



---

# REPORTE DE ENTREGA

---

40% del sistema



19 DE ABRIL DE 2018

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

## Índice

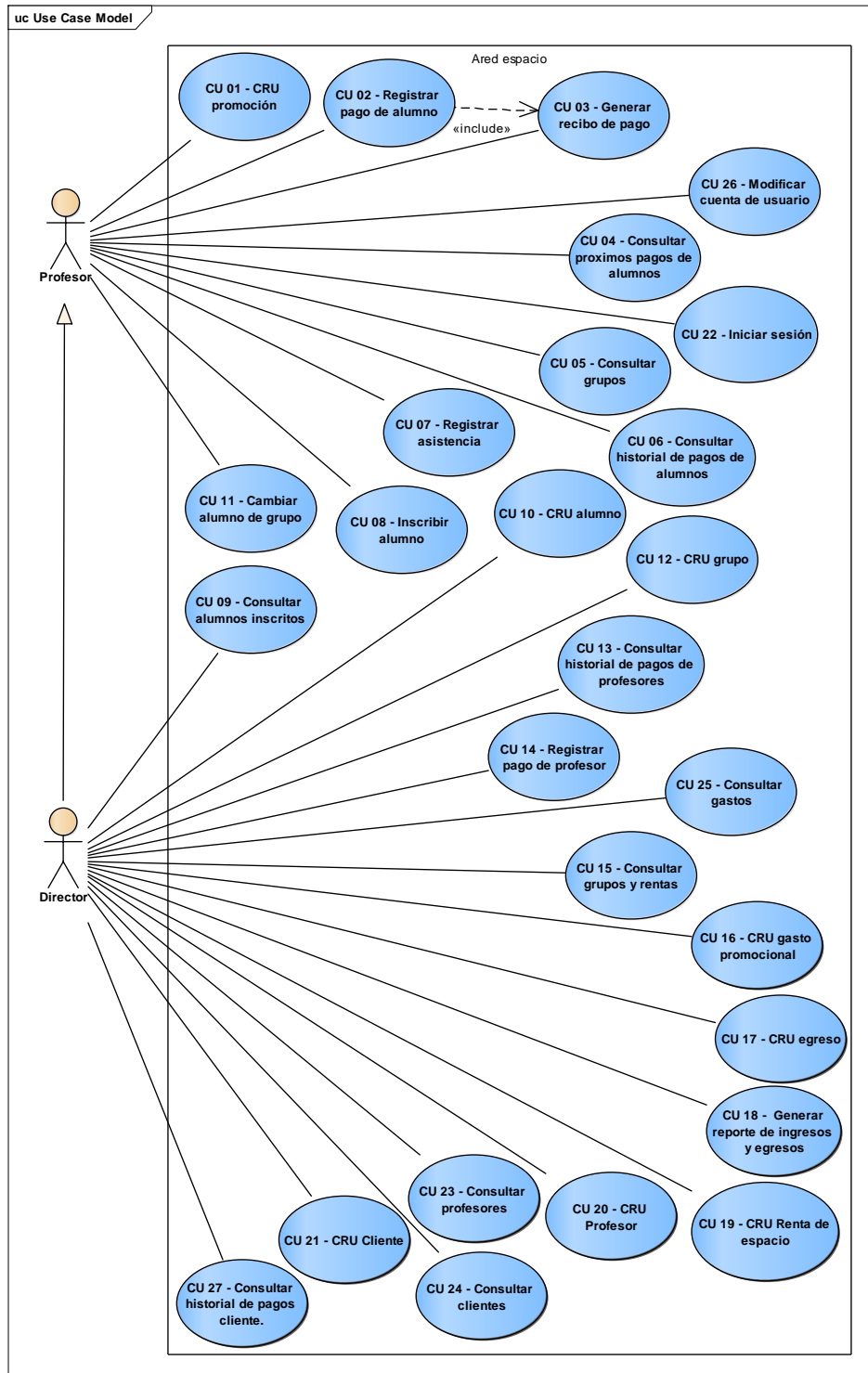
1.	Modelos de diseño.....	3
1.1.	Modelo de casos de uso.....	3
1.2.	Descripciones de casos de uso.....	4
1.3.	Diagramas de robustez.....	15
1.3.1.	CU 01 - CRU promoción.....	15
1.3.2.	CU 02 - Registrar pago del alumno.....	16
1.3.3.	CU 05 – Consultar grupos.....	17
1.3.4.	CU 09 – Consultar alumnos. ....	18
1.3.5.	CU 10 – CRU Alumno.....	19
1.3.6.	CU 12 – CRU Grupo. ....	20
1.3.7.	CU 13 – Consultar historial de pago de profesores.....	21
1.3.8.	CU 14 – Registrar pago de profesor. ....	21
1.3.9.	CU 16 – CRU gasto promocional. ....	22
1.3.10.	CU 17 – CRU Egreso.....	23
1.3.11.	CU 20 – CRU profesor.....	24
1.3.12.	CU 21 – CRU cliente.....	25
1.3.13.	CU 22 - Iniciar sesión. ....	26
1.3.14.	CU 23 – Consultar profesores.....	26
1.3.15.	CU 24 – Consultar clientes. ....	27
1.3.16.	CU 25 – Consultar gastos.....	27
1.3.17.	CU 26 – Modificar cuenta de usuario.....	28
1.4.	Diagramas de secuencia.....	29
1.4.1.	CU 01 – CRU Promoción.....	29
1.4.2.	CU 02 – Registrar pago de alumno.....	30
1.4.3.	CU 05 – Consultar grupos.....	31
1.4.4.	CU 09 – Consultar alumnos. ....	31
1.4.5.	CU 10 – CRU Alumno.....	32
1.4.6.	CU 12 – CRU Grupo. ....	33
1.4.7.	CU 13 – Consultar historial de pago de profesores.....	34
1.4.8.	CU 14 – Registrar pago de profesor. ....	35
1.4.9.	CU 16 – CRU Gasto promocional.....	36
1.4.10.	CU 17 – CRU Egreso.....	37

1.4.11.	CU 20 – CRU Profesor.....	38
1.4.12.	CU 21 – CRU Cliente. ....	39
1.4.13.	CU 22 – Iniciar sesión. ....	40
1.4.14.	CU 23 – Consultar profesores.....	41
1.4.15.	CU 24 – Consultar clientes. ....	41
1.4.16.	CU 25 – Consultar gastos.....	42
1.4.17.	CU 26 – Modificar cuenta de usuario.....	43
1.5.	Diagrama de clases.....	44
1.6.	Diagrama de paquetes. ....	45
2.	Modelos de datos.....	46
2.1.	Diagrama entidad – relación. ....	46
2.2.	Modelo relacional. ....	47
3.	Plan de pruebas.....	47
3.1.	Introducción. ....	47
3.1.1.	Objetivo general.....	47
3.1.2.	Objetivos específicos.....	47
3.2.	Plantillas de pruebas. ....	48
3.2.1.	Modulo egresos.....	48
3.2.2.	Modulo sesiones. ....	50
3.2.3.	Modulo pagos.....	51
3.2.4.	Modulo catálogos.....	55
3.2.5.	Modulo grupos. ....	62

## 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.

<b>Nombre del caso de uso.</b>	CU-01 CRU promoción.
<b>Actor.</b>	Profesor, director.
<b>Descripción.</b>	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.
<b>Precondiciones.</b>	<ol style="list-style-type: none"> <li>1. El profesor o director debe estar autenticado.</li> <li>2. En caso de crearse una promoción, esta no debe existir previamente en el sistema.</li> </ol>
<b>Postcondiciones.</b>	<ol style="list-style-type: none"> <li>1. Debe crearse una promoción en el sistema.</li> <li>2. La promoción creada solo puede ser vista por el usuario que la creó.</li> </ol>
<b>Flujo normal.</b>	<ol style="list-style-type: none"> <li>1. El usuario del sistema selecciona promociones.</li> <li>2. El sistema muestra la ventana general de promociones existentes.</li> <li>3. El usuario selecciona agregar promoción.</li> <li>4. El sistema solicita los datos necesarios para crear la promoción (nombre y descripción) y selecciona guardar.</li> <li>5. El usuario ingresa el nombre, descripción, porcentaje y selecciona registrar.</li> <li>6. El sistema valida los datos del usuario, guarda la promoción y notifica la operación al usuario.</li> </ol>
<b>Flujo alterno.</b>	<p>Modificar promoción:</p> <ol style="list-style-type: none"> <li>3.1. El usuario selecciona editar en una promoción existente.</li> <li>4.1. El sistema muestra los detalles existentes de la promoción seleccionada.</li> <li>5.1. El usuario modifica los datos necesarios y selecciona guardar cambios.</li> <li>6.1. El sistema valida los datos del usuario, guarda la promoción y notifica la operación al usuario.</li> </ol>
<b>Excepciones.</b>	<p>No pueden almacenarse los datos</p> <ol style="list-style-type: none"> <li>4.1. El sistema muestra al usuario que los datos no pueden ser guardados, que intente más tarde.</li> </ol>

<b>Nombre del caso de uso.</b>	CU-02 Registrar pago del alumno.
<b>Actor.</b>	Profesor, director.
<b>Descripción.</b>	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.
<b>Precondiciones.</b>	<ol style="list-style-type: none"> <li>1. El usuario debe estar autenticado en el sistema.</li> </ol>

	2. Debe existir un registro del alumno al que se desea asignar el pago.
<b>Postcondiciones.</b>	1. Debe generarse un registro del pago del alumno. 2. Debe eliminarse cualquier notificación de próximos pagos en esa semana para ese alumno.
<b>Flujo normal.</b>	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.
<b>Flujo alterno.</b>	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.
<b>Excepciones.</b>	No pueden almacenarse los datos 4.1. El sistema muestra al usuario que los datos no pueden ser guardados, que intente más tarde.

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

	<p>2.1. El sistema muestra un mensaje informando sobre el error.</p> <p>4.1. El sistema muestra un mensaje informando sobre el error.</p>
--	---

<b>Nombre del caso de uso.</b>	CU 09 – Consultar alumnos.
<b>Actor.</b>	Director.
<b>Descripción.</b>	El director puede consultar todos los alumnos existentes en la institución.
<b>Precondiciones.</b>	<ol style="list-style-type: none"> <li>1. Deben existir alumnos inscritos en la institución.</li> <li>2. El director tiene que estar autenticado en el sistema.</li> </ol>
<b>Postcondiciones.</b>	
<b>Flujo normal.</b>	<ol style="list-style-type: none"> <li>1. El director selecciona alumnos en la institución.</li> <li>2. El sistema busca los alumnos existentes en el sistema y los muestra al director.</li> </ol>
<b>Flujo alterno.</b>	<p>No hay alumnos inscritos.</p> <ol style="list-style-type: none"> <li>1.1. El sistema muestra al director que no existen alumnos en la institución.</li> <li>1.2. El sistema regresa a una pantalla anterior del sistema.</li> </ol>
<b>Excepciones.</b>	<p>No existe conexión a la base de datos</p> <ol style="list-style-type: none"> <li>1. El sistema muestra al usuario que no hay conexión con la base de datos, que intente nuevamente más tarde.</li> </ol>

<b>Nombre del caso de uso.</b>	CU 10 – CRU Alumno.
<b>Actor.</b>	Director.
<b>Descripción.</b>	En este caso de uso, el director podrá administrar la información de los alumnos creando y modificando los registros.
<b>Precondiciones.</b>	
<b>Postcondiciones.</b>	<ol style="list-style-type: none"> <li>1. Los profesores podrán ver los cambios y el director podrá ver los nuevos registros.</li> </ol>
<b>Flujo normal.</b>	<ol style="list-style-type: none"> <li>1. El director selecciona Registrar alumno.</li> <li>2. El sistema muestra la ventana registro de alumno.</li> <li>3. El director ingresa el nombre, teléfono, dirección, correo, selecciona una imagen y selecciona Registrar.</li> <li>4. El sistema valida los campos, guarda el registro y muestra un mensaje de confirmación.</li> </ol>
<b>Flujo alterno.</b>	<p>Modificar alumno.</p> <ol style="list-style-type: none"> <li>1.1. El profesor selecciona Editar alumno.</li> <li>2.2. El sistema muestra la ventana de edición de alumno</li> <li>3.1. El director modifica los campos deseados y selecciona Guardar.</li> </ol>

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

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

<b>Nombre del caso de uso.</b>	CU 13 – Consultar historial de pago de profesores.
--------------------------------	--



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

<b>Nombre del caso de uso.</b>	CU 14 – Registrar pago de profesor.
<b>Actor.</b>	Director.
<b>Descripción.</b>	El director deberá guardar un registro de un pago realizado por el profesor.
<b>Precondiciones.</b>	1. Deben existir profesores registrados en el sistema. 2. Debe estar autenticado el director en el sistema.
<b>Postcondiciones.</b>	1. Debe generarse un registro de pago en el sistema. 2. Debe mostrar el pago del profesor en el historial de pagos realizado.
<b>Flujo normal.</b>	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.
<b>Flujo alterno.</b>	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.
<b>Excepciones.</b>	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.

