ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD: INFORMÁTICA Y ELECTRÓNICA
CARRERA: SOFTWARE

GUÍA DE LABORATORIO DE CONSTRUCCIÓN DE SOFTWARE
PARALELO: A

PRÁCTICA No. 1 - (Parsers)

1. DATOS GENERALES:

NOMBRE: (estudiante)	**CODIGO: (de estudiante)**
Diego Logroño	6799
Cristhian Romero	6805
John Cuvi	6680
Sebastián Játiva	6695
Samantha Guerreo	6618
Michael Santillan	6610
Stalyn Londo	6738

Periodo Académico: PERIODO ACADÉMICO ORDINARIO OCTUBRE 2021 – MARZO 2021
Semestre: Quinto
Tutor: Ing. Vinicio Ramos

FECHA DE REALIZACIÓN:	**FECHA DE ENTREGA:**
16/12/2021	**18/12/2021**

2. OBJETIVO:

Desarrollar un parser ya sea DOM o SAX que permita extraer información de documentos XML generados de manera estática o dinámica.

3. INSTRUCCIONES

Tomando como referencia los siguientes repositorios github:

- xmlsaxparser
- xmldomparser
- xmlserving

disponibles en github.com/omargomez2, elabora un parser en otro lenguaje distinto al usado en clases el cual sea capaz de extraer y formatear la información contenida en un documento xml generado de manera estática o dinámica. Emplea uno de los dos enfoques vistos en clase DOM o SAX.

4. EQUIPOS Y MATERIALES:

Equipos:

- Laptop Intel Core i5 Décima generación 16GB RAM 1TB ssd
- Laptop Intel Core i7 octava generación 12 GB RAM 1TB HDD
- laptop lenovo Core i5 Décima generación 8gb ram 256gb ssd
- Laptop HP Core i7 de Décima generación y 8gb de ram
- Laptop Toshiba Intel Core i5 8GB RAM 1TB
- Laptop Lenovo Ryzen 7 3500u 12 Ram 128 SSD
- Laptop HP Core i7, 8 RAM 250 SSD

Software que utilizaremos:

El desarrollo del parserDom para leer archivos xml de manera estática fue realizado en el lenguaje de programación c# en la aplicación Visual Studio 2019 mediante las librerías XML donde se incluye los elementos necesarios.

5. ACTIVIDADES POR DESARROLLAR:

A continuación, se listan las actividades a efectuar

1. **Describir el contexto del problema**

   La importancia que radica en el manejo de archivos xml hoy en día es grande ya que en el ámbito laboral es de suma importancia conocer como está estructurado

este tipo de archivo. Por eso es fundamental conocer las técnicas a implementar y lenguajes para la resolución de problemas.

Con respecto a los datos que se va a realizar a través de este informe se puede decir que hoy en día la utilización de archivos xml dentro de una institución académica es muy fundamental para organizar los datos de cada estudiante y así llevar un registro de cada uno de ellos, como puede ser las notas, información personal, código de los estudiantes récord académico. Por ellos se ha realizado un programa en donde abarque en este caso la información personal de cada estudiante y posteriormente que el programa genere un archivo xml con el resultado final de los datos.

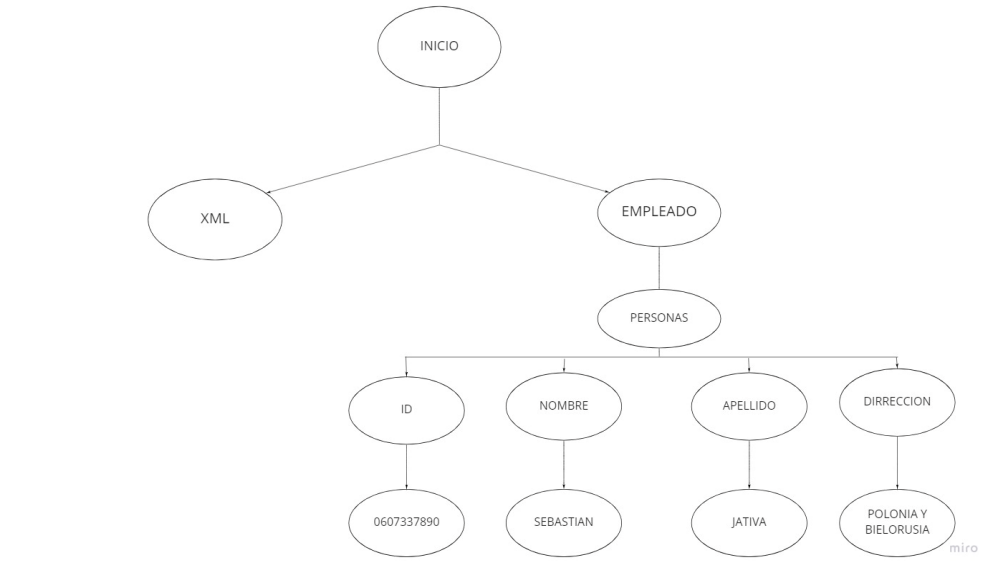
2. **Describir la estructura del documento XML a leer.**

