



**INSTITUTO SUPERIOR TECNOLÓGICO PRIVADO CIBERTEC**

**CARRERAS PROFESIONALES**

**EXPERIENCIAS FORMATIVAS EN SITUACIONES REALES DE TRABAJO II**

## **SISTEMA DE REGISTRO Y MATRÍCULA DE ALUMNOS: VIRTUAL DATA**

**TERCER CICLO**

**SECCIÓN EA4906**

**Integrantes:**

**Castillo Villacorta, Luis Eduardo**

**Tineo Kam, Jeffrey**

**CARRERAS PROFESIONALES**

**CIBERTEC**

**LIMA, 2022**

# ÍNDICE

INTRODUCCIÓN .....	4
JUSTIFICACIÓN.....	5
OBJETIVOS .....	6
DIAGRAMA BD.....	21
SCRIPT BD .....	8
CONCLUSIONES .....	60
RECOMENDACIONES .....	60
BIBLIOGRAFÍA .....	61

# ÍNDICE DE FIGURAS

Figura N° 1 Login .....	21
Figura N° 2: Menú Principal.....	26
Figura N° 3: Módulo Consulta .....	27
Figura N° 4: Módulo Mantenimiento .....	27
Figura N° 5: Módulo Registro .....	27
Figura N° 6: Módulo Reporte .....	28
Figura N° 12: Acerca de tienda .....	<b>¡Error! Marcador no definido.</b>

# INTRODUCCIÓN

La informática, por su rapidez de crecimiento y expansión, ha venido transformando dinámicamente a las sociedades actuales; sin embargo, el público en general solo las conoce superficialmente y no reconoce la cantidad de posibles soluciones que puede llegar a ofrecer si se identifican las ganancias significativas que brinda a una empresa o a la sociedad en general.

Con base en lo anteriormente expuesto, el presente proyecto, **“Sistema de Registro y Matrícula de Alumnos: Virtual Data”**, ha sido desarrollado para dar una solución tecnológica eficiente a una entidad educativa, el cual nos permitirá gestionar diversos servicios y, sobretodo, realizarlos desde el momento en que un alumno ingresa a la base de datos.

El programa ha sido elaborado implementando una solución algorítmica expresada en lenguaje Java a través del software Eclipse. Lo cual nos ha permitido crear una interfaz con cinco menús: Consulta, Mantenimiento, Registro, Reporte y Acerca de, cada uno con opciones propias. El menú **Consulta** permite visualizar la información referente a los alumnos, cursos, docentes, matrículas y retiros respectivamente; el menú **Mantenimiento** permite realizar el mantenimiento de la información correspondiente a alumnos, cursos y docentes; el menú **Registro** permite realizar matrículas y retiros; el menú **Reporte** brinda diversos reportes requeridos por la administración; y, finalmente, el menú **Acerca de** proporciona información del programa.

Como limitaciones tenemos que, debido a la naturaleza de este proyecto, las interfaces gráficas solamente son aplicables en una computadora de escritorio, utilizando Sistema operativo Windows 10 (de preferencia) y no está diseñado para aplicaciones web o en línea.

Está diseñado para brindar a cualquier entidad educativa, independientemente de su tamaño, un software que le permita organizar, ejecutar y monitorear la base de datos de sus alumnos, cursos, docentes, matrículas y retiros.

Versión de Java : Java SE Development Kit 8 Update 291 (64-bit)  
Versión de Eclipse : 2021-03R

## JUSTIFICACIÓN

El desarrollo del software “VIRTUAL DATA” generará un gran avance para la eficiencia en los procesos de registro y matrícula de alumnos.

Desde el punto de vista económico, la implementación del programa disminuirá el uso de recursos como tiempo, personas, materiales de escritorio; y, por supuesto, el costo del procesamiento. Esto se debe a que el software desarrollado logrará ordenar y generar información exacta en los procesos antes descritos.

Desde el punto de vista tecnológico, el proyecto automatizará procesos realizados por la organización.

Desde el punto de vista sistémico, el proyecto no sólo beneficiará al área de atención al alumno de las empresas educativas, sino que también tendrá un importante impacto en el área administrativa.

Desde el punto de vista metodológico, el desarrollo del proyecto plasma conceptos y técnicas para formular e implementar algoritmos apropiados que facilitarán una mejor gestión de la data de los procesos de registro y matrícula.

Desde el punto de vista práctico, el proyecto se ha realizado para diseñar un programa que garantizará una interacción sencilla y eficiente con el usuario.

Con relación a los **beneficiarios del proyecto** podemos citar a las áreas de atención al alumno y administrativa de las instituciones educativas como **beneficiarios directos**, por hacer uso de las bondades del programa. Del mismo modo, son considerados como **beneficiarios indirectos**, los alumnos y docentes que recibirán una atención precisa y más rápida.

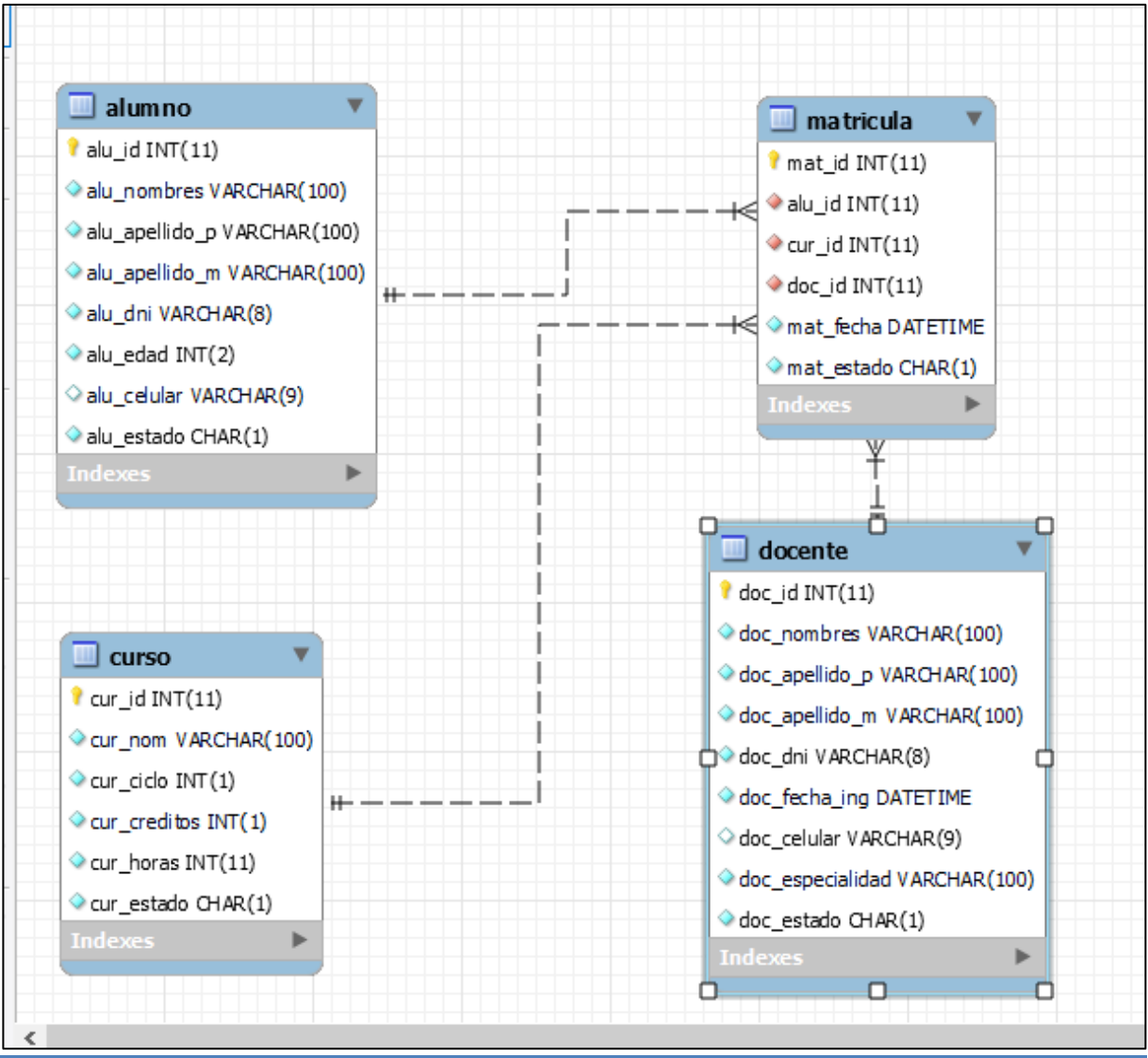
## OBJETIVOS

<b>OBJETIVO SMART:</b> Identificar las necesidades de los clientes educativos potenciales a usar el programa.
<b>Específico:</b> Investigar sobre los requerimientos de automatización de gestión de servicios (registro y matrícula) para los potenciales clientes.
<b>Medible:</b> Para reducir el procesamiento de información y automatizar proceso de gestión de servicios (registro y matrícula) en un 20%.
<b>Alcanzable:</b> Revisión de información bibliográfica disponible físicamente y por internet (libros, artículos y tesis).
<b>Relevante:</b> Para generar primer prototipo y primer entregable en formato Word.
<b>Tiempo:</b> 10 días hábiles.

