Proyecto EDUTEC

**INDICE**

Datos Generales 3

Objetivo 3

Alcance 3

Modelo de Datos 4

Requerimientos 4

Seguridad (SEG) 4

Procesos de Negocio (NEG) 4

Mantenimiento de Tablas (MAN) 5

Consultas (CON) 6

Reportes (REP) 6

Software 6

Arquitectura de la Solución 6

Acceso a la Base de Datos 8

Requerimientos de Seguridad 10

REQSEG001: Ingreso al Sistema 10

Diseño del Servicio 10

Modelamiento del Proceso 10

Programación del Servicio 10

Interfaz de Usuario 11

REQSEG002: Cambio de Clave 11

Requerimientos de Negocio 12

REQNEG001: Generación de un nuevo ciclo 12

Diseño del Servicio 12

REQNEG002: Programar Curso 12

Diseño del Servicio 12

Modelo del Proceso 12

Programación del servicio 12

REQNEG009: Eliminar Matrícula 15

Diseño del Servicio 15

Modelo del Proceso 15

Programación del servicio 15

REQNEG010: Cambio de Curso 16

Diseño del Servicio 16

Modelo del Proceso 16

Programación del servicio 16

# Datos Generales

Curso: Taller de Programación II

Profesor: Gustavo Coronel

Ciclo: VI

Periodo: 2018-0

Integrantes:

* Dick Mendoza
* Pablo
* Cesar Jason Azañero Fernádez

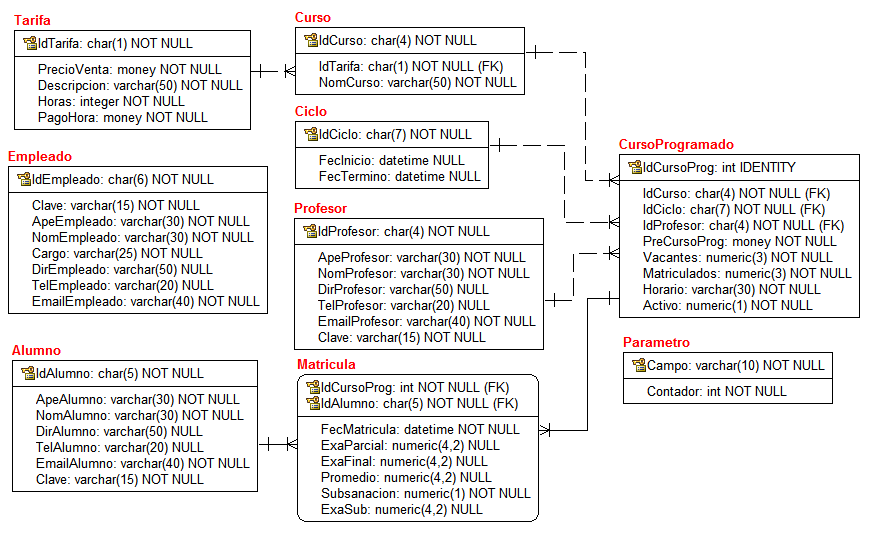
# Objetivo

# Alcance

Con la finalidad de clarificar su presentación, los hemos subdividido en alcances del proyecto y alcances específicos de la solución para Edutec.

El plan de desarrollo del Software describe de manera general el plan que será utilizado para el desarrollo del Sistema de Edutec. Durante el proceso de desarrollo en lo concerniente a la Visión se definen las características del producto a desarrollar desde la perspectiva del cliente.