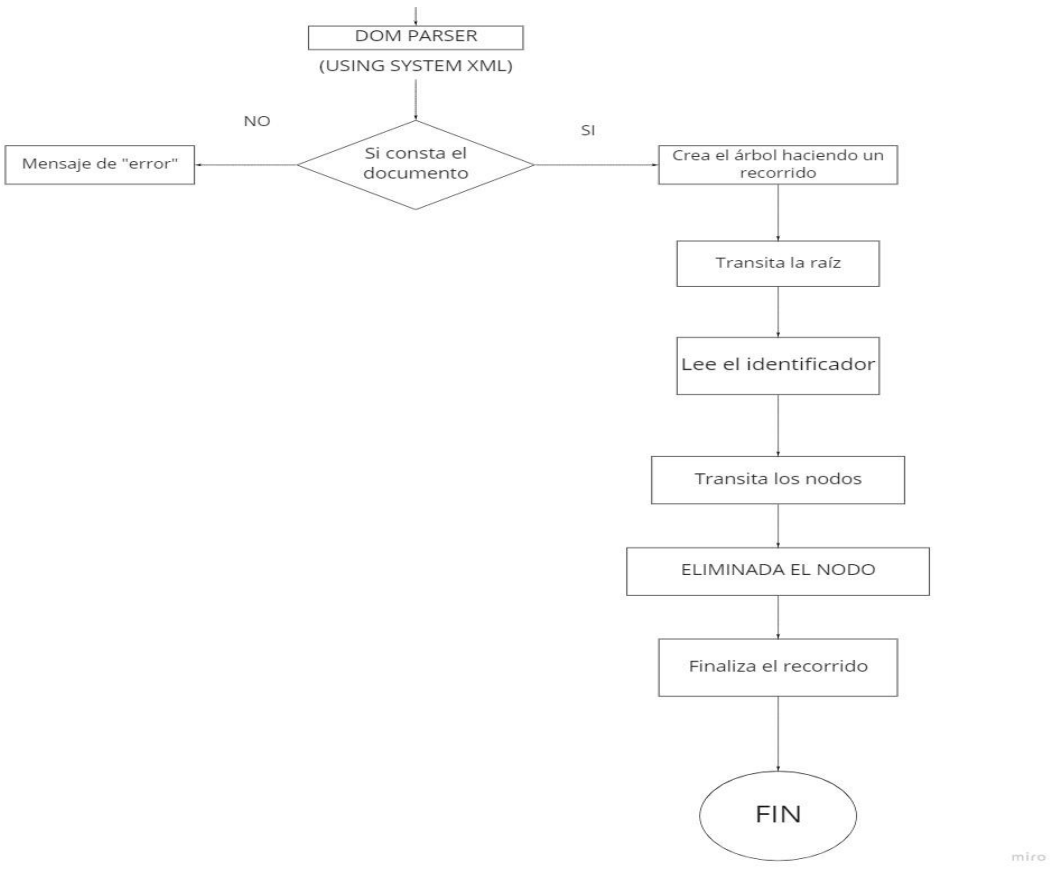
```
<?xml version="1.0" encoding="UTF-8"?>
<Empleados>
  <persona>
    <id>2</id>
    <nombre>SEBAS</nombre>
    <apellidos>JATIVA</apellidos>
    <direccion>Universitario</direccion>
  </persona>
  <persona>
    <id>3</id>
    <nombre>SAMY</nombre>
    <apellidos>GUERRERO</apellidos>
    <direccion>Universitario</direccion>
  </persona>
  <persona>
    <id>4</id>
    <nombre>MAIK</nombre>
    <apellidos>SANTILLAN</apellidos>
    <direccion>Universitario</direccion>
  </persona>
  <persona>
    <id>5</id>
    <nombre>DIEGO</nombre>
    <apellidos>LOGROÑO</apellidos>
    <direccion>Universitario</direccion>
  </persona>
  <persona>
    <id>6</id>
    <nombre>ROMERO</nombre>
    <apellidos>ROMERO</apellidos>
    <direccion>Universitario</direccion>
  </persona>
</Empleados>
```

3. **Decidir la estrategia para leer el documento XML (DOM o SAX).**

   Decidimos usar la estrategia DOM con la ayuda del lenguaje c# para su realización.

