REPORTE DE ENTREGA

40% del sistema

28 DE MAYO DE 2018

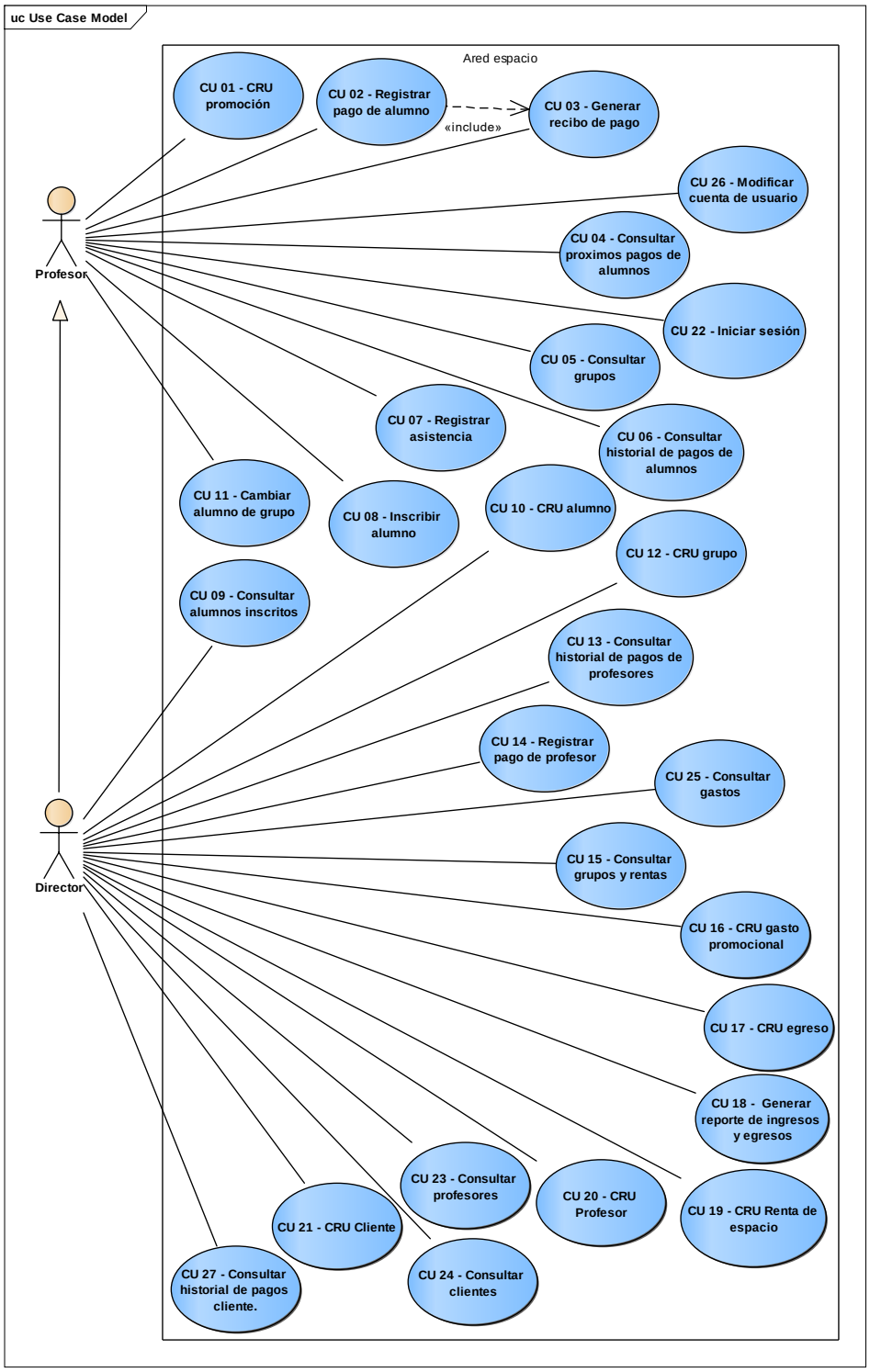
UNIVERSIDAD VERACRUZANA – DESARROLLO DE SOFTWARE
Mario Hurtado López – Víctor Javier García Mascareñas

Índice

1. Modelos de diseño.

En esta sección se presentan los modelos correspondientes al diseño del sistema. Para el caso de los diagramas de robustez, secuencia, clases y paquetes, se incluyen únicamente los elementos correspondientes al 40% del sistema implementado hasta ahora.

1.1. Modelo de casos de uso.



1.2. Descripciones de casos de uso.

Nombre del caso de uso.	CU-01 CRU promoción.
Actor.	Profesor, director.
Descripción.	En este caso de uso el profesor o director deberá poder crear y modificar promociones del sistema para cobros de los alumnos ya sea para pago de inscripción o mensualidad.
Precondiciones.	1. El profesor o director debe estar autenticado. 2. En caso de crearse una promoción, esta no debe existir previamente en el sistema.
Postcondiciones.	1. Debe crearse una promoción en el sistema. 2. La promoción creada solo puede ser vista por el usuario que la creó.
Flujo normal.	1. El usuario del sistema selecciona promociones. 2. El sistema muestra la ventana general de promociones existentes. 3. El usuario selecciona agregar promoción. 4. El sistema solicita los datos necesarios para crear la promoción (nombre y descripción) y selecciona guardar. 5. El usuario ingresa el nombre, descripción, porcentaje y selecciona registrar. 6. El sistema valida los datos del usuario, guarda la promoción y notifica la operación al usuario.
Flujo alterno.	Modificar promoción: 3.1. El usuario selecciona editar en una promoción existente. 4.1. El sistema muestra los detalles existentes de la promoción seleccionada. 5.1. El usuario modifica los datos necesarios y selecciona guardar cambios. 6.1. El sistema valida los datos del usuario, guarda la promoción y notifica la operación al usuario.
Excepciones.	No pueden almacenarse los datos 4.1. El sistema muestra al usuario que los datos no pueden ser guardados, que intente más tarde.

Nombre del caso de uso.	CU-02 Registrar pago del alumno.
Actor.	Profesor, director.
Descripción.	En este caso de uso el profesor o director deberá poder guardar un registro de un pago especificado quien es la fuente y el destinatario ya sea de inscripción o mensualidad.
Precondiciones.	1. El usuario debe estar autenticado en el sistema.

	2. Debe existir un registro del alumno al que se desea asignar el pago.
Postcondiciones.	1. Debe generarse un registro del pago del alumno. 2. Debe eliminarse cualquier notificación de próximos pagos en esa semana para ese alumno.
Flujo normal.	1. El usuario selecciona registrar pago. 2. El sistema muestra los datos necesarios para el registro. 3. El usuario selecciona el nombre del estudiante, ingresa la cantidad de mensualidad y selecciona registrar. 4. El sistema valida los datos ingresados, crea un registro y notifica al usuario que los datos fueron creados exitosamente.
Flujo alterno.	Registrar pago con promoción: 2.1. El usuario selecciona el nombre del estudiante, ingresa la cantidad de mensualidad y selecciona promociones. 2.2. El sistema muestra todas las promociones existentes para ese usuario. 2.3. El usuario selecciona alguna promoción y aceptar. 2.4. El sistema valida los datos y notifica el registro exitoso al usuario.
Excepciones.	No pueden almacenarse los datos 4.1. El sistema muestra al usuario que los datos no pueden ser guardados, que intente más tarde.

Nombre del caso de uso.	CU 05 – Consultar grupos.
Actor.	Profesor.
Descripción.	En este caso de uso, el profesor puede visualizar los cursos que imparte.
Precondiciones.	1. El profesor debe tener al menos un curso registrado.
Postcondiciones.	1. El profesor debe poder consultar los alumnos por grupo.
Flujo normal.	1. El profesor selecciona Mis grupos. 2. El sistema muestra los grupos del profesor.
Flujo alterno.	Seleccionar alumnos por grupo. 3. El profesor selecciona alumnos en un grupo 4. El sistema muestra los alumnos inscritos al curso seleccionado.
Excepciones.	No se pueden obtener los alumnos o los cursos.

	2.1. El sistema muestra un mensaje informando sobre el error. 4.1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 09 – Consultar alumnos.
Actor.	Director.
Descripción.	El director puede consultar todos los alumnos existentes en la institución.
Precondiciones.	1. Deben existir alumnos inscritos en la institución. 2. El director tiene que estar autenticado en el sistema.
Postcondiciones.	
Flujo normal.	1. El director selecciona alumnos en la institución. 2. El sistema busca los alumnos existentes en el sistema y los muestra al director.
Flujo alterno.	No hay alumnos inscritos. 1.1. El sistema muestra al director que no existen alumnos en la institución. 1.2. El sistema regresa a una pantalla anterior del sistema.
Excepciones.	No existe conexión a la base de datos 1. El sistema muestra al usuario que no hay conexión con la base de datos, que intente nuevamente más tarde.

Nombre del caso de uso.	CU 10 – CRU Alumno.
Actor.	Director.
Descripción.	En este caso de uso, el director podrá administrar la información de los alumnos creando y modificando los registros.
Precondiciones.	
Postcondiciones.	1. Los profesores podrán ver los cambios y el director podrá ver los nuevos registros.
Flujo normal.	1. El director selecciona Registrar alumno. 2. El sistema muestra la ventana registro de alumno. 3. El director ingresa el nombre, teléfono, dirección, correo, selecciona una imagen y selecciona Registrar. 4. El sistema valida los campos, guarda el registro y muestra un mensaje de confirmación.
Flujo alterno.	Modificar alumno. 1.1. El profesor selecciona Editar alumno. 2.2. El sistema muestra la ventana de edición de alumno 3.1. El director modifica los campos deseados y selecciona Guardar.

	4.1. El sistema valida los campos, actualiza el registro y muestra un mensaje de confirmación. Los campos no son válidos. 4.2. El sistema muestra un mensaje informando el error y mencionando la acción a realizar.
Excepciones.	No se pueden guardar/actualizar los datos. 4.1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 12 – CRU Grupo.
Actor.	Director.
Descripción.	En este caso de uso, el director podrá crear nuevos grupos, así como modificar los ya existentes.
Precondiciones.	
Postcondiciones.	1. Los usuarios del sistema podrán ver los nuevos grupos y las modificaciones a los mismos.
Flujo normal.	1. El director selecciona Nuevo grupo. 2. El sistema despliega la ventana de registro de grupo. 3. El director selecciona el profesor, el horario, el salón, ingresa el nombre y selecciona Crear. 4. El sistema valida el horario, guarda el registro y muestra un mensaje de confirmación.
Flujo alterno.	Editar grupo. 1.1. El director selecciona editar grupo. 2.1. El sistema despliega la ventana de edición de grupo. 3.1. El director modifica los campos deseados y selecciona Guardar. 4.1. El sistema valida el horario, actualiza el registro y muestra un mensaje de confirmación. El horario choca. 4.2. El sistema muestra un mensaje informando que el horario seleccionado crea conflicto con otro grupo o renta.
Excepciones.	No se pueden guardar/actualizar los datos. 4.3. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 13 – Consultar historial de pago de profesores.

Actor.	Director.
Descripción.	En este caso de uso, el director podrá revisar el historial de los pagos realizados por los profesores.
Precondiciones.	1. Debe existir al menos un pago de un profesor registrado.
Postcondiciones.	1. El director debe poder ver la información de los registros de pago.
Flujo normal.	1. El director selecciona Pagos realizados. 2. El sistema muestra la lista de pagos realizados del profesor seleccionado.
Flujo alterno.	No hay pagos realizados. 2.1. El sistema muestra un mensaje informando que no existen pagos realizados por parte del profesor.
Excepciones.	No se pueden obtener los datos. 2.2. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 14 – Registrar pago de profesor.
Actor.	Director.
Descripción.	El director deberá guardar un registro de un pago realizado por el profesor.
Precondiciones.	1. Deben existir profesores registrados en el sistema. 2. Debe estar autenticado el director en el sistema.
Postcondiciones.	1. Debe generarse un registro de pago en el sistema. 2. Debe mostrar el pago del profesor en el historial de pagos realizado.
Flujo normal.	1. El director selecciona registrar pago. 2. El sistema solicita el nombre del profesor. 3. El director selecciona el nombre del profesor y selecciona aceptar. 4. El sistema valida las entradas, guarda los datos y notifica al director que el pago se ha realizado exitosamente.
Flujo alterno.	Existe un pago previo de ese mes / quincena. 2.1. El sistema muestra un mensaje de que el pago del profesor en ese mes se ha realizado anteriormente 2.2. El sistema regresa al paso 2 del flujo normal.
Excepciones.	No existe conexión a base de datos del sistema 1. El sistema muestra al usuario que no existe conexión con el sistema, que intente más tarde.

Nombre del caso de uso.	CU 16 – CRU gasto promocional.

Actor.	Director.
Descripción.	El director podrá administrar los gastos promocionales de Facebook.
Precondiciones.	1. No debe existir el gasto promocional en un registro previo.
Postcondiciones.	1. Debe crearse un registro de promoción en el sistema.
Flujo normal.	1. El director selecciona gasto promocional. 2. El sistema solicita los detalles para generar un registro gasto promocional. 3. El director ingresa una descripción, un enlace, un monto, fecha inicio, fecha de fin y selecciona aceptar. 4. El sistema valida los datos ingresados al sistema, guarda los datos y notifica al director el registro exitoso.
Flujo alterno.	Editar gasto 1.1. El director selecciona editar 2.1. El sistema muestra la ventana de edición de gastos 3.1. El director selecciona las fechas de inicio, fin y selecciona guardar. 4.1. El sistema actualiza el registro y notifica al director el registro. Los datos son incorrectos 3.1. el sistema muestra un mensaje al usuario que los datos son incorrectos. 4.1 el flujo regresa al paso 2 del flujo normal.
Excepciones.	No existe conexión a base de datos del sistema 1. El sistema muestra al usuario que no existe conexión con el sistema, que intente más tarde.

Nombre del caso de uso.	CU 17 – CRU Egreso.
Actor.	Director.
Descripción.	En este caso de uso, el director puede administrar los egresos de la institución.
Precondiciones.	
Postcondiciones.	1. El director debe poder ver el registro realizado.
Flujo normal.	1. El director selecciona registrar egreso. 2. El sistema muestra la ventana de registro de egreso. 3. El director selecciona la fecha, ingresa una descripción, el costo y selecciona registrar. 4. El sistema valida los datos, registra el gasto y muestra un mensaje de confirmación.

Flujo alterno.	Editar egreso 1.1. El director selecciona editar. 2.1. El sistema muestra la ventana de edición de egreso. 3.1. El director selecciona una fecha y guardar. 4.1. El sistema actualiza el registro y muestra el mensaje de confirmación al director. Los datos son incorrectos 3.1. el sistema muestra un mensaje al usuario que los datos son incorrectos. 4.1 el flujo regresa al paso 2 del flujo normal.
Excepciones.	No se puede guardar el registro. 4.2. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 20 – CRU profesor.
Actor.	Director.
Descripción.	En este caso de uso el director podrá crear un nuevo registro de un profesor o en su defecto actualizar los datos de un profesor existente en el sistema.
Precondiciones.	1. Para crear un nuevo registro el profesor que se desea almacenar no debe encontrarse dentro de los datos existentes.
Postcondiciones.	1. Debe crearse un nuevo registro de un profesor en el sistema. 2. En su defecto debe crearse una actualización de los datos del profesor seleccionado.
Flujo normal.	1. El director selecciona agregar profesor. 2. El sistema muestra todos los datos que deben ser cubiertos por el profesor para crear el registro. 3. El director ingresa los datos del nuevo profesor como su nombre, dirección, teléfono, correo, monto, tipo de pago, monto y selecciona guardar. 4. El sistema verifica la validez de los datos, crea un registro y notifica al director que el nuevo profesor ha sido creado con éxito.
Flujo alterno.	Editar profesor. 1.1. El director selecciona editar profesor. 2.1. El sistema muestra los datos del usuario seleccionado. 3.1. El director modifica los datos del usuario que se ha seleccionado y que considera pertinentes y selecciona guardar.

	4.1. El sistema valida los datos, guarda el registro y confirma al director que la operación se realizó con éxito. Los datos no son válidos. 4.2. El sistema muestra un mensaje indicando el error y como proceder.
Excepciones.	No se pueden guardar/actualizar los datos. 1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 21 – CRU cliente.
Actor.	Director.
Descripción.	En este caso de uso el director podrá crear un nuevo registro de un cliente o actualizar los datos de un cliente con un registro previo.
Precondiciones.	1. Para crear un nuevo registro el cliente que se desea almacenar no debe encontrarse dentro de los datos existentes.
Postcondiciones.	1. Debe crearse un nuevo registro de un cliente en el sistema. 2. En su defecto debe crearse una actualización de los datos del cliente seleccionado.
Flujo normal.	1. El director selecciona agregar cliente. 2. El sistema muestra todos los datos que deben ser cubiertos por el director para crear el registro. 3. El director ingresa los datos del nuevo cliente como su nombre, dirección, teléfono, correo y selecciona guardar. 4. El sistema verifica la validez de los datos, crea un registro y notifica al director que el nuevo cliente ha sido creado con éxito.
Flujo alterno.	Editar cliente. 1.1. El director selecciona editar cliente. 2.1. El sistema muestra los datos del cliente seleccionado. 3.1. El director modifica los datos del cliente que se ha seleccionado y que considera pertinentes y selecciona guardar. 4.1. El sistema valida los datos, guarda el registro y confirma al director que la operación se realizó con éxito. Los datos no son válidos.

	4.2. EL sistema muestra un mensaje indicando el error y cómo proceder.
Excepciones.	No se pueden guardar/actualizar los datos. 1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU-22 Iniciar sesión.
Actor.	Director, profesor.
Descripción.	En este caso de uso el usuario ingresa al sistema dependiendo los privilegios proporcionados en su registro.
Precondiciones.	1. El usuario debe existir en un registro previo.
Postcondiciones.	1. El usuario deberá visualizar todos los contenidos relacionados con su registro previo.
Flujo normal.	1. El sistema muestra al usuario los datos que son necesarios para ingresar al sistema. 2. El usuario ingresa su nombre de usuario, contraseña y selecciona aceptar. 3. El sistema valida los datos verifica que existan registros previos e ingresa a la pantalla de inicio de ese usuario.
Flujo alterno.	El usuario no se encuentra en los registros del sistema. 3.1. El sistema muestra un mensaje al usuario que el registro no existe o es una combinación incorrecta de usuario contraseña, que intente nuevamente. 3.2. El flujo del sistema regresa al paso 1 del flujo normal.
Excepciones.	No existe conexión con la base de datos. 1. El sistema muestra un mensaje al usuario de que no es posible acceder a la base de datos que intente mas tarde.

Nombre del caso de uso.	CU 23 – consultar profesores.
Actor.	Director.
Descripción.	En este caso de uso el director puede consultar todos los profesores existentes en el sistema.
Precondiciones.	1. El director debe tener una sesión iniciada en el sistema. 2. Deben existir profesores registrados en el sistema.
Postcondiciones.	
Flujo normal.	1. El director selecciona consultar profesores. 2. El sistema busca todos los profesores existentes en el sistema y los muestra al director.
Flujo alterno.	El director busca a un profesor especifico

	1.1. El sistema solicita un criterio de búsqueda para el profesor que se desea consultar. 1.2. El director ingresa el nombre del profesor que desea y selecciona buscar. 1.3. El sistema muestra todos los resultados coincidentes con el criterio de búsqueda.
Excepciones.	No puede consultarse los datos en la base del sistema, 1. El sistema muestra un mensaje sobre la conexión y le solicita que intente más tarde.

Nombre del caso de uso.	CU 24 – consultar clientes.
Actor.	Director.
Descripción.	En este caso de uso el director puede consultar todos los profesores clientes registrados en el sistema.
Precondiciones.	3. El director debe tener una sesión iniciada en el sistema. 4. Deben existir clientes registrados en el sistema.
Postcondiciones.	
Flujo normal.	3. El director selecciona consultar clientes. 4. El sistema busca todos los clientes existentes en el sistema y los muestra al director.
Flujo alterno.	El director busca a un cliente especifico 1.4. El sistema solicita un criterio de búsqueda para el cliente que se desea consultar. 1.5. El director ingresa el nombre del cliente que desea y selecciona aceptar. 1.6. El sistema muestra todos los resultados coincidentes con el criterio de búsqueda.
Excepciones.	No puede consultarse los datos en la base del sistema, 2. El sistema muestra un mensaje sobre la conexión y le solicita que intente más tarde.

Nombre del caso de uso.	CU 25 – Consultar gastos.
Actor.	Director.
Descripción.	En este caso de uso, el director puede consultar todos los gastos realizados de promociones de Facebook y egresos.
Precondiciones.	1. Debe de existir al menos un gasto/promoción registrado.
Postcondiciones.	1. El director debe poder ver los registros.
Flujo normal.	1. El director selecciona todos los gastos 2. El sistema muestra la selección del periodo de consulta. 3. El director selecciona el periodo.

	4. El sistema muestra dos listas, una con todas las promociones de Facebook y otra con todos los egresos del periodo seleccionado.
Flujo alterno.	No hay gastos registrados. 2.1. El sistema muestra un mensaje informando que no existen gastos registrados.
Excepciones.	No se pueden obtener los registros. 1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 26 – Modificar cuenta de usuario.
Actor.	Profesor.
Descripción.	El este caso de uso el profesor podrá modificar sus datos de usuario y contraseña del sistema.
Precondiciones.	5. Deben existir un registro previo del profesor que desea modificar los datos. 6. No debe existir un usuario con el nombre indicado por el profesor.
Postcondiciones.	1. Deben actualizase los registros de usuario y contraseña de ese profesor. 2. Los datos ingresados por el usuario deben ser diferentes a los establecidos en una sesión anterior.
Flujo normal.	5. El profesor selecciona modificar cuenta. 6. El sistema muestra la ventana de modificar cuenta. 7. El profesor ingresa un nuevo nombre de usuario, contraseña, la confirmación de esta y selecciona guardar cambios. 8. El sistema verifica, guarda el registro y notifica al profesor que sus datos fueron actualizados correctamente.
Flujo alterno.	El usuario esta registrado previamente. 4.1. El sistema muestra al director que el usuario ya existe en el sistema, la ejecución regresa al paso 2 del flujo normal.
Excepciones.	No existe conexión con la base de datos. 3. El sistema muestra un mensaje sobre la conexión y le solicita que intente más tarde.

Nombre del caso de uso.	CU 07 – Registrar asistencia.
Actor.	Profesor, director.
Descripción.	En este caso de uso, el profesor o director podrá asignar la asistencia de los alumnos de alguno de sus grupos.

Precondiciones.	1. El profesor debe tener al menos un grupo asignado con al menos un alumno inscrito.
Postcondiciones.	1. Se debe guardar el registro de la asistencia a la sesión.
Flujo normal.	1. El usuario selección Registrar asistencia, marca a los alumnos que asistieron a la sesión y selecciona Guardar. 2. El sistema registra los datos y muestra un mensaje de confirmación.
Flujo alterno.	Se desea marcar todos los alumnos. 1.1. El usuario selecciona Marcar todos. 1.1.1. El sistema marca a todos los alumnos. 1.1.2. El usuario selecciona Guardar. 2.1. El sistema registra los datos y muestra un mensaje de confirmación.
Excepciones.	No se pueden guardar los datos. 2.1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU – 08 inscribir alumno.
Actor.	Profesor.
Descripción.	El profesor deberá poder registrar un nuevo alumno a algún curso existente relacionado con el profesor.
Precondiciones.	1. El profesor debe estar autenticado en el sistema. 2. Debe existir el alumno en un registro previo. 3. Deben existir cursos vigentes en el sistema
Postcondiciones.	1. Debe crearse un registro del alumno en algún curso.
Flujo normal.	1. El profesor selecciona inscribir alumno. 2. El sistema solicita curso y los criterios de búsqueda para inscribir el alumno. 3. El profesor selecciona un curso, ingresa el nombre del alumno y selecciona aceptar. 4. El sistema muestra todos los resultados resultantes de los criterios de búsqueda. 5. El profesor selecciona algún alumno encontrado y selecciona inscribir. 6. El sistema valida los datos, guarda el registro y notifica al profesor el éxito de la operación.
Flujo alterno.	No hay grupos registrados 1.1. El sistema muestra al profesor que no hay grupos disponibles en ese momento, que verifique los datos.
Excepciones.	No existe conexión a base de datos 4. El sistema muestra al profesor que no existe conexión con los datos del sistema, que intente más tarde.

Nombre del caso de uso.	CU 11 – Cambiar alumno de grupo
Actor.	Profesor.
Descripción.	El profesor puede reasignar el registro actual de un alumno a otro grupo.
Precondiciones.	1. El profesor debe estar autenticado en el sistema. 2. Deben existir alumnos inscritos en al menos un curso. 3. El alumno a cambiar de grupo no debe estar inscrito al grupo que desea cambiarse. 4. Deben existir grupos vigentes en el sistema.
Postcondiciones.	1. Debe modificarse el grupo a que existe el alumno seleccionado.
Flujo normal.	1. El profesor selecciona cambiar alumno. 2. El sistema muestra los grupos a los que pertenece el alumno, así como los grupos existentes en ese momento. 3. El profesor selecciona un grupo de procedencia, un grupo al que desea mover el registro y selecciona aceptar. 4. El sistema verifica las entradas del director, actualiza los datos y muestra una notificación.
Flujo alterno.	
Excepciones.	No existe conexión a la base de datos 2. El sistema muestra al usuario que no hay conexión con la base de datos, que intente nuevamente más tarde.

Nombre del caso de uso.	CU 15 – Consultar grupos y rentas.
Actor.	Director.
Descripción.	En este caso de uso, el director podrá visualizar todas las rentas, los grupos y sus respectivos alumnos.
Precondiciones.	1. Debe existir al menos un grupo/renta registrado.
Postcondiciones.	1. El director debe poder ver todos los grupos y rentas existentes.
Flujo normal.	1. El director selecciona Grupos y rentas. 2. El sistema muestra una lista con todos los grupos y rentas existentes.
Flujo alterno.	No hay grupos ni rentas. 2.1. El sistema muestra un mensaje informando que no existen grupos y rentas.
Excepciones.	No se pueden obtener los datos.