# 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 | Permite a los empleados cambiar su clave. |

## Procesos de Negocio (NEG)

| **CODIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| --- | --- | --- |
| **REQNEG001** | Generación de un nuevo ciclo | 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 | 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 | 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 | Este proceso permite registrar la matrícula de un estudiante. |
| **REQNEG006** | Registrar notas de curso. | Este proceso permite registrar las notas de los estudiantes de un curso programado. |
| **REQNEG007** | Registrar nota de estudiante. | Este proceso permite registrar las notas de un estudiante. |
| **REQNEG008** | Registrar nota de examen de subsanación. | Este proceso permite registrar la nota de examen de subsanación de un estudiante. |
| **REQNEG009** | Eliminar matricula. | 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. | 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 |  |
| **REQMAN002** | Mantenimiento de Cursos |  |
| **REQMAN003** | Mantenimiento de Profesores |  |
| **REQMAN004** | Mantenimiento de Alumnos |  |
| **REQMAN005** | Mantenimiento de Empleados |  |

## Consultas (CON)

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

## Reportes (REP)

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

# Software

Base de datos: SQL Server

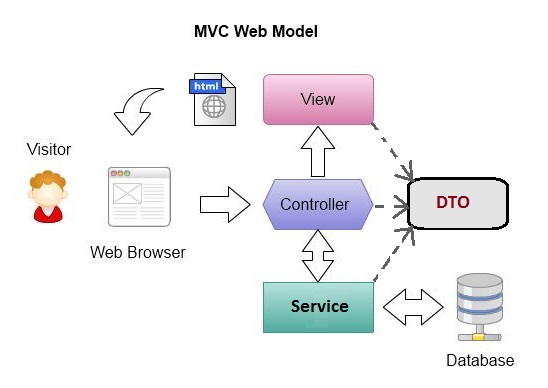
Lenguaje de programación: Java y JavaScript

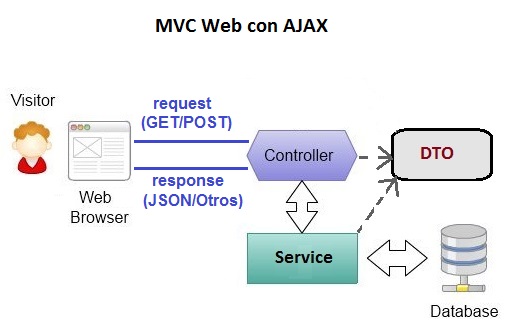
Servidor Web: Apache / GlassFish

IDE: NetBeans / Eclipse

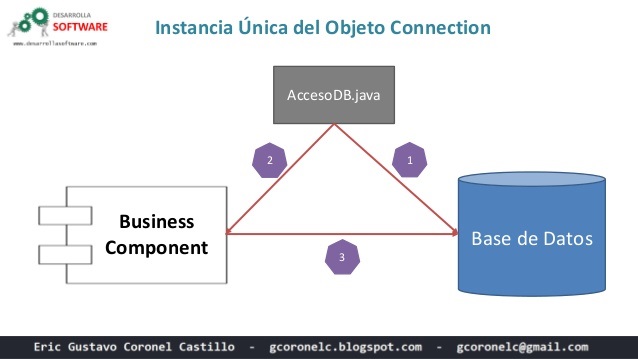
# Arquitectura de la Solución

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;

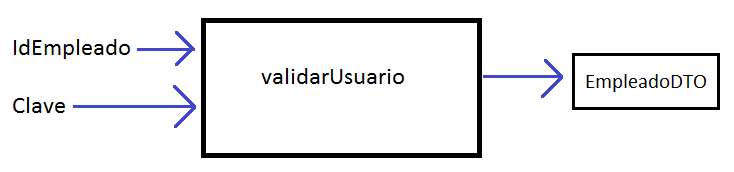
}

}

# 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) {

}

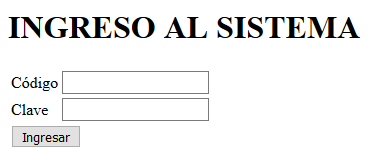
}

return empleadoDTO;

