3K1 Parcial Recuperatorio - Backend de Aplicaciones 2024

Objetivo

Poner en práctica lo aprendido acerca de programación en Java y desarrollo de Microservicios. Para el desarrollo del presente parcial será necesario conocer:

- El lenguaje de programación Java y sus diferentes librerías.
- El administrador de proyectos Maven y su configuración y uso.

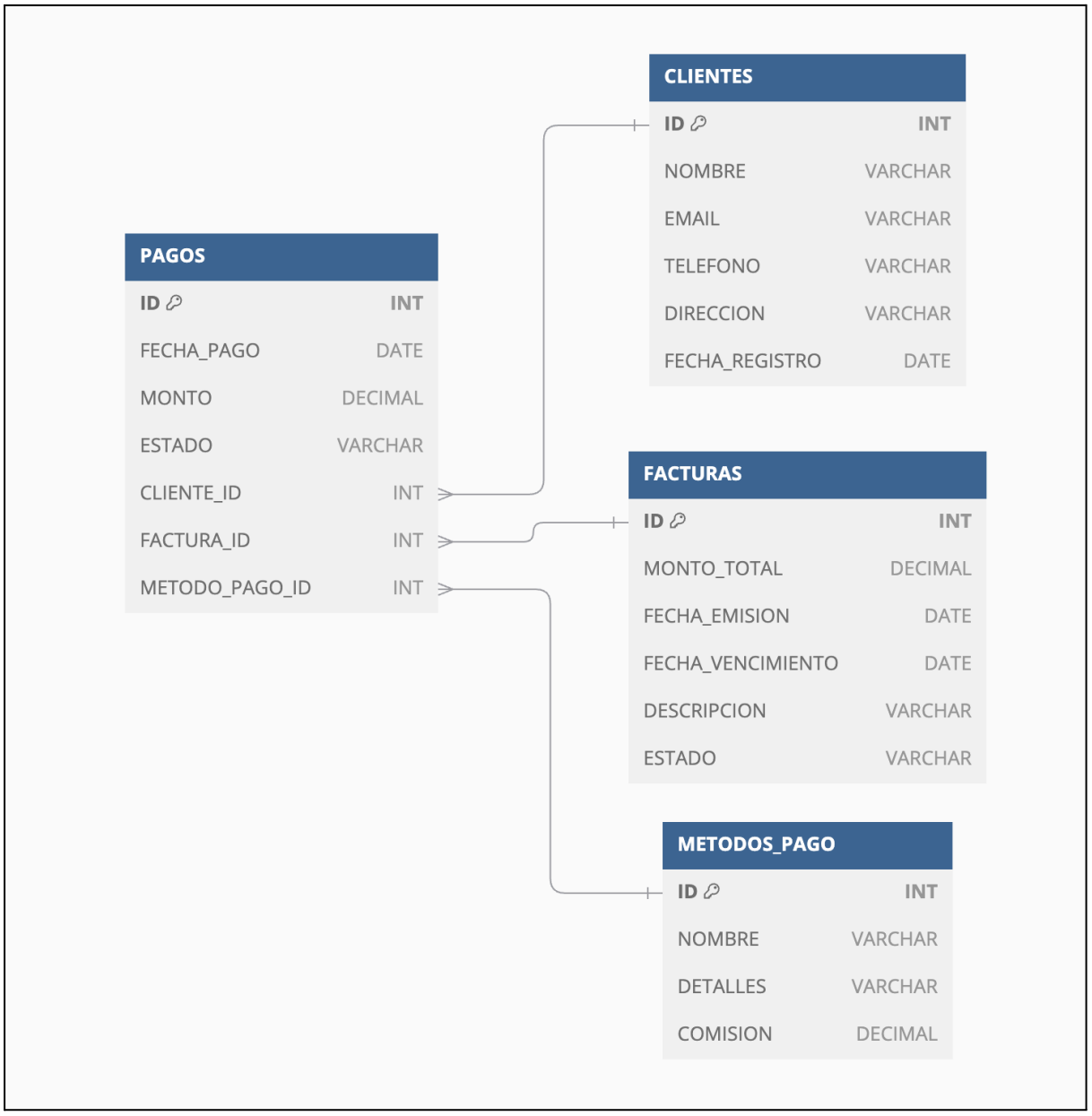
En cuanto al objetivos del presente pre-enunciado es lograr que el estudiante pueda llevar a cabo la configuración del proyecto y la construcción de los elementos básicos sin la presión del tiempo del parcial para que luego el día del parcial pueda dedicarse a programar lo solicitado que tendrá foco en programar en java para cumplir los objetivos de un par de requerimientos extra.

Tienen las herramientas necesarias para realizar las entidades de datos, los mapeos, el parseo del CSV y asegurarse que la base de datos quede cargada con los elementos correctos.

Sistema de Gestión de Pagos y Facturación

Una empresa de servicios requiere un sistema que les permita administrar los pagos realizados por sus clientes, vinculándolos con las facturas emitidas. Para esto, cuentan con un archivo CSV denominado `pagos.csv`.

A continuación se presenta el DER de la Base de datos que tendrán que popular con los datos del archivo CSV:



Pre-Enunciado

1. Cargar el archivo CSV a una estructura de datos en memoria: Una vez definido el modelo de clases con el mapeo a la base de datos, se espera que usted cree instancias de esas clases a partir de los datos del archivo CSV. Para esto dejamos algunas consideraciones a tener en cuenta:
    - Cada fila del archivo contiene información completa para poblar las tablas **Pagos**, **Clientes**, **Facturas**, y **Métodos de Pago** [Total de filas a cargar 85].
    - Se usa un **prefijo** para identificar las columnas asociadas a cada tabla:
        - `PAGO_` para las columnas de la tabla **Pagos**.
        - `CLIENTE_` para las columnas de la tabla **Clientes**.
        - `FACTURA_` para las columnas de la tabla **Facturas**.

    - `METODO_PAGO_` para las columnas de la tabla **Métodos de Pago**.
- El archivo utiliza `|` como separador.
- Cada línea del archivo CSV se transformará en una instancia de la clase **Pago**, que estará vinculada con instancias de las clases relacionadas: **Cliente**, **Factura**, y **MétodoPago**. Aquí hay consideraciones específicas para cada caso:
    - **Cliente**:
        - El `CLIENTE_ID` será el identificador único de cada cliente.
        - Si un cliente ya existe en memoria (identificado por su `CLIENTE_ID`), se reutilizará la instancia correspondiente. Si no, se creará una nueva instancia con los valores de la línea actual.
        - Los campos como `EMAIL`, `TELEFONO` y `DIRECCION` se establecerán como `null` si están vacíos en el archivo CSV.
    - **Factura**:
        - El `FACTURA_ID` será el identificador único de cada factura.
        - Si una factura ya existe en memoria (identificada por su `FACTURA_ID`), se reutilizará la instancia. Si no, se creará una nueva.
        - Asegúrate de que las fechas (`FECHA_EMISION` y `FECHA_VENCIMIENTO`) estén en el formato correcto (`YYYY-MM-DD`) o se establecerán como `null`.
    - **Método de Pago**:
        - El `METODO_PAGO_ID` será el identificador único de cada método de pago.
        - Si un método de pago ya existe en memoria, se reutilizará. Si no, se creará una nueva instancia.
        - Los campos como `DETALLES` y `COMISION` se establecerán como `null` o `0.0` si están vacíos.
    - **Pago**:
        - Cada línea del archivo corresponde a una instancia única de la clase **Pago**.
        - Cada instancia incluirá referencias a las clases relacionadas (**Cliente**, **Factura**, y **MétodoPago**) previamente procesadas.
- **Aclaraciones sobre el parseo de fechas y montos**
    - **Fechas**: Los campos `FECHA_PAGO`, `FECHA_EMISION` y `FECHA_VENCIMIENTO` están en formato `YYYY-MM-DD` y debe ser parseado a objetos de tipo `Date`. Si una fecha tiene un formato inválido, debe manejarse una excepción y registrarse el error.

```
// Formato esperado (YYYY-MM-DD)
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd", Locale.US);

try {
    // Parsear la fecha
    Date fecha = formatter.parse(fechaTexto);
    System.out.println("Fecha parseada correctamente: " + fecha);
```

```
    // Formatear la fecha a otro formato (DD/MM/YYYY)
    SimpleDateFormat nuevoFormato = new SimpleDateFormat("dd/MM/yyyy");
    String fechaFormateada = nuevoFormato.format(fecha);

} catch (ParseException e) {
    // Manejo de errores si la fecha tiene formato incorrecto
    System.err.println("Error al parsear la fecha: " + fechaTexto);
}
```

- **Montos**: El campo `MONTO` está en formato decimal (con `.` como separador de decimales). Debe convertirse a objetos `BigDecimal` para garantizar precisión en las operaciones financieras. En caso de un formato inválido, se sugiere registrar el error y decidir cómo manejarlo.

```
NumberFormat nf = NumberFormat.getInstance(Locale.US);

try {

    BigDecimal monto = new BigDecimal(nf.parse(montoTexto).toString());
    System.out.println("Monto parseado correctamente: " + monto);

} catch (ParseException e) {
    System.err.println("Error al parsear el monto: " + montoTexto);
}
```

2. Popular la base de datos con todos los objetos cargados en memoria teniendo en cuenta las siguiente consideraciones:
    - Cada **cliente** único corresponderá a una fila en la tabla `CLIENTES`. Los pagos estarán relacionados con los clientes mediante el campo `CLIENTE_ID`.
    - Cada **factura** única corresponderá a una fila en la tabla `FACTURAS`. Los pagos estarán relacionados con las facturas mediante el campo `FACTURA_ID`.
    - Cada **método de pago** único corresponderá a una fila en la tabla `METODOS_PAGO`. Los pagos estarán relacionados con los métodos mediante el campo `METODO_PAGO_ID`.
    - Cada **pago** en memoria corresponderá a una fila en la tabla `PAGOS`.

Algunos comentarios:

- Tenga en cuenta que todo lo que usted programe le servirá para enfrentar los requerimientos el día del parcial, es decir mientras más acabado tenga la estructura CRUD para estas tablas más simple será la programación pedida el día del parcial.
- Por otro lado, tenga en cuenta también a modo de información previa que los requerimientos del día del parcial serán, una inserción o modificación compuesta, es decir que implique interactuar con más de una entidad y un listado que

también requerirá devolver información procesando más de una tabla/entidad de datos.

Repositorio

1. El alumno dispone, de un repositorio personal en el gitlab de la universidad (tenga en cuenta que no tendrá acceso a repositorios externos).
2. Estos repositorios están disponibles para subir la construcción previa, para clonar el día del parcial.
3. Es responsabilidad de cada alumno el acceso al repositorio personal.

- Esperamos que cada alumno tenga los elementos necesarios para que el día del parcial no tenga contratiempos.

Cátedra de Backend de Aplicaciones