

**UNIVERSIDAD CIENTIFICA DEL SUR**

**FACULTAD DE CIENCIAS EMPRESARIALES**

**CARRERA DE INGENIERÍA DE SISTEMAS EMPRESARIALES**

**Asignatura :** TALLER DE PROGRAMACION II

**Profesor :** ING. GUSTAVO CORONEL

**Integrantes :** Karlheinz Fernández Bermúdez

Shirley Castillo Yarleque

Lima, 05 de Febrero del 2018

Proyecto EDUTEC

**INDICE**

Datos Generales 2

Objetivo 2

Alcance 2

Modelo de Datos 3

Requerimientos 3

Seguridad (SEG) 3

Procesos de Negocio (NEG) 3

Mantenimiento de Tablas (MAN) 4

Consultas (CON) 5

Reportes (REP) 5

Software 5

Arquitectura de la Solución 5

Acceso a la Base de Datos 7

Requerimientos de Seguridad 9

REQSEG001: Ingreso al Sistema 9

Diseño del Servicio 9

Modelamiento del Proceso 9

Programación del Servicio 9

Interfaz de Usuario 10

REQSEG002: Cambio de Clave 10

Requerimientos de Negocio 11

REQNEG001: Generación de un nuevo ciclo 11

Diseño del Servicio 11

REQNEG002: Programar Curso 11

# Datos Generales

Curso: Taller de Programación II

Profesor: Gustavo Coronel

Ciclo: VI

Periodo: 2018-0

# Objetivo

**OBJETIVO GENERAL:**

Diseñar un sistema de gestión educacional para el centro educativo EDUTEC que cumplan con las expectativas y las funcionalidades que el cliente requiere.

**OBJETIVO ESPECIFICOS:**

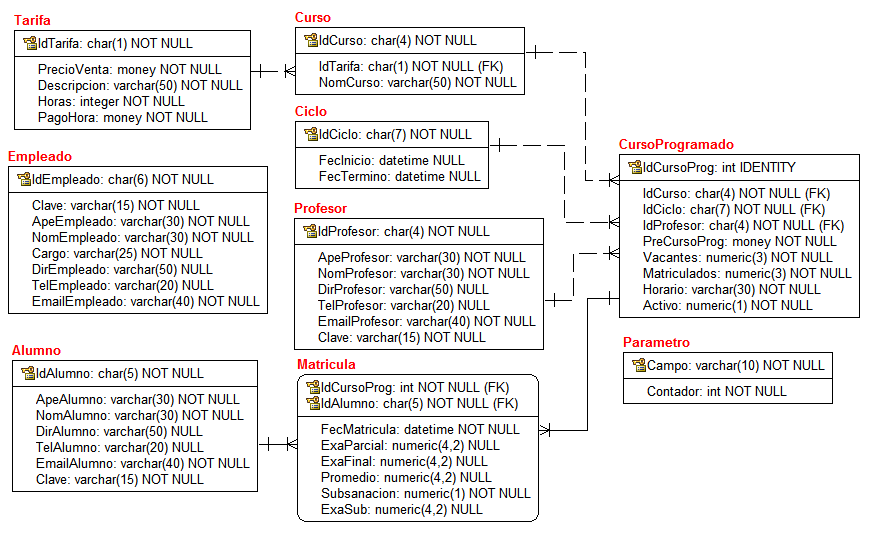
1. Lograr establecer una conexión con la base de datos de la institución EDUTEC de manera que tengamos el historial en nuestro sistema.
2. Reducir el tiempo de atención al usuario, en el proceso de matrícula, registro de notas, registro de alumnos, registro de profesores, eliminación de la matrícula con la implementación del sistema.
3. Optimizar el consumo de recursos con la implementación del sistema.
4. Integrar las tablas para poder fortalecer la recopilación de datos de la empresa y lograr el objetivo general.