	2.2. El sistema muestra un mensaje informando sobre el error.

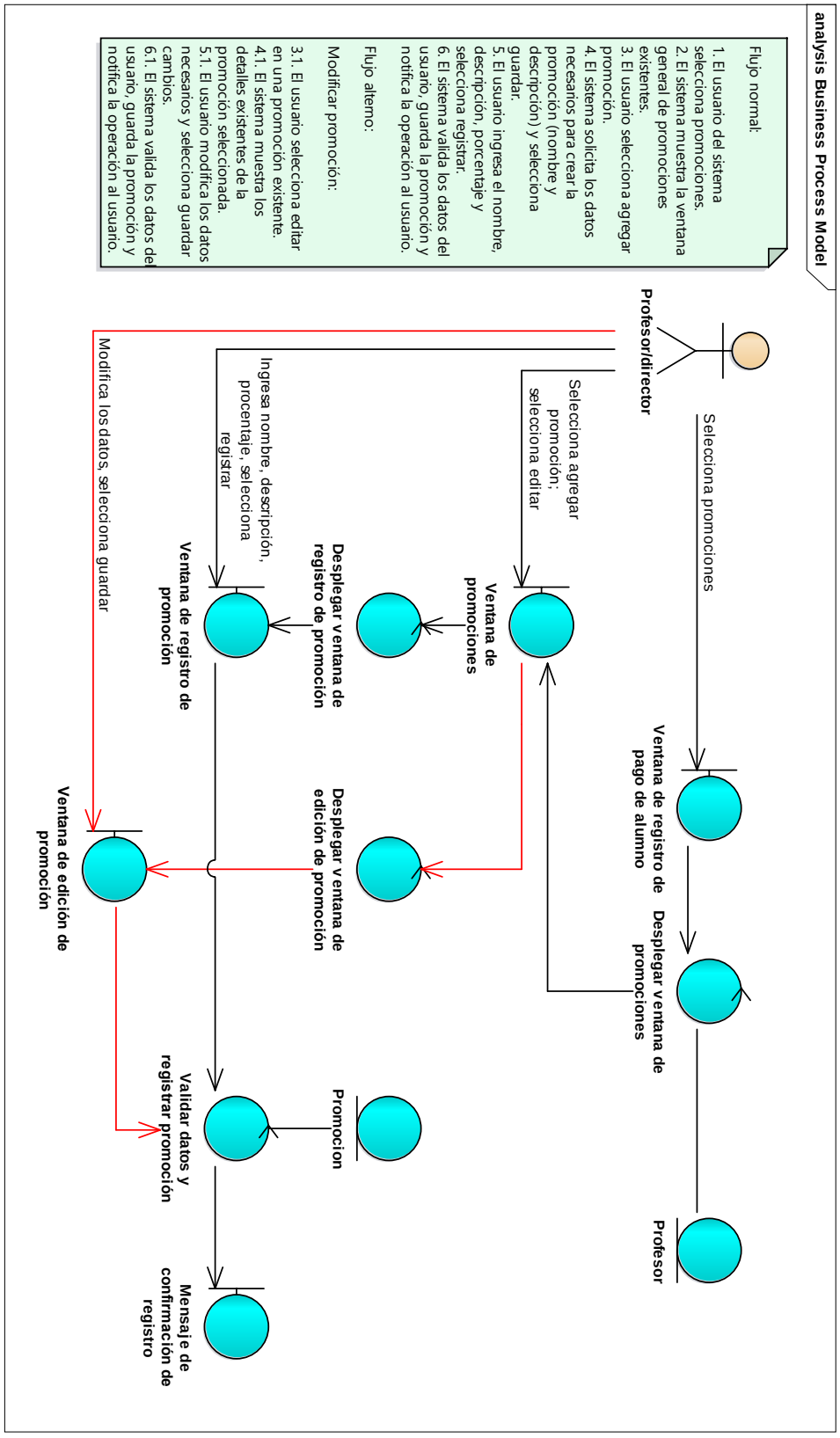
Nombre del caso de uso.	CU 18 – Generar reporte de ingresos y egresos.
Actor.	Director.
Descripción.	En este caso de uso, el director puede todos los movimientos mensuales y exportar un reporte general.
Precondiciones.	1. Debe haber pasado al menos un mes de actividad.
Postcondiciones.	1. El director debe poder ver todos los movimientos mensuales.
Flujo normal.	1. El director selecciona ver reporte mensual. 2. El sistema despliega la ventana de reporte. 3. El director selecciona el periodo. 4. El sistema muestra todos los movimientos del mes.
Flujo alterno.	Exportar reporte. 5. El profesor selecciona Exportar. 6. El sistema guarda el reporte en formato PDF y muestra un mensaje de confirmación.
Excepciones.	No se pueden obtener los datos. 4.1. El sistema muestra un mensaje informando sobre el error. No se puede guardar el reporte. 6.1. El sistema muestra un mensaje informando sobre el error.

Nombre del caso de uso.	CU 19 – CRU Renta de espacio.
Actor.	Director.
Descripción.	En este caso de uso, el director podrá administrar las rentas de los clientes.
Precondiciones.	
Postcondiciones.	1. El director podrá ver y editar los registros de renta.
Flujo normal.	1. El director selecciona rentar. 2. El sistema muestra la ventana de renta. 3. El director selecciona el cliente, el horario, el salón, el monto y selecciona registrar. 4. El sistema valida el horario, guarda el registro, el pago y muestra un mensaje de confirmación.
Flujo alterno.	Editar renta.

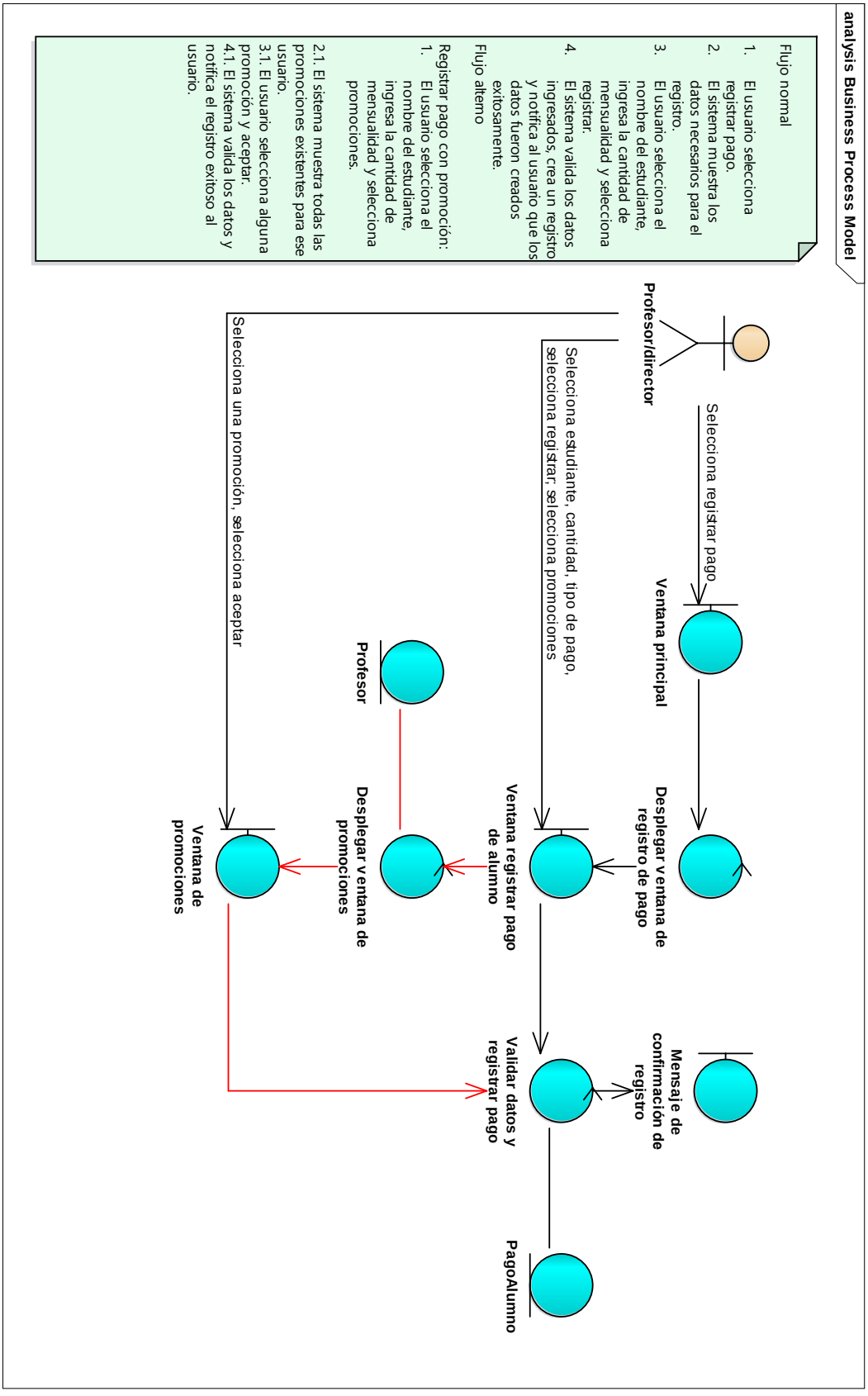
	1.1. El director selecciona Editar. 2.1. El sistema muestra la ventana de edición de renta. 3.1. El director modifica el horario, el salón y selecciona Guardar. 4.1. El sistema valida el horario, actualiza el registro y muestra un mensaje de confirmación. El horario choca. 4.2. El sistema muestra un mensaje informando que el horario causa conflicto con otra renta o grupo.
Excepciones.	No se puede actualizar/guardar el registro. 4.3. El sistema muestra un mensaje informando sobre el error.

1.3. Diagramas de robustez.

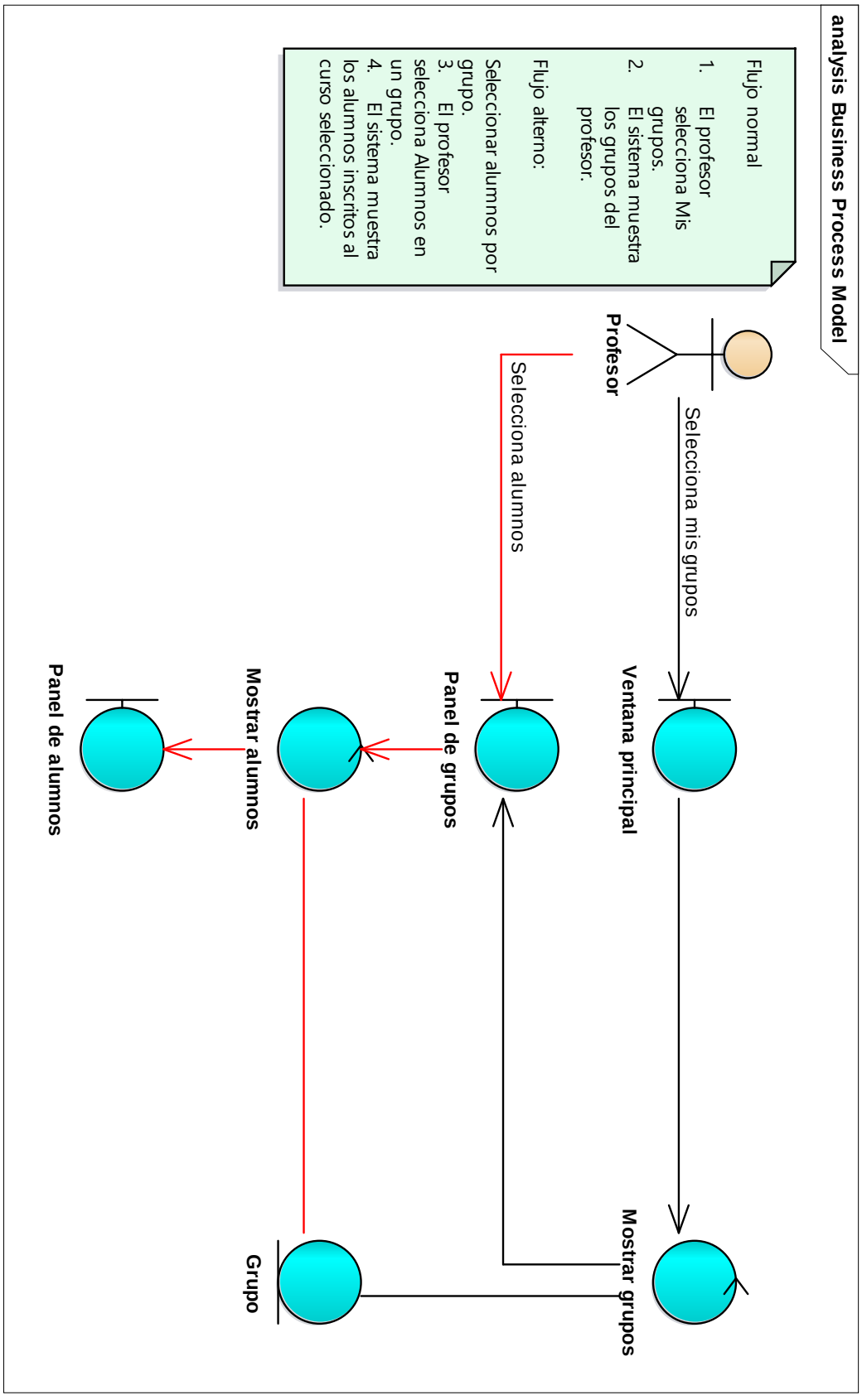
1.3.1. CU 01 - CRU promoción.



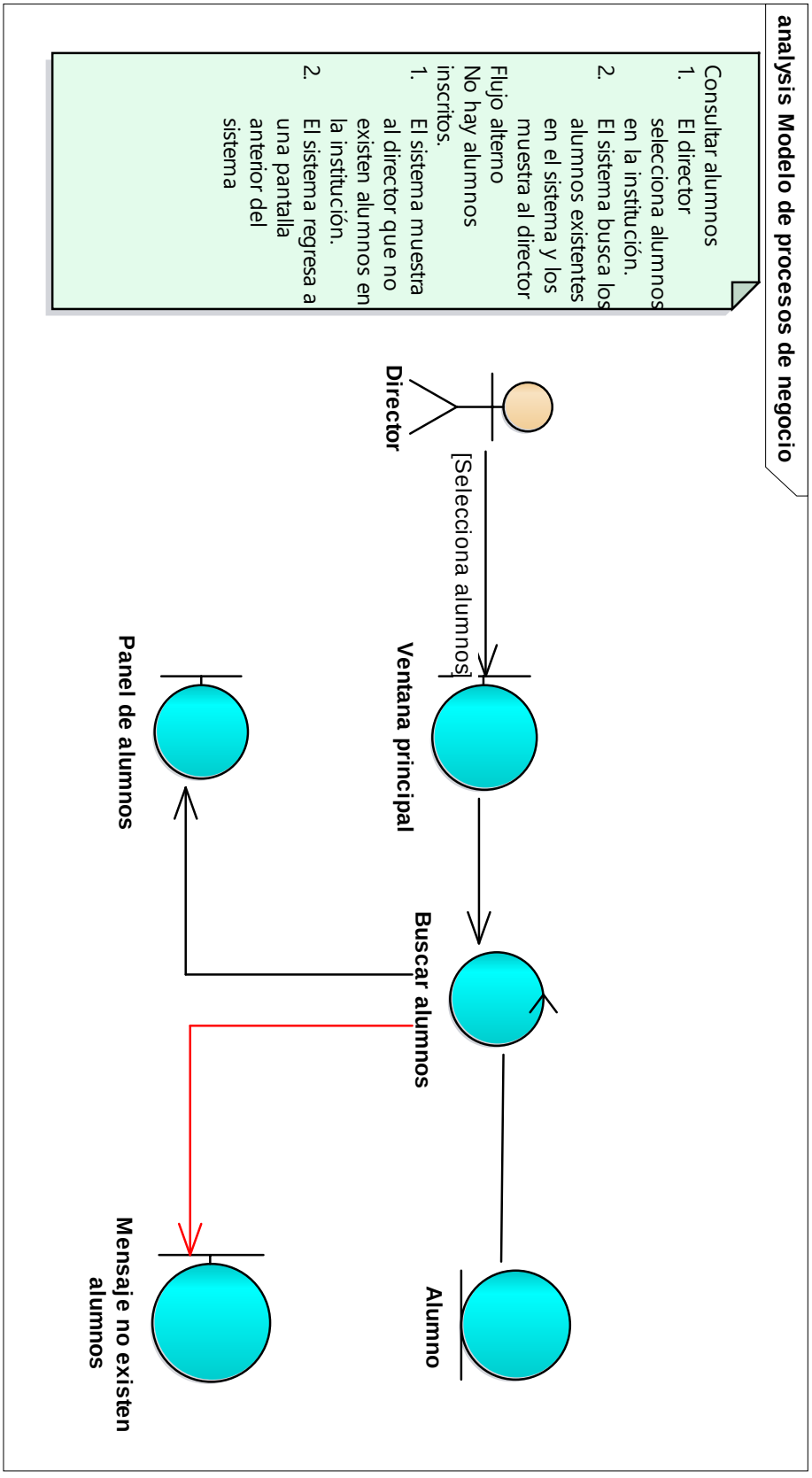
1.3.2. CU 02 - Registrar pago del alumno.



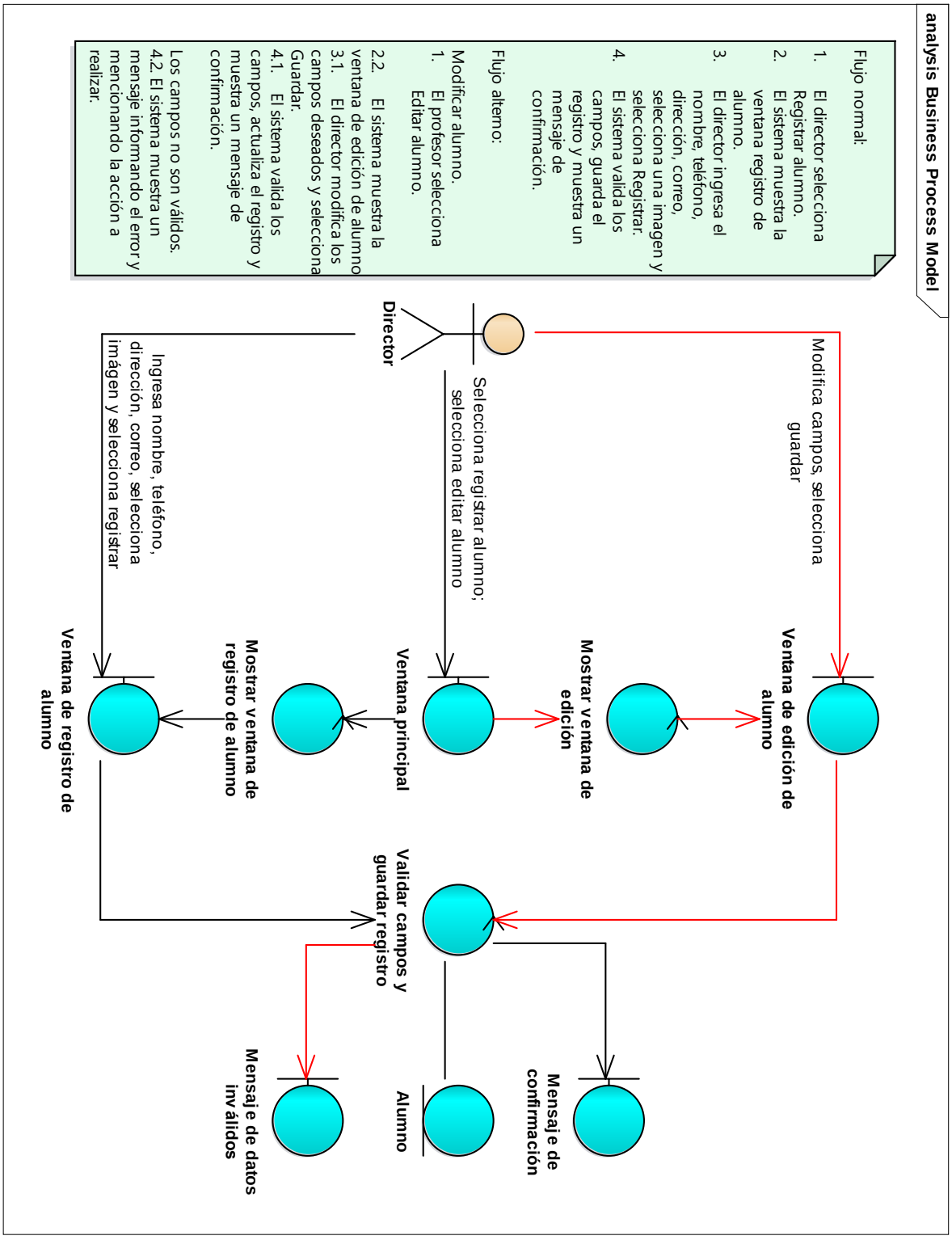
1.3.3. CU 05 – Consultar grupos.



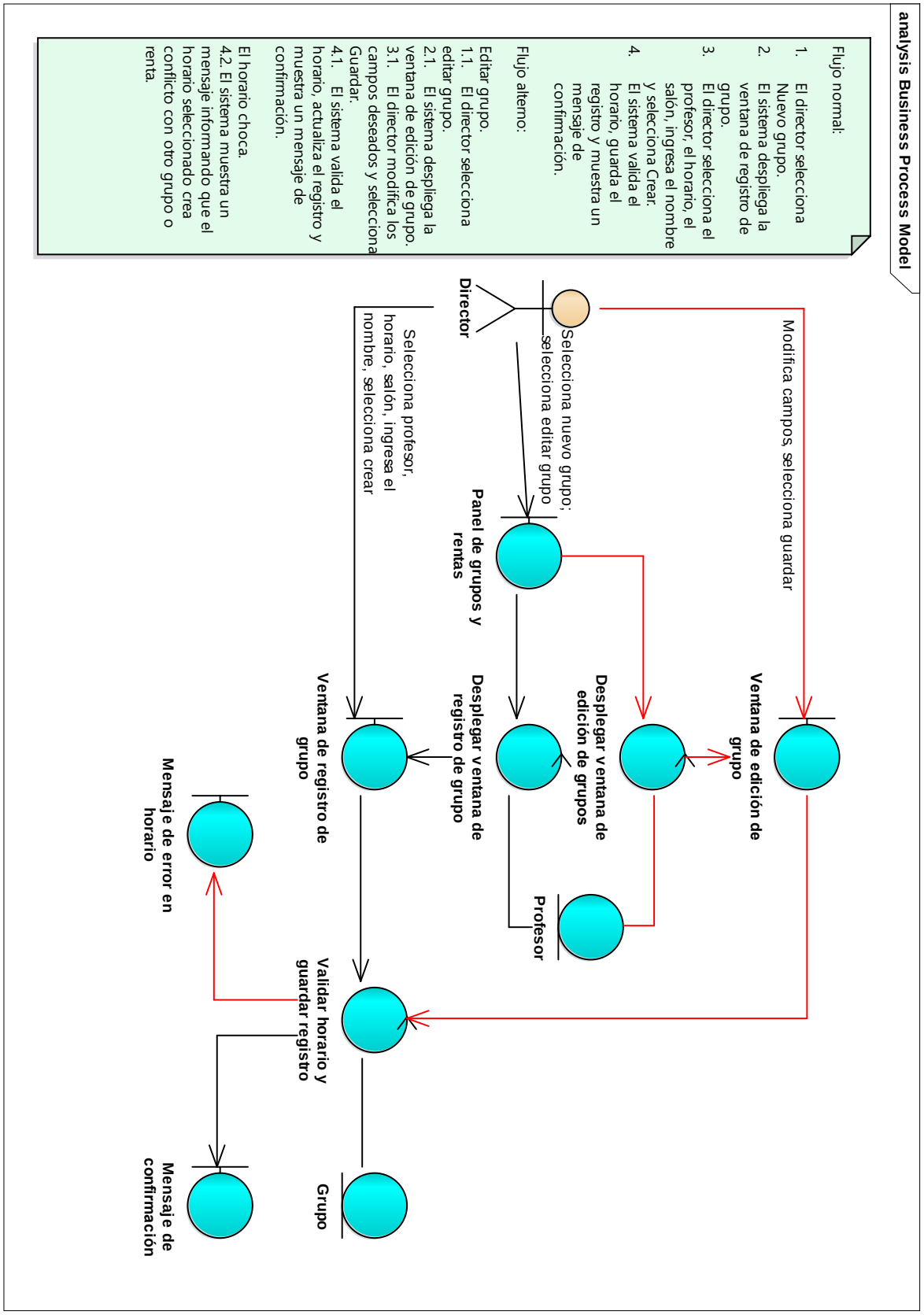
1.3.4. CU 09 – Consultar alumnos.



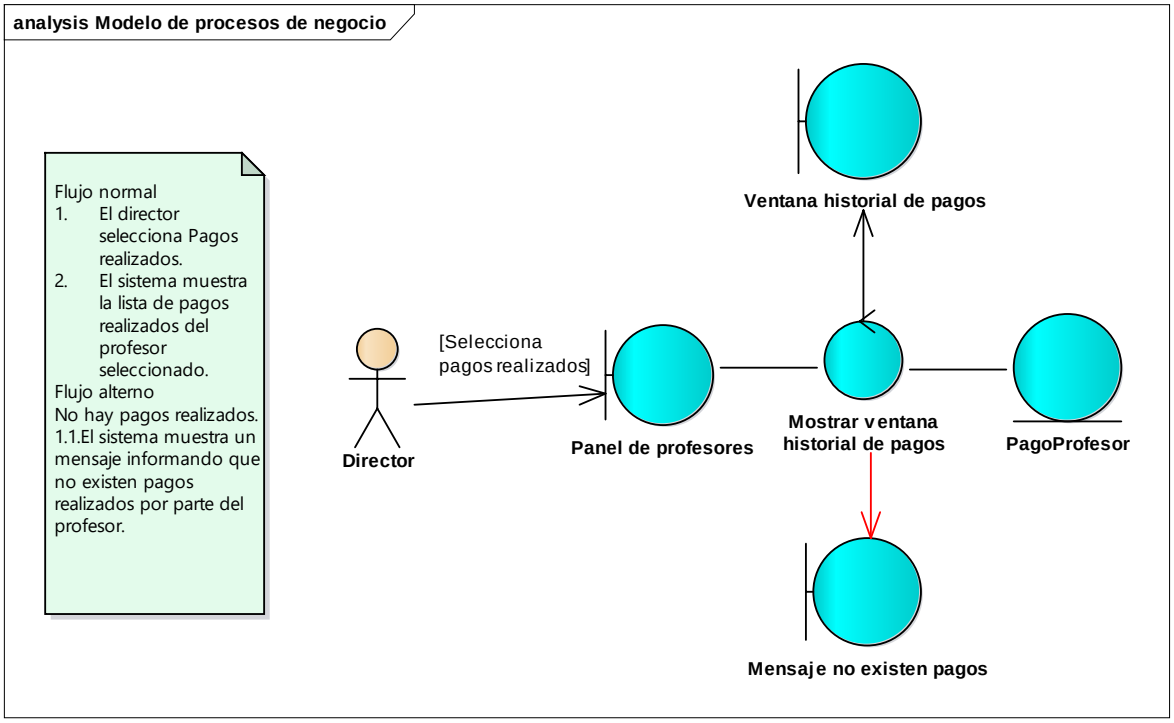
1.3.5. CU 10 – CRU Alumno.