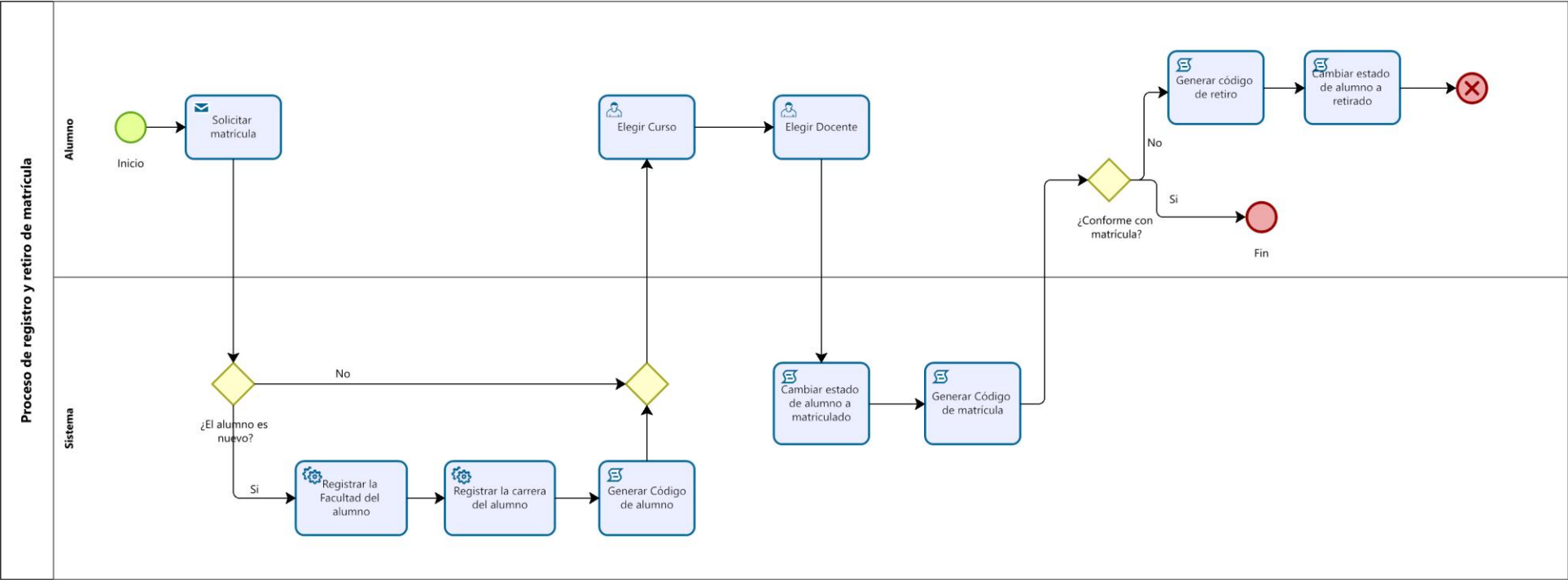
<b>OBJETIVO SMART:</b> Estructurar el programa y diseñar prototipo.
<b>Específico:</b> Generar un documento Word con esquemas de las interfaces gráficas del programa VIRTUAL DATA.
<b>Medible:</b> A través de reuniones virtuales realizadas una vez por semana con los integrantes del proyecto.
<b>Alcanzable:</b> Con uso de app.diagrams.net e interfaz gráfica de Eclipse.
<b>Relevante:</b> Para escribir primera versión del código fuente.
<b>Tiempo:</b> 15 días hábiles.

<b>OBJETIVO SMART:</b> Escribir código fuente y lanzar el programa.
<b>Específico:</b> Diseñar la solución algorítmica del programa VIRTUAL DATA.
<b>Medible:</b> Repartir una parte del proyecto a desarrollar por cada uno de los integrantes con entrega de avances una vez a la semana.
<b>Alcanzable:</b> Con el uso de Eclipse y SharePoint (donde guardamos las versiones y las copias del programa)
<b>Relevante:</b> Para ejecución del proyecto en una computadora personal o laptop.
<b>Tiempo:</b> 20 días hábiles.

# DIAGRAMA DE LA BASE DE DATOS



# DIAGRAMA DE FLUJO BPMN





# SCRIPT DE LA BASE DE DATOS

```
-- borra la bd si existe
```

```
DROP DATABASE IF EXISTS INSTITUTO;
```

```
-- creamos la bd
```

```
CREATE DATABASE INSTITUTO;
```

```
-- activamos la bd
```

```
USE INSTITUTO;
```

```
-- =====  
-- TABLA CURSO  
-- =====
```

```
CREATE TABLE CURSO(  
    cur_id int auto_increment primary key,  
    cur_nom varchar(100) not null,  
    cur_ciclo int(1) not null,  
    cur_creditos int(1) not null,  
    cur_horas int not null,  
    cur_estado char(1) not null default 1,  
    CONSTRAINT CK_CicloCur CHECK (cur_ciclo >= 1 and (cur_ciclo <= 6)),  
    CONSTRAINT CK_CreditosCur CHECK (cur_creditos > 0)  
);  
ALTER TABLE CURSO AUTO_INCREMENT=101;
```

```
-- INSERTS - CURSO
```

```
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('FUNDAMENTOS DE GESTIÓN EMPRESARIAL', 1, 4, 30);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('DESARROLLO DE HABILIDADES PROFESIONALES I', 1, 2, 10);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('FOTOGRAFÍA DIGITAL', 2, 3, 25);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('PRODUCCIÓN AUDIOVISUAL', 3, 4, 30);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('LENGUAJE DE PROGRAMACIÓN I', 3, 3, 52);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('MATEMÁTICA I', 1, 2, 25);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('DIBUJO DE ARQUITECTURA 2D', 2, 2, 40);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('NORMATIVIDAD EN LA CONTRUCCIÓN', 4, 2, 26);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('MARKETING DIGITAL', 3, 4, 50);  
INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas) VALUES ('PLANNING Y SOCIAL MEDIA', 6, 5, 48);
```

```
-- =====  
-- TABLA ALUMNO  
-- =====
```

```
CREATE TABLE ALUMNO(  
    alu_id int auto_increment not null primary key,  
    alu_nombres varchar(100) not null,  
    alu_apellido_p varchar(100) not null,  
    alu_apellido_m varchar(100) not null,  
    alu_dni varchar(8) not null,  
    alu_edad int(2) not null,  
    alu_celular varchar(9) null,  
    alu_estado char(1) not null default 1  
);  
ALTER TABLE ALUMNO AUTO_INCREMENT=202210001;
```

```
-- INSERTS - ALUMNO
```

```
INSERT INTO ALUMNO (alu_nombres, alu_apellido_p, alu_apellido_m, alu_dni, alu_edad, alu_celular) VALUES ('JOSÉ', 'ROBERTO', 'MENDEIETA', '25554188', 18, '965148222');  
INSERT INTO ALUMNO (alu_nombres, alu_apellido_p, alu_apellido_m, alu_dni, alu_edad, alu_celular)
```

```
VALUES ('MARÍA', 'SANCHEZ', 'RAMOS', '16584050', 21, '993551404');
INSERT INTO ALUMNO (alu_nombres, alu_apellido_p, alu_apellido_m, alu_dni, alu_edad, alu_celular)
VALUES ('CESAR', 'MUÑOZ', 'VIVANCO', '40001524', 26, '931022845');
INSERT INTO ALUMNO (alu_nombres, alu_apellido_p, alu_apellido_m, alu_dni, alu_edad, alu_celular)
VALUES ('ALEXANDRA', 'RAMOS', 'HUAMAN', '10230088', 30, '966585147');
INSERT INTO ALUMNO (alu_nombres, alu_apellido_p, alu_apellido_m, alu_dni, alu_edad, alu_celular)
VALUES ('NELLY', 'SALVATIERRA', 'CABRERA', '24775844', 19, '964074447');
```

```
-- =====
-- TABLA DOCENTE
-- =====
```

```
CREATE TABLE DOCENTE(
    doc_id int auto_increment not null primary key,
    doc_nombres varchar(100) not null,
    doc_apellido_p varchar(100) not null,
    doc_apellido_m varchar(100) not null,
    doc_dni varchar(8) not null,
    doc_fecha_ing datetime not null,
    doc_celular varchar(9) null,
    doc_especialidad varchar(100) not null,
    doc_estado char(1) not null default 1
);
```

```
ALTER TABLE DOCENTE AUTO_INCREMENT=2001;
```

```
-- INSERTS - DOCENTE
```

```
INSERT INTO DOCENTE (doc_nombres, doc_apellido_p, doc_apellido_m, doc_dni, doc_fecha_ing, doc_celular, doc_es-
pecialidad)
VALUES ('JUAN CARLOS', 'SAAVEDRA', 'CERVANTES', '24457100', '2010-12-10', '963852741', 'NEGOCIOS');
INSERT INTO DOCENTE (doc_nombres, doc_apellido_p, doc_apellido_m, doc_dni, doc_fecha_ing, doc_celular, doc_es-
pecialidad)
VALUES ('LUIS ERNESTO', 'MARTICORENA', 'SANDOVAL', '11209844', '2008-08-02', '951862453', 'DISEÑO');
INSERT INTO DOCENTE (doc_nombres, doc_apellido_p, doc_apellido_m, doc_dni, doc_fecha_ing, doc_celular, doc_es-
pecialidad)
VALUES ('SUSANA', 'ALBARADO', 'TORRES', '2552088', '2016-10-05', '964125764', 'TI');
INSERT INTO DOCENTE (doc_nombres, doc_apellido_p, doc_apellido_m, doc_dni, doc_fecha_ing, doc_celular, doc_es-
pecialidad)
VALUES ('ANA LIZ', 'PEREZ', 'PRADO', '10008452', '2004-02-18', '984002999', 'COMUNICACIONES');
INSERT INTO DOCENTE (doc_nombres, doc_apellido_p, doc_apellido_m, doc_dni, doc_fecha_ing, doc_celular, doc_es-
pecialidad)
VALUES ('PEDRO PABLO', 'ORBEGOZO', 'SALINAS', '24457101', '2021-12-01', '987444222', 'LITERATURA');
```

```
-- =====
-- TABLA MATRICULA
-- =====
```

```
CREATE TABLE MATRICULA(
    mat_id int auto_increment primary key,
    alu_id int not null,
    cur_id int not null,
    doc_id int not null,
    -- mat_fecha datetime not null,
    mat_fecha DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    mat_estado char(1) not null default 1,
    FOREIGN KEY (alu_id) REFERENCES ALUMNO(alu_id),
    FOREIGN KEY (cur_id) REFERENCES CURSO(cur_id),
    FOREIGN KEY (doc_id) REFERENCES DOCENTE(doc_id)
);
```

```
SELECT DATE_FORMAT(mat_fecha, '%d/$m/%Y') FROM MATRICULA;
ALTER TABLE MATRICULA AUTO_INCREMENT=200001;
```

```
-- INSERTS - MATRICULA
```

```

INSERT INTO MATRICULA (alu_id, cur_id, doc_id) VALUES (202210001, 101, 2001);
UPDATE ALUMNO SET alu_estado=2 WHERE alu_id="202210001";
INSERT INTO MATRICULA (alu_id, cur_id, doc_id) VALUES (202210002, 101, 2001);
UPDATE ALUMNO SET alu_estado=2 WHERE alu_id="202210002";
INSERT INTO MATRICULA (alu_id, cur_id, doc_id) VALUES (202210003, 102, 2002);
UPDATE ALUMNO SET alu_estado=2 WHERE alu_id="202210003";

```

```

-- =====
-- TABLA TIPOS
-- =====

```

```

CREATE TABLE tipo(
tipo_id int not null primary key,
tipo_des varchar(50)
);

```

```

INSERT INTO tipo VALUES(1, "EMPLEADO");
INSERT INTO tipo VALUES(2, "ADMINISTRADOR");

```

```

-- =====
-- TABLA USUARIOS
-- =====

```

```

CREATE TABLE usuario(
usu_id int auto_increment primary key,
usu_usuario varchar(10) unique not null,
usu_contra varchar(10) not null,
usu_nom varchar(20) not null,
usu_tipo int DEFAULT 1 not null,
usu_foto blob DEFAULT null,
foreign key(usu_tipo) references tipo (tipo_id)
);

```

```

-- INSERT CUENTAS
INSERT INTO USUARIO VALUES (1,"admi2022", "Cibertec2", "ADMINISTRADOR", 2, null);
INSERT INTO USUARIO VALUES (2,"usu2022", "Cibertec1", "EMPLEADO", 1, null);

```

---

```

-- =====
-- PROCEDURES CURSOS
-- =====

```

```

/* ----- */
/* MANTENIMIENTO CURSOS*/
/* ----- */

```

```

-- SELECT
DELIMITER //
CREATE PROCEDURE usp_listCurso()
BEGIN
    SELECT * FROM CURSO WHERE cur_estado = 1;
END
DELIMITER ;

```

```

-- INSERT
DELIMITER //
CREATE PROCEDURE usp_registrarCurso(
    p_cur_nom varchar(100),
    p_cur_ciclo int(1),
    p_cur_creditos int(1),

```

```

        p_cur_horas int
    )
BEGIN
    INSERT INTO CURSO (cur_nom, cur_ciclo, cur_creditos, cur_horas)
    VALUES (p_cur_nom, p_cur_ciclo, p_cur_creditos, p_cur_horas);
END
DELIMITER ;

-- UPDATE
DELIMITER //
CREATE PROCEDURE usp_actualizarCurso(
    p_cur_id int,
    p_cur_nom varchar(100),
    p_cur_ciclo int(1),
    p_cur_creditos int(1),
    p_cur_horas int,
    p_cur_estado char(1)
)
BEGIN
    UPDATE CURSO
    SET cur_nom = p_cur_nom, cur_ciclo = p_cur_ciclo, cur_creditos = p_cur_creditos, cur_horas = p_cur_horas,
    cur_estado = p_cur_estado
    WHERE cur_id = p_cur_id;
END
DELIMITER ;

-- DELETE
DELIMITER //
CREATE PROCEDURE usp_eliminarCurso(
    p_cur_id int
)
BEGIN
    UPDATE CURSO
    SET cur_estado = 0
    WHERE cur_id = p_cur_id;
END
DELIMITER ;

/* ----- */
/* CONSULTA CURSOS*/
/* ----- */

DELIMITER //
CREATE PROCEDURE usp_nombreCurso(
    p_cur_id int
)
BEGIN
    SELECT cur_nom FROM CURSO WHERE cur_id = p_cur_id;
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_consultaCurso(
    p_cur_id int
)
BEGIN
    SELECT * FROM CURSO WHERE cur_id = p_cur_id AND cur_estado = 1;
END
DELIMITER ;

DELIMITER //

```

```

CREATE PROCEDURE usp_consultaDocenteCurso(
    p_cur_id int
)
BEGIN
    SELECT doc.doc_id, doc.doc_nombres, doc.doc_apellido_p, doc.doc_apellido_m, doc.doc_dni,
    doc.doc_fecha_ing, doc.doc_celular, doc.doc_especialidad
    FROM matricula as mat
    INNER JOIN docente as doc on mat.doc_id = doc.doc_id
    INNER JOIN curso as cur on cur.cur_id = p_cur_id and cur.cur_estado = 1 and cur.cur_id = mat.alu_id;
END
DELIMITER ;

-- =====
-- PROCEDURES ALUMNOS
-- =====

/* ----- */
/* MANTENIMIENTO ALUMNOS*/
/* ----- */

-- SELECT
DELIMITER //
CREATE PROCEDURE usp_listAlumno()
BEGIN
    SELECT * FROM ALUMNO;
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_listAlumnoActivo()
BEGIN
    SELECT * FROM ALUMNO WHERE alu_estado!=0;
END
DELIMITER ;

-- INSERT
DELIMITER //
CREATE PROCEDURE usp_registrarAlumno(
    p_alu_nombres varchar(100),
    p_alu_apellido_p varchar(100),
    p_alu_apellido_m varchar(100),
    p_alu_dni varchar(8),
    p_alu_edad int(2),
    p_alu_celular varchar(9)
)
BEGIN
    INSERT INTO ALUMNO (alu_nombres, alu_apellido_p, alu_apellido_m, alu_dni, alu_edad, alu_celular)
    VALUES (p_alu_nombres, p_alu_apellido_p, p_alu_apellido_m, p_alu_dni, p_alu_edad, p_alu_celular);
END
DELIMITER ;

-- UPDATE
DELIMITER //
CREATE PROCEDURE usp_actualizarAlumno(
    p_alu_id int,
    p_alu_nombres varchar(100),
    p_alu_apellido_p varchar(100),
    p_alu_apellido_m varchar(100),
    p_alu_dni varchar(8),
    p_alu_edad int(2),

```

```

        p_alu_celular varchar(9),
        p_alu_estado char(1)
    )
BEGIN
    UPDATE ALUMNO
    SET    alu_nombres = p_alu_nombres, alu_apellido_p = p_alu_apellido_p, alu_apellido_m = p_alu_apellido_m,
    alu_dni = p_alu_dni, alu_edad = p_alu_edad, alu_celular = p_alu_celular, alu_estado = p_alu_estado
    WHERE alu_id = p_alu_id;
END
DELIMITER ;

-- DELETE
DELIMITER //
CREATE PROCEDURE usp_eliminarAlumno(
    p_alu_id int
)
BEGIN
    UPDATE ALUMNO
    SET    alu_estado = 0
    WHERE alu_id = p_alu_id;
END
DELIMITER ;

/* ----- */
/* CONSULTA */
/* ----- */

DELIMITER //
CREATE PROCEDURE usp_buscarAlumnoActivo(
    p_alu_id int
)
BEGIN
    SELECT * FROM ALUMNO WHERE alu_estado!=0 and alu_id = p_alu_id;
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_consultaCursoAlumno(
    p_alu_id int
)
BEGIN
    SELECT cur.cur_id, cur.cur_nom, cur.cur_ciclo, cur.cur_creditos, cur.cur_horas
    FROM matricula as mat
    INNER JOIN curso as cur on mat.cur_id = cur.cur_id
    INNER JOIN alumno as alu on alu.alu_id = p_alu_id and alu.alu_estado != 0 and alu.alu_id = mat.alu_id;
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_nombreAlumno(
    p_alu_id int
)
BEGIN
    SELECT alu_nombres, alu_apellido_p, alu_apellido_m FROM ALUMNO WHERE alu_id = p_alu_id;
END
DELIMITER ;

-- =====

```

```
-- PROCEDURES DOCENTES
```

```
-- =====
```

```
/* ----- */  
/* MANTENIMIENTO DOCENTES*/  
/* ----- */
```

```
-- SELECT
```

```
DELIMITER //
```

```
CREATE PROCEDURE usp_listDocente()
```

```
BEGIN
```

```
    SELECT * FROM DOCENTE WHERE doc_estado = 1;
```

```
END
```

```
DELIMITER ;
```

```
-- INSERT
```

```
DELIMITER //
```

```
CREATE PROCEDURE usp_registrarDocente(  
    p_doc_nombres varchar(100),  
    p_doc_apellido_p varchar(100),  
    p_doc_apellido_m varchar(100),  
    p_doc_dni varchar(8),  
    p_doc_fecha_ing datetime,  
    p_doc_celular varchar(9),  
    p_doc_especialidad varchar(100)  
)
```

```
BEGIN  
    INSERT INTO DOCENTE (doc_nombres, doc_apellido_p, doc_apellido_m, doc_dni, doc_fecha_ing, doc_celular,  
doc_especialidad)  
    VALUES (p_doc_nombres, p_doc_apellido_p, p_doc_apellido_m, p_doc_dni, p_doc_fecha_ing, p_doc_celular,  
p_doc_especialidad);  
END
```

```
DELIMITER ;
```

```
-- UPDATE
```

```
DELIMITER //
```

```
CREATE PROCEDURE usp_actualizarDocente(  
    p_doc_id int,  
    p_doc_nombres varchar(100),  
    p_doc_apellido_p varchar(100),  
    p_doc_apellido_m varchar(100),  
    p_doc_dni varchar(8),  
    p_doc_fecha_ing datetime,  
    p_doc_celular varchar(9),  
    p_doc_especialidad varchar(100),  
    p_doc_estado char(1)  
)
```

```
BEGIN  
    UPDATE DOCENTE  
    SET    doc_nombres = p_doc_nombres, doc_apellido_p = p_doc_apellido_p, doc_apellido_m = p_doc_ape-  
llido_m, doc_dni = p_doc_dni, doc_fecha_ing = p_doc_fecha_ing, doc_celular = p_doc_celular, doc_especialidad =  
p_doc_especialidad, doc_estado = p_doc_estado  
    WHERE doc_id = p_doc_id;
```

```
END
```

```
DELIMITER ;
```

```
-- DELETE
```

```
DELIMITER //
```

```
CREATE PROCEDURE usp_eliminarDocente(  
    p_doc_id int
```

```

        p_doc_id int
    )
BEGIN
    UPDATE DOCENTE
    SET    doc_estado = 0
    WHERE doc_id = p_doc_id;
END
DELIMITER ;

/* ----- */
/* CONSULTA */
/* ----- */

DELIMITER //
CREATE PROCEDURE usp_nombreDocente(
    p_doc_id int
)
BEGIN
    SELECT doc_nombres, doc_apellido_p, doc_apellido_m FROM DOCENTE WHERE doc_id = p_doc_id;
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_consultaDocente(
    p_doc_id int
)
BEGIN
    SELECT * FROM DOCENTE WHERE doc_id = p_doc_id AND doc_estado = 1;
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_consultaCursoDocente(
    p_doc_id int
)
BEGIN
    SELECT cur.cur_id, cur.cur_nom, cur.cur_ciclo, cur.cur_creditos, cur.cur_horas
    FROM matricula as mat
    INNER JOIN curso as cur on mat.cur_id = cur.cur_id
    INNER JOIN docente as doc on doc.doc_id = p_doc_id and doc.doc_estado = 1 and doc.doc_id = mat.alu_id;
END
DELIMITER ;

-- =====
-- PROCEDURES MATRICULA
-- =====

/* ----- */
/* MANTENIMIENTO MATRICULA */
/* ----- */
-- SELECT
DELIMITER //
CREATE PROCEDURE usp_listMatricula()
BEGIN
    SELECT * FROM MATRICULA WHERE mat_estado = 1;
END
DELIMITER ;

-- INSERT
DELIMITER //

```



```

CREATE PROCEDURE usp_registrarMatricula(
    p_alu_id int,
    p_cur_id int,
    p_doc_id int
)
BEGIN
    INSERT INTO MATRICULA (alu_id, cur_id, doc_id)
    VALUES (p_alu_id, p_cur_id, p_doc_id);
    UPDATE ALUMNO SET alu_estado=2 WHERE alu_id=p_alu_id;
END
DELIMITER ;

-- UPDATE
DELIMITER //
CREATE PROCEDURE usp_actualizarMatricula(
    p_mat_id int,
    p_alu_id int,
    p_cur_id int,
    p_doc_id int,
    p_mat_fecha datetime
)
BEGIN
    UPDATE MATRICULA
    SET alu_id = p_alu_id, cur_id = p_cur_id, doc_id = p_doc_id, mat_fecha = p_mat_fecha
    WHERE mat_id = p_mat_id;
END
DELIMITER ;

-- DELETE
DELIMITER //
CREATE PROCEDURE usp_eliminarMatricula(
    p_mat_id int
)
BEGIN
    UPDATE MATRICULA
    SET mat_estado = 0
    WHERE mat_id = p_mat_id;
    UPDATE ALUMNO SET alu_estado = 1 WHERE alu_id= (SELECT alu_id FROM MATRICULA WHERE mat_id = p_mat_id);
END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_verificarMatricula(
    p_alu_id int,
    p_cur_id int
)
BEGIN
    SELECT * FROM MATRICULA WHERE alu_id = p_alu_id AND cur_id = p_cur_id;
END
DELIMITER ;

/* ----- */
/* RETIRO */
/* ----- */

-- SELECT
DELIMITER //
CREATE PROCEDURE usp_listRetiro()

```

```

BEGIN
    SELECT * FROM MATRICULA WHERE mat_estado = 0;
END
DELIMITER ;

-- INSERT
DELIMITER //
CREATE PROCEDURE usp_registrarRetiro(
    p_mat_id int
)
BEGIN
    UPDATE MATRICULA
    SET    mat_estado = 0
    WHERE mat_id = p_mat_id;
    UPDATE ALUMNO SET alu_estado = 1 WHERE alu_id= (SELECT alu_id FROM MATRICULA WHERE mat_id =
p_mat_id);
END
DELIMITER ;

-- UPDATE
DELIMITER //
CREATE PROCEDURE usp_actualizarRetiro(
    p_mat_id int,
    p_mat_estado char(1)
)
BEGIN
    UPDATE MATRICULA
    SET    mat_estado = p_mat_estado
    WHERE mat_id = p_mat_id;
    UPDATE ALUMNO SET alu_estado = IF (p_mat_estado = 1,2,1); -- IF (CONDITION,ACTION TRUE, ACTION FALSE)
END
DELIMITER ;

-- DELETE
DELIMITER //
CREATE PROCEDURE usp_eliminarRetiro(
    p_mat_id int
)
BEGIN
    DELETE FROM MATRICULA
    WHERE mat_id = p_mat_id;
    UPDATE ALUMNO SET alu_estado = 1 WHERE alu_id= (SELECT alu_id FROM MATRICULA WHERE mat_id =
p_mat_id);
END
DELIMITER ;

/* ----- */
/*  CONSULTAS  */
/* ----- */

DELIMITER //
CREATE PROCEDURE usp_consultaAlumnoMatricula(
    p_mat_id int,
    p_mat_estado int
)
BEGIN
    SELECT alu.alu_id, alu.alu_nombres, alu.alu_apellido_p, alu.alu_apellido_m, alu.alu_dni, alu.alu_edad,
alu.alu_celular
    FROM matricula as mat
    INNER JOIN alumno as alu on mat.alu_id = alu.alu_id
    WHERE mat.mat_id = p_mat_id and mat.mat_estado = p_mat_estado;

```

```

END
DELIMITER ;

DELIMITER //
CREATE PROCEDURE usp_consultaDocenteMatricula(
    p_mat_id int,
    p_mat_estado int
)
BEGIN
    SELECT doc.doc_id, doc.doc_nombres, doc.doc_apellido_p, doc.doc_apellido_m, doc.doc_dni,
    doc.doc_fecha_ing, doc.doc_celular, doc.doc_especialidad
    FROM matricula as mat
    INNER JOIN docente as doc on mat.doc_id = doc.doc_id
    WHERE mat.mat_id = p_mat_id and mat.mat_estado = p_mat_estado;
END
DELIMITER ;

```

```

DELIMITER //
CREATE PROCEDURE usp_consultaCursoMatricula(
    p_mat_id int,
    p_mat_estado int
)
BEGIN
    SELECT cur.cur_id, cur.cur_nom, cur.cur_ciclo, cur.cur_creditos, cur.cur_horas
    FROM matricula as mat
    INNER JOIN curso as cur on mat.cur_id = cur.cur_id
    WHERE mat.mat_id = p_mat_id and mat.mat_estado = p_mat_estado;
END
DELIMITER ;

```

```

-- =====
-- PROCEDURES USUARIOS
-- =====

```

```

/* ----- */
/* MANTENIMIENTO USUARIOS*/
/* ----- */

```

```

-- SELECT
DELIMITER //
CREATE PROCEDURE usp_listUsuario(
    p_usu_usuario varchar(10)
)
BEGIN
    SELECT * FROM USUARIO WHERE usu_usuario = p_usu_usuario;
END
DELIMITER ;

-- INSERT
DELIMITER //
CREATE PROCEDURE usp_registrarUsuario(
    p_usu_usuario varchar(10),
    p_usu_contra varchar(10),
    p_usu_nom varchar(20),
    p_usu_tipo int,
    p_usu_foto blob
)
BEGIN
    INSERT INTO USUARIO (usu_usuario, usu_contra, usu_nom, usu_tipo, usu_foto)
    VALUES (p_usu_usuario, p_usu_contra, p_usu_nom, p_usu_tipo, p_usu_foto);

```

```

END
DELIMITER ;

SELECT * FROM USUARIO;

-- =====
-- PROCEDURES REPORTES
-- =====

-- REPORTE DE ALUMNOS CON MATRÍCULA PENDIENTE
DELIMITER //
CREATE PROCEDURE usp_reporteAlumnosMatriculaPendiente()
BEGIN
    SELECT * FROM ALUMNO WHERE alu_estado = 1;
END
DELIMITER ;

-- REPORTE DE ALUMNOS CON MATRÍCULA VIGENTE
DELIMITER //
CREATE PROCEDURE usp_reporteAlumnosMatriculaVigente()
BEGIN
    SELECT * FROM ALUMNO WHERE alu_estado = 2;
END
DELIMITER ;

-- REPORTE DE ALUMNOS RETIRADOS
DELIMITER //
CREATE PROCEDURE usp_reporteAlumnosRetirados()
BEGIN
    SELECT al.*
    FROM MATRICULA AS mat
    INNER JOIN ALUMNO AS al ON al.alu_id = mat.alu_id
    WHERE mat_estado = 0;
END
DELIMITER ;

-- REPORTE DE ALUMNOS MATRICULADOS POR CURSO
DELIMITER //
CREATE PROCEDURE usp_reporteMatriculaPorCurso()
BEGIN
    SELECT cur.cur_id as 'ID_CURSO', cur.cur_nom as 'CURSO', count(mat.cur_id) as 'MATRICULADOS'
    FROM matricula as mat
    INNER JOIN curso as cur ON mat.cur_id = cur.cur_id
    GROUP BY mat.cur_id;
END
DELIMITER ;

-- REPORTE DE DOCENTES CON CARGA HORARIA
DELIMITER //
CREATE PROCEDURE usp_reporteDocenteConHorarios()
BEGIN
    SELECT doc.*
    FROM matricula as mat
    INNER JOIN docente as doc ON mat.doc_id = doc.doc_id
    GROUP BY mat.doc_id;
END
DELIMITER ;

```

## DEFINICIÓN

### VIRTUAL DATA 1.0

Al abrir el programa se mostrará el login del programa que nos conducirá al formulario principal y los menús en los paneles establecidos para este proyecto.



Figura N° 1 Login

Para la ventana principal se tiene la siguiente estructura de menús:

#### **Consulta**

- Alumnos
- Docentes
- Cursos
- Matrículas
- Retiros

#### **Mantenimiento**

- Alumno
  - Adicionar
    - ➔ Genere código correlativo
    - ➔ ingrese nombres, apellido paterno, apellido materno, dni, edad y celular
    - ➔ Fije el estado en 0 (*registrado*)
    - ➔ El dni no debe repetirse

- Consultar, Modificar, Eliminar
  - ➔ Visualice todos los alumnos con sus datos completos
  - ➔ Puede modificar los datos en cualquier momento menos código ni dni
  - ➔ Puede eliminar a un alumno sólo cuando su estado es 0 (*registrado*)
  - ➔ La eliminación es física y se efectuará previa confirmación
  - ➔ Grabar los cambios
- Curso
  - Adicionar
    - ➔ Ingrese código de cuatro dígitos evitando repetidos
    - ➔ ingrese asignatura, ciclo, número de créditos y cantidad de horas
    - ➔ Luego de adicionar un curso la lista debe quedar ordenada por código
  - Consultar, Modificar, Eliminar
    - ➔ Visualice todos los cursos con sus datos completos
    - ➔ Puede modificar los datos en cualquier momento menos el código
    - ➔ Un curso puede eliminarse sólo cuando ningún alumno esté matriculado en él
    - ➔ La eliminación es física y se efectuará previa confirmación
    - ➔ Grabar los cambios
- Docente
  - Adicionar
    - ➔ Genere código correlativo
    - ➔ ingrese nombres, apellido paterno, apellido materno, dni, celular, especialidad y fecha de ingreso.
    - ➔ El dni no debe repetirse
  - Consultar, Modificar, Eliminar
    - ➔ Visualice todos los docentes con sus datos completos
    - ➔ Puede modificar los datos en cualquier momento menos código ni dni
    - ➔ Un docente puede eliminarse sólo cuando no tenga cursos asignados en el registro matrícula.
    - ➔ La eliminación es física y se efectuará previa confirmación
    - ➔ Grabar los cambios

## Registro

- Matrícula
  - Adicionar
    - ➔ Genere código correlativo
    - ➔ Tome la fecha y hora del sistema
    - ➔ Matricule a un alumno una sola vez y en un solo curso con un determinado docente
    - ➔ Fije el estado del alumno en 1 (*matriculado*)
  - Consultar, Modificar, Eliminar
    - ➔ Visualice todas las matrículas con sus datos completos

- ➔ Sólo puede cambiar en cualquier momento el curso matriculado
- ➔ Puede cancelar una matrícula sólo cuando el estado del alumno no sea 2 (*retirado*)
- ➔ La cancelación de la matrícula es física y se efectuará previa confirmación
- ➔ Grabar los cambios
- Retiro
  - Adicionar
    - ➔ Genere código correlativo
    - ➔ Tome la fecha y hora del sistema
    - ➔ Desactive temporalmente una matrícula fijando el estado del alumno en 2 (*retirado*)
  - Consultar, Modificar, Eliminar
    - ➔ Visualice todos los retiros con sus datos completos
    - ➔ Sólo puede cambiar en cualquier momento el curso de una matrícula desactivada
    - ➔ Puede cancelar un retiro sólo si el estado del alumno es 2 (*retirado*)
    - ➔ La cancelación del retiro, es físico y se efectuará previa confirmación
    - ➔ Grabar los cambios

## Reporte

- Alumnos con matrícula pendiente. Mostrar los datos completos de aquellos alumnos que solamente están registrados.
- Alumnos con matrícula vigente. Mostrar los datos completos de aquellos alumnos que solamente están matriculados.
- Alumnos retirados. mostrar los datos completos de aquellos alumnos que están retirados.
- Alumnos matriculados por curso. Mostrar los nombres de los alumnos matriculados en cada uno de los cursos.
- Docentes con carga horaria. Mostrar los nombres completos de todos los docentes con curso asignado.

## Tabla alumno

Atributo	Tipo	Observación
alu_id	int	PK correlativo (202210001)
alu_nombres	Varchar(100)	
alu_apellido_p	Varchar(100)	
alu_apellido_m	Varchar(100)	
alu_dni	Varchar(8)	
alu_edad	Int(2)	
alu_celular	Varchar(9)	
alu_estado	Char(1)	0 = retirado 1 = registrado 2 = matriculado

### **Tabla curso**

Atributo	Tipo	Observación
cur_id	Int	PK correlativo (101)
cur_nom	Varchar(100)	
cur_ciclo	Int(1)	1 = primero 2 = segundo ... 6 = sexto
cur_creditos	Int(1)	
cur_horas	Int(1)	
cur_estado	Char(1)	0 = retirado 1 = registrado

### **Tabla docente**

Atributo	Tipo	Observación
doc_id	Int	PK correlativo (2001)
doc_nombres	Varchar(100)	
doc_apellido_p	Varchar(100)	
doc_apellido_m	Varchar(100)	
doc_dni	Varchar(8)	
doc_fecha_ing	Date	
doc_celular	Varchar(9)	
doc_especialidad	Varchar(100)	
doc_estado	Char(1)	0 = retirado 1 = registrado

### **Tabla matrícula**

Atributo	Tipo	Observación
mat_id	Int	PK correlativo (200001)
alu_id	Int	FK(alumno)
cur_id	Int	FK(curso)
doc_id	Int	FK(docente)
mat_fecha	Date	Current Time
mat_estado	Char(1)	0 = retirado 1 = matriculado




### **Tabla tipo**

Atributo	Tipo	Observación
tipo_id	Int(1)	PK 1 = empleado 2 = administrador
tipo_des	Varchar(50)	

### **Tabla usuario**

Atributo	Tipo	Observación
usu_id	Int(11)	PK correlativo (1)
usu_nom	Varchar(10)	Unique
usu_contra	Varchar(6)	
usu_tipo	char(1)	FK(tipo)
usu_foto	blob	

  
USER\_GENERIC

CONSULTA

MANTENIMIENTO

REGISTRO

REPORTE

ACERCA DE

"Sé el protagonista de tu propio Aprendizaje"

CONSULTA

ALUMNOS

BUSCAR

CURSOS MATRICULADOS

Código	Asignatura	Ciclo	N° Créditos	Cant. Horas	N° Alumnos
--------	------------	-------	-------------	-------------	------------

DATOS DEL ALUMNO

Código:

DNI:

Nombre(s):

Ap. Paterno:

Ap. Materno:


Edad:


Celular:

Estado:


## Consulta

Permite visualizar la información referente a los alumnos, docentes, cursos, matrículas y retiros respectivamente.






USER\_GENERIC



# VIRTUAL DATA



## MANTENIMIENTO

ALUMNO

DATOS DEL ALUMNO

Código:

Nombre(s):


Ap. Paterno:


Ap. Materno:


DNI:


Edad:

Celular:


CONSULTAR :


ADICIONAR


MODIFICAR


ELIMINAR

ID	Nombres	Ap. Paterno	Ap. Materno	DNI	Edad	Celular	Estado
1	JOSÉ	ROBERTO	MENDIETA	25554188	18	965148222	1
2	MARÍA	SANCHEZ	RAMOS	16584050	21	993551404	1
3	CESAR	MUÑOZ	VIVANCO	40001524	26	931022845	1
4	ALEXANDRA	RAMOS	HUAMAN	10230088	30	966585147	1
5	NELLY	SALVATIER	CABRERA	24775844	19	964074447	1

"Sé el protagonista de tu propio Aprendizaje"

Figura N° 3: Módulo Consulta

## Mantenimiento

Permite realizar el mantenimiento de la información correspondiente a alumnos, cursos y docentes; para lo cual se debe ingresar con el perfil de administrador.

Código	Nombre(s)	Ap. Paterno	Ap. Materno	DNI	Edad	Celular	Estado
202110001	Carlos Rolando	Mendo	Bravo	10013698	18	999888777	Matriculado
202110002	David Elias	Espinoza	Berrospl	10013999	22	920526766	Registrado
202110003	Jose Alberto	Carbonel	Travesaño	20013729	25	999528721	Matriculado
202110004	Luis Ernesto	Camacho	Ramirez	30013711	29	979528745	Matriculado
202110005	Ernesto Juan	Quezada	Martinez	40099712	32	999526725	Retrado
202110006	Jorge Alberto	Luna	Quezada	44445244	18	987654321	Matriculado
202110007	Miguel Angelo	Cueva	Rivera	12254366	32	913254110	Retrado
202110008	Cesar	Gomez	Martinez	14223095	23	966120011	Registrado

Figura N° 4: Módulo Mantenimiento

## Registro

Este módulo permite realizar matrículas y retiros; para lo cual se debe ingresar con el perfil de administrador.

N° Matricula	Código Alumno	Código Curso	Código Docente	Fecha	Hora
200001	202110001	1018	2001	05/10/2021	05:16:15
200002	202110001	1047	2004	10/10/2021	08:20:20
200003	202110004	1031	2003	12/10/2021	09:12:10
200006	202110003	1018	2001	13/10/2021	11:11:23
200007	202110006	1047	2004	26/11/2021	05:37:09
200008	202110002	1031	2005	28/11/2021	22:40:18
200010	202110006	1031	2005	29/11/2021	10:44:50

Figura N° 5: Módulo Registro

## Reporte

Este módulo permite obtener diversos reportes requeridos por la administración.



Figura N° 6: Módulo Reporte

## CODIGO FUENTE

```
package entidades;

public class AlumnoEntity extends PersonaEntity {

    private int edad;
    //int id, String nombres, String apellidoPaterno, String apellidoMaterno, String
    dni, int edad,String celular, int estado
    public AlumnoEntity(int id, String nombres, String apellidoPaterno, String
    apellidoMaterno, String dni, int edad,
    String celular, int estado) {
        super(id, nombres, apellidoPaterno, apellidoMaterno, dni, celular, estado);
        this.edad = edad;
    }

    public AlumnoEntity(String nombres, String apellidoPaterno, String apellido-
    Materno, String dni, int edad,
    String celular) {
        super(nombres, apellidoPaterno, apellidoMaterno, dni, celular);
        this.edad = edad;
    }

    public AlumnoEntity(String nombres, String apellidoPaterno, String apellidoMa-
    terno) {
        super(nombres, apellidoPaterno, apellidoMaterno);
    }
}
```

```
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
}

}

package entidades;

public class CursoEntity {

    private int curId;
    private String curNombre;
    private int curCiclo;
    private int curCreditos;
    private int curHoras;
    private int curEstado;

    public CursoEntity(int curId, String curNombre) {
        super();
        this.curId = curId;
        this.curNombre = curNombre;
    }

    public CursoEntity(String curNombre, int curCiclo, int curCreditos, int curHoras) {
        super();
        this.curNombre = curNombre;
        this.curCiclo = curCiclo;
        this.curCreditos = curCreditos;
        this.curHoras = curHoras;
    }

    public CursoEntity() {
    }

    public CursoEntity(int curId, String curNombre, int curCiclo, int curCreditos, int
        curHoras, int curEstado) {
        super();
        this.curId = curId;
        this.curNombre = curNombre;
        this.curCiclo = curCiclo;
        this.curCreditos = curCreditos;
        this.curHoras = curHoras;
        this.curEstado = curEstado;
    }

    public int getCurId() {
        return curId;
    }

    public void setCurId(int curId) {
```

```
this.curId = curId;
}

public String getCurNombre() {
return curNombre;
}

public void setCurNombre(String curNombre) {
this.curNombre = curNombre;
}

public int getCurCiclo() {
return curCiclo;
}

public void setCurCiclo(int curCiclo) {
this.curCiclo = curCiclo;
}

public int getCurCreditos() {
return curCreditos;
}

public void setCurCreditos(int curCreditos) {
this.curCreditos = curCreditos;
}

public int getCurHoras() {
return curHoras;
}

public void setCurHoras(int curHoras) {
this.curHoras = curHoras;
}

public int getCurEstado() {
return curEstado;
}

public void setCurEstado(int curEstado) {
this.curEstado = curEstado;
}

}

package entidades;

import java.util.Date;

public class DocenteEntity extends PersonaEntity {

private Date fechaIngreso;
private String especialidad;

public DocenteEntity(String nombres, String apellidoPaterno, String apellido-
Materno, String dni, Date fechaIngreso, String celular, String especialidad) {
super(nombres, apellidoPaterno, apellidoMaterno, dni, celular);
this.fechaIngreso = fechaIngreso;
this.especialidad = especialidad;
}

public DocenteEntity(int id, String nombres, String apellidoPaterno, String
apellidoMaterno, String dni, Date fechaIngreso,
String celular, String especialidad, int estado) {
super(id, nombres, apellidoPaterno, apellidoMaterno, dni, celular, estado);
```

```
this.fechaIngreso = fechaIngreso;
this.especialidad = especialidad;
}

public DocenteEntity(String nombres, String apellidoPaterno, String apellido-
Materno) {
    super(nombres, apellidoPaterno, apellidoMaterno);
}

public Date getFechaIngreso() {
    return fechaIngreso;
}

public void setFechaIngreso(Date fechaIngreso) {
    this.fechaIngreso = fechaIngreso;
}

public String getEspecialidad() {
    return especialidad;
}

public void setEspecialidad(String especialidad) {
    this.especialidad = especialidad;
}

}

package entidades;

import java.util.Date;

public class MatriculaEntity {

    private int matId;
    private int aluId;
    private int curId;
    private int docId;
    private Date fecha;
    private int matEstado;

    public MatriculaEntity(int matId) {
        super();
        this.matId = matId;
    }

    public MatriculaEntity(int matId, int matEstado) {
        super();
        this.matId = matId;
        this.matEstado = matEstado;
    }

    public MatriculaEntity(int matId, int aluId, int curId, int docId, Date fecha) {
        super();
    }
}
```

```
this.matId = matId;
this.aluId = aluId;
this.curId = curId;
this.docId = docId;
this.fecha = fecha;
}

public MatriculaEntity(int aluId, int curId, int docId) {
    super();
    this.aluId = aluId;
    this.curId = curId;
    this.docId = docId;
}

public MatriculaEntity(int matId, int aluId, int curId, int docId, Date fecha, int
matEstado) {
    super();
    this.matId = matId;
    this.aluId = aluId;
    this.curId = curId;
    this.docId = docId;
    this.fecha = fecha;
    this.matEstado = matEstado;
}

public int getMatId() {
    return matId;
}

public void setMatId(int matId) {
    this.matId = matId;
}

public int getAluId() {
    return aluId;
}

public void setAluId(int aluId) {
    this.aluId = aluId;
}

public int getCurId() {
    return curId;
}

public void setCurId(int curId) {
    this.curId = curId;
}

public int getDocId() {
    return docId;
}

public void setDocId(int docId) {
    this.docId = docId;
}

public Date getFecha() {
    return fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

public int getMatEstado() {
    return matEstado;
}
```



```
}  
public void setMatEstado(int matEstado) {  
    this.matEstado = matEstado;  
}  
  
package gestion;  
  
import java.sql.CallableStatement;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.util.ArrayList;  
import java.util.Date;  
  
import entidades.AlumnoEntity;  
import entidades.CursoEntity;  
import interfaces.AlumnoInterface;  
import util.MySQLConexion;  
  
public class GestionAlumno implements AlumnoInterface{  
  
    @Override  
    public ArrayList<AlumnoEntity> listarAlumnos() {  
  
        ArrayList<AlumnoEntity> listaAlumnos = new ArrayList<AlumnoEntity>();  
  
        Connection cn = null;  
  
        CallableStatement csmt = null;  
        ResultSet rs = null;  
  
        String mysql = "{call usp_listAlumno}";  
  
        try {  
  
            cn = MySQLConexion.getConnection();  
            csmt = cn.prepareCall(mysql);  
            rs = csmt.executeQuery();  
  
            while (rs.next()) {  
                AlumnoEntity alumnoEntity =  
                    new AlumnoEntity(  
                        rs.getInt(1), //Se puede poner el nombre  
de la columna como lo hizo el profesor en clase, o también el número de columna  
como yo lo he puesto acá, ustedes elijan  
                        rs.getString(2),  
                        rs.getString(3),  
                        rs.getString(4),  
                        rs.getString(5),  
                        rs.getInt(6),  
                        rs.getString(7),  
                        rs.getInt(8)  
                    );  
                listaAlumnos.add(alumnoEntity);  
            }  
        }  
    }  
}
```

```

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaAlumnos;
}

@Override
public int registrarAlumno(AlumnoEntity alumno) {

    int resultado = 0;

    Connection cn = null;

    //CON PROCEDURES
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_registrarAlumno(?,?,?,?,?,?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setString(1, alumno.getNombres());
        csmt.setString(2, alumno.getApellidoPaterno());
        csmt.setString(3, alumno.getApellidoMaterno());
        csmt.setString(4, alumno.getDni());
        csmt.setInt(5, alumno.getEdad());
        csmt.setString(6, alumno.getCelular());

        resultado = csmt.executeUpdate(); //devuelve 1 si la insercción fue
correcta

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }

    return resultado;
}

@Override
public int editarAlumno(AlumnoEntity alumno) {

```

```
int resultado = 0;

Connection cn = null;
CallableStatement csmt = null;

try {
    cn = MySQLConexion.getConnection();
    String mysql = "call usp_actualizarAlumno(?,?,?,?,?,?,?,?)";

    csmt = cn.prepareCall(mysql);
    csmt.setInt(1, alumno.getId());
    csmt.setString(2, alumno.getNombres());
    csmt.setString(3, alumno.getApellidoPaterno());
    csmt.setString(4, alumno.getApellidoMaterno());
    csmt.setString(5, alumno.getDni());
    csmt.setInt(6, alumno.getEdad());
    csmt.setString(7, alumno.getCelular());
    csmt.setInt(8, alumno.getEstado());

    resultado = csmt.executeUpdate();

} catch (Exception e) {
    e.printStackTrace();
} finally {
    //liberar los recursos que hemos utilizado
    try {
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
return resultado;
}

@Override
public int eliminarAlumno(int idAlumno) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_eliminarAlumno(?)";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idAlumno);

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
```

```
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
    return resultado;
}

@Override
public ArrayList<AlumnoEntity> buscarAlumnos(int idAlumno) {

    ArrayList<AlumnoEntity> listaAlumno = new ArrayList<AlumnoEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    try {

        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_buscarAlumnoActivo(?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idAlumno);
        rs = csmt.executeQuery();
        //int id, String nombres, String apellidoPaterno, String apellido-
Materno, String dni, int edad,
        //String celular, int estado
        while (rs.next()) {
            AlumnoEntity alumnoEntity =
                new AlumnoEntity(
                    rs.getInt(1),
                    rs.getString(2),
                    rs.getString(3),
                    rs.getString(4),
                    rs.getString(5),
                    rs.getInt(6),
                    rs.getString(7),
                    rs.getInt(8)
                );
            listaAlumno.add(alumnoEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}
```

```
        return listaAlumno;
    }

    @Override
    public ArrayList<CursoEntity> cursosAlumnoMatriculado(int idAlumno) {

        ArrayList<CursoEntity> listaCurso = new ArrayList<CursoEntity>();

        Connection cn = null;

        CallableStatement csmt = null;
        ResultSet rs = null;

        try {

            cn = MySQLConexion.getConnection();
            String mysql = "{call usp_consultaCursoAlumno(?)}";
            csmt = cn.prepareCall(mysql);
            csmt.setInt(1, idAlumno);
            rs = csmt.executeQuery();
            //int id, String nombres, String apellidoPaterno, String apellido-
Materno, String dni, int edad,
            //String celular, int estado
            while (rs.next()) {
                CursoEntity cursoEntity = new CursoEntity();
                cursoEntity.setCurId(rs.getInt(1));
                cursoEntity.setCurNombre(rs.getString(2));
                cursoEntity.setCurCiclo(rs.getInt(3));
                cursoEntity.setCurCreditos(rs.getInt(4));
                cursoEntity.setCurHoras(rs.getInt(5));
                listaCurso.add(cursoEntity);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (rs != null) rs.close();
                if (csmt != null) csmt.close();
                if (cn != null) cn.close();
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }

        return listaCurso;
    }

    @Override
    public ArrayList<AlumnoEntity> listarAlumnosMatriculaPendiente() {

        ArrayList<AlumnoEntity> listaAlumno = new ArrayList<AlumnoEntity>();

        Connection cn = null;

        //CON PROCEDURES
```

```
CallableStatement csmt = null;
ResultSet rs = null;

String mysql = "{call usp_reporteAlumnosMatriculaPendiente}";

try {

    cn = MySQLConexion.getConnection();
    csmt = cn.prepareCall(mysql);
    rs = csmt.executeQuery();

    while (rs.next()) {
        AlumnoEntity alumnoEntity =
            new AlumnoEntity(
                rs.getInt(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4),
                rs.getString(5),
                rs.getInt(6),
                rs.getString(7),
                rs.getInt(8)
            );
        listaAlumno.add(alumnoEntity);
    }

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (rs != null) rs.close();
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

return listaAlumno;
}

@Override
public ArrayList<AlumnoEntity> listarAlumnosMatriculaVigente() {

    ArrayList<AlumnoEntity> listaAlumno = new ArrayList<AlumnoEntity>();

    Connection cn = null;

    //CON PROCEDURES
    CallableStatement csmt = null;
    ResultSet rs = null;

    String mysql = "{call usp_reporteAlumnosMatriculaVigente}";

    try {

        cn = MySQLConexion.getConnection();
```

```
        csmt = cn.prepareStatement(mysql);
        rs = csmt.executeQuery();

        while (rs.next()) {
            AlumnoEntity alumnoEntity =
                new AlumnoEntity(
                    rs.getInt(1),
                    rs.getString(2),
                    rs.getString(3),
                    rs.getString(4),
                    rs.getString(5),
                    rs.getInt(6),
                    rs.getString(7),
                    rs.getInt(8)
                );
            listaAlumno.add(alumnoEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaAlumno;
}

@Override
public ArrayList<AlumnoEntity> listarAlumnosRetirados() {

    ArrayList<AlumnoEntity> listaAlumno = new ArrayList<AlumnoEntity>();

    Connection cn = null;

    //CON PROCEDURES
    CallableStatement csmt = null;
    ResultSet rs = null;

    String mysql = "{call usp_reporteAlumnosRetirados}";

    try {

        cn = MySQLConexion.getConnection();
        csmt = cn.prepareCall(mysql);
        rs = csmt.executeQuery();

        while (rs.next()) {
            AlumnoEntity alumnoEntity =
                new AlumnoEntity(
                    rs.getInt(1),
                    rs.getString(2),
```

```

        rs.getString(3),
        rs.getString(4),
        rs.getString(5),
        rs.getInt(6),
        rs.getString(7),
        rs.getInt(8)
    );
    listaAlumno.add(alumnoEntity);
}

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (rs != null) rs.close();
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

return listaAlumno;
}

}
}

}

package gestion;

import java.sql.Connection;
import java.sql.ResultSet;
import java.util.ArrayList;

import java.sql.CallableStatement;

import entidadesCursoEntity;
import entidades.DocenteEntity;
import interfaces.CursoInterface;
import util.MySQLConexion;

public class GestionCurso implements CursoInterface{

    @Override
    public ArrayList<CursoEntity> listarCursos() {

        ArrayList<CursoEntity> listaCursos = new ArrayList<CursoEntity>();

        Connection cn = null;

        CallableStatement csmt = null;
        ResultSet rs = null;

```



```
String mysql = "{call usp_listCurso}";

try {

    cn = MySQLConexion.getConnection();
    csmt = cn.prepareCall(mysql);
    rs = csmt.executeQuery();

    while (rs.next()) {
        CursoEntity cursoEntity =
            new CursoEntity(
                rs.getInt(1),
                rs.getString(2),
                rs.getInt(3),
                rs.getInt(4),
                rs.getInt(5),
                rs.getInt(6)
            );
        listaCursos.add(cursoEntity);
    }

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (rs != null) rs.close();
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

return listaCursos;
}

@Override
public int registrarCurso(CursoEntity curso) {

    int resultado = 0;

    Connection cn = null;

    //CON PROCEDURES
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_registrarCurso(?,?,?,?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setString(1, curso.getCurNombre());
        csmt.setInt(2, curso.getCurCiclo());
        csmt.setInt(3, curso.getCurCreditos());
```

```
        csmt.setInt(4, curso.getCurHoras());

        resultado = csmt.executeUpdate(); //devuelve 1 si la insercción fue correcta

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }

    return resultado;
}

@Override
public int editarCurso(CursoEntity curso) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_actualizarCurso(?,?,?,?,?,?)";

        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, curso.getCurId());
        csmt.setString(2, curso.getCurNombre());
        csmt.setInt(3, curso.getCurCiclo());
        csmt.setInt(4, curso.getCurCreditos());
        csmt.setInt(5, curso.getCurHoras());
        csmt.setInt(6, curso.getCurEstado());

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}
```

```
    }
    return resultado;
}

@Override
public int eliminarCurso(int idCurso) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_eliminarCurso(?)";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idCurso);

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
    return resultado;
}

@Override
public ArrayList<CursoEntity> buscarCursos(int idCurso) {

    ArrayList<CursoEntity> listaCurso = new ArrayList<CursoEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    try {

        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_consultaCurso(?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idCurso);
        rs = csmt.executeQuery();

    }
```

```
        while (rs.next()) {
            CursoEntity cursoEntity = new CursoEntity();
            cursoEntity.setCurId(rs.getInt(1));
            cursoEntity.setCurNombre(rs.getString(2));
            cursoEntity.setCurCiclo(rs.getInt(3));
            cursoEntity.setCurCreditos(rs.getInt(4));
            cursoEntity.setCurHoras(rs.getInt(5));
            listaCurso.add(cursoEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaCurso;
}

@Override
public ArrayList<DocenteEntity> docenteCurso(int idCurso) {

    ArrayList<DocenteEntity> listaDocente = new ArrayList<DocenteEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    try {

        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_consultaDocenteCurso(?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idCurso);
        rs = csmt.executeQuery();

        while (rs.next()) {
            DocenteEntity docenteEntity = new DocenteEntity(
                rs.getInt(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4),
                rs.getString(5),
                rs.getDate(6),
                rs.getString(7),
                rs.getString(8),
```

```
                1
                );
            listaDocente.add(docenteEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaDocente;
}

}

package gestion;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.util.ArrayList;

import entidadesCursoEntity;
import entidadesDocenteEntity;
import interfaces.DocenteInterface;
import util.MySQLConexion;

public class GestionDocente implements DocenteInterface{

    @Override
    public ArrayList<DocenteEntity> listarDocentes() {

        ArrayList<DocenteEntity> listaDocentes = new ArrayList<DocenteEntity>();

        Connection cn = null;

        CallableStatement csmt = null;
        ResultSet rs = null;

        String mysql = "{call usp_listDocente}";

        try {
```

```
        cn = MySQLConexion.getConnection();
        csmt = cn.prepareCall(mysql);
        rs = csmt.executeQuery();

        while (rs.next()) {
            DocenteEntity docenteEntity =
                new DocenteEntity(
                    rs.getInt(1),
                    rs.getString(2),
                    rs.getString(3),
                    rs.getString(4),
                    rs.getString(5),
                    rs.getDate(6),
                    rs.getString(7),
                    rs.getString(8),
                    rs.getInt(9)
                );
            listaDocentes.add(docenteEntity);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaDocentes;
}

@Override
public int registrarDocente(DocenteEntity docente) {

    int resultado = 0;

    Connection cn = null;

    //CON PROCEDURES
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_registrarDocente(?,?,?,?,?,?,?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setString(1, docente.getNombres());
        csmt.setString(2, docente.getApellidoPaterno());
        csmt.setString(3, docente.getApellidoMaterno());
        csmt.setString(4, docente.getDni());
```

```

        Date dateSqlDate = new Date(docente.getFechaIngreso().getTime());
        csmt.setDate(5, dateSqlDate);
        csmt.setString(6, docente.getCelular());
        csmt.setString(7, docente.getEspecialidad());

        resultado = csmt.executeUpdate(); //devuelve 1 si la insercción fue correcta

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }

    return resultado;
}

@Override
public int editarDocente(DocenteEntity docente) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_actualizarDocente(?,?,?,?,?,?,?,?)";

        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, docente.getId());
        csmt.setString(2, docente.getNombres());
        csmt.setString(3, docente.getApellidoPaterno());
        csmt.setString(4, docente.getApellidoMaterno());
        csmt.setString(5, docente.getDni());
        Date dateSqlDate = new Date(docente.getFechaIngreso().getTime());
        csmt.setDate(6, dateSqlDate);
        csmt.setString(7, docente.getCelular());
        csmt.setString(8, docente.getEspecialidad());
        csmt.setInt(9, docente.getEstado());

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado

```

```
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
    return resultado;
}
```

```
@Override
public int eliminarDocente(int idDocente) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_eliminarDocente(?)";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idDocente);

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
    return resultado;
}
```

```
@Override
public ArrayList<DocenteEntity> buscarDocentes(int idDocente) {

    ArrayList<DocenteEntity> listaDocente = new ArrayList<DocenteEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    try {
```



```
        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_consultaDocente(?)}";
        csmt = cn.prepareStatement(mysql);
        csmt.setInt(1, idDocente);

        rs = csmt.executeQuery();

        while (rs.next()) {
            DocenteEntity docenteEntity = new DocenteEntity(
                rs.getInt(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4),
                rs.getString(5),
                rs.getDate(6),
                rs.getString(7),
                rs.getString(8),
                1
            );
            listaDocente.add(docenteEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaDocente;
}

@Override
public ArrayList<CursoEntity> cursosDocente(int idDocente) {
    // TODO Auto-generated method stub
    return null;
}

@Override
public ArrayList<DocenteEntity> listarDocentesConHorarios() {

    ArrayList<DocenteEntity> listaDocentes = new ArrayList<DocenteEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;
```

```
String mysql = "{call usp_reporteDocenteConHorarios}";

try {

    cn = MySQLConexion.getConnection();
    csmt = cn.prepareCall(mysql);
    rs = csmt.executeQuery();

    while (rs.next()) {
        DocenteEntity docenteEntity =
            new DocenteEntity(
                rs.getInt(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4),
                rs.getString(5),
                rs.getDate(6),
                rs.getString(7),
                rs.getString(8),
                rs.getInt(9)
            );
        listaDocentes.add(docenteEntity);
    }

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (rs != null) rs.close();
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

return listaDocentes;
}

}
```

```
package gestion;
```

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.util.ArrayList;
```

```
import entidades.AlumnoEntity;
import entidadesCursoEntity;
import entidades.DocenteEntity;
```

```
import entidades.MatriculaEntity;
import interfaces.MatriculaInterface;
import util.MySQLConexion;

public class GestionMatricula implements MatriculaInterface{

    @Override
    public ArrayList<MatriculaEntity> listarMatricula() {

        ArrayList<MatriculaEntity> listaMatriculas = new ArrayList<MatriculaEntity>();

        Connection cn = null;

        CallableStatement csmt = null;
        ResultSet rs = null;

        String mysql = "{call usp_listMatricula}";

        try {

            cn = MySQLConexion.getConexion();
            csmt = cn.prepareCall(mysql);
            rs = csmt.executeQuery();

            while (rs.next()) {
                MatriculaEntity alumnoEntity =
                    new MatriculaEntity(
                        rs.getInt(1),
                        rs.getInt(2),
                        rs.getInt(3),
                        rs.getInt(4),
                        rs.getDate(5),
                        rs.getInt(6)
                    );
                listaMatriculas.add(alumnoEntity);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (rs != null) rs.close();
                if (csmt != null) csmt.close();
                if (cn != null) cn.close();
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }

        return listaMatriculas;
    }

    @Override
```

```
public int registrarMatricula(MatriculaEntity matricula) {

    int resultado = 0;

    Connection cn = null;

    //CON PROCEDURES
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_registrarMatricula(?,?,?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, matricula.getAluld());
        csmt.setInt(2, matricula.getCurId());
        csmt.setInt(3, matricula.getDocId());

        resultado = csmt.executeUpdate(); //devuelve 1 si la insercción fue correcta

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }

    return resultado;
}
```

```
@Override
public int editarMatricula(MatriculaEntity matricula) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_actualizarMatricula(?,?,?,?);";

        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, matricula.getMatId());
        csmt.setInt(2, matricula.getAluld());
        csmt.setInt(3, matricula.getCurId());
        csmt.setInt(4, matricula.getDocId());
    }
```

```
        Date dateSqlDate = new Date(matricula.getFecha().getTime());
        csmt.setDate(5, dateSqlDate);

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
    return resultado;
}
```

```
@Override
public int eliminarMatricula(int idMatricula) {

    int resultado = 0;

    Connection cn = null;
    CallableStatement csmt = null;

    try {
        cn = MySQLConexion.getConnection();
        String mysql = "call usp_eliminarMatricula(?)";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idMatricula);

        resultado = csmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //liberar los recursos que hemos utilizado
        try {
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
    return resultado;
}
```

```
@Override
public String nombreAlumno(int idAlumno) {
```

```
String nomAlumno = "";

Connection cn = null;

CallableStatement csmt = null;
ResultSet rs = null;

String mysql = "{call usp_nombreAlumno(?)}";

try {

    cn = MySQLConexion.getConexion();
    csmt = cn.prepareCall(mysql);
    csmt.setInt(1, idAlumno);
    rs = csmt.executeQuery();

    if (rs.next()) {
        AlumnoEntity entity = new AlumnoEntity(
            rs.getString(1),
            rs.getString(2),
            rs.getString(3)
        );
        nomAlumno = entity.getNombres()+" "+entity.getApellidoPaterno()+"
"+entity.getApellidoMaterno();
    }

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (rs != null) rs.close();
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

return nomAlumno;
}

@Override
public String nombreCurso(int idCurso) {

    String nomCurso = "";

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    String mysql = "{call usp_nombreCurso(?)}";
```

```
try {

    cn = MySQLConexion.getConnection();
    csmt = cn.prepareCall(mysql);
    csmt.setInt(1, idCurso);
    rs = csmt.executeQuery();

    if (rs.next()) {
        CursoEntity entity = new CursoEntity();
        entity.setCurNombre(rs.getString(1));
        nomCurso = entity.getCurNombre();
    }

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (rs != null) rs.close();
        if (csmt != null) csmt.close();
        if (cn != null) cn.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

return nomCurso;
}

@Override
public String nombreDocente(int idDocente) {

    String nomDocente = "";

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    String mysql = "{call usp_nombreDocente(?)}";

    try {

        cn = MySQLConexion.getConnection();
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idDocente);
        rs = csmt.executeQuery();

        if (rs.next()) {
            DocenteEntity entity = new DocenteEntity(
                rs.getString(1),
                rs.getString(2),
                rs.getString(3)
            );
        }
    }
```

```

        nomDocente = entity.getNombres()+" "+entity.getApellidoPaterno()+"
"+entity.getApellidoMaterno();
    }

```

```

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return nomDocente;
}

```

@Override

```
public Boolean verificarMatricula(int idAlumno, int idCurso) {
```

```
    boolean verificar = false;
```

```
    Connection cn = null;
```

```
    CallableStatement csmt = null;
```

```
    ResultSet rs = null;
```

```
    String mysql = "{call usp_verificarMatricula(?,?)}";
```

```
    try {
```

```
        cn = MySQLConexion.getConnection();
```

```
        csmt = cn.prepareCall(mysql);
```

```
        csmt.setInt(1, idAlumno);
```

```
        csmt.setInt(2, idCurso);
```

```
        rs = csmt.executeQuery(); //devuelve filas de datos
```

```
        if (rs.next()) { //rs.next() indica que tiene elementos que le van a permitir ite-
```

rar

```
            verificar = true;
```

```
        }
```

```
    } catch (Exception e) {
```

```
        e.printStackTrace();
```

```
    } finally {
```

```
        //liberar los recursos que hemos utilizado
```

```
        try {
```

```
            if (rs != null) rs.close();
```

```
            if (csmt != null) csmt.close();
```

```
            if (cn != null) cn.close();
```

```
        } catch (Exception e2) {
```

```
            // TODO: handle exception
```



```
                e2.printStackTrace();
            }
        }

        return verificar;
    }

//CONSULTAS
@Override
public ArrayList<AlumnoEntity> consultaAlumnoMatricula(int idMatricula) {

    ArrayList<AlumnoEntity> listaAlumno = new ArrayList<AlumnoEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    try {

        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_consultaAlumnoMatricula(?,?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idMatricula);
        csmt.setInt(2, 1);
        rs = csmt.executeQuery();

        while (rs.next()) {
            AlumnoEntity alumnoEntity = new AlumnoEntity(
                rs.getInt(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4),
                rs.getString(5),
                rs.getInt(6),
                rs.getString(7),
                1
            );
            listaAlumno.add(alumnoEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}
```

```
        return listaAlumno;
    }

    @Override
    public ArrayList<DocenteEntity> consultaDocenteMatricula(int idMatricula) {

        ArrayList<DocenteEntity> listaDocente = new ArrayList<DocenteEntity>();

        Connection cn = null;

        CallableStatement csmt = null;
        ResultSet rs = null;

        try {

            cn = MySQLConexion.getConnection();
            String mysql = "{call usp_consultaDocenteMatricula(?,?)}";
            csmt = cn.prepareCall(mysql);
            csmt.setInt(1, idMatricula);
            csmt.setInt(2, 1);
            rs = csmt.executeQuery();

            while (rs.next()) {
                DocenteEntity docenteEntity = new DocenteEntity(
                    rs.getInt(1),
                    rs.getString(2),
                    rs.getString(3),
                    rs.getString(4),
                    rs.getString(5),
                    rs.getDate(6),
                    rs.getString(7),
                    rs.getString(8),
                    1
                );
                listaDocente.add(docenteEntity);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (rs != null) rs.close();
                if (csmt != null) csmt.close();
                if (cn != null) cn.close();
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }

        return listaDocente;
    }

    @Override
```

```
public ArrayList<CursoEntity> consultaCursoMatricula(int idMatricula) {

    ArrayList<CursoEntity> listaCurso = new ArrayList<CursoEntity>();

    Connection cn = null;

    CallableStatement csmt = null;
    ResultSet rs = null;

    try {

        cn = MySQLConexion.getConnection();
        String mysql = "{call usp_consultaCursoMatricula(?,?)}";
        csmt = cn.prepareCall(mysql);
        csmt.setInt(1, idMatricula);
        csmt.setInt(2, 1);
        rs = csmt.executeQuery();

        while (rs.next()) {
            CursoEntity cursoEntity = new CursoEntity();
            cursoEntity.setCurId(rs.getInt(1));
            cursoEntity.setCurNombre(rs.getString(2));
            cursoEntity.setCurCiclo(rs.getInt(3));
            cursoEntity.setCurCreditos(rs.getInt(4));
            cursoEntity.setCurHoras(rs.getInt(5));
            listaCurso.add(cursoEntity);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) rs.close();
            if (csmt != null) csmt.close();
            if (cn != null) cn.close();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }

    return listaCurso;
}

}
```

## **CONCLUSIONES**

- Este proyecto fue 100% virtual sin la necesidad de hacer reuniones presenciales por parte de los integrantes del grupo. Todo esto debido a la coyuntura que estamos atravesando, pero a pesar de estas limitaciones el proyecto fue realizado exitosamente.
- Todos los integrantes han aportado ideas desde el análisis de las necesidades del negocio hasta la escritura del código fuente e integración de las distintas partes del programa.
- El proyecto tendrá un impacto positivo en vuestra formación como estudiantes del curso, debido a que afianzaremos el conocimiento impartido en las clases en la elaboración de programas para optimizar procesos de negocios o trabajos en nuestros centros de labores.
- Pensamos que, a la velocidad que se mueve la economía digital, todas las empresas deben incorporar en su núcleo el desarrollo de software a fin que siempre puedan estar innovando y ofreciendo mejores productos y servicios a los clientes.
- Las empresas demandan herramientas que puedan enriquecer las cadenas de valor que les permita ser altamente competitivos y el proyecto en curso les va permitir encastrarlos hacia ese objetivo.

## **RECOMENDACIONES**

- El proyecto se ha programado para ser ejecutado únicamente en una computadora de escritorio con Sistema operativo Windows 10 o Mac.
- Definir variables generales para todos los integrantes del grupo, para que al momento de hacer la repartición de las tareas utilicen las mismas y al integrar el proyecto todo quede sin ningún error.
- Se recomienda poner en funcionamiento la aplicación a fin de acercarlo con la realidad y poder contrastar los beneficios, puntos en contra y oportunidades de mejora.
- Se recomienda mantenerse en contacto permanente con el código fuente, revisarlo, analizarlos, buscar los puntos de mejora y simplificación. Tener siempre en mente la mejora continua.

## **BIBLIOGRAFÍA**

JOYANES AGUILAR, Luís

2008 Fundamentos de programación: algoritmos, estructuras de datos y objetos.  
Madrid, España: McGraw-Hill 2003  
(005.1 JOYA/A 2008)

DEITEL, Harvey

2008 Cómo programar en Java  
México, D.F.: Pearson Educación.  
(005.133J DEIT 2008)

Holzner, S.

(2000). La Biblia de Java 2. Anaya Multimedia, S.A. <https://books.google.com.pe/books?id=2qmHAAAACAAJ>

La Web del Programador

El portal de los programadores. (s. f.). Recuperado 4 de agosto de 2021, de <https://www.lawebdelprogramador.com/>

pildorasinformaticas. (s. f.).

Curso Java. Presentación. Vídeo 1. Recuperado 4 de agosto de 2021, de <https://www.youtube.com/watch?v=coK4jM5wvko>

The Java Programming Language. (2000).

Pearson Education. <https://books.google.com.pe/books?id=6vwJIH1aYCUC>

Martínez Alvarado, J. I., & Rivera Castillo, O. W. (2012).

Desarrollo de un sistema para la gestión de ventas de servicios y productos de la empresa Clínica del Pie (Doctoral dissertation, Universidad Centroamericana)