# Alcance

El sistema será aplicable en la institución EDUTEC, el cual mejorará la gestión administrativa con respecto a:

* SEGURIDAD
* PROCESOS DE NEGOCIO
* MANTENIMIENTO
* CONSULTAS
* REPORTES

# Modelo de Datos



# Requerimientos

## Seguridad (SEG)

| **CODIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| --- | --- | --- |
| **REQSEG001** | Acceso al sistema | Los empleados deben identificarse utilizando su IdEmpleado y Clave para iniciar sesión en el sistema. |
| **REQSEG002** | Cambio de clave (GRUPO ) | Permite a los empleados cambiar su clave. |

## Procesos de Negocio (NEG)

| **CODIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| --- | --- | --- |
| **REQNEG001** | Generación de un nuevo ciclo (grupo 1) | El proceso genera el siguiente ciclo en función al último que existe en la tabla **CICLO**.  La fecha de inicio es el primer día del mes y la fecha de término es el último día del mes. |
| **REQNEG002** | Programar Curso | * Solo se pueden programar cursos a partir del ciclo actual hacia adelante. * El precio del curso se debe leer de la tabla **TARIFA**. * Las vacantes es un campo editable, pero, por defecto debe mostrar 20. * Por defecto, el curso programado debe estar activado. * Puede ser que no se asigne profesor. |
| **REQNEG003** | Anular curso programado (grupo 2) | Consiste en desactivar el curso programado, en este caso se deben cumplir las siguientes condiciones:   * Solo se permite eliminar cursos del ciclo actual, hacia adelante. * El curso programado debe tener a lo más 5 matriculados.   Este proceso implica eliminar todas sus matrículas, y el campo **IdProfesor** debe establecerse en nulo (NULL). |
| **REGNEG004** | Asignar profesor (grupo 3) | Este proceso consiste en asignar un profesor a un curso programado.  La condición es que un profesor solo puede estar programado a los mas en 3 curso por ciclo. |
| **REQNEG005** | Matricula (grupo 4) | Este proceso permite registrar la matrícula de un estudiante. |
| **REQNEG006** | Registrar notas de curso.(grupo 5 **NOSOTROS**) | Este proceso permite registrar las notas de los estudiantes de un curso programado. |
| **REQNEG007** | Registrar nota de estudiante.(GRUPO 1) | Este proceso permite registrar las notas de un estudiante. |
| **REQNEG008** | Registrar nota de examen de subsanación.(GRUPO 2) | Este proceso permite registrar la nota de examen de subsanación de un estudiante. |
| **REQNEG009** | Eliminar matricula.(GRUPO 3) | Se elimina el registro de la matricula solo en los siguientes casos:   * **Ciclo Actual:** Si el curso tiene a lo más 5 alumnos matriculados. * **Ciclos Futuros:** Procede sin problemas. |
| **REQNEG010** | Cambio de curso.(GRUPO 4) | Este caso permite a un estudiante cambiar de curso, solo procede en el ciclo actual y ciclos futuros. |

## Mantenimiento de Tablas (MAN)

| **CODIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| --- | --- | --- |
| **REQMAN001** | Mantenimiento de Tarifas (GRUPO 01) |  |
| **REQMAN002** | Mantenimiento de Cursos (GRUPO 02) |  |
| **REQMAN003** | Mantenimiento de Profesores (GRUPO 03) |  |
| **REQMAN004** | Mantenimiento de Alumnos (GRUPO 04) |  |
| **REQMAN005** | Mantenimiento de Empleados (GRUPO 05 **NOSOTROS)** |  |

## Consultas (CON)

| **CODIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| --- | --- | --- |
| **REQCON001** | Consulta de alumnos (GRUPO 04) | Consultar los alumnos de un curso programado |
| **REQCON002** | Consulta financiera de un ciclo (GRUPO 05 **NOSOTROS**) | Consulta los ingresos y gastos de un ciclo y la ganancia por cada alumno |
|  |  |  |
|  |  |  |
|  |  |  |

## Reportes (REP)

| **CODIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Software

Base de datos: SQL Server

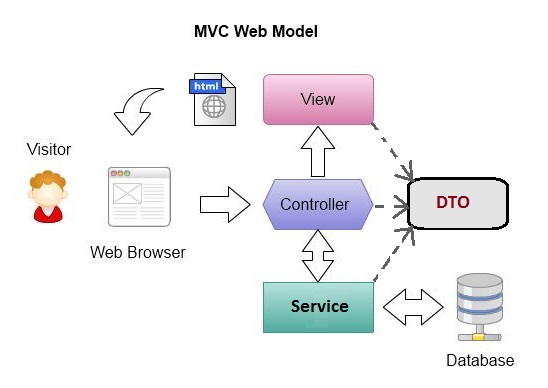
Lenguaje de programación: Java y JavaScript

