Integrantes: Agustina Waigel, Felipe Correa, Dylan Gross, Valentino Badaracco y Agustin Zuleiman.

Universidad Adventista del Plata
Ingeniería en Sistemas

Ingeniería de software II

Trabajo práctico nro. 2

En grupos de hasta 6 personas, lean atentamente el siguiente párrafo, que detalla una situación de trabajo con la que un ingeniero en sistemas puede enfrentarse, y luego, realice las actividades planteadas en las consignas. La entrega se hará a través de un documento .pdf, en la primera página deben figurar los integrantes del grupo con apellido y nombre. La presentación de este debe tener una estructura definida, consistente y concisa. Se descontarán puntos por mala redacción, errores ortográficos y una pobre estructuración del documento.

Situación:

La aseguradora "SegurAr SA" quiere implementar un sistema de gestión nuevo y centralizar la gestión de los datos recopilados a lo largo de los 30 años. Para ello, contrataron a un grupo de ingenieros en sistemas egresados de la Universidad Adventista del Plata. Luego de algunas reuniones con el encargado de comunicar los requerimientos para el sistema a desarrollar, el grupo de ingenieros pregunta, en buena hora, si se requiere además del desarrollo del sistema, la migración de los datos actuales al sistema nuevo: la respuesta es un rotundo sí. Luego de la reunión, nos vamos con un listado de requerimientos y la información de que existen datos a migrar de una base de datos legacy con 30 tablas, fichas físicas de clientes junto con el detalle del plan de seguros que han contratado y archivos de texto plano con el formato CSV (valores separados por coma).

Por ejemplo, un archivo CSV podría verse así:

"Nombre","Edad","Ciudad"
"Juan Pérez",28,"Madrid"
"Ana García",35,"Barcelona"
"Carlos López",42,"Valencia"

Donde la primera línea del archivo especifica el nombre de cada campo y las siguientes líneas son los valores de los campos correspondientes de cada registro.

Teniendo en cuenta que estimamos la misma cantidad de tiempo para el desarrollo del nuevo sistema y para la migración de datos, formule un plan para la migración de datos teniendo en cuenta las siguientes consignas:

1. Suponiendo un presupuesto total de $30.000 dólares (desarrollo de software + migración de datos) detallar cuál será el costo de desarrollo de software y cuál será el costo de la migración de datos.

2. En el plan, debe estar especificado en pseudocódigo o un script cómo hacer para transformar un archivo con formato CSV a una tabla de una base de datos relacional. En el script debe estar definida la creación de la tabla con sus respectivos campos y la inserción de los datos.

3. Suponiendo que la base de datos legacy admite la exportación de datos a formato CSV. La tabla “clientes” tiene un campo DNI que admite un tipo de dato “int”, es decir, números enteros. Uno de los requerimientos del nuevo sistema es admitir distintos medios de identificación, tales como pasaporte o documentos de identificación de otros países. Estos distintos medios de identificación pueden contener caracteres que no sean números. Detallar, con pseudocódigo o un script, cómo haría para admitir este nuevo requerimiento en la base de datos nueva.

4. Existe un archivo de texto plano con formato CSV con datos de clientes de años previos al 2000, el dato del año de nacimiento del cliente utiliza solo dos bytes, es decir, los últimos dos números del año en que nació. Ejemplo: “añoDeNacimiento”, “97”. Detallar en pseudocódigo o script cómo haría para migrar los datos de dicho archivo con formato CSV a la base de datos nueva, concatenando el número 19 al dato del año de nacimiento. Ejemplo: “19” + “97” = “1997”.

5. Algunas de las fichas físicas tienen datos incompletos. ¿Cómo harían para tratarlas?

Recuerden que este TP no tiene respuestas definidas, se evaluará tanto la creatividad como la efectividad de los planes presentados.

DESARROLLO

1.
Este problema lo resolvimos buscando e investigando cuáles eran los precios de las distintas tareas en desarrollo de software y migración de datos.