4. **Elaborar un diagrama de flujo en el que se describa el proceso de lectura del documento XML bajo uno de los dos enfoques antes mencionados (DOM o SAX).**





5. **Describir de manera general el IDE (entorno de desarrollo integrado) a utilizar**.

   Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Monaco.

   Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.

6. **Codificar el parser.**

   - **Clase Principal (añadir datos al Xml de forma manual)**

```
using System;

namespace XM_OPERACIONES
       {
        class Program
              {
              static void Main(string[] args)
              {
             Xml mixml = new Xml();
             mixml._crearXml("Datos.xml", "Empleados");

             mixml._Añadir("1", "Jhon", "CUVI", "Universitario");
             mixml._Añadir("2", "SEBAS", "JATIVA", "Universitario");
             mixml._Añadir("3", "SAMY", "GUERRERO", "Universitario");
             mixml._Añadir("4", "MAIK", "SANTILLAN", "Universitario");
             mixml._Añadir("5", "DIEGO", "LOGROÑO", "Universitario");
             mixml._Añadir("6", "ROMERO", "ROMERO", "Universitario");

             Console.WriteLine("\t\t  Parser para extraer información de un documento
             XML\n");
             Console.WriteLine("\n-------------------------------------------------
             -----------");
             Console.WriteLine("\n\t  LECTURA DE UN XML MEDIANTE PARSERDOM");
             mixml._ReadXml();
             mixml._DeleteNodo("1");

             Console.WriteLine("\n\n\t  ELIMINANDO A ID: 1 y MOFICANDO XML");
```

```
                    Console.WriteLine("\n-----------------------------------
                ------------------------");

                mixml._ReadXml();

                Console.ReadKey();

                    }
                }
            }
```

- **Clase XML**

```
using System;
using System.Xml;
class Xml
{

    XmlDocument doc;
    string rutaXml;

    public void _crearXml(string ruta, string nodoRaiz)
    {

        this.rutaXml = ruta;
        doc = new XmlDocument();

        XmlDeclaration xmlDeclaration = doc.CreateXmlDeclaration("1.0", "UTF-8", null);

        XmlNode root = doc.DocumentElement;
        doc.InsertBefore(xmlDeclaration, root);


        XmlNode element1 = doc.CreateElement(nodoRaiz);
        doc.AppendChild(element1);
        doc.Save(ruta);
    }

    public void _Añadir(string id, string nom, string ape, string dir)
    {
        doc.Load(rutaXml);

        XmlNode empleado = _Crear_Empleado(id, nom, ape, dir);

        XmlNode nodoRaiz = doc.DocumentElement;

        nodoRaiz.InsertAfter(empleado, nodoRaiz.LastChild);

        doc.Save(rutaXml);


    }

    private XmlNode _Crear_Empleado(string id, string nom, string ape, string dir)
    {

        XmlNode empleado = doc.CreateElement("persona");


        XmlElement xid = doc.CreateElement("id");
        xid.InnerText = id;
        empleado.AppendChild(xid);
```

```
            XmlElement xnombre = doc.CreateElement("nombre");
            xnombre.InnerText = nom;
            empleado.AppendChild(xnombre);


            XmlElement xapellidos = doc.CreateElement("apellidos");
            xapellidos.InnerText = ape;
            empleado.AppendChild(xapellidos);


            XmlElement xdireccion = doc.CreateElement("direccion");
            xdireccion.InnerText = dir;
            empleado.AppendChild(xdireccion);


            return empleado;
        }

        public void _ReadXml()
        {

            doc.Load(rutaXml);

            XmlNodeList listaEmpleados = doc.SelectNodes("Empleados/persona");

            XmlNode unEmpleado;

            for (int i = 0; i < listaEmpleados.Count; i++)
            {

                unEmpleado = listaEmpleados.Item(i);

                string id = unEmpleado.SelectSingleNode("id").InnerText;
                string nombre = unEmpleado.SelectSingleNode("nombre").InnerText;
                string apellidos = unEmpleado.SelectSingleNode("apellidos").InnerText;

                Console.Write("Id: {0}, Nombre: {1}, Apellidos: {2}\n", id, nombre,
apellidos);

            }


        }

        public void _DeleteNodo(string id_borrar)
        {
            doc.Load(rutaXml);

            XmlNode empleados = doc.DocumentElement;

            XmlNodeList listaEmpleados = doc.SelectNodes("Empleados/persona");

            foreach (XmlNode item in listaEmpleados)
            {

                if (item.SelectSingleNode("id").InnerText == id_borrar)
                {

                    XmlNode nodoOld = item;

                    empleados.RemoveChild(nodoOld);
                }
            }

            doc.Save(rutaXml);
```

```
}

public void _UpdateXml(string id_update, string nom, string ape, string dir)
{

    XmlElement empleados = doc.DocumentElement;

    XmlNodeList listaEmpleados = doc.SelectNodes("Empleados/persona");

    XmlNode nuevo_empleado = _Crear_Empleado(id_update, nom, ape, dir);

    foreach (XmlNode item in listaEmpleados)
    {

        if (item.FirstChild.InnerText == id_update)
        {
            XmlNode nodoOld = item;
            empleados.ReplaceChild(nuevo_empleado, nodoOld);

        }
    }

    doc.Save(rutaXml);
}
```