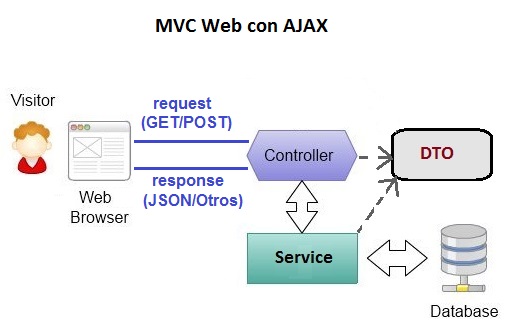
Servidor Web: Apache / GlassFish

IDE: NetBeans / Eclipse

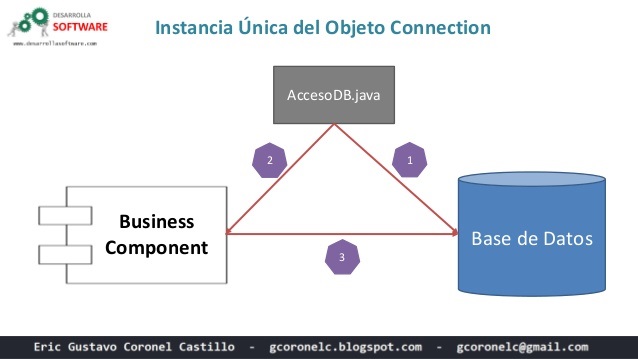
# Arquitectura de la Solución

Arquitectura en capas utilizando MVC-WEB.





# Acceso a la Base de Datos



A continuación, se tiene la clase AccesoDB:

package pe.egcc.edutec.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

/\*\*

\*

\* @author Eric Gustavo Coronel Castillo

\* @blog gcoronelc.blogspot.com

\*/

public class AccesoDB {

private AccesoDB() {

}

public static Connection getConnection() throws SQLException {

Connection cn = null;

try {

// Datos SQL Server

String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";

String url = "jdbc:sqlserver://localhost:1433;databaseName=EDUTEC";

String user = "super";

String pass = "admin";

// Cargar el driver a memoria

Class.forName(driver).newInstance();

// Obtener el objeto Connection

cn = DriverManager.getConnection(url, user, pass);

} catch (SQLException e) {

throw e;

} catch(ClassNotFoundException e){

throw new SQLException("ERROR, no se encuentra el driver.");

} catch(Exception e){

throw new SQLException("ERROR, no se tiene acceso al servidor.");

}

return cn;

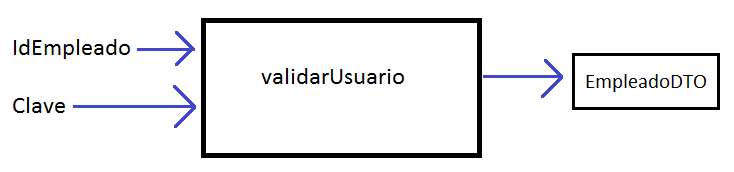
}

}

# Requerimientos de Seguridad

## REQSEG001: Ingreso al Sistema

### Diseño del Servicio



Si los datos son correctos, retorna un objeto de tipo **EmpleadoDTO**, en caso contrario genera una excepción.

### Modelamiento del Proceso

### Programación del Servicio

package pe.egcc.edutec.service;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import pe.egcc.edutec.db.AccesoDB;

import pe.egcc.edutec.dto.EmpleadoDTO;

public class LogonService {

public EmpleadoDTO validarUsuario(String codigo, String clave){

EmpleadoDTO empleadoDTO = null;

Connection cn = null;

try {

cn = AccesoDB.getConnection();

String sql = "select idempleado,apeempleado,nomempleado,"

+ "cargo,dirempleado,telempleado,emailempleado "

+ "from empleado "

+ "where idempleado = ? and clave = ?";

PreparedStatement pstm = cn.prepareStatement(sql);

pstm.setString(1, codigo);

pstm.setString(2, clave);

ResultSet rs = pstm.executeQuery();

if(!rs.next()){

throw new SQLException("Datos incorrectos");

}

empleadoDTO = new EmpleadoDTO();

empleadoDTO.setCodigo(rs.getString("idempleado"));

empleadoDTO.setApellido(rs.getString("apeempleado"));

empleadoDTO.setNombre(rs.getString("nomempleado"));

empleadoDTO.setCargo(rs.getString("cargo"));

empleadoDTO.setDireccion(rs.getString("dirempleado"));

empleadoDTO.setTelefono(rs.getString("telempleado"));

empleadoDTO.setEmail(rs.getString("emailempleado"));

empleadoDTO.setClave("");

rs.close();

pstm.close();