Desarrollo del Software
Capacitación en Tecnologías Requeridas: $1,500
Herramientas y Licencias de Software: $800
Reuniones y Consultas con Expertos: $800
Planificación y Análisis de Requisitos: $1,500
Diseño del Sistema: $1,000
Salarios para el Equipo de Desarrollo: $7,000
Infraestructura de Desarrollo: $800
Pruebas de Calidad y Seguridad: $800
Pruebas y Aseguramiento de Calidad: $1,500
Documentación del Sistema: $800
Capacitación de Usuarios: $800
Subtotal Desarrollo del Software: $17,300

Migración de Datos
Capacitación en la Base de Datos Legacy: $1,500
Contratación de un Experto en la Base de Datos Legacy: $2,500
Extracción de Datos de la Base de Datos Legacy: $1,500
Digitalización de Fichas Físicas: $1,500
Herramientas ETL y Procesos de Limpieza de Datos: $2,000
Configuración de la Nueva Base de Datos: $800
Carga de Datos y Validación: $800
Pruebas de Integridad y Consistencia: $800
Documentación del Proceso de Migración: $800
Soporte y Mantenimiento Inicial: $500
Subtotal Migración de Datos: $12,700

Resumen del Presupuesto Ajustado
Desarrollo de Software: $17,300
Migración de Datos: $12,700
Total: $30,000

2.

Pseudocódigo:

Inicio del programa

Conectar a la base de datos SQL Server:

Utilizar el controlador específico y proporcionar los detalles de la conexión (servidor, base de datos, usuario, contraseña).

Crear la tabla "clientes":

Crear una nueva tabla con los siguientes campos:

- **Nombre**: cadena de caracteres de hasta 255 caracteres.
- **Edad**: entero.
- **Ciudad**: cadena de caracteres de hasta 255 caracteres.

Abrir el archivo CSV "clientes.csv":

Leer el contenido del archivo línea por línea.

Saltar la primera línea (cabecera) del archivo CSV.

Para cada línea en el archivo CSV:

Separar los valores de la línea en campos individuales.

Validar y convertir los datos según el tipo de cada campo:

- ***Nombre***: cadena de caracteres.
- ***Edad***: convertir a entero.
- ***Ciudad***: cadena de caracteres.

Insertar los datos validados en la tabla "clientes" de la base de datos:

- ***Nombre, Edad, Ciudad.***

Confirmar los cambios realizados en la base de datos.

Cerrar la conexión a la base de datos.

Fin del programa

Script:

Creación de un script para transformar un archivo CSV en una tabla en la base de datos: Para esto decidimos utilizar el lenguaje Python ya que es el que estamos usando actualmente y para la Base de datos elegimos SQLServer por el mismo motivo.

Para poder importar un archivo CSV a python se debe importar la librería “csv” y para poder conectar la Base de datos se debe importar la librería “pyodbc”.
También decidimos que debíamos utilizar “try-except” para manejar posibles errores y verificar y/o convertir el tipo de dato que se migra del csv a la base de datos.

```python
import csv
#para manejar SQLServer
import pyodbc

#try esta puesto para el manejo de errores
try:
    #Conexión a la base de datos SQL Server
    conexion = pyodbc.connect('DRIVER={SQL Server};SERVER=servidor;DATABASE=SegurAr;UID=usuario;PWD=contraseña')
    cursor = conexion.cursor()

    #Crear la tabla en SQLServer
    cursor.execute('''
    CREATE TABLE clientes (
            Nombre VARCHAR(255),
            Edad INT,
            Ciudad VARCHAR(255)
            )
    ''')

    #Abrir el archivo CSV y leer su contenido
    with open('clientes.csv', 'r') as archivo_csv:
        lectura_csv = csv.reader(archivo_csv)
        next(lectura_csv)  #Saltar la primera linea
        for fila in lectura_csv:
            try:
                #Validar y convertir los datos, para que no haya errores
                nombre = fila[0]
                edad = int(fila[1])  #Convertir a entero
                ciudad = fila[2]
```

```python
                #Insertar cada fila en la tabla
                cursor.execute('INSERT INTO clientes (Nombre, Edad, Ciudad) VALUES (?, ?, ?)', (nombre, edad, ciudad))
            except ValueError as ve:
                print(f"Error al convertir datos: {ve}. Fila: {fila}")
            except pyodbc.Error as db_err:
                print(f"Error al insertar en la base de datos: {db_err}. Fila: {fila}")

    #Confirmar y cerrar
    conexion.commit()
except pyodbc.Error as e:
    print(f"Error de conexión o ejecución: {e}")
finally:
    if conexion:
        conexion.close()
```