<b>Nombre del caso de uso.</b>	CU 16 – CRU gasto promocional.
--------------------------------	--------------------------------

<b>Actor.</b>	Director.
<b>Descripción.</b>	El director podrá administrar los gastos promocionales de Facebook.
<b>Precondiciones.</b>	1. No debe existir el gasto promocional en un registro previo.
<b>Postcondiciones.</b>	1. Debe crearse un registro de promoción en el sistema.
<b>Flujo normal.</b>	<ol style="list-style-type: none"> <li>1. El director selecciona gasto promocional.</li> <li>2. El sistema solicita los detalles para generar un registro gasto promocional.</li> <li>3. El director ingresa una descripción, un enlace, un monto, fecha inicio, fecha de fin y selecciona aceptar.</li> <li>4. El sistema valida los datos ingresados al sistema, guarda los datos y notifica al director el registro exitoso.</li> </ol>
<b>Flujo alterno.</b>	<p>Editar gasto</p> <ol style="list-style-type: none"> <li>1.1. El director selecciona editar</li> <li>2.1. El sistema muestra la ventana de edición de gastos</li> <li>3.1. El director selecciona las fechas de inicio, fin y selecciona guardar.</li> <li>4.1. El sistema actualiza el registro y notifica al director el registro.</li> </ol> <p>Los datos son incorrectos</p> <ol style="list-style-type: none"> <li>3.1. el sistema muestra un mensaje al usuario que los datos son incorrectos.</li> <li>4.1 el flujo regresa al paso 2 del flujo normal.</li> </ol>
<b>Excepciones.</b>	<p>No existe conexión a base de datos del sistema</p> <ol style="list-style-type: none"> <li>1. El sistema muestra al usuario que no existe conexión con el sistema, que intente más tarde.</li> </ol>

<b>Nombre del caso de uso.</b>	CU 17 – CRU Egreso.
<b>Actor.</b>	Director.
<b>Descripción.</b>	En este caso de uso, el director puede administrar los egresos de la institución.
<b>Precondiciones.</b>	
<b>Postcondiciones.</b>	1. El director debe poder ver el registro realizado.
<b>Flujo normal.</b>	<ol style="list-style-type: none"> <li>1. El director selecciona registrar egreso.</li> <li>2. El sistema muestra la ventana de registro de egreso.</li> <li>3. El director selecciona la fecha, ingresa una descripción, el costo y selecciona registrar.</li> <li>4. El sistema valida los datos, registra el gasto y muestra un mensaje de confirmación.</li> </ol>

<b>Flujo alterno.</b>	<p>Editar egreso</p> <ol style="list-style-type: none"> <li>1.1. El director selecciona editar.</li> <li>2.1. El sistema muestra la ventana de edición de egreso.</li> <li>3.1. El director selecciona una fecha y guardar.</li> <li>4.1. El sistema actualiza el registro y muestra el mensaje de confirmación al director.</li> </ol> <p>Los datos son incorrectos</p> <ol style="list-style-type: none"> <li>3.1. el sistema muestra un mensaje al usuario que los datos son incorrectos.</li> <li>4.1 el flujo regresa al paso 2 del flujo normal.</li> </ol>
<b>Excepciones.</b>	<p>No se puede guardar el registro.</p> <ol style="list-style-type: none"> <li>4.2. El sistema muestra un mensaje informando sobre el error.</li> </ol>

<b>Nombre del caso de uso.</b>	CU 20 – CRU profesor.
<b>Actor.</b>	Director.
<b>Descripción.</b>	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.
<b>Precondiciones.</b>	1. Para crear un nuevo registro el profesor que se desea almacenar no debe encontrarse dentro de los datos existentes.
<b>Postcondiciones.</b>	<ol style="list-style-type: none"> <li>1. Debe crearse un nuevo registro de un profesor en el sistema.</li> <li>2. En su defecto debe crearse una actualización de los datos del profesor seleccionado.</li> </ol>
<b>Flujo normal.</b>	<ol style="list-style-type: none"> <li>1. El director selecciona agregar profesor.</li> <li>2. El sistema muestra todos los datos que deben ser cubiertos por el profesor para crear el registro.</li> <li>3. El director ingresa los datos del nuevo profesor como su nombre, dirección, teléfono, correo, monto, tipo de pago, monto y selecciona guardar.</li> <li>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.</li> </ol>
<b>Flujo alterno.</b>	<p>Editar profesor.</p> <ol style="list-style-type: none"> <li>1.1. El director selecciona editar profesor.</li> <li>2.1. El sistema muestra los datos del usuario seleccionado.</li> <li>3.1. El director modifica los datos del usuario que se ha seleccionado y que considera pertinentes y selecciona guardar.</li> </ol>

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

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

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

<b>Nombre del caso de uso.</b>	CU-22 Iniciar sesión.
<b>Actor.</b>	Director, profesor.
<b>Descripción.</b>	En este caso de uso el usuario ingresa al sistema dependiendo los privilegios proporcionados en su registro.
<b>Precondiciones.</b>	1. El usuario debe existir en un registro previo.
<b>Postcondiciones.</b>	1. El usuario deberá visualizar todos los contenidos relacionados con su registro previo.
<b>Flujo normal.</b>	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.
<b>Flujo alterno.</b>	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.
<b>Excepciones.</b>	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.

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

	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.
<b>Excepciones.</b>	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.

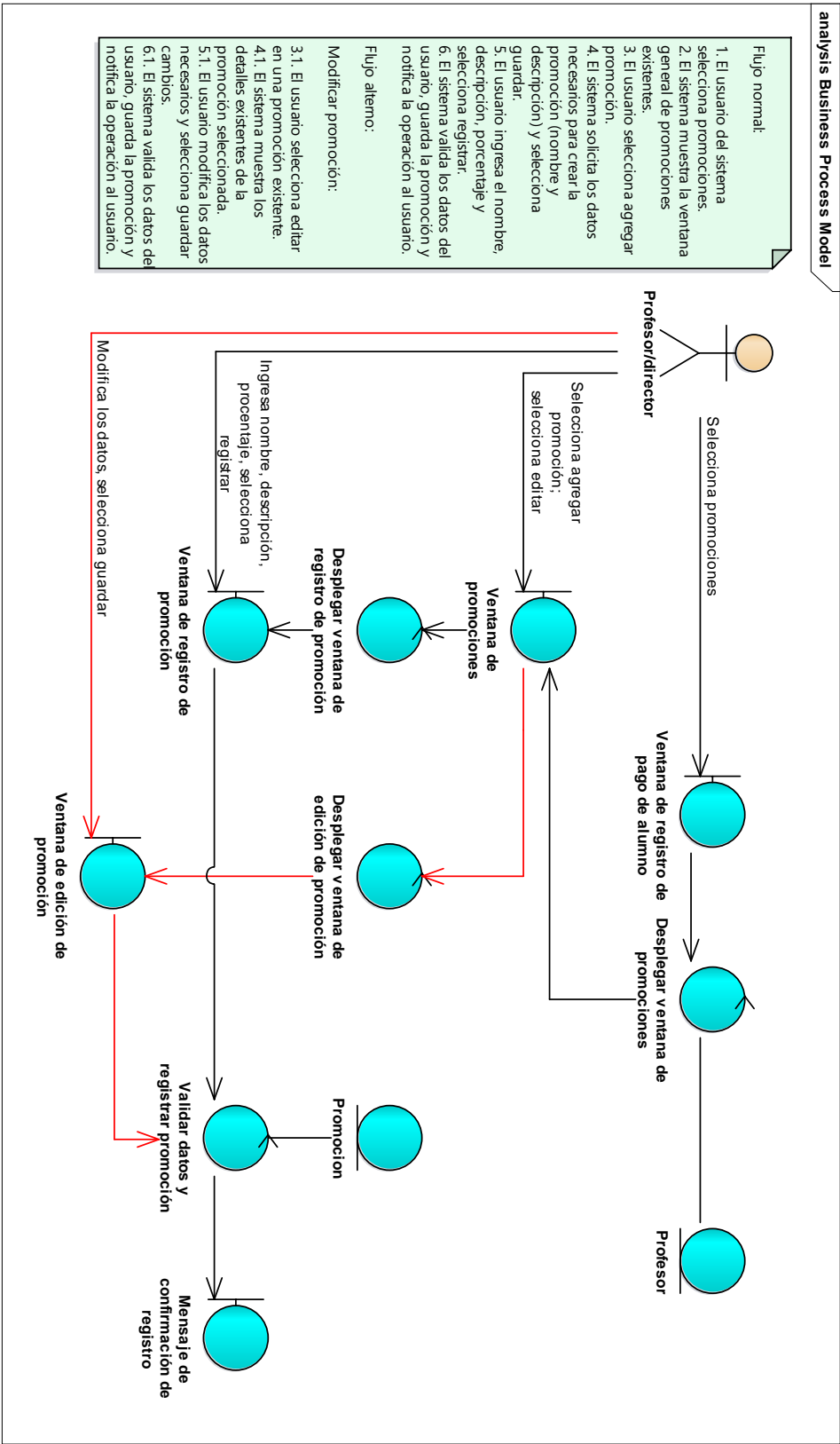
<b>Nombre del caso de uso.</b>	CU 24 – consultar clientes.
<b>Actor.</b>	Director.
<b>Descripción.</b>	En este caso de uso el director puede consultar todos los profesores clientes registrados en el sistema.
<b>Precondiciones.</b>	3. El director debe tener una sesión iniciada en el sistema. 4. Deben existir clientes registrados en el sistema.
<b>Postcondiciones.</b>	
<b>Flujo normal.</b>	3. El director selecciona consultar clientes. 4. El sistema busca todos los clientes existentes en el sistema y los muestra al director.
<b>Flujo alterno.</b>	El director busca a un cliente específico 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.
<b>Excepciones.</b>	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.

<b>Nombre del caso de uso.</b>	CU 25 – Consultar gastos.
<b>Actor.</b>	Director.
<b>Descripción.</b>	En este caso de uso, el director puede consultar todos los gastos realizados de promociones de Facebook y egresos.
<b>Precondiciones.</b>	1. Debe de existir al menos un gasto/promoción registrado.
<b>Postcondiciones.</b>	1. El director debe poder ver los registros.
<b>Flujo normal.</b>	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.
<b>Flujo alterno.</b>	No hay gastos registrados.  2.1. El sistema muestra un mensaje informando que no existen gastos registrados.
<b>Excepciones.</b>	No se pueden obtener los registros.  1. El sistema muestra un mensaje informando sobre el error.

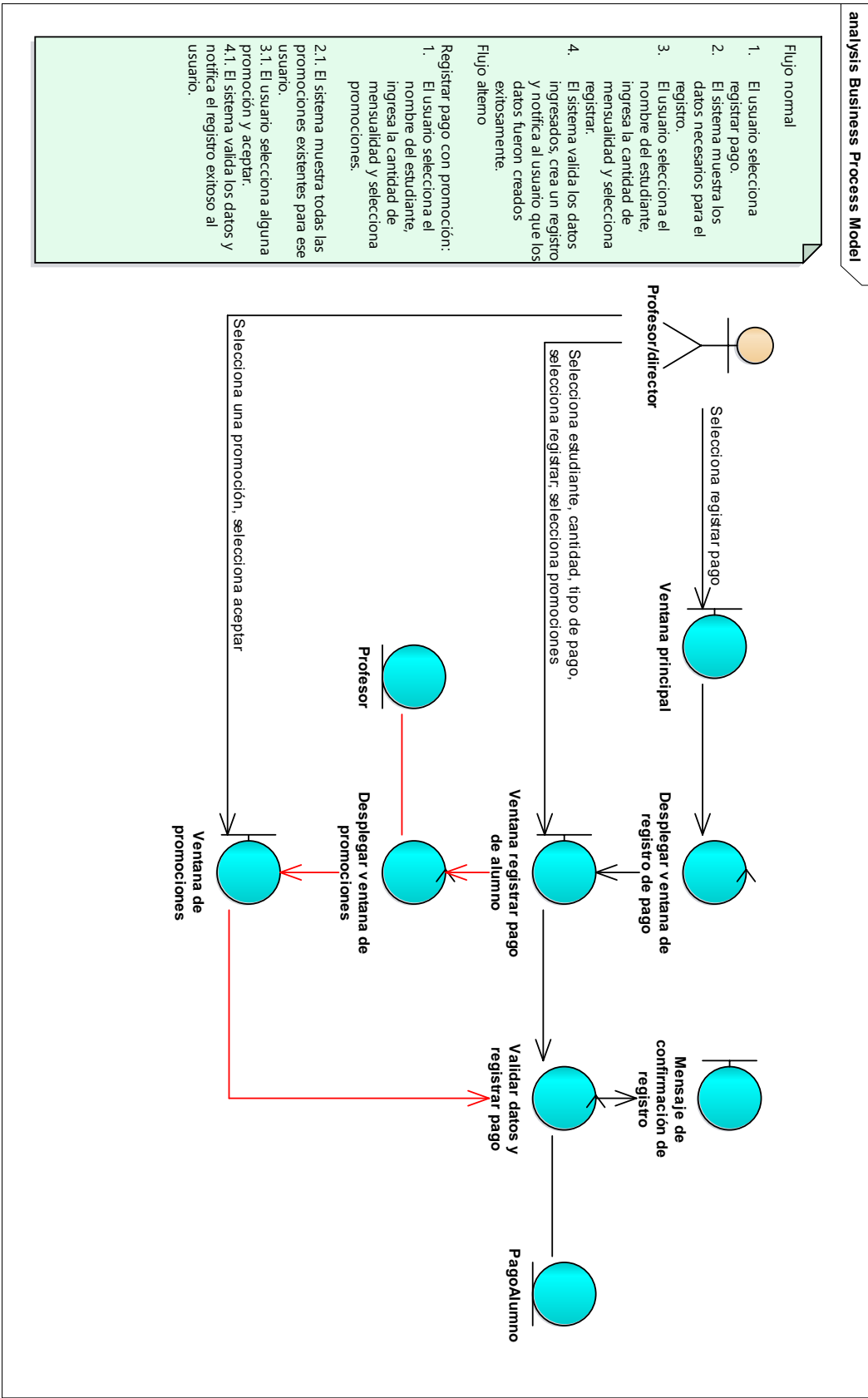
<b>Nombre del caso de uso.</b>	CU 26 – Modificar cuenta de usuario.
<b>Actor.</b>	Profesor.
<b>Descripción.</b>	El este caso de uso el profesor podrá modificar sus datos de usuario y contraseña del sistema.
<b>Precondiciones.</b>	5. Deben existir un registro previo del profesor que desea modificar los datos. 6. No debe existir un usuario con el nombre indicado por el profesor.
<b>Postcondiciones.</b>	1. Deben actualizarse 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.
<b>Flujo normal.</b>	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.
<b>Flujo alterno.</b>	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.
<b>Excepciones.</b>	No existe conexión con la base de datos. 3. El sistema muestra un mensaje sobre la conexión y le solicita que intente más tarde.