} catch (SQLException e) {

throw new RuntimeException(e.getMessage());

} catch (Exception e) {

throw new RuntimeException("Error en el proceso");

} finally {

try {

cn.close();

} catch (Exception e2) {

}

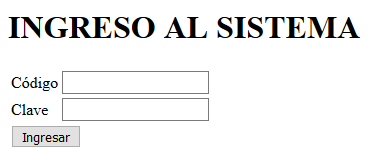
}

return empleadoDTO;

}

}

### Interfaz de Usuario



## REQSEG002: Cambio de Clave

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario

# Requerimientos de Negocio

## REQNEG001: Generación de un nuevo ciclo

### Diseño del Servicio

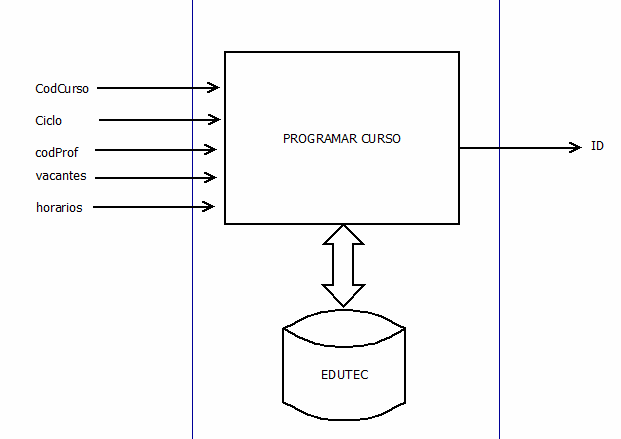
### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario

## REQNEG002: Programar Curso

### Diseño del Servicio



### Modelo del Proceso

Aquí debes desarrollar un diagrama de flujo del proceso.

### Programación del servicio

public int programarCurso(String ciclo, String curso,

String profesor, String horario, int vacantes){

int id = 0;

Connection cn = null;

try {

// Iniciando la Tx

cn = AccesoDB.getConnection();

cn.setAutoCommit(false);

// Validar el ciclo

int anio = Integer.parseInt(ciclo.substring(0,4));

int mes = Integer.parseInt(ciclo.substring(5,7));

Calendar fechaActual = Calendar.getInstance();

int anioSistema = fechaActual.get(Calendar.YEAR);

int mesSistema = fechaActual.get(Calendar.MONTH) + 1;

if(anio < anioSistema){

throw new SQLException("Ciclo incorrecto.");

}

if(anio == anioSistema && mes < mesSistema){

throw new SQLException("Ciclo incorrecto.");

}

// Leer precio

String sql = "select t.PrecioVenta precio "

+ "from Tarifa t join Curso c "

+ "on t.IdTarifa = c.IdTarifa "

+ "where c.IdCurso = ?";

PreparedStatement pstm = cn.prepareStatement(sql);

pstm.setString(1, curso);

ResultSet rs = pstm.executeQuery();

if(!rs.next()){

throw new SQLException("Codigo de curso no existe.");

}

double precio = rs.getDouble("precio");

rs.close();

pstm.close();

// Profesor

if(profesor != null && profesor.length() == 0){

profesor = null;

}

// Insertar registro

sql = "insert into CursoProgramado(IdCiclo,IdCurso,"

+ "IdProfesor,Vacantes,Matriculados,"

+ "PreCursoProg,Horario,Activo)"

+ "values(?,?,?,?,0,?,?,1)";

pstm = cn.prepareStatement(sql);

pstm.setString(1, ciclo);

pstm.setString(2, curso);

pstm.setString(3, profesor);

pstm.setInt(4, vacantes);

pstm.setDouble(5, precio);

pstm.setString(6, horario);

pstm.executeUpdate();

pstm.close();

// Leer el id

sql = "select IDENT\_CURRENT( 'CursoProgramado' ) id";

pstm = cn.prepareStatement(sql);

rs = pstm.executeQuery();

rs.next();

id = rs.getInt("id");

rs.close();

pstm.close();

// Confirmar Tx

cn.commit();

} catch (SQLException e) {

try {

cn.rollback();

} catch (Exception e2) {

}

throw new RuntimeException(e.getMessage());

} catch (Exception e) {

try {

cn.rollback();

} catch (Exception e2) {

}

throw new RuntimeException("Se ha producido un error.");

} finally{

try {

cn.close();

} catch (Exception e2) {

}

}

return id;