7. Imprimir las capturas de pantalla de la salida del programa.

```
using System;

namespace XM_OPERACIONES
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            Xml mixml = new Xml();
            mixml._crearXml("Datos.xml", "Empleados");

            mixml._Añadir("1", "Jhon", "CUVI", "Universitario");
            mixml._Añadir("2", "SEBAS", "JATIVA", "Universitario");
            mixml._Añadir("3", "SAMY", "GUERRERO", "Universitario");
            mixml._Añadir("4", "MAIK", "SANTILLAN", "Universitario");
            mixml._Añadir("5", "DIEGO", "LOGROÑO", "Universitario");
            mixml._Añadir("6", "ROMERO", "ROMERO", "Universitario");

            Console.WriteLine("\t\t  Parser para extraer información de un documento XML\n");
            Console.WriteLine("\n-----------------------------------------------------------------");
            Console.WriteLine("\n\t  LECTURA DE UN XML MEDIANTE PARSERDOM");
            mixml._ReadXml();
            mixml._DeleteNodo("1");

            Console.WriteLine("\n\n\t  ELIMINANDO A ID: 1 y MOFICANDO XML");
            Console.WriteLine("\n-----------------------------------------------------------------");

            mixml._ReadXml();

            Console.ReadKey();

        }
    }
}
```

No se encontraron problemas. Línea: 30 Carácter: 31 SPC CRLF

```
Consola de depuración de Microsoft Visual Studio
                Parser para extraer información de un documento XML


-------------------------------------------------------------

          LECTURA DE UN XML MEDIANTE PARSERDOM
Id: 1, Nombre: Jhon, Apellidos: CUVI
Id: 2, Nombre: SEBAS, Apellidos: JATIVA
Id: 3, Nombre: SAMY, Apellidos: GUERRERO
Id: 4, Nombre: MAIK, Apellidos: SANTILLAN
Id: 5, Nombre: DIEGO, Apellidos: LOGROÑO
Id: 6, Nombre: ROMERO, Apellidos: ROMERO


          ELIMINANDO A ID: 1 y MOFICANDO XML

-------------------------------------------------------------
Id: 2, Nombre: SEBAS, Apellidos: JATIVA
Id: 3, Nombre: SAMY, Apellidos: GUERRERO
Id: 4, Nombre: MAIK, Apellidos: SANTILLAN
Id: 5, Nombre: DIEGO, Apellidos: LOGROÑO
Id: 6, Nombre: ROMERO, Apellidos: ROMERO
1
C:\Users\diego\Desktop\XM-OPERACIONES\bin\Debug\netcoreapp3.1\XM-OPERACIONES.exe (proceso 16704) se cerró con el código
0.
Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->
Cerrar la consola automáticamente al detenerse la depuración.
Presione cualquier tecla para cerrar esta ventana. . .
```

6. RESULTADOS OBTENIDOS

Como resultado obtenido vemos que el programa realizado mediante el id de Visual Studio en el lenguaje de c# funciona correctamente con lo planteado en un inicio. Se pudo observar la ejecución del programa con cada uno de los estudiantes y su información personal, posterior a ello se genero un archivo xml con el nombre de datos y fue todo un éxito. Vimos como funciona el programa y su procedencia. Gracias a ello se gano conocimiento con respecto al tema de XML y la estrategia escogida a utilizar DOM.

7. CONCLUSIONES

Como conclusión se puede decir que los archivos XML son de suma importancia ya que en una empresa o cualquier tipo de institución es vital el manejo diario de estos. A su ves como programadores y creadores de aplicaciones o sistemas informáticos, es muy fundamental conocer como se realiza este tipo de archivos y como corregir errores, como también el mantenimiento y mejoras a un programa, en este caso cuidar la integridad de los datos a través de los conocimientos adquiridos.

8. RECOMENDACIONES

- Se recomienda utilizar los parámetros en la estructura en el lenguaje C# como por ejemplo las librerías “using System.xml” y la implementación de las clases necesarias.
- Se recomienda investigar mas acerca de la implementación de la estrategia SAX, por motivo que este trabajo se realizó acerca de la estrategia DOM.
- Se recomienda establecer bien los parámetros como por ejemplo las clases, los objetos, las variables, la unidad donde se va a establecer el archivo xml, etc , para su correcta ejecución