1.3. Diagramas de robustez.  
1.3.1. CU 01 - CRU promoción.





1.3.2. CU 02 - Registrar pago del alumno.



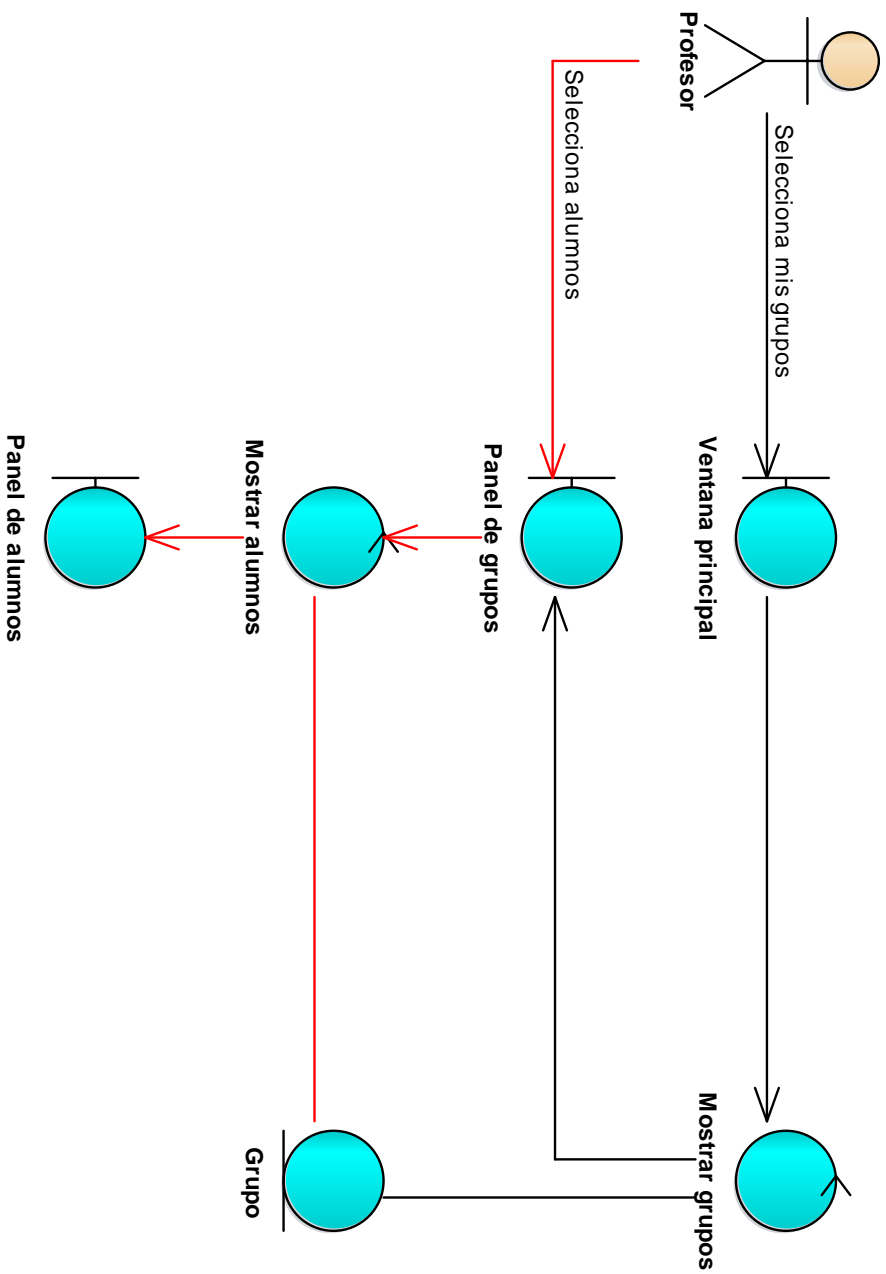
analysis Business Process Model

Flujo normal

1. El profesor selecciona Mis grupos.
2. El sistema muestra los grupos del profesor.

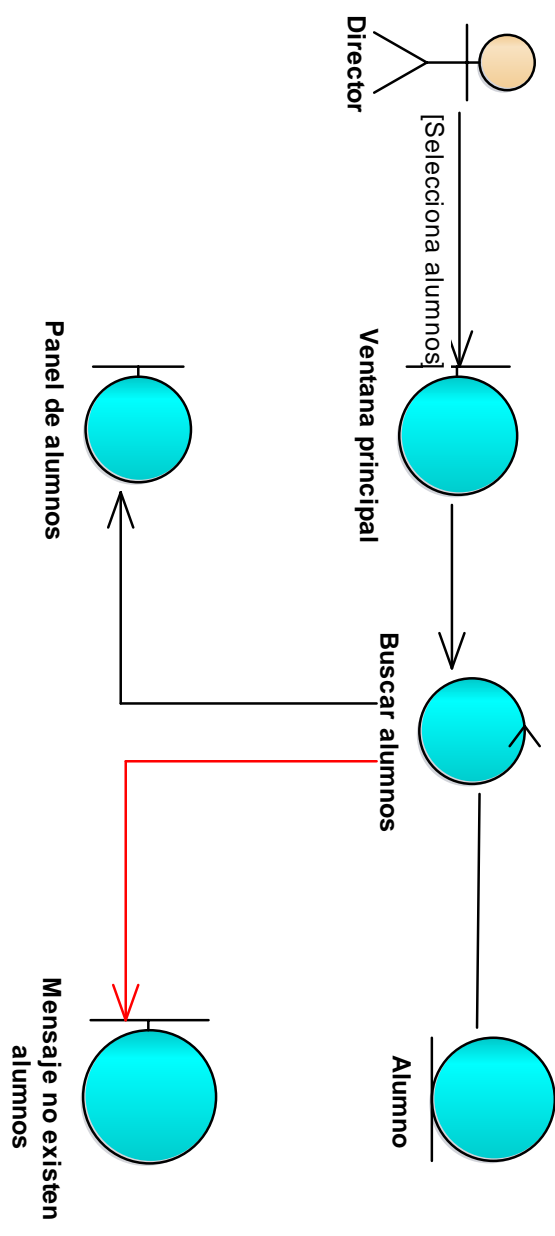
Flujo alterno:

3. El profesor selecciona Alumnos en un grupo.
4. El sistema muestra los alumnos inscritos al curso seleccionado.

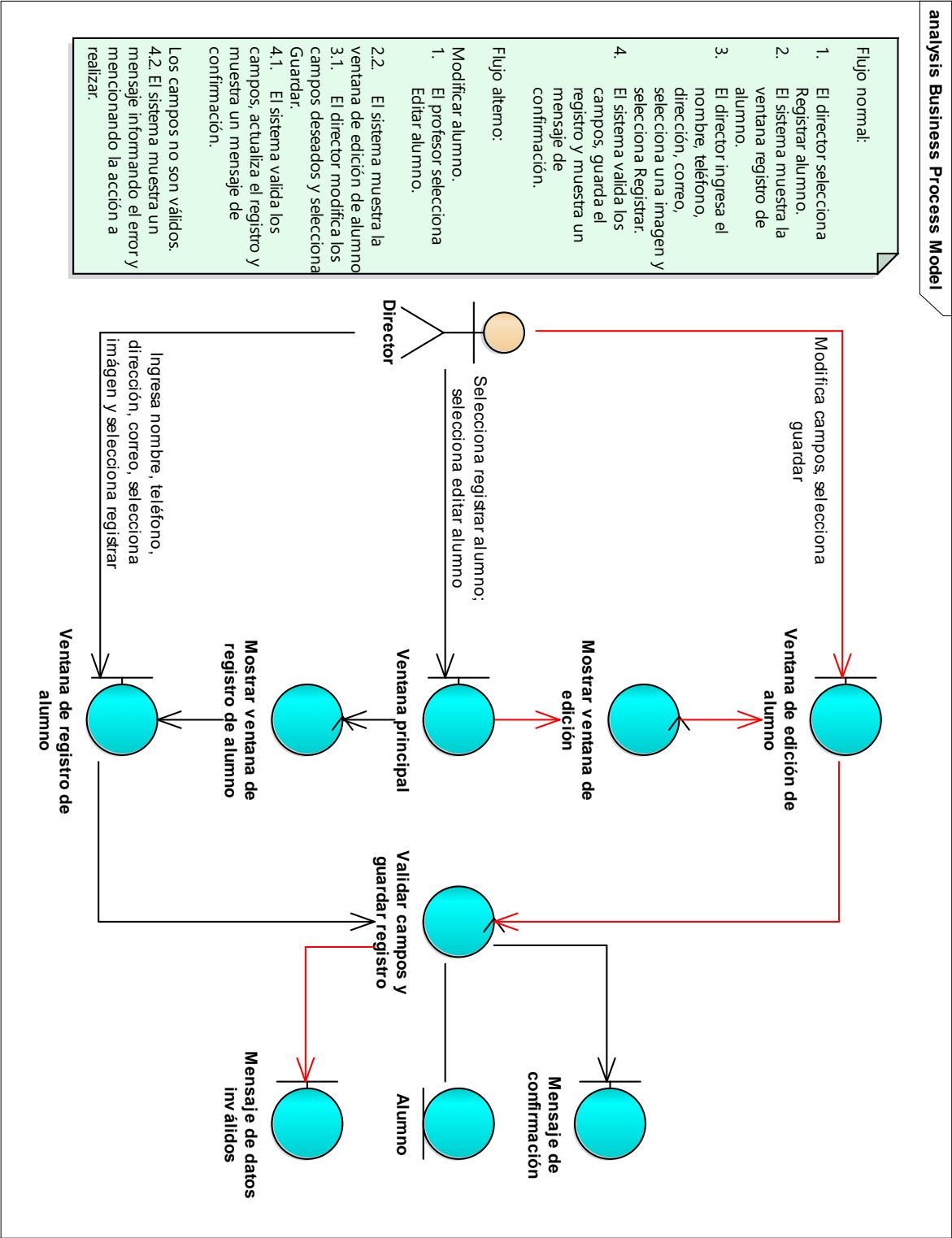


1.3.3. CU 05 – Consultar grupos.

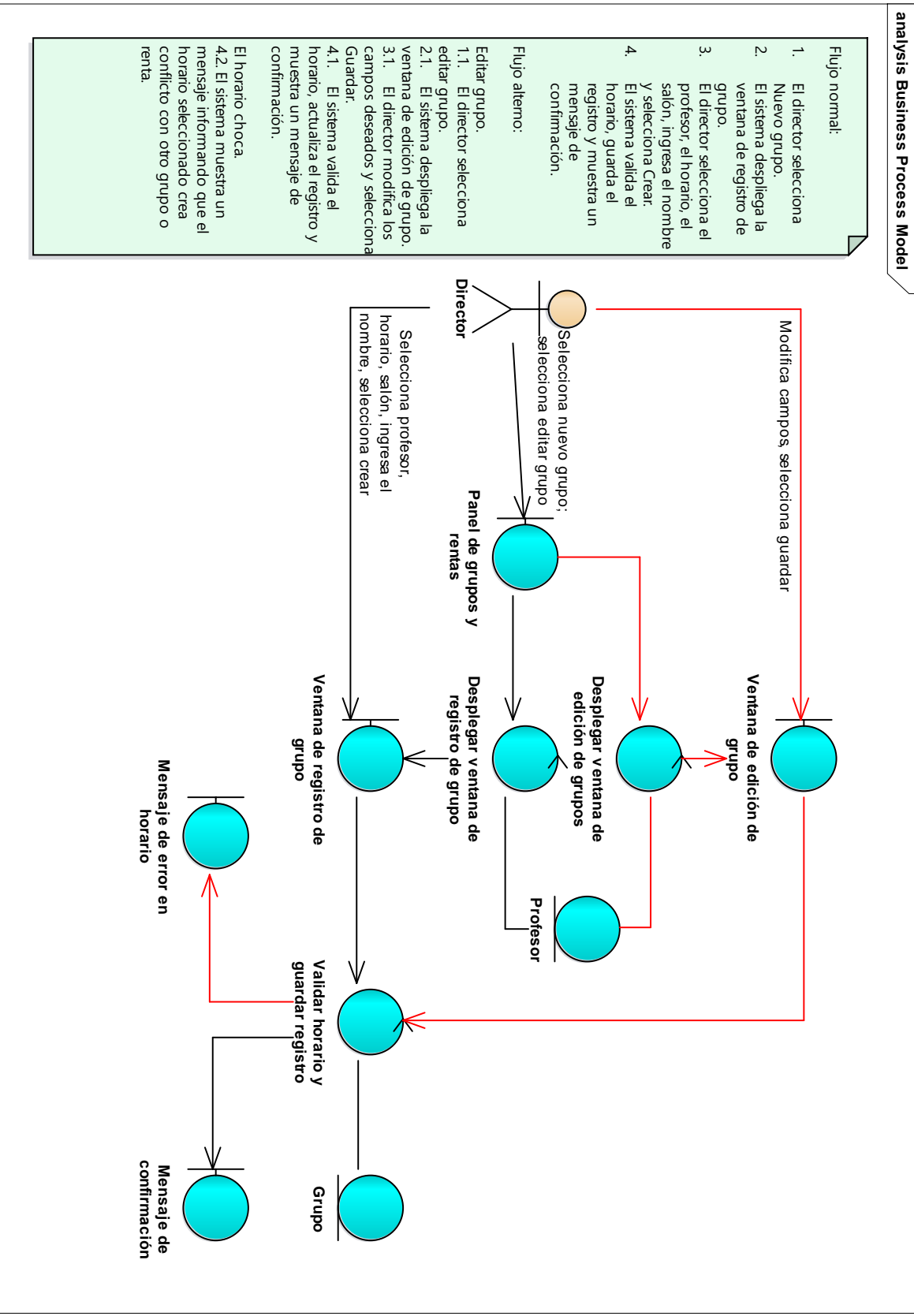
- Consultar alumnos
1. El director selecciona alumnos en la institución.
  2. El sistema busca los alumnos existentes en el sistema y los muestra al director
- Flujo alterno  
No hay alumnos inscritos.
1. El sistema muestra al director que no existen alumnos en la institución.
  2. El sistema regresa a una pantalla anterior del sistema



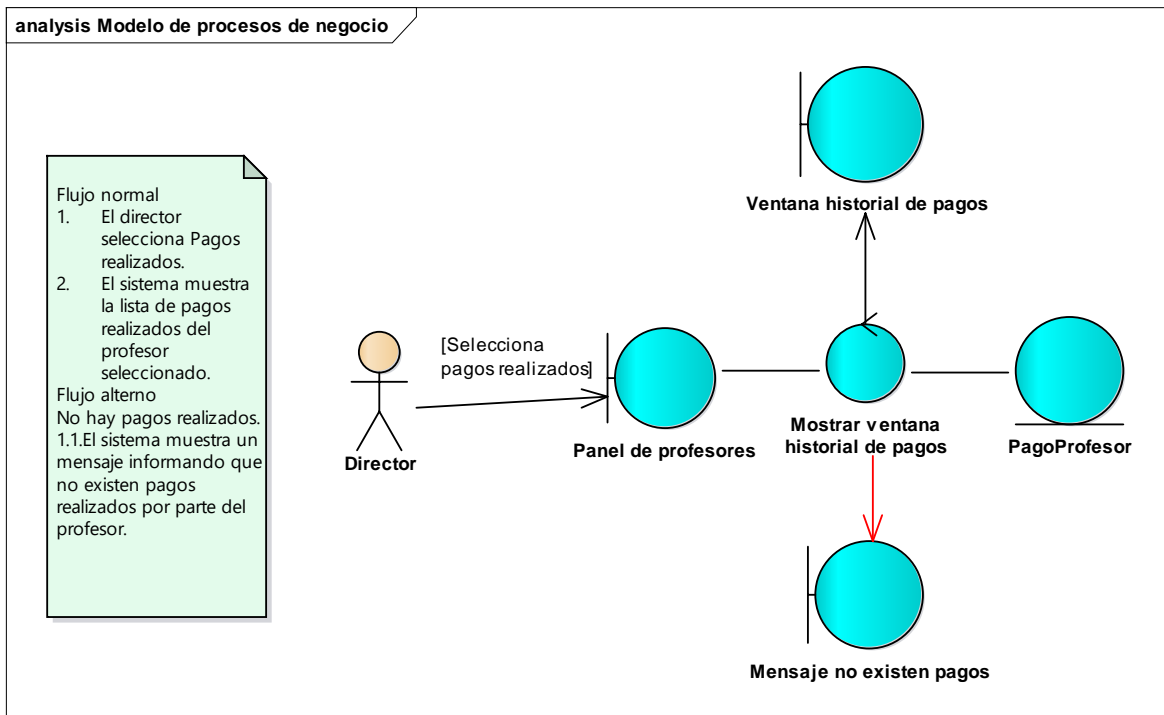
1.3.5. CU 10 – CRU Alumno.



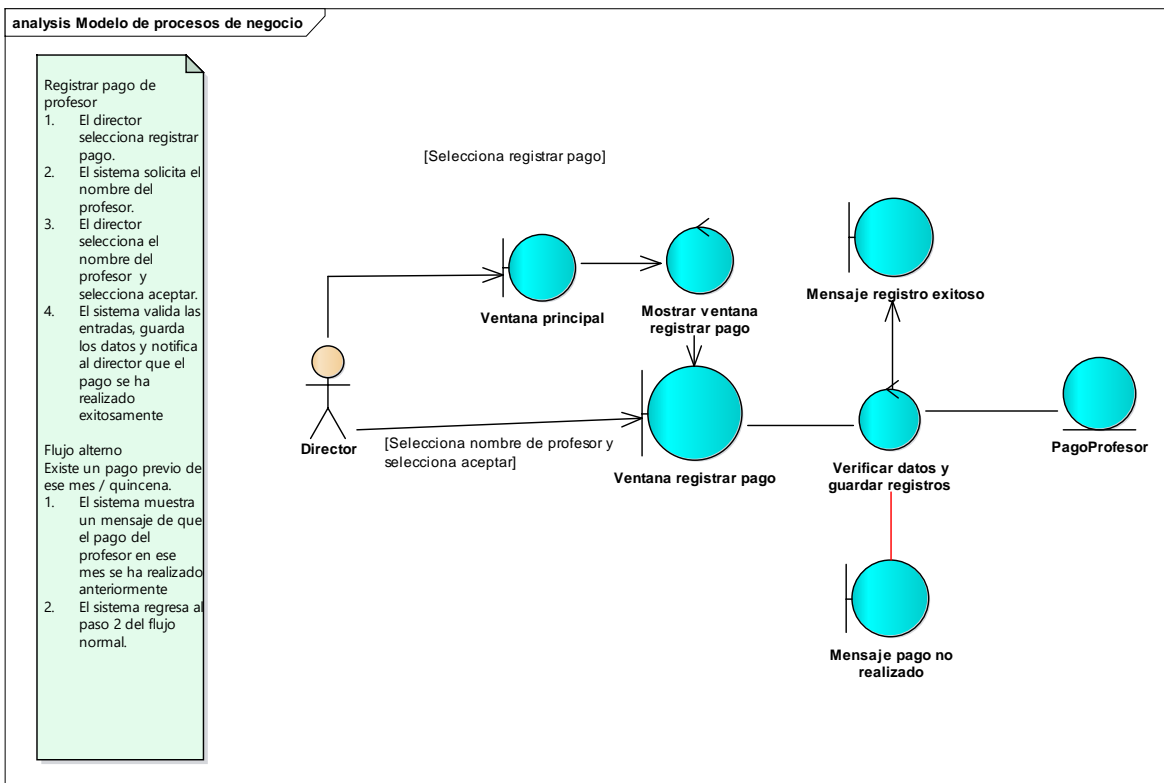
1.3.6. CU 12 – CRU Grupo.



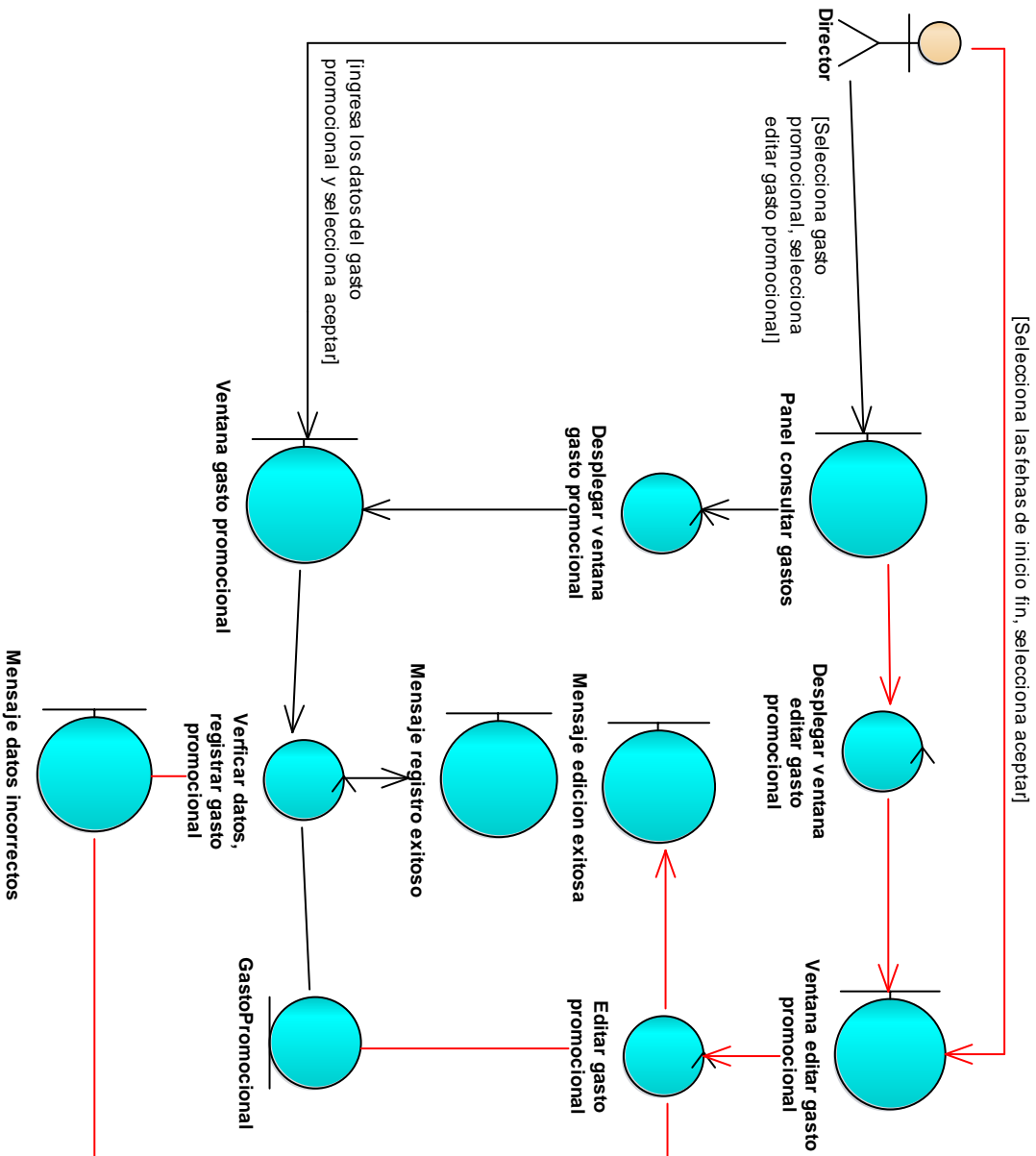
### 1.3.7. CU 13 – Consultar historial de pago de profesores.