3.
Para llevar a cabo este problema, primero agregaremos el atributo DNI en la tabla de la base de datos con el tipo de dato VARCHAR, luego vamos a tomar el valor que corresponde al DNI del archivo CSV (linea por linea) y vamos a convertir ese valor a string; cuando tenemos ese nuevo valor que en este caso se almacena en una variable “dni”, se inserta en la tabla de la base de datos.

El formato CSV suele trabajar con valores de tipo string cuando encierra ese valor entre comillas, pero para evitar errores optamos por la conversión del valor a string igualmente.

Pseudocódigo:

Inicio del programa

Conectar a la base de datos SQL Server:

Utilizar el controlador específico y proporcionar los detalles de la conexión (servidor, base de datos, usuario, contraseña).

Crear la tabla "clientes":

Crear una nueva tabla con los siguientes campos:

- **Nombre**: cadena de caracteres de hasta 255 caracteres.
- **Edad**: entero.
- **Ciudad**: cadena de caracteres de hasta 255 caracteres.
- **DNI:** cadena de caracteres de hasta 30 caracteres.

Abrir el archivo CSV "clientes.csv":

Leer el contenido del archivo línea por línea.

Saltar la primera línea (cabecera) del archivo CSV.

Para cada línea en el archivo CSV:

Separar los valores de la línea en campos individuales.

Validar y convertir los datos según el tipo de cada campo:

- ***Nombre***: cadena de caracteres.
- ***Edad***: convertir a entero.
- ***Ciudad***: cadena de caracteres.
- **DNI:** convertir a cadena de caracteres.

Insertar los datos validados en la tabla "clientes" de la base de datos:

- ***Nombre, Edad, Ciudad, DNI.***

Confirmar los cambios realizados en la base de datos.

Cerrar la conexión a la base de datos.

Fin del programa

Script:

```python
import csv
#para manejar SQLServer
import pyodbc

#try esta puesto para el manejo de errores
try:
    #Conexión a la base de datos SQL Server
    conexion = pyodbc.connect('DRIVER={SQL Server};SERVER=servidor;DATABASE=SegurAr;UID=usuario;PWD=contraseña')
    cursor = conexion.cursor()

    #Crear la tabla en SQLServer
    cursor.execute('''
    CREATE TABLE clientes (
            Nombre VARCHAR(255),
            Edad INT,
            Ciudad VARCHAR(255)
            DNI VARCHAR(30)
            )
    ''')

    #Abrir el archivo CSV y leer su contenido
    with open('clientes.csv', 'r') as archivo_csv:
        lectura_csv = csv.reader(archivo_csv)
        next(lectura_csv)  #Saltar la primera linea
        for fila in lectura_csv:
            try:
                #Validar y convertir los datos, para que no haya errores
                nombre = fila[0]
                edad = int(fila[1])  #Convertir a entero
                ciudad = fila[2]
                dni = str(fila[3])
```

```python
                #Insertar cada fila en la tabla
                cursor.execute('INSERT INTO clientes (Nombre, Edad, Ciudad, DNI) VALUES (?, ?, ?, ?)', (nombre, edad, ciudad, dni))
            except ValueError as ve:
                print(f"Error al convertir datos: {ve}. Fila: {fila}")
            except pyodbc.Error as db_err:
                print(f"Error al insertar en la base de datos: {db_err}. Fila: {fila}")

    #Confirmar y cerrar
    conexion.commit()
except pyodbc.Error as e:
    print(f"Error de conexión o ejecución: {e}")
finally:
    if conexion:
        conexion.close()
```

4.
Desarrollo del problema: primero deberíamos agregar un atributo para el año de nacimiento a la tabla de la base de datos de tipo INT, luego (linea por linea) tomar ese valor del archivo CSV y convertirlo a entero, luego decidimos que para que el valor final sea de 4 números (ej: 1976) sumariamos el valor tomado del archivo CSV y le sumariamos 1900; por último agregar ese valor a la base de datos.