}

### Interfaz de Usuario

## REQNEG003: Anular curso programado

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario

## REQNEG004: Asignar profesor

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario

## REQNEG005: Matricula

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

PROGRAMACION DEL SERVICIO

**public** **int** matricula(String alumno, String ciclo, String curso){

Connection cn = **null**;

**try** {

// Iniciando la Tx

cn = AccesoDB.getConnection();

cn.setAutoCommit(**false**);

// Insertar registro

sql = "insert into Alumno(ApeAlumno, NomAlumno,"

+ "DirAlumno, TelAlumno, EmailAlumno,"

+ "values(?,?,?,?,?)";

pstm = cn.prepareStatement(sql);

pstm.setString(1, apellidos);

pstm.setString(2, nombres);

pstm.setString(3, direccion);

pstm.setString(4, telefono);

pstm.setString(5, email);

pstm.close();

/ Leer el id

sql = "select IDENT\_CURRENT( 'Matricula' ) id";

pstm = cn.prepareStatement(sql);

rs = pstm.executeQuery();

rs.next();

id = rs.getInt("id");

rs.close();

pstm.close();

// Confirmar Tx

cn.commit();

}**catch** (Exception e2) {

}

**throw** **new** RuntimeException(e.getMessage());

} **catch** (Exception e) {

**try** {

cn.rollback();

} **catch** (Exception e2) {

}

**throw** **new** RuntimeException("Se ha producido un error.");

} **finally**{

**try** {

cn.close();

} **catch** (Exception e2) {

}

}

**return** id;

}

}

### Interfaz de Usuario

## REQNEG006: Registrar notas de curso

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

**private** **void** registrarNotas(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

String pagina = "registrarNotas.jsp";

String curso = "";

String alumno = "";

**int** idalumno = 0;

**try** {

// Datos

// curso = request.getParameter("curso");

alumno = request.getParameter("alumno");

} **catch** (Exception e) {

request.setAttribute("error", e.getMessage());

}

request.setAttribute("alumno", alumno);

// request.setAttribute("curso", curso);

programarCursoView(request, response);

}

### Interfaz de Usuario



## REQNEG007: Registrar notas de estudiante

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario



## REQNEG008: Registrar nota de examen de subsanacion

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario

## REQNEG009: Eliminar matricula

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario



## REQNEG010: Cambio de curso

### Diseño del Servicio

### Modelamiento del Proceso

### Programación del Servicio

### Interfaz de Usuario

**Requerimientos de Mantenimiento**

**REQMAN001: Mantenimiento de Tarifas**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**Interfaz de Usuario**

**REQMAN002: Mantenimiento de Cursos**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**Interfaz de Usuario**

**REQMAN003: Mantenimiento de Profesores**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**Interfaz de Usuario**

**REQMAN004: Mantenimiento de Alumnos**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**Interfaz de Usuario**

**REQMAN005: Mantenimiento de Empleados**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**public** **class** ConsultaService {

**public** String eliminarEmpleado( String codigo ){

String mensaje = "";

Connection cn = **null**;

**try** {

cn = AccesoDB.*getConnection*();

// Consulta

// Insertar registro

String sql = " DELETE FROM [dbo].[Empleado] "

+ " WHERE IdEmpleado = ?";

PreparedStatement pstm = cn.prepareStatement(sql);

pstm.setString(1, codigo);

pstm.executeUpdate();

pstm.close();

mensaje = "Se Elimino satisfactoriamente";

cn.commit();

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("Error al acceder vista de cursos programados.");

} **finally** {

**try** {

cn.close();

} **catch** (Exception e2) {

mensaje = "0";

}

}

**return** mensaje;

}

**public** String actualizarEmpleado( String codigo, String clave, String nombre, String apellido, String cargo,String direccion,String telefono,String email ){

String mensaje = "";

Connection cn = **null**;

**try** {

cn = AccesoDB.*getConnection*();

// Consulta

// Insertar registro

String sql = " UPDATE [dbo].[Empleado] "

+ " SET Clave = ? , NomEmpleado = ? , ApeEmpleado = ? , "

+ " Cargo = ? , DirEmpleado = ? , TelEmpleado = ? , EmailEmpleado = ? "

+ " WHERE IdEmpleado = ? ";