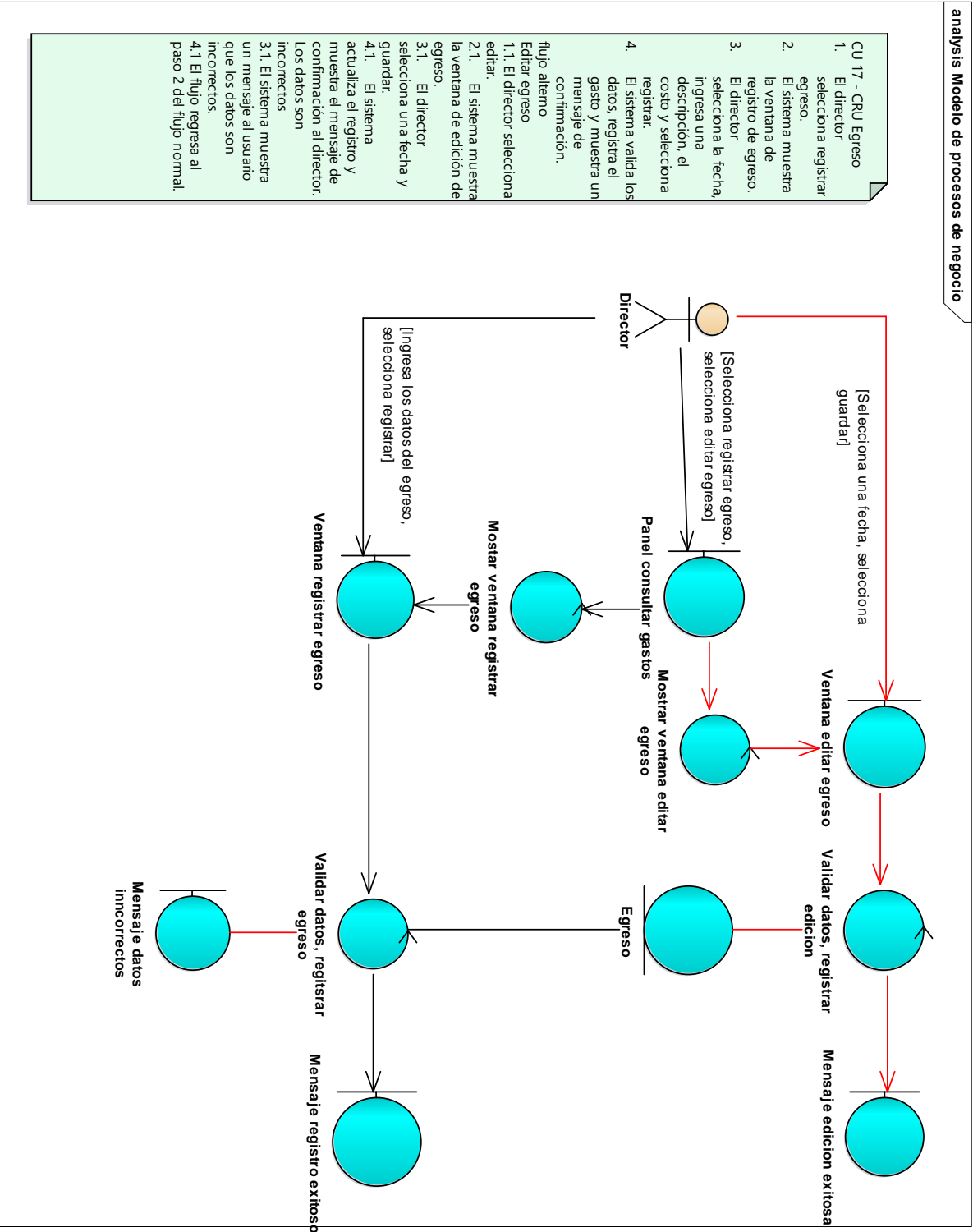


### 1.3.8. CU 14 – Registrar pago de profesor.



- CU 16 – CRU gasto promocional.
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. El sistema valida los datos ingresados al sistema, guarda los datos y notifica al director el registro exitoso
  4. Editar gasto
1. El director selecciona editar
  2. 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.







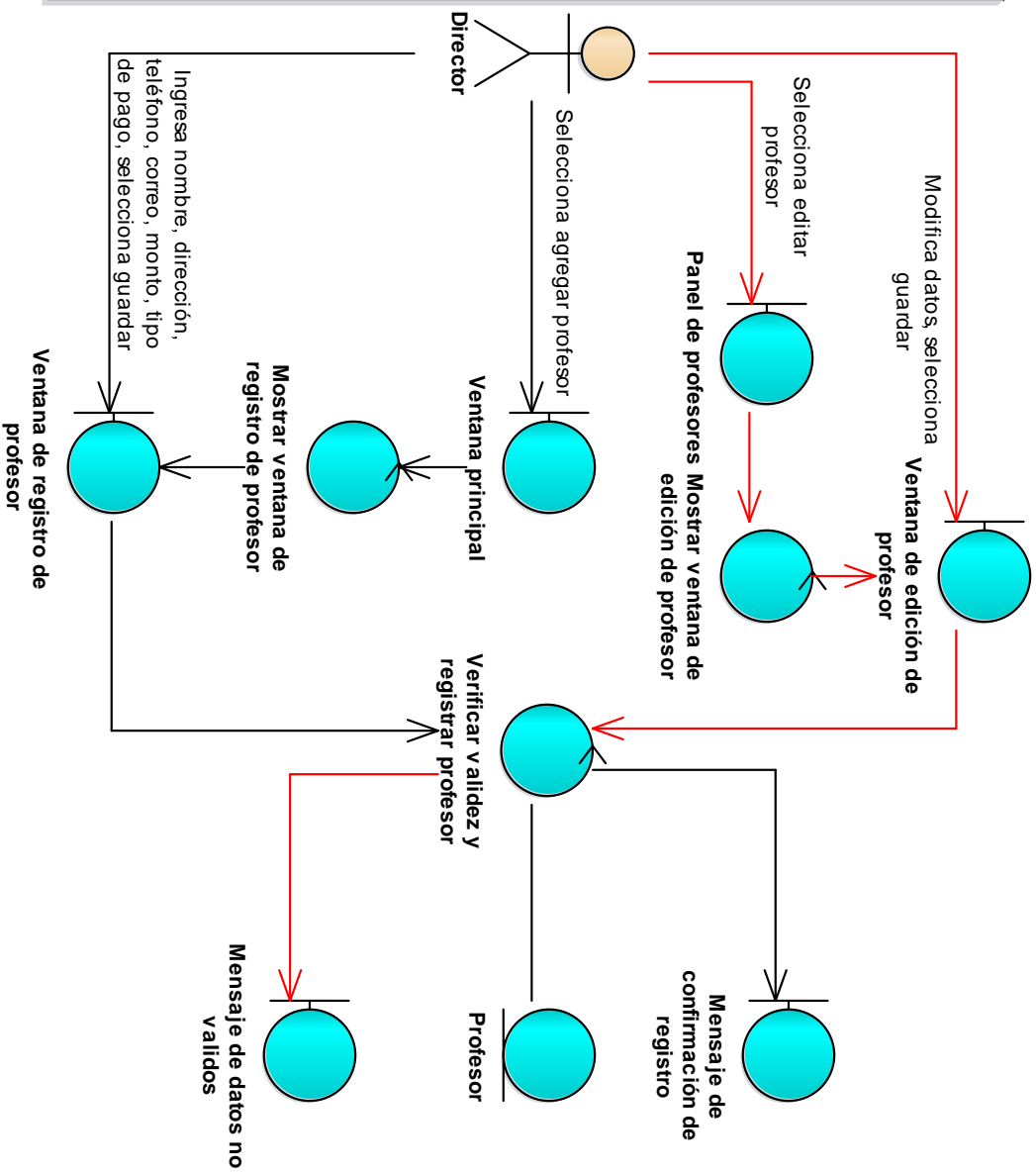
## analysis Business Process Model

Flujo normal:

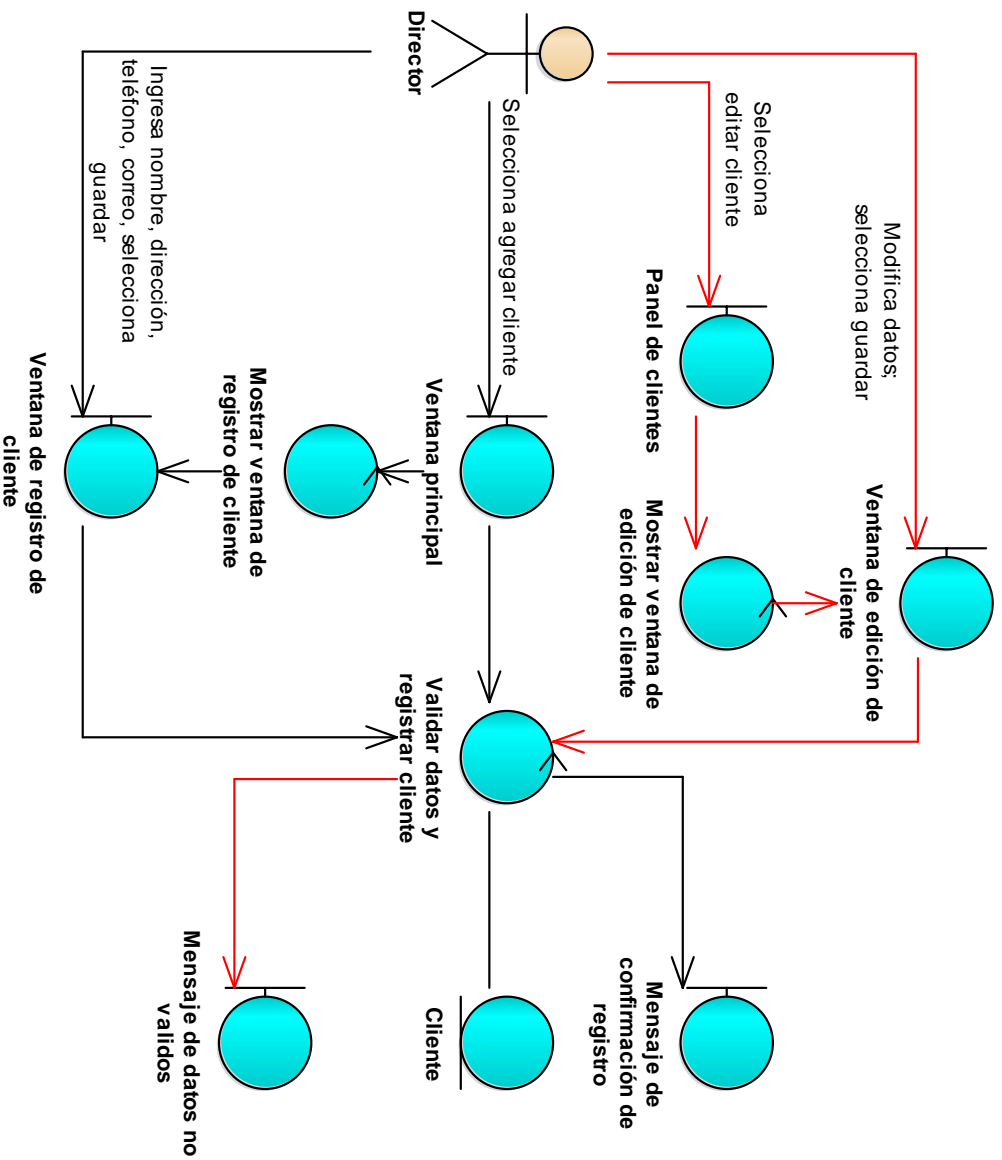
1. El director selecciona agregar profesor.
2. El sistema muestra todos los datos que deben ser cubiertos por el profesor para crear el registro.
3. El director ingresa los datos del nuevo profesor como su nombre, dirección, teléfono, correo, monto, tipo de pago, monto y selecciona guardar.
4. El sistema 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.