Pseudocódigo:

Inicio del programa

Conectar a la base de datos SQL Server:

Utilizar el controlador específico y proporcionar los detalles de la conexión (servidor, base de datos, usuario, contraseña).

Crear la tabla "clientes":

Crear una nueva tabla con los siguientes campos:

- **Nombre**: cadena de caracteres de hasta 255 caracteres.
- **Edad**: entero.
- **Ciudad**: cadena de caracteres de hasta 255 caracteres.
- **DNI:** cadena de caracteres de hasta 30 caracteres.
- **AnoDeNacimiento:** entero

Abrir el archivo CSV "clientes.csv":

Leer el contenido del archivo línea por línea.

Saltar la primera línea (cabecera) del archivo CSV.

Para cada línea en el archivo CSV:

Separar los valores de la línea en campos individuales.

Validar y convertir los datos según el tipo de cada campo:

- ***Nombre***: cadena de caracteres.
- ***Edad***: convertir a entero.
- ***Ciudad***: cadena de caracteres.
- **DNI:** convertir a cadena de caracteres.
- **AñoDeNacimiento:** convertir a entero, sumar ese valor a 1900

Insertar los datos validados en la tabla "clientes" de la base de datos:

- ***Nombre, Edad, Ciudad, DNI, AnoDeNacimiento.***

Confirmar los cambios realizados en la base de datos.

Cerrar la conexión a la base de datos.

Fin del programa

Script:

```
import csv
#para manejar SQLServer
import pyodbc

#try esta puesto para el manejo de errores
try:
    #Conexión a la base de datos SQL Server
    conexion = pyodbc.connect('DRIVER={SQL Server};SERVER=servidor;DATABASE=SegurAr;UID=usuario;PWD=contraseña')
    cursor = conexion.cursor()

    #Crear la tabla en SQLServer
    cursor.execute('''
    CREATE TABLE clientes (
            Nombre VARCHAR(255),
            Edad INT,
            Ciudad VARCHAR(255)
            DNI VARCHAR(30)
            AnoDeNacimiento INT
            )
    ''')

    #Abrir el archivo CSV y leer su contenido
    with open('clientes.csv', 'r') as archivo_csv:
        lectura_csv = csv.reader(archivo_csv)
        next(lectura_csv)  #Saltar la primera linea
        for fila in lectura_csv:
            try:
                #Validar y convertir los datos, para que no haya errores
                nombre = fila[0]
                edad = int(fila[1])   #Convertir a entero
                ciudad = fila[2]
                dni = str(fila[3])    #Convertir a string
                año_de_nacimiento = int(fila[4])    #Convertir a entero
                año_de_nacimiento = 1900 + año_de_nacimiento
```

```
                #Insertar cada fila en la tabla
                cursor.execute(
                    'INSERT INTO clientes (Nombre, Edad, Ciudad, DNI, AnoDeNacimiento) VALUES (?, ?, ?, ?, ?)',
                    (nombre, edad, ciudad, dni, año_de_nacimiento))

            except ValueError as ve:
                print(f"Error al convertir datos: {ve}. Fila: {fila}")
            except pyodbc.Error as db_err:
                print(f"Error al insertar en la base de datos: {db_err}. Fila: {fila}")

    #Confirmar y cerrar
    conexion.commit()
except pyodbc.Error as e:
    print(f"Error de conexión o ejecución: {e}")
finally:
    if conexion:
        conexion.close()
```

5.
Para tratar las fichas con datos incompletos, intentaremos recuperar los datos faltantes. Lo que haremos para recuperar los datos es realizar un sorteo que consta de 12 meses gratis de seguro, que para participar se deben acercar a los locales de la ciudad a la que pertenecen y completarlo con nombre completo, edad y ciudad a la que pertenecen. Este sorteo será lanzado durante un mes a través de las redes sociales de la página (Instagram y Facebook) y en un anuncio televisivo.