PreparedStatement pstm = cn.prepareStatement(sql);

pstm.setString(1, clave);

pstm.setString(2, nombre );

pstm.setString(3, apellido);

pstm.setString(4, cargo);

pstm.setString(5, direccion);

pstm.setString(6, telefono);

pstm.setString(7, email);

pstm.setString(8, codigo);

pstm.executeUpdate();

pstm.close();

mensaje = "Se Actualizaron satisfactoriamente";

cn.commit();

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("Error al acceder vista de cursos programados.");

} **finally** {

**try** {

cn.close();

} **catch** (Exception e2) {

mensaje = "0";

}

}

**return** mensaje;

}

**public** List<Map<String,Object>> consultaEmpleados(String codigo){

List<Map<String,Object>> lista = **null**;

Connection cn = **null**;

**try** {

cn = AccesoDB.*getConnection*();

// Consulta

String sql = " select e.IdEmpleado as idempleado, e.Clave as clave, e.NomEmpleado as nombre, e.ApeEmpleado as apellido, e.Cargo as cargo ,e.DirEmpleado as direccion , e.TelEmpleado as telefono ,e.EmailEmpleado as email "

+ " from [dbo].[Empleado] e " ;

**if**(!codigo.equals("")){

sql = sql.concat(" WHERE e.IdEmpleado = ? ") ;

}

;

PreparedStatement pstm = cn.prepareStatement(sql);

**if**(!codigo.equals("")){

pstm.setString(1, codigo);

}

ResultSet rs = pstm.executeQuery();

lista = JdbcUtil.*rsToList*(rs);

rs.close();

pstm.close();

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("Error al acceder vista de cursos programados.");

} **finally** {

**try** {

cn.close();

} **catch** (Exception e2) {

}

}

**return** lista;

}

**public** List<Map<String,Object>> consultaCurso(String cursoprog){

List<Map<String,Object>> lista = **null**;

Connection cn = **null**;

**try** {

cn = AccesoDB.*getConnection*();

// Consulta

String sql = " select C.IdCurso as code, c.NomCurso as name "

+ " from [dbo].[Alumno] a "

+ " inner join [dbo].[Matricula] m "

+ " on a.IdAlumno = m.IdAlumno "

+ " inner join [dbo].[CursoProgramado] cu "

+ " on cu.IdCursoProg = m.IdCursoProg "

+ " inner join [dbo].[Curso] c "

+ " on c.IdCurso = cu.IdCurso "

+ " where a.IdAlumno = ? "

;

PreparedStatement pstm = cn.prepareStatement(sql);

pstm.setString(1, cursoprog);

ResultSet rs = pstm.executeQuery();

lista = JdbcUtil.*rsToList*(rs);

rs.close();

pstm.close();

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("Error al acceder vista de cursos programados.");

} **finally** {

**try** {

cn.close();

} **catch** (Exception e2) {

}

}

**return** lista;

}

**Interfaz de Usuario**



**Requerimientos de Consulta**

**REQCON001: Consulta de alumnos**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**Interfaz de Usuario**

**REQCON002: Consulta financiera de un ciclo**

**Diseño del Servicio**

**Modelamiento del Proceso**

**Programación del Servicio**

**public** List<Map<String,Object>> consultaFinanciero(String nombre){

List<Map<String,Object>> lista = **null**;

Connection cn = **null**;

**try** {

cn = AccesoDB.*getConnection*();

// Consulta

String sql = " SELECT distinct (a.NomAlumno +a.ApeAlumno) as nombre , c.nomcurso as nombrecurso , "

+ " p.PreCursoProg as ingreso , (p.PreCursoProg-t.PagoHora) gastos , (p.PreCursoProg+t.PagoHora) ganancias "

+ " FROM [dbo].[Tarifa] t inner join [dbo].[Curso] c on t.IdTarifa=c.IdTarifa "

+ " inner join [dbo].[CursoProgramado] p on p.IdCurso=c.IdCurso "

+ " inner join [dbo].[Matricula] m on p.IdCursoProg = m.IdCursoProg "

+ " inner join [dbo].[Alumno] a on m.IdAlumno = a.IdAlumno "

+ " where (a.NomAlumno +a.ApeAlumno) like ?";

PreparedStatement pstm = cn.prepareStatement(sql);

pstm.setString(1, "%" + nombre + "%" );

ResultSet rs = pstm.executeQuery();

lista = JdbcUtil.*rsToList*(rs);

rs.close();

pstm.close();

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("Error al acceder vista de cursos programados.");

} **finally** {

**try** {

cn.close();

} **catch** (Exception e2) {

}

}

**return** lista;

}

**Interfaz de Usuario**