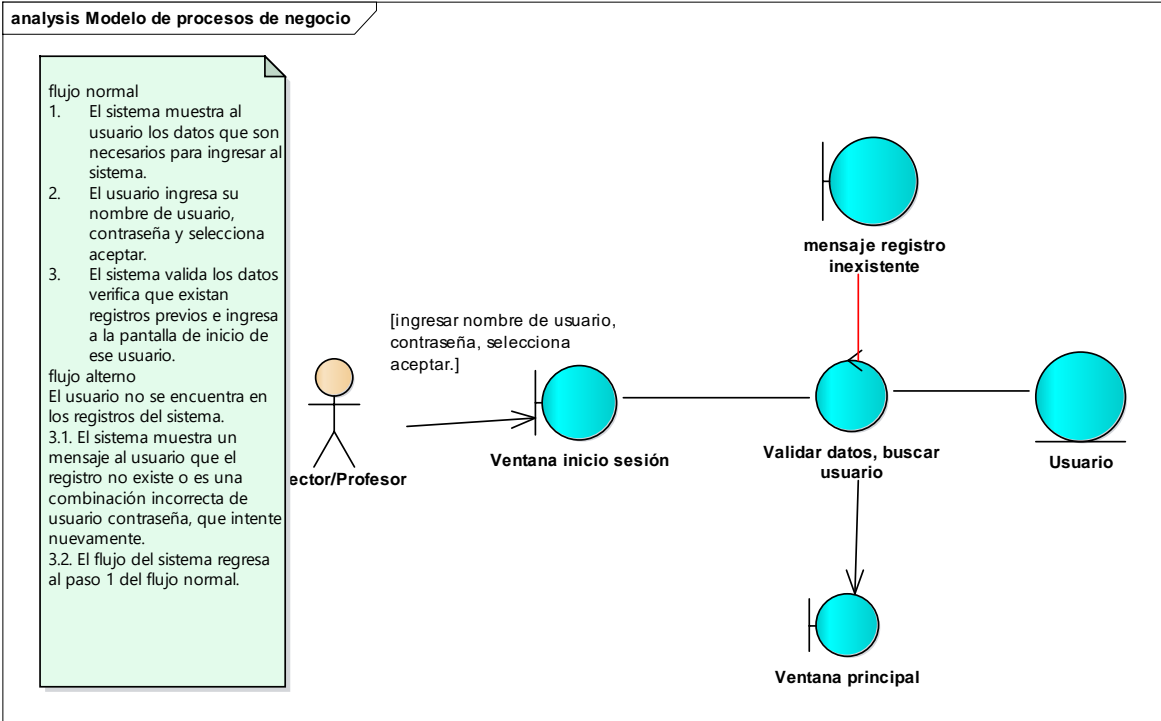
- 1.1. El director selecciona editar profesor.
- 2.1. El sistema muestra los datos del usuario seleccionado.
- 3.1. El director modifica los datos del usuario que se ha seleccionado y que considera pertinentes y selecciona guardar.
- 4.1. El sistema valida los datos, guarda el registro y confirma al director que la operación se realizó con éxito.



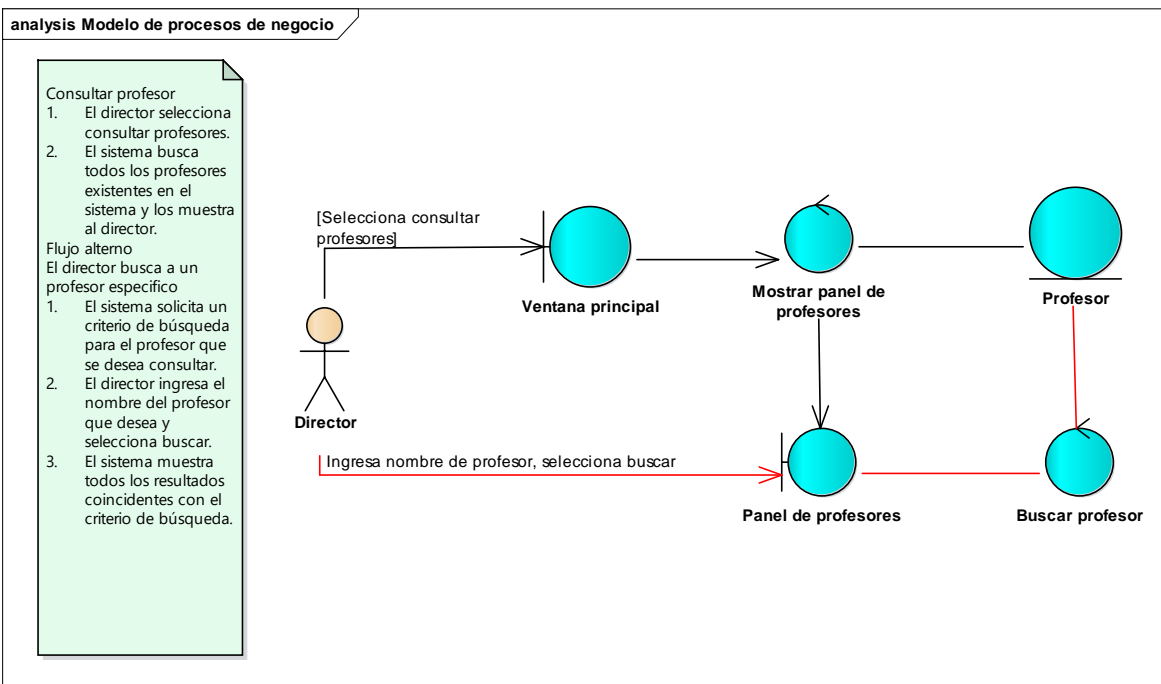
- Flujo normal:**
1. El director selecciona agregar cliente.
  2. El sistema muestra todos los datos que deben ser cubiertos por el director para crear el registro.
  3. El director ingresa los datos del nuevo cliente como su nombre, dirección, teléfono, correo y selección guardar.
  4. El sistema verifica la validez de los datos, crea un registro y notifica al director que el nuevo cliente ha sido creado con éxito.
- Flujo alternativo:  
Editar cliente.**
1. El director selecciona editar cliente.
  2. El sistema muestra los datos del cliente seleccionado.
  1. El director modifica los datos del cliente que se ha seleccionado y que considera pertinentes y selección guardar.
  4. 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.



### 1.3.13. CU 22 - Iniciar sesión.



### 1.3.14. CU 23 – Consultar profesores.



### 1.3.15. CU 24 – Consultar clientes.

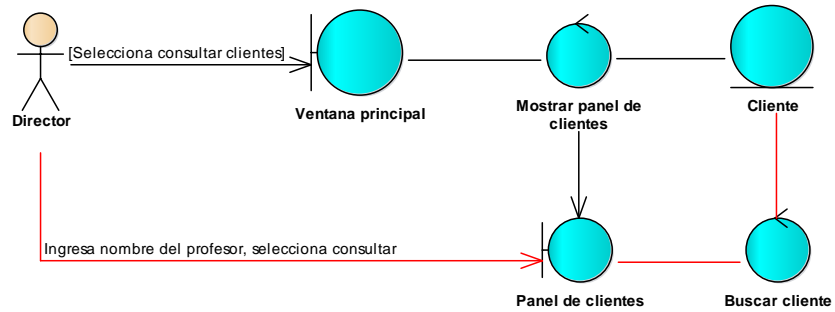
#### analysis Modelo de procesos de negocio

Consultar clientes

1. El director selecciona consultar clientes.
2. El sistema busca todos los clientes existentes en el sistema y los muestra al director

El director busca a un cliente específico

1. El sistema solicita un criterio de búsqueda para el cliente que se desea consultar.
2. El director ingresa el nombre del cliente que desea y selecciona aceptar.
3. El sistema muestra todos los resultados coincidentes con el criterio de búsqueda.



### 1.3.16. CU 25 – Consultar gastos.

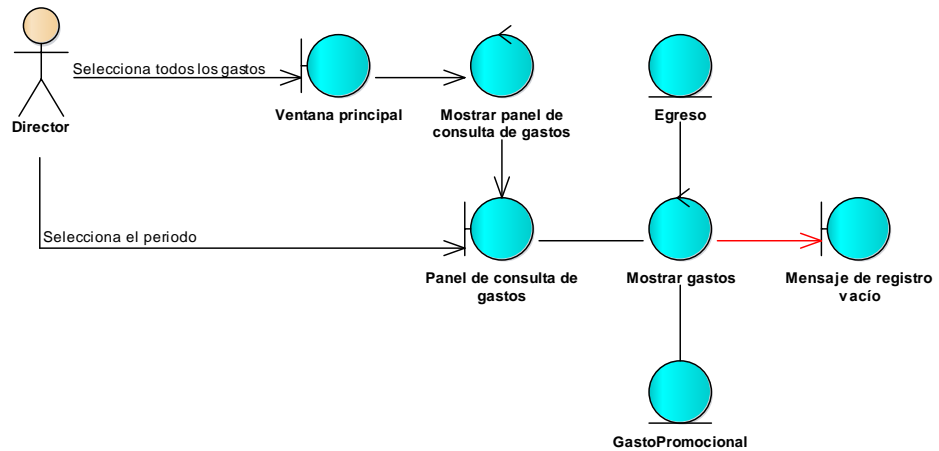
#### analysis Business Process Model

Flujo normal:

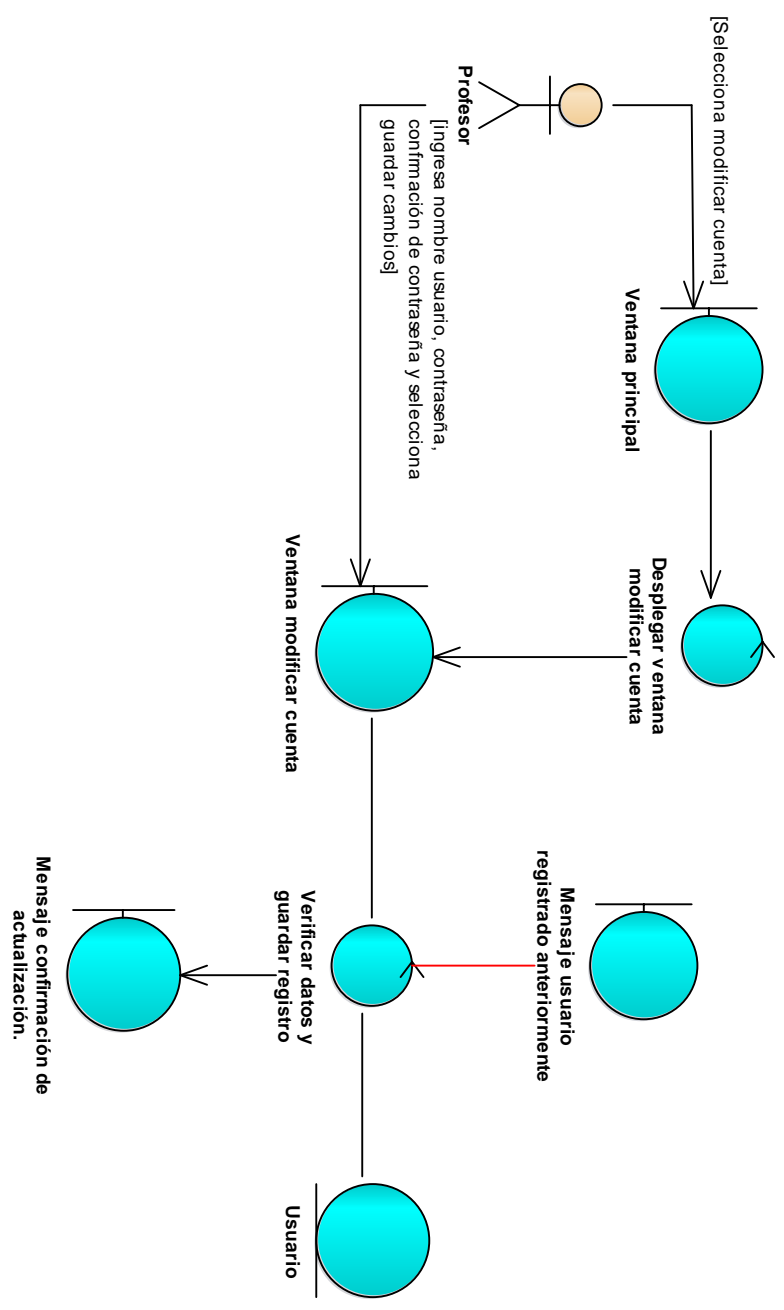
1. El director selecciona todos los gastos
2. El sistema muestra la selección del periodo de consulta.
3. El director selecciona el periodo.
4. El sistema muestra dos listas, una con todas las promociones de Facebook y otra con todos los egresos del periodo seleccionado.

Flujo alterno:

No hay gastos registrados.  
2.1. El sistema muestra un mensaje informando que no existen gastos registrados.



