**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS**



**ESCUELA DE INGENIERÍA DE SISTEMAS**

**Curso:**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Proyecto:**

**Sistema de Gestión Escolar**

**Autores:**

Canales Lujan, Adolfo Alexander ([orcid.org/0009-0008-5920-0011](http://orcid.org/0009-0008-5920-0011))

Diaz Ordoñez, Alexander ([orcid.org/0009-0003-4416-9633](http://orcid.org/0009-0003-4416-9633))

Martinez Arista, Gabriel Gibson ([orcid.org/0009-0005-6054-5701](https://orcid.org/0009-0005-6054-5701))

Nuñez Principe, Luis Rodrigo ([orcid.org/0009-0007-4133-7733](http://orcid.org/0009-0007-4133-7733))

Pacotaipe Huacles, Daniel Elias ([orcid.org/0009-0003-0158-5688](http://orcid.org/0009-0003-0158-5688))

**Docente:**

Mag. Coronel Castillo, Eric Gustavo ([orcid.org/0000-0003-0494-5629](http://orcid.org/0000-0003-0494-5629))

**LIMA – PERÚ**

**2024**

# DEDICATORIA

A nuestro querido profesor,

Este proyecto, más allá de ser un simple conjunto de líneas de código, es el reflejo de todo lo que hemos aprendido bajo su guía. Cada decisión tomada, cada algoritmo implementado y cada validación diseñada ha sido posible gracias a la base sólida que nos ha proporcionado. En un mundo que avanza a la velocidad de la inteligencia artificial, donde la tecnología redefine constantemente lo posible, usted ha sido nuestro mentor, enseñándonos no solo el "cómo", sino también el "por qué" de la programación.  
“DIVIDE Y VENCERÁS”

# ÍNDICE GENERAL

Pág.

[**DEDICATORIA 3**](#_heading=h.gjdgxs)

[**ÍNDICE GENERAL 4**](#_heading=h.30j0zll)

[**RESUMEN 6**](#_heading=h.1fob9te)

[**ABSTRACT 7**](#_heading=h.3znysh7)

[**INTRODUCCIÓN 8**](#_heading=h.2et92p0)

[**OBJETIVOS 9**](#_heading=h.tyjcwt)

[OBJETIVO GENERAL 9](#_heading=h.3dy6vkm)

[OBJETIVOS ESPECÍFICOS 9](#_heading=h.1t3h5sf)

[**REQUERIMIENTOS 10**](#_heading=h.2s8eyo1)

[REQUERIMIENTOS FUNCIONALES 10](#_heading=h.17dp8vu)

[REQUERIMIENTOS NO FUNCIONALES 11](#_heading=h.3rdcrjn)

[ALCANCES 11](#_heading=h.26in1rg)

[LÍMITES 13](#_heading=h.lnxbz9)

[**ANÁLISIS DE REQUERIMIENTOS 15**](#_heading=h.1ksv4uv)

[IDENTIFICCIÓN DE ACTORES 15](#_heading=h.44sinio)

[CASOS DE USOS 15](#_heading=h.2jxsxqh)

[DIAGRAMA DE CASOS DE USO 16](#_heading=h.z337ya)

[**CU01 – MATRÍCULA 17**](#_heading=h.3j2qqm3)

[ESPECIFICACIÓN DEL CASO DE USO 17](#_heading=h.1y810tw)

[ENFOQUE DE LA CAJA NEGRA 18](#_heading=h.4i7ojhp)

[DIAGRAMA DE SECUENCIA 19](#_heading=h.2xcytpi)

[MODELO DE BASE DE DATOS 20](#_heading=h.1ci93xb)

[CRITERIOS DE ACEPTACIÓN 20](#_heading=h.3whwml4)

[**CU02 – REGISTRAR PAGO DE CUOTA 23**](#_heading=h.2bn6wsx)

[ESPECIFICACIÓN DEL CASO DE USO 23](#_heading=h.qsh70q)

[CAJA NEGRA 24](#_heading=h.3as4poj)

[DIAGRAMA DE SECUENCIA 25](#_heading=h.1pxezwc)

[MODELO DE BASE DE DATOS 26](#_heading=h.49x2ik5)

[CRITERIOS DE ACEPTACIÓN 26](#_heading=h.2p2csry)

[**CU03 – CONSULTAS 29**](#_heading=h.147n2zr)

[ESPECIFICACIÓN DEL CASO DE USO 29](#_heading=h.3o7alnk)

[CAJA NEGRA 31](#_heading=h.23ckvvd)

[MODELO DE BASE DE DATOS 31](#_heading=h.32hioqz)

[CRITERIOS DE ACEPTACIÓN 32](#_heading=h.1hmsyys)

[**MODELO DE BASE DE DATOS 35**](#_heading=h.4k668n3)

[MODELO RELACIONAL 35](#_heading=h.1egqt2p)

[DICCIONARIO DE DATOS 35](#_heading=h.3ygebqi)

[**IMPLEMENTACIÓN DE LA BASE DE DATOS 40**](#_heading=h.2dlolyb)

[CREACIÓN DE LA BASE DE DATOS 40](#_heading=h.sqyw64)

[CREACIÓN DE LOS OBJETOS DE LA BASE DE DATOS 40](#_heading=h.3cqmetx)

[CARGAR DATOS DE PRUEBA 45](#_heading=h.1rvwp1q)

[**CÓDIGO FUENTE 61**](#_heading=h.4bvk7pj)

[SERVICIO DE MATRICULA\_CU01 61](#_heading=h.jspm9ylpi476)

[SERVICIO DE PAGOS\_CU02 69](#_heading=h.8vfq1okb38tm)

[SERVICIO DE CONSULTAS\_CU03 75](#_heading=h.jjl0kc5hb6my)

[**CONCLUSIONES 81**](#_heading=h.2r0uhxc)

[**LECCIONES APRENDIDAS 83**](#_heading=h.1664s55)

[**BIBLIOGRAFÍA 84**](#_heading=h.3q5sasy)

[**ANEXOS 85**](#_heading=h.25b2l0r)

[Anexo 1: Titulo 86](#_heading=h.kgcv8k)

[Anexo 2: Titulo 87](#_heading=h.34g0dwd)

# RESUMEN

Este documento describe el desarrollo del backend para un sistema de gestión escolar dirigido a colegios de primaria. El sistema, implementado utilizando Spring Boot y SQL Server, está diseñado para gestionar tres áreas clave: la matrícula estudiantil, los pagos de los estudiantes y la asignación de profesores, con funcionalidades específicas como la visualización de horarios de los docentes. A través de la integración de servicios backend, se garantiza una gestión eficiente y precisa de los registros, validaciones y operaciones asociadas a cada uno de estos procesos. Además, se implementaron controles y validaciones a nivel de base de datos para asegurar la integridad y coherencia de la información, incluyendo la verificación de pagos, matrículas y asignación de profesores a los cursos. El sistema permite a los administradores gestionar los pagos de matrícula y visualizar horarios de los profesores de manera sencilla y confiable.

**Palabras clave:** Spring Boot, SQL Server, Desarrollo Backend, Gestión Escolar Primaria, Matrícula Estudiantil, Procesamiento de Pagos, Asignación de Profesores, Visualización de Horarios, Validación de Base de Datos, Sistema Educativo.

# ABSTRACT

This document describes the development of the backend for a school management system aimed at primary schools. The system, implemented using Spring Boot and SQL Server, is designed to manage three key areas: student enrollment, payments, and teacher assignments, with specific functionalities such as visualization of teacher schedules. Through the integration of backend services, efficient and accurate management of records, validations, and operations associated with each of these processes is ensured. Additionally, database-level controls and validations were implemented to ensure the integrity and consistency of the information, including payment, enrollment, and teacher course assignment verifications. The system allows administrators to manage tuition payments and view teacher schedules in a simple and reliable way.

**Keywords:** Spring Boot, SQL Server, Backend Development, Primary School Management, Student Enrollment, Payment Processing, Teacher Assignment, Schedule Visualization, Database Validation, Educational System.

# INTRODUCCIÓN

El proyecto desarrollado tiene como objetivo la creación de un sistema para gestionar diferentes procesos administrativos en un colegio de primaria. Este sistema está orientado a la automatización y optimización de tareas relacionadas con la matrícula de los estudiantes, la gestión de pagos de cuotas escolares, y la asignación de horarios y cursos a los profesores.

La implementación del sistema se realizó utilizando Spring Boot como framework principal para el desarrollo de los servicios backend y SQL Server como sistema de gestión de bases de datos. La arquitectura del sistema permite una interacción fluida con el usuario a través de diversas funcionalidades que garantizan la correcta gestión de datos y la eficiencia en los procesos educativos y administrativos del colegio.

Dentro de las funcionalidades más destacadas del sistema, se incluyen la validación de pagos mediante el registro de fechas y cuotas correspondientes, la asignación de profesores a cursos y secciones específicas, y la visualización de horarios de clases de acuerdo con la asignación realizada. Asimismo, se incorpora un robusto sistema de validaciones que asegura la integridad de los datos ingresados y permite el seguimiento adecuado de cada uno de los procesos involucrados.

Este documento detalla las principales características del backend desarrollado, los requerimientos tanto funcionales como no funcionales, y la implementación de cada uno de los casos de uso más relevantes. Además, se incluye una descripción de la base de datos utilizada, la creación y gestión de los objetos relacionados con el sistema, y los criterios de aceptación que garantizan la correcta implementación del mismo.

# OBJETIVOS

## OBJETIVO GENERAL

Desarrollar un sistema de gestión para un colegio de primaria que permita la administración de matrículas, pagos de cuotas, asignación de profesores y la visualización de horarios, asegurando una integración eficiente de los procesos de pago y asignación en tiempo real.

## OBJETIVOS ESPECÍFICOS

* Implementar un sistema que permita la gestión de las matrículas de los estudiantes y asignarle un cronograma de pagos correspondiente.
* Desarrollar un módulo para el registro y validación de pagos de cuotas, que permita gestionar los pagos en función del cronograma establecido. Este módulo incluirá la asignación automática de un monto adicional por mora en caso de retrasos, además de ofrecer la opción de liquidar anticipadamente el total correspondiente a las 10 cuotas.
* Establecer un sistema de autenticación y autorización restringido exclusivamente al personal administrativo y los profesores, garantizando el acceso seguro a las funcionalidades del sistema.
* Desarrollar un sistema robusto que valide todos los datos ingresados en el sistema mediante consultas en la base de datos, asegurando la coherencia y consistencia de la información relacionada con los pagos, matrículas y asignaciones.

# REQUERIMIENTOS

## REQUERIMIENTOS FUNCIONALES

**Gestión de Matrículas**

* El sistema debe permitir registrar nuevos estudiantes y asignarlos a una sección específica, respetando el número máximo de vacantes establecido para cada sección.
* Generar automáticamente un cronograma de pagos para cada estudiante al matricularse.

**Gestión de Pagos**

* Validar los pagos realizados por los estudiantes en función del cronograma de pagos preestablecido.
* Registrar el estado de los pagos (pendientes, completados o atrasados) y calcular automáticamente recargos por mora en caso de retrasos.
* Permitir el registro de pagos anticipados que liquiden todas las cuotas correspondientes de manera anticipada.

**Visualización de Horarios**

* Proveer información detallada sobre los horarios de cada profesor y sección, incluyendo: nombre del curso, horario de inicio y fin, y días de la semana asignados.
* Mostrar el horario programado en tiempo real basado en la fecha actual.

**Restricciones y Acceso**

* El sistema será accesible únicamente por el personal administrativo y los profesores del colegio, mediante un sistema de autenticación que requiere usuario y contraseña.
* Los empleados del personal administrativo tendrán permisos para gestionar las matrículas, registrar y validar los pagos, así como consultar la información general del sistema relacionada con las operaciones asignadas.
* Los profesores sólo podrán acceder a su horario y consultar información sobre los cursos que tienen asignados. No tendrán acceso a datos sensibles, como los detalles de matrículas o pagos de los estudiantes.

## REQUERIMIENTOS NO FUNCIONALES

**Escalabilidad**

* El sistema debe ser capaz de ampliar su funcionalidad para incluir niveles educativos adicionales, como secundaria, sin afectar su rendimiento actual.
* Debe permitir la gestión de beneficios por pago anticipado, así como la validación y control de puntualidad del personal administrativo y docente.
* El sistema debe admitir nuevas funcionalidades, como el traslado de estudiantes entre secciones internas o entre colegios externos, y el registro detallado de calificaciones por bimestre.

**Usabilidad para integración con frontend**

* El backend debe devolver todas las respuestas en formato JSON, con mensajes claros para facilitar la integración con el futuro desarrollo del frontend.
* Los mensajes de error deben incluir descripciones claras para cada validación fallida, como fallos en la autenticación o datos inválidos en las operaciones.

## ALCANCES

**Usuarios principales**

* El sistema está diseñado principalmente para el personal administrativo, que tendrá acceso a las funcionalidades de gestión de matrículas y pagos.
* Los profesores podrán consultar horarios y cursos asignados, pero no tendrán acceso a la gestión de matrículas ni pagos.

**Gestión de estudiantes**

* El sistema permitirá la matrícula de estudiantes que ya existan en la base de datos. Por el momento, no se contempla la funcionalidad para registrar nuevos estudiantes directamente desde el backend.

**Gestión de pagos**

* El backend será responsable de toda la lógica relacionada con los pagos, incluyendo la validación de pagos realizados, el cálculo de recargos por mora, y el registro de pagos anticipados.
* Actualmente, no se consideran descuentos por pagos anticipados, pero esta funcionalidad podría implementarse en futuras versiones.

**Visualización de horarios**

* El sistema permitirá a los profesores consultar horarios programados, mostrando información como el nombre del curso, horario de inicio y fin, y días de la semana asignados.
* No se permitirá la modificación de horarios desde el backend; esta funcionalidad será exclusiva del personal administrativo desde un sistema frontend.

**Niveles educativos**

* Aunque el sistema está diseñado inicialmente para el nivel de primaria, es adaptable para incluir la gestión del nivel de secundaria en futuras iteraciones.

**Restricciones de acceso**

* Los profesores sólo podrán consultar información sobre sus horarios y cursos asignados.
* El personal administrativo tendrá acceso exclusivo para gestionar matrículas, cronogramas de pagos, y validar los pagos realizados.

**Planes a futuro**

* Aunque no está implementado en la versión actual, se planea incluir módulos adicionales para el control de asistencia, el registro de notas, y el traslado interno o externo de estudiantes.

## LÍMITES

**Gestión de estudiantes**

* El sistema no permite el registro de nuevos estudiantes directamente desde el backend. Solo se podrán matricular estudiantes que ya existan previamente en la base de datos.

**Gestión de horarios**

* No se incluye la funcionalidad para modificar o asignar horarios desde el backend. Solo se permite la consulta de horarios programados, y cualquier ajuste deberá realizarse en un sistema frontend específico o en otra herramienta administrativa.

**Gestión de pagos**

* Actualmente, no se consideran funcionalidades para aplicar descuentos por pagos anticipados. El sistema solo manejará los pagos según el cronograma establecido y calculará recargos por mora en caso de retrasos.

**Restricciones de usuarios**

* Los profesores tienen acceso limitado a consultar sus horarios y cursos asignados, sin acceso a funcionalidades de gestión de matrículas o pagos.
* El personal administrativo no podrá realizar acciones relacionadas con el control de asistencia o el registro de notas desde este sistema.

**Alcance de niveles educativos**

* El sistema está diseñado inicialmente para gestionar estudiantes y operaciones del nivel de primaria. Aunque es adaptable para incluir secundaria, no está implementado en la versión actual.

**Coordinación con otros módulos**

* El sistema no incluye funcionalidades relacionadas con el control de asistencia, el registro de notas bimestrales, o la generación de reportes avanzados. Estas áreas quedan fuera del alcance de la versión actual.

**Generación de reportes**

* No se implementarán herramientas de generación de reportes en esta versión del sistema backend. Los reportes serán manejados en futuros desarrollos o mediante herramientas externas.

**Interacciones externas**

* No se contempla la integración con APIs de terceros ni módulos externos en esta etapa. Todo el sistema está diseñado para funcionar de manera independiente dentro del contexto de un único colegio.