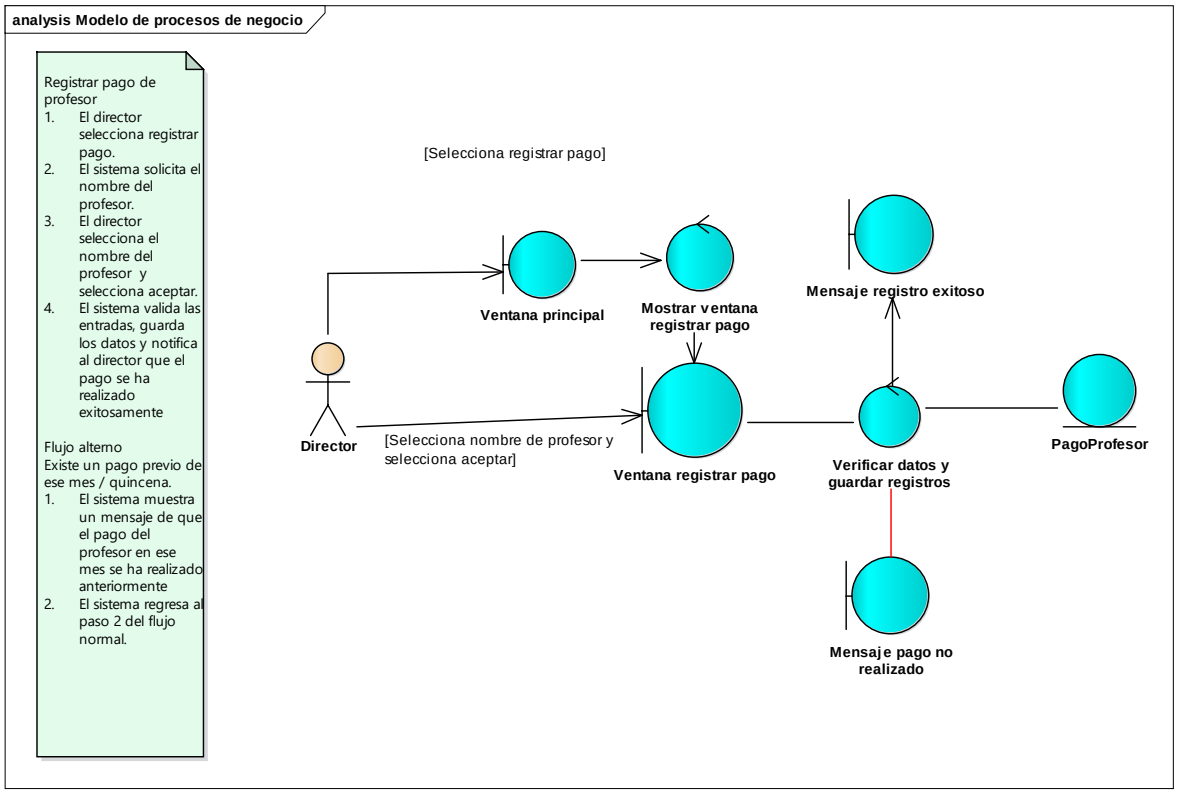
1.3.6. CU 12 – CRU Grupo.



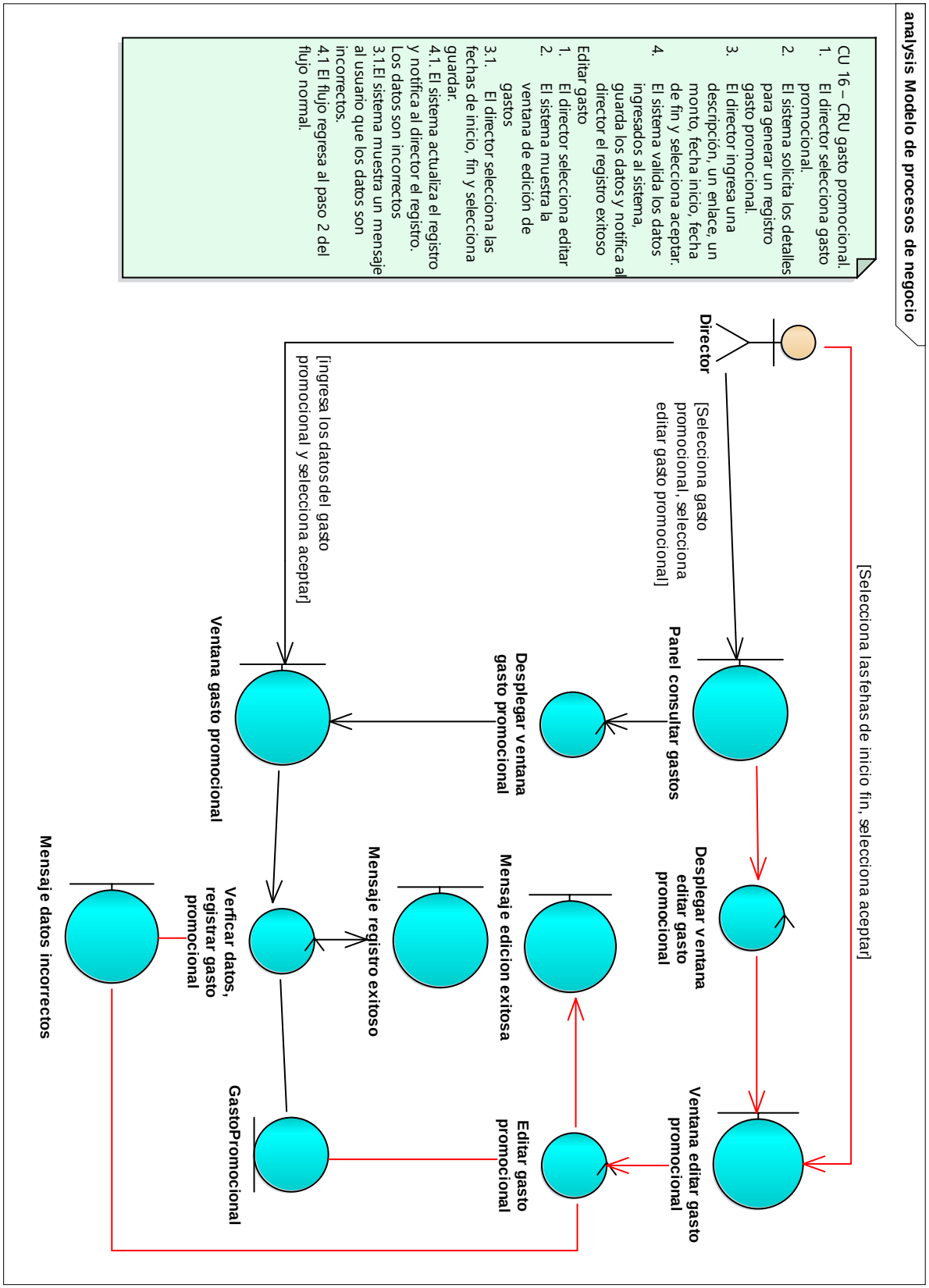
1.3.7. CU 13 – Consultar historial de pago de profesores.



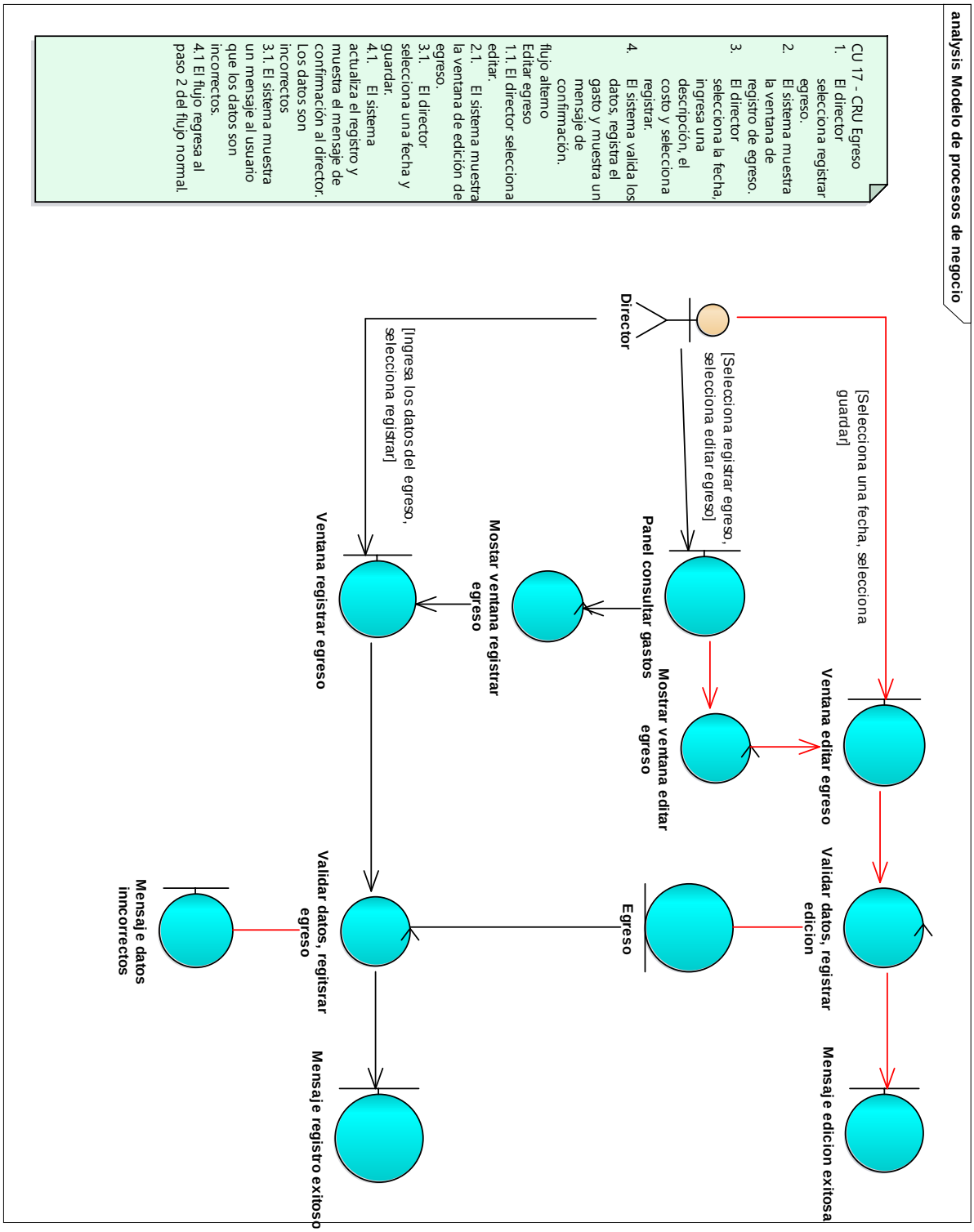
1.3.8. CU 14 – Registrar pago de profesor.



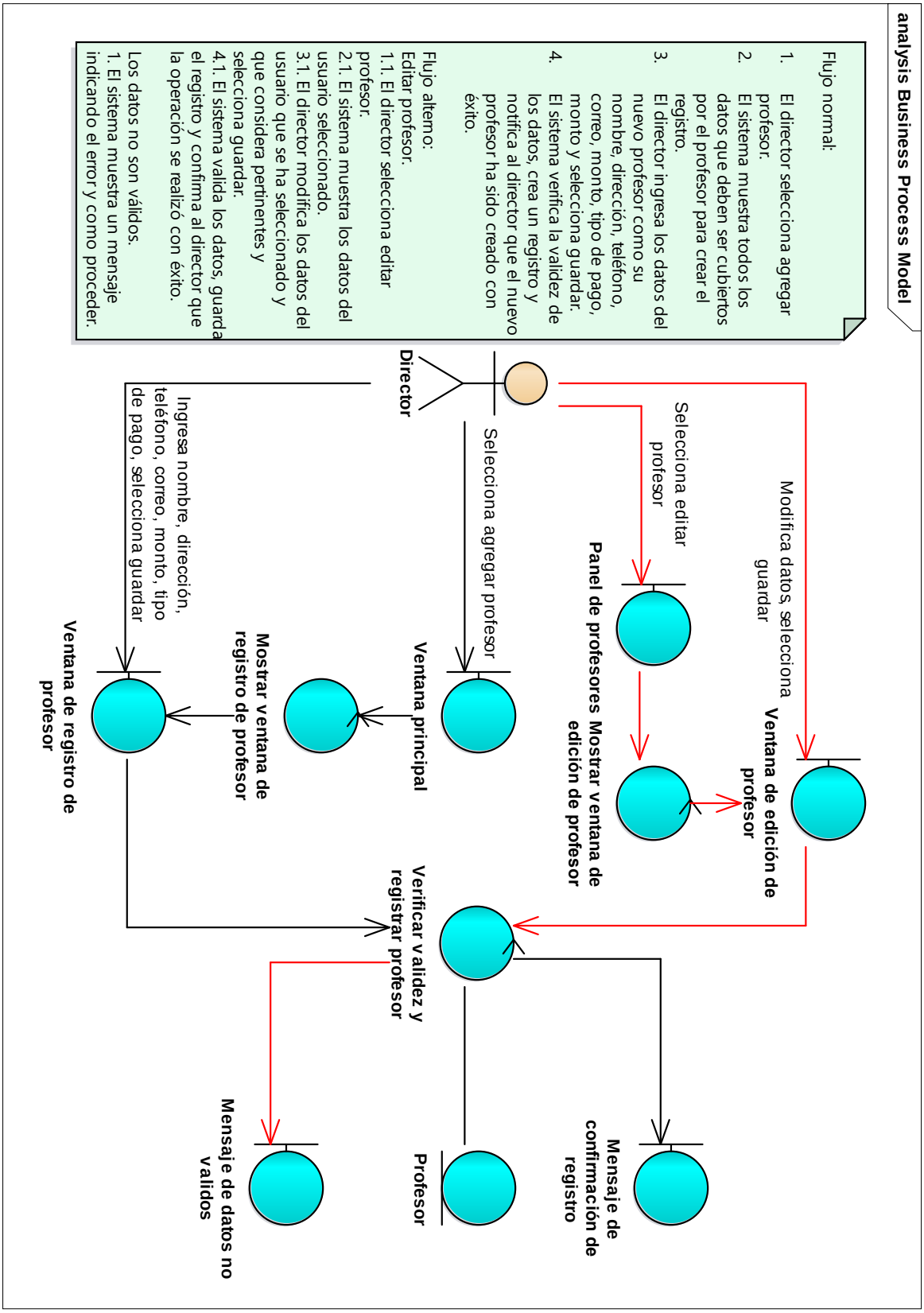
1.3.9. CU 16 – CRU gasto promocional.



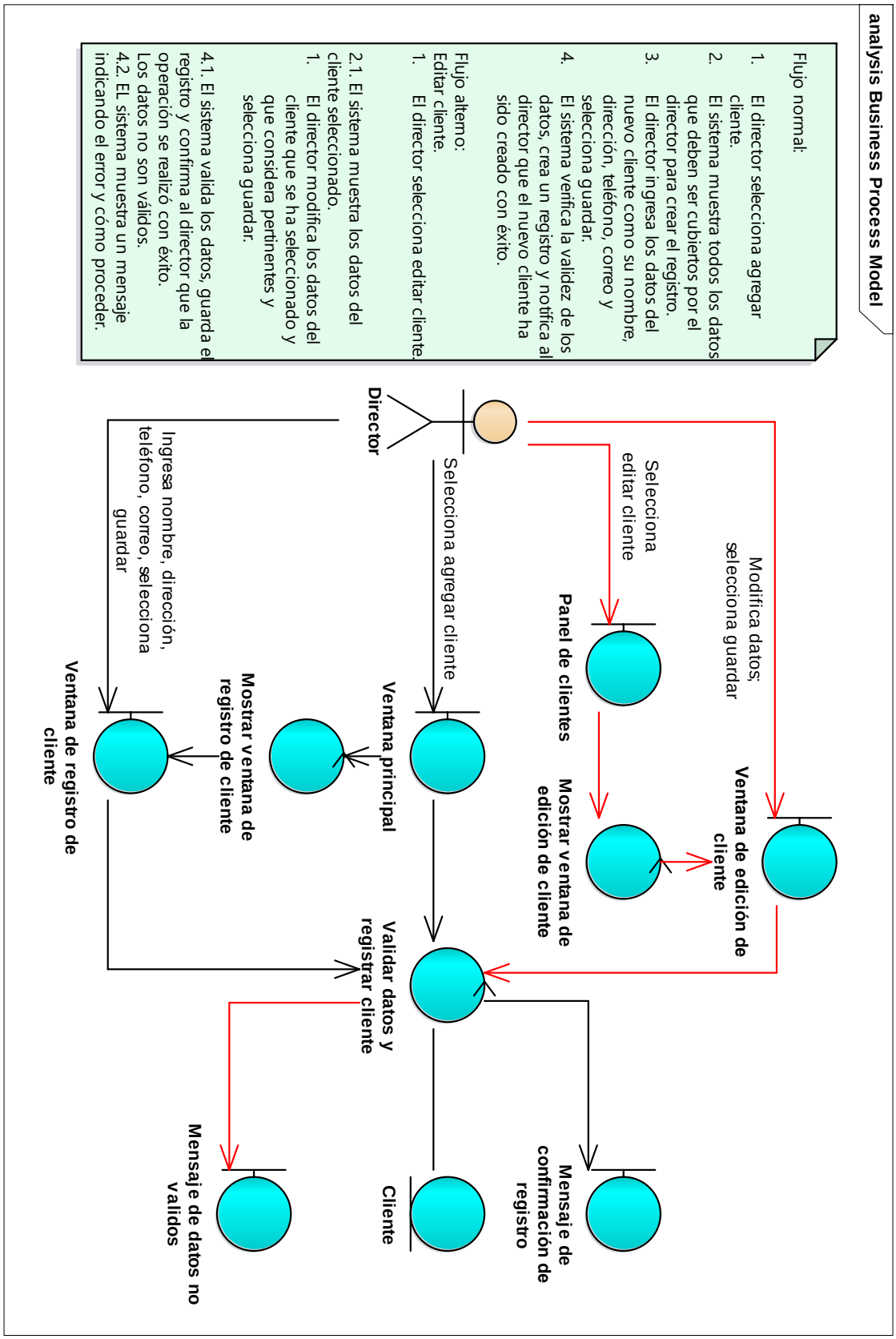
1.3.10. CU 17 – CRU Egreso.



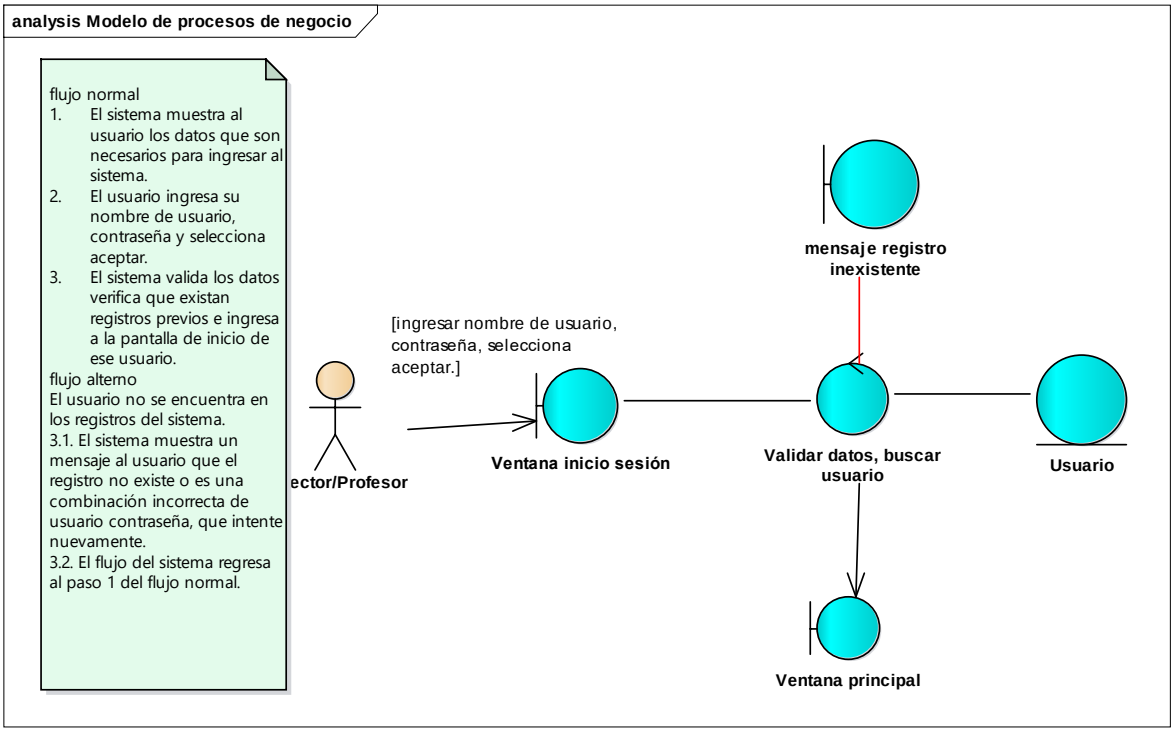
1.3.11. CU 20 – CRU profesor.



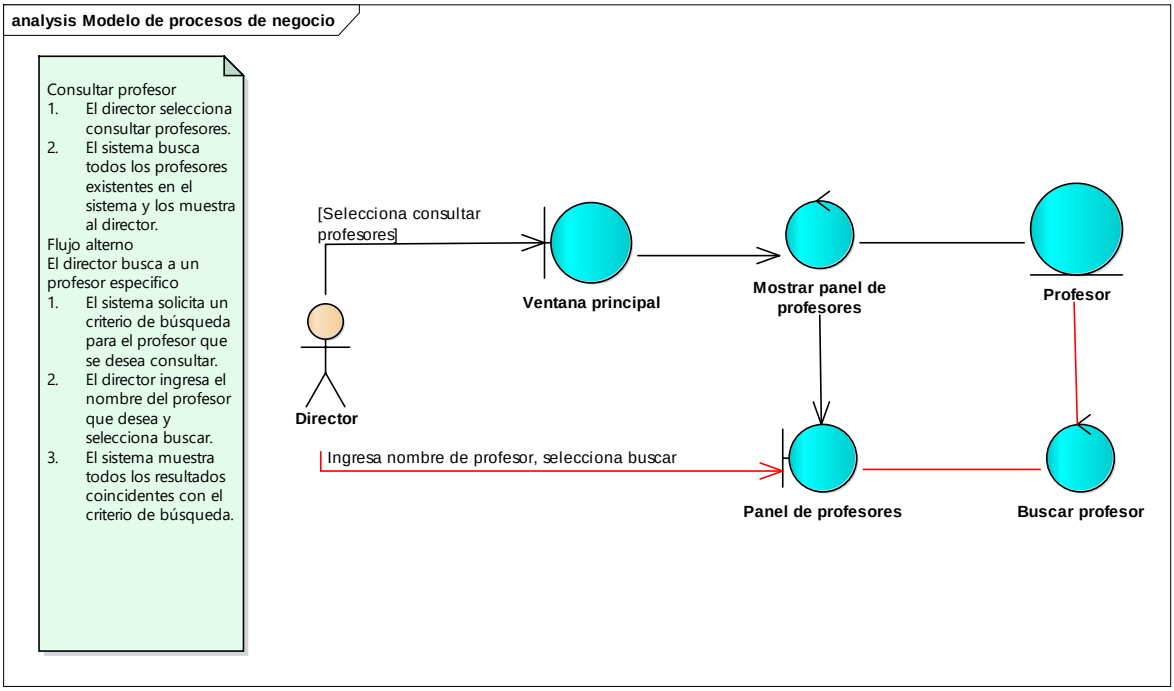
1.3.12. CU 21 – CRU cliente.



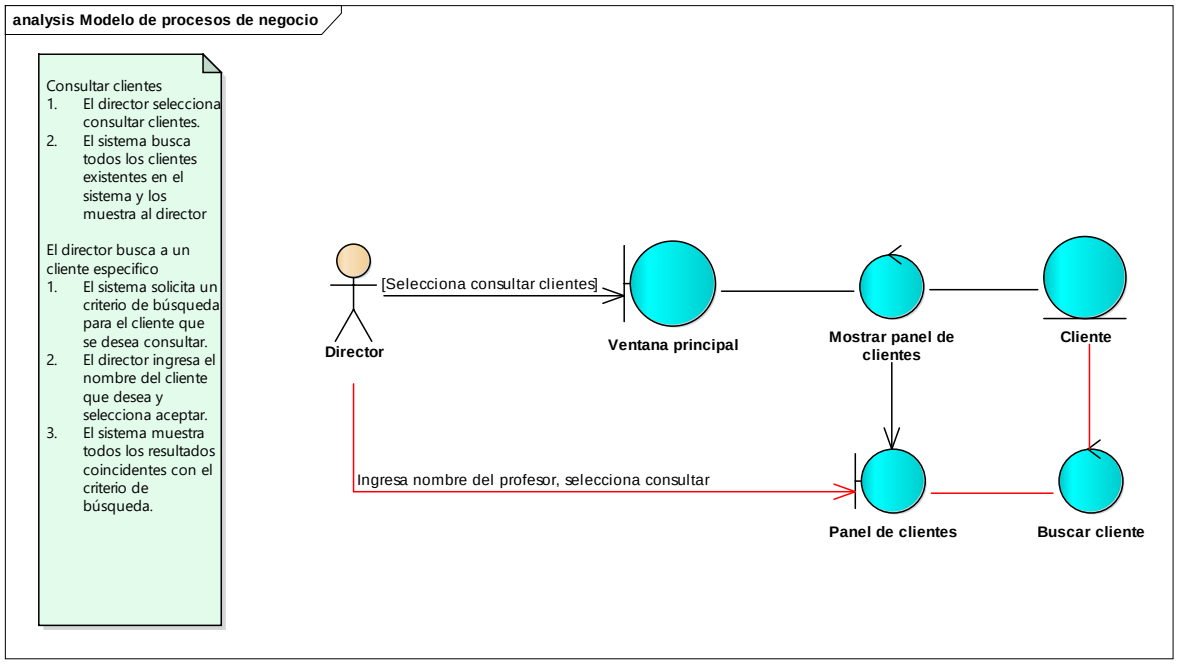
1.3.13. CU 22 - Iniciar sesión.



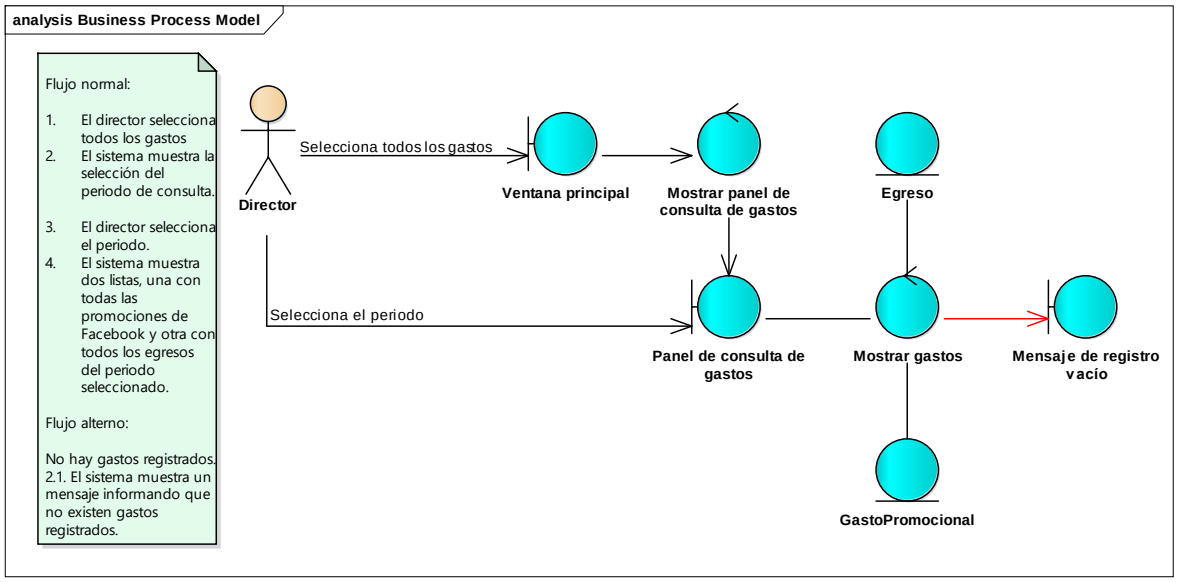
1.3.14. CU 23 – Consultar profesores.