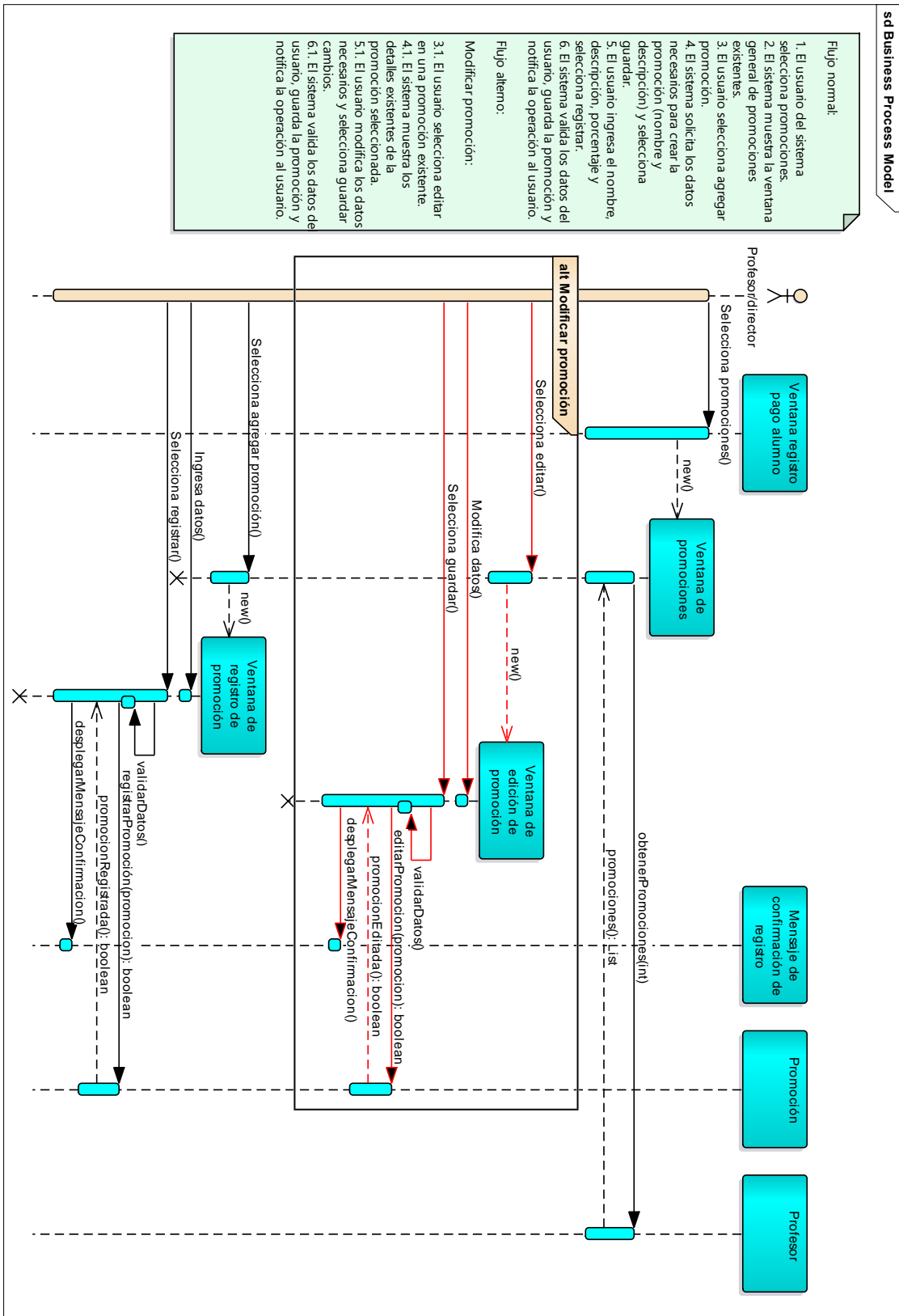
- CU 23 – Modificar cuenta de usuario.
1. El profesor selecciona modificar cuenta.
  2. El sistema muestra la ventana de modificar cuenta.
  3. El profesor ingresa un nuevo nombre de usuario, contraseña, la confirmación de esta y selecciona guardar cambios.
  4. El sistema verifica, guarda el registro y notifica al profesor que sus datos fueron actualizados correctamente.
- 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.



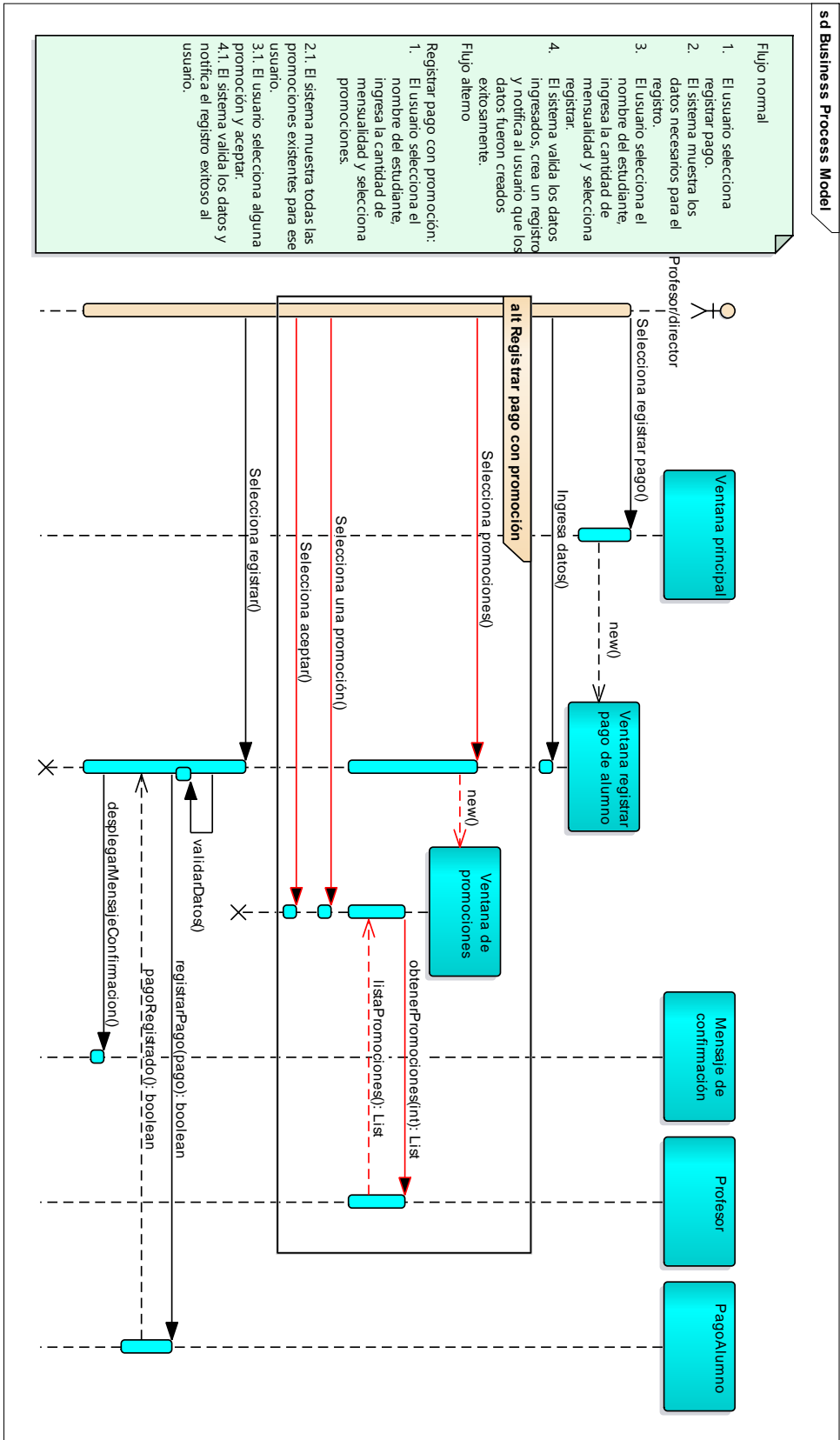
1.3.17. CU 26 – Modificar cuenta de usuario.

## 1.4. Diagramas de secuencia.

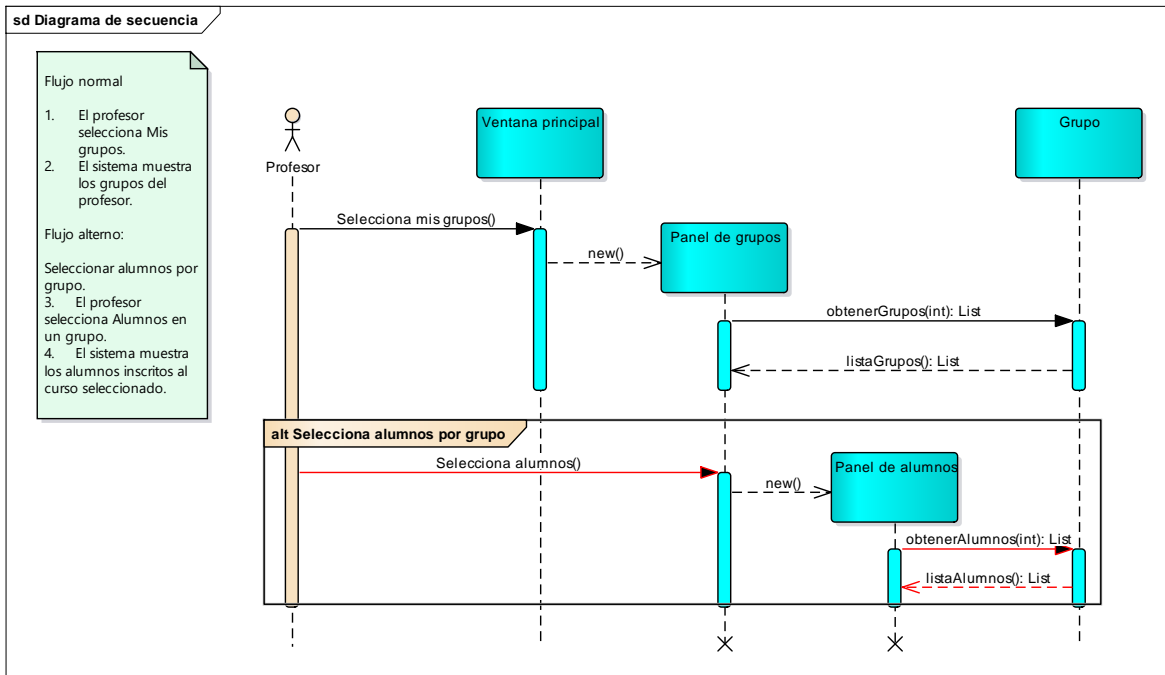
### 1.4.1. CU 01 – CRU Promoción.



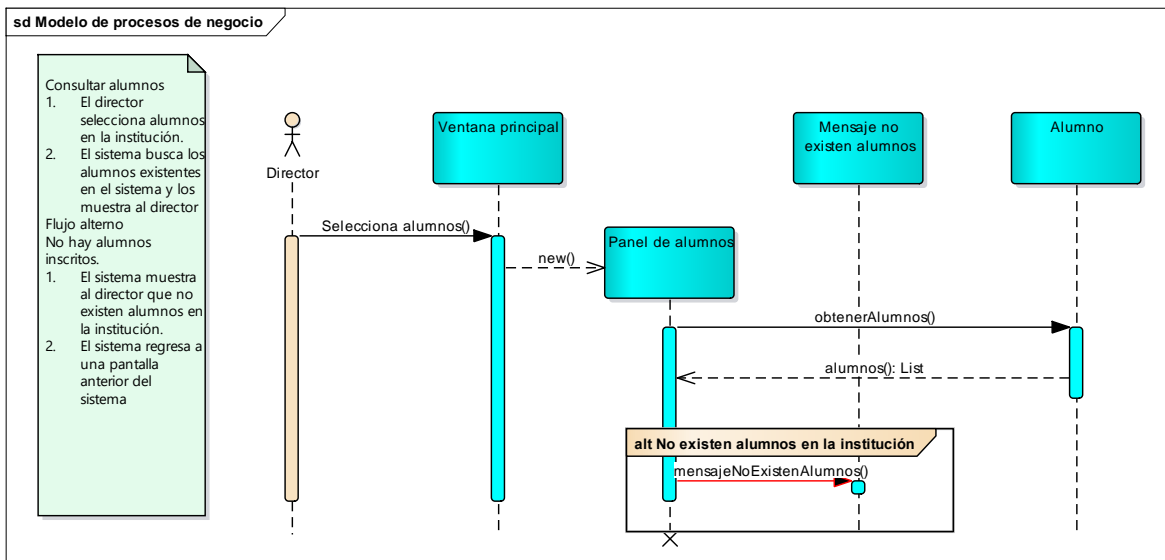
#### 1.4.2. CU 02 – Registrar pago de alumno.



### 1.4.3. CU 05 – Consultar grupos.

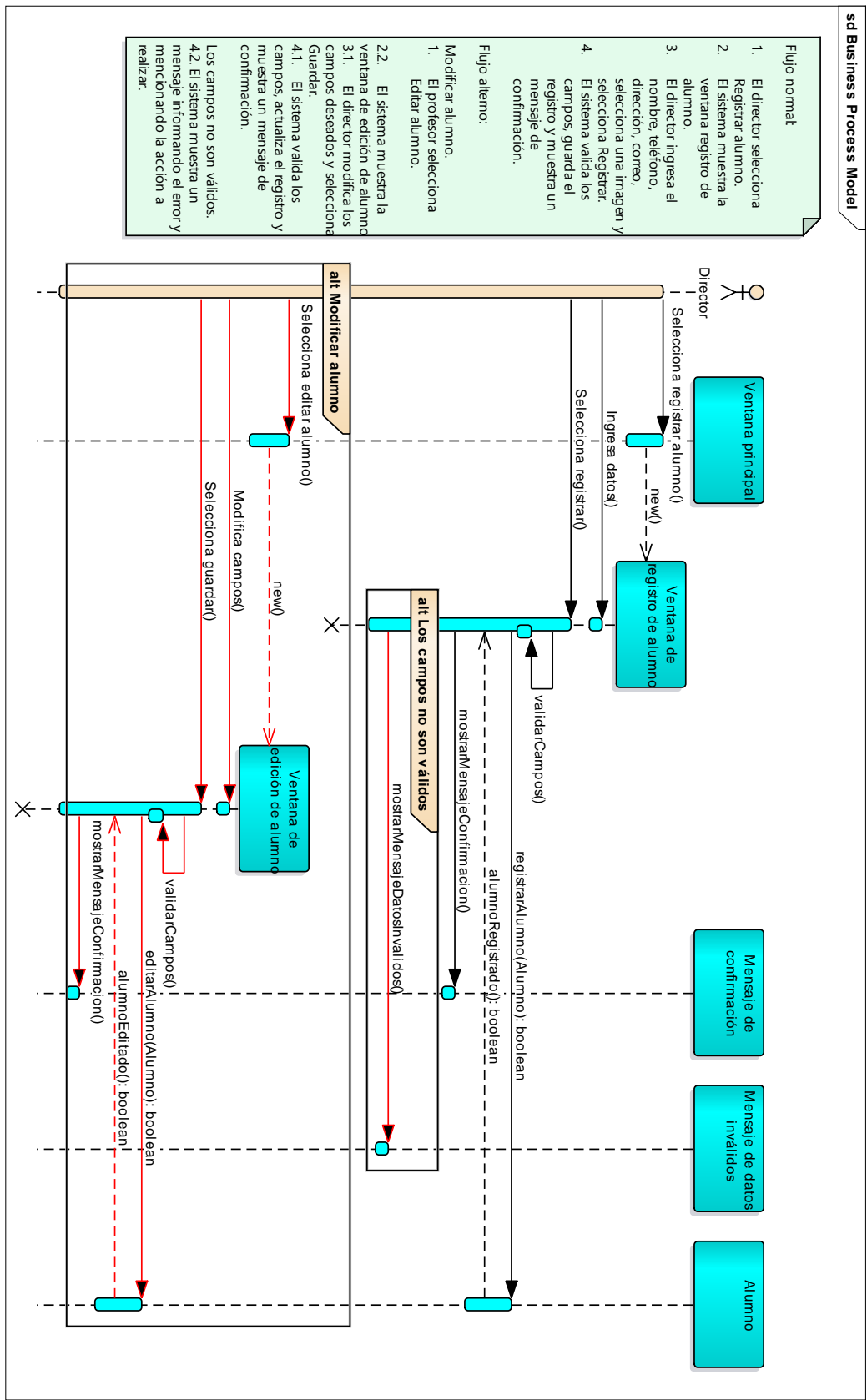


### 1.4.4. CU 09 – Consultar alumnos.

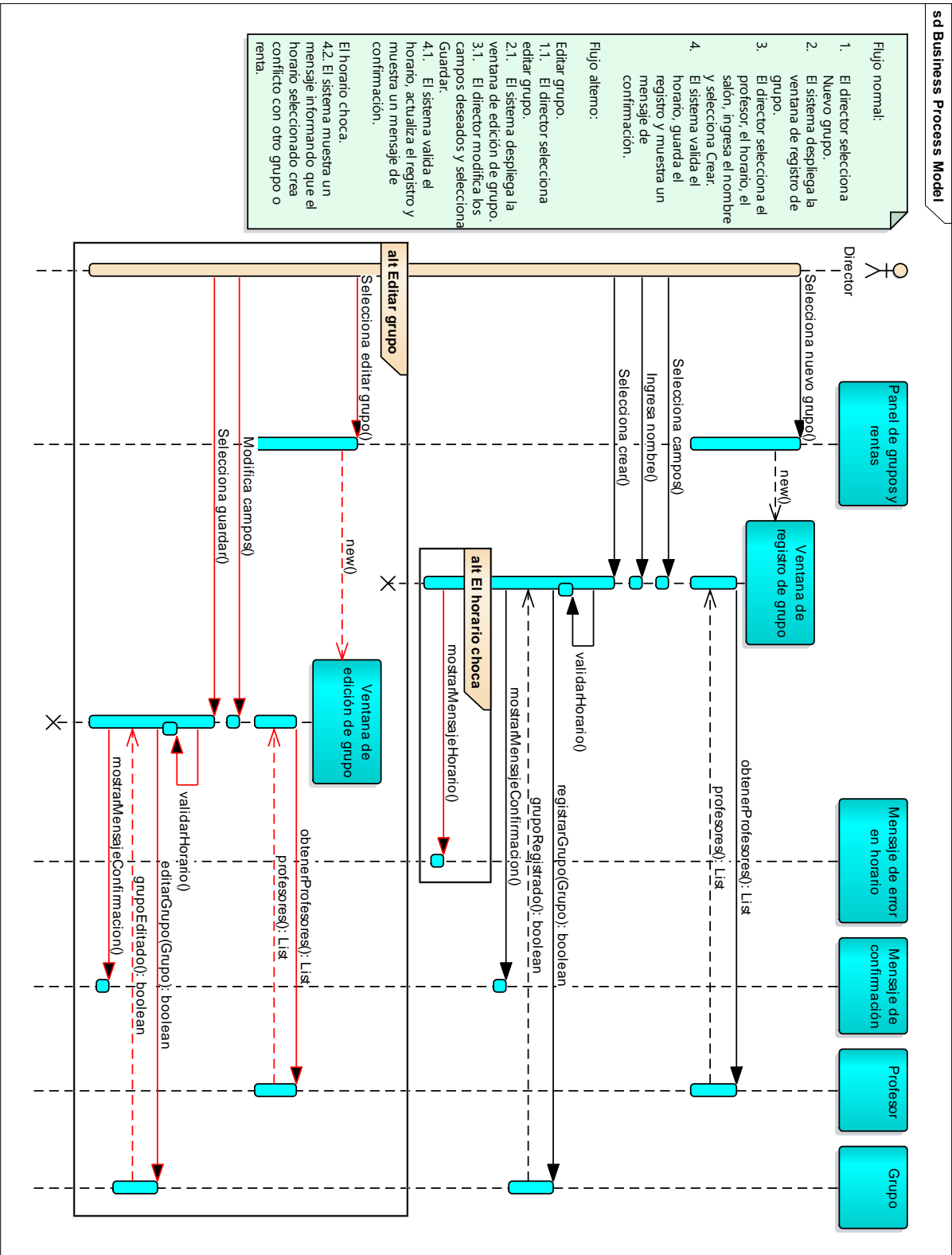




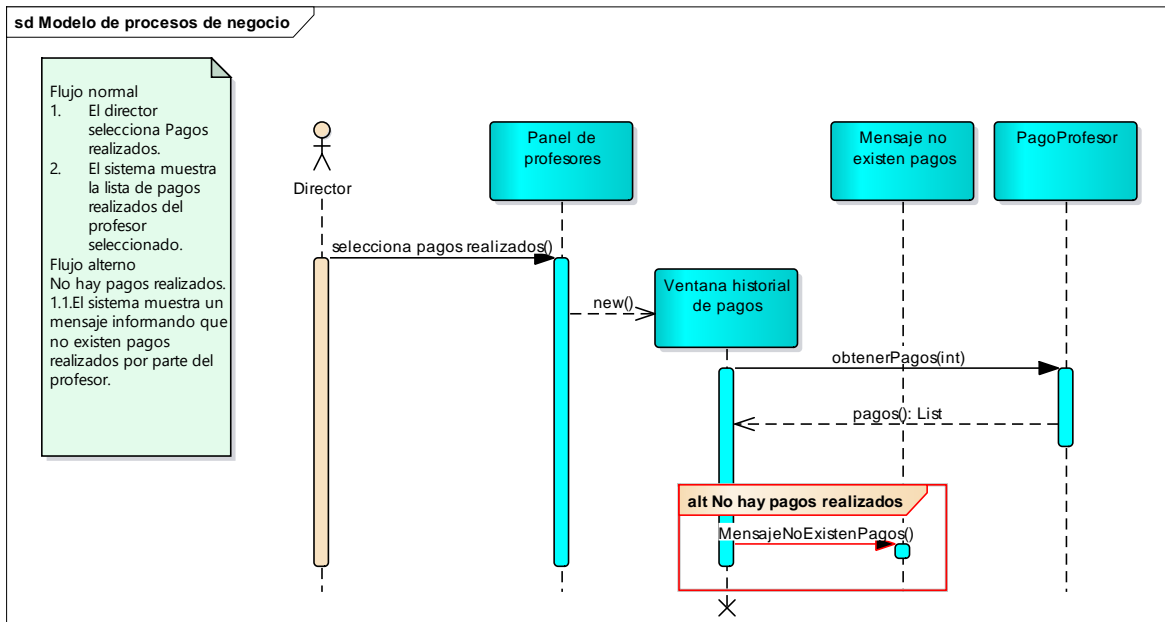
1.4.5. CU 10 – CRU Alumno.



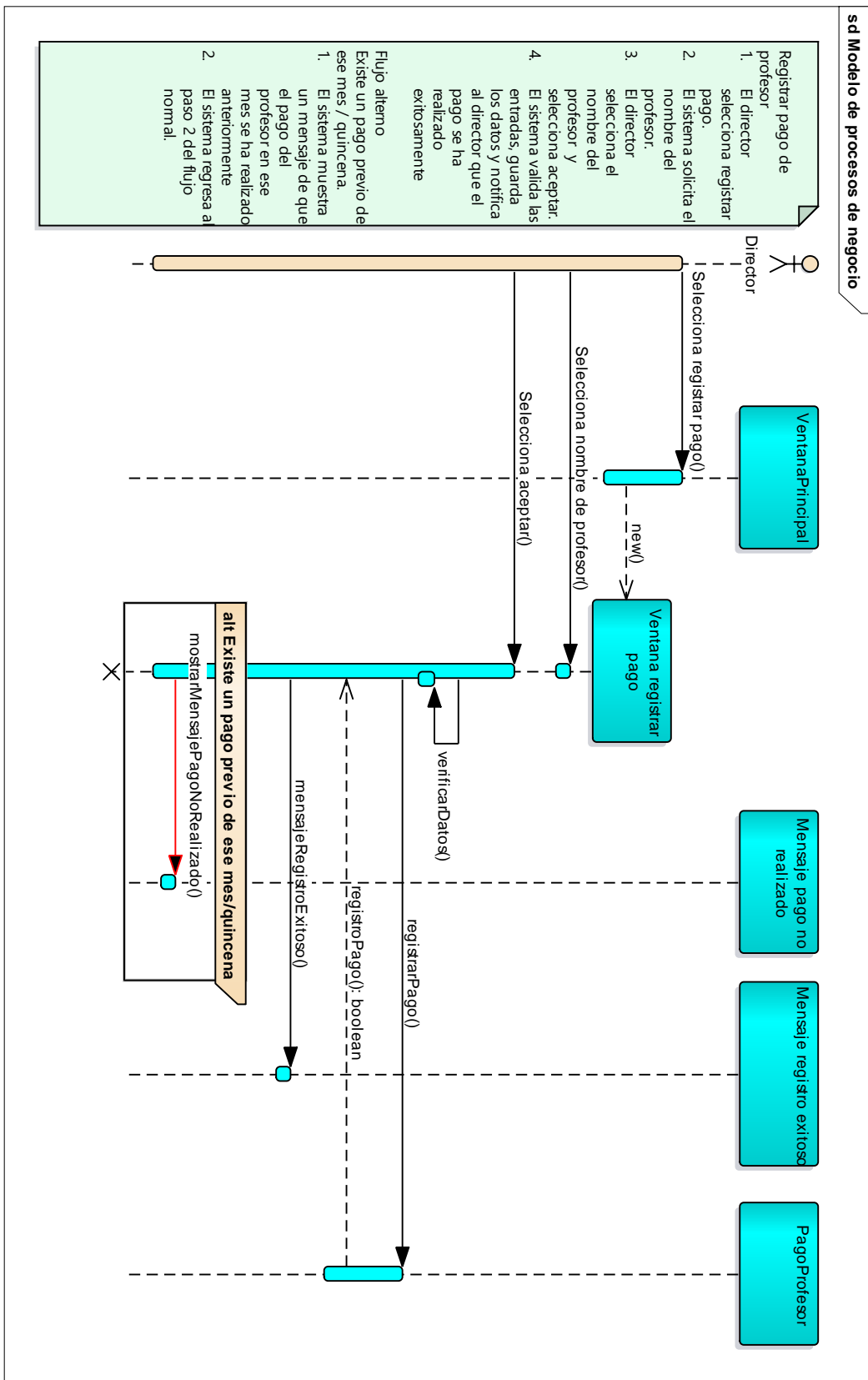
#### 1.4.6. CU 12 – CRU Grupo.