# ANÁLISIS DE REQUERIMIENTOS

## IDENTIFICCIÓN DE ACTORES

|  |  |  |
| --- | --- | --- |
| **CÓDIGO** | **NOMBRE** | **DESCRIPCIÓN** |
| **ACT01** | Profesor | Representa al docente que utiliza el sistema para consultar su horario programado, incluyendo los cursos asignados, horarios de inicio y fin, y días. |
| **ACT02** | Alumno | Representa al estudiante registrado en la base de datos del colegio, quien es gestionado por el sistema a través de matrículas y cronogramas de pago. |
| **ACT03** | Personal Administrativo  (Empleado) | Representa al empleado encargado de gestionar matrículas, registrar pagos, validar pagos según cronogramas establecidos y administrar los datos del sistema. |

## CASOS DE USOS

|  |  |  |  |
| --- | --- | --- | --- |
| **CÓDIGO** | **ACTOR** | **NOMBRE** | **DESCRIPCIÓN** |
| **CU01** | Personal Administrativo | Registrar Matrícula | Permite registrar a un alumno en una sección y genera automáticamente su cronograma de pagos. |
| **CU02** | Personal Administrativo | Registrar Pago | Valida y registra el pago de una cuota correspondiente, incluyendo recargos por mora si aplica. |
| **CU03** | Profesor, Alumno | Consultar Horarios | Horarios Permite a los profesores visualizar sus horarios programados y a los alumnos consultar el horario de su sección. |

## 

## DIAGRAMA DE CASOS DE USO

# CU01 – MATRÍCULA

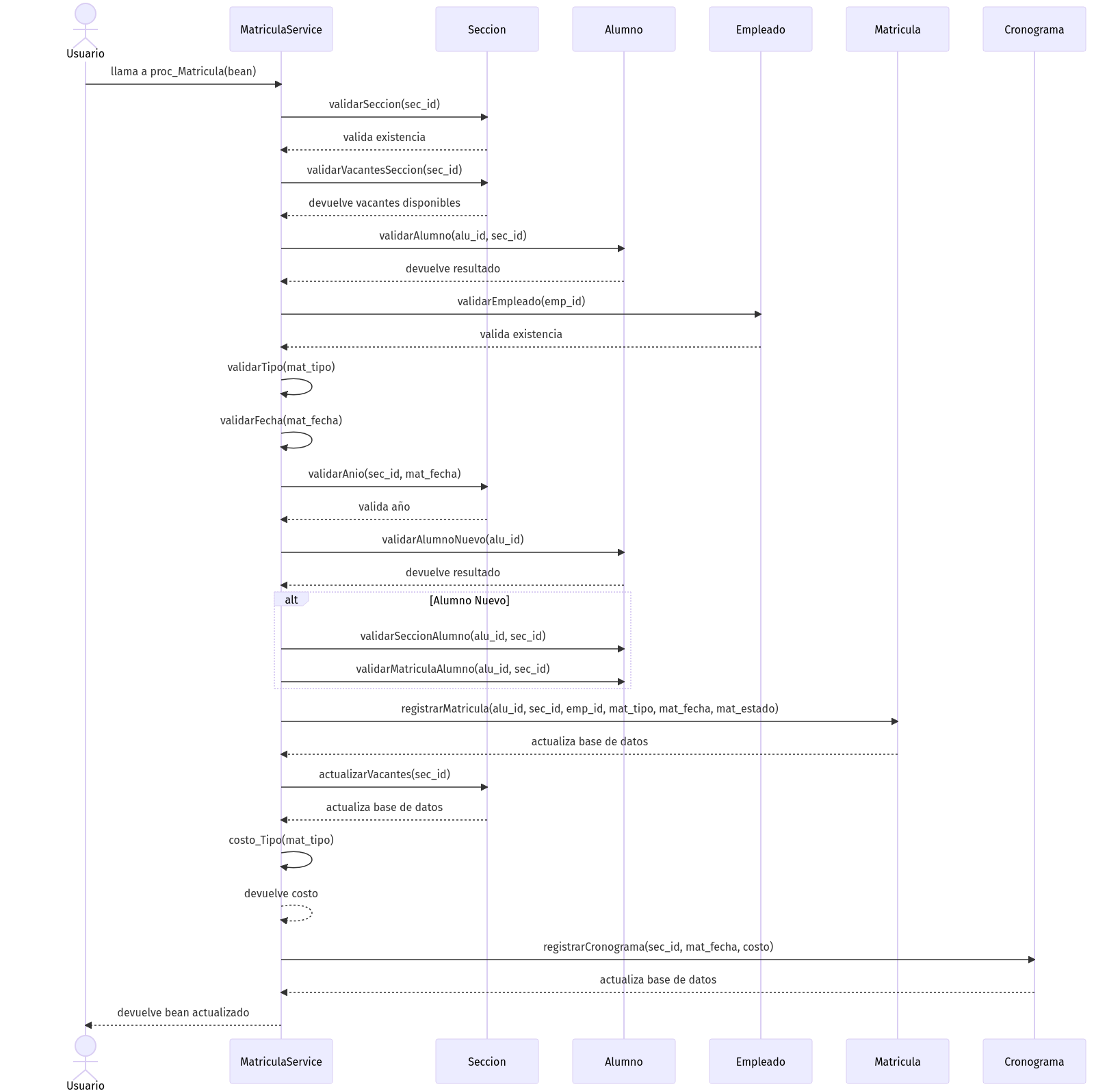
## ESPECIFICACIÓN DEL CASO DE USO

|  |  |  |
| --- | --- | --- |
| **Código** | **CU01** | |
| **Nombre** | Proceso de matrícula de un alumno | |
| **Descripción** | Permite al personal administrativo registrar a un estudiante en una sección específica del colegio. Durante el proceso de matrícula, el sistema valida las vacantes disponibles en la sección, la información del estudiante, y el tipo de matrícula (beca, mediabeca o regular). Una vez registrada la matrícula, se genera automáticamente un cronograma de pagos asociado. | |
| **Actores** | **Personal Administrativo** | |
| **Requerimiento asociado** | El sistema debe permitir registrar nuevos estudiantes en una sección y generar automáticamente el cronograma de pagos correspondiente. | |
| **Casos de uso asociados** | N/A | |
| **Precondición** | El estudiante debe existir en la base de datos.  El personal administrativo debe estar autenticado en el sistema.  La sección debe tener vacantes disponibles. | |
| **Secuencia normal** | **Paso** | **Acción** |
| **1** | El personal administrativo selecciona al estudiante a matricular. |
| **2** | El sistema valida que el estudiante existe en la base de datos. |
| **3** | El personal administrativo selecciona la sección en la que se matriculará al estudiante. |
| **4** | El sistema valida que el alumno exista y pueda ser matriculado en esa sección. |
| **5** | El personal administrativo ingresa el tipo de matrícula (beca, mediabeca, regular) y la fecha de matrícula. |
| **6** | El sistema valida la fecha y calcula el costo correspondiente según el tipo de matrícula. |
| **7** | El sistema registra la matrícula en la base de datos y actualiza las vacantes disponibles. |
| **8** | El sistema genera automáticamente el cronograma de pagos asociado al estudiante. |
| **Postcondición** | El estudiante queda registrado en la sección seleccionada.  El cronograma de pagos se genera automáticamente en la base de datos. | |
| **Excepciones** | **Paso** | **Acción** |
| **2** | El estudiante no existe en la base de datos, lo que impide continuar con la matrícula. |
| **4** | La sección no existe o no tiene vacantes disponibles. |
| **5** | El tipo de matrícula ingresado no es válido (sólo se permiten "beca", "mediabeca" o "regular"). |
| **6** | La fecha ingresada no tiene un formato válido o no corresponde al año académico. |
| **7** | Error al registrar la matrícula en la base de datos. |
| **Comentarios** | Este caso de uso es fundamental para el inicio del proceso académico de los estudiantes. Las validaciones integradas aseguran que solo se matriculen estudiantes en secciones con vacantes disponibles y que el cronograma de pagos se genere correctamente según el tipo de matrícula. | |

## ENFOQUE DE LA CAJA NEGRA

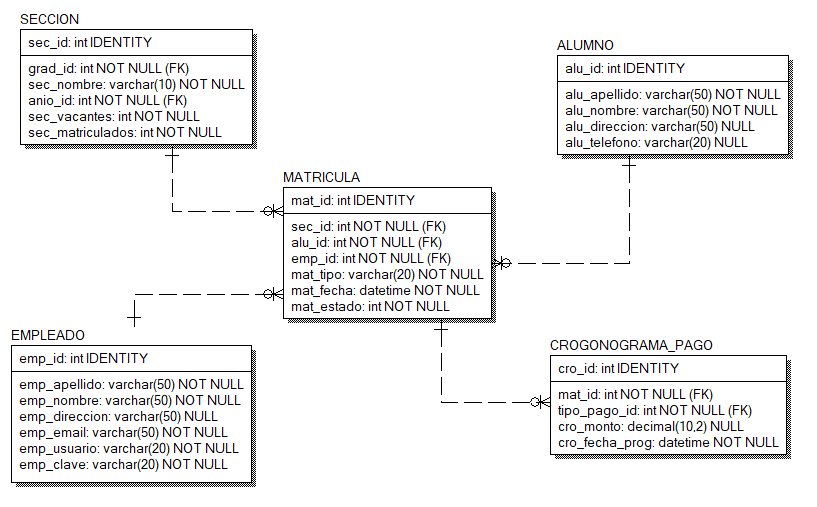


## DIAGRAMA DE SECUENCIA



## 

## MODELO DE BASE DE DATOS



## CRITERIOS DE ACEPTACIÓN

#### 1. Validaciones Previas

* **Verificación de Vacantes:**
  + Se debe verificar que la sección seleccionada (sec\_id) tenga vacantes disponibles antes de registrar al estudiante.
* **Existencia del Alumno:**
  + El alumno (alu\_id) debe existir en la tabla ALUMNO.
* **Empleado Autorizado:**
  + El empleado (emp\_id) debe existir en la tabla EMPLEADO y estar autorizado para realizar la matrícula.
* **Validación del Tipo de Matrícula:**
  + El tipo de matrícula (mat\_tipo) debe ser uno de los valores permitidos: REGULAR, BECA o MEDIABECA.
* **Formato de Fecha:**
  + La fecha de matrícula (mat\_fecha) debe estar en el formato dd/MM/yyyy y corresponder al año académico de la sección.
* **Restricción de Grado del Alumno:**
  + El alumno solo puede matricularse en el último grado cursado o el siguiente grado.
* **Matrícula Única por Año Académico:**
  + El alumno no puede estar matriculado en otra sección del mismo año académico.

#### 2. Registro de Matrícula

* **Asociación a una Sección:**
  + El estudiante debe quedar asociado a la sección seleccionada (sec\_id), y la tabla MATRICULA debe actualizarse correctamente.
* **Actualización de Vacantes:**
  + Al registrar la matrícula, el número de matriculados (sec\_matriculados) debe incrementarse, y el número de vacantes (sec\_vacantes) debe disminuir en la tabla SECCION.

#### 3. Generación de Cronograma de Pagos

* **Generación Automática:**
  + Al completar la matrícula, se debe generar un cronograma de pagos en la tabla CRONOGRAMA\_PAGO, asociado al registro de matrícula (mat\_id).
* **Validación del Cronograma:**
  + El cronograma debe incluir todas las cuotas correspondientes (de febrero a diciembre), con los montos y fechas programadas correctamente establecidos.

#### 4. Confirmación del Proceso

* **Respuesta Exitosa:**
  + El sistema debe devolver una confirmación de que la matrícula se registró exitosamente, incluyendo los detalles del cronograma generado.
* **Manejo de Errores:**
  + Si alguna de las validaciones falla, el sistema debe devolver un mensaje claro indicando la causa del error (por ejemplo, "No hay vacantes disponibles" o "El alumno ya está matriculado").

#### 5. Consistencia e Integridad

* **Relaciones entre Tablas:**
  + La relación entre MATRICULA y ALUMNO, SECCION, y CRONOGRAMA\_PAGO debe mantenerse consistente a través de claves foráneas.
* **Estado de la Matrícula:**
  + La columna mat\_estado en la tabla MATRICULA debe ser actualizada para reflejar que la matrícula está activa (1).

# CU02 – REGISTRAR PAGO DE CUOTA

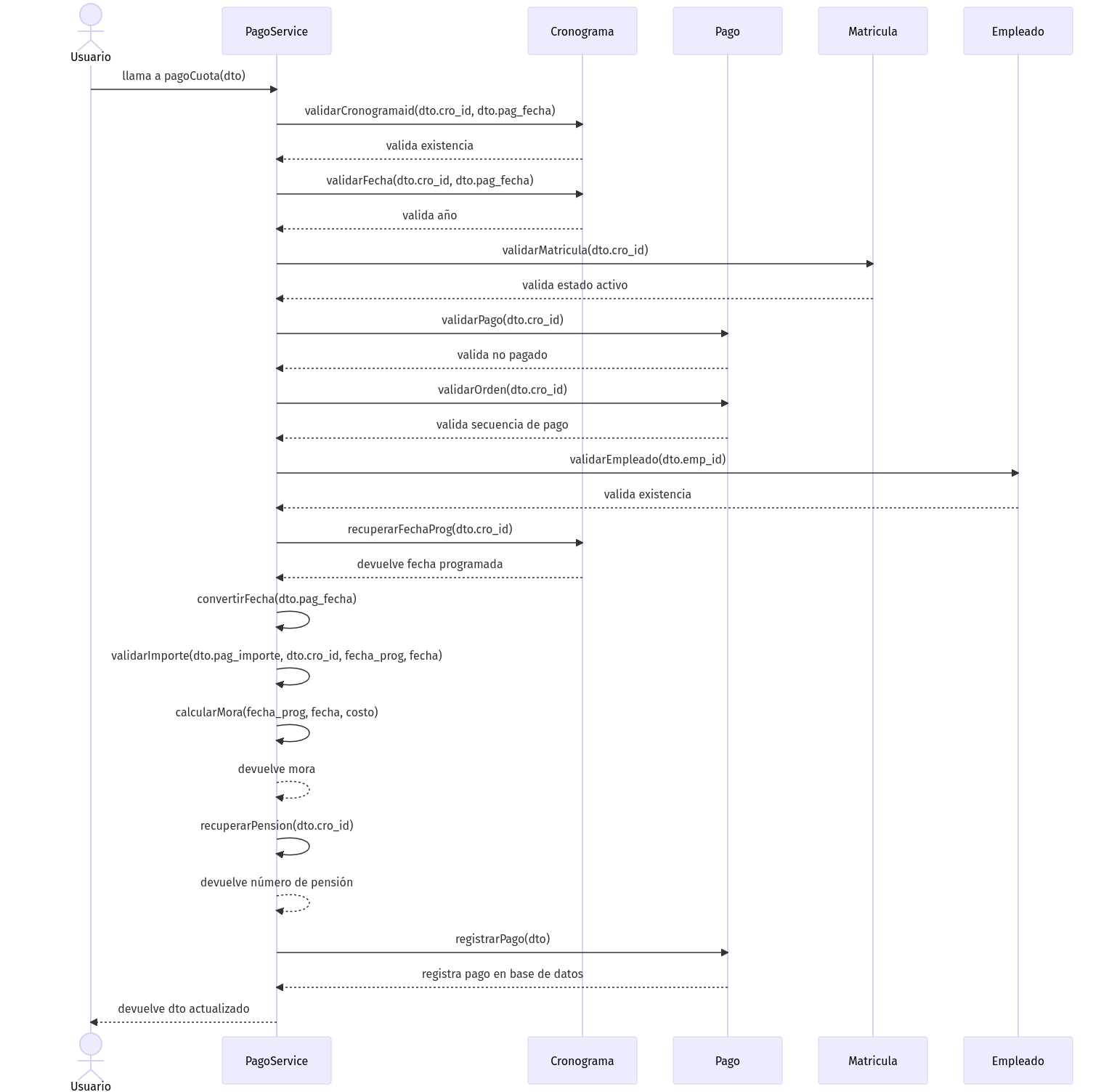
## ESPECIFICACIÓN DEL CASO DE USO

|  |  |  |
| --- | --- | --- |
| **Código** | **CU02** | |
| **Nombre** | **Registrar Pago de Cuota** | |
| **Descripción** | Permite validar y registrar el pago de una cuota correspondiente al cronograma del estudiante, incluyendo recargos por mora en caso de retraso. Este proceso asegura la integridad de los datos de matrícula y los pagos realizados. | |
| **Actores** | **Personal Administrativo** | |
| **Requerimiento asociado** | Validación del cronograma de pagos y cálculo de mora para pagos retrasados.  Registro en la base de datos del pago, asegurando su correspondencia con el cronograma. | |
| **Casos de uso asociados** | **CU01:** Registrar Matrícula (ya que el cronograma de pagos depende de la matrícula activa). | |
| **Precondición** | * El empleado debe estar autenticado en el sistema. * La matrícula del estudiante debe estar activa. * El cronograma de pagos debe incluir la cuota correspondiente al pago que se intenta registrar. * No se debe haber realizado un pago previo para la misma cuota. | |
| **Secuencia normal** | **Paso** | **Acción** |
| **1** | El empleado selecciona al estudiante y la cuota a registrar como pagada. |
| **2** | El sistema verifica que el cronograma de pagos correspondiente al estudiante existe. |
| **3** | El sistema valida que la matrícula del estudiante está activa. |
| **4** | El sistema verifica que la cuota no haya sido previamente pagada. |
| **5** | El sistema valida que la fecha de pago corresponde al año del cronograma programado. |
| **6** | El sistema calcula el recargo por mora, si aplica, basándose en la diferencia entre la fecha programada y la fecha actual. |
| **7** | El empleado registra el importe del pago. |
| **8** | El sistema verifica que el importe ingresado es correcto, considerando mora, si aplica. |
| **9** | Si todas las validaciones son correctas, el sistema registra el pago en la base de datos. |
| **Postcondición** | * El pago queda registrado en la tabla PAGO junto con su fecha, importe, recargo por mora (si aplica), y el cronograma asociado. * La cuota se marca como pagada en el cronograma de pagos. | |
| **Excepciones** | **Paso** | **Acción** |
| **2** | Si el cronograma no existe para el estudiante, el sistema genera un error: "Cronograma no encontrado". |
| **3** | Si la matrícula del estudiante no está activa, se muestra un mensaje: "Matrícula inactiva". |
| **4** | Si la cuota ya fue pagada, el sistema genera un mensaje: "La cuota ya ha sido pagada". |
| **5** | Si la fecha no corresponde al año del cronograma, el sistema muestra un error: "Fecha inválida". |
| **8** | Si el importe ingresado no corresponde al monto esperado, el sistema genera un error: "Importe incorrecto". |
| **9** | Si ocurre un error al registrar el pago en la base de datos, el sistema muestra: "Error al registrar el pago". |
| **Comentarios** | * Este caso de uso garantiza que solo se puedan registrar pagos válidos para el cronograma asignado. * Cálculo de mora: El sistema utiliza la diferencia en días entre la fecha programada y la fecha de pago para calcular recargos. * El proceso es transaccional, lo que asegura que no se registren pagos inconsistentes. | |

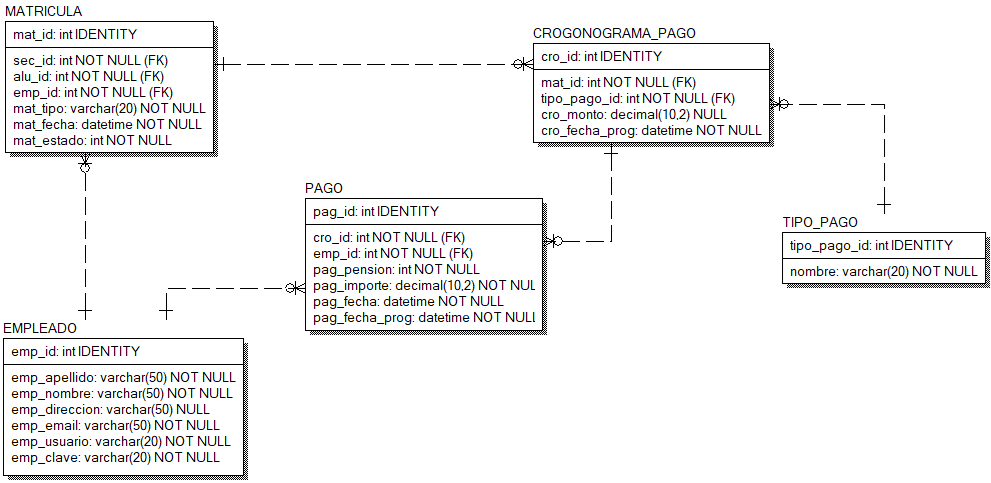
## CAJA NEGRA



## DIAGRAMA DE SECUENCIA



## MODELO DE BASE DE DATOS



## CRITERIOS DE ACEPTACIÓN

**Validación de Cronograma de Pagos**

* El sistema debe validar que el cronograma de pagos existe para el estudiante antes de permitir el registro de un pago.
* Si el cronograma no existe, se debe generar un mensaje de error: *"Cronograma de pagos no encontrado para el estudiante seleccionado"*.

**Verificación de Matrícula Activa**

* El sistema debe verificar que la matrícula del estudiante esté activa.
* Si la matrícula no está activa, se debe generar un mensaje: *"La matrícula del estudiante está inactiva"*.

**Verificación de Cuota Pagada**

* El sistema no debe permitir registrar un pago para una cuota que ya ha sido pagada previamente.
* Si la cuota ya está pagada, se debe mostrar un mensaje de error: *"Esta cuota ya ha sido pagada"*.

**Validación de Fecha de Pago**

* La fecha ingresada debe corresponder al año del cronograma del estudiante.
* Si la fecha no es válida, el sistema debe mostrar un error: *"La fecha ingresada no corresponde al año del cronograma"*.

**Cálculo de Mora**

* Si la fecha actual excede la fecha programada para la cuota, el sistema debe calcular correctamente el recargo por mora, aplicando la fórmula:

mora = (diferencia de días) \* (tasa de mora) \* costo

* El recargo por mora debe reflejarse en el monto final esperado.

**Validación de Importe**

* El sistema debe validar que el importe ingresado por el empleado corresponde al monto esperado, incluyendo el recargo por mora (si aplica).
* Si el importe no coincide, se debe generar un error: *"El importe ingresado no corresponde al monto esperado"*.

**Registro del Pago**

* El sistema debe registrar el pago correctamente en la base de datos con los siguientes datos:
  + Identificador del cronograma de pagos.
  + Identificador del empleado que registra el pago.
  + Monto de la cuota.
  + Fecha programada y fecha real del pago.
* Si ocurre un error al registrar el pago, el sistema debe mostrar un mensaje: *"Error al registrar el pago"*.

**Secuencia de Pagos**

* El sistema debe garantizar que los pagos se realicen en orden. Es decir, no se permite pagar cuotas posteriores sin haber pagado las anteriores.
* Si el pago no sigue el orden establecido, se debe generar un mensaje de error: *"El pago debe realizarse en el orden correcto"*.

**Transacción Segura**

* Todas las validaciones deben realizarse dentro de una transacción, de forma que si hay alguna falla, el sistema revierta cualquier operación incompleta.

**Confirmación de Registro**

* Al completar exitosamente el registro del pago, el sistema debe generar un mensaje de confirmación: *"Pago registrado exitosamente"*.

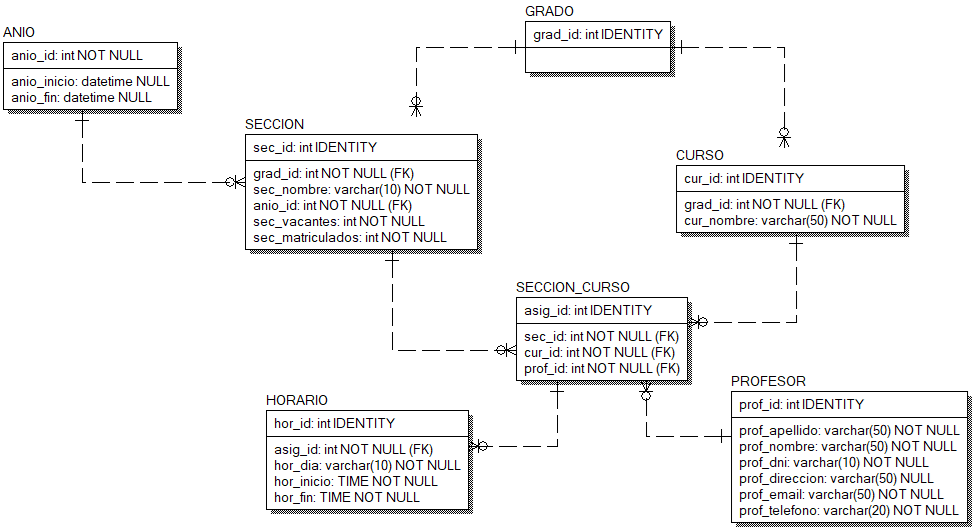
# CU03 – CONSULTAS

## ESPECIFICACIÓN DEL CASO DE USO

|  |  |  |
| --- | --- | --- |
| **Código** | **CU03** | |
| **Nombre** | **Consultar información general del sistema** | |
| **Descripción** | Este caso de uso permite realizar consultas específicas en el sistema por parte de los actores involucrados. Los estudiantes pueden consultar cronogramas de pago y horarios de su sección, mientras que los profesores pueden visualizar horarios y cursos asignados. | |
| **Actores** | **Profesor:** Consulta horarios y cursos asignados.  **Alumno:** Consulta cronogramas de pago y horarios. | |
| **Requerimiento asociado** | * Visualización de horarios para secciones específicas. * Consulta de cronogramas de pago de estudiantes. * Acceso a información detallada de cursos para profesores. | |
| **Casos de uso asociados** | **CU01 –** Registrar Matrícula (el estudiante debe estar matriculado para visualizar horarios y cronogramas de pago).  **CU02 –** Registrar Pago (los cronogramas de pago dependen del registro de matrículas y pagos asociados). | |
| **Precondición** | * El usuario debe estar registrado en el sistema como Profesor o Alumno. * Los datos requeridos (horarios, cronogramas de pago, cursos) deben estar previamente registrados en la base de datos. | |
| **Secuencia normal** | **Paso** | **Acción** |
| **1** | El usuario (Profesor o Alumno) accede al sistema e inicia sesión. |
| **2** | El usuario selecciona la opción correspondiente en el sistema: "Cronograma de Pagos" o "Horarios". |
| **3** | El sistema valida que el usuario exista en la base de datos mediante su ID. |
|  |  |
| **4** | Caso Alumno: Si selecciona "Cronograma de Pagos", el sistema verifica que esté matriculado en alguna sección.. |
| **5** | Caso Alumno: El sistema consulta y organiza el cronograma de pagos del estudiante, mostrando las fechas y montos en un formato claro. |
|  | **6** | Caso Alumno: Si selecciona "Horarios", el sistema verifica que esté matriculado en una sección válida. |
|  | **7** | Caso Alumno: El sistema consulta los horarios asignados a la sección del estudiante y organiza la información en un formato visual (días, horas). |
|  | **8** | Caso Profesor: Si selecciona "Horarios", el sistema verifica que tenga cursos asignados en la base de datos. |
|  | **9** | Caso Profesor: El sistema consulta los horarios y cursos asignados al profesor y organiza la información en un formato visual (días, horas, secciones). |
|  | **10** | El sistema muestra la información consultada al usuario (Cronograma de Pagos, Horarios del Estudiante o Horarios del Profesor). |
| **Postcondición** | El sistema garantiza que los datos visualizados son correctos y actualizados.  El actor puede realizar consultas adicionales si lo desea.. | |
| **Excepciones** | **Paso** | **Acción** |
| **3** | Si el usuario no está registrado en la base de datos, el sistema muestra un mensaje indicando que no tiene acceso. |
| **5** | Si el Alumno no está matriculado, el sistema muestra un mensaje indicando "Alumno no matriculado". |
| **9** | Si el Profesor no tiene cursos asignados, el sistema muestra un mensaje indicando "Sin horarios asignados". |
| **Comentarios** | * La funcionalidad de consulta utiliza múltiples validaciones para garantizar que solo los usuarios autorizados accedan a la información relevante. * El sistema debe garantizar que las consultas sean rápidas y que los datos estén actualizados en tiempo real. * El backend organiza los datos en un formato predefinido (como JSON o tabla) para facilitar la visualización en el frontend. | |

## CAJA NEGRA

## MODELO DE BASE DE DATOS



## CRITERIOS DE ACEPTACIÓN

### **Validación del Usuario**

* **Descripción:** El sistema debe verificar que el usuario que realiza la consulta está registrado en la base de datos.
* **Criterios:**
  + El sistema debe mostrar un mensaje de error si el usuario no está registrado: *"Usuario no registrado en el sistema."*.
  + Esta validación aplica tanto para profesores como para alumnos, pero no se hace diferenciación explícita de roles en esta versión del sistema.

### **Consulta de Cronograma de Pagos (Alumno)**

* **Descripción:** El sistema debe permitir consultar el cronograma de pagos de un usuario, considerando que los datos consultados serán únicamente de alumnos.
* **Criterios:**
  + El cronograma debe incluir todas las cuotas pendientes y pagadas.
  + Los datos deben mostrarse en un formato estructurado: *Cuota, Monto a Pagar, Fecha de Pago*.
  + Si el alumno no está matriculado, el sistema debe mostrar el mensaje: *"El alumno no está matriculado."*.

### **Consulta de Horarios (Estudiantes y Profesores)**

* **Descripción:** El sistema debe permitir la consulta de horarios asignados para estudiantes (sección) y profesores (cursos y secciones asignadas).
* **Criterios:**
  + Los horarios deben incluir información sobre días, horas y cursos correspondientes.
  + Si el usuario no tiene horarios asignados, el sistema debe mostrar el mensaje: *"No se han encontrado horarios asignados."*.
  + La información debe consolidarse y mostrarse en una única tabla para estudiantes y profesores, sin separación explícita.

### **Consolidación y Presentación de Datos**

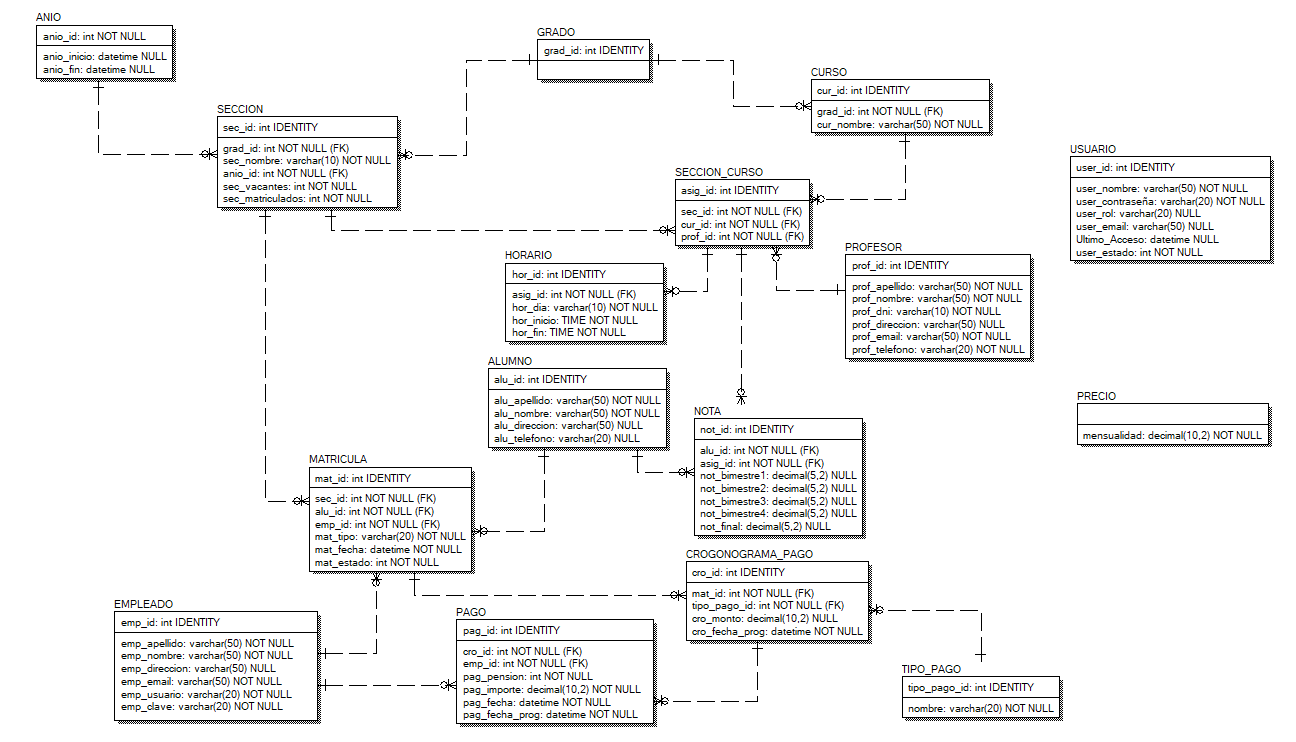
* **Descripción:** La información obtenida en cualquier consulta debe estar consolidada y organizada para una presentación clara al usuario.
* **Criterios:**
  + El sistema debe mostrar los datos en un formato visualmente comprensible (tablas en la interfaz o estructuras JSON para el backend).
  + La información debe estar ordenada cronológicamente en el caso de pagos y horarios.
  + Los encabezados de las tablas deben ser claros y representativos de los datos mostrados.

### **Manejo de Errores y Excepciones**

* **Descripción:** El sistema debe manejar errores y excepciones de manera que no afecte la experiencia del usuario ni la integridad del sistema.
* **Criterios:**
  + El sistema debe mostrar mensajes de error amigables al usuario en caso de fallas o datos inexistentes.
  + Ejemplo de mensajes:
    - *"Alumno no existe."*
    - *"No se encontraron horarios asignados."*
    - *"Sección no encontrada."*
  + Los mensajes de error no deben exponer detalles técnicos del sistema.

# MODELO DE BASE DE DATOS

## MODELO RELACIONAL



## DICCIONARIO DE DATOS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Database Name** | **Table Name** | **Schema Name** | **Column Name** | **Data Type** | **Max Length** | **Is Null** | **Identity** |
| BDCOLEGIO | ALUMNO | dbo | alu\_id | int | 4 | NO | YES |
| BDCOLEGIO | ALUMNO | dbo | alu\_apellido | varchar | 100 | NO | NO |
| BDCOLEGIO | ALUMNO | dbo | alu\_nombre | varchar | 100 | NO | NO |
| BDCOLEGIO | ALUMNO | dbo | alu\_direccion | varchar | 100 | NO | NO |
| BDCOLEGIO | ALUMNO | dbo | alu\_telefono | varchar | 20 | YES | NO |
| BDCOLEGIO | ANIO | dbo | anio\_id | int | 4 | NO | NO |
| BDCOLEGIO | ANIO | dbo | anio\_inicio | datetime | 8 | NO | NO |
| BDCOLEGIO | ANIO | dbo | anio\_fin | datetime | 8 | NO | NO |
| BDCOLEGIO | CRONOGRAMA\_PAGO | dbo | cro\_id | int | 4 | NO | YES |
| BDCOLEGIO | CRONOGRAMA\_PAGO | dbo | mat\_id | int | 4 | NO | NO |
| BDCOLEGIO | CRONOGRAMA\_PAGO | dbo | tipo\_pago\_id | int | 4 | NO | NO |
| BDCOLEGIO | CRONOGRAMA\_PAGO | dbo | cro\_monto | decimal | 9 | NO | NO |
| BDCOLEGIO | CRONOGRAMA\_PAGO | dbo | cro\_fecha\_prog | datetime | 8 | NO | NO |
| BDCOLEGIO | CURSO | dbo | cur\_id | int | 4 | NO | YES |
| BDCOLEGIO | CURSO | dbo | grad\_id | int | 4 | NO | NO |
| BDCOLEGIO | CURSO | dbo | cur\_nombre | varchar | 100 | NO | NO |
| BDCOLEGIO | EMPLEADO | dbo | emp\_id | int | 4 | NO | YES |
| BDCOLEGIO | EMPLEADO | dbo | emp\_apellido | varchar | 100 | NO | NO |
| BDCOLEGIO | EMPLEADO | dbo | emp\_nombre | varchar | 100 | NO | NO |
| BDCOLEGIO | EMPLEADO | dbo | emp\_direccion | varchar | 100 | NO | NO |
| BDCOLEGIO | EMPLEADO | dbo | emp\_email | varchar | 100 | NO | NO |
| BDCOLEGIO | EMPLEADO | dbo | emp\_usuario | varchar | 20 | NO | NO |
| BDCOLEGIO | EMPLEADO | dbo | emp\_clave | varchar | 100 | NO | NO |
| BDCOLEGIO | GRADO | dbo | grad\_id | int | 4 | NO | YES |
| BDCOLEGIO | HORARIO | dbo | hor\_id | int | 4 | NO | YES |
| BDCOLEGIO | HORARIO | dbo | asig\_id | int | 4 | NO | NO |
| BDCOLEGIO | HORARIO | dbo | hor\_dia | varchar | 10 | NO | NO |
| BDCOLEGIO | HORARIO | dbo | hor\_inicio | time | 5 | NO | NO |
| BDCOLEGIO | HORARIO | dbo | hor\_fin | time | 5 | NO | NO |
| BDCOLEGIO | MATRICULA | dbo | mat\_id | int | 4 | NO | YES |
| BDCOLEGIO | MATRICULA | dbo | alu\_id | int | 4 | NO | NO |
| BDCOLEGIO | MATRICULA | dbo | sec\_id | int | 4 | NO | NO |
| BDCOLEGIO | MATRICULA | dbo | emp\_id | int | 4 | NO | NO |
| BDCOLEGIO | MATRICULA | dbo | mat\_tipo | varchar | 20 | NO | NO |
| BDCOLEGIO | MATRICULA | dbo | mat\_fecha | datetime | 8 | NO | NO |
| BDCOLEGIO | MATRICULA | dbo | mat\_estado | varchar | 10 | YES | NO |
| BDCOLEGIO | NOTA | dbo | not\_id | int | 4 | NO | YES |
| BDCOLEGIO | NOTA | dbo | alu\_id | int | 4 | NO | NO |
| BDCOLEGIO | NOTA | dbo | asig\_id | int | 4 | NO | NO |
| BDCOLEGIO | NOTA | dbo | not\_bimestre1 | decimal | 5 | YES | NO |
| BDCOLEGIO | NOTA | dbo | not\_bimestre2 | decimal | 5 | YES | NO |
| BDCOLEGIO | NOTA | dbo | not\_bimestre3 | decimal | 5 | YES | NO |
| BDCOLEGIO | NOTA | dbo | not\_bimestre4 | decimal | 5 | YES | NO |
| BDCOLEGIO | NOTA | dbo | not\_final | decimal | 5 | YES | NO |
| BDCOLEGIO | PAGO | dbo | pag\_id | int | 4 | NO | YES |
| BDCOLEGIO | PAGO | dbo | cro\_id | int | 4 | NO | NO |
| BDCOLEGIO | PAGO | dbo | emp\_id | int | 4 | NO | NO |
| BDCOLEGIO | PAGO | dbo | pag\_pension | int | 4 | NO | NO |
| BDCOLEGIO | PAGO | dbo | pag\_importe | decimal | 9 | NO | NO |
| BDCOLEGIO | PAGO | dbo | pag\_fecha | datetime | 8 | NO | NO |
| BDCOLEGIO | PAGO | dbo | pag\_fecha\_prog | datetime | 8 | NO | NO |
| BDCOLEGIO | PRECIO | dbo | mensualidad | decimal | 9 | NO | NO |
| BDCOLEGIO | PROFESOR | dbo | prof\_id | int | 4 | NO | YES |
| BDCOLEGIO | PROFESOR | dbo | prof\_apellido | varchar | 100 | NO | NO |
| BDCOLEGIO | PROFESOR | dbo | prof\_nombre | varchar | 100 | NO | NO |
| BDCOLEGIO | PROFESOR | dbo | prof\_dni | varchar | 15 | NO | NO |
| BDCOLEGIO | PROFESOR | dbo | prof\_direccion | varchar | 100 | YES | NO |
| BDCOLEGIO | PROFESOR | dbo | prof\_email | varchar | 100 | YES | NO |
| BDCOLEGIO | PROFESOR | dbo | prof\_telefono | varchar | 15 | YES | NO |
| BDCOLEGIO | SECCION | dbo | sec\_id | int | 4 | NO | YES |
| BDCOLEGIO | SECCION | dbo | grad\_id | int | 4 | NO | NO |
| BDCOLEGIO | SECCION | dbo | sec\_nombre | varchar | 10 | NO | NO |
| BDCOLEGIO | SECCION | dbo | anio\_id | int | 4 | NO | NO |
| BDCOLEGIO | SECCION | dbo | sec\_vacantes | int | 4 | NO | NO |
| BDCOLEGIO | SECCION | dbo | sec\_matriculados | int | 4 | NO | NO |
| BDCOLEGIO | SECCION\_CURSO | dbo | asig\_id | int | 4 | NO | YES |
| BDCOLEGIO | SECCION\_CURSO | dbo | sec\_id | int | 4 | NO | NO |
| BDCOLEGIO | SECCION\_CURSO | dbo | cur\_id | int | 4 | NO | NO |
| BDCOLEGIO | SECCION\_CURSO | dbo | prof\_id | int | 4 | NO | NO |
| BDCOLEGIO | TIPO\_PAGO | dbo | tipo\_pago\_id | int | 4 | NO | YES |
| BDCOLEGIO | TIPO\_PAGO | dbo | tipo\_pago\_nombre | varchar | 20 | NO | NO |
| BDCOLEGIO | USUARIO | dbo | id\_usuario | int | 4 | NO | YES |
| BDCOLEGIO | USUARIO | dbo | nombre\_usuario | varchar | 50 | NO | NO |
| BDCOLEGIO | USUARIO | dbo | contrasena | varchar | 255 | NO | NO |
| BDCOLEGIO | USUARIO | dbo | rol | varchar | 20 | NO | NO |
| BDCOLEGIO | USUARIO | dbo | correo\_electronico | varchar | 100 | YES | NO |
| BDCOLEGIO | USUARIO | dbo | ultimo\_acceso | datetime | 8 | YES | NO |
| BDCOLEGIO | USUARIO | dbo | estado\_activo | bit | 1 | YES | NO |

# IMPLEMENTACIÓN DE LA BASE DE DATOS

## CREACIÓN DE LA BASE DE DATOS

USE MASTER;

GO

IF NOT EXISTS (SELECT NAME FROM sys.databases WHERE NAME = 'BDCOLEGIO')

CREATE DATABASE BDCOLEGIO;

GO

USE BDCOLEGIO;

GO

## CREACIÓN DE LOS OBJETOS DE LA BASE DE DATOS

-- Tabla ANIO

CREATE TABLE ANIO (

anio\_id INT PRIMARY KEY,

anio\_inicio DATETIME NULL,

anio\_fin DATETIME NULL

);

-- Tabla GRADO

CREATE TABLE GRADO (

grad\_id INT PRIMARY KEY IDENTITY(1,1)

);

-- Tabla PRECIO

CREATE TABLE PRECIO (

mensualidad DECIMAL(10,2) NOT NULL

);

-- Tabla SECCION

CREATE TABLE SECCION (

sec\_id INT PRIMARY KEY IDENTITY(1,1),

grad\_id INT NOT NULL,

sec\_nombre VARCHAR(10) NOT NULL,

anio\_id INT NOT NULL,

sec\_vacantes INT NOT NULL,

sec\_matriculados INT NOT NULL,

FOREIGN KEY (anio\_id) REFERENCES ANIO(anio\_id),

FOREIGN KEY (grad\_id) REFERENCES GRADO(grad\_id)

);

-- Tabla CURSO

CREATE TABLE CURSO (

cur\_id INT PRIMARY KEY IDENTITY(1,1),

grad\_id INT NOT NULL,

cur\_nombre VARCHAR(50) NOT NULL,

FOREIGN KEY (grad\_id) REFERENCES GRADO(grad\_id)

);

-- Tabla ALUMNO

CREATE TABLE ALUMNO (

alu\_id INT PRIMARY KEY IDENTITY(1,1),

alu\_apellido VARCHAR(50) NOT NULL,

alu\_nombre VARCHAR(50) NOT NULL,

alu\_direccion VARCHAR(50) NOT NULL,

alu\_telefono VARCHAR(20) NULL

);

-- Tabla EMPLEADO

CREATE TABLE EMPLEADO (

emp\_id INT PRIMARY KEY IDENTITY(1,1),

emp\_apellido VARCHAR(50) NOT NULL,

emp\_nombre VARCHAR(50) NOT NULL,

emp\_direccion VARCHAR(50) NOT NULL,

emp\_email VARCHAR(50) NOT NULL,

emp\_usuario VARCHAR(20) NOT NULL,

emp\_clave VARCHAR(20) NOT NULL

);

-- Tabla MATRICULA

CREATE TABLE MATRICULA (

mat\_id INT PRIMARY KEY IDENTITY(1,1),

alu\_id INT NOT NULL,

sec\_id INT NOT NULL,

emp\_id INT NOT NULL,

mat\_tipo VARCHAR(20) NOT NULL CHECK (mat\_tipo IN ('REGULAR', 'BECA', 'MEDIABECA')),

mat\_fecha DATETIME NOT NULL,

mat\_estado INT NOT NULL CHECK (mat\_estado IN (0,1)),

FOREIGN KEY (alu\_id) REFERENCES ALUMNO(alu\_id),

FOREIGN KEY (sec\_id) REFERENCES SECCION(sec\_id),

FOREIGN KEY (emp\_id) REFERENCES EMPLEADO(emp\_id)

);

-- Tabla TIPO\_PAGO

CREATE TABLE TIPO\_PAGO (

tipo\_pago\_id INT PRIMARY KEY IDENTITY(1,1),

tipo\_pago\_nombre VARCHAR(20) NOT NULL

);

-- Tabla CRONOGRAMA\_PAGO

CREATE TABLE CRONOGRAMA\_PAGO (

cro\_id INT PRIMARY KEY IDENTITY(1,1),

mat\_id INT NOT NULL, -- Cambio realizado: Se agregó mat\_id como clave foránea y NOT NULL para asociar con MATRICULA

tipo\_pago\_id INT NOT NULL, -- Cambio realizado: Se agregó tipo\_pago\_id como clave foránea y NOT NULL para asociar con TIPO\_PAGO

cro\_monto DECIMAL(10,2) NOT NULL,

cro\_fecha\_prog DATETIME NOT NULL,

FOREIGN KEY (mat\_id) REFERENCES MATRICULA(mat\_id),

FOREIGN KEY (tipo\_pago\_id) REFERENCES TIPO\_PAGO(tipo\_pago\_id)

);

-- Tabla PAGO

CREATE TABLE PAGO (

pag\_id INT PRIMARY KEY IDENTITY(1,1),

cro\_id INT NOT NULL, -- Cambio realizado: Se agregó cro\_id como clave foránea NOT NULL para asociar con CRONOGRAMA\_PAGO

emp\_id INT NOT NULL,

pag\_pension INT NOT NULL,

pag\_importe DECIMAL(10,2) NOT NULL,

pag\_fecha DATETIME NOT NULL,

pag\_fecha\_prog DATETIME NOT NULL,

FOREIGN KEY (cro\_id) REFERENCES CRONOGRAMA\_PAGO(cro\_id),

FOREIGN KEY (emp\_id) REFERENCES EMPLEADO(emp\_id)

);

-- Tabla PROFESOR

CREATE TABLE PROFESOR (

prof\_id INT PRIMARY KEY IDENTITY(1,1),

prof\_apellido VARCHAR(50) NOT NULL,

prof\_nombre VARCHAR(50) NOT NULL,

prof\_dni VARCHAR(10) NOT NULL,

prof\_direccion VARCHAR(50) NULL,

prof\_email VARCHAR(50) NOT NULL,

prof\_telefono VARCHAR(20) NOT NULL

);

-- Tabla SECCION\_CURSO

CREATE TABLE SECCION\_CURSO (

asig\_id INT PRIMARY KEY IDENTITY(1,1),

sec\_id INT NOT NULL,

cur\_id INT NOT NULL,

prof\_id INT NOT NULL,

FOREIGN KEY (sec\_id) REFERENCES SECCION(sec\_id),

FOREIGN KEY (cur\_id) REFERENCES CURSO(cur\_id),

FOREIGN KEY (prof\_id) REFERENCES PROFESOR(prof\_id)

);

-- Tabla NOTA

CREATE TABLE NOTA (

not\_id INT PRIMARY KEY IDENTITY(1,1),

alu\_id INT NOT NULL,

asig\_id INT NOT NULL,

not\_bimestre1 DECIMAL(5,2) NULL,

not\_bimestre2 DECIMAL(5,2) NULL,

not\_bimestre3 DECIMAL(5,2) NULL,

not\_bimestre4 DECIMAL(5,2) NULL,

not\_final DECIMAL(5,2) NULL,

FOREIGN KEY (alu\_id) REFERENCES ALUMNO(alu\_id),

FOREIGN KEY (asig\_id) REFERENCES SECCION\_CURSO(asig\_id)

);

-- Tabla HORARIO

CREATE TABLE HORARIO (

hor\_id INT PRIMARY KEY IDENTITY(1,1),

asig\_id INT NOT NULL,

hor\_dia VARCHAR(10) NOT NULL CHECK (hor\_dia IN ('LUNES', 'MARTES', 'MIERCOLES', 'JUEVES', 'VIERNES')),

hor\_inicio TIME NOT NULL,

hor\_fin TIME NOT NULL,

FOREIGN KEY (asig\_id) REFERENCES SECCION\_CURSO(asig\_id)

);

-- Tabla USUARIO

CREATE TABLE USUARIO (

user\_id INT PRIMARY KEY IDENTITY(1,1),

user\_nombre VARCHAR(50) UNIQUE NOT NULL,

user\_contrasena VARCHAR(20) NOT NULL,

user\_rol VARCHAR(20) NOT NULL CHECK (user\_rol IN ('DIRECTOR', 'SUBDIRECTOR', 'PROFESOR', 'ADMINISTRATIVO', 'SUPERADMIN')),

user\_email VARCHAR(50) NOT NULL,

ultimo\_acceso DATETIME NULL,

user\_estado INT NOT NULL DEFAULT 1

);

## CARGAR DATOS DE PRUEBA

-- Insertar Grados

INSERT INTO GRADO DEFAULT VALUES;

GO 6

-- Insertar Usuarios

INSERT INTO USUARIO (user\_nombre,user\_contrasena,user\_rol,user\_email)

VALUES

('director', '0001', 'DIRECTOR', 'director@colegio.com'),

('subdirector', '0002', 'SUBDIRECTOR', 'subdirector@colegio.com'),

('administrativo', '0003', 'ADMINISTRATIVO', 'administrativo@colegio.com');

-- Insertar en PRECIO

INSERT INTO PRECIO (mensualidad)

VALUES

(400.00);

-- Insertar Anios

DECLARE @anio INT = 2023;

WHILE @anio <= 2024

BEGIN

DECLARE @anio\_inicio DATE = DATEADD(DAY, (9 - DATEPART(WEEKDAY, DATEFROMPARTS(@anio, 3, 1))) % 7, DATEFROMPARTS(@anio, 3, 1));

DECLARE @anio\_fin DATE = DATEADD(DAY, (5 - DATEPART(WEEKDAY, DATEFROMPARTS(@anio, 12, 1)) + 7) % 7 + 7, DATEFROMPARTS(@anio, 12, 1));

INSERT INTO ANIO (anio\_id, anio\_inicio, anio\_fin)

VALUES

(@anio, @anio\_inicio, @anio\_fin);

SET @anio = @anio + 1;

END;

-- Insertar Secciones asegurando que sec\_vacantes y sec\_matriculados sumen 30

DECLARE @grad\_id INT = 1;

WHILE @grad\_id <= 6

BEGIN

INSERT INTO SECCION (anio\_id, grad\_id, sec\_nombre, sec\_vacantes, sec\_matriculados)

VALUES

(2023, @grad\_id, 'A', 30, 0),

(2023, @grad\_id, 'B', 30, 0)

SET @grad\_id = @grad\_id + 1;

END;

SET @grad\_id = 1;

WHILE @grad\_id <= 6

BEGIN

INSERT INTO SECCION (anio\_id, grad\_id, sec\_nombre, sec\_vacantes, sec\_matriculados)

VALUES

(2024, @grad\_id, 'A', 30, 0),

(2024, @grad\_id, 'B', 30, 0)

SET @grad\_id = @grad\_id + 1;

END;

-- Insertar Cursos

SET @grad\_id = 1;

-- Cursos para cada grado de Primaria (1 a 6)

WHILE @grad\_id <= 6

BEGIN

INSERT INTO CURSO (grad\_id, cur\_nombre) VALUES

(@grad\_id, 'COMUNICACION'),

(@grad\_id, 'RAZONAMIENTO VERBAL'),

(@grad\_id, 'MATEMATICAS'),

(@grad\_id, 'RAZONAMIENTO MATEMATICO'),

(@grad\_id, 'INGLES'),

(@grad\_id, 'PERSONAL SOCIAL'),

(@grad\_id, 'RELIGION'),

(@grad\_id, 'CIENCIA Y TECNOLOGIA'),

(@grad\_id, 'COMPUTO');

SET @grad\_id = @grad\_id + 1;

END;

-- Insertar Alumnos

INSERT INTO ALUMNO (alu\_apellido, alu\_nombre, alu\_direccion, alu\_telefono)

VALUES

('PEREZ GUTIERREZ', 'JUAN CARLOS', 'Av. Siempre Viva 123', '984731256'),

('GOMEZ RAMIREZ', 'MARIA', 'Calle Falsa 456', '972643851'),

('LOPEZ FERNANDEZ', 'CARLOS ALBERTO', 'Jr. Primavera 789', '953172684'),

('RODRIGUEZ SALAZAR', 'ANA LUCIA', 'Av. Los Rosales 321', '968234715'),

('MARTINEZ SOTO', 'JOSE', 'Calle Los Olivos 654', '987415263'),

('GARCIA HERNANDEZ', 'LUIS MIGUEL', 'Jr. Lima 123', '945362178'),

('HERNANDEZ PEREIRA', 'CLAUDIA', 'Av. Tupac Amaru 456', '962487315'),

('TORRES QUISPE', 'ROSA MARIA', 'Jr. Ayacucho 789', '976543218'),

('SANCHEZ RIVERA', 'PEDRO', 'Av. Grau 321', '953267481'),

('DIAZ ROMERO', 'ELENA PATRICIA', 'Calle Central 654', '987354621');

-- Insertar Empleados

INSERT INTO EMPLEADO (emp\_apellido, emp\_nombre, emp\_direccion, emp\_email, emp\_usuario, emp\_clave)

VALUES

('GOMEZ PEREZ', 'CARLOS ALBERTO', 'Av. Los Cedros 145', 'carlos@gcolegio.com', 'cgomez', '92837465'),

('LOPEZ DIAZ', 'ANA MARIA', 'Jr. La Esperanza 231', 'ana@colegio.com', 'alopez', '73529184'),

('MARTINEZ LOPEZ', 'PEDRO', 'Calle Los Tulipanes 678', 'pedro@colegio.com', 'pmartinez', '18273645'),

('FERNANDEZ CASTRO', 'LUCIA CAROLINA', 'Av. Los Pinos 321', 'lucia@colegio.com', 'lfernandez', '64739281'),

('SANCHEZ RAMIREZ', 'JORGE', 'Jr. Los Rosales 555', 'jorge@colegio.com', 'jsanchez', '53928471');

-- Insertar Profesores con nombres y apellidos en mayúsculas y DNIs de 8 dígitos

INSERT INTO PROFESOR (prof\_apellido, prof\_nombre, prof\_dni, prof\_direccion, prof\_email, prof\_telefono)

VALUES

('RAMIREZ GUTIERREZ', 'ALBERTO', '72456391', 'Calle Los Sauces 123', 'ramirez.alberto@colegio.com', '987643201'),

('QUISPE RODRIGUEZ', 'MARTA ELENA', '81234567', 'Av. Las Flores 456', 'quispe.marta@colegio.com', '912354678'),

('CASTRO TORRES', 'JULIA PATRICIA', '76543210', 'Jr. Pinos 789', 'castro.julia@colegio.com', '945672389'),

('SALAZAR GOMEZ', 'FERNANDO JOSE', '63527182', 'Av. San Juan 321', 'salazar.fernando@colegio.com', '976823491'),

('ROJAS VASQUEZ', 'INES', '85341297', 'Calle Sol 654', 'rojas.ines@colegio.com', '984561230'),

('MENDOZA SANCHEZ', 'PABLO EDUARDO', '72458963', 'Jr. Lirios 123', 'mendoza.pablo@colegio.com', '987345621'),

('PAREDES HERNANDEZ', 'LUZ MARINA', '83725491', 'Av. Olivos 456', 'paredes.luz@colegio.com', '944567283'),

('CHAVEZ REYES', 'ANDRES', '61235478', 'Jr. Palmeras 789', 'chavez.andres@colegio.com', '972345678'),

('VEGA RIVERA', 'CARMEN JOSEFINA', '79243865', 'Calle Los Alamos 321', 'vega.carmen@colegio.com', '953217846'),

('ORTEGA VALDEZ', 'LUIS', '63421587', 'Av. Nogales 654', 'ortega.luis@colegio.com', '987312549'),

('NAVARRO QUINTANA', 'SILVIA TERESA', '81243657', 'Calle Azucenas 123', 'navarro.silvia@colegio.com', '961243578'),

('REYES PEREZ', 'RICARDO ANTONIO', '72536419', 'Av. Jazmines 456', 'reyes.ricardo@colegio.com', '975312486'),

('GUTIERREZ DIAZ', 'ANGELA', '84321765', 'Jr. Rosas 789', 'gutierrez.angela@colegio.com', '982435617'),

('VALDEZ HERRERA', 'MARIO ENRIQUE', '76231458', 'Av. Sauce 321', 'valdez.mario@colegio.com', '956213487'),

('ESCOBAR RAMOS', 'PATRICIA', '69231485', 'Calle Lima 654', 'escobar.patricia@colegio.com', '968245316'),

('RIOS MARTINEZ', 'HUGO CESAR', '78245639', 'Av. Central 123', 'rios.hugo@colegio.com', '943215867'),

('FIGUEROA CARRILLO', 'CELIA', '84123765', 'Jr. Esperanza 456', 'figueroa.celia@colegio.com', '967432815'),

('VILLANUEVA TORRES', 'RAUL FERNANDO', '65421378', 'Calle Nueva 789', 'villanueva.raul@colegio.com', '953278416'),

('ACOSTA ROSALES', 'BLANCA', '73241568', 'Av. Los Heroes 321', 'acosta.blanca@colegio.com', '986345127'),

('RUIZ PEREZ', 'EDUARDO ALONSO', '82134675', 'Jr. Alameda 654', 'ruiz.eduardo@colegio.com', '958214637');

-- Asegúrate de que estos registros existen antes de activar el trigger

INSERT INTO TIPO\_PAGO (tipo\_pago\_nombre)

VALUES

('Matricula'),

('Pension');

-- Insertar Matrículas (asegurando la referencia de sec\_id, alu\_id y emp\_id existentes)

INSERT INTO MATRICULA (alu\_id, sec\_id, emp\_id, mat\_tipo, mat\_fecha, mat\_estado)

VALUES

(1, 1, 1, 'REGULAR', '2023-03-01', 1),

(2, 2, 1, 'REGULAR', '2023-03-01', 1),

(3, 3, 1, 'REGULAR', '2023-03-01', 0);

GO

UPDATE SECCION

SET sec\_matriculados = sec\_matriculados + 1

WHERE sec\_id IN (1, 2, 3);

UPDATE SECCION

SET sec\_vacantes = sec\_vacantes - 1

WHERE sec\_id IN (1, 2, 3);

-- Paso 2: Generar Cronograma de Pagos para cada matrícula recién insertada

DECLARE @mat\_id INT;

DECLARE @anio INT = 2023; -- Usar el año actual para los cronogramas

DECLARE mat\_cursor CURSOR FOR

SELECT mat\_id FROM MATRICULA;

OPEN mat\_cursor;

FETCH NEXT FROM mat\_cursor INTO @mat\_id;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Insertar el cronograma de pago de matrícula (tipo\_pago\_id = 1) a finales de febrero

INSERT INTO CRONOGRAMA\_PAGO (mat\_id, tipo\_pago\_id, cro\_monto, cro\_fecha\_prog)

VALUES (@mat\_id, 1,400, EOMONTH(DATEFROMPARTS(@anio, 2, 1)));

-- Insertar cronogramas de pagos mensuales (tipo\_pago\_id = 2) desde marzo hasta noviembre

DECLARE @mes INT = 3;

WHILE @mes <= 11

BEGIN

-- Insertar cronogramas de marzo a noviembre (fin de cada mes)

INSERT INTO CRONOGRAMA\_PAGO (mat\_id, tipo\_pago\_id, cro\_monto, cro\_fecha\_prog)

VALUES (@mat\_id, 2, 400, EOMONTH(DATEFROMPARTS(@anio, @mes, 1)));

SET @mes = @mes + 1;

END;

-- Insertar el cronograma de diciembre con fecha programada al 15 de diciembre

INSERT INTO CRONOGRAMA\_PAGO (mat\_id, tipo\_pago\_id, cro\_monto, cro\_fecha\_prog)

VALUES (@mat\_id, 2, 400, DATEFROMPARTS(@anio, 12, 15));

-- Pasar a la siguiente matrícula en el cursor

FETCH NEXT FROM mat\_cursor INTO @mat\_id;

END;

CLOSE mat\_cursor;

DEALLOCATE mat\_cursor;

GO

INSERT INTO PAGO(cro\_id, emp\_id, pag\_pension, pag\_importe, pag\_fecha, pag\_fecha\_prog) VALUES

(1,1,0,400,'2023-02-28','2023-02-28'),

(12,1,0,400,'2023-02-28','2023-02-28'),

(23,1,0,400,'2023-02-28','2023-02-28'),

(2,1,1,400,'2023-03-31','2023-03-31'),

(13,1,1,400,'2023-03-31','2023-03-31'),

(24,1,1,400,'2023-03-31','2023-03-31'),

(3,1,2,400,'2023-04-30','2023-04-30'),

(14,1,2,400,'2023-04-30','2023-04-30'),

(25,1,2,400,'2023-04-30','2023-04-30');

-- Inserciones simplificadas en SECCION\_CURSO

INSERT INTO SECCION\_CURSO (sec\_id, cur\_id, prof\_id) VALUES

(1, 1, 1), (1, 2, 1), (1, 3, 5), (1, 4, 5), (1, 5, 9), (1, 6, 11), (1, 7, 11), (1, 8, 15), (1, 9, 18),

(2, 1, 1), (2, 2, 1), (2, 3, 5), (2, 4, 5), (2, 5, 9), (2, 6, 11), (2, 7, 11), (2, 8, 15), (2, 9, 18),

(3, 10, 1), (3, 11, 1), (3, 12, 5), (3, 13, 5), (3, 14, 9), (3, 15, 11), (3, 16, 11), (3, 17, 15), (3, 18, 18),

(4, 10, 2), (4, 11, 2), (4, 12, 6), (4, 13, 6), (4, 14, 9), (4, 15, 12), (4, 16, 12), (4, 17, 15), (4, 18, 18),

(5, 19, 2), (5, 20, 2), (5, 21, 6), (5, 22, 6), (5, 23, 9), (5, 24, 12), (5, 25, 12), (5, 26, 16), (5, 27, 18),

(6, 19, 2), (6, 20, 2), (6, 21, 6), (6, 22, 6), (6, 23, 9), (6, 24, 12), (6, 25, 12), (6, 26, 16), (6, 27, 18),

(7, 28, 3), (7, 29, 3), (7, 30, 7), (7, 31, 7), (7, 32, 10), (7, 33, 13), (7, 34, 13), (7, 35, 16), (7, 36, 19),

(8, 28, 3), (8, 29, 3), (8, 30, 7), (8, 31, 7), (8, 32, 10), (8, 33, 13), (8, 34, 13), (8, 35, 16), (8, 36, 19),

(9, 37, 3), (9, 38, 3), (9, 39, 7), (9, 40, 7), (9, 41, 10), (9, 42, 13), (9, 43, 13), (9, 44, 17), (9, 45, 19),

(10, 37, 4), (10, 38, 4), (10, 39, 8), (10, 40, 8), (10, 41, 10), (10, 42, 14), (10, 43, 14), (10, 44, 17), (10, 45, 19),

(11, 46, 4), (11, 47, 4), (11, 48, 8), (11, 49, 8), (11, 50, 10), (11, 51, 14), (11, 52, 14), (11, 53, 17), (11, 54, 19),

(12, 46, 4), (12, 47, 4), (12, 48, 8), (12, 49, 8), (12, 50, 10), (12, 51, 14), (12, 52, 14), (12, 53, 17), (12, 54, 19);

-- Inserciones de horarios

INSERT INTO HORARIO(asig\_id, hor\_dia, hor\_inicio, hor\_fin) VALUES

(1,'MARTES','10:30','11:15'),(1,'MARTES','11:15','12:00'),(1,'MIERCOLES','10:30','11:15'),(1,'MIERCOLES','11:15','12:00'),

(2,'MIERCOLES','12:15','13:00'),(2,'MIERCOLES','13:00','13:45'),(2,'MIERCOLES','13:45','14:30'),(2,'JUEVES','08:00','08:45'),

(3,'MARTES','12:15','13:00'),(3,'MARTES','13:00','13:45'),(3,'MARTES','13:45','14:30'),(3,'JUEVES','12:15','13:00'),(3,'JUEVES','13:00','13:45'),(3,'JUEVES','13:45','14:30'),

(4,'JUEVES','08:45','09:30'),(4,'JUEVES','09:30','10:15'),(4,'JUEVES','10:30','11:15'),(4,'JUEVES','11:15','12:00'),

(5,'LUNES','08:00','08:45'),(5,'LUNES','08:45','09:30'),(5,'LUNES','09:30','10:15'),(5,'LUNES','10:30','11:15'),

(6,'VIERNES','08:00','08:45'),(6,'VIERNES','08:45','09:30'),(6,'VIERNES','09:30','10:15'),(6,'VIERNES','10:30','11:15'),

(7,'VIERNES','11:15','12:00'),(7,'VIERNES','12:15','13:00'),(7,'VIERNES','13:00','13:45'),(7,'VIERNES','13:45','14:30'),

(8,'MARTES','08:00','08:45'),(8,'MARTES','08:45','09:30'),(8,'MARTES','09:30','10:15'),(8,'MIERCOLES','08:00','08:45'),(8,'MIERCOLES','08:45','09:30'),(8,'MIERCOLES','09:30','10:15'),

(9,'LUNES','11:15','12:00'),(9,'LUNES','12:15','13:00'),(9,'LUNES','13:00','13:45'),(9,'LUNES','13:45','14:30'),

(10,'LUNES','08:00','08:45'),(10,'LUNES','08:45','09:30'),(10,'LUNES','09:30','10:15'),(10,'LUNES','10:30','11:15'),

(11,'LUNES','11:15','12:00'),(11,'LUNES','12:15','13:00'),(11,'LUNES','13:00','13:45'),(11,'LUNES','13:45','14:30'),

(12,'MIERCOLES','11:15','12:00'),(12,'MIERCOLES','12:15','13:00'),(12,'MIERCOLES','13:00','13:45'),(12,'VIERNES','12:15','13:00'),(12,'VIERNES','13:00','13:45'),(12,'VIERNES','13:45','14:30'),

(13,'MIERCOLES','08:00','08:45'),(13,'MIERCOLES','08:45','09:30'),(13,'MIERCOLES','09:30','10:15'),(13,'MIERCOLES','10:30','11:15'),

(14,'MARTES','08:00','08:45'),(14,'MARTES','08:45','09:30'),(14,'MARTES','09:30','10:15'),(14,'MARTES','10:30','11:15'),

(15,'MIERCOLES','13:45','14:30'),(15,'JUEVES','08:00','08:45'),(15,'JUEVES','08:45','09:30'),(15,'JUEVES','09:30','10:15'),

(16,'JUEVES','10:30','11:15'),(16,'JUEVES','11:15','12:00'),(16,'JUEVES','12:15','13:00'),(16,'JUEVES','13:00','13:45'),

(17,'JUEVES','13:45','14:30'),(17,'VIERNES','08:00','08:45'),(17,'VIERNES','08:45','09:30'),(17,'VIERNES','09:30','10:15'),(17,'VIERNES','10:30','11:15'),(17,'VIERNES','11:15','12:00'),

(18,'MARTES','11:15','12:00'),(18,'MARTES','12:15','13:00'),(18,'MARTES','13:00','13:45'),(18,'MARTES','13:45','14:30'),

(19,'VIERNES','11:15','12:00'),(19,'VIERNES','12:15','13:00'),(19,'VIERNES','13:00','13:45'),(19,'VIERNES','13:45','14:30'),

(20,'JUEVES','11:15','12:00'),(20,'JUEVES','12:15','13:00'),(20,'JUEVES','13:00','13:45'),(20,'JUEVES','13:45','14:30'),

(21,'LUNES','08:00','08:45'),(21,'LUNES','08:45','09:30'),(21,'LUNES','09:30','10:15'),(21,'MARTES','08:00','08:45'),(21,'MARTES','08:45','09:30'),(21,'MARTES','09:30','10:15'),

(22,'VIERNES','08:00','08:45'),(22,'VIERNES','08:45','09:30'),(22,'VIERNES','09:30','10:15'),(22,'VIERNES','10:30','11:15'),

(23,'MIERCOLES','08:00','08:45'),(23,'MIERCOLES','08:45','09:30'),(23,'MIERCOLES','09:30','10:15'),(23,'MIERCOLES','10:30','11:15'),

(24,'LUNES','10:30','11:15'),(24,'LUNES','11:15','12:00'),(24,'LUNES','12:15','13:00'),(24,'LUNES','13:00','13:45'),

(25,'LUNES','13:45','14:30'),(25,'MARTES','10:30','11:15'),(25,'MARTES','11:15','12:00'),(25,'MIERCOLES','11:15','12:00'),

(26,'MARTES','12:15','13:00'),(26,'MARTES','13:00','13:45'),(26,'MARTES','13:45','14:30'),(26,'JUEVES','08:45','09:30'),(26,'JUEVES','09:30','10:15'),(26,'JUEVES','10:30','11:15'),

(27,'MIERCOLES','12:15','13:00'),(27,'MIERCOLES','13:00','13:45'),(27,'MIERCOLES','13:45','14:30'),(27,'JUEVES','08:00','08:45'),

(28,'LUNES','11:15','12:00'),(28,'LUNES','12:15','13:00'),(28,'LUNES','13:00','13:45'),(28,'LUNES','13:45','14:30'),

(29,'LUNES','08:00','08:45'),(29,'LUNES','08:45','09:30'),(29,'LUNES','09:30','10:15'),(29,'LUNES','10:30','11:15'),

(30,'MARTES','12:15','13:00'),(30,'MARTES','13:00','13:45'),(30,'MARTES','13:45','14:30'),(30,'MIERCOLES','08:00','08:45'),(30,'MIERCOLES','08:45','09:30'),(30,'VIERNES','08:00','08:45'),

(31,'MARTES','10:30','11:15'),(31,'MARTES','11:15','12:00'),(31,'MIERCOLES','10:30','11:15'),(31,'MIERCOLES','11:15','12:00'),

(32,'JUEVES','08:00','08:45'),(32,'JUEVES','08:45','09:30'),(32,'JUEVES','09:30','10:15'),(32,'JUEVES','10:30','11:15'),

(33,'VIERNES','08:45','09:30'),(33,'VIERNES','09:30','10:15'),(33,'VIERNES','10:30','11:15'),(33,'VIERNES','11:15','12:00'),

(34,'MARTES','08:00','08:45'),(34,'MARTES','08:45','09:30'),(34,'MARTES','09:30','10:15'),(34,'MIERCOLES','09:30','10:15'),

(35,'MIERCOLES','12:15','13:00'),(35,'MIERCOLES','13:00','13:45'),(35,'MIERCOLES','13:45','14:30'),(35,'VIERNES','12:15','13:00'),(35,'VIERNES','13:00','13:45'),(35,'VIERNES','13:45','14:30'),

(36,'JUEVES','11:15','12:00'),(36,'JUEVES','12:15','13:00'),(36,'JUEVES','13:00','13:45'),(36,'JUEVES','13:45','14:30'),

(37,'MARTES','11:15','12:00'),(37,'MARTES','12:15','13:00'),(37,'MARTES','13:00','13:45'),(37,'MARTES','13:45','14:30'),

(38,'MARTES','08:00','08:45'),(38,'MARTES','08:45','09:30'),(38,'MARTES','09:30','10:15'),(38,'MARTES','10:30','11:15'),

(39,'LUNES','08:00','08:45'),(39,'LUNES','08:45','09:30'),(39,'LUNES','09:30','10:15'),(39,'JUEVES','08:00','08:45'),(39,'JUEVES','08:45','09:30'),(39,'JUEVES','09:30','10:15'),

(40,'LUNES','10:30','11:15'),(40,'LUNES','11:15','12:00'),(40,'JUEVES','10:30','11:15'),(40,'JUEVES','11:15','12:00'),

(41,'VIERNES','08:00','08:45'),(41,'VIERNES','08:45','09:30'),(41,'VIERNES','09:30','10:15'),(41,'VIERNES','10:30','11:15'),

(42,'MIERCOLES','10:30','11:15'),(42,'MIERCOLES','11:15','12:00'),(42,'MIERCOLES','12:15','13:00'),(42,'MIERCOLES','13:00','13:45'),

(43,'MIERCOLES','13:45','14:30'),(43,'JUEVES','12:15','13:00'),(43,'JUEVES','13:00','13:45'),(43,'JUEVES','13:45','14:30'),

(44,'LUNES','12:15','13:00'),(44,'LUNES','13:00','13:45'),(44,'LUNES','13:45','14:30'),(44,'MIERCOLES','08:00','08:45'),(44,'MIERCOLES','08:45','09:30'),(44,'MIERCOLES','09:30','10:15'),

(45,'VIERNES','11:15','12:00'),(45,'VIERNES','12:15','13:00'),(45,'VIERNES','13:00','13:45'),(45,'VIERNES','13:45','14:30'),

(46,'MIERCOLES','11:15','12:00'),(46,'MIERCOLES','12:15','13:00'),(46,'MIERCOLES','13:00','13:45'),(46,'MIERCOLES','13:45','14:30'),

(47,'MIERCOLES','08:00','08:45'),(47,'MIERCOLES','08:45','09:30'),(47,'MIERCOLES','09:30','10:15'),(47,'MIERCOLES','10:30','11:15'),

(48,'JUEVES','12:15','13:00'),(48,'JUEVES','13:00','13:45'),(48,'JUEVES','13:45','14:30'),(48,'VIERNES','08:45','09:30'),(48,'VIERNES','09:30','10:15'),(48,'VIERNES','10:30','11:15'),

(49,'VIERNES','11:15','12:00'),(49,'VIERNES','12:15','13:00'),(49,'VIERNES','13:00','13:45'),(49,'VIERNES','13:45','14:30'),

(50,'LUNES','11:15','12:00'),(50,'LUNES','12:15','13:00'),(50,'LUNES','13:00','13:45'),(50,'LUNES','13:45','14:30'),

(51,'MARTES','10:30','11:15'),(51,'MARTES','11:15','12:00'),(51,'JUEVES','10:30','11:15'),(51,'JUEVES','11:15','12:00'),

(52,'MARTES','12:15','13:00'),(52,'MARTES','13:00','13:45'),(52,'MARTES','13:45','14:30'),(52,'VIERNES','08:00','08:45'),

(53,'MARTES','08:00','08:45'),(53,'MARTES','08:45','09:30'),(53,'MARTES','09:30','10:15'),(53,'JUEVES','08:00','08:45'),(53,'JUEVES','08:45','09:30'),(53,'JUEVES','09:30','10:15'),

(54,'LUNES','08:00','08:45'),(54,'LUNES','08:45','09:30'),(54,'LUNES','09:30','10:15'),(54,'LUNES','10:30','11:15'),

(55,'MARTES','10:30','11:15'),(55,'MARTES','11:15','12:00'),(55,'MIERCOLES','10:30','11:15'),(55,'MIERCOLES','11:15','12:00'),

(56,'MIERCOLES','12:15','13:00'),(56,'MIERCOLES','13:00','13:45'),(56,'MIERCOLES','13:45','14:30'),(56,'JUEVES','08:00','08:45'),

(57,'MARTES','12:15','13:00'),(57,'MARTES','13:00','13:45'),(57,'MARTES','13:45','14:30'),(57,'JUEVES','12:15','13:00'),(57,'JUEVES','13:00','13:45'),(57,'JUEVES','13:45','14:30'),

(58,'JUEVES','08:45','09:30'),(58,'JUEVES','09:30','10:15'),(58,'JUEVES','10:30','11:15'),(58,'JUEVES','11:15','12:00'),

(59,'LUNES','08:00','08:45'),(59,'LUNES','08:45','09:30'),(59,'LUNES','09:30','10:15'),(59,'LUNES','10:30','11:15'),

(60,'VIERNES','08:00','08:45'),(60,'VIERNES','08:45','09:30'),(60,'VIERNES','09:30','10:15'),(60,'VIERNES','10:30','11:15'),

(61,'VIERNES','11:15','12:00'),(61,'VIERNES','12:15','13:00'),(61,'VIERNES','13:00','13:45'),(61,'VIERNES','13:45','14:30'),

(62,'MARTES','08:00','08:45'),(62,'MARTES','08:45','09:30'),(62,'MARTES','09:30','10:15'),(62,'MIERCOLES','08:00','08:45'),(62,'MIERCOLES','08:45','09:30'),(62,'MIERCOLES','09:30','10:15'),

(63,'LUNES','11:15','12:00'),(63,'LUNES','12:15','13:00'),(63,'LUNES','13:00','13:45'),(63,'LUNES','13:45','14:30'),

(64,'LUNES','08:00','08:45'),(64,'LUNES','08:45','09:30'),(64,'LUNES','09:30','10:15'),(64,'LUNES','10:30','11:15'),

(65,'LUNES','11:15','12:00'),(65,'LUNES','12:15','13:00'),(65,'LUNES','13:00','13:45'),(65,'LUNES','13:45','14:30'),

(66,'MIERCOLES','11:15','12:00'),(66,'MIERCOLES','12:15','13:00'),(66,'MIERCOLES','13:00','13:45'),(66,'VIERNES','12:15','13:00'),(66,'VIERNES','13:00','13:45'),(66,'VIERNES','13:45','14:30'),

(67,'MIERCOLES','08:00','08:45'),(67,'MIERCOLES','08:45','09:30'),(67,'MIERCOLES','09:30','10:15'),(67,'MIERCOLES','10:30','11:15'),

(68,'MARTES','08:00','08:45'),(68,'MARTES','08:45','09:30'),(68,'MARTES','09:30','10:15'),(68,'MARTES','10:30','11:15'),

(69,'MIERCOLES','13:45','14:30'),(69,'JUEVES','08:00','08:45'),(69,'JUEVES','08:45','09:30'),(69,'JUEVES','09:30','10:15'),

(70,'JUEVES','10:30','11:15'),(70,'JUEVES','11:15','12:00'),(70,'JUEVES','12:15','13:00'),(70,'JUEVES','13:00','13:45'),

(71,'JUEVES','13:45','14:30'),(71,'VIERNES','08:00','08:45'),(71,'VIERNES','08:45','09:30'),(71,'VIERNES','09:30','10:15'),(71,'VIERNES','10:30','11:15'),(71,'VIERNES','11:15','12:00'),

(72,'MARTES','11:15','12:00'),(72,'MARTES','12:15','13:00'),(72,'MARTES','13:00','13:45'),(72,'MARTES','13:45','14:30'),

(73,'VIERNES','11:15','12:00'),(73,'VIERNES','12:15','13:00'),(73,'VIERNES','13:00','13:45'),(73,'VIERNES','13:45','14:30'),

(74,'JUEVES','11:15','12:00'),(74,'JUEVES','12:15','13:00'),(74,'JUEVES','13:00','13:45'),(74,'JUEVES','13:45','14:30'),

(75,'LUNES','08:00','08:45'),(75,'LUNES','08:45','09:30'),(75,'LUNES','09:30','10:15'),(75,'MARTES','08:00','08:45'),(75,'MARTES','08:45','09:30'),(75,'MARTES','09:30','10:15'),

(76,'VIERNES','08:00','08:45'),(76,'VIERNES','08:45','09:30'),(76,'VIERNES','09:30','10:15'),(76,'VIERNES','10:30','11:15'),

(77,'MIERCOLES','08:00','08:45'),(77,'MIERCOLES','08:45','09:30'),(77,'MIERCOLES','09:30','10:15'),(77,'MIERCOLES','10:30','11:15'),

(78,'LUNES','10:30','11:15'),(78,'LUNES','11:15','12:00'),(78,'LUNES','12:15','13:00'),(78,'LUNES','13:00','13:45'),

(79,'LUNES','13:45','14:30'),(79,'MARTES','10:30','11:15'),(79,'MARTES','11:15','12:00'),(79,'MIERCOLES','11:15','12:00'),

(80,'LUNES','13:00','13:45'),(80,'LUNES','13:45','14:30'),(80,'LUNES','08:00','08:45'),(80,'LUNES','08:45','09:30'),(80,'LUNES','09:30','10:15'),(80,'LUNES','10:30','11:15'),

(81,'MIERCOLES','12:15','13:00'),(81,'MIERCOLES','13:00','13:45'),(81,'MIERCOLES','13:45','14:30'),(81,'JUEVES','08:00','08:45'),

(82,'LUNES','11:15','12:00'),(82,'LUNES','12:15','13:00'),(82,'LUNES','13:00','13:45'),(82,'LUNES','13:45','14:30'),

(83,'LUNES','08:00','08:45'),(83,'LUNES','08:45','09:30'),(83,'LUNES','09:30','10:15'),(83,'LUNES','10:30','11:15'),

(84,'MARTES','12:15','13:00'),(84,'MARTES','13:00','13:45'),(84,'MARTES','13:45','14:30'),(84,'MIERCOLES','08:00','08:45'),(84,'MIERCOLES','08:45','09:30'),(84,'VIERNES','08:00','08:45'),

(85,'MARTES','10:30','11:15'),(85,'MARTES','11:15','12:00'),(85,'MIERCOLES','10:30','11:15'),(85,'MIERCOLES','11:15','12:00'),

(86,'JUEVES','08:00','08:45'),(86,'JUEVES','08:45','09:30'),(86,'JUEVES','09:30','10:15'),(86,'JUEVES','10:30','11:15'),

(87,'VIERNES','08:45','09:30'),(87,'VIERNES','09:30','10:15'),(87,'VIERNES','10:30','11:15'),(87,'VIERNES','11:15','12:00'),

(88,'MARTES','08:00','08:45'),(88,'MARTES','08:45','09:30'),(88,'MARTES','09:30','10:15'),(88,'MIERCOLES','09:30','10:15'),

(89,'MIERCOLES','12:15','13:00'),(89,'MIERCOLES','13:00','13:45'),(89,'MIERCOLES','13:45','14:30'),(89,'VIERNES','12:15','13:00'),(89,'VIERNES','13:00','13:45'),(89,'VIERNES','13:45','14:30'),

(90,'JUEVES','11:15','12:00'),(90,'JUEVES','12:15','13:00'),(90,'JUEVES','13:00','13:45'),(90,'JUEVES','13:45','14:30'),

(91,'MARTES','11:15','12:00'),(91,'MARTES','12:15','13:00'),(91,'MARTES','13:00','13:45'),(91,'MARTES','13:45','14:30'),

(92,'MARTES','08:00','08:45'),(92,'MARTES','08:45','09:30'),(92,'MARTES','09:30','10:15'),(92,'MARTES','10:30','11:15'),

(93,'LUNES','08:00','08:45'),(93,'LUNES','08:45','09:30'),(93,'LUNES','09:30','10:15'),(93,'JUEVES','08:00','08:45'),(93,'JUEVES','08:45','09:30'),(93,'JUEVES','09:30','10:15'),

(94,'LUNES','10:30','11:15'),(94,'LUNES','11:15','12:00'),(94,'JUEVES','10:30','11:15'),(94,'JUEVES','11:15','12:00'),

(95,'VIERNES','08:00','08:45'),(95,'VIERNES','08:45','09:30'),(95,'VIERNES','09:30','10:15'),(95,'VIERNES','10:30','11:15'),

(96,'MIERCOLES','10:30','11:15'),(96,'MIERCOLES','11:15','12:00'),(96,'MIERCOLES','12:15','13:00'),(96,'MIERCOLES','13:00','13:45'),

(97,'MIERCOLES','13:45','14:30'),(97,'JUEVES','12:15','13:00'),(97,'JUEVES','13:00','13:45'),(97,'JUEVES','13:45','14:30'),

(98,'LUNES','12:15','13:00'),(98,'LUNES','13:00','13:45'),(98,'LUNES','13:45','14:30'),(98,'MIERCOLES','08:00','08:45'),(98,'MIERCOLES','08:45','09:30'),(98,'MIERCOLES','09:30','10:15'),

(99,'VIERNES','11:15','12:00'),(99,'VIERNES','12:15','13:00'),(99,'VIERNES','13:00','13:45'),(99,'VIERNES','13:45','14:30'),

(100,'MIERCOLES','11:15','12:00'),(100,'MIERCOLES','12:15','13:00'),(100,'MIERCOLES','13:00','13:45'),(100,'MIERCOLES','13:45','14:30'),

(101,'MIERCOLES','08:00','08:45'),(101,'MIERCOLES','08:45','09:30'),(101,'MIERCOLES','09:30','10:15'),(101,'MIERCOLES','10:30','11:15'),

(102,'JUEVES','12:15','13:00'),(102,'JUEVES','13:00','13:45'),(102,'JUEVES','13:45','14:30'),(102,'VIERNES','08:45','09:30'),(102,'VIERNES','09:30','10:15'),(102,'VIERNES','10:30','11:15'),

(103,'VIERNES','11:15','12:00'),(103,'VIERNES','12:15','13:00'),(103,'VIERNES','13:00','13:45'),(103,'VIERNES','13:45','14:30'),

(104,'LUNES','11:15','12:00'),(104,'LUNES','12:15','13:00'),(104,'LUNES','13:00','13:45'),(104,'LUNES','13:45','14:30'),

(105,'MARTES','10:30','11:15'),(105,'MARTES','11:15','12:00'),(105,'JUEVES','10:30','11:15'),(105,'JUEVES','11:15','12:00'),

(106,'MARTES','12:15','13:00'),(106,'MARTES','13:00','13:45'),(106,'MARTES','13:45','14:30'),(106,'VIERNES','08:00','08:45'),

(107,'MARTES','08:00','08:45'),(107,'MARTES','08:45','09:30'),(107,'MARTES','09:30','10:15'),(107,'JUEVES','08:00','08:45'),(107,'JUEVES','08:45','09:30'),(107,'JUEVES','09:30','10:15'),

(108,'LUNES','08:00','08:45'),(108,'LUNES','08:45','09:30'),(108,'LUNES','09:30','10:15'),(108,'LUNES','10:30','11:15');

-- Inserciones de notas

INSERT INTO NOTA (alu\_id,asig\_id,not\_bimestre1) VALUES

(1,1,16),

(1,2,17),

(1,3,15),

(1,4,18),

(1,5,14),

(1,6,13),

(1,7,17),

(1,8,12),

(1,9,16),

(2,10,14),

(2,11,17),

(2,12,16),

(2,13,13),

(2,14,15),

(2,15,11),

(2,16,15),

(2,17,14),

(2,18,13),

(3,19,12),

(3,20,15),

(3,21,13),

(3,22,15),

(3,23,11),

(3,24,9),

(3,25,14),

(3,26,15),

(3,27,11);

# CÓDIGO FUENTE

## SERVICIO DE MATRICULA\_CU01 (MatriculaService)

|  |
| --- |
| @Service public class MatriculaService {  @Autowired  private JdbcTemplate jdbcTemplate;  @Transactional(propagation = Propagation.REQUIRES\_NEW, rollbackFor = Exception.class)  public MatriculaDto proc\_Matricula(MatriculaDto bean) {  //DATOS DTO   int alu\_id=bean.getAlu\_id();  int sec\_id=bean.getSec\_id();  int emp\_id=bean.getEmp\_id();  String mat\_tipo=bean.getMat\_tipo();  String mat\_fecha=bean.getMat\_fecha();  String mat\_estado="1";  //auxiliar   double costo;    // Validaciones    //validar seccion  validarSeccion(sec\_id);    //validar si existen vacantes en la seccion  validarVacantesSeccion(sec\_id);    //validando alumno  validarAlumno(alu\_id, sec\_id);    //validar empleado  validarEmpleado(emp\_id);    //validar tipo sea beca , regular  mat\_tipo=validarTipo(mat\_tipo);  bean.setMat\_tipo(mat\_tipo);      //valdiar fecha ingresada en formato dd/mm/aa  bean.setMat\_fecha( validarFecha(mat\_fecha));  mat\_fecha=validarFecha(mat\_fecha);    // VALIDAR ANIO DE MATRICULA   validarAnio(sec\_id, mat\_fecha);      //validar seccion   int val=validarAlumnoNuevo(alu\_id);  if(val!=0) {  validarSeccionAlumno(alu\_id, sec\_id);   validarMatriculaAlumno(alu\_id, sec\_id);  }       //Actualizando estado  bean.setMat\_estado(mat\_estado);  //Registro de la tabla Matricula  registrarMatricula(alu\_id, sec\_id, emp\_id, mat\_tipo, mat\_fecha, mat\_estado);  //Actualizando vacantes   actualizarVacantes(sec\_id);    //Calculando costo x tipo   costo =costo\_Tipo(mat\_tipo);     //generando cronograma de pagos para el alumno  registrarCronograma(sec\_id, mat\_fecha, costo);     return bean;  }    @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarVacantesSeccion(int sec\_id) {  // Consulta SQL para obtener la diferencia entre vacantes y matriculados  String sql = """  SELECT sec\_vacantes - sec\_matriculados AS vac\_dis  FROM SECCION  WHERE sec\_id = ?;  """;   // Obtener la diferencia entre vacantes y matriculados  Integer vacantesDisponibles = jdbcTemplate.queryForObject(sql, Integer.class, sec\_id);   // Verificar si hay vacantes disponibles  if (vacantesDisponibles == null || vacantesDisponibles <= 0) {  throw new RuntimeException("No hay vacantes disponibles en esta sección.");  }  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void registrarMatricula(int alu\_id, int sec\_id, int emp\_id, String mat\_tipo,  String mat\_fecha, String mat\_estado) {  String sql = """  INSERT INTO MATRICULA   ([alu\_id], [sec\_id], [emp\_id], [mat\_tipo], [mat\_fecha], [mat\_estado])  VALUES (?, ?, ?, ?, ?, ?)  """;   jdbcTemplate.update(sql, alu\_id, sec\_id, emp\_id, mat\_tipo, mat\_fecha, mat\_estado);  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void actualizarVacantes(int secId) {  String sql = """  UPDATE SECCION  SET sec\_matriculados = sec\_matriculados + 1,  sec\_vacantes = sec\_vacantes - 1  WHERE sec\_id = ?;  """;   int rowsUpdated = jdbcTemplate.update(sql, secId);   if (rowsUpdated != 1) {  throw new RuntimeException("La sección de ID: " + secId + " no existe");  }  }     @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void registrarCronograma(int sec\_id, String mat\_fecha, double costo) {  // Obtener el ID de la última inserción  String sql = """  SELECT @@IDENTITY AS cro\_id  """;  int cro\_id = jdbcTemplate.queryForObject(sql, Integer.class);   // Obtener el año de la matrícula  sql = """  SELECT YEAR(?) AS anio  """;  String anio = jdbcTemplate.queryForObject(sql, String.class, mat\_fecha);   // Procedimiento almacenado para insertar cronograma de pagos  sql = """  DECLARE @matId INT = ?; -- ID de la matrícula  DECLARE @anio INT = ?; -- Año del cronograma  DECLARE @mesInicio INT = 2; -- Mes de inicio (2 = febrero)  DECLARE @mesFin INT = 12; -- Mes de fin (12 = diciembre)  DECLARE @montoMensual DECIMAL(10,2) = ?; -- Monto de los pagos mensuales   -- Insertar cronograma de pago inicial (tipo\_pago\_id = 1) a finales de febrero  INSERT INTO CRONOGRAMA\_PAGO (mat\_id, tipo\_pago\_id, cro\_monto, cro\_fecha\_prog)  VALUES (@matId, 1, @montoMensual, EOMONTH(DATEFROMPARTS(@anio, @mesInicio, 1)));   -- Insertar cronogramas de pagos mensuales (tipo\_pago\_id = 2) desde marzo hasta diciembre  DECLARE @mes INT = @mesInicio + 1;   WHILE @mes <= @mesFin  BEGIN  IF @mes = @mesFin  BEGIN  -- Para el último mes (diciembre), establecer la fecha como el 15  INSERT INTO CRONOGRAMA\_PAGO (mat\_id, tipo\_pago\_id, cro\_monto, cro\_fecha\_prog)  VALUES (@matId, 2, @montoMensual, DATEFROMPARTS(@anio, @mes, 15));  END  ELSE  BEGIN  -- Para los demás meses, establecer la fecha al final del mes  INSERT INTO CRONOGRAMA\_PAGO (mat\_id, tipo\_pago\_id, cro\_monto, cro\_fecha\_prog)  VALUES (@matId, 2, @montoMensual, EOMONTH(DATEFROMPARTS(@anio, @mes, 1)));  END   SET @mes = @mes + 1;  END;  """;   jdbcTemplate.update(sql, cro\_id, anio, costo);  }         @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarAlumno(int alu\_id,int sec\_id) {  String sql = """  SELECT COUNT(\*) cont FROM ALUMNO WHERE alu\_id=?   """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id);  if(cont==0) {  throw new RuntimeException("El alumno no Existe.");  }  sql = """  SELECT COUNT(\*) cont FROM MATRICULA WHERE alu\_id=? AND sec\_id=?   """;  cont = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id,sec\_id);  if(cont!=0) {  throw new RuntimeException("El alumno ya se encuentra matriculado en la seccion programada de Id : "+sec\_id);  }  }  @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarSeccion(int sec\_id) {  String sql = """  SELECT COUNT(\*) cont FROM SECCION WHERE sec\_id=?   """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, sec\_id);  if(cont==0) {  throw new RuntimeException("La Seccion no existe ");  }  }  @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarEmpleado(int emp\_id) {  String sql = """  SELECT COUNT(\*) cont FROM EMPLEADO WHERE emp\_id=?   """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, emp\_id);  if(cont==0) {  throw new RuntimeException("El empleado no existe");  }  }  @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private String validarTipo(String mat\_tipo) {  // Convertir a mayúsculas y eliminar espacios en blanco al inicio y final  String tipoMayuscula = mat\_tipo.trim().toUpperCase();   // Validar si el tipo es válido  if (!Arrays.asList("BECA", "REGULAR", "MEDIABECA").contains(tipoMayuscula)) {  throw new IllegalArgumentException("El tipo de matrícula debe ser BECA, REGULAR o MEDIABECA");  }     return tipoMayuscula;  }    @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private String validarFecha(String mat\_fecha) {  String fecha\_format;   // Eliminar espacios en blanco al inicio y al final  String fechaLimpia = mat\_fecha.trim();   // Expresión regular para validar el formato dd/MM/yyyy  String regex = "^\\d{2}/\\d{2}/\\d{4}$";   if (!fechaLimpia.matches(regex)) {  throw new IllegalArgumentException("La fecha debe tener el formato dd/MM/yyyy");  }   // Validar día, mes y año  String[] partesFecha = fechaLimpia.split("/");  int dia = Integer.parseInt(partesFecha[0]);  int mes = Integer.parseInt(partesFecha[1]);  int anio = Integer.parseInt(partesFecha[2]);   if (dia < 1 || dia > 31 || mes < 1 || mes > 12) {  throw new IllegalArgumentException("La fecha ingresada no es válida. Verifique el día y el mes.");  }   // Validar que el mes sea enero o febrero  if (mes != 1 && mes != 2) {  throw new IllegalArgumentException("La fecha debe estar en los meses de enero o febrero.");  }   // Validar el número de días en febrero, considerando si el año es bisiesto  if (mes == 2) {  // Verificar si el año es bisiesto  boolean esBisiesto = (anio % 4 == 0 && (anio % 100 != 0 || anio % 400 == 0));   // Si el año es bisiesto, febrero puede tener 29 días, si no, solo 28 días  if ((esBisiesto && dia > 29) || (!esBisiesto && dia > 28)) {  throw new IllegalArgumentException("Febrero solo puede tener " + (esBisiesto ? "29" : "28") + " días en el año " + anio);  }  }   // Si pasa todas las validaciones, procedemos con la conversión  String sql = "SELECT CONVERT(DATETIME, ?, 103) fecha";  fecha\_format = jdbcTemplate.queryForObject(sql, String.class, fechaLimpia);   return fecha\_format;  }     @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarSeccionAlumno(int alu\_id, int sec\_id) {  String sql = """  WITH GradoAnterior AS (  SELECT TOP 1 S.grad\_id  FROM MATRICULA M  JOIN SECCION S ON M.sec\_id = S.sec\_id  WHERE M.alu\_id = ? -- Parámetro para ID del alumno  ORDER BY M.mat\_fecha DESC  )  SELECT COUNT(\*) AS Cantidad\_de\_Matriculas\_Validas  FROM SECCION S  JOIN GradoAnterior GA ON S.grad\_id = GA.grad\_id OR S.grad\_id = GA.grad\_id + 1  WHERE S.sec\_id = ?; -- Parámetro para ID de la sección  """;    int cont = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id, sec\_id);  if (cont == 0) {  throw new RuntimeException("El alumno solo puede matricularse en el ultimo grado y el que le sigue");  }  }    @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private int validarAlumnoNuevo(int alu\_id) {     String sql = """    SELECT count (\*) cont FROM MATRICULA WHERE alu\_id=?   """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id);     return cont;  }        @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private double costo\_Tipo(String mat\_tipo) {   // Consulta para obtener el precio base desde la base de datos  String sql = "SELECT mensualidad precio FROM PRECIO";  double precioBase = jdbcTemplate.queryForObject(sql, Double.class);   double costo = 0;   switch (mat\_tipo) {  case "BECA":  costo = 0;  break;  case "MEDIABECA":  costo = precioBase \* 0.5;  break;  case "REGULAR":  costo = precioBase;  break;  }   return costo;  }  @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarMatriculaAlumno(int alu\_id, int sec\_id) {  String sql = """  SELECT COUNT(\*) AS existe  FROM MATRICULA M  JOIN SECCION S ON M.sec\_id = S.sec\_id  JOIN GRADO G ON S.grad\_id = G.grad\_id  JOIN ANIO A ON S.anio\_id = A.anio\_id  WHERE M.alu\_id = ?   AND A.anio\_id = (  SELECT anio\_id  FROM SECCION  WHERE sec\_id = ?  );  """;    int cont = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id, sec\_id);    if (cont > 0) {  throw new RuntimeException("El alumno ya está matriculado en otro grado o sección en este año académico.");  }  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarAnio( int sec\_id, String mat\_fecha) {  String sql = """  SELECT COUNT(\*) AS existe  FROM SECCION S  JOIN ANIO A ON S.anio\_id = A.anio\_id  WHERE S.sec\_id = ?  AND YEAR(?) BETWEEN YEAR(A.anio\_inicio) AND YEAR(A.anio\_fin);  """;    int cont = jdbcTemplate.queryForObject(sql, Integer.class, sec\_id, mat\_fecha);      if (cont == 0) {  throw new RuntimeException("El año de la fecha de matrícula no coincide con el año de la sección seleccionada.");  }  }     } |

## SERVICIO DE PAGOS\_CU02 (PagoService)

|  |
| --- |
| @Service public class PagoService {   @Autowired  private JdbcTemplate jdbcTemplate;   @Transactional(propagation = Propagation.REQUIRES\_NEW, rollbackFor = Exception.class)  public PagoDto pagoCuota(PagoDto dto) {  // Validaciones  validarCronogramaid(dto.getCro\_id(), dto.getPag\_fecha());  validarFecha(dto.getCro\_id(), dto.getPag\_fecha());  validarMatricula(dto.getCro\_id());  validarPago(dto.getCro\_id());  validarOrden(dto.getCro\_id());  validarEmpleado(dto.getEmp\_id());  dto.setPag\_fecha\_prog(recuperarFechaProg(dto.getCro\_id()));  dto.setPag\_fecha(convertirFecha(dto.getPag\_fecha()));  validarImporte(dto.getPag\_importe(), dto.getCro\_id(), dto.getPag\_fecha\_prog(), dto.getPag\_fecha());   // Proceso  dto.setPag\_pension(recuperarPension(dto.getCro\_id()));  registrarPago(dto);   // Reporte final  System.out.println("Proceso ok.");  return dto;  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void registrarPago(PagoDto bean) {  String sql = """  INSERT INTO PAGO (cro\_id, emp\_id, pag\_pension, pag\_importe, pag\_fecha, pag\_fecha\_prog)  VALUES (?, ?, ?, ?, ?, ?)  """;  jdbcTemplate.update(sql, bean.getCro\_id(), bean.getEmp\_id(), bean.getPag\_pension(), bean.getPag\_importe(),  bean.getPag\_fecha(), bean.getPag\_fecha\_prog());  }    @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private int recuperarPension(int idcronograma) {  int numero\_pension = (idcronograma - 1) % 11;  return numero\_pension;  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarImporte(double importe, int idcronograma, String fecha\_prog, String fecha) {  String sql = """  SELECT cro\_monto FROM CRONOGRAMA\_PAGO  WHERE cro\_id = ?  """;  double costo = jdbcTemplate.queryForObject(sql, Double.class, idcronograma);  double mora = calcularMora(fecha\_prog, fecha, costo);  costo = costo + mora;   if (costo != importe) {  throw new RuntimeException("El importe no es correcto");  }  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  public double calcularMora(String fecha\_prog, String fecha, double costo) {  // Definir los formatos esperados para cada fecha  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");   // Convertir las cadenas de fechas a LocalDate con el formato especificado para  // cada una  LocalDate fechaProg = LocalDate.parse(fecha\_prog, formatter);  LocalDate fechaActual = LocalDate.parse(fecha, formatter);   // Inicializar la variable para la mora  double mora = 0;   // Calcular la diferencia en días solo si fechaActual es después de fechaProg  if (fechaActual.isAfter(fechaProg)) {  double diasDiferencia = ChronoUnit.DAYS.between(fechaProg, fechaActual);  double tasaMora = 0.05;  mora = diasDiferencia \* tasaMora \* costo;  }   return mora;  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  public String convertirFecha(String fecha) {  // Definir los formatos: de entrada (dd/MM/yyyy) y de salida (yyyy-MM-dd)  DateTimeFormatter inputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");  DateTimeFormatter outputFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");   // Parsear la fecha de entrada y formatearla al nuevo formato  LocalDate date = LocalDate.parse(fecha, inputFormatter);  return date.format(outputFormatter);  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  public String recuperarFechaProg(int idcronograma) {  String sql = """  SELECT cro\_fecha\_prog FROM CRONOGRAMA\_PAGO  WHERE cro\_id = ?  """;   // Recuperar la fecha como Timestamp  Timestamp timestamp = jdbcTemplate.queryForObject(sql, Timestamp.class, idcronograma);   // Convertir el Timestamp a LocalDate  LocalDate fecha = timestamp.toLocalDateTime().toLocalDate();   // Formatear la fecha a String en el formato yyyy-MM-dd  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");  return fecha.format(formatter);  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarEmpleado(int idempleado) {  String sql = """  SELECT COUNT(\*) FROM EMPLEADO  WHERE emp\_id = ?  """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, idempleado);  if (cont != 1) {  throw new RuntimeException("El id del empleado no existe");  }  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarOrden(int idcronograma) {  // Recuperando el id de cronograma  String sql = """  SELECT m.mat\_id FROM MATRICULA m  JOIN CRONOGRAMA\_PAGO c ON m.mat\_id = c.mat\_id   WHERE c.cro\_id = ?  """;  int idmatricula = jdbcTemplate.queryForObject(sql, Integer.class, idcronograma);  sql = """  SELECT COUNT(\*)  FROM PAGO p  JOIN CRONOGRAMA\_PAGO c ON p.cro\_id = c.cro\_id  WHERE c.mat\_id = ?  """;    // Consulta que cuenta la cantidad de filas de la tabla Pago por id de matricula  int cont = jdbcTemplate.queryForObject(sql, Integer.class, idmatricula);   // Numero de cuota que va a pagar  int cuota = (idcronograma - 1) % 11;   // Verificar si el contador coincide con el numero de cuota que va a pagar  if (cuota != cont) {  throw new RuntimeException("El pago se debe realizar en orden.");  }   }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarPago(int idcronograma) {  // Validar si ya está pagado  String sql = """  SELECT COUNT(\*) FROM PAGO WHERE cro\_id = ?  """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, idcronograma);  if (cont != 0) {  throw new RuntimeException("El pago ya se ha realizado");  }  }    @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarMatricula(int idcronograma) {  String sql = """  SELECT m.mat\_estado FROM MATRICULA m  JOIN CRONOGRAMA\_PAGO c ON m.mat\_id = c.mat\_id  WHERE cro\_id = ?  """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, idcronograma);  if (cont != 1) {  throw new RuntimeException("La matricula no se encuentra activa");  }  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarFecha(int idcronograma, String fecha) {  // Validar que la fecha sea igual a la del año programado  // Recuperando año programado  String sql = """  SELECT YEAR(cro\_fecha\_prog) FROM CRONOGRAMA\_PAGO WHERE cro\_id = ?  """;  int anio\_prog = jdbcTemplate.queryForObject(sql, Integer.class, idcronograma);   // Definir el formato de las fechas  LocalDate fechaPaga = convertirYValidarFecha(fecha);   // Convertir LocalDate a java.sql.Date para usarlo en la consulta SQL  java.sql.Date sqlFecha = java.sql.Date.valueOf(fechaPaga);   // Consultar el año de la fecha proporcionada  sql = """  SELECT YEAR(?)  """;  int anio = jdbcTemplate.queryForObject(sql, Integer.class, sqlFecha);   // Comparar el año proporcionado con el año programado  if (anio\_prog != anio) {  throw new RuntimeException("La Fecha no corresponde al año programado.");  }  }    @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private LocalDate convertirYValidarFecha(String fecha) {  try {  // Separar los componentes de la fecha (dd/MM/yyyy)  String[] partes = fecha.split("/");  if (partes.length != 3) {  throw new RuntimeException("Formato de fecha invalido. Debe ser dd/MM/yyyy: " + fecha);  }   int dia = Integer.parseInt(partes[0]);  int mes = Integer.parseInt(partes[1]);  int anio = Integer.parseInt(partes[2]);   // Validar que la fecha sea válida usando LocalDate.of  return LocalDate.of(anio, mes, dia);  } catch (NumberFormatException | DateTimeException e) {  throw new RuntimeException("La fecha proporcionada no es valida: " + fecha, e);  }  }   @Transactional(propagation = Propagation.MANDATORY, rollbackFor = Exception.class)  private void validarCronogramaid(int idcronograma, String fecha) {  // Validar que el id del cronograma exista  String sql = """  SELECT COUNT(\*) FROM CRONOGRAMA\_PAGO  WHERE cro\_id = ?  """;  int cont = jdbcTemplate.queryForObject(sql, Integer.class, idcronograma);  if (cont != 1) {  throw new RuntimeException("El id del cronograma no existe");  }  }  } |

## SERVICIO DE CONSULTAS\_CU03 (ConsultasService)

|  |
| --- |
| @Service public class ConsultasService {      @Autowired  private JdbcTemplate jdbcTemplate;    public String cronogramaPago(int alu\_id) throws Exception {  int contador = 0;  StringBuilder reporte = new StringBuilder();  List<FechapagoDto> lista = new LinkedList<>();  try {    String sql = """  SELECT COUNT(\*)  FROM ALUMNO  WHERE alu\_id = ?  """;  int aux = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id);  if (aux == 0) {  throw new Exception("Alumno no existe");  }    sql = """  SELECT  COUNT(\*)  FROM MATRICULA  WHERE alu\_id = ?  """;  aux = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id);  if (aux == 0) {  throw new Exception("Alumno no está matriculado");  }    sql = """  SELECT  cro\_monto mensualidad,  cro\_fecha\_prog fecha  FROM CRONOGRAMA\_PAGO  WHERE mat\_id=?  """;   List<Map<String, Object>> rows = jdbcTemplate.queryForList(sql, alu\_id);  for (Map<String, Object> row : rows) {  FechapagoDto dto = new FechapagoDto();  dto.setFecha(row.get("fecha").toString());  dto.setMonto(Double.parseDouble(row.get("mensualidad").toString()));  dto.setCuota(contador);  lista.add(dto);  contador++;  }   reporte.append(String.format("%-15s %-27s %-27s%n", "CUOTA", "MONTO A PAGAR", "FECHA DE PAGO"));  reporte.append("------------------------------------------------------------------------\n");  String cuota = "";  for (FechapagoDto elemento : lista) {  cuota = Integer.toString(elemento.getCuota());  if (elemento.getCuota() == 0) {  cuota = "Matricula";  }  reporte.append(String.format("%-15s %-27s %-27s%n", cuota, elemento.getMonto(), elemento.getFecha()));  }  reporte.append("------------------------------------------------------------------------\n");     } catch (DataAccessException e) {  System.err.println("ERROR AL OBTENER LA INFORMACION: " + e.getMessage());  }   return reporte.toString();  }   public List<FechapagoDto> pagoJSON(int alu\_id) throws Exception {  int contador = 0;  List<FechapagoDto> lista = new LinkedList<>();  try {    String sql = """  SELECT COUNT(\*)  FROM ALUMNO  WHERE alu\_id = ?  """;  int aux = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id);  if (aux == 0) {  throw new Exception("Alumno no existe");  }    sql = """  SELECT  COUNT(\*)  FROM MATRICULA  WHERE alu\_id = ?  """;  aux = jdbcTemplate.queryForObject(sql, Integer.class, alu\_id);  if (aux == 0) {  throw new Exception("Alumno no está matriculado");  }    sql = """  SELECT  cro\_monto mensualidad,  cro\_fecha\_prog fecha  FROM CRONOGRAMA\_PAGO  WHERE mat\_id=?  """;    List<Map<String, Object>> rows = jdbcTemplate.queryForList(sql, alu\_id);  for (Map<String, Object> row : rows) {  FechapagoDto dto = new FechapagoDto();  dto.setFecha(row.get("fecha").toString());  dto.setMonto(Double.parseDouble(row.get("mensualidad").toString()));  dto.setCuota(contador);  lista.add(dto);  contador++;  }   } catch (DataAccessException e) {  System.err.println("ERROR AL OBTENER LA INFORMACION: " + e.getMessage());  }   return lista;  }   @SuppressWarnings("deprecation")  public CursosProfesorResponse cursosProfesor(int id\_prof) throws Exception {  String sql;  CursosProfesorResponse response = new CursosProfesorResponse();  ProfesorDto pdto = new ProfesorDto();  List<ClasesDto> lista = new LinkedList<>();  try {  sql = """  SELECT  COUNT(prof\_id)  FROM SECCION\_CURSO  WHERE prof\_id = ?  """;  int aux = jdbcTemplate.queryForObject(sql, Integer.class, id\_prof);  if (aux == 0) {  throw new Exception("Profesor no existe");  }  sql = """  SELECT  CONCAT(prof\_apellido,', ',prof\_nombre) profesor,  prof\_dni dni,  prof\_email email,  prof\_telefono telefono  FROM PROFESOR  WHERE prof\_id=?  """;   pdto = jdbcTemplate.queryForObject(  sql,  new Object[]{id\_prof},  new BeanPropertyRowMapper<>(ProfesorDto.class)  );    sql = """  SELECT  CONCAT(a.grad\_id, a.sec\_nombre) seccion,  c.cur\_nombre curso  FROM SECCION\_CURSO s JOIN CURSO c ON S.cur\_id=C.cur\_id  JOIN SECCION a ON a.sec\_id=s.sec\_id  WHERE prof\_id=?  """;   List<Map<String, Object>> rows = jdbcTemplate.queryForList(sql, id\_prof);  for (Map<String, Object> row : rows) {  ClasesDto dto = new ClasesDto();  dto.setCurso(row.get("curso").toString());  dto.setSeccion(row.get("seccion").toString());  lista.add(dto);  }  response.setProfesor(pdto);  response.setClases(lista);  } catch (DataAccessException e) {  System.err.println("ERROR AL OBTENER LA INFORMACION: " + e.getMessage());  }   return response;  }   public String horario(int sec\_id) throws Exception {  StringBuilder reporte = new StringBuilder();  List<HorarioDto> lista = new LinkedList<>();  List<HoraDto> horas = new LinkedList<>();  String lunes = "";  String martes = "";  String miercoles = "";  String jueves = "";  String viernes = "";  String sql = "";  try {  sql = """  SELECT  COUNT(h.hor\_id)  FROM HORARIO h  JOIN SECCION\_CURSO s ON h.asig\_id = s.asig\_id  WHERE s.sec\_id = ?;  """;  int aux = jdbcTemplate.queryForObject(sql, Integer.class, sec\_id);  if (aux == 0) {  throw new Exception("Seccion no existe");  }  sql = """  SELECT  CONVERT(VARCHAR(8), h.hor\_inicio, 108) AS inicio,  h.hor\_dia AS dia,  c.cur\_nombre AS nombre  FROM HORARIO h  JOIN SECCION\_CURSO s ON h.asig\_id = s.asig\_id  JOIN CURSO c ON c.cur\_id = s.cur\_id  WHERE s.sec\_id = ?;  """;   List<Map<String, Object>> rows = jdbcTemplate.queryForList(sql, sec\_id);  for (Map<String, Object> row : rows) {  HorarioDto dto = new HorarioDto();  dto.setInicio(row.get("inicio").toString());  dto.setDia(row.get("dia").toString());  dto.setNombre(row.get("nombre").toString());  lista.add(dto);  }   sql = """  SELECT  CONVERT(VARCHAR(8), hor\_inicio, 108) AS inicio,  CONVERT(VARCHAR(8), hor\_fin, 108) AS fin  FROM HORARIO  GROUP BY  CONVERT(VARCHAR(8), hor\_inicio, 108),  CONVERT(VARCHAR(8), hor\_fin, 108);  """;   List<Map<String, Object>> filas = jdbcTemplate.queryForList(sql);  for (Map<String, Object> fila : filas) {  HoraDto dto = new HoraDto();  dto.setInicio(fila.get("inicio").toString());  dto.setFin(fila.get("fin").toString());  horas.add(dto);  }   reporte.append(String.format("%-15s %-15s %-27s %-27s %-27s %-27s %-27s%n", "INICIO", "FIN", "LUNES",  "MARTES", "MIERCOLES", "JUEVES", "VIERNES"));  reporte.append(  "--------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");   for (HoraDto elemento : horas) {  for (HorarioDto dto : lista) {  if (dto.getDia().equals("LUNES") && elemento.getInicio().equals(dto.getInicio())) {  lunes = dto.getNombre();  }  if (dto.getDia().equals("MARTES") && elemento.getInicio().equals(dto.getInicio())) {  martes = dto.getNombre();  }  if (dto.getDia().equals("MIERCOLES") && elemento.getInicio().equals(dto.getInicio())) {  miercoles = dto.getNombre();  }  if (dto.getDia().equals("JUEVES") && elemento.getInicio().equals(dto.getInicio())) {  jueves = dto.getNombre();  }  if (dto.getDia().equals("VIERNES") && elemento.getInicio().equals(dto.getInicio())) {  viernes = dto.getNombre();  }  }  reporte.append(String.format("%-15s %-15s %-27s %-27s %-27s %-27s %-27s%n", elemento.getInicio(),  elemento.getFin(), lunes, martes, miercoles, jueves, viernes));  if (elemento.getInicio().equals("09:30:00")) {  reporte.append(String.format("%-15s %-15s %-27s %-27s %-27s %-27s %-27s%n", "10:15:00", "10:30:00",  "RECREO", "RECREO", "RECREO", "RECREO", "RECREO"));  }  if (elemento.getInicio().equals("11:15:00")) {  reporte.append(String.format("%-15s %-15s %-27s %-27s %-27s %-27s %-27s%n", "12:00:00", "12:15:00",  "RECREO", "RECREO", "RECREO", "RECREO", "RECREO"));  }  }  reporte.append(  "--------------------------------------------------------------------------------------------------------------------------------------------------------------------\n");  } catch (Exception e) {  throw new Exception(e.getMessage());  }   return reporte.toString();  } } |

# 

# CONCLUSIONES

**Evaluación General del Proyecto**:

* El proyecto logró establecer una base sólida para la gestión escolar a nivel primario, permitiendo optimizar procesos clave como la matrícula, pagos y asignación de horarios.
* La implementación con Spring Boot y SQL Server garantiza escalabilidad y mantenimiento futuro.

**Limitaciones y Futuro Desarrollo**:

* Aunque se ha cumplido con los objetivos principales, el sistema presenta áreas de mejora, como la inclusión de funciones para el registro directo de estudiantes y el manejo avanzado de reportes.
* Se recomienda priorizar la integración de un frontend y la automatización de procesos más complejos en futuras iteraciones.

**Impacto Tecnológico**:

* Este sistema puede servir como base para expandirse a otras instituciones educativas, integrando niveles superiores y funcionalidades adicionales.

**Importancia de la Validación de Datos**:

* La implementación de validaciones y controles asegura la integridad de los datos, lo que es esencial en entornos educativos.

# LECCIONES APRENDIDAS

**Importancia de los Requerimientos Iniciales**:

* La claridad en la recopilación y análisis de requerimientos es fundamental para el éxito del proyecto.

**Impacto de la Documentación**:

* La documentación técnica detallada ayuda a garantizar la mantenibilidad del sistema y facilita futuras mejoras.

**Gestión de Alcances**:

* Definir un alcance realista desde el inicio evita sobrecarga de trabajo y garantiza entregables de calidad.

**Validaciones y Seguridad**:

* Incorporar validaciones a nivel de base de datos y backend refuerza la seguridad y coherencia del sistema.

**Aprendizaje Colaborativo**:

* El proyecto destaca la importancia del trabajo en equipo y la distribución efectiva de tareas.

# BIBLIOGRAFÍA

Creately. (2022, 21 de octubre). Tutorial del diagrama de secuencia: Guía completa con ejemplos. <https://creately.com/blog/es/diagramas/tutorial-del-diagrama-de-secuencia/>

Daniel. (2023, 30 de octubre). SQL: Todo lo que necesitas saber sobre el lenguaje de programación de bases de datos. *Formación en Ciencia de Datos | DataScientest.com*. <https://datascientest.com/es/sql-todo-lo-que-necesitas-saber-sobre-el-lenguaje-de-programacion-de-bases-de-datos>

GeeksforGeeks. (2024, 1 de enero). How to send different types of requests (GET, POST, PUT, DELETE) in Postman. *GeeksforGeeks*. <https://www.geeksforgeeks.org/how-to-send-different-types-of-requests-get-post-put-delete-in-postman/>

Introducing EDrawMax 10. (n.d.). Edrawsoft. <https://www.edrawsoft.com/es/er-diagram/>

Lucid Software Inc. (2024). *Lucid.app* [Herramienta en línea]. Lucid Software Inc. <https://www.lucid.app>

Microsoft. (2024). *SQL Server Management Studio* (versión 20) [Herramienta de gestión de bases de datos]. Microsoft. <https://learn.microsoft.com/en-us/sql/ssms/>

OpenWebinars. (2019, 28 de noviembre). *CÓMO INSTALAR y USAR LOMBOK- TUTORIAL EN ESPAÑOL* [Video]. YouTube. <https://www.youtube.com/watch?v=EDGTS8SwzPA>

VMware. (2024). *Spring Tools Suite 4* (versión X.X) [Software de desarrollo]. VMware. <https://spring.io/tools>