Caso de Estudio

La Secretaría de Asuntos Estudiantiles y Bienestar Universitario (SAEBU) desea un sistema de cobros que administre los pagos de los alumnos para el Departamento de Educación Física y Deporte UNSL. Para esto, es necesario administrar la información de los alumnos, profesores, disciplinas, pagos y equipos.

De los alumnos, necesitamos sus datos personales, características (escolar, universitario, extrauniversitario) y posibles observaciones. Los alumnos pueden estar inscriptos en más de una disciplina deportiva.

Cuentan con disciplinas identificadas con un código, pudiendo realizarse en forma recreativa o competitiva. La recreativa, es sin costo para los alumnos escolares o universitarios. En caso de ser extrauniversitario, deberá pagar cuotas mensuales y abonar el costo de la inscripción una única vez (por disciplina) para estar presente en el sistema. La competitiva, le permite a todos los alumnos representar a la UNSL en el equipo oficial, teniendo que abonar una cuota mensual para realizar las actividades. En cualquiera de los casos, el alumno estará cubierto por el seguro (tipo 1, 2 o 3).

El sistema deberá registrar los pagos efectuados por equipos deportivos para poder pagar diversos recursos destinados a competiciones, como pueden ser los arbitrajes.

Cuando se abonan los pagos, el sistema debe encargarse de generar un ticket, el cual tendrá como información el número del mismo, la fecha de emisión y el operador que realizó el cobro, sabiendo su nombre completo y dni.

También se deben tener los datos personales y la fecha de ingreso de los profesores que imparten las disciplinas, los cuales pueden enseñar más de una.

El sistema puede ser accedido por distintos usuarios, siendo estos: cajero, administrador y director general. Estos se encuentran en orden ascendente, siendo el director general el que tiene acceso a todas las funcionalidades del sistema y permitir el registro de nuevos usuarios.

Requerimientos funcionales y no funcionales

Requerimientos funcionales:

- ABM de alumnos/profesores/disciplinas/equipos.
- Consultar alumnos/profesores/disciplinas/equipos/pagos.
- Asentar/Anular pagos de Alumnos (cuotas y/o inscripción) o Equipos (pago único).
- Restringir pago.
- Imprimir ticket.
- Imprimir triplicado.
- Consultar historial de pagos.
- Listar alumnos/profesores (con la posibilidad de aplicar filtros).
- Consultar recaudación (con la posibilidad de aplicar filtros).
- Generar tablas de Excel con los datos que se requieran.
- Sistema de usuarios.

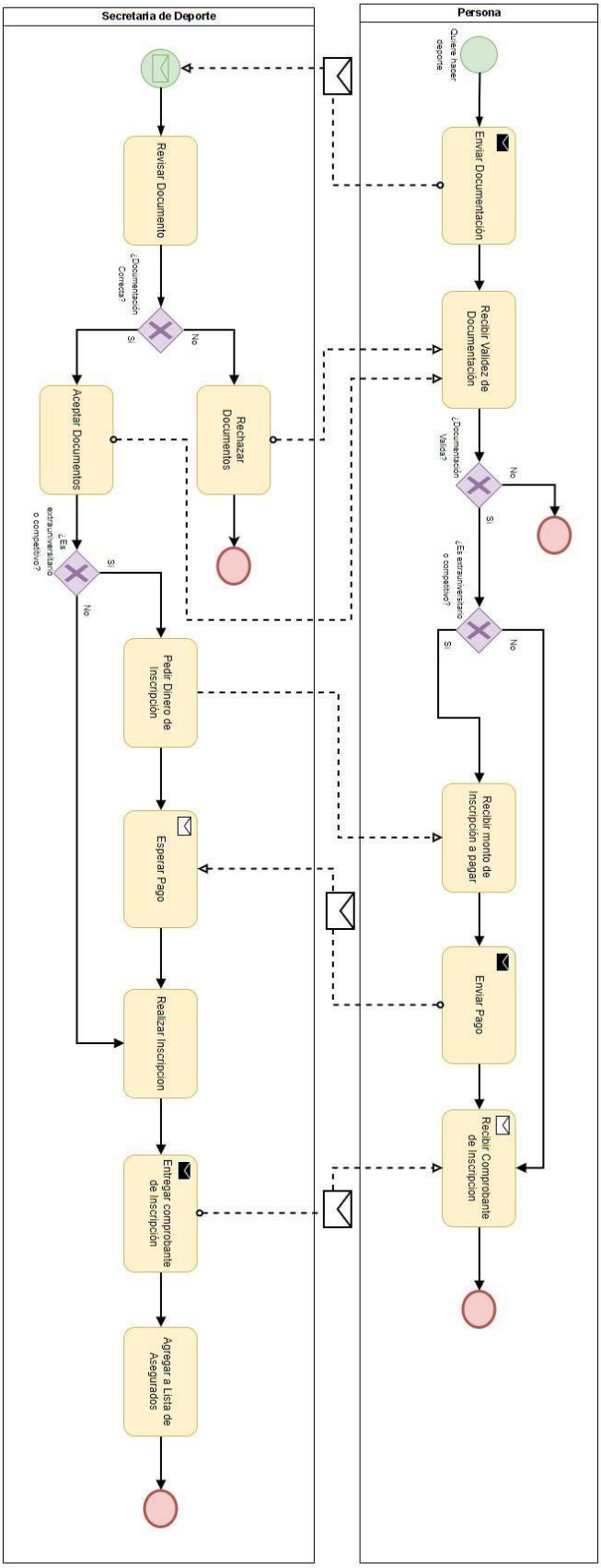
Requerimientos no funcionales:

- Administración de una base de datos.
- Usabilidad.
- Privacidad.
- Robustez.
- Fiabilidad.
- Seguridad.
- Portabilidad.

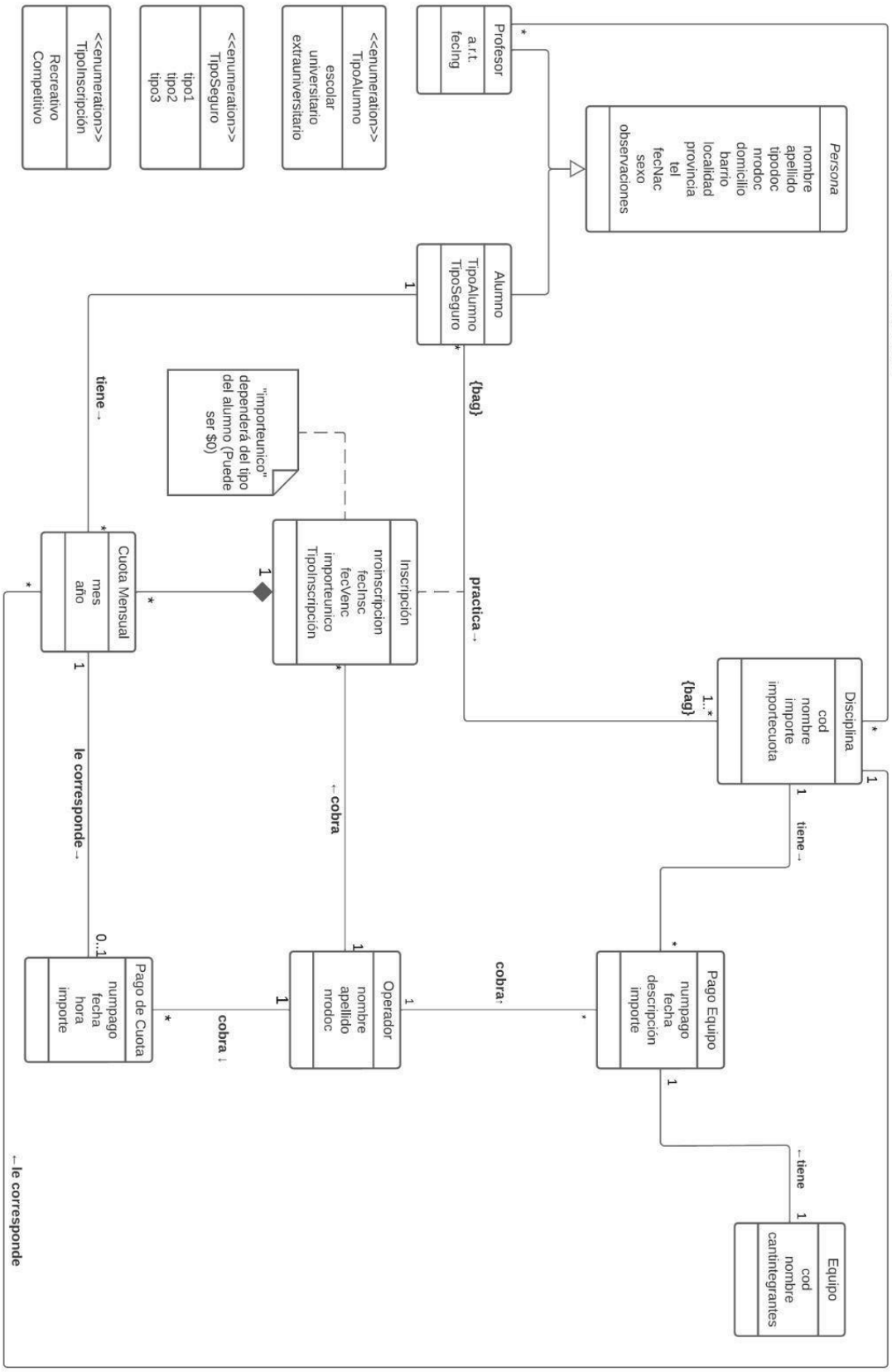
Visión

Nuestra visión es desarrollar un sistema de gestión de cobros eficiente y confiable que facilite el acceso y mejore la experiencia de los usuarios. Visualizamos un futuro en donde el sistema mejore la infraestructura tecnológica que posee SAEBU.

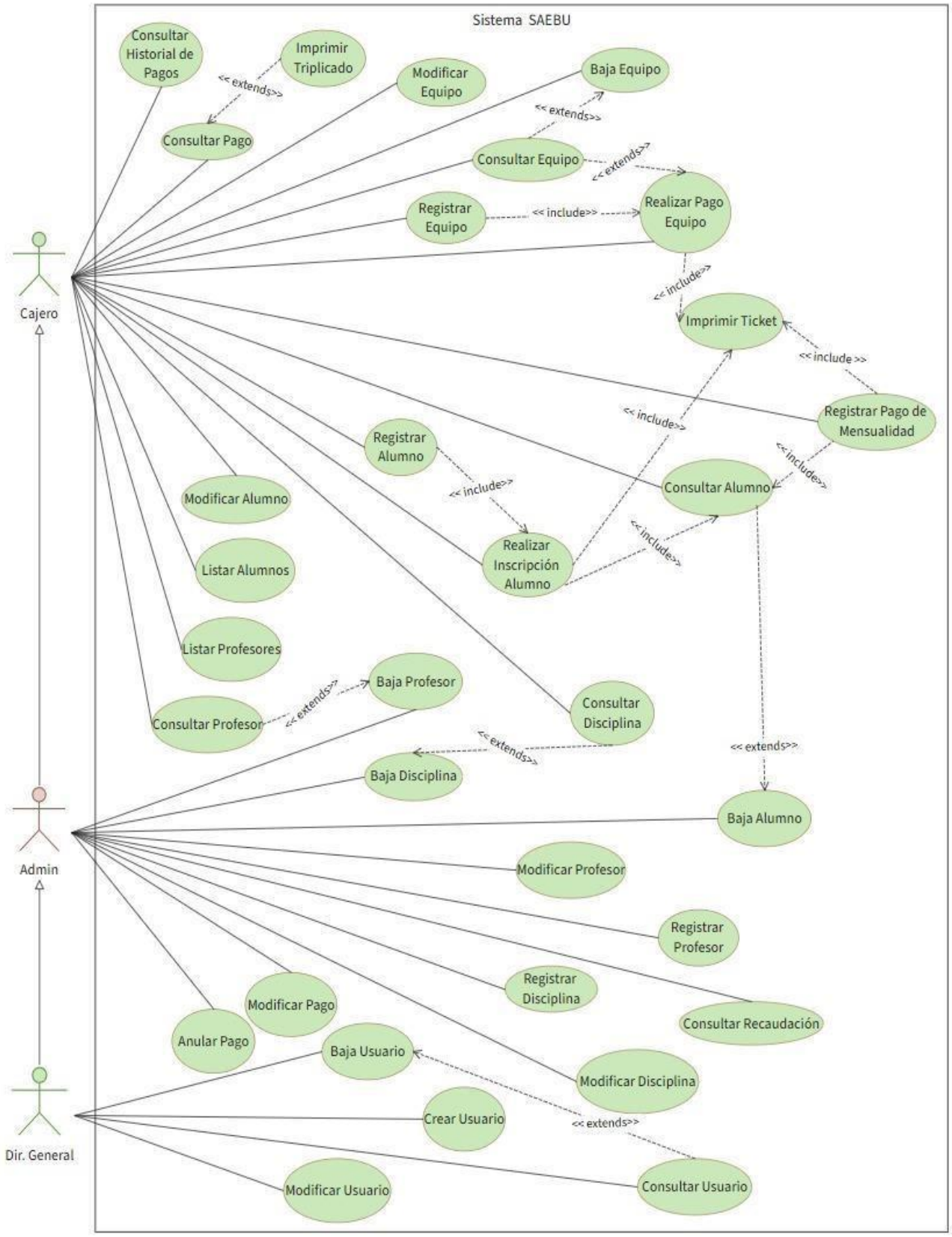
Diagrama BPMN para el proceso “Inscripción”



Modelo de Dominio



Modelo de Casos de Uso



Modelos de *diseño* para el CU “Registrar Pago de Mensualidad”

Diagrama de clases:

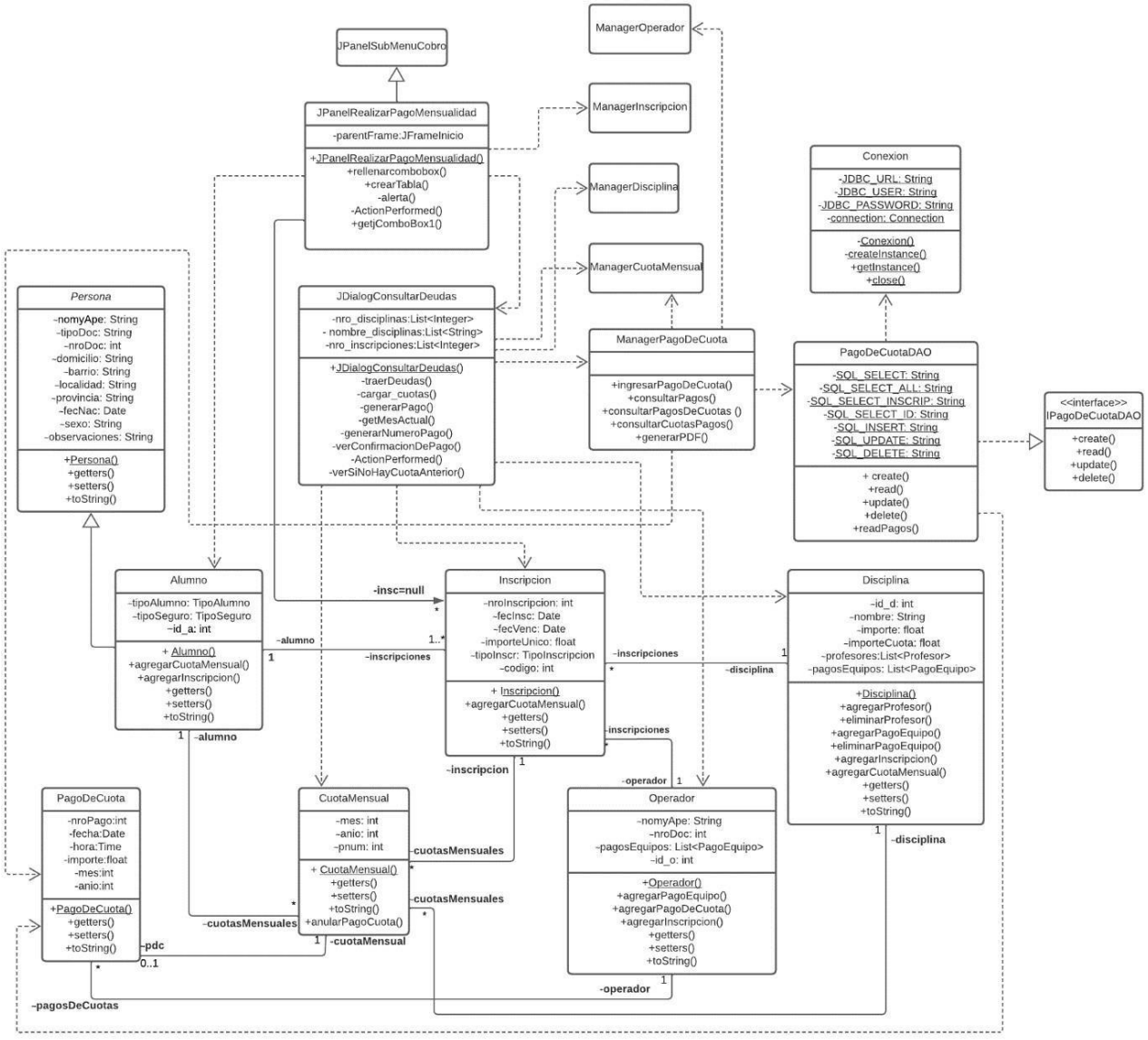
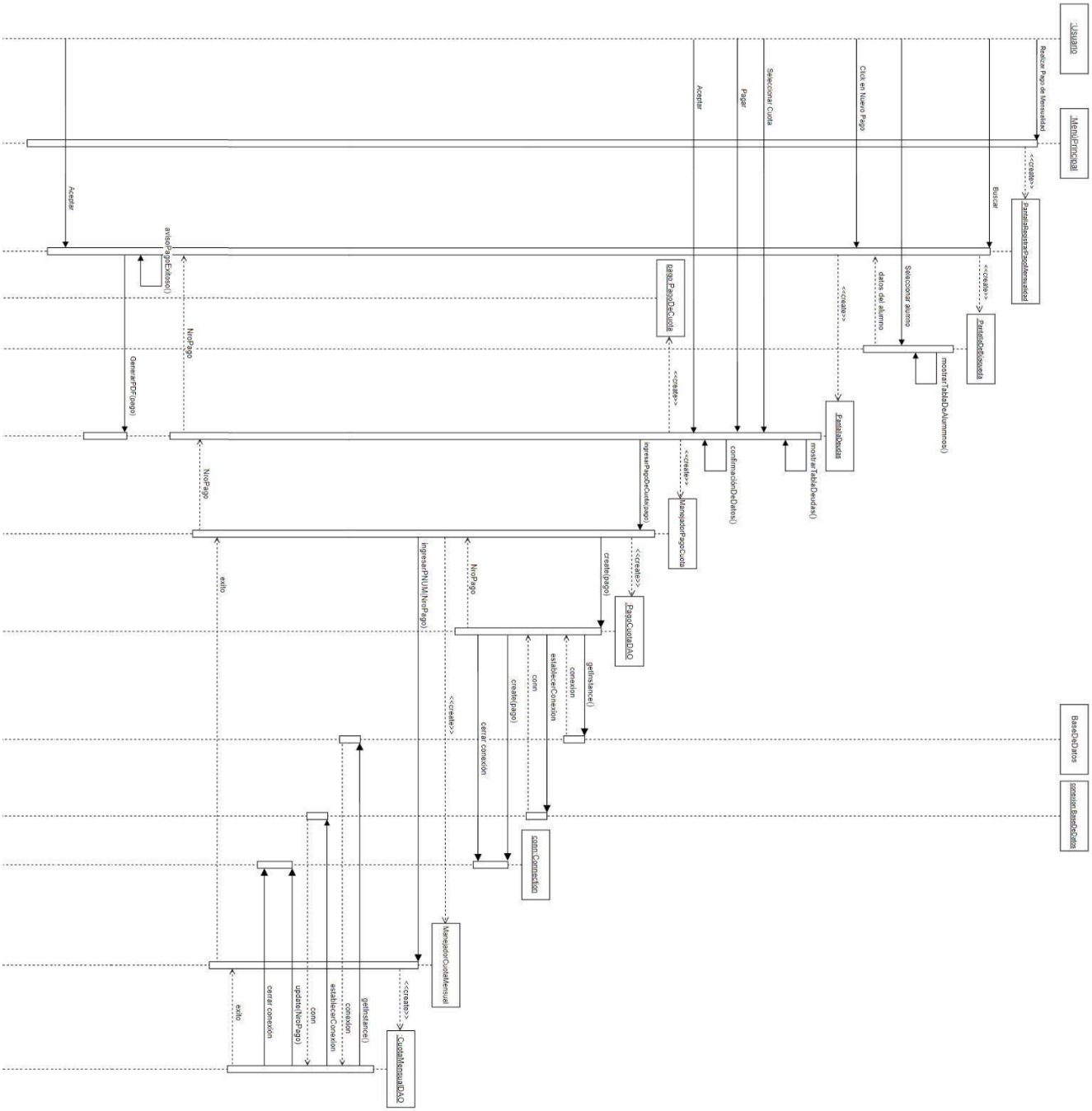


Diagrama de Secuencia:



Modelos de *diseño* para el CU “Realizar Inscripción Alumno”

Diagrama de clases:

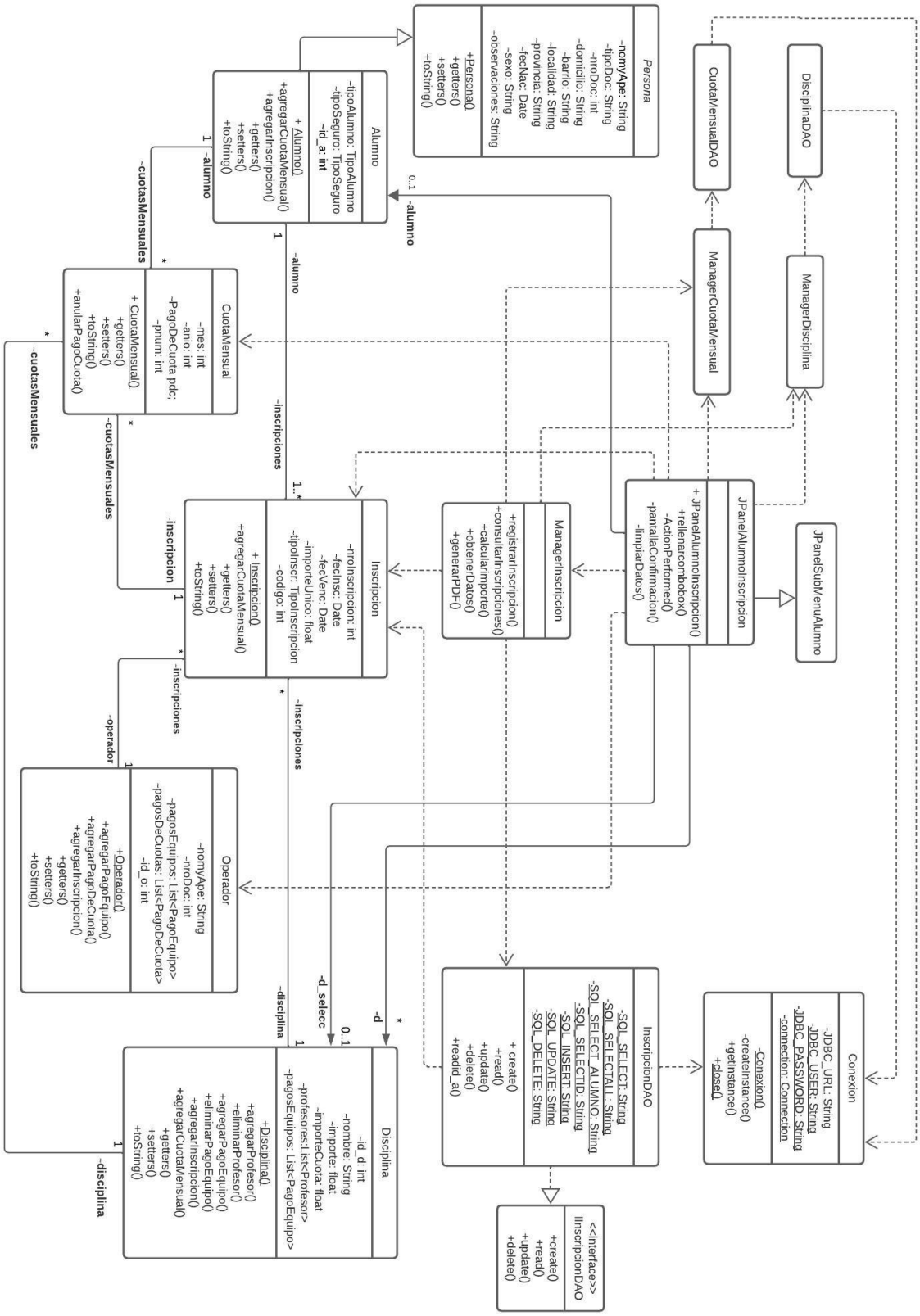


Diagrama de Secuencia:

Especificación de Casos de Uso

Caso de Uso: Registrar Pago de Mensualidad

1 Breve Descripción

Este caso de uso describe como el actor puede registrar en el sistema el pago de una cuota de un alumno inscripto a una disciplina.

2 Actores

Usuario (Cajero/Administrador/Dir. General).

3 Precondiciones

---

4 Flujo básico de eventos

1. El sistema muestra la pantalla de login.
2. El actor inicia sesión.
3. El sistema muestra el menú principal.
4. El actor selecciona “Pago”.
5. El actor selecciona “Registrar Pago de Mensualidad”.
6. El sistema despliega la pantalla “Registrar Pago de Mensualidad”.
7. El actor selecciona “Buscar”.
8. Se ejecuta el caso de uso “Consultar Alumno”.
9. El sistema despliega nuevamente la pantalla del paso 6 y rellena la tabla con todos los pagos realizados del alumno.
10. El actor selecciona “Nuevo Pago”.
11. El sistema despliega la pantalla con la tabla de cuotas pendientes que pertenecen al alumno.
12. El actor selecciona una de las cuotas pendientes y presiona “Pagar”.
13. El sistema muestra una ventana de confirmación de pago.
14. El actor confirma el pago.
15. El sistema registra el pago en la base de datos.
16. Se ejecuta el caso de uso “Imprimir Ticket”.
17. El sistema despliega la pantalla del paso 6.
18. El caso de uso finaliza.

5 Flujo alternativo de eventos

Si en el paso 7 del flujo principal de eventos el actor presiona “Nuevo Pago” sin buscar el alumno:

1. El sistema muestra un mensaje de error y solicita que busque un alumno.

Si en el paso 11 del flujo principal de eventos el actor presiona “Volver”:

1. El sistema cancela la operación y vuelve a la pantalla “Registrar Pago de Mensualidad”.

Si en el paso 12 del flujo principal de eventos el actor presiona “Pagar” sin seleccionar una cuota:

1. El sistema muestra un mensaje de error y solicita que seleccione una cuota pendiente.

Si en el paso 12 del flujo principal de eventos el actor presiona “Pagar” seleccionando una cuota no permitida:

1. El sistema muestra un mensaje de error y solicita que seleccione una cuota permitida.

Si en el paso 13 del flujo principal de eventos el actor presiona “No”:

1. El sistema cancela el pago y vuelve a la pantalla de cuotas.

Caso de Uso: Realizar Inscripción Alumno

1. Breve Descripción

Refleja las operaciones necesarias para realizar la inscripción de un alumno y registrar la misma en la base de datos.

2. Actores

Usuario (Cajero/Administrador/Dir. General).

3. Precondiciones

---

4. Flujo básico de eventos

1. El sistema muestra la pantalla de login.
2. El actor inicia sesión.
3. El actor muestra el menú principal.
4. El actor presiona “Alumno”.
5. El actor presiona “Inscripción”.
6. El actor presiona “Buscar”.
7. Se ejecuta el caso de uso “Consultar Alumno”.
8. El actor selecciona la disciplina deseada.
9. El actor selecciona el tipo de inscripción (Recreativo/Competitivo).
10. El sistema genera el importe a pagar dependiendo de los datos ingresados.
11. El actor ingresa el periodo en el que la inscripción estará vigente.
12. El actor presiona “Realizar Inscripción”.
13. El sistema muestra un cartel de confirmación.
14. El actor presiona “OK”.
15. El sistema registra la Inscripción en la base de datos.
16. El sistema registra las cuotas mensuales correspondientes en la base de datos.
17. El sistema muestra un cartel de inscripción exitosa.
18. El actor presiona “OK”.
19. Se ejecuta el caso de uso “Imprimir Ticket”.
20. El Caso de Uso finaliza.

5. Flujo alternativo de eventos

Si en el paso 14 del flujo principal de eventos el usuario presiona “Cancelar”

1. El sistema cancela el registro de la inscripción.

Si en el paso 15 del flujo principal de eventos ocurre un error al registrar la inscripción (inscripción ya vigente / periodo inválido / falta de algún campo)

1. El sistema informa cuál fue el problema.

Modelo de *diseño* del caso de uso “Consultar Pago”

Diagrama de clases:

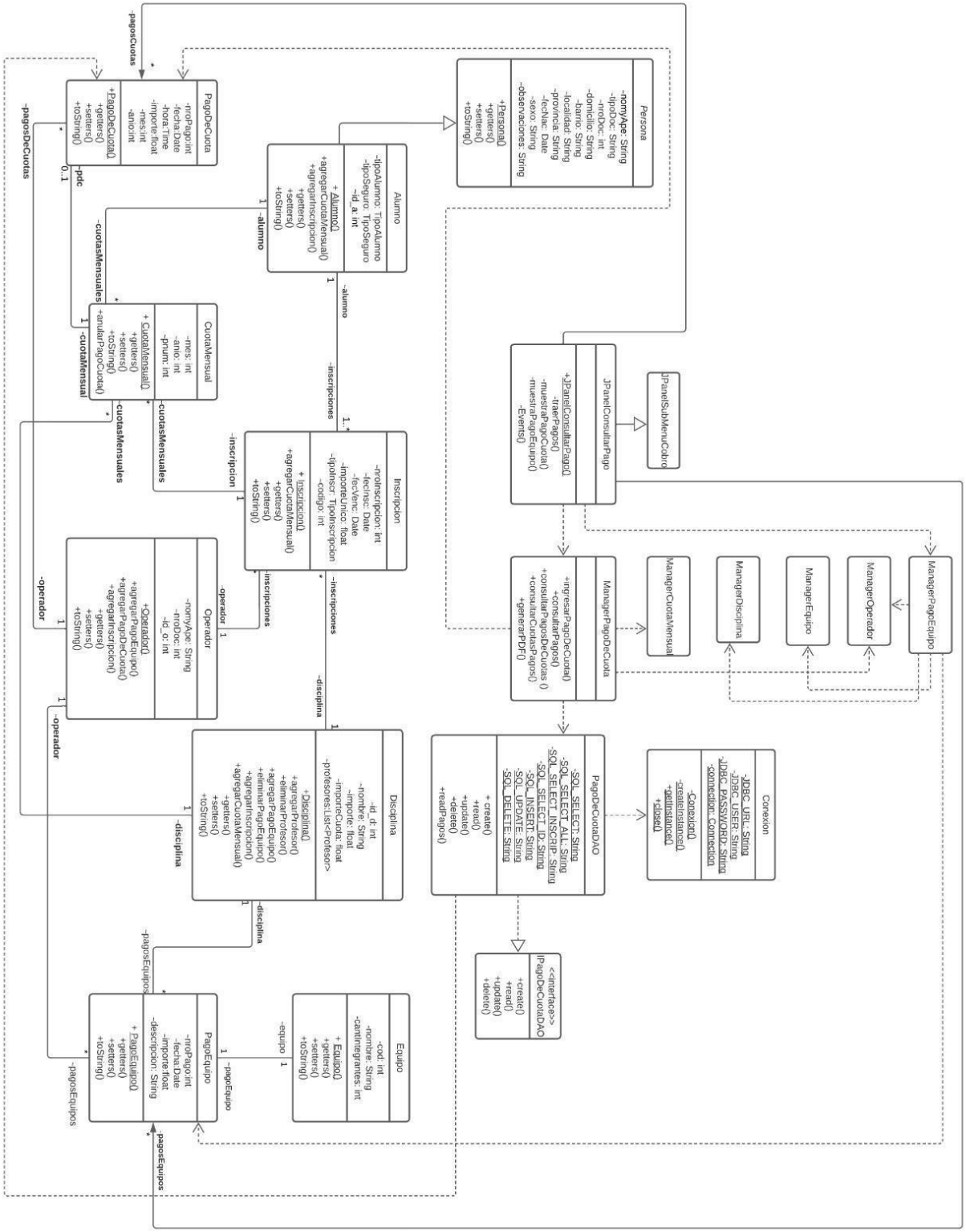
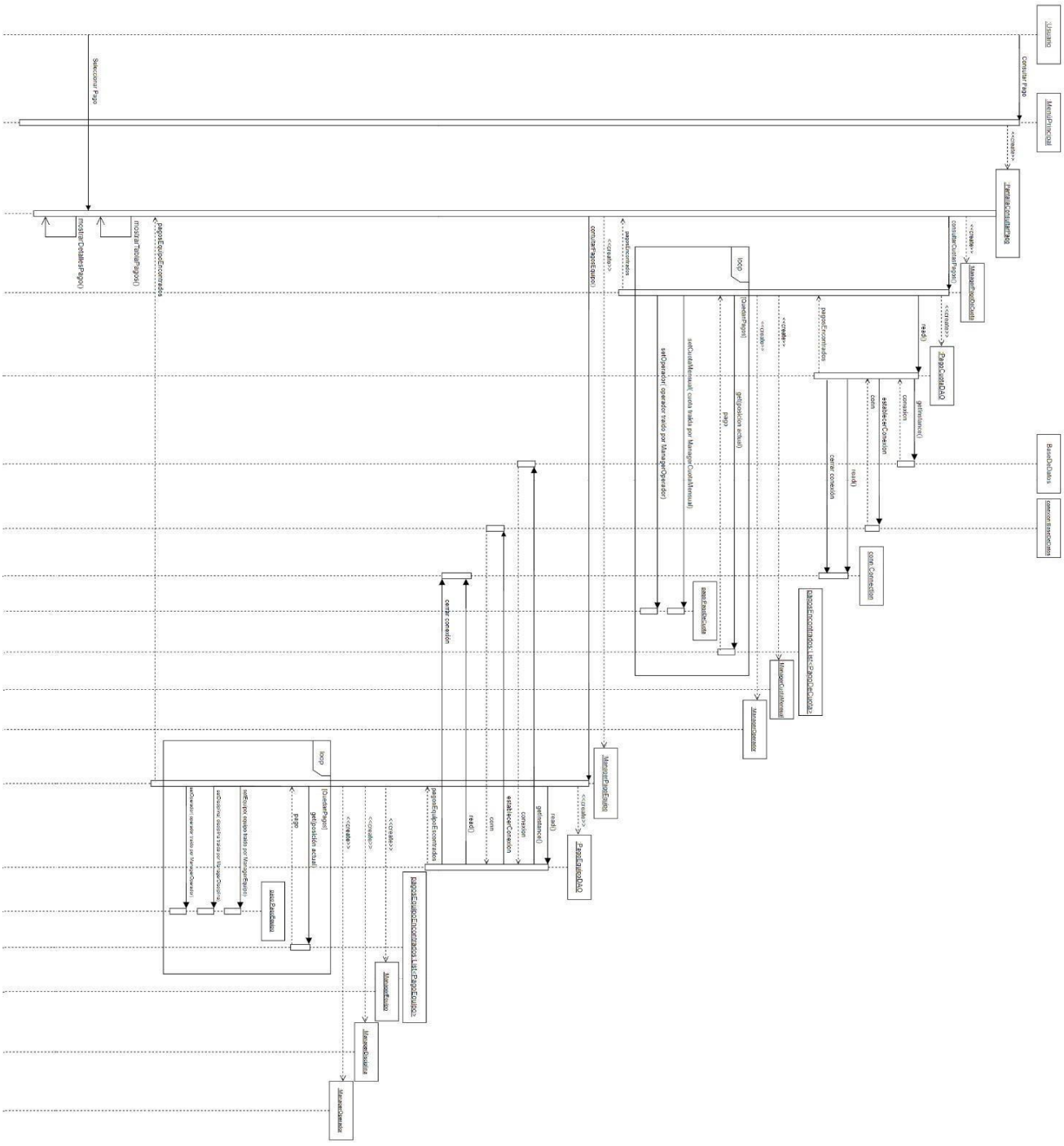


Diagrama de Secuencia:



Especificación del caso de uso “Consultar Pago”

Caso de Uso: Consultar Pago

1 Breve Descripción

Refleja las operaciones necesarias para consultar un pago realizado en el sistema.

2 Actores

Usuario (Cajero/Administrador/Dir. General).

3 Precondiciones

---

4 Flujo básico de eventos

1. El sistema muestra la pantalla de login.
2. El usuario inicia sesión.
3. El sistema muestra el menú principal.
4. El usuario presiona “Pago”.
5. El usuario presiona “Consultar”.
6. El sistema muestra una tabla con todos los pagos realizados.
7. El usuario filtra y/o hace click en el pago deseado.
8. El sistema muestra un cartel con los datos detallados del pago seleccionado.
9. El usuario cierra el cartel.
10. El caso de uso finaliza.

5 Flujo alternativo de eventos

Si en el paso 8 el usuario presiona “Si” :

1. Se ejecuta el caso de uso “Imprimir Triplicado”.

6 Requerimientos Especiales

Problemas surgidos durante el desarrollo

Implementación del patrón de diseño "template method":

Al tener una clase abstracta en donde se encuentra el template method y los métodos abstractos que representan los pasos que debe seguir el algoritmo, tenemos la necesidad de que las clases hereden de ella para poder sobrescribir los métodos abstractos ya mencionados. Nuestro problema surge al querer heredar la clase abstracta, ya que nos topamos con una barrera del lenguaje Java, que no permite herencia múltiple, porque las clases que nos interesan que realicen una variación del algoritmo, ya están heredando de un JFrame/JDialog.

Nuestra solución a este problema fue inevitablemente crear una clase intermedia que sirva como nexo de la clase abstracta y de las clases que quieren utilizar el algoritmo. Para esto, la clase intermedia hereda la clase abstracta y es la encargada de sobrescribir los métodos abstractos. Las clases que utilizan el algoritmo, tienen una dependencia hacía la clase intermedia y modifican atributos mediante setters para que estos sean usados en los métodos abstractos sobrescritos por la clase intermedia.

Gran cantidad de consultas a la base de datos:

El problema surgía cuando necesitábamos realizar consultas para obtener información sobre, por ejemplo, la disciplina a la que pertenece una cuota o un pago específico. Como no teníamos una relación explícita entre las dos clases, sino que se conectaban por medio de clases intermedias, las consultas se volvían complicadas y lentas, teniendo que consultar por esas clases intermedias antes de llegar a la clase que nos interesaba.

Para solucionar este problema, decidimos agregar más asociaciones entre las clases que nos interesaban para facilitar la navegabilidad y que las consultas sean más directas y eficientes.

Conclusión del desarrollo del CE

Aunque se presentaron dificultades para comprender la realidad planteada por el cliente, estos obstáculos nos brindaron experiencia y aprendizaje sobre cómo es la comunicación con el mismo y sobre cómo afrontar cada reunión para sacarles más provecho. La inmersión en el desarrollo de software y la implementación de lo que se nos pedía desde la cátedra nos permitió adquirir un conocimiento más profundo sobre patrones de diseño y arquitectura en capas, lo que contribuyó significativamente a nuestra evolución como desarrolladores.

Además, desarrollar el software en Java proporcionó una base sólida y práctica, ampliando nuestras habilidades técnicas con el mismo, ya que tuvimos que investigar bastante sobre cómo funcionan varias cosas propias del lenguaje.

Aclaración: Este proyecto fue desarrollado de manera grupal junto a dos compañeros, Juan y Agustín, para la materia "Ingeniería de Software II" dictada en la UNSL.

Solo fueron implementados los casos de uso pedidos por la cátedra.