

Facultad: Ciencias Técnicas

Escuela: Ciencias de la computación

Período académico: Marzo- Julio 2025

Docentes: Marcela Venegas, Ronnie Martinez

Asignatura: Estructura, modelado almacenamiento de base de datos, Programación estructurada y funcional

Actividad: Proyecto de Fin de Parcial

Integrantes: Ariel Melo , Mateo Yáñez, Maria Chango

Proyecto Integrador: “Sistema de Gestión de Notas Universitarias”

Tabla de Contenidos

1. **Introducción** 1.1. Planteamiento del Problema 1.2. Justificación y Alcance del Proyecto 1.3. Objetivos
2. **Fase 1: Planificación y Modelado de la Base de Datos** 2.1. Identificación de Usuarios y Necesidades de Información 2.2. Diseño Conceptual y Lógico: Diagrama Entidad-Relación (ER) 2.3. Normalización de la Base de Datos
3. **Fase 2: Implementación de la Base de Datos** 3.1. Estructura de Tablas y Script DDL 3.2. Población de la Base de Datos (Script DML) 3.3. Consultas de Prueba (Queries)
4. **Fase 3: Desarrollo de la Aplicación en Java** 4.1. Arquitectura del Software y Patrones de Diseño 4.1.1. Patrón Modelo-Vista-Controlador (MVC) 4.1.2. Abstracción del Acceso a Datos (DAO) 4.1.3. Principios de Diseño SOLID 4.2. Diseño e Implementación de la Interfaz de Usuario (UI) 4.3. Implementación de la Lógica de Negocio
5. **Pruebas y Validación del Sistema** 5.1. Estrategia de Pruebas 5.2. Resultados de las Pruebas y Manejo de Errores 5.3. Seguridad del Sistema
6. **Conclusión** 6.1. Síntesis de Logros 6.2. Trabajo Futuro
7. **Referencias**
8. **Anexos** 8.1. Anexo A: Script DDL Completo

1. Introducción

1.1. Planteamiento del Problema

La gestión de la información académica en cuanto a instituciones de educación superior es un proceso crítico que exige una alta precisión, la seguridad de sus datos e información, y la eficiencia. El uso de sistemas manuales o descentralizados para el registro de las calificaciones, la gestión de inscripciones, y el acceso a los expedientes de los estudiantes, resulta muchas veces en ineficiencias operativas, errores y falta de transparencia para los estudiantes (Pressman & Maxim, 2020). Los problemas que se presentan inciden sobre la carga de trabajo del personal administrativo y docente, y sobre la capacidad de los estudiantes para poder consultar su rendimiento en tiempo real.

1.2. Justificación y Alcance del Proyecto

La necesidad de solución planteada ha dado lugar a la creación del "Sistema de Gestión de Notas Universitarias", una robusta y autónoma aplicación de escritorio. En este sentido, el proyecto es una simulación para la fake Universidad Tecnológica de Inteligencia Artificial (UTIA), que proporciona solución integral para la centralización y automatización del área académica. La elección de Java como lenguaje, JavaFX como interfaz gráfica y SQLite como DataBase embebida, ha permitido que la aplicación sea multiplataforma y con un fácil modo de distribución, sin requerir dependencias externas o instaladores los cuales que incrementan la complejidad de este tipo de herramienta.

El alcance del proyecto cubre el ciclo de vida completo del desarrollo del software, abarcando desde su concepción y el diseño, hasta la implementación y prueba. Las funcionalidades son las de la gestión de usuarios, la estructura académica (carreras, materias), y un sistema de calificaciones.

1.3. Objetivos

Objetivo General: Diseñar, desarrollar e implementar una aplicación de escritorio operacional para la gestión de notas universitarias, aplicando los principios de la ingeniería del software y las mejores prácticas en el diseño de bases de datos y la programación orientada a objetos.

Objetivos Específicos:

1. Modelar y construir una base de datos relacional normalizada (3FN) para almacenar la información académica de manera segura e íntegra.
2. Implementar una arquitectura de software modular y ampliable basada en el patrón Modelo-Vista-Controlador (MVC).
3. Desarrollar una interfaz de usuario gráfica (GUI) intuitiva y funcional utilizando JavaFX.
4. Implementar un sistema de roles (Administrador, Profesor, Estudiante) con permisos y funcionalidades diferenciadas.

Asegurar la calidad del software mediante un plan de pruebas que valide la funcionalidad, el manejo de errores y la seguridad del sistema.

El análisis de requisitos identificó tres roles de usuario fundamentales. Sus necesidades de información se resumen en la siguiente tabla.

Tabla 1
Roles de Usuario y Requerimientos de Información

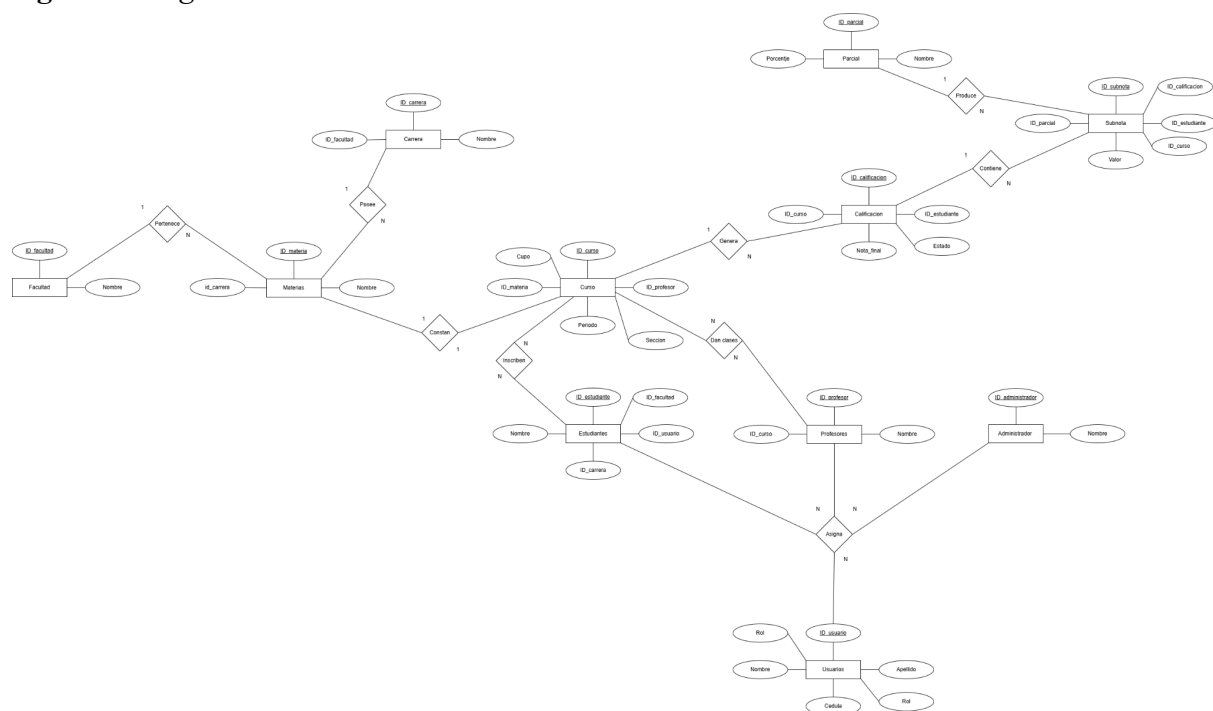
Rol de Usuario	Necesidades de Consulta	Necesidades de Manipulación (CRUD)
Administrador	Acceso sin restricciones a todos los datos del sistema (usuarios, carreras, materias, calificaciones).	Control total: Crear, leer, actualizar y eliminar todos los registros del sistema. Asignar roles.