}

}

### Interfaz de Usuario



## REQSEG002: Cambio de Clave

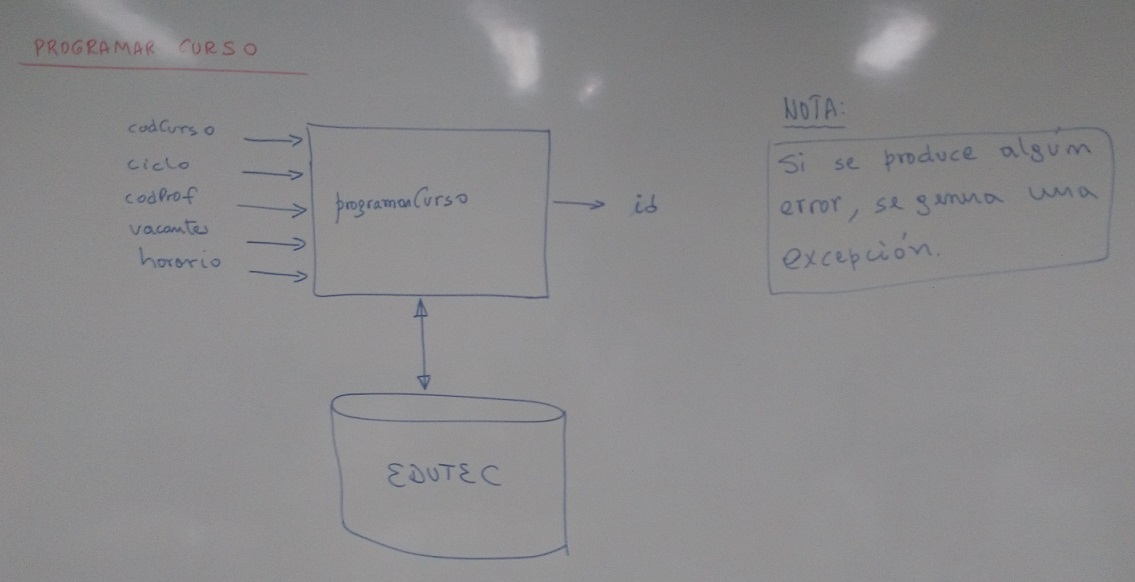
# Requerimientos de Negocio

## REQNEG001: Generación de un nuevo ciclo

### Diseño del Servicio

## 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) {

}

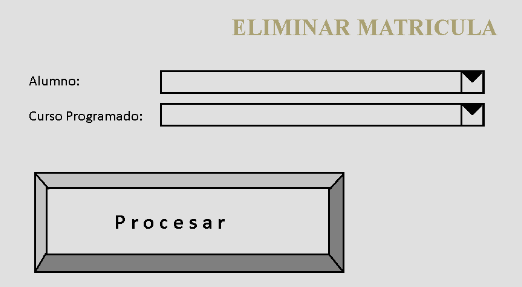
}

return id;

}

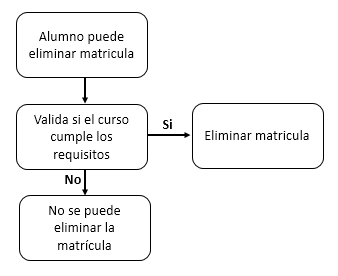
## REQNEG009: Eliminar Matrícula

### Diseño del Servicio



### Modelo del Proceso

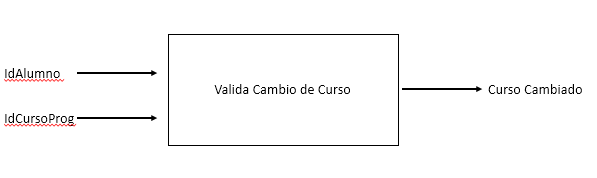
Aquí debes desarrollar un diagrama de flujo del proceso.



### Programación del servicio

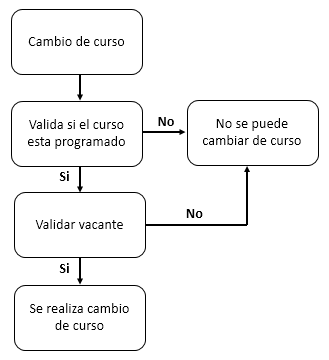
## REQNEG010: Cambio de Curso

### Diseño del Servicio



### Modelo del Proceso

Aquí debes desarrollar un diagrama de flujo del proceso.



### Programación del servicio