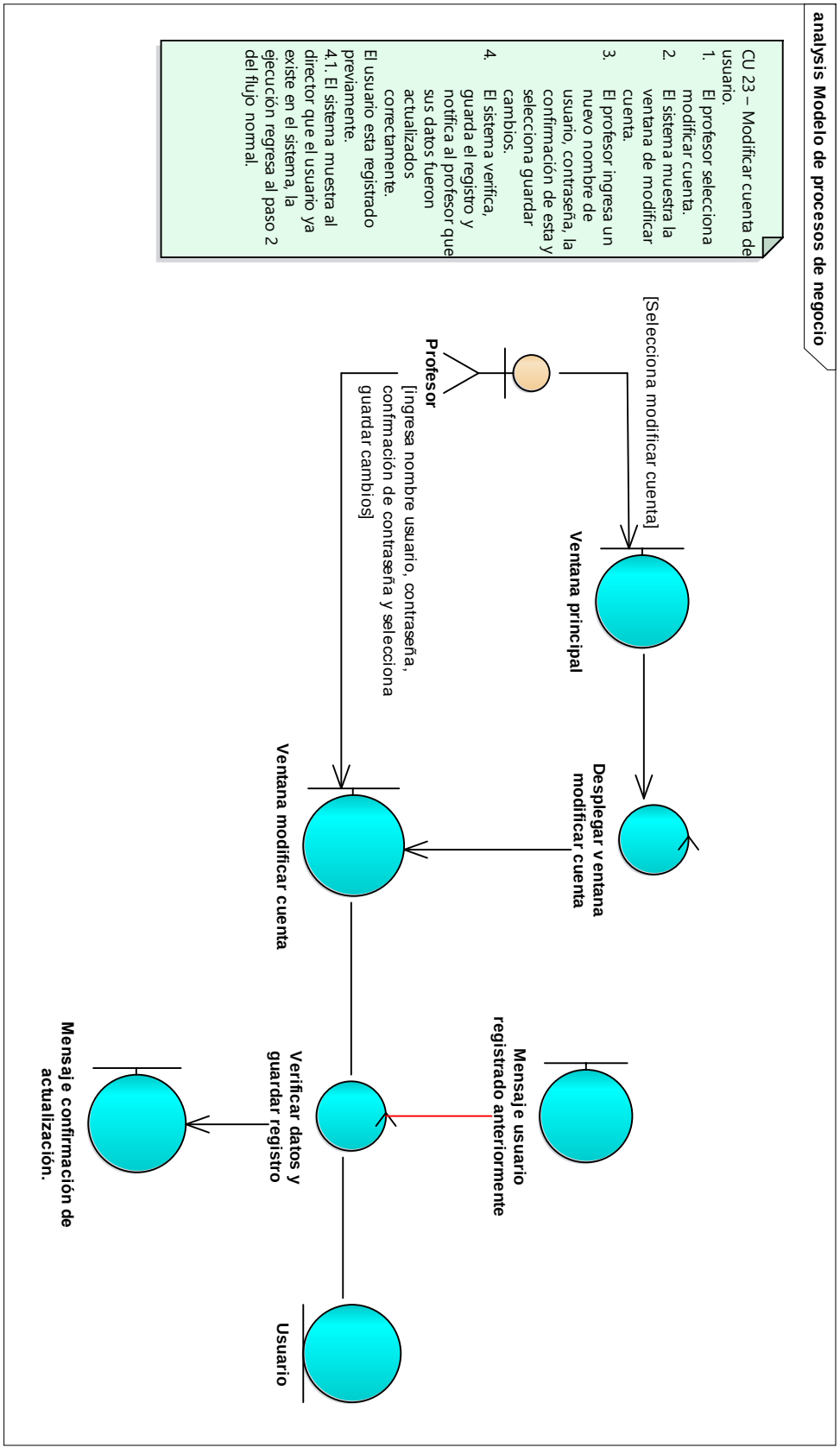
1.3.15. CU 24 – Consultar clientes.



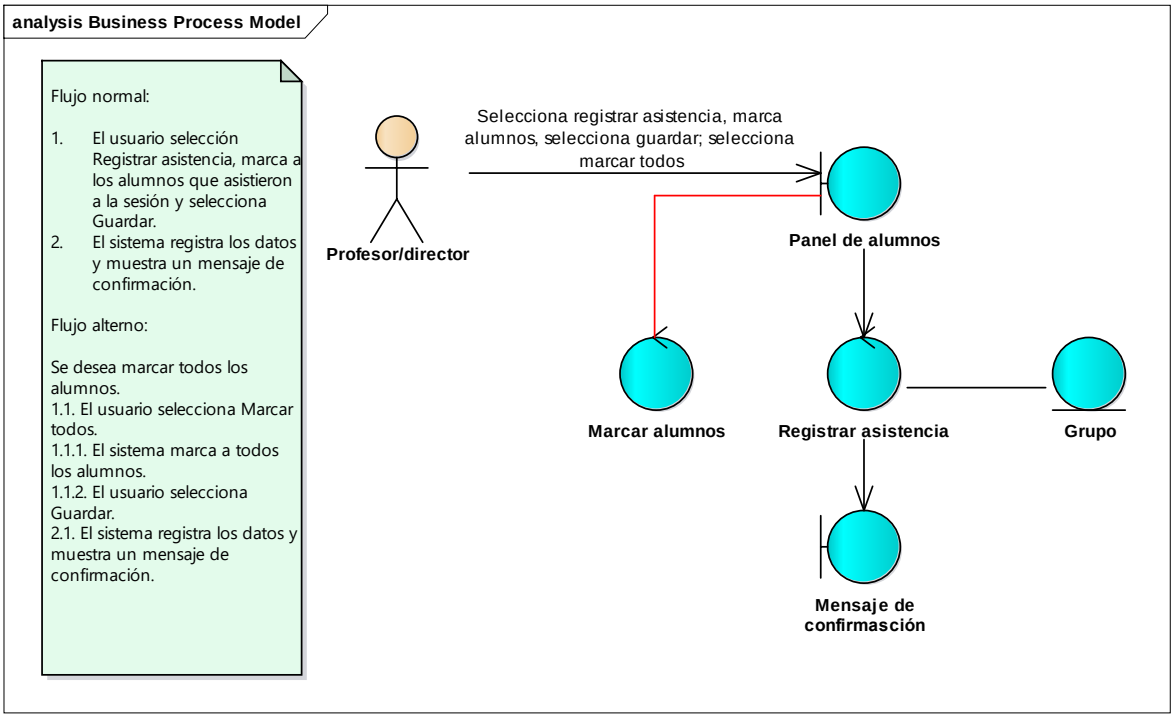
1.3.16. CU 25 – Consultar gastos.



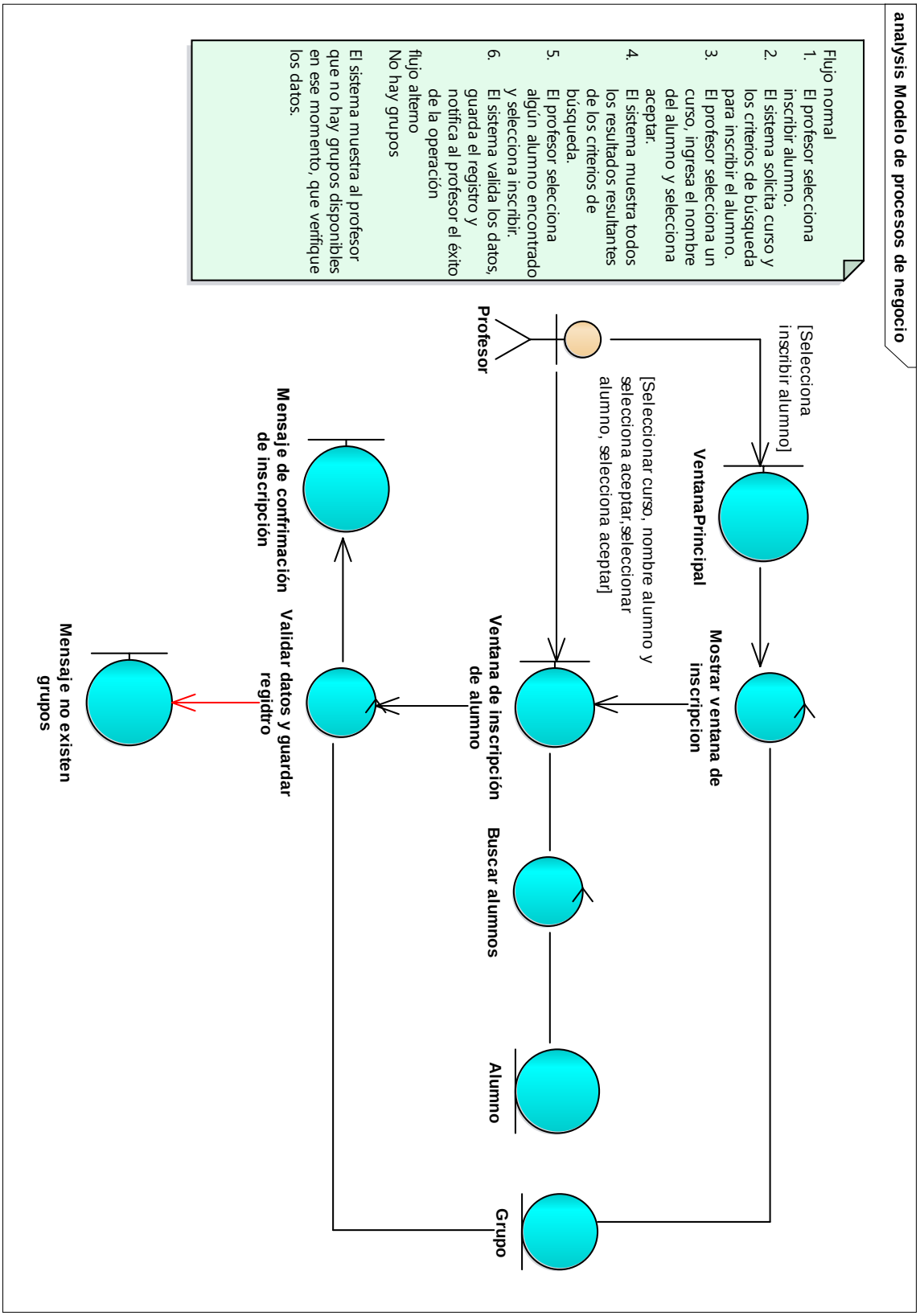
1.3.17. CU 26 – Modificar cuenta de usuario.



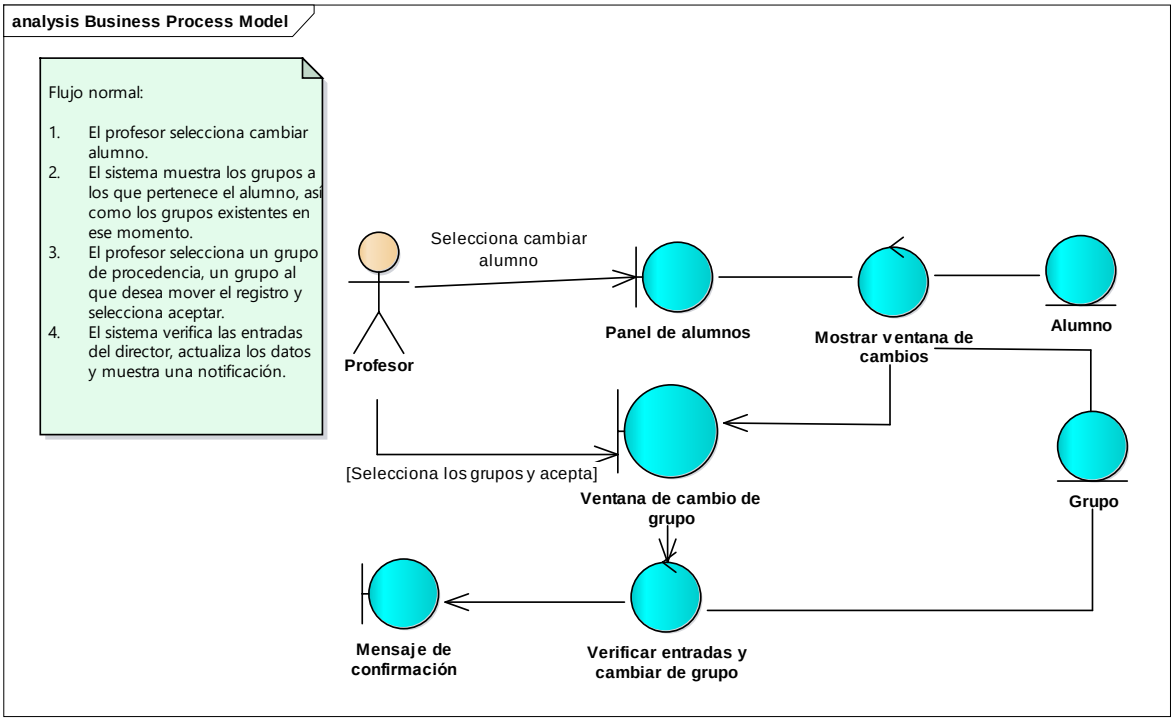
1.3.18. CU 7 – Registrar asistencia.



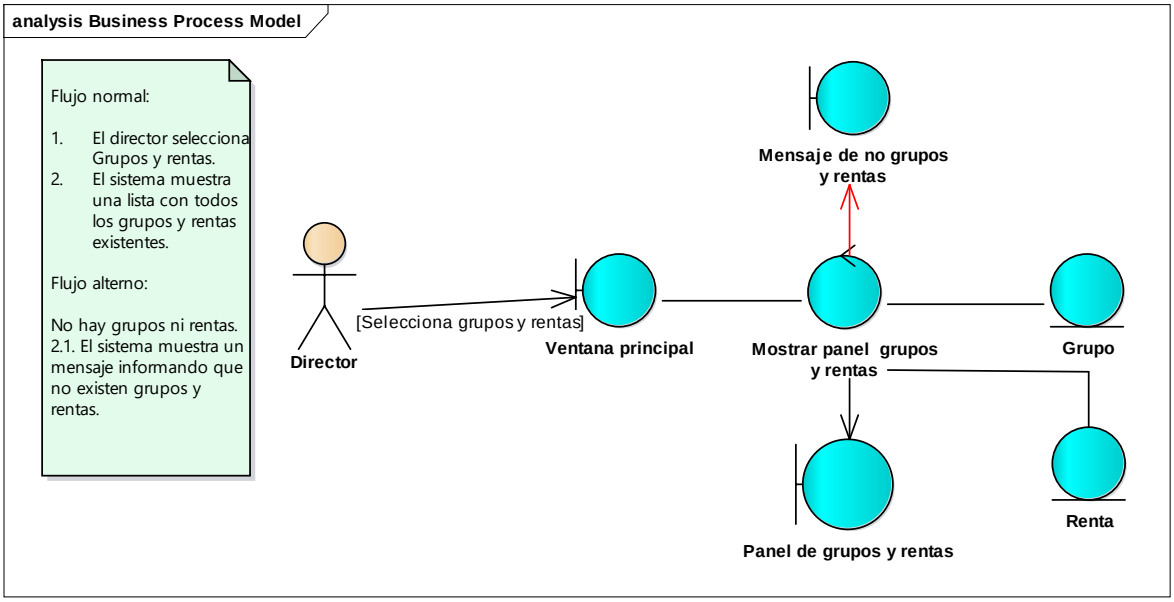
1.3.19. CU 8 – Inscribir alumno.



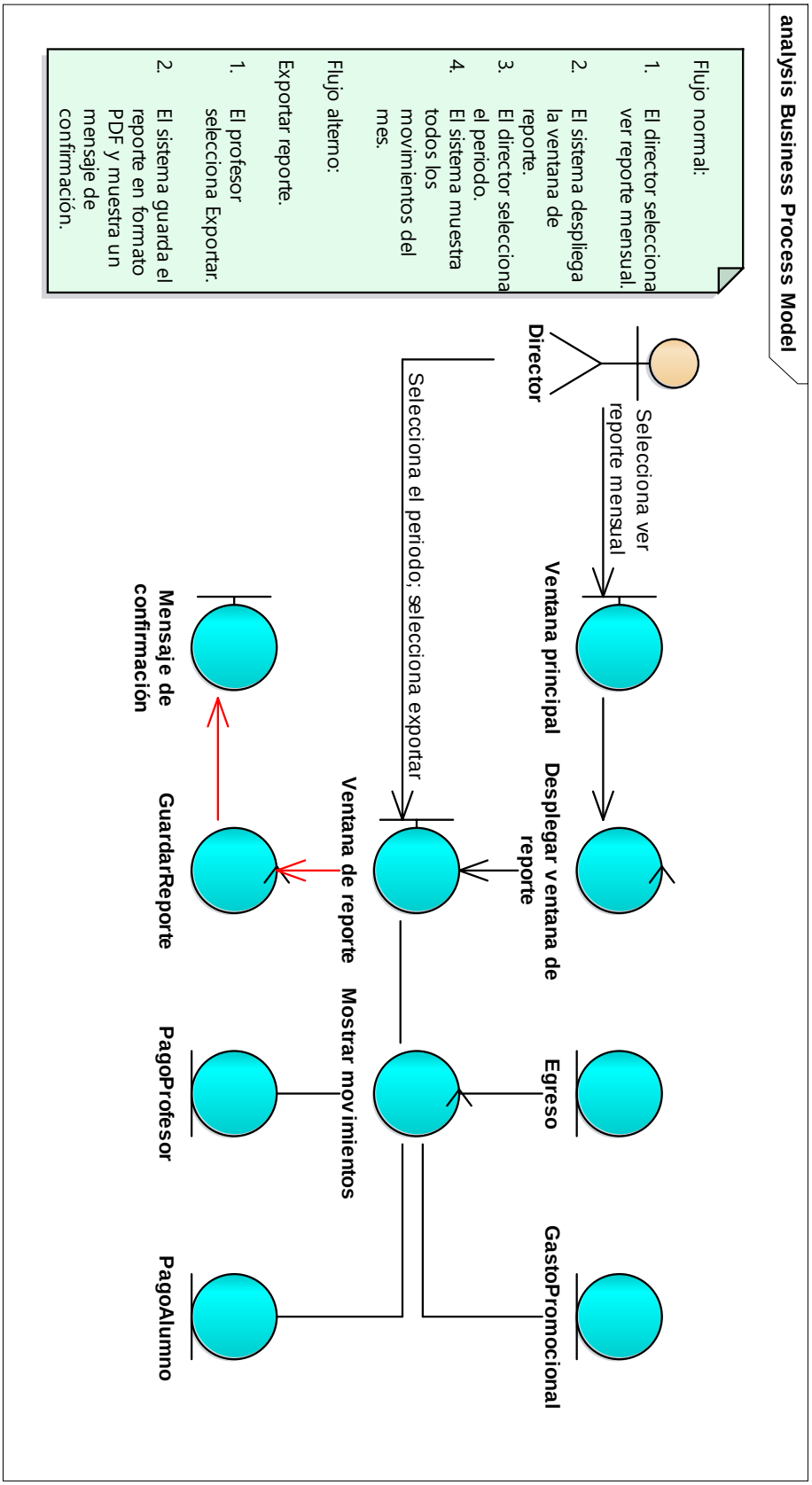
1.3.20. CU 11 – Cambiar alumno de grupo.



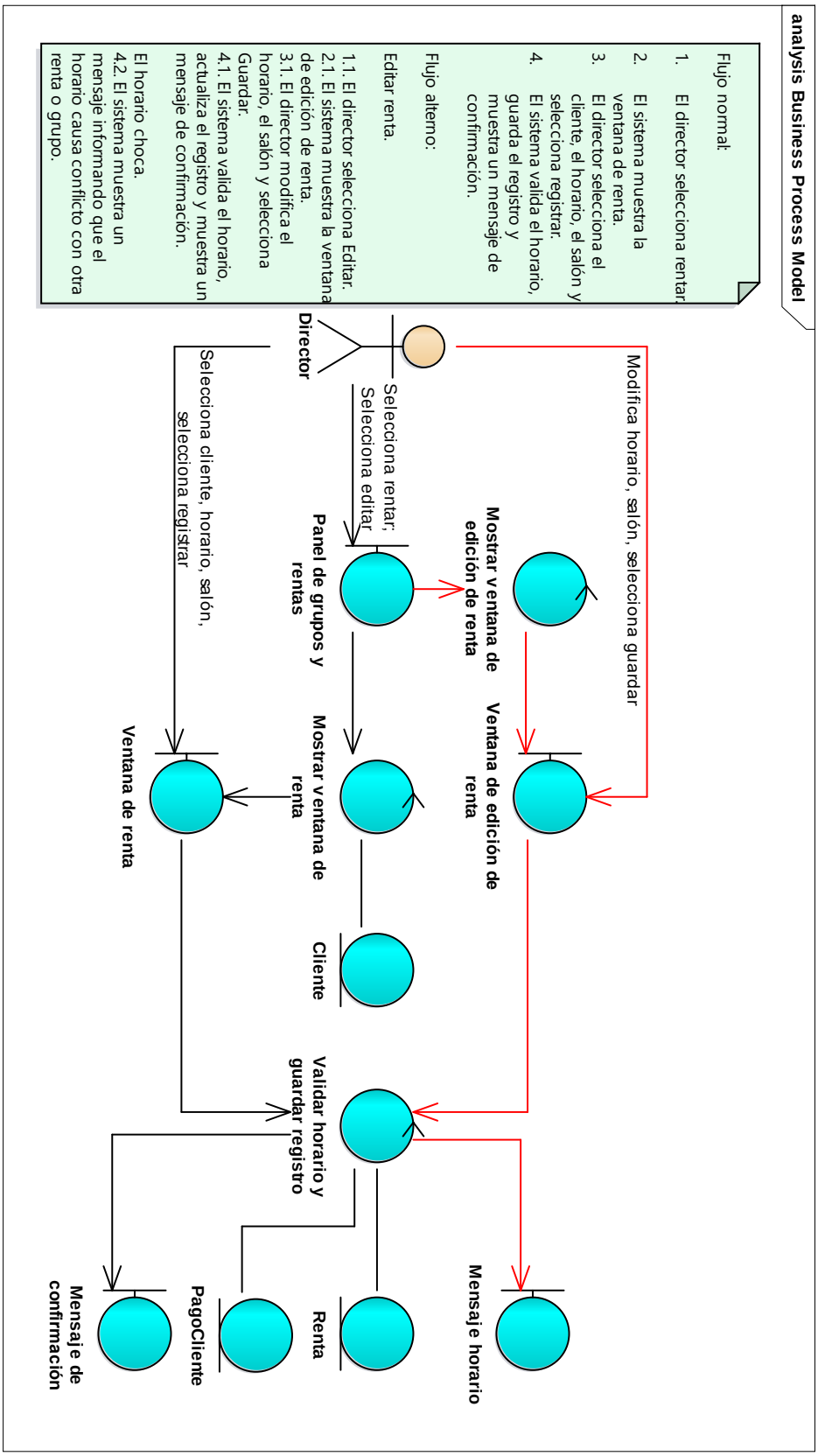
1.3.21. CU 15 – Consultar grupos y rentas.



1.3.22. CU 18 – Generar reporte de ingresos y egresos.

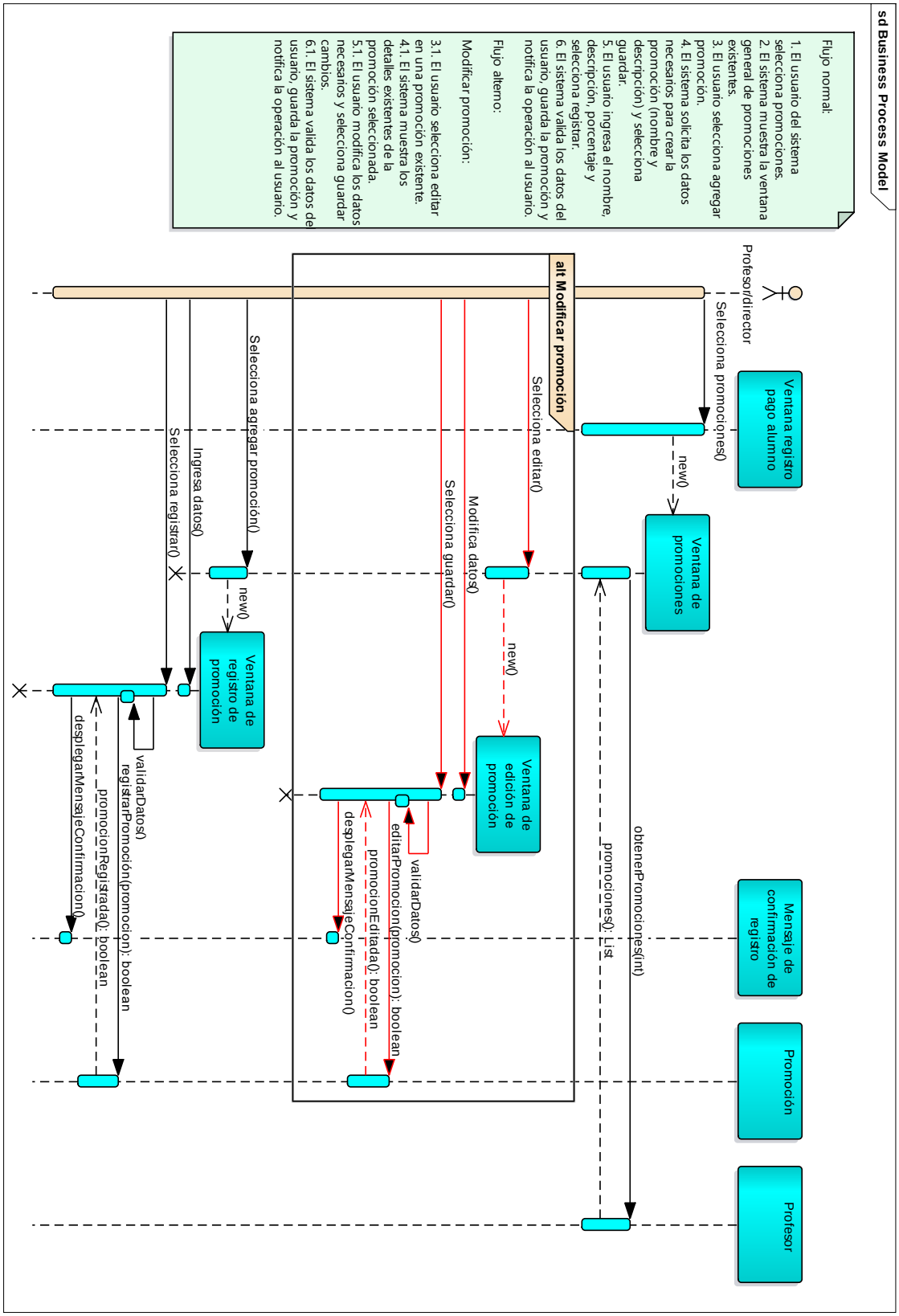


1.3.23. CU 19 – CRU Renta de espacio.

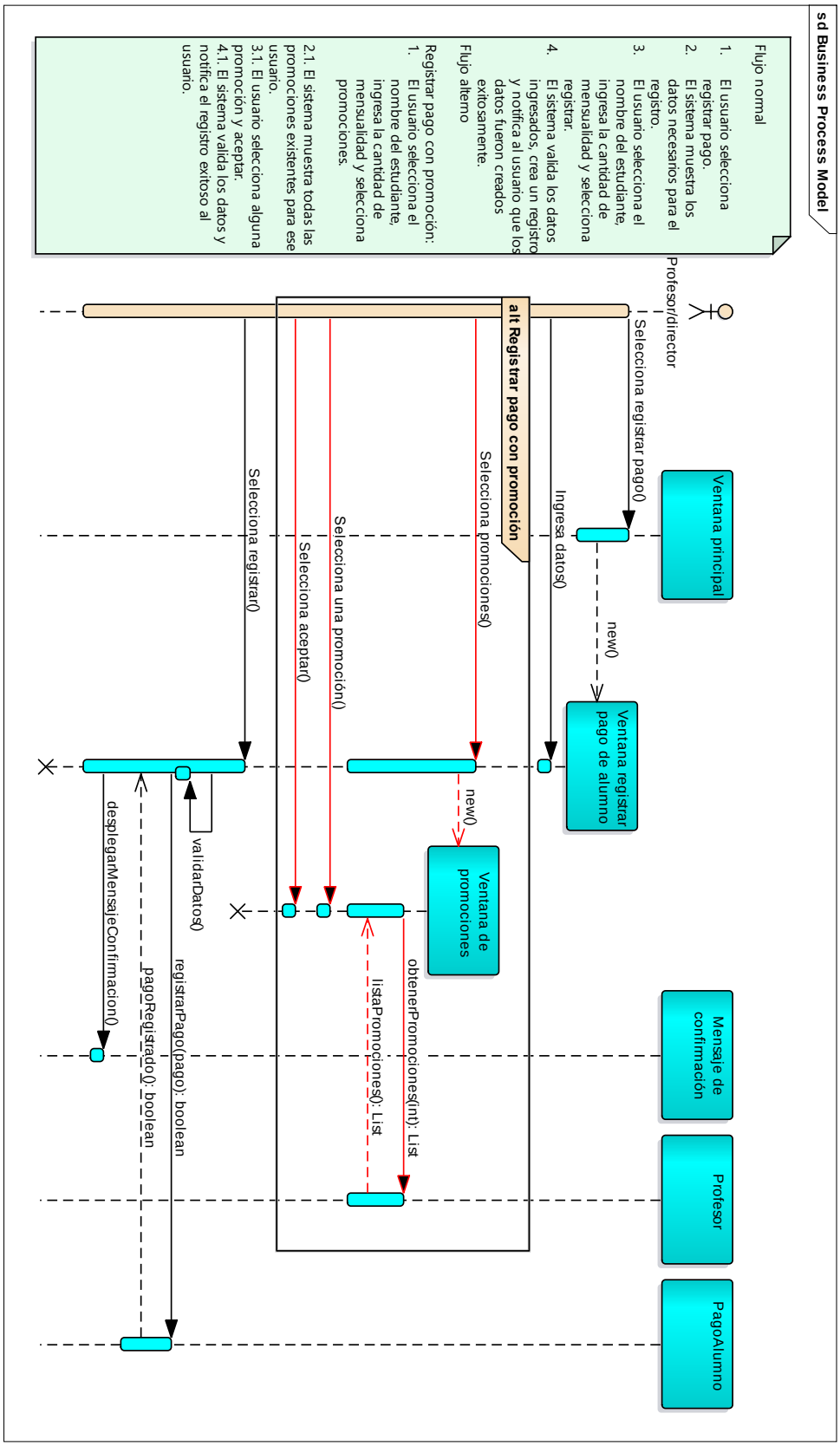


1.4. Diagramas de secuencia.

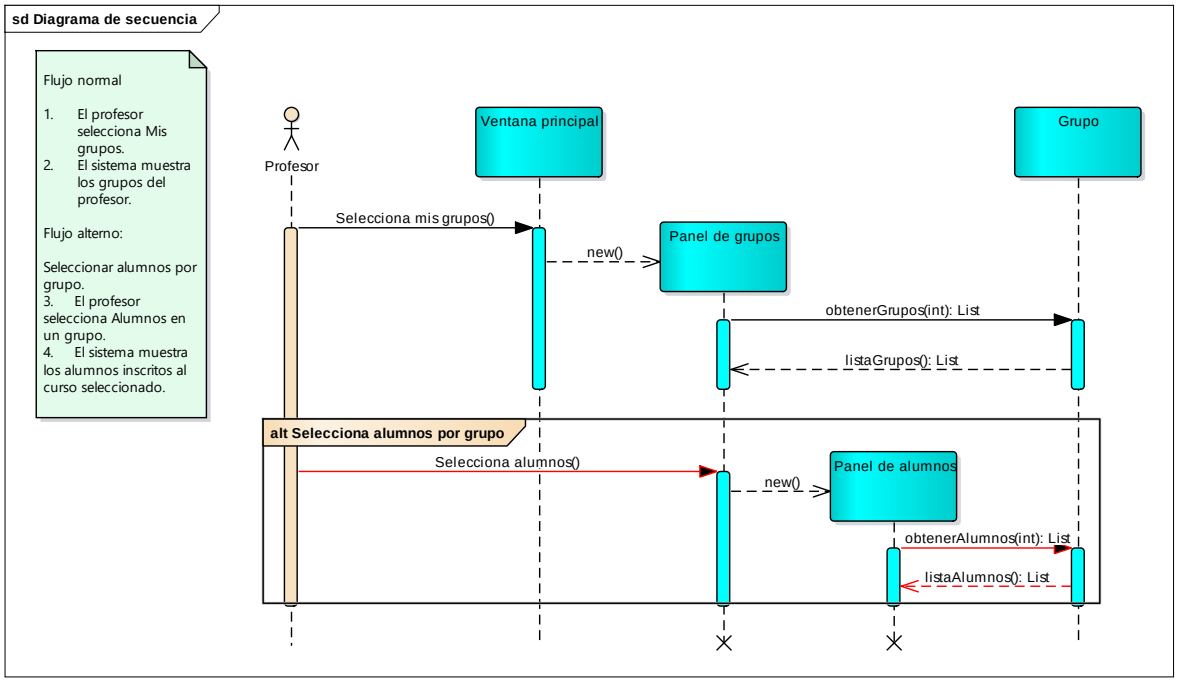
1.4.1. CU 01 – CRU Promoción.



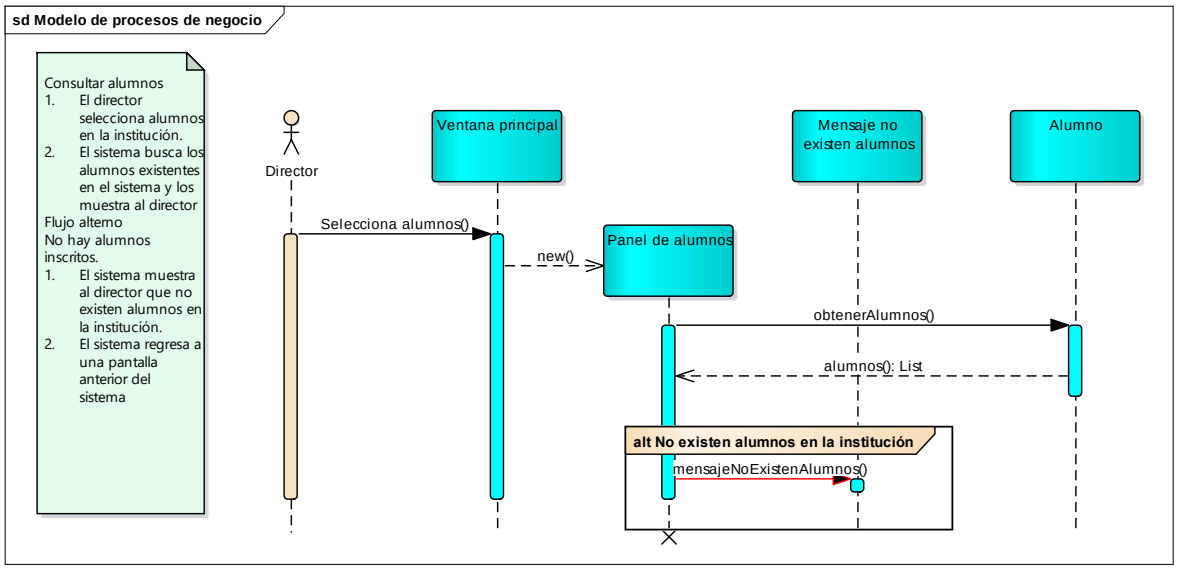
1.4.2. CU 02 – Registrar pago de alumno.



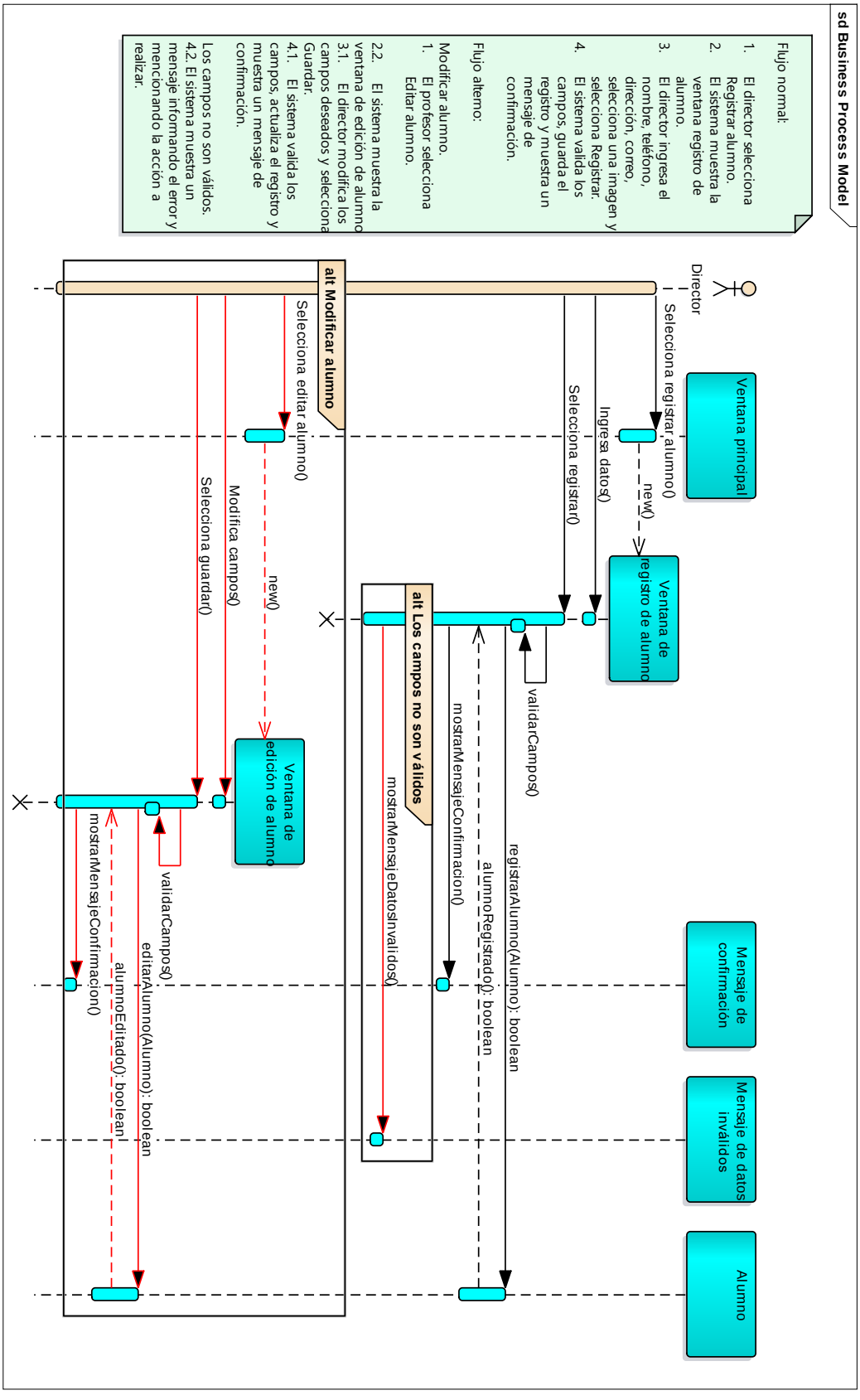
1.4.3. CU 05 – Consultar grupos.



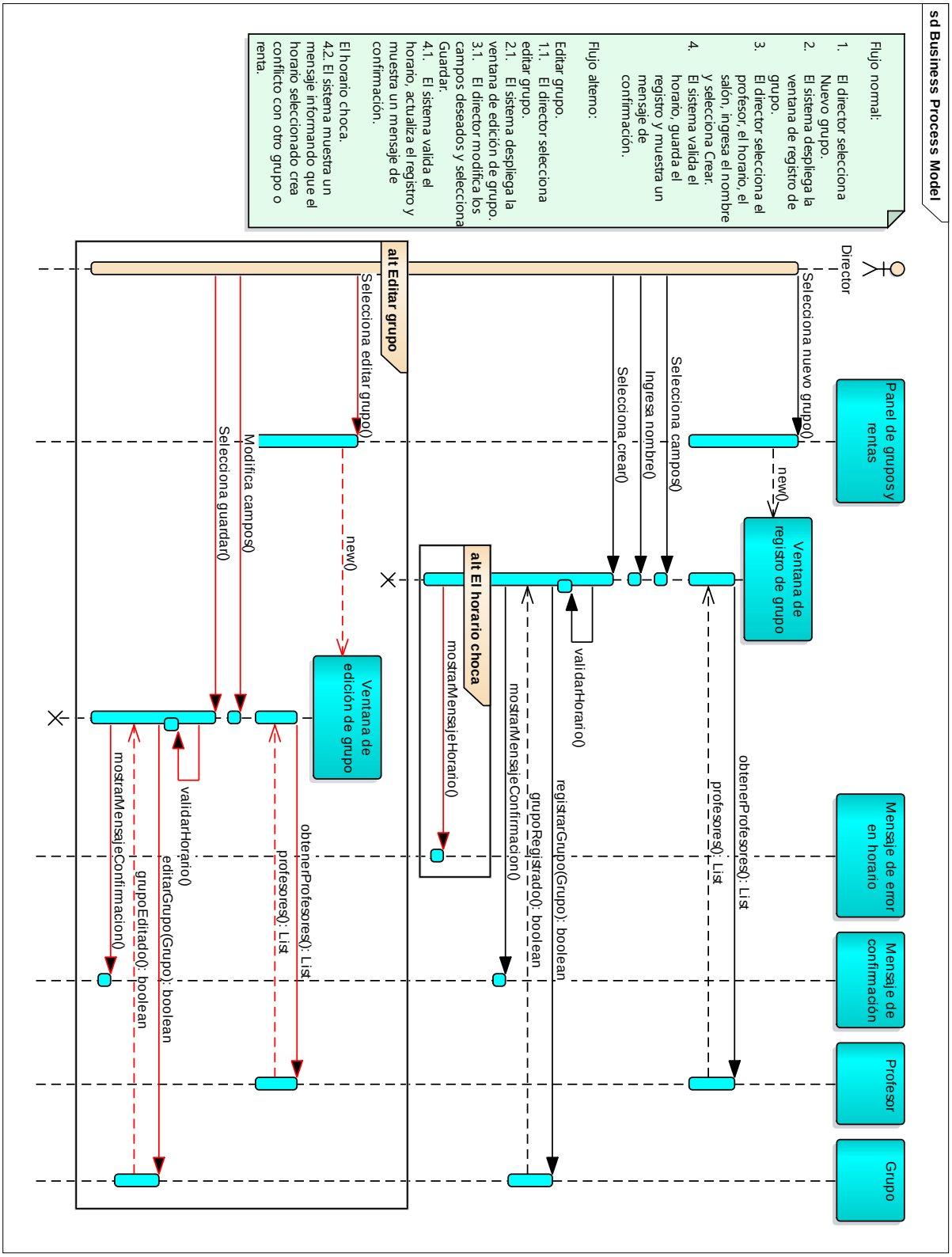
1.4.4. CU 09 – Consultar alumnos.



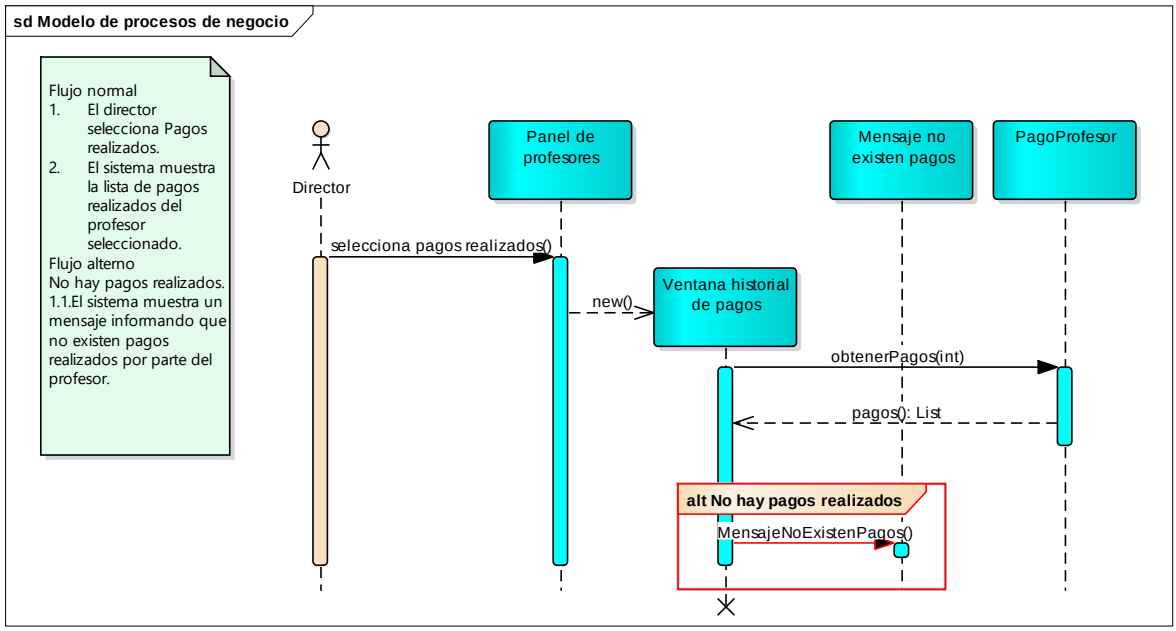
1.4.5. CU 10 – CRU Alumno.



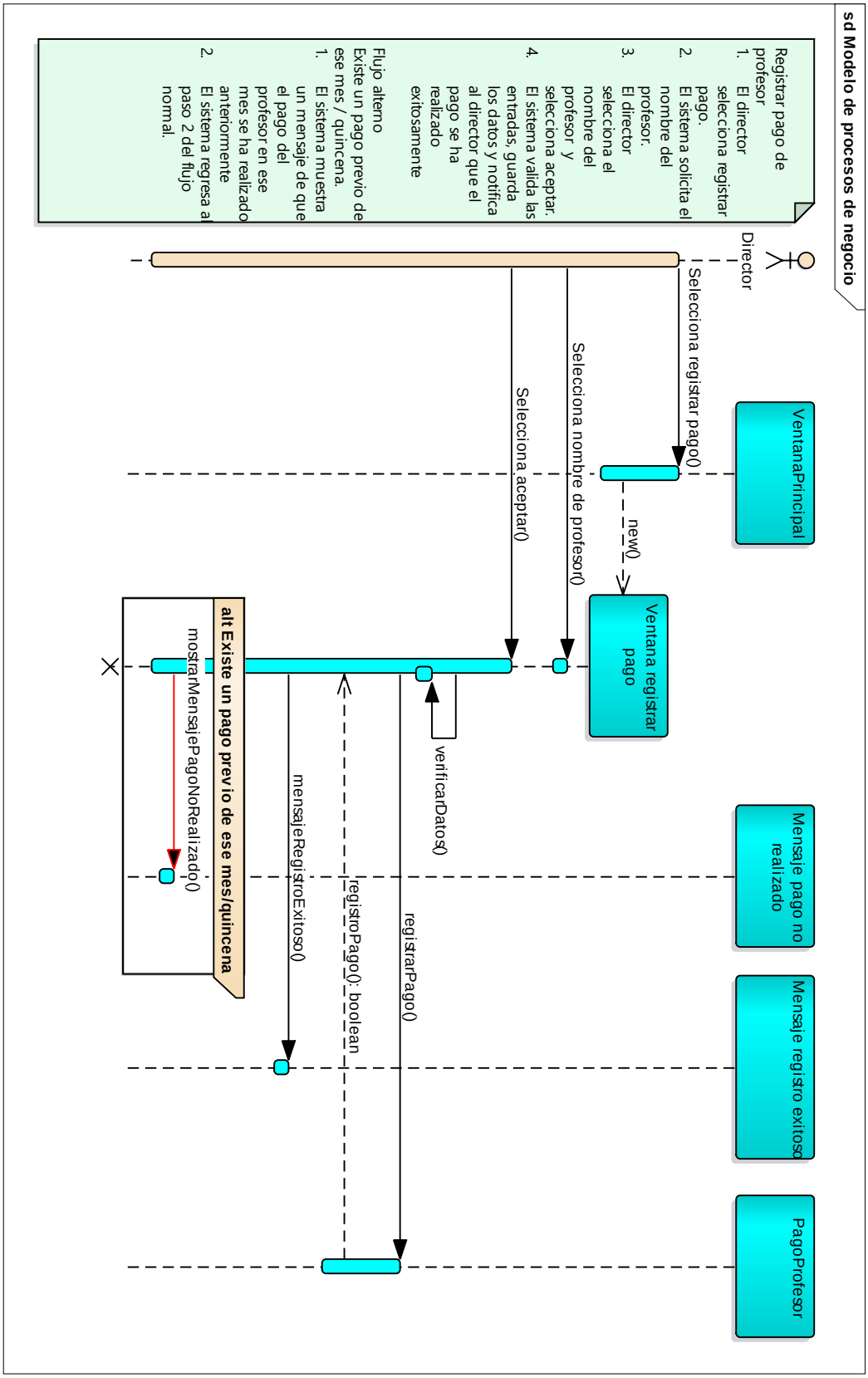
1.4.6. CU 12 – CRU Grupo.



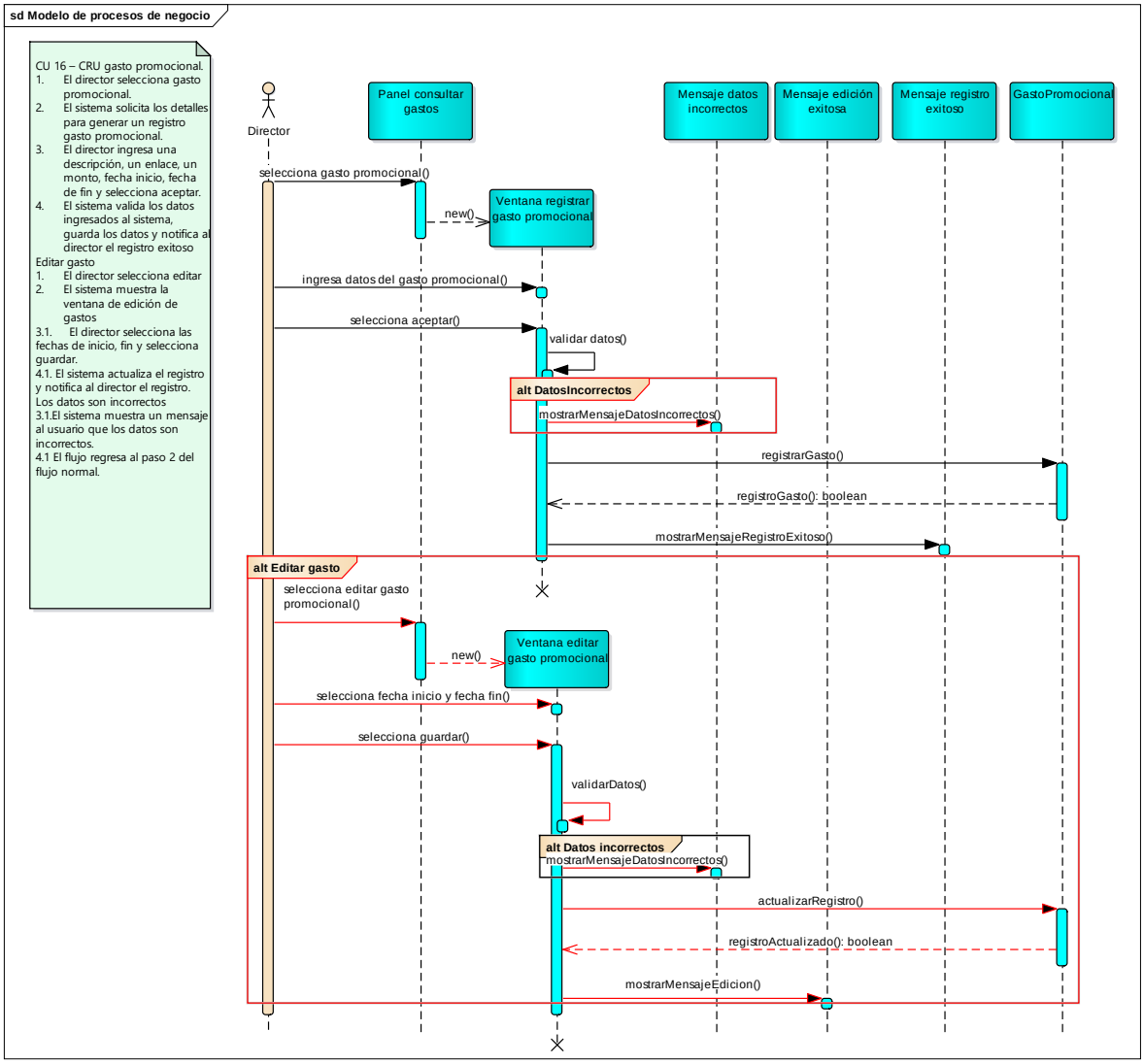
1.4.7. CU 13 – Consultar historial de pago de profesores.



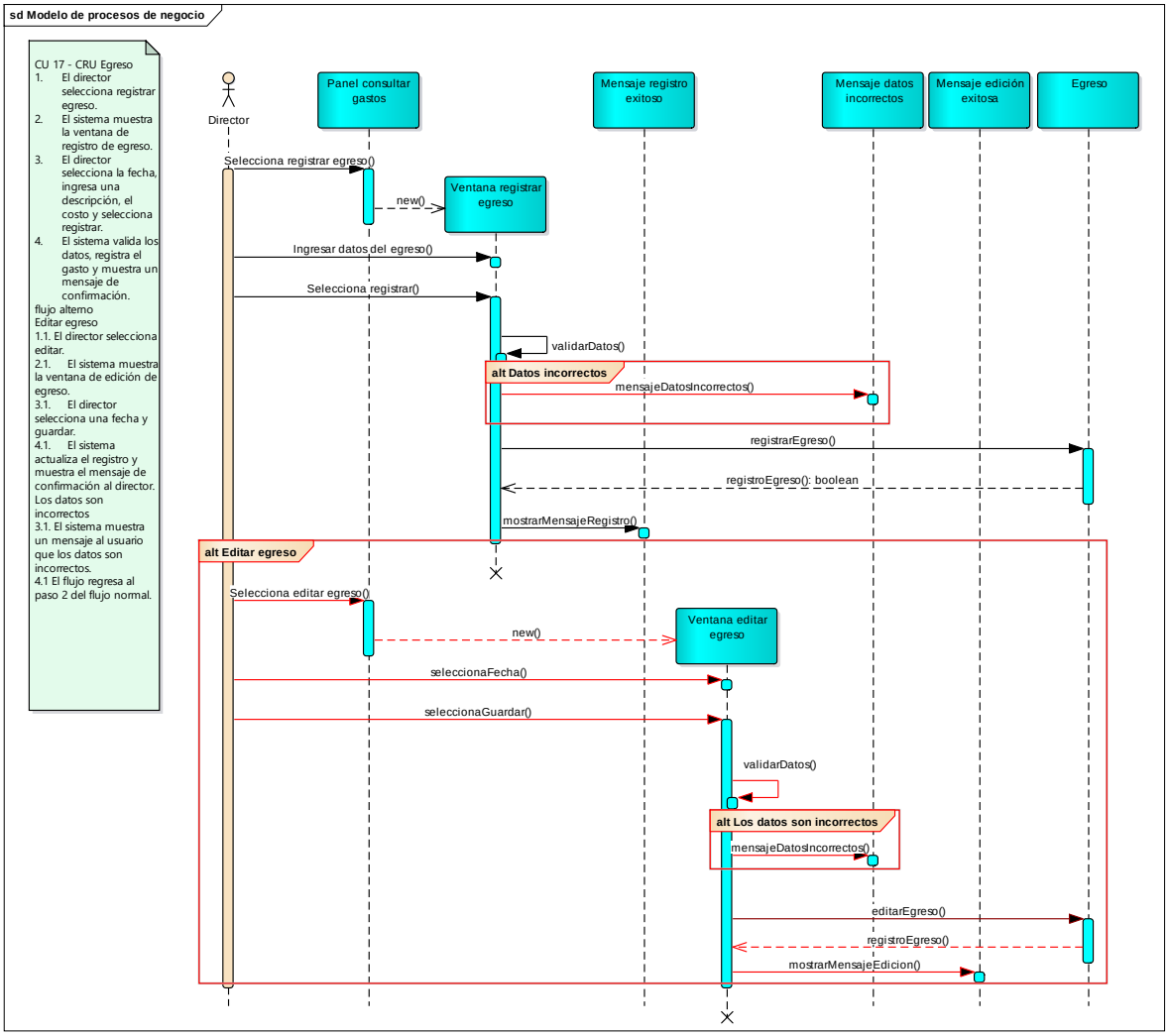
1.4.8. CU 14 – Registrar pago de profesor.



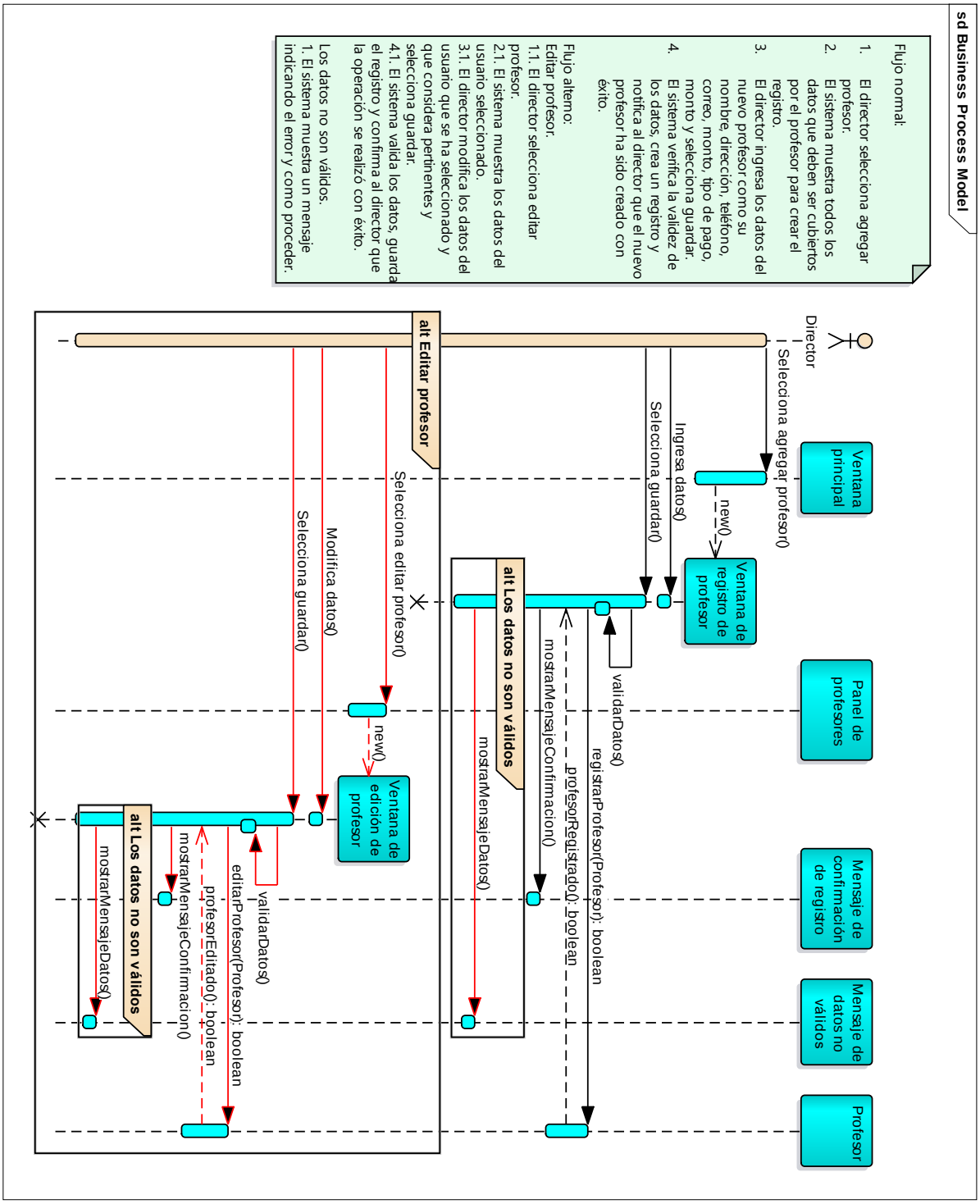
1.4.9. CU 16 – CRU Gasto promocional.



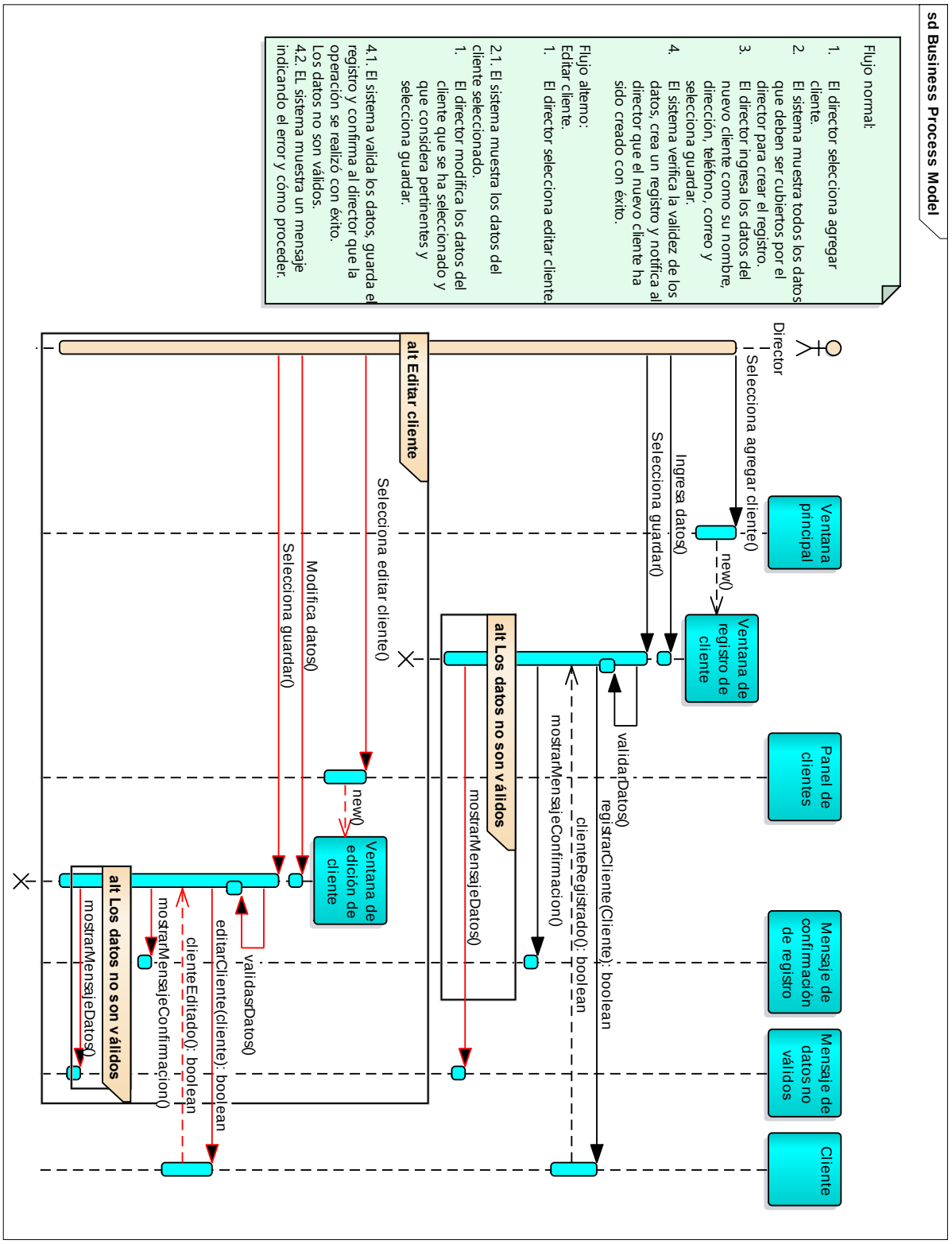
1.4.10. CU 17 – CRU Egreso.



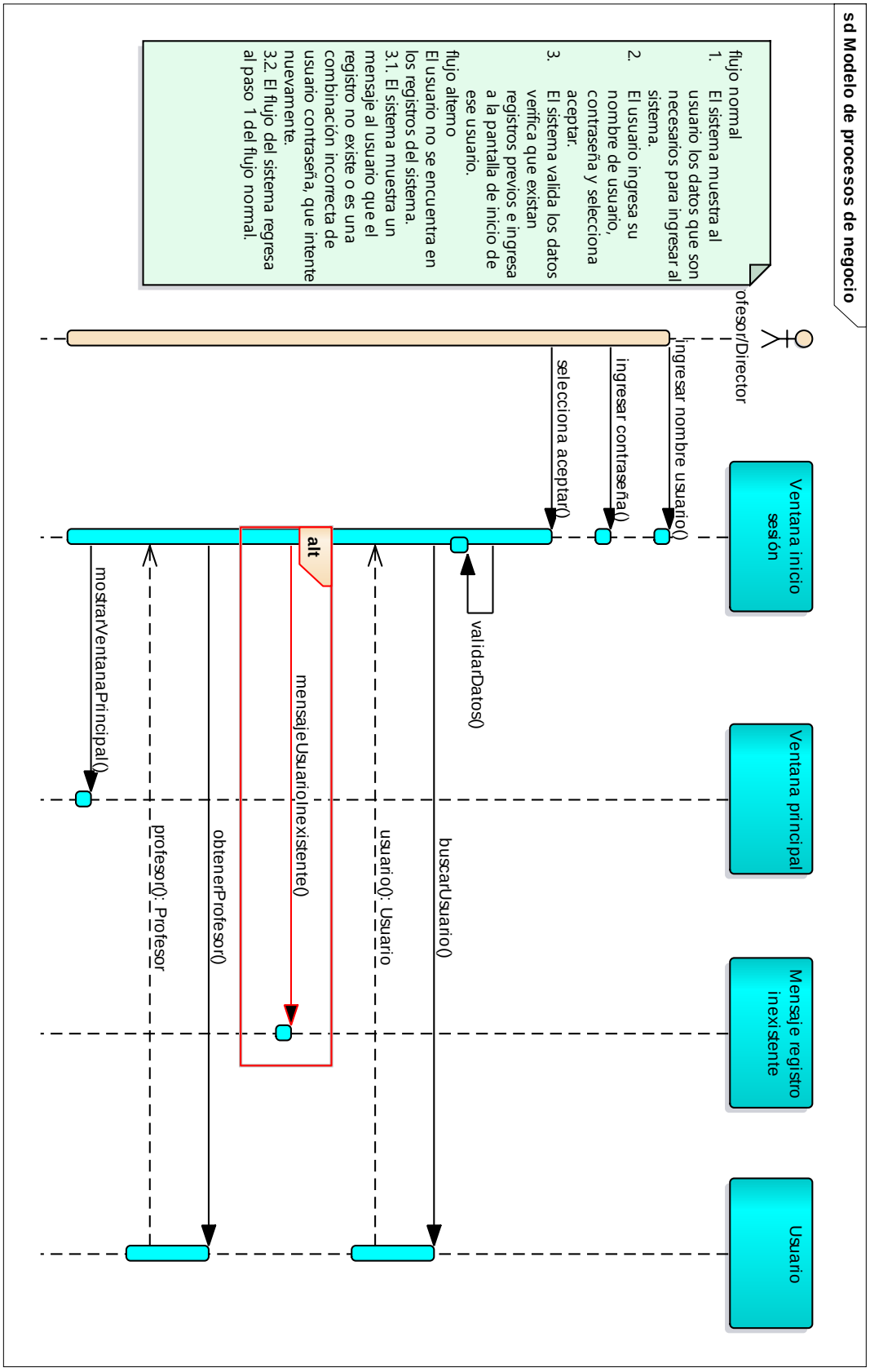
1.4.11. CU 20 – CRU Profesor.



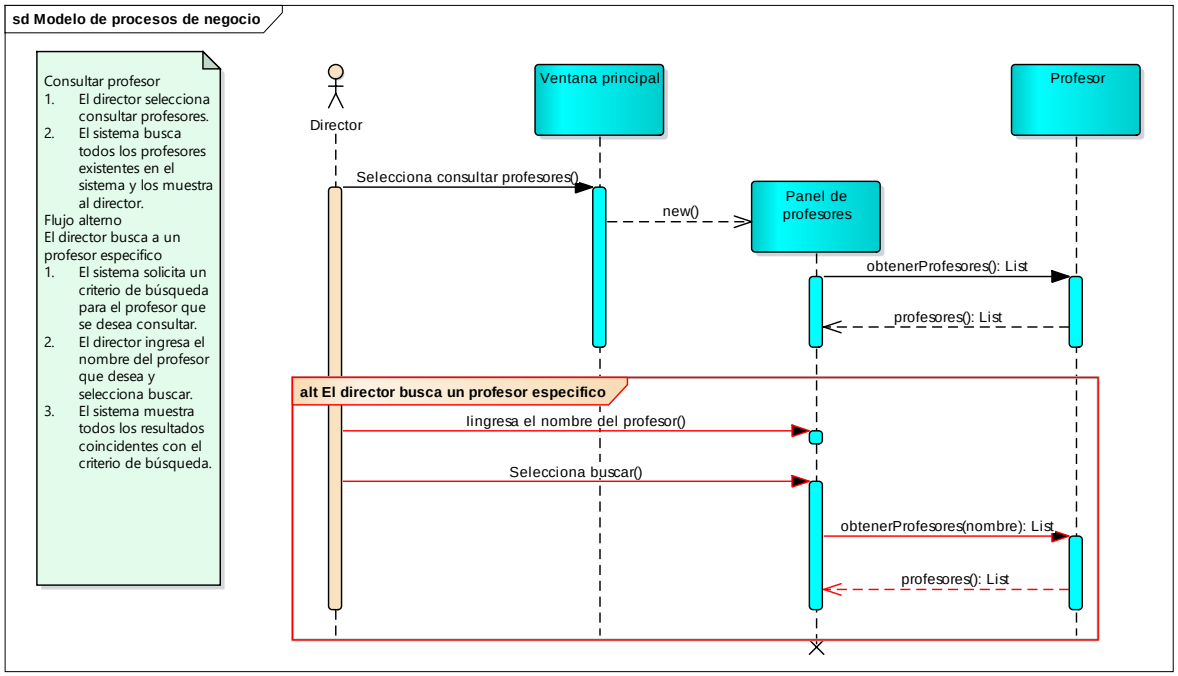
1.4.12. CU 21 – CRU Cliente.



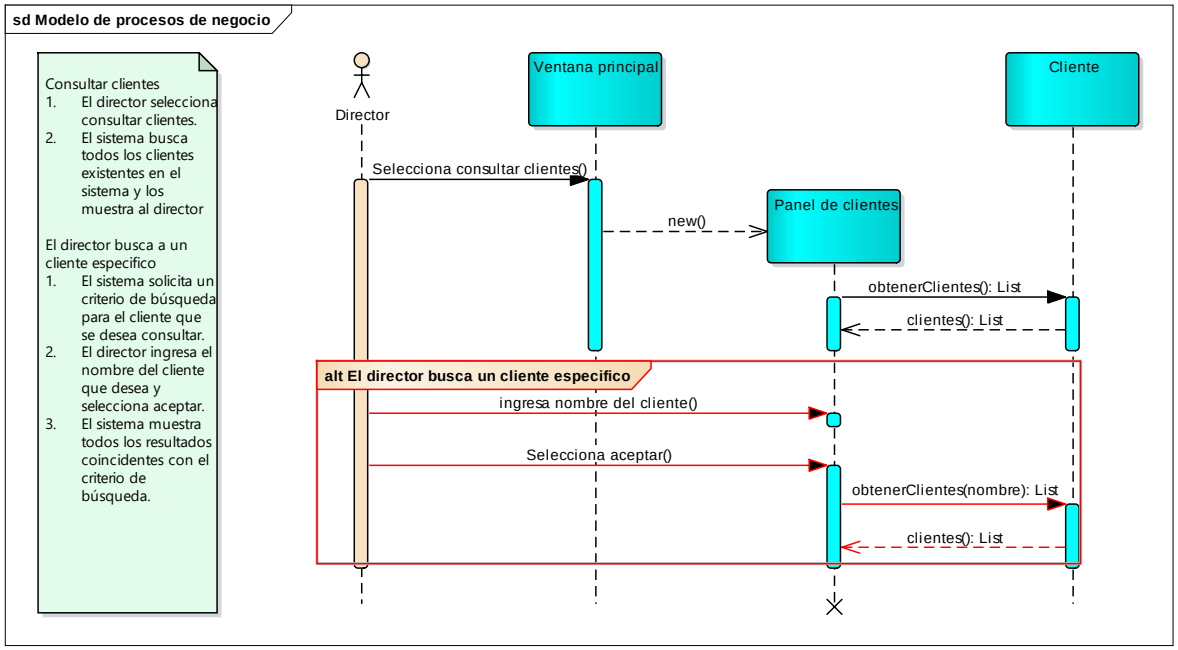
1.4.13. CU 22 – Iniciar sesión.



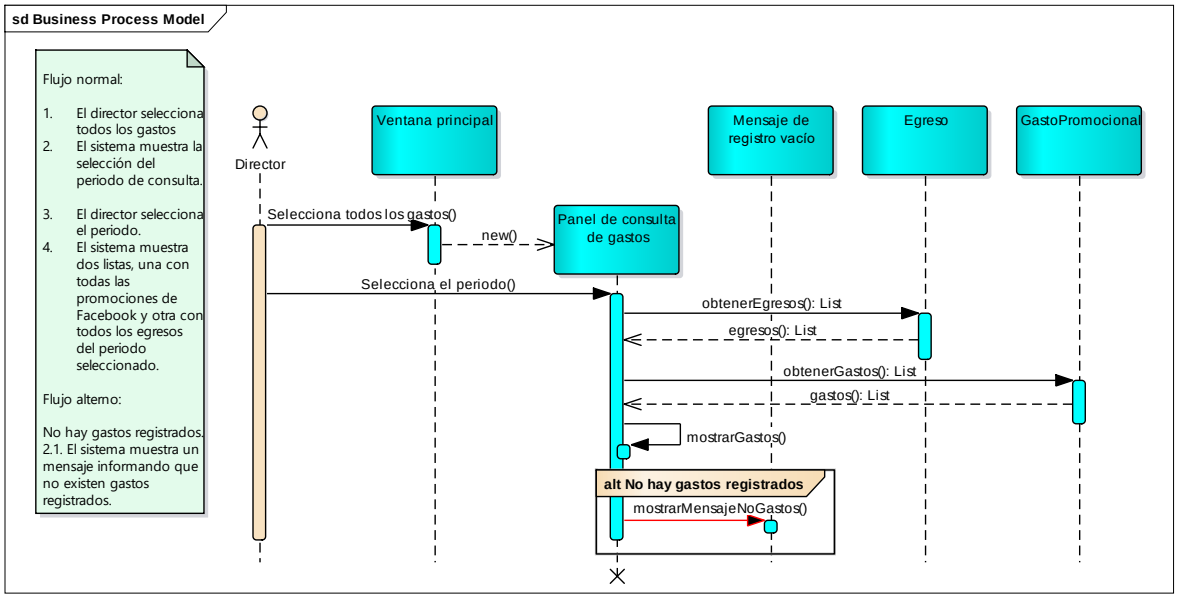
1.4.14. CU 23 – Consultar profesores.



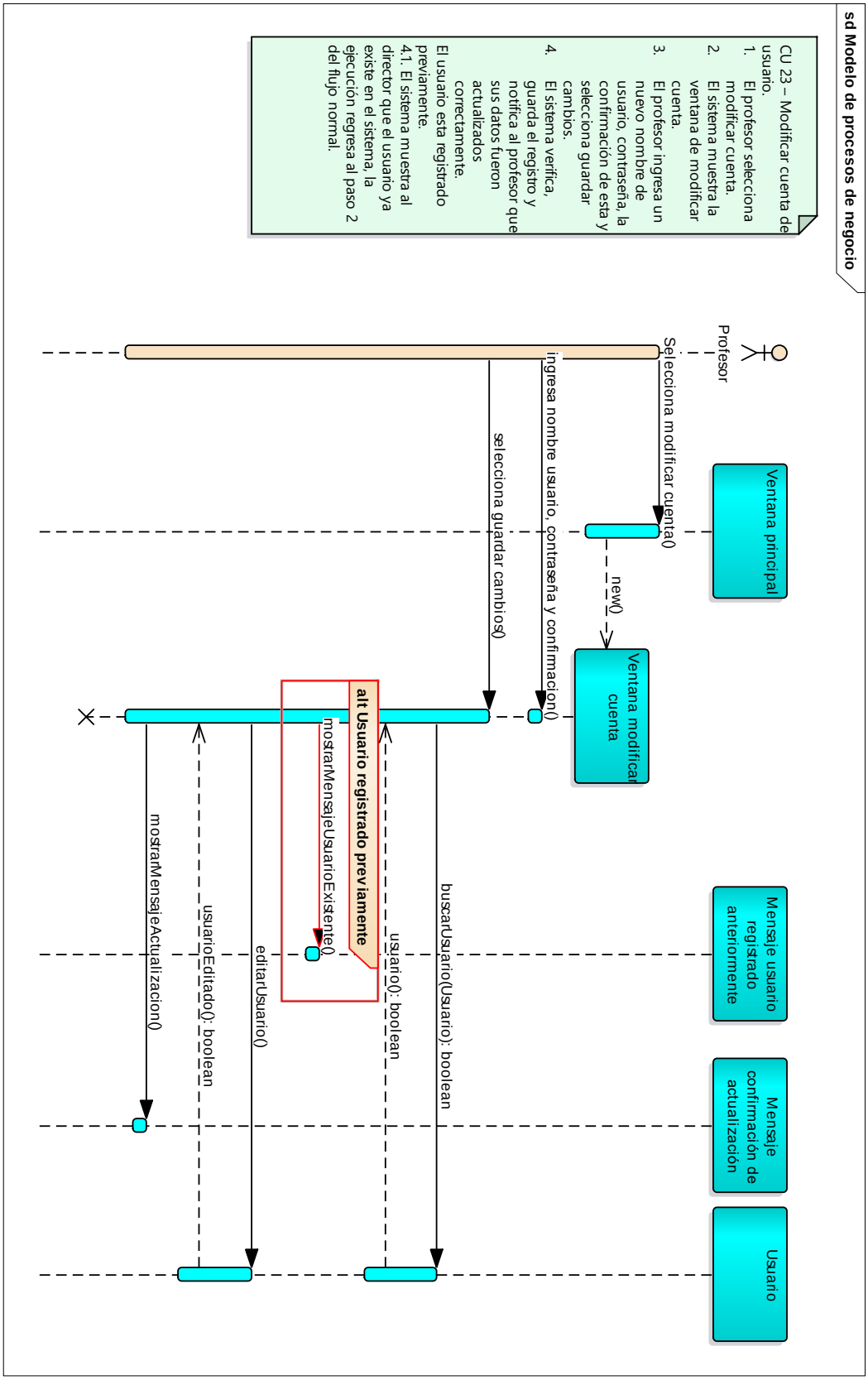
1.4.15. CU 24 – Consultar clientes.



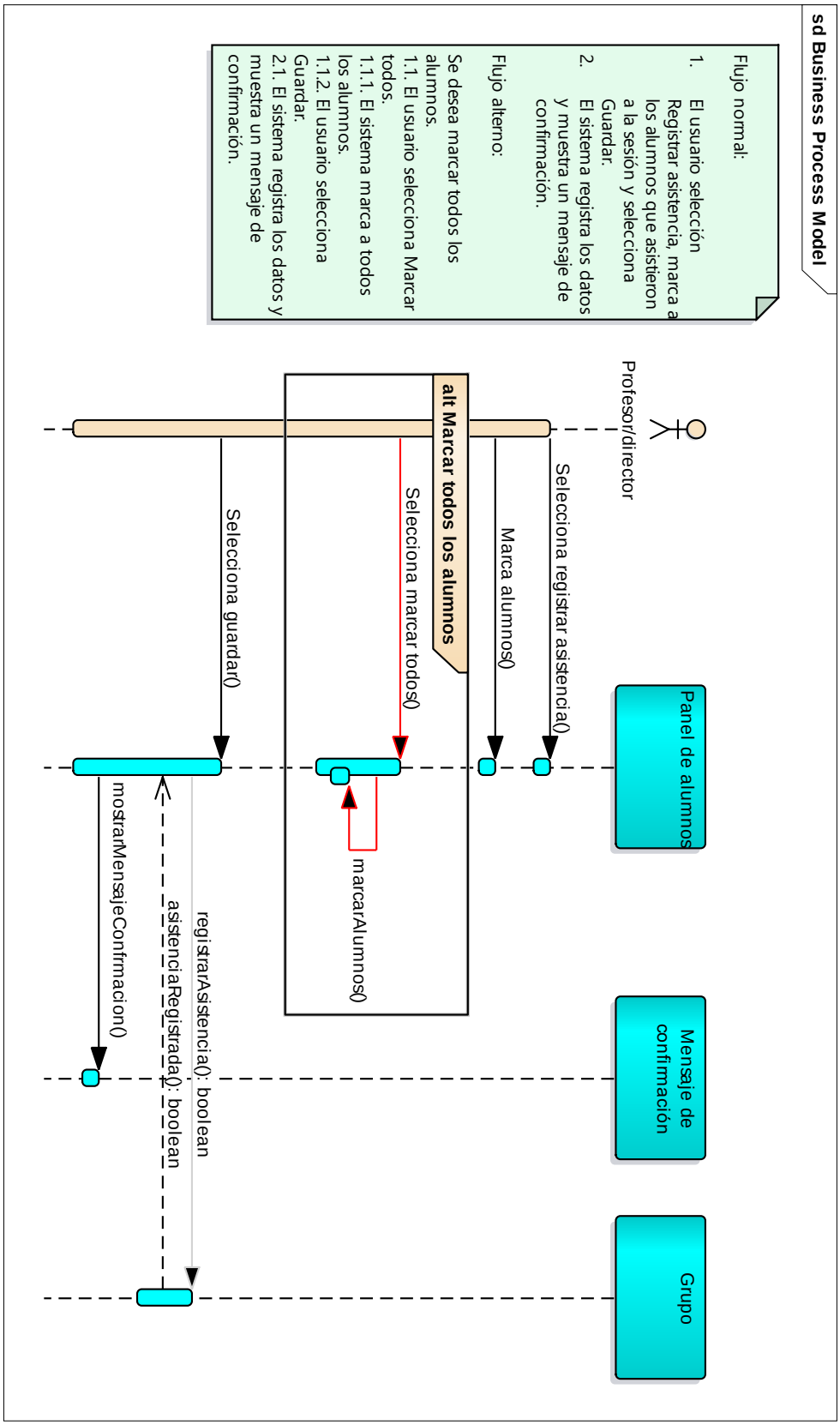
1.4.16. CU 25 – Consultar gastos.



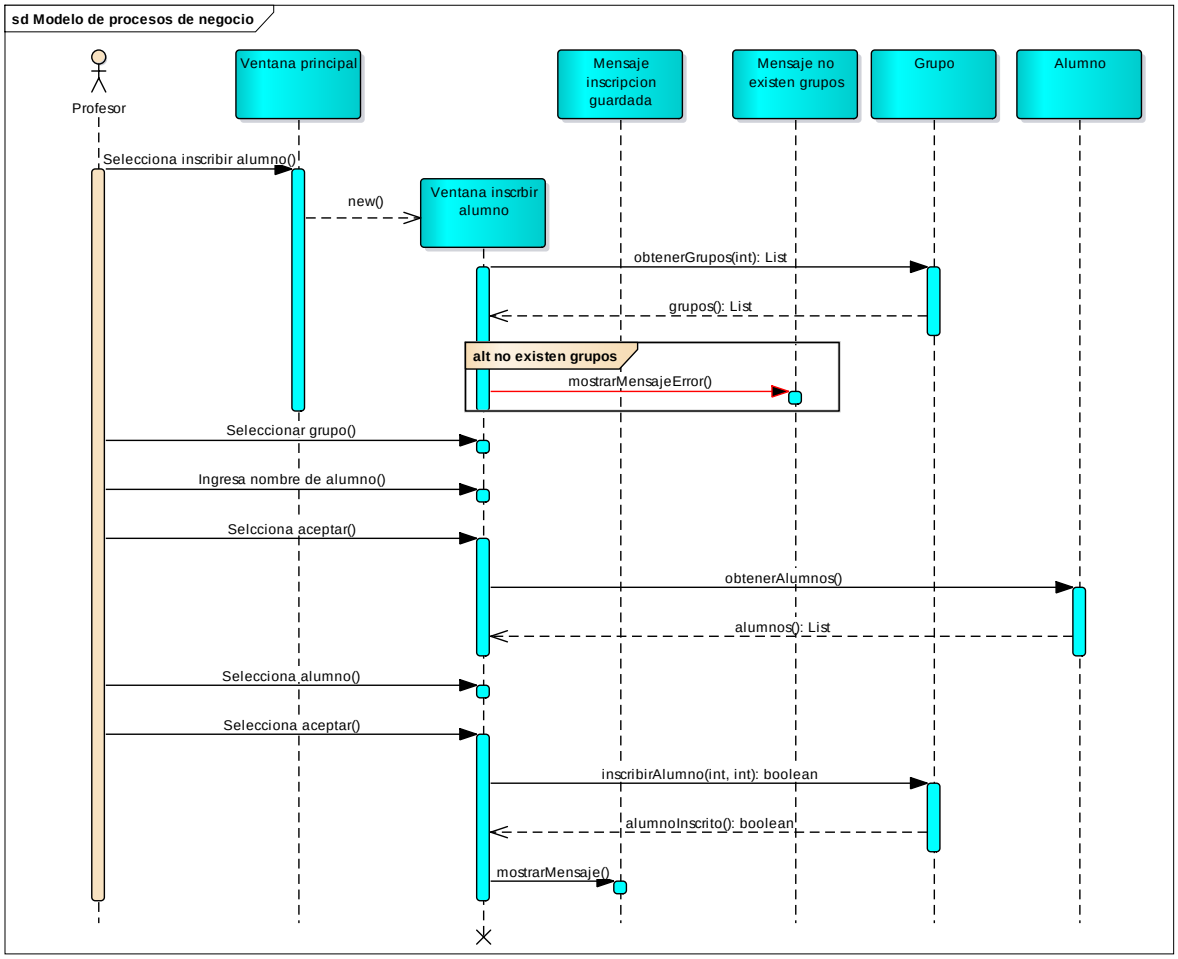
1.4.17. CU 26 – Modificar cuenta de usuario.



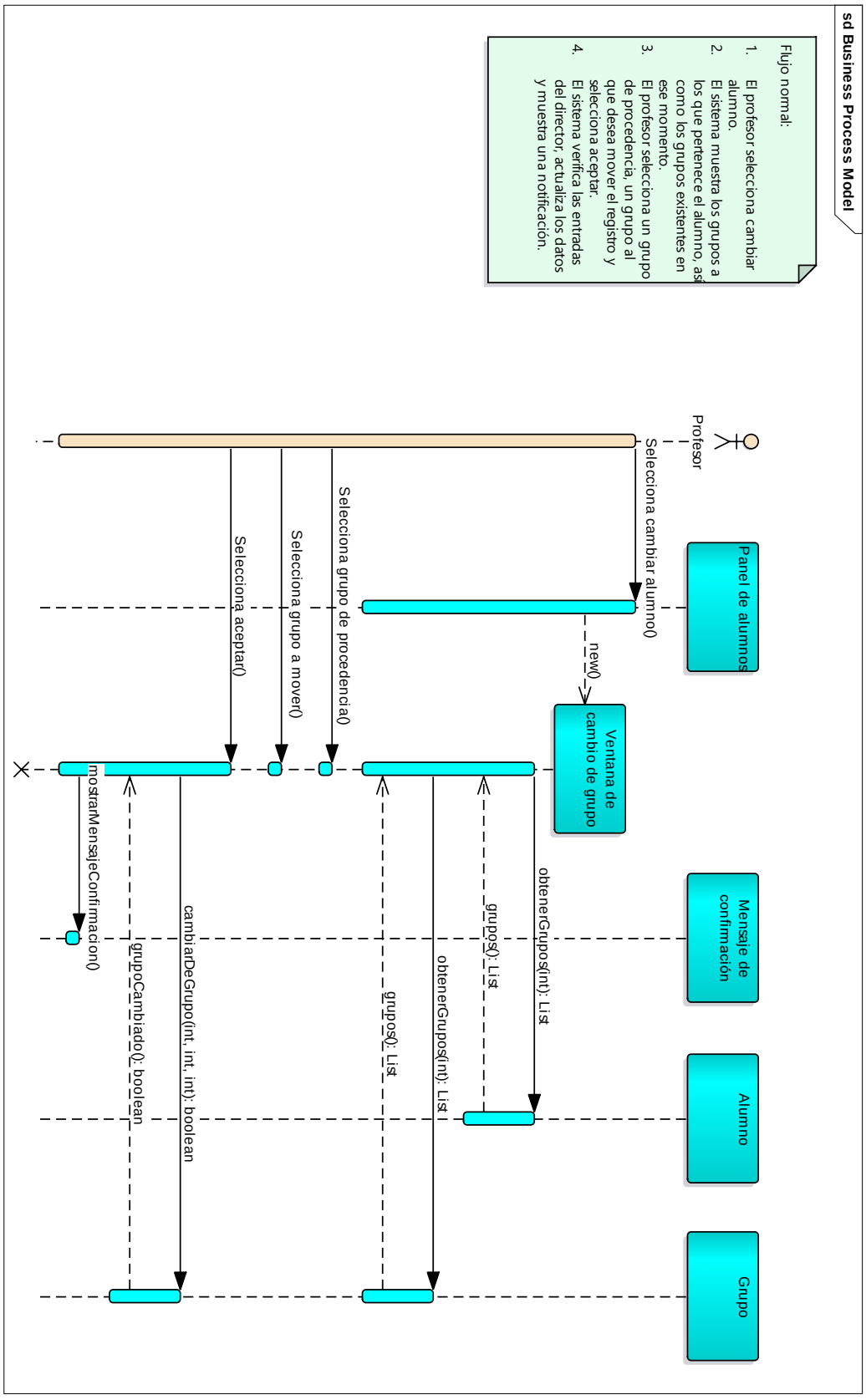
1.4.18. CU 7 – Registrar asistencia.



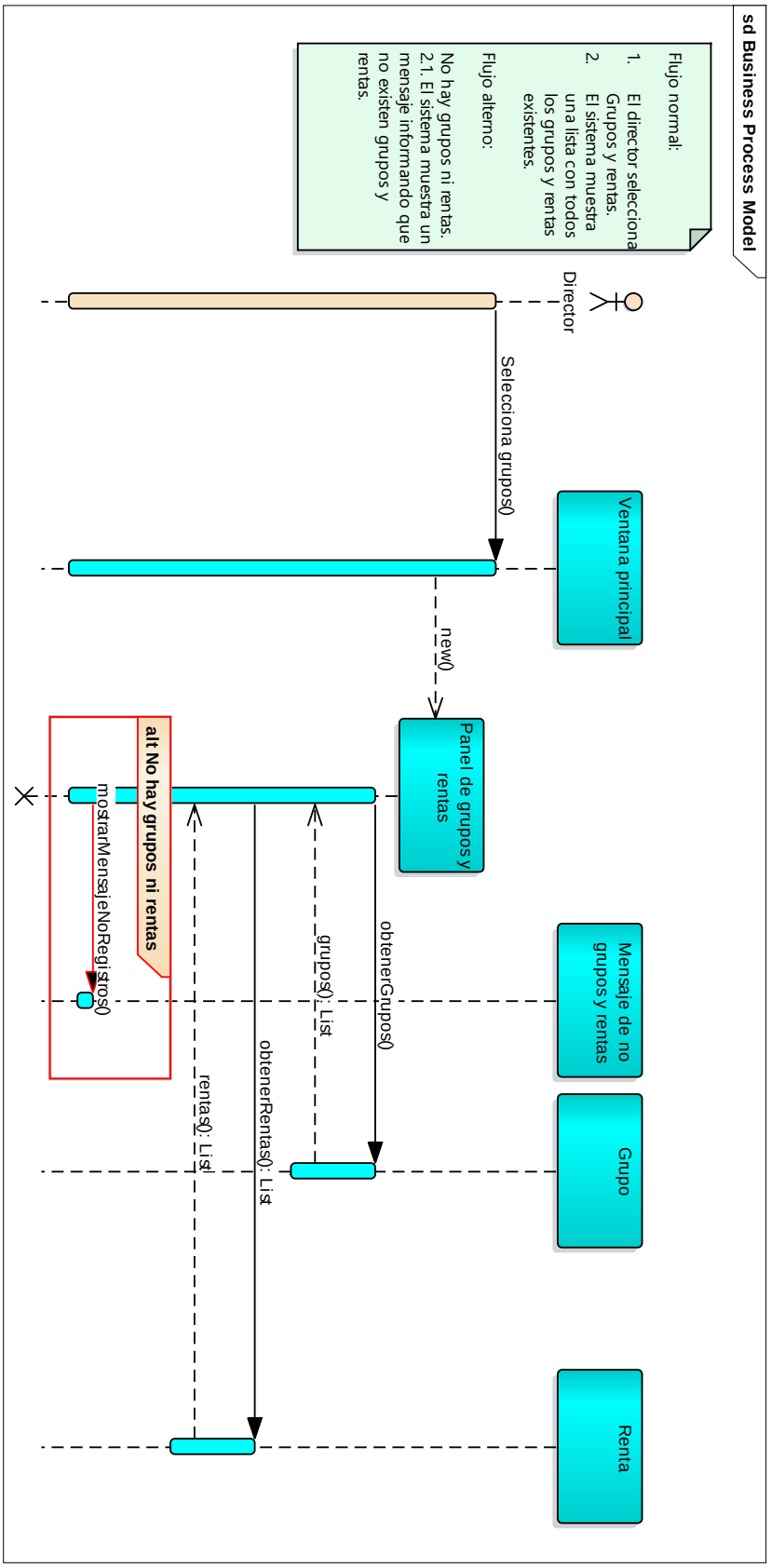
1.4.19. CU 8 – Inscribir alumno.



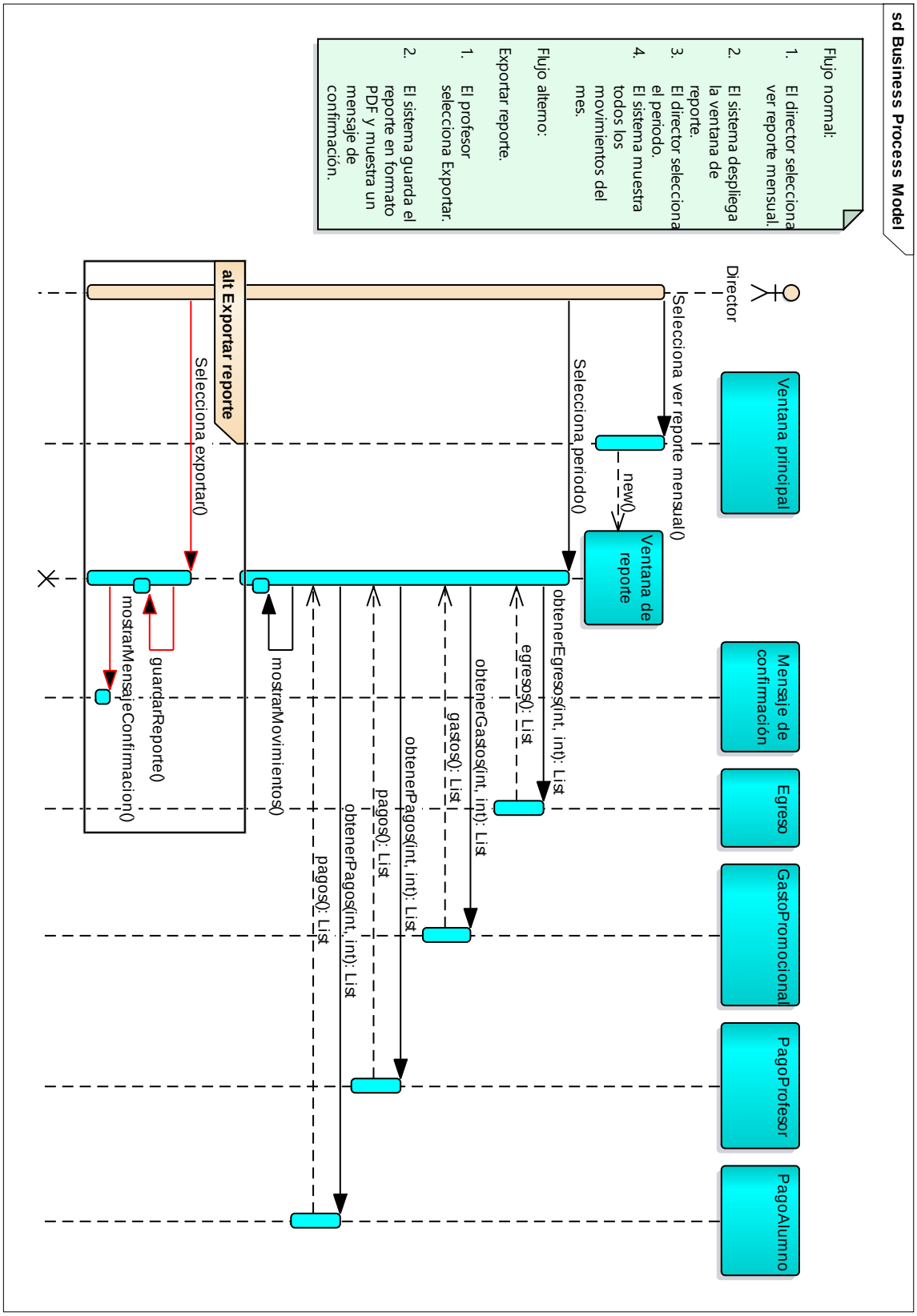
1.4.20. CU 11 – Cambiar alumno de grupo.



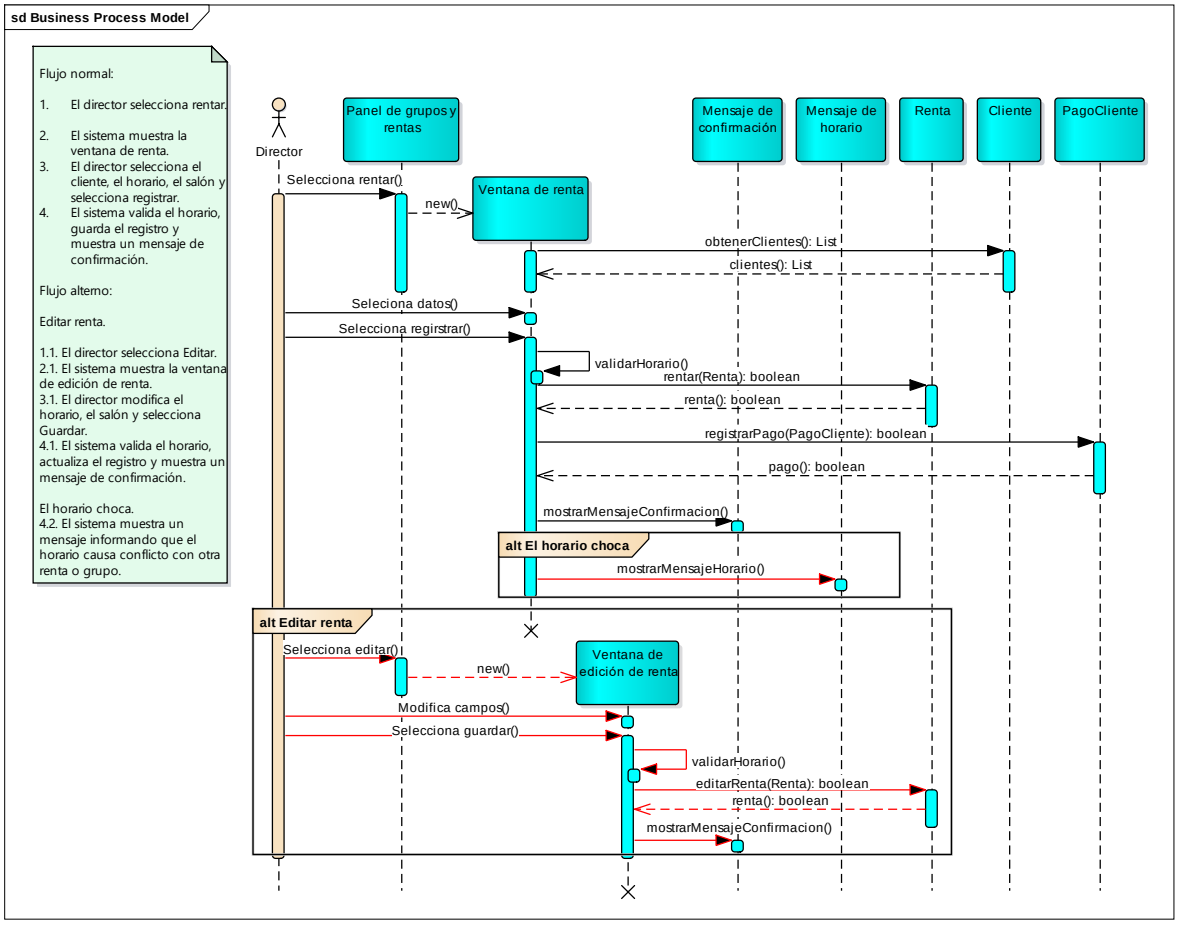
1.4.21. CU 15 – Consultar grupos y rentas.



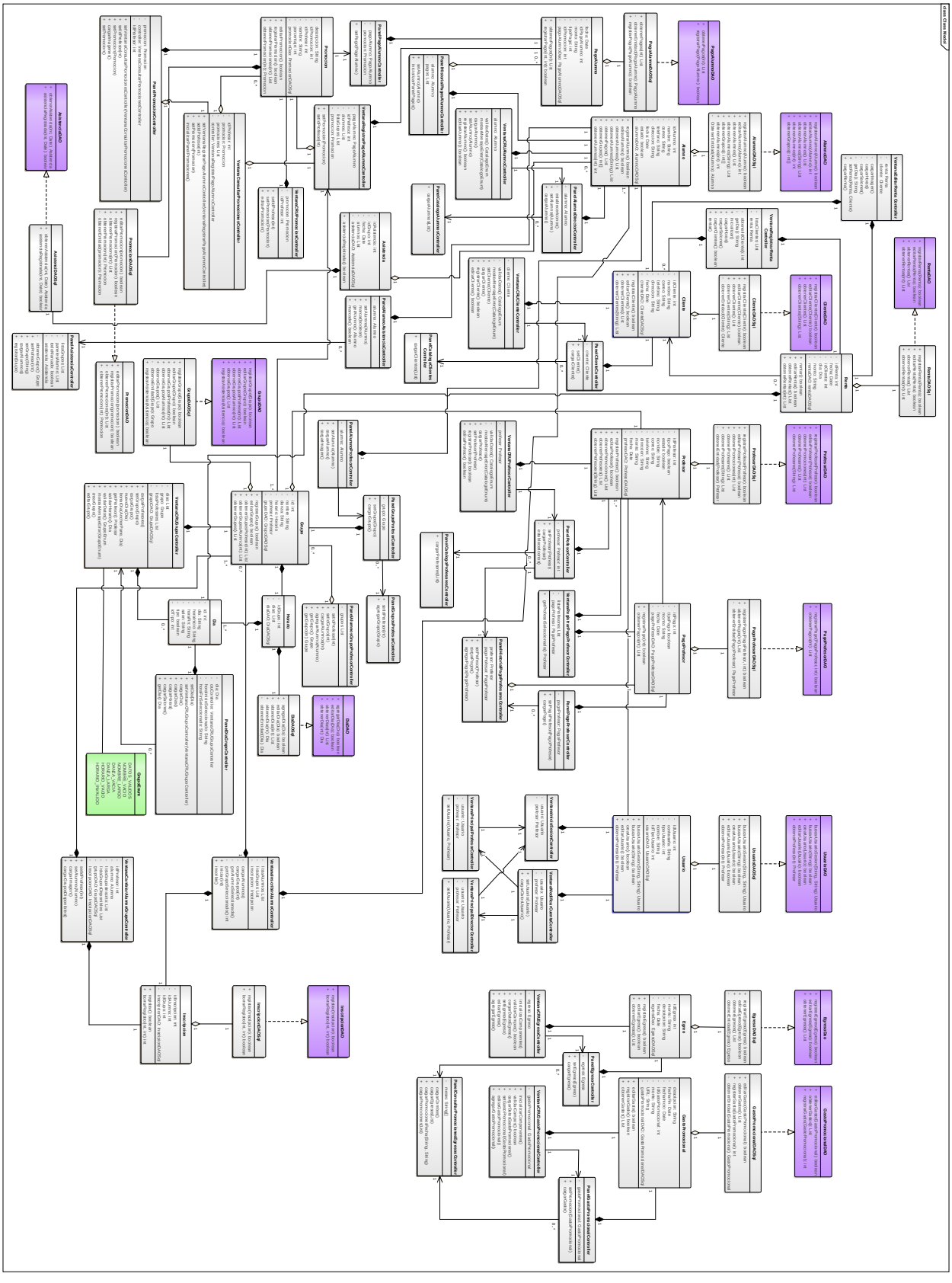
1.4.22. CU 18 – Generar reporte de ingresos y egresos.



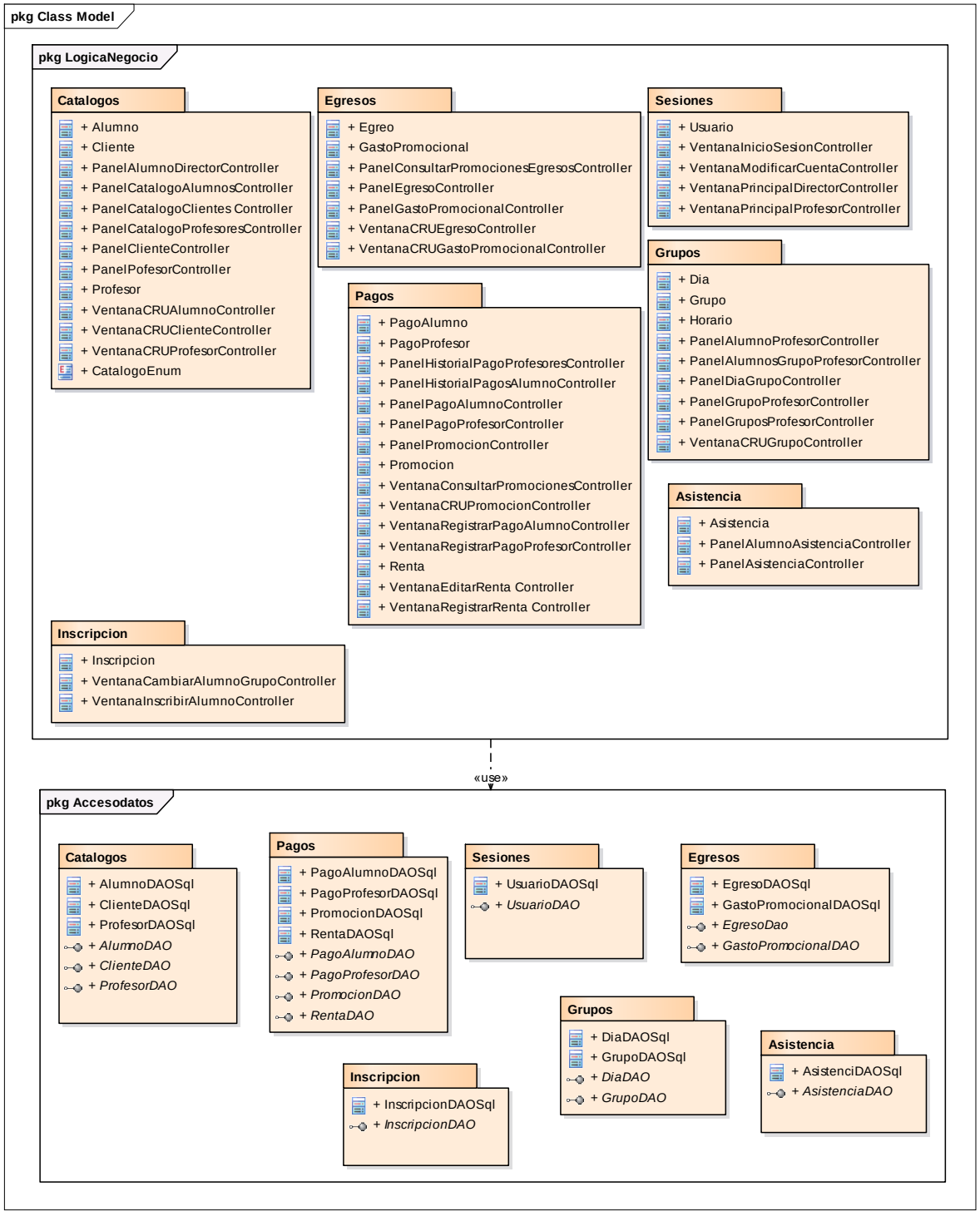
1.4.23. CU 19 – CRU Renta de espacio.



1.5. Diagrama de clases.

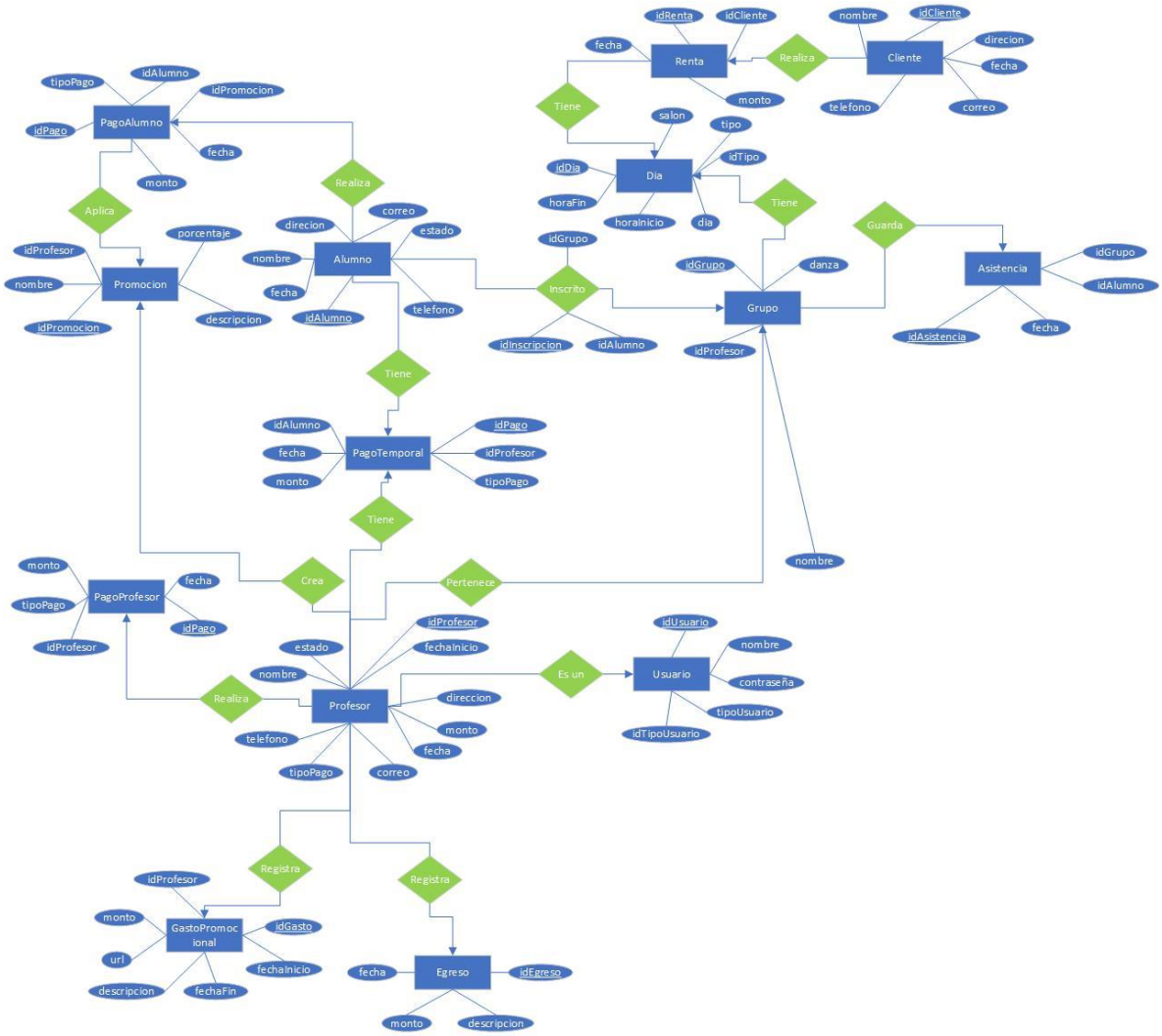


1.6. Diagrama de paquetes.



2. Modelos de datos.

2.1. Diagrama entidad – relación.



2.2. Modelo relacional.

PagoAlumno {idPago, idAlumno, idPromocion, tipoPago, monto, fecha}.

Promoción {idPromocion, idProfesor, nombre, porcentaje, descripcion}.

Alumno {idAlumno, nombre, fecha, dirección, correo, estado, teléfono}.

PagoProfesor {idProfesor, idPago, fecha, tipoPago, monto}.

Profesor {idProfesor, estado, nombre, teléfono, tipoPago, correo, fecha, dirección, monto, fechaInicio}.

Usuario {idUsuario, idTipousuario, tipoUsuario, contraseña, nombre}.

GastoPromocional {idGasto, fechaInicio, fechaFin, descripción, url, monto, idProfesor}.

Egreso {idEgreso, descripción, monto, fecha}.

Grupo {idGrupo, idProfesor, danza, nombre}.

Dia {idDia, horaInicio, horaFin, dia, idTipo, tipo, salón}.

Asistencia {idAsistencia, fecha, idAlumno, idGrupo}.

Renta {idRenta, idCliente, fecha, monto}.

Cliente {idCliente, nombre, teléfono, fecha, correo, direccion}.

PagoTemporal {idPago, tipoPago, fecha, monto, idAlumno, idProfesor}.

3. Plan de pruebas.

3.1. Introducción.

El proyecto de la asignatura desarrollo de Software para el que se realizó el presente plan de pruebas consiste en un centro de control para la escuela de danza Ared espacio, en esta existen diversas actividades como registro de estudiantes, egresos, pagos y grupos, y en conjunto la administración de todos estos (modificar y visualizar datos), dentro de este solo existen dos tipos de usuario, profesor y director, donde este último es un profesor con mayores privilegios de administración en el sistema.

3.1.1. Objetivo general.

El objetivo general del presente documento es validar el correcto funcionamiento de los módulos desarrollados en el proyecto.

3.1.2. Objetivos específicos.

I. Mostrar el correcto funcionamiento del sistema en diversas situaciones que podrían presentarse en el flujo normal de la aplicación.

II. Mostrar como reaccionará el sistema cuando una situación irregular suceda.

3.2. Plantillas de pruebas.

3.2.1. Modulo egresos.

3.2.1.1. Clase EgresoDAOSql.

Función	registrarEgreso: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Monto: 1200 - Descripción: Una escoba. - Fecha: 2012-12-12	Registro valido. true	N/A.	Registro valido de un nuevo egreso.	Registro valido. true
- Monto: monto invalido - Descripción: Una escoba. - Fecha: 2012-12-12	Registro no valido. false	N/A.	Notificación de un registro invalido.	Registro no valido. false

Función	editarEgreso: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- IdCosto: 3 - Monto: 1200 - Descripción: Una escoba. - Fecha: 2012-12-12	Edición valida: true	N/A.	Edición correcta.	Edición valida: true
- IdCosto: 500 - Monto: monto - Descripción: Una escoba. - Fecha: 2012-12-12	Edición no valida: false	N/A.	Edición incorrecta.	Edición no valida: false

Función	obtenerEgresos: list			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**

	List<egreso>	N/A.	Una lista con valores correctos de egresos.	List<Egreso>

3.2.1.2. Clase GastoPromocionalDAOSql.

Función	editarGasto: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Descripción: módulo de prueba. Fecha inicio:2012-12-12, - fecha fin: 2012-12-12 - monto: "1200" - URL: "www.uv.mx"	Edición de gasto valido	N/A.	Una edición de gasto valida.	Edición de gasto valido
- New Gasto();	Gasto invalido	N/A	Confirmación de que no puede registrarse el gasto.	gasto invalido.

Función	obtenerGastos : list			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
	List<Gasto>	N/A.	Una lista con valores correctos de gastos.	List<Gasto>

Función	registrarGasto: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Descripción: módulo de prueba. Fecha inicio:2012-12-12, - fecha fin: 2012-12-12 - monto: "1200"	Gasto valido. true	N/A.	Un registro de un profesor existente en el sistema.	Gasto valido. True

- URL: "www.uv.mx"				
Gasto invalido	Gasto invalido false	N/A	Confirmación de que no pueden registrarse gastos sin datos.	Gasto invalido false

3.2.2. Modulo sesiones.

3.2.2.1. Clase UsuarioDAOSql.

Función	buscarUsuarioSsion: Usuario			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Nombre: jose - Contraseña: snape	Usuario valido.	N/A.	Usuario existente en el sistema con todos los atributos correspondientes.	Usuario valido.
- Nombre: Mario. - Contraseña: 123	Usuario no valido.	N/A.	Un usuario nulo o confirmación de usuario no existente.	Usuario no valido.
- Nombre: jose. - Contraseña: snape	Identificador de usuario valido.	N/A	Un id de un usuario existente en el sistema.	Identificador de usuario valido.
- Nombre Gabriela - Contraseña: - Nemesis	Id de usuarios diferente.	N/A	Dos usuarios existentes no iguales.	Id de usuarios diferente.

Función	buscarUsuario: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Nombre: mario	False.	N/A.	Un usuario nulo o confirmación de usuario no existente.	False.
- Nombre: jose.	True.	N/A.	Un registro de un usuario existente.	True.

Función	crearUsuario: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- contraseña: 1234 - nombre: Antonio - idTipoUsuario: 1 - idUsuario: 1 - tipoUsuario: 1	true	N/A.	Un registro exitoso de un nuevo usuario.	true.

Función	obtenerProfesor: Profesor			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- contraseña: nemesis - nombre: Gabriela	Profesor valido.	N/A.	Un registro de un profesor existente en el sistema.	Profesor valido.
- idTipoUsuario: -1	Profesor invalido	N/A	Confirmación de que no existen los registros solicitados.	Profesor invalido.

3.2.3. Modulo pagos.

3.2.3.1. Clase PagoProfesorDAOSql.

Función	obtenerPagos(int idProfesor): List<PagoProfesor>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
idProfesor: 1.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	- El profesor con el id ‘1’ está registrado. - Existen o no pagos del profesor registrados.	Lista válida.
idProfesor: -1.	NullPointer-Exception	NullPointer-Exception	- El profesor con el id ‘-1’ no existe.	Lista vacía.

Función	registrarPago(PagoProfesor pago, int idProfesor): boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
PagoProfesor: - tipoPago: true. - monto: 900.5. idProfesor: 1.	True.	N/A.	- PagoProfesor con todas sus propiedades establecidas correctamente. - El profesor con el id ‘1’ está registrado.	True.
PagoProfesor: - tipoPago: true. - monto: 3t. idProfesor: 1.	False.	N/A.	- PagoProfesor con un monto no válido. - El profesor con el id ´1´ está registrado.	False.
PagoProfesor: - tipoPago: true. - monto: 900.5. idProfesor: -1.	False.	N/A.	- El profesor con el id ‘-1’ no existe.	False.

3.2.3.2. Clase PromocionDAOSql.

Función	editarPromocion(Promocion promocion): boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Promocion: - Descripción: promoción de prueba. - idPromocion: 1 - idProfesor: 1 - Nombre: promoción de decuento editada. - Porcentaje: 400	True.	N/A.	Se espera la adicción de una promoción existente.	True.

Promocion: - Descripción: promoción de prueba. - idPromocion: 1 - idProfesor: 0 - Nombre: promoción de decuento editada. Porcentaje: 20	False.	N/A.	Se espera que no se cree un registro de pago con el id de un profesor inexistente.	False.

Función	registrarPromocion(Promocion promocion): boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Nombre: promoción de prueba - idPromocion: 0 - idProfesor: 1 - Descripción: promoción de descuento. - Porcentaje: 20	True.	N/A.	Se espera el registro correcto de una promoción con un id de profesor existente.	True.
- Nombre: promoción de prueba - idPromocion: 0 - idProfesor: - 1 - Descripción: promoción de descuento. Porcentaje: 20	False.	N/A.	Se espera que no se cree un registro de una promoción con un id de un profesor no existente.	False.

Función	ObtenerPromociones(int idProfesor): list<promocion>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**

- Id profesor: 1	List <Promocion>	N/A.	Se espera una lista de promociones registradas por un profesor.	List <Promocion>
- Id profesor: 0	NullPointer Exception.	N/A.	Se espera que se lance una excepción sobre un id de profesor que no existe.	NullPointer Exception.

Función	obtenerPromocion(int idPromocion): Promocion			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- idPromocion. 1	Promocion.	N/A.	Para que esta prueba funcione debe existir en base de datos una promoción con ese id.	Promocion
- idPromocion: 0	NullPointer Exception.	N/A.	Para que esta prueba funcione no debe existir en base de datos una promoción con ese id.	NullPointer Exception.

3.2.3.3. Clase PagoAlumnoDAOSql.

Función	registrarPago(PagoAlumno pago, idProfesor, int idAlumno, int idPromocion: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Pago alumno: fecha: new Date(). idPago: 0. Monto: “500” idTipoPago: 1 - idAlumno: 1 - idPromocion: 1	True.	N/A.	Se espera un registro de un nuevo pago de alumno con todos los atributos pertenecientes.	True.

- Pago alumno: fecha: new Date(). idPago: 0. Monto: “no es un pago” idTipoPago: 1 - idAlumno: 1 - idPromocion: 1	False.	N/A.	Se espera que no se registre un pago de alumno con una cantidad no valida o con incoherencia de datos.	False.
- Pago alumno: fecha: new Date(). idPago: 0. Monto: “500” idTipoPago: 1 - idAlumno: 1 - TidPromocion: 0	True.	N/A.	Se espera el registro de un pago de alumno sin la propiedad de promoción.	True.

Función	registrarPago(PagoAlumno pago, idProfesor, int idAlumno, int idPromocion: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- idAlumno. 1	List <PagoAlumno>	N/A.	Se espera una lista de pagos de un alumno.	List <PagoAlumno>
- idAlumno: 0	NullPointer Exception.	N/A.	Se espera que el proceso arroje una excepción de tipo null debido a la no existencia del alumno.	NullPointer Exception.

3.2.3.4. Clase RentaDAOSql.

Función	obtenerRentas(int idCliente): List<Renta>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
idUsuario: 1	List<Renta>	N/A	Deben existir registros previos de rentas de ese cliente.	List<Renta>
idUsuario: 2	NullPointer Exception	N/A	El cliente ingresado no debe tener registros de rentas.	NullPointer Exception

Función	registrarRenta(Renta renta): boolen			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Renta: - fecha: 2018-05-12. - idCliente: 1. - idRenta: 0. - Monto: 250. - Dia: o Dia: Lunes. o horaFin: 12:00. o horaInicio: o 11:30. o Id: 0. o Tipo: false. o Salon: X.	True.	True.	- El cliente con id 1 existe en base de datos. - Todos los atributos de la renta están establecidos correctamente. - Las horas de inicio y fin del día no chocan con otro horario.	True
Renta: - fecha: 2018-05-12. - idCliente: 1. - idRenta: 0. - Monto: w. - Dia: o Dia: Lunes. o horaFin: 12:00. o horaInicio: o 11:30. o Id: 0. o Tipo: false. o Salon: X.	False.	False.	- El cliente con id 1 existe en base de datos. - El monto no es válido. - Las horas de inicio y fin del día no chocan con otro horario.	False.
Renta: - fecha: 2018-05-12. - idCliente: 1. - idRenta: 0.	Horario Exception.	“Las horas establecidas chocan con otro horario”.	- El cliente con id 1 existe en base de datos. - Todos los atributos de la	Horario Exception.

- Monto: 250.
- Dia:
  - Dia: Lunes.
  - horaFin: 18:00.
  - horaInicio: 17:30.
  - Id: 0.
  - Tipo: false.
  - Salon: X.
			renta están establecidos correctamente.
			- Las horas de inicio y fin del día chocan con otro horario.	

Función	editarRenta(Renta renta): boolen			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Renta: - Monto: 250. - Dia: o horaFin: 12:00. o horaInicio: o 11:30.	True.	True.	- Existe al menos un cliente registrado. - Existe al menos una renta registrada para el cliente. - Todos los atributos de la renta están establecidos correctamente. - Las horas de inicio y fin del día no chocan con otro horario.	True
Renta: - Id: 0. - Monto: 250. - Dia: o horaFin: 12:00. o horaInicio: 11:30.	False.	False.	- La renta con el id 0 no existe.	False.
Renta: - Monto: w. - Dia: o horaFin: 12:00. o horaInicio: o 11:30.	False.	False.	- El monto no es válido.	False.
Renta: - Monto: 250. - Dia: o horaFin: 12:00.	Horario Exception.	“Las horas establecidas chocan con otro horario”.	- Las horas establecidas chocan con otro horario.	Horario Exception.

○ horaInicio : 10:00.				

Función	List<Renta> obtenerRentas().			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
N/A.	Ninguna excepción relativa a la persistencia.	Ninguna excepción relativa a la persistencia.	Existen o no rentas registradas.	Lista válida.

3.2.4. Modulo catálogos.

3.2.4.1. Clase AlumnoDAOSql.

Función	registrarAlumno: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Alumno: - **Nombre**: Alumno de prueba. - **Correo**: alumno@prueba.clase - **Teléfono**: 2281917762. - **Dirección**: Dirección de prueba. - **Estado**: true.	N/A.	N/A.	Alumno con todas sus propiedades establecidas correctamente.	True.
Alumno: - **Nombre**: Alumno de prueba. - **Correo**: @com.gmail - **Teléfono**: 2281917762. - **Dirección**: Dirección de prueba. - **Estado**: true.	N/A.	N/A.	Alumno con todas sus propiedades establecidas correctamente, pero un correo no permitido.	False.
Alumno:	N/A.	N/A.	Alumno con todas sus propiedades	False.

- **Nombre**: Alumno de prueba. - **Correo**: alumno@prueba.clase - **Teléfono**: 228262z221. - **Dirección**: Dirección de prueba. - **Estado**: true.			establecidas correctamente, pero un teléfono no permitido.	

Función	editarAlumno: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Alumno existente en base.	N/A.	N/A.	Alumno ya registrado en base.	True.
Alumno existente en base: - Id: 0.	N/A.	N/A.	Alumno ya registrado en base, pero con un id inexistente.	False.
Alumno existente en base. - Correo: kjh.	N/A.	N/A.	Alumno ya registrado en base, pero con un correo no válido.	False.
Alumno existente en base. - Teléfono: 2281716s.	N/A.	N/A.	Alumno ya registrado en base, pero con un teléfono no válido.	False.

Función	obtenerAlumno: List<Alumno>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
N/A.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	Existen o no alumnos registrados.	Lista válida.

Función	obtenerAlumno: List<Alumno>

Entradas	Salidas esperadas	Salidas obtenidas	Condiciones de salida	Resultado
Nombre: victor.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	Cadena de texto para búsqueda por nombre. Existen o no alumnos registrados.	Lista válida.

Función	List<Alumno> obtenerAlumnos(int idGrupo)			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
idGrupo: 18.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	El id del grupo puede hacer referencia a un grupo que existe o no.	Lista válida.

3.2.4.2. Clase ClienteDAOSql.

Función	registrarCliente: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Cliente: - **Nombre**: Cliente de prueba. - **Correo**: cliente@prueba.clase - **Teléfono**: 2289365356. - **Dirección**: Dirección de prueba.	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente.	True.
Cliente: - **Nombre**: Cliente de prueba. - **Correo**: @com.gmail - **Teléfono**: 2289365356.	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente, pero un correo no permitido.	False.

- **Dirección**: Dirección de prueba.				
Cliente: - **Nombre**: Alumno de prueba. - **Correo**: cliente@prueba.clase - **Teléfono**: 228262z221. - **Dirección**: Dirección de prueba.	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente, pero un teléfono no permitido.	False.

Función	editarCliente: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Cliente existente en base.	N/A.	N/A.	Cliente ya registrado en base.	True.
Cliente existente en base: - Id: 0.	N/A.	N/A.	Cliente ya registrado en base, pero con un id inexistente.	False.
Cliente existente en base. - Correo: vija.com.	N/A.	N/A.	Cliente ya registrado en base, pero con un correo no válido.	False.
Cliente existente en base. - Teléfono: 1.	N/A.	N/A.	Cliente ya registrado en base, pero con un teléfono no válido.	False.

Función	obtenerCientes: List<Cliente>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
N/A.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	Existen o no clientes registrados.	Lista válida.

Función	obtenerCientes: List<Cliente>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Nombre: cli.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	Cadena de texto para búsqueda por nombre. Existen o no clientes registrados.	Lista válida.

3.2.4.3. Clase ProfesorDAOSql.

Función	registrarProfesor: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Profesor: - **Nombre**: Profesor de prueba. - **Correo**: profe@sor.clase - **Teléfono**: 2281916654. - **Dirección**: Dirección de prueba. - **Monto**: 1500.	N/A.	N/A.	Profesor con todas sus propiedades establecidas correctamente.	True.
Profesor: - **Nombre**: Profesor de prueba. - **Correo**: lkjhlkj - **Teléfono**: 2281916654. - **Dirección**: Dirección de prueba. - **Monto**: 1500.	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente, pero un correo no permitido.	False.
Cliente:	N/A.	N/A.	Cliente con todas sus propiedades establecidas	False.

- **Nombre**: Profesor de prueba. - **Correo**: profe@sor.clase - **Teléfono**: lkjhlkj. - **Dirección**: Dirección de prueba. - **Monto**: 1500.			correctamente, pero un teléfono no permitido.	
Profesor: - **Nombre**: Profesor de prueba. - **Correo**: profe@sor.clase - **Teléfono**: 2281916654. - **Dirección**: Dirección de prueba. - **Monto**: 1iuy.	N/A.	N/A.	Profesor con todas sus propiedades establecidas correctamente, pero un monto no permitido.	False.

Función	editarProfesor: boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Profesor existente en base.	N/A.	N/A.	Profesor ya registrado en base.	True.
Profesor existente en base: - Id: 0.	N/A.	N/A.	Profesor ya registrado en base, pero con un id inexistente.	False.
Profesor existente en base. - Correo: s.	N/A.	N/A.	Profesor ya registrado en base, pero con un correo no válido.	False.
Profesor existente en base. - Teléfono: 876.	N/A.	N/A.	Profesor ya registrado en base, pero con un teléfono no válido.	False.

Profesor existente en base. - Monto: 1.	N/A.	N/A.	Profesor ya registrado en base, pero con un monto no válido.	False.

Función	obtenerProfesores: List<Profesor>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
N/A.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	Existen o no profesores registrados.	Lista válida.

Función	obtenerProfesores: List<profesor>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Nombre: angel.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	Cadena de texto para búsqueda por nombre. Existen o no profesores registrados.	Lista válida.

3.2.5. Modulo grupos.

3.2.5.1. Clase DiaDAOSql.

Función	List<Dia> obtenerDias(int idGrupo)			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
idGrupo: 18.	Ninguna excepción relativa a la persistencia,	Ninguna excepción.	El id del grupo corresponde a un grupo que existe o no.	Lista válida.

Función	boolean agregarDia(Dia dia)			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Día: - Día: Lunes. - HoraFIn: 20:00.	True.	True.	El día tienes establecidos todos sus	True.

- HoraInicio: 19:00. - Id: 0. - idTipo: 5. - Salón: Y. - Tipo: true.			atributos correctamente.	

Función	boolean editarDia(Dia dia)			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Día: - Día: Viernes. - HoraFIn: 09:30. - HoraInicio: 09:00. - Id: 35. - idTipo: 18. - Salón: X. - Tipo: true.	True.	True.	El día existe en base de datos y tiene sus atributos establecidos correctamente.	True.
Día: - Día: Viernes. - HoraFIn: 09:30. - HoraInicio: 09:00. - Id: 0. - idTipo: 18. - Salón: X. - Tipo: true.	False.	False.	El día con el id 0 no existe.	False.

3.2.5.2. Clase GrupoDAOSql.

Función	List<Grupo> obtenerGrupos()			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Ninguna.	Ninguna excepción relativa a la persistencia.	Ninguna excepción.	Existen o no grupos registrados.	Lista válida.

Función	boolean registrarGrupo(Grupo grupo)			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**

Grupo: - Nombre: Soy un grupo. - Danza: Un tipo de danza. - Id: 0. - Profesor: id: 1. - Dia: o Dia: Miércoles. o Id: 0. o Inicio: 09:30. o Fin: 11:00. o Salón: X. o Tipo: true. o idTipo: 0.	True.	True.	- El grupo tiene sus atributos asignados correctamente - El día marcado tiene sus atributos correctos y no choca con otro horario en la base de datos. - El profesor con id 1 existe en base de datos.	True.
Grupo: - Nombre: Soy un grupo. - Danza: Un tipo de danza. - Id: 0. - Profesor: id: 0. - Dia: o Dia: Miércoles. o Id: 0. o Inicio: 09:30. o Fin: 11:00. o Salón: X. o Tipo: true. o idTipo: 0.	False.	False.	- El grupo tiene sus atributos asignados correctamente - El día marcado tiene sus atributos correctos y no choca con otro horario en la base de datos. - El profesor con id 0 no existe en base de datos.	False
Grupo: - Nombre: Soy un grupo. - Danza: Un tipo de danza. - Id: 0. - Profesor: id: 0. - Dia: o Dia: Lunes.	Horario Exception.	Horario Exception.	- El grupo tiene sus atributos asignados correctamente - El día marcado choca con otro horario en la base de datos.	Horario Exception.

○ Id: 0. ○ Inicio: 04:00. ○ Fin: 15:00. ○ Salón: X. ○ Tipo: true. ○ idTipo: 0.			- El profesor con id 1 existe en base de datos.	

Función	List<Grupo> obtenerGruposProfesor(int idProfesor)			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- idProfesor: 1.	Ninguna excepción relativa a la persistencia.	Ninguna excepción.	- El profesor con id 1 existe en base de datos.	Lista válida.
- idProfesor: 0.	Null Pointer Exception.	Null Pointer Exception.	- El profesor con id 0 no existe.	Null Pointer Exception.

Función	List<Grupo> obtenerGrupos()			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Ninguna.	Ninguna excepción relativa a la persistencia.	Ninguna excepción.	Existen o no grupos registrados.	Lista válida.

3.2.6. Modulo asistencia.

3.2.6.1. Clase AsistenciaDAOSql.

Función	asistenciaRegistrada(int idGrupo, Date fecha): boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
idGrupo: primer registro de la base. Fecha: actual.	True.	True.	1- Existe al menos un grupo registrado. 2- Existe al menos un registro de asistencia ara	True.

			el grupo en la fecha actual.	
idGrupo: 0. Fecha: actual.	False.	False.	3- El grupo con el id 0 no existe.	False.
idGrupo: primer registro de la base. Fecha: “2018-05-05”.	False.	False.	4- No existen registros de asistencia con la fecha establecida para el grupo.	False.

Función	obtenerAsistencias(int idGrupo): List<Asistencia>			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- Id grupo: 1	List < Asistencia >	N/A.	Se espera una lista de asistencias dependiente del id del grupo	List < Asistencia >

3.2.7. Modulo inscripciones.

3.2.7.1. Clase InscripcionDAOSql.

Función	Registrar(Inscripcion inscripcion): Usuario			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
Inscripcion: - idAlumno: primer id de la BD. - idGrupo: primer id de la BD. - idInscripcion: 0.	True.	True.	Existen al menos un alumno y un grupo en la BD.	True.
Inscripcion: - idAlumno: 0. - idGrupo: primer id de la BD. - idInscripcion: 0.	False.	False.	- Existe al menos un grupo en la BD. - El alumno con id 0 no existe.	False.
Inscripcion:	False.	False.	- Existe al menos un	False.

- idAlumno: primer id de la BD. - idGrupo: 0. - idInscripcion: 0.			grupo en la BD. - El grupo con id 0 no existe.	

Función	borrarRegistro(int idAlumno, int idGrupo): boolean			
Entradas	**Salidas esperadas**	**Salidas obtenidas**	**Condiciones de salida**	**Resultado**
- idAlumno: primer id de la BD. - idGrupo: primer id de la BD.	True.	True.	Existir al menos un grupo al que pertenezca el alumno.	True.
- idAlumno: 0. - idGrupo: 0	False.	False.	- El grupo con el id 0 no existe. - El alumno con id 0 no existe.	False.