Profesor Powered by Arizona State University	Visualizar cursos asignados, listas de estudiantes matriculados y sus respectivas calificaciones.	Registrar, actualizar y eliminar las sub-notas de los estudiantes en los cursos que imparte.
Estudiante	Consultar datos personales, materias inscritas y el detalle de sus calificaciones por parcial y sub-nota.	Ninguna. Es un rol de solo lectura para garantizar la integridad de sus calificaciones.

2.2. Diseño Conceptual y Lógico: Diagrama Entidad-Relación (ER)

El diseño de la base de datos es la base del sistema. Para ello se realizó un diagrama de entidad-relación (ER) en el que se representaron las entidades, sus atributos y las relaciones que existían entre ellas, asegurando así una representación fidedigna del dominio del problema.

Figura 1 Diagrama Entidad-Relación Final del Sistema de Notas Nota.



Nota. El diagrama muestra las relaciones uno a muchos y muchos a muchos que estructuran la base de datos académica. Fuente: Documentación del proyecto original.

2.3. Normalización de la Base de Datos

Se llevó a cabo un proceso de normalización hasta la 3ª Forma Normal (3FN). Esta técnica es fundamental a la hora de realizar el diseño de Bases de Datos relacionales debido a que se busca eliminar redundancias para mejorar la integridad de los datos (Elmasri & Navathe, 2017). En el caso del estudiante (nombre, cédula) se mantiene en la tabla Usuario y se referencian mediante una clave foránea (id_usuario) en la tabla Estudiante, permitiendo no repetir dichos datos en todos los registros de calificación. Esta forma evita las posibles anomalías de actualización y permite que el cambio de nombre de un usuario se vea reflejado siempre en el sistema.

3. Fase 2: Implementación de la Base de Datos

3.1. Estructura de Tablas y Script DDL

Basado en el diagrama ER, se escribió un script DDL (Data Definition Language) para crear la estructura en SQLite.

Tabla 2

Descripción de las Tablas Principales de la Base de Datos

Tabla	Propósito Principal	Clave Primaria	Claves Foráneas Notables
Usuario	Almacena datos comunes y credenciales de todos los usuarios.	id_usuario	-
Estudiante	Vincula un usuario al rol de estudiante y a una carrera.	id_estudiante	id_usuario, id_carrera
Carrera	Define las carreras académicas y su facultad.	id_carrera	id_facultad

Curso	Representa una instancia de una materia impartida por un profesor en un período.	id_curso	id_materia, id_profesor
Calificación	Almacena la nota de un estudiante en un curso específico.	id_calificacion	id_estudiante, id_curso
Subnota	Detalla las calificaciones parciales que componen la nota de una calificación.	id_subnota	id_calificacion, id_parcial

3.2. Población de la Base de Datos (Script DML)

Se elaboró el correspondiente script DML (Data Manipulation Language), que da cuenta de un grupo de 30 tuplas de datos con valor de prueba, el cual fue significativo para las fases de desarrollo y de prueba/validación.

3.3. Consultas de Prueba (Queries)

Se ejecutaron queries complejas para validar el modelo: en concreto, el ejemplo de la query siguiente valida la correcta obtención de todos los cursos que imparte un determinado profesor/a, enlazando 4 tablas.

```
-- Consulta para obtener los cursos de un profesor
SELECT M.nombre_materia, C.periodo, C.seccion, COUNT(CA.id_estudiante) AS
inscritos
FROM Curso C
JOIN Profesor P ON C.id_profesor = P.id_profesor
JOIN Usuario U ON P.id_usuario = U.id_usuario
JOIN Materia M ON C.id_materia = M.id_materia
LEFT JOIN Calificacion CA ON C.id_curso = CA.id_curso
WHERE U.cedula = '1753797065' -- Cédula de un profesor de prueba
GROUP BY M.nombre_materia, C.periodo, C.seccion;
```

4. Fase 3: Desarrollo de la Aplicación en Java

4.1. Arquitectura del Software y Patrones de Diseño

La calidad del sistema depende de una arquitectura correctamente definida y de patrones de diseño conocidos.

4.1.1. Patrón Modelo-Vista-Controlador (MVC)

El sistema está perfectamente estructurado siguiendo el patrón MVC, esto es, no hay confusión de responsabilidades (Sommerville, 2016).

Modelo: Clases como Usuario.java, que son simplemente POJOs (Plain Old Java Objects), no presentan lógica de negocio ni lógica de presentación.

Vista: Archivos FXML (Login.fxml, AdminView.fxml, etc.) que dan cuenta de la estructura de la UI de forma declarativa, permitiendo que diseñadores de interfaces trabajen de forma paralela a los desarrolladores.

Controlador: Clases como la clase LoginController.java, que son las encargadas de implementar la lógica y de las que se ejecuta el "cerebro" de cada vista, ya que aquí se incluye la lógica necesaria para llevar a cabo la gestión de los eventos de la UI (ej. @FXML private void handleLogin()) en interacción con la base de datos y posterior actualización de la vista. Por ejemplo el LoginController es quien determina la vista a cargar a continuación según el rol del usuario, por tanto, se trata de quien se encarga de gestionar la navegación.

4.1.2. Abstracción del Acceso a Datos (DAO)

De hecho, no se ha implementado el patrón de acceso a datos DAO en clases distintas, ya que se emplean directamente los controladores para la lógica de acceso a los datos. La decisión de utilizar solamente java.sql.PreparedStatement para las consultas SQL es una elección de diseño que previene la ejecución de ataques de inyección SQL, una de las vulnerabilidades de seguridad más frecuentes en aplicaciones que manipulan bases de datos (OWASP, 2021). La clase DatabaseConnection.java centraliza de manera común la forma de obtener la conexión, es una buena práctica de diseño que ayuda a poder gestionar la configuración de la base de datos.

4.1.3. Principios de Diseño SOLID

Única Responsabilidad: El LoginController es el único que hace la autenticación. El AdminController, solo hace las cosas administrativas.

Abierto/Cerrado: Se puede agregar un nuevo rol (p.ej, "Dirección de Carrera") y la vista correspondiente, sin tocar el LoginController, extendiendo solamente la estructura switch-case.

Sustitución de Liskov: Todos los controladores principales implementan la interfaz MainController para que cada uno pueda recibir un objeto Usuario polimórficamente tras el login realizado.

4.2. Diseño e Implementación de la Interfaz de Usuario (UI)

La interfaz de usuario fue concebida para que fuera natural y práctica. El proceso iterativo se realizó desde bocetos de baja fidelidad hasta las últimas versiones que se implementaron utilizando JavaFX y que se ha utilizando CSS para mostrar una misma identidad visual.

Figura 2 *Evolución del Diseño de la Interfaz de Usuario*



Nota. Se muestra el boceto inicial del diseño de interfaz de usuario limpia y organizada.

Figura 3 Evolución del Diseño de la Interfaz de Administrador



Nota. Se muestra el boceto inicial del diseño de interfaz de administrador de usuarios limpia y organizada.


Figura 4 Evolución del Diseño de la Interfaz de registro de calificaciones del profesor



Nota. Se muestra el boceto inicial del diseño de interfaz de profesor en el registro de calificaciones.

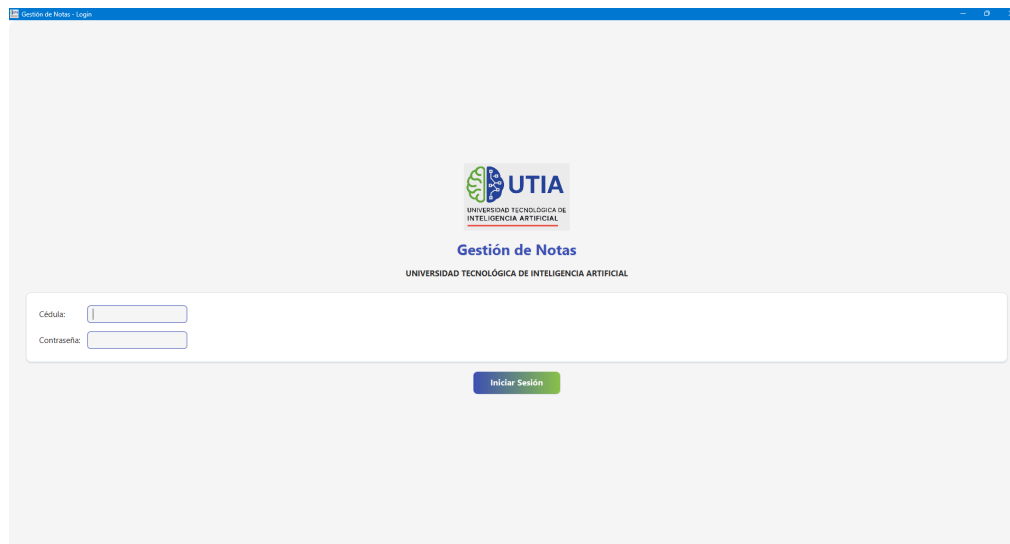
Figura 5 *Evolución del Diseño de la Interfaz de Calificaciones*



Calificaciones 	
Calificación	10/10
	10/10
	10/10
	10/10
	10/10
	10/10
	10/10
	10/10
	10/10
	10/10
Porcentaje	100%

Nota. Se muestra el boceto inicial del diseño de interfaz de calificaciones.

Figura 6 *Diseño de la Interfaz de Gestión de Calificaciones*



UTIA
UNIVERSIDAD TECNOLÓGICA DE
INTELIGENCIA ARTIFICIAL

Gestión de Notas
UNIVERSIDAD TECNOLÓGICA DE INTELIGENCIA ARTIFICIAL

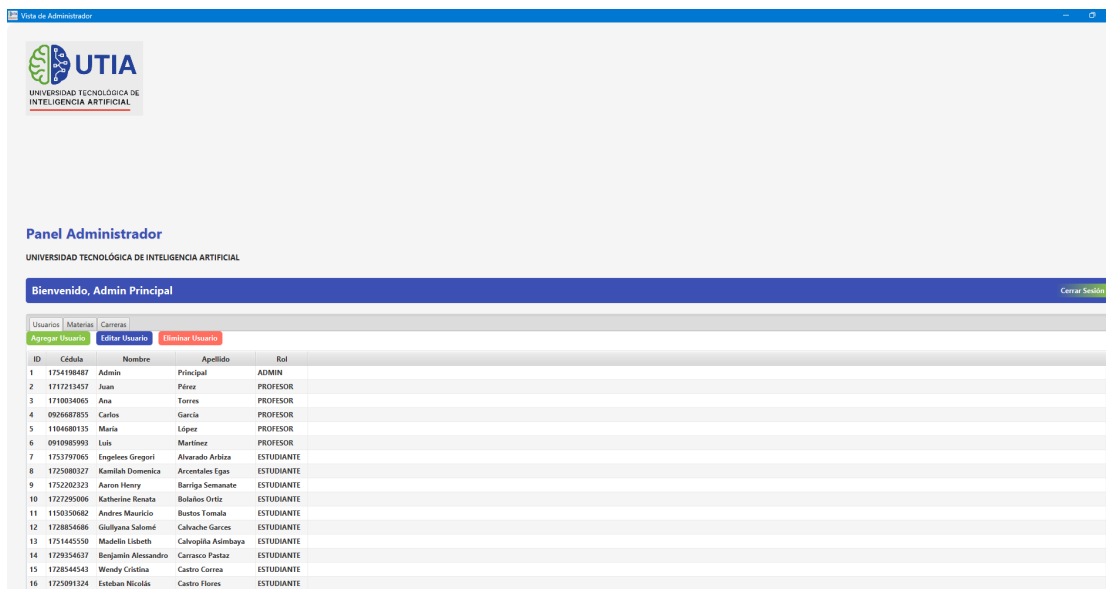
Cédula:

Contraseña:

[Iniciar Sesión](#)

Nota. Se muestra el diseño de interfaz de gestión de notas.

Figura 7 *Diseño de la Interfaz del Panel Administrador para usuarios*



Panel Administrador
UNIVERSIDAD TECNOLÓGICA DE INTELIGENCIA ARTIFICIAL

Bienvenido, Admin Principal

Usuarios | Materias | Carreras

Agregar Usuario | Editar Usuario | Eliminar Usuario

ID	Cédula	Nombre	Apellido	Rol
1	1754198487	Adrián	Principal	ADMIN
2	1717213457	Juan	Pérez	PROFESOR
3	1719814065	Ana	Torres	PROFESOR
4	9026687935	Carlos	García	PROFESOR
5	1104640135	Maria	López	PROFESOR
6	091985993	Luis	Martínez	PROFESOR
7	1753797065	Engelens Gregori	Alvarado Arbizu	ESTUDIANTE
8	1725080327	Kamilah Domencia	Arcentales Egas	ESTUDIANTE
9	1752202323	Aaron Henry	Barriga Semanate	ESTUDIANTE
10	1727295006	Katherine Remata	Bolados Ortiz	ESTUDIANTE
11	1150350682	Andres Mauricio	Bustos Tomala	ESTUDIANTE
12	1728854686	Guilhyana Solimé	Calache-Garcés	ESTUDIANTE
13	1751445530	Madelin Lideth	Calonilla Adimbaya	ESTUDIANTE
14	1729354637	Benjamin Alejandro	Carrasco Pastaz	ESTUDIANTE
15	1728545453	Wendy Cristina	Castro Correa	ESTUDIANTE
16	1725091324	Esteban Nicolás	Castro Flores	ESTUDIANTE

Nota. Se muestra el diseño de interfaz del panel administrador de todos los usuarios.

Figura 8 *Diseño de la Interfaz del Panel Administrador para materias*



Panel Administrador
UNIVERSIDAD TECNOLÓGICA DE INTELIGENCIA ARTIFICIAL

Bienvenido, Admin Principal

Usuarios | Materias | Carreras

Agregar Materia | Editar Materia | Eliminar Materia

ID▲	Nombre	Carrera
1	Programación I	Ingeniería en Sistemas
2	Bases de Datos	Ingeniería en Sistemas
3	Redes de Computadoras	Ingeniería en Sistemas
4	Sistemas Operativos	Ingeniería en Sistemas
5	Desarrollo Web	Ingeniería en Sistemas
6	Mecánica I	Ingeniería en Mecatrónica
7	Electrónica Digital	Ingeniería en Mecatrónica
8	Control Automático	Ingeniería en Mecatrónica
9	Robótica	Ingeniería en Mecatrónica
10	Programación para Mecatrónica	Ingeniería en Mecatrónica

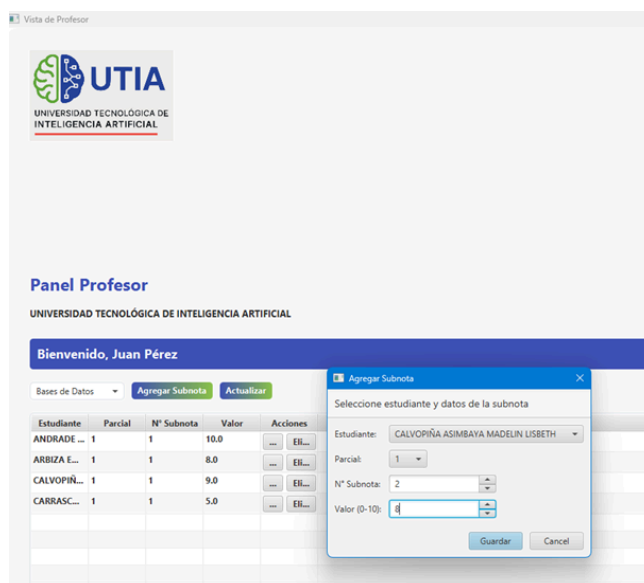
Nota. Se muestra el diseño de interfaz del panel de administrador para las materias asignadas.

Figura 9 *Diseño de la Interfaz del Panel Administrador para carreras*



Nota. Se muestra el diseño de interfaz del panel de administrador para las carreras asignadas.

Figura 10 *Diseño de la Interfaz del Panel Profesor*



4.3. Implementación de la Lógica de Negocio

Los controladores son los encargados de implementar la lógica de negocio. Como ejemplo, en el AdminController.java aparecen ya métodos complejos como mostrarDialogoUsuario(), que sólo construye una ventana de diálogo dinámica para la creación o actualización de usuarios, sino que también "llena" funcionalidades con la lógica de negocio para la lógica de validación de duplicados, la asignación de materias a estudiantes y profesores, y también la generación de contraseñas seguras. El ProfesorController.java también implementa la lógica para validar que la suma de subnotas de un parcial no exceda el ponderado máximo.

5. Pruebas y Validación del Sistema

5.1. Estrategia de Pruebas

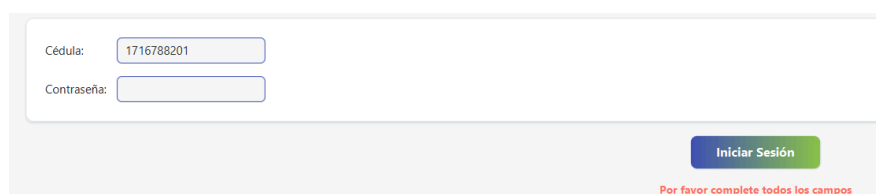
Se puso en práctica una estrategia de pruebas multinivel que asegurasen la calidad:

- Pruebas de Componente: Se comprobó el comportamiento de los métodos clave de forma individual, por ejemplo, el método de validación de cédula (validar cedula ecuatoriana).
- Pruebas de Integración: Se probó la correcta interacción de los componentes. Por ejemplo, si se clicaba el "Login", se llamaba al correcto controlador de acceso a la base de datos de forma correcta y se levantaba la vista adecuada.
- Pruebas del Sistema (Casos de Uso): Se probaron el ciclo completo de los flujos de trabajo completos, por ejemplo: "Crear nuevo estudiante, asignar materias, después como profesor, calificar al mismo".

5.2. Resultados de las Pruebas y Manejo de Errores

Las revisiones confirmaron que el sistema cumple con todos los requisitos funcionales. El manejo de errores se validó como robusto, se proporciona una clara retroalimentación al usuario cuando las entradas no son válidas, o cuando no son permitidas las acciones.

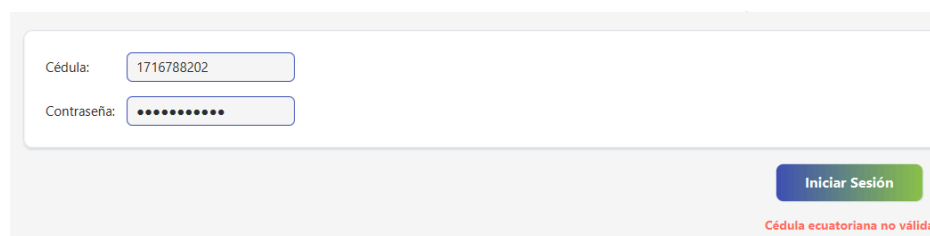
Figura 12 *Diálogo de Notificación de error por campos incompletos*



The screenshot shows a login form with two input fields: 'Cédula:' containing '1716788201' and 'Contraseña:' which is empty. A green 'Iniciar Sesión' button is at the bottom right. Below the button, a red error message reads 'Por favor complete todos los campos'.

Nota. Se muestra una notificación de error al no completar los campos pedidos.

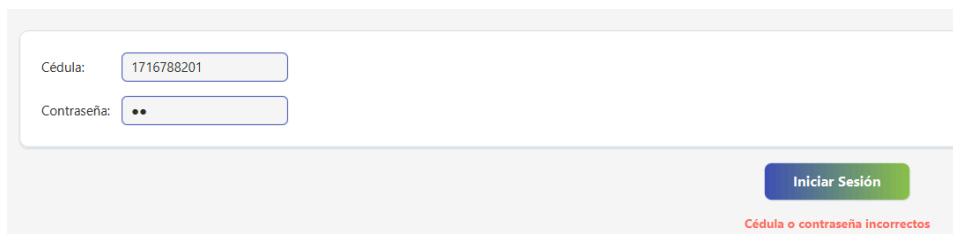
Figura 13 *Diálogo de Notificación de error por cédula inválida*



The screenshot shows the same login form, but the 'Cédula:' field now contains '1716788202'. The 'Contraseña:' field is masked with dots. The 'Iniciar Sesión' button is still present. Below the button, a red error message reads 'Cédula ecuatoriana no válida'.

Nota. Se muestra una notificación de error al no ingresar una cédula ecuatoriana válida.

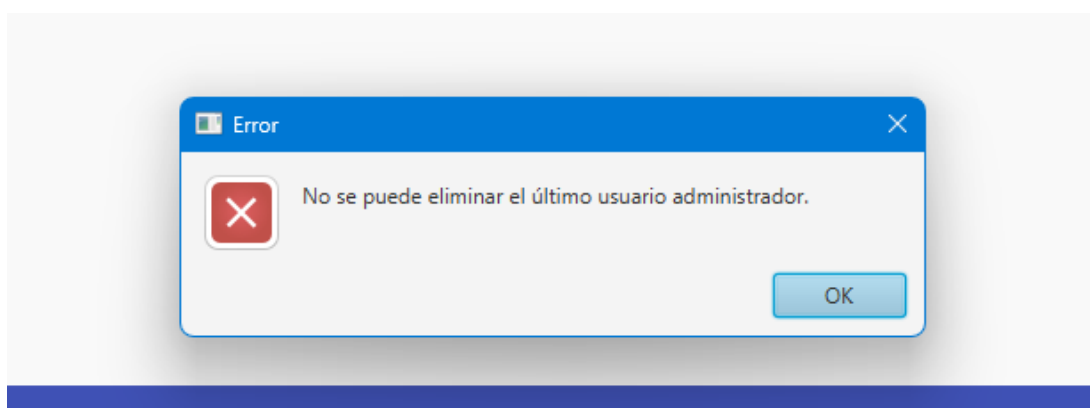
Figura 14 *Diálogo de Notificación de error por campos incorrectos*



A login form with two input fields: 'Cédula:' containing '1716788201' and 'Contraseña:' containing two dots. A green 'Iniciar Sesión' button is on the right. Below the button, the text 'Cédula o contraseña incorrectos' is displayed in red.

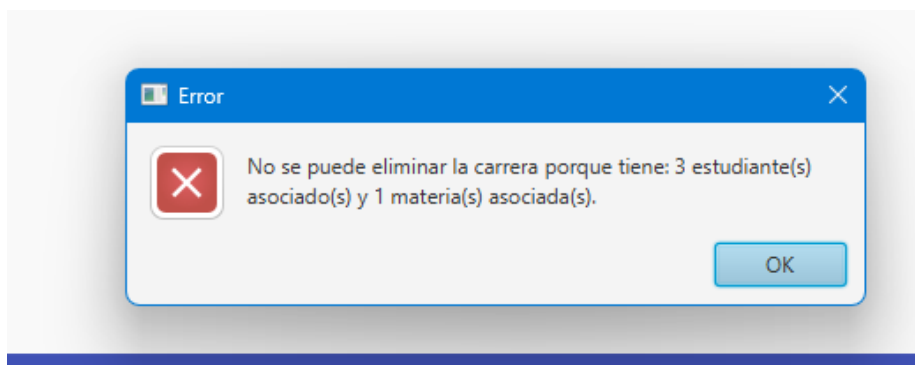
Nota. Se muestra una notificación de error al ingresar una cédula o contraseña incorrecta.

Figura 15 *Diálogo de Notificación de error por eliminación de usuario administrador*



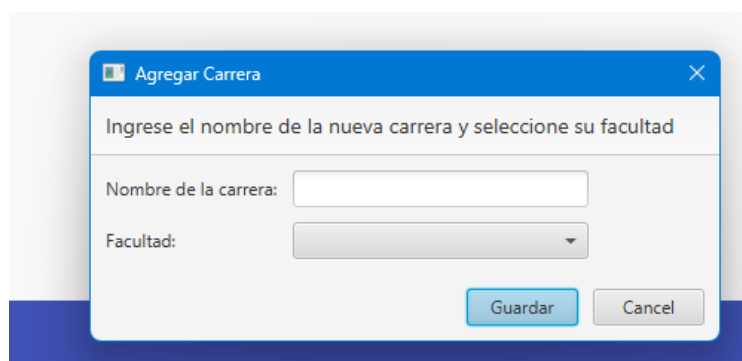
Nota. Error de validación por formato de cédula inválido. Error de integridad referencial que impide eliminar una carrera con estudiantes asociados.

Figura 16 *Diálogo de Notificación de error por intento de eliminación de carrera, aún con valores asociados a ella*



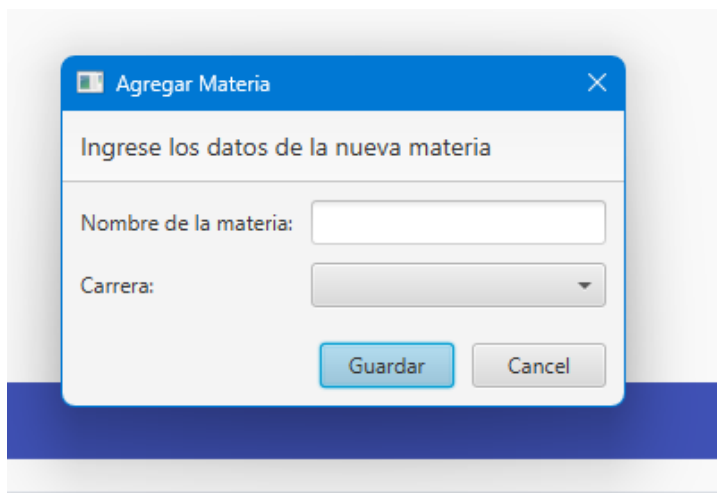
Nota. Error de validación por formato de cédula inválido. Error de integridad referencial que impide eliminar una carrera con estudiantes asociados.

Figura 17 *Diálogo de Notificación para añadir carrera y su facultad*



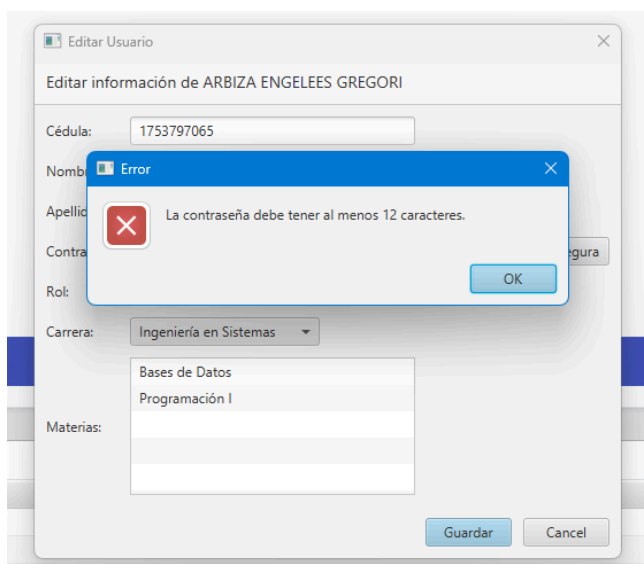
Nota. Este diálogo muestra que en esta interfaz se añade los campos de nombre de la carrera y la facultad a la que pertenece.

Figura 18 *Diálogo de Notificación para añadir materia y su carrera*



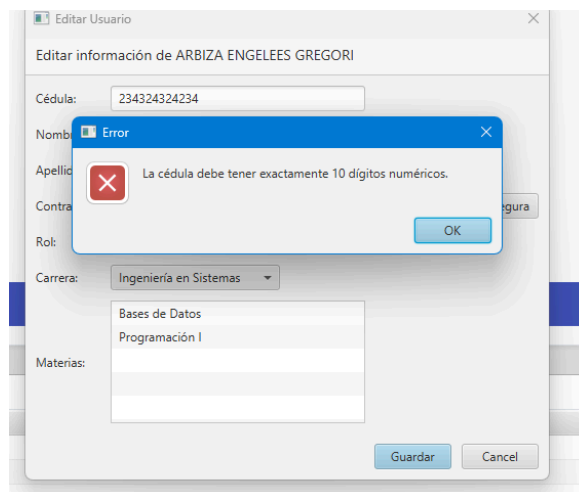
Nota. Este diálogo muestra que en esta interfaz se añade los campos de nombre de materia y carrera.

Figura 19 *Diálogo de Notificación de error por falta de número de caracteres ingresados para la contraseña*



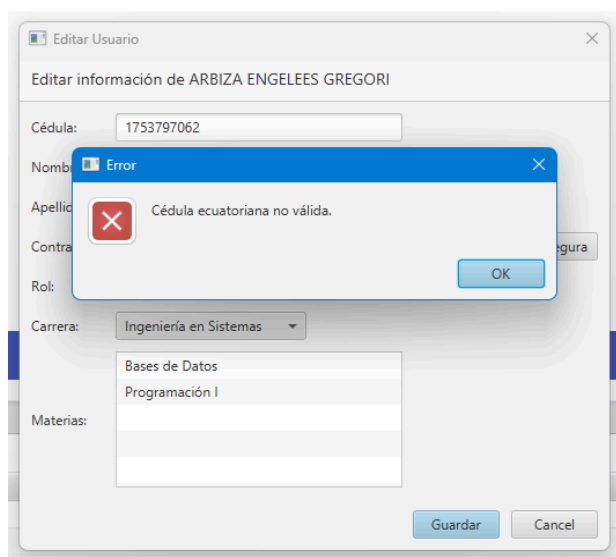
Nota. Error de validación del campo contraseña, este campo requiere de al menos doce caracteres para ser válido.

Figura 20 *Diálogo de Notificación de error por falta de dígitos numéricos ingresados para la cédula*



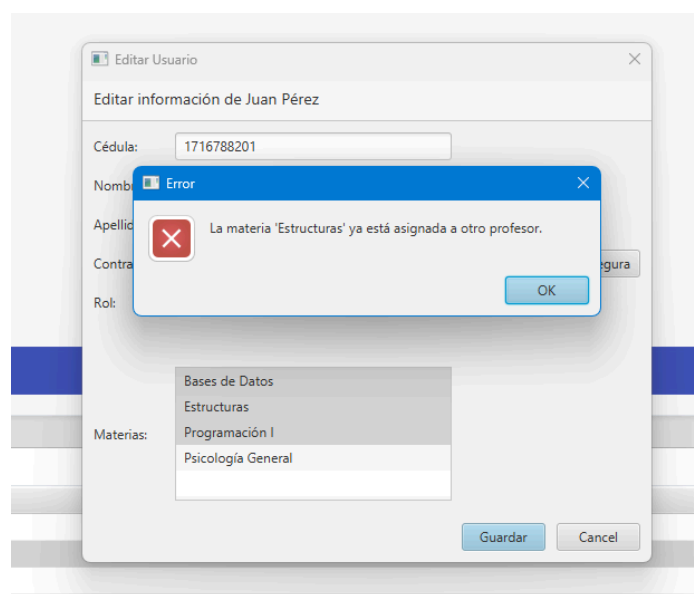
Nota. Error de validación de valores ingresos en el campo cedula, este campo sólo requiere un número de diez dígitos numéricos.

Figura 21 *Diálogo de Notificación de error por cédula ecuatoriana no válida*



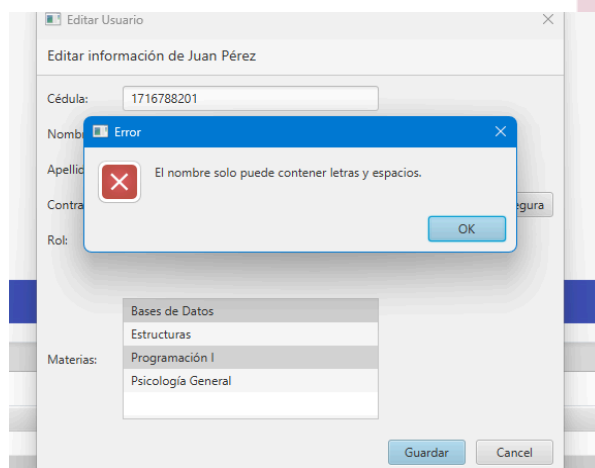
Nota. Error de validación por formato de cédula inválido. Error de integridad referencial que impide eliminar una carrera con estudiantes asociados.

Figura 22 *Diálogo de Notificación de error por la asignación de una materia ya asignada a otro profesor*



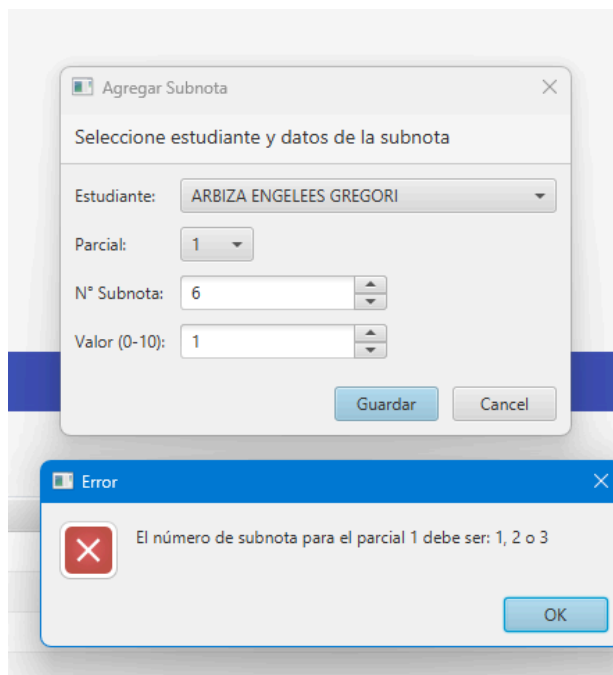
Nota. Error de asignación de una materia, que ya está designada a otro profesor en específico.

Figura 23 *Diálogo de Notificación de error por que el campo nombre solo permite letras y espacios*



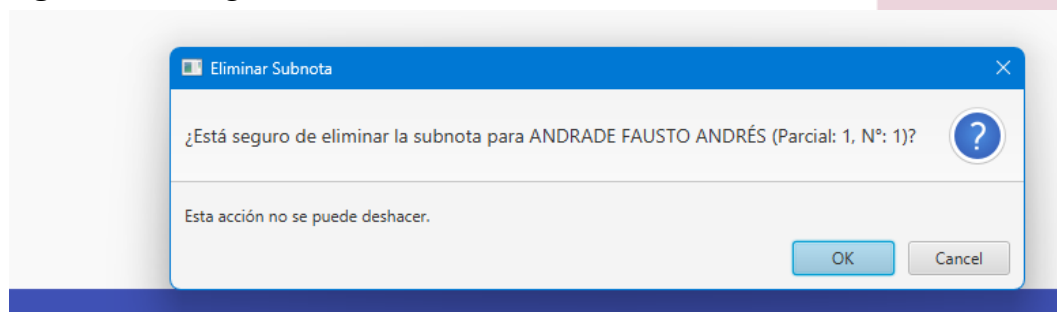
Nota. Error al ingresar valores que no son letras y espacios en el campo de nombre del usuario.

Figura 24 *Diálogo de Notificación de error en el campo de subnota ya que de ser 1, 2 o 3*



Nota. Error de validación por formato de cédula inválido. Error de integridad referencial que impide eliminar una carrera con estudiantes asociados.

Figura 25 *Diálogo de Advertencia al eliminar la subnota de un estudiante*



Nota. El diálogo muestra un mensaje de advertencia al ejecutar la acción de eliminación de la subnota de un estudiante, ya que esta será una acción que no se puede deshacer.

5.3. Seguridad del Sistema

La seguridad se trata de una forma global:

- Validación de Entradas: En el cliente (interfaz) y en la lógica del controlador para evitar datos corruptos.
- Acceso Seguro a Datos: Utilizar PreparedStatement es la forma principal de protección contra inyección SQL.
- Control de Acceso Basado en Roles (RBAC): El sistema verifica constantemente el rol del usuario en cada operación crítica de forma que ningún estudiante pueda hacer uso de una operación de un administrador.

6. Conclusión

6.1. Síntesis de Logros

El proyecto que lleva por nombre "Sistema de Gestión de Notas Universitarias" ha finalizado de una manera satisfactoria, entregando un software de escritorio completamente funcional y que satisface del todo los objetivos planteados. Se ha demostrado una sólida aplicación de los principios de la ingeniería de software desde el diseño de una base de datos normalizada, hasta la implementación de una arquitectura MVC limpia o la implementación de interfaces amigables para los usuarios. Este trabajo no sólo ha significado un gran éxito técnico, sino también una experiencia de aprendizaje de gran valor, ya que ha recopilado los saberes adquiridos en amplios ámbitos de la formación en una solución tangible del mismo modo que funcional.

6.2. Trabajo Futuro

El propósito de detallar estas mejoras es explorar y profundizar en los temas tratados, como parte fundamental de nuestro proceso de aprendizaje. No se trata de un listado de cambios que serán implementados

- La actual arquitectura constituye una buena base para implementar futuras ampliaciones y sobre la que se tienen que implementar mejoras que son las siguientes:
- Migrar a una arquitectura Web: Durante el curso del proyecto volveremos a definir la aplicación para que contenga un backend en el que se contemple una API RESTful para el backend y en el que se disponga del frontend web dentro del cual se integren frameworks como React o Angular de manera que permita que la aplicación sea usable por cualquier navegador.
- Módulo de reportes: Se tienen que implementar funcionalidades que permitan la exportación de reportes académicos con formato pdf, como por ejemplo la exportación de expedientes académicos por estudiante o también reportes estadísticos de rendimiento por curso.
- Notificaciones: Implementar un sistema de generación de notificaciones (ej: por correo) que avise a los estudiantes cuando las notas de las pruebas correspondientes a las actividades han sido publicadas.
- Auditoría de cambios: Incluir un sistema de logs que permita la disponibilidad de la información relativa a los cambios sustantivos en los datos (quién, qué y cuándo) para así tener una mejor trazabilidad y seguridad.

7. Referencias

Bitix. (2021, 31 de marzo). *Los conceptos de encapsulación, herencia, polimorfismo y composición de la programación orientada a objetos*. Blog Bitix.
<https://picodotdev.github.io/blog-bitix/2021/03/los-conceptos-de-encapsulacion-herencia-polimorfismo-y-composicion-de-la-programacion-orientada-a-objetos/>

Blancarte, O. (2018, 10 de diciembre). *Data Access Object (DAO) Pattern*. Oscar Blancarte - Software Architecture.

<https://www.oscarblancarte.com/2018/12/10/data-access-object-dao-pattern/>

Datacamp. (s.f.). *Database Normalization Explained: 1NF, 2NF, 3NF, BCNF, 4NF & 5NF*.

Recuperado el 9 de junio de 2025, de

<https://www.datacamp.com/tutorial/normalization-in-sql>

Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems* (7th ed.). Pearson.

García, L., Pérez, M., y Rodríguez, J. (2021). *Sistemas de Información para la Gestión Educativa en la Era Digital*. Editorial Tecnológica.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.

Modelo-vista-controlador. (2023, 15 de noviembre). En *Wikipedia*.

<https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>

OWASP Foundation. (2021). *OWASP Top Ten*. <https://owasp.org/www-project-top-ten/>

Peña, E. (2020, 2 de julio). *Principios esenciales para desarrollar código de calidad*. CodeYourApps.

<https://codeyourapps.com/solid-5-principios-esenciales-para-desarrollar-de-calidad/>

Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.

Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.

8. Anexos

8.1. Anexo A: Script DDL Completo

-- DDL (Data Definition Language) - Creación de Tablas Completas

-- Eliminar tablas si existen para empezar desde cero

DROP TABLE IF EXISTS "Subnota";

DROP TABLE IF EXISTS "Calificacion";


```
DROP TABLE IF EXISTS "Curso";  
DROP TABLE IF EXISTS "Parcial";  
DROP TABLE IF EXISTS "Estudiante";  
DROP TABLE IF EXISTS "Profesor";  
DROP TABLE IF EXISTS "Administrador";  
DROP TABLE IF EXISTS "Usuario";  
DROP TABLE IF EXISTS "Materia";  
DROP TABLE IF EXISTS "Carrera";  
DROP TABLE IF EXISTS "Facultad";
```

-- Tabla: Facultad

```
CREATE TABLE "Facultad" (  
    "id_facultad"    INTEGER,  
    "nombre_facultad" TEXT NOT NULL UNIQUE,  
  
    PRIMARY KEY("id_facultad" AUTOINCREMENT)  
);
```

-- Tabla: Carrera

```
CREATE TABLE "Carrera" (  
    "id_carrera"    INTEGER,  
    "nombre_carrera" TEXT NOT NULL UNIQUE,  
    "id_facultad"    INTEGER NOT NULL,  
    PRIMARY KEY("id_carrera" AUTOINCREMENT),  
    FOREIGN KEY("id_facultad") REFERENCES "Facultad"("id_facultad")  
);
```

-- Tabla: Usuario

```
CREATE TABLE "Usuario" (  
    "id_usuario"    INTEGER,  
    "cedula"        TEXT NOT NULL UNIQUE,  
    "nombre_usuario" TEXT NOT NULL,  
    "apellido_usuario" TEXT NOT NULL,  
    "password"      TEXT NOT NULL,  
    "rol"           TEXT NOT NULL CHECK("rol" IN ('ESTUDIANTE',  
'PROFESOR', 'ADMIN')),  
    PRIMARY KEY("id_usuario" AUTOINCREMENT)  
);
```

-- Tabla: Administrador

```
CREATE TABLE "Administrador" (  
    "id_administrador" INTEGER,
```



UIDE

Powered by
Arizona State University

```
"id_usuario"    INTEGER NOT NULL UNIQUE,  
PRIMARY KEY("id_administrador" AUTOINCREMENT),  
FOREIGN KEY("id_usuario") REFERENCES "Usuario"("id_usuario")  
);
```

-- Tabla: Profesor

```
CREATE TABLE "Profesor" (  
    "id_profesor"    INTEGER,  
    "id_usuario"     INTEGER NOT NULL UNIQUE,  
    PRIMARY KEY("id_profesor" AUTOINCREMENT),  
    FOREIGN KEY("id_usuario") REFERENCES "Usuario"("id_usuario")  
);
```

-- Tabla: Estudiante

```
CREATE TABLE "Estudiante" (  
    "id_estudiante"  INTEGER,  
    "id_usuario"     INTEGER NOT NULL UNIQUE,  
    "id_carrera"     INTEGER NOT NULL,  
    PRIMARY KEY("id_estudiante" AUTOINCREMENT),  
    FOREIGN KEY("id_carrera") REFERENCES "Carrera"("id_carrera"),  
    FOREIGN KEY("id_usuario") REFERENCES "Usuario"("id_usuario")  
);
```

-- Tabla: Materia

```
CREATE TABLE "Materia" (  
    "id_materia"     INTEGER,  
    "nombre_materia" TEXT NOT NULL,  
    "id_carrera"     INTEGER NOT NULL,  
    PRIMARY KEY("id_materia" AUTOINCREMENT),  
    FOREIGN KEY("id_carrera") REFERENCES "Carrera"("id_carrera")  
);
```

-- Tabla: Curso

```
CREATE TABLE "Curso" (  
    "id_curso"       INTEGER,  
    "id_materia"     INTEGER NOT NULL,  
    "id_profesor"    INTEGER NOT NULL,  
    "periodo"        TEXT NOT NULL,  
    "seccion"        TEXT NOT NULL,  
    "cupo"           INTEGER NOT NULL,  
    PRIMARY KEY("id_curso" AUTOINCREMENT),
```

```
FOREIGN KEY("id_materia") REFERENCES "Materia"("id_materia"),
FOREIGN KEY("id_profesor") REFERENCES "Profesor"("id_profesor")
);

-- Tabla: Calificacion
CREATE TABLE "Calificacion" (
    "id_calificacion" INTEGER,
    "id_estudiante" INTEGER NOT NULL,
    "id_curso" INTEGER NOT NULL,
    "nota_final" DECIMAL(4, 2),
    "estado" TEXT NOT NULL DEFAULT 'NO_CALIFICADO'
CHECK("estado" IN ('NO_CALIFICADO', 'CALIFICADO')),
    PRIMARY KEY("id_calificacion" AUTOINCREMENT),
    UNIQUE("id_estudiante", "id_curso"),

FOREIGN KEY("id_curso") REFERENCES "Curso"("id_curso"),
    FOREIGN KEY("id_estudiante") REFERENCES "Estudiante"("id_estudiante")
);

-- Tabla: Parcial
CREATE TABLE "Parcial" (
    "id_parcial" INTEGER,
    "nombre" TEXT NOT NULL,
    "porcentaje" INTEGER NOT NULL,
    PRIMARY KEY("id_parcial" AUTOINCREMENT)
);

-- Tabla: Subnota
CREATE TABLE "Subnota" (
    "id_subnota" INTEGER,
    "id_calificacion" INTEGER NOT NULL,
    "id_parcial" INTEGER NOT NULL,
    "numero_notas" INTEGER NOT NULL,
    "valor" NUMERIC,
    PRIMARY KEY("id_subnota" AUTOINCREMENT),
    FOREIGN KEY("id_calificacion") REFERENCES
"Calificacion"("id_calificacion"),
    FOREIGN KEY("id_parcial") REFERENCES "Parcial"("id_parcial")
);
```