Iteración 3 – Manejo transaccional de información

Carlos Eduardo Rincón, Henry Fabián Vargas

Universidad de los Andes, Bogotá, Colombia

{ce.rincon10, hf.vargas10}@uniandes.edu.co

Fecha de presentación: 25 de septiembre de 2015

1. Introducción

En esta iteración se pretende realizar las fases necesarias para el diseño e implementación de una aplicación web con fines transaccionales, para este fin se lleva a cabo el analisis de requerimientos funcionales y se toman decision de diseño para satisfacerlos, tales como modelo de entidades. La implementacion se basa en un estilo arquitectonico MVC (Modelo, Vista, Controlador) con acceso a una base de datos para los requerimientos transaccionales.

2. Fase de Analisis

2.1. Modelo de entidades

Una vez desarrollada la fase de analisis de los requerimientos funcionales se identificaron las entidades pertientes y las relaciones entre ellas. A partir de este modelo se construyo los controladores para satisfacer las operaciones necesarias para satisfacer los requerimientos. Para esta iteración se modificó el modelo anterior agregando la relación empleados entre las entidades Cliente y Empleado, donde un cliente tipo juridico puede asociar sus empleados.

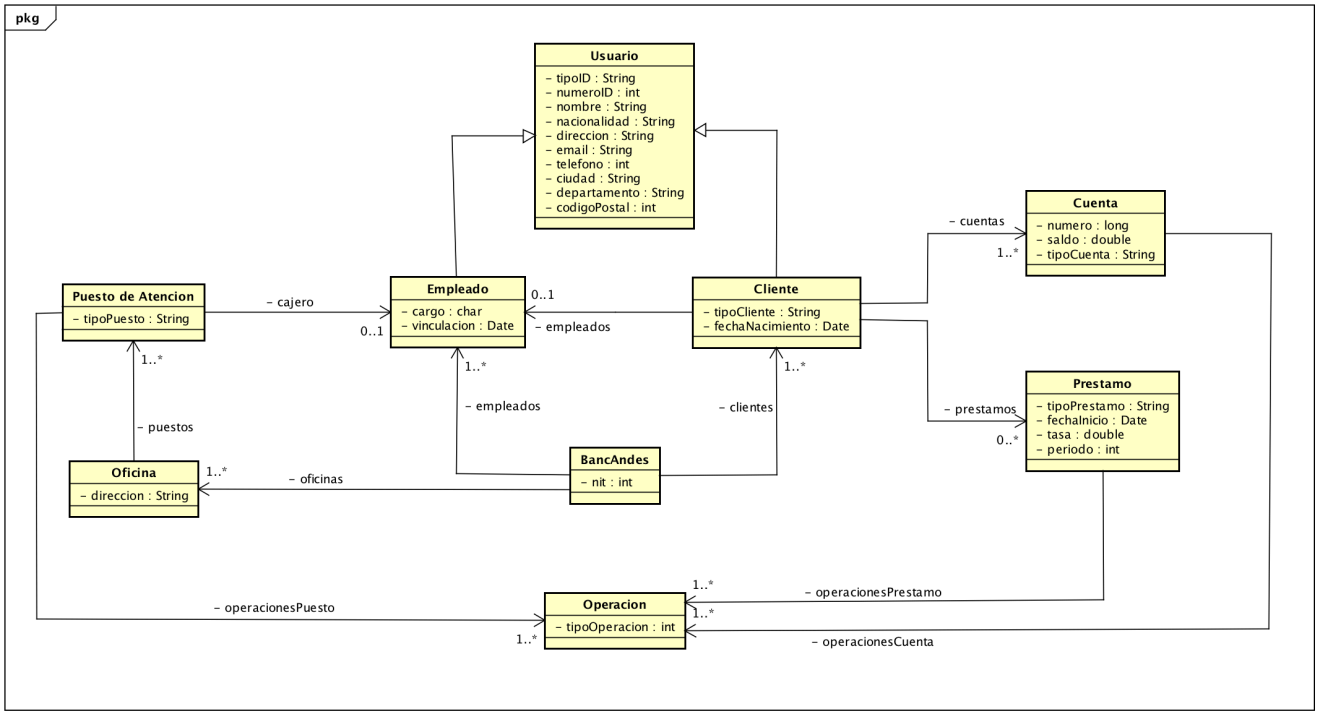

Figura 1. Modelo de entidades(1)

3. Fase de implementacion

3.1.Desarrollo

Para satisfacer el requerimiento de transaccionalidad se implementaron las busquedas de tal forma que una ves se establece la conexión con la base de datos se le da la instrucción `conexion.setAutoCommit(false);` , adicionalmente por cada ejecucion de llamados a la base de datos, se implementó la ejecucion del query dentro de un `try catch`, de tal forma que si la ejecución fue exitosa se hace `commit()`, o `rollback()` de lo contrario.

```
/**
 * Método que se encarga de crear la conexión con el Driver Manager
 * a partir de los parametros recibidos.
 * @param url direccion url de la base de datos a la cual se desea conectar
 * @param usuario nombre del usuario que se va a conectar a la base de datos
 * @param clave clave de acceso a la base de datos
 * @throws SQLException si ocurre un error generando la conexión con la base de datos.
 */
  public Connection establecerConexion() throws SQLException
  {
    try
      {
    conexion = DriverManager.getConnection(cadenaConexion,usuario,clave);
    conexion.setAutoCommit(false);
    return conexion;
      }
      catch( SQLException exception )
      {
          throw new SQLException( "ERROR: ConsultaDAO obteniendo una conexin."+cadenaConexion );
      }
  }
```

Figura 2. Implementacion (1)

```
public boolean cerrarCuenta(long id) throws Exception
{
  Cuenta actual=darCuentaId(id);
  if(actual.getMonto()==0)
  {
    Connection conexion=null;
    try
    {
      conexion=ConsultaDAO.darInstancia().establecerConexion();
      Statement st=conexion.createStatement();
      st.executeUpdate(cerrarCuenta+id);
      conexion.commit();
      return true;
    }
    catch(SQLException e)
    {
      e.printStackTrace();
      System.out.println(cerrarCuenta);
      conexion.rollback();
      throw new Exception("ERROR = ConsultaDAO: loadRowsBy(..) Agregando parametros y executando el statement!!!");
    }
    finally
    {
      ConsultaDAO.darInstancia().closeConnection(conexion);
    }
  }
  else
  {
    return false;
  }
}
```

Figura 3. Implementacion (1)

3.2. Vistas

Para las vistas se implementaron las funcionalidades de estilo css de bootstrap y los servicios de scala

4. Bibliografia

Tutoriales de acceso a los servicios de Oracle: http://sistemas.uniandes.edu.co/~isis2304