. La aplicación decidirá cuáles reglas aplicará basándose en el tipo de documento de entrada:

- Cfdi versión 3.3
- Retenciones versión 2.0

Puedes asumir lo siguiente durante la solución:

- Solo aplicarás las reglas de validación que se especifican en el documento. No se considerarán fallos en la validación a atributos que no sean parte de las reglas de negocio.
- Los documentos xml a procesar serán menor a 1 MB.

Validaciones CFDI versión 3.3

Atributo	**Reglas de validación para CFDI versión 3.3**	**CÓDIGO**	**MENSAJE ERROR**
Fecha	El atributo cfdi:Comprobante:fecha debe cumplir con el patrón (20[1-9][0-9])-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])T(([01][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9])	CFDI33101	El campo Fecha no cumple con el patrón requerido.
FormaPago	El atributo cfdi:Comprobante:FormaPago, debe contener un valor del catálogo c_FormaPago.	CFDI33104	El campo FormaPago no contiene un valor del catálogo c_FormaPago.
Receptor:UsoCFDI	El valor que se registre en este atributo debe aplicar para el tipo de persona del receptor.	CFDI33141	La clave del campo UsoCFDI debe corresponder con el tipo de persona (física o moral).

Validaciones Retenciones versión 2.0

Atributo	Reglas de validación para Retenciones 2.0	CÓDIGO	MENSAJE ERROR
CveRetenc	Si el valor de este atributo es "25", debe existir el atributo DescRetenc.	Reten20106	El campo DescRetenc debe existir.
FechaExp	Este debe cumplir con el patrón (20[1-9][0-9])-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])T(([01][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]) .	Reten20103	El campo FechaExp no cumple con el patrón requerido.
Retenciones:Periodo:MesIni	La clave vigente del catálogo c_Periodo registrada en este atributo debe ser menor o igual que el atributo MesFin.	Reten20122	El campo MesIni no es menor o igual que el campo MesFin.

** Se anexan en un documento aparte los catálogos necesarios para algunas de las reglas de validación (catCFDI_V_33_code_challenge.xls) así como un ejemplo de xml cfdi 3.3 y retenciones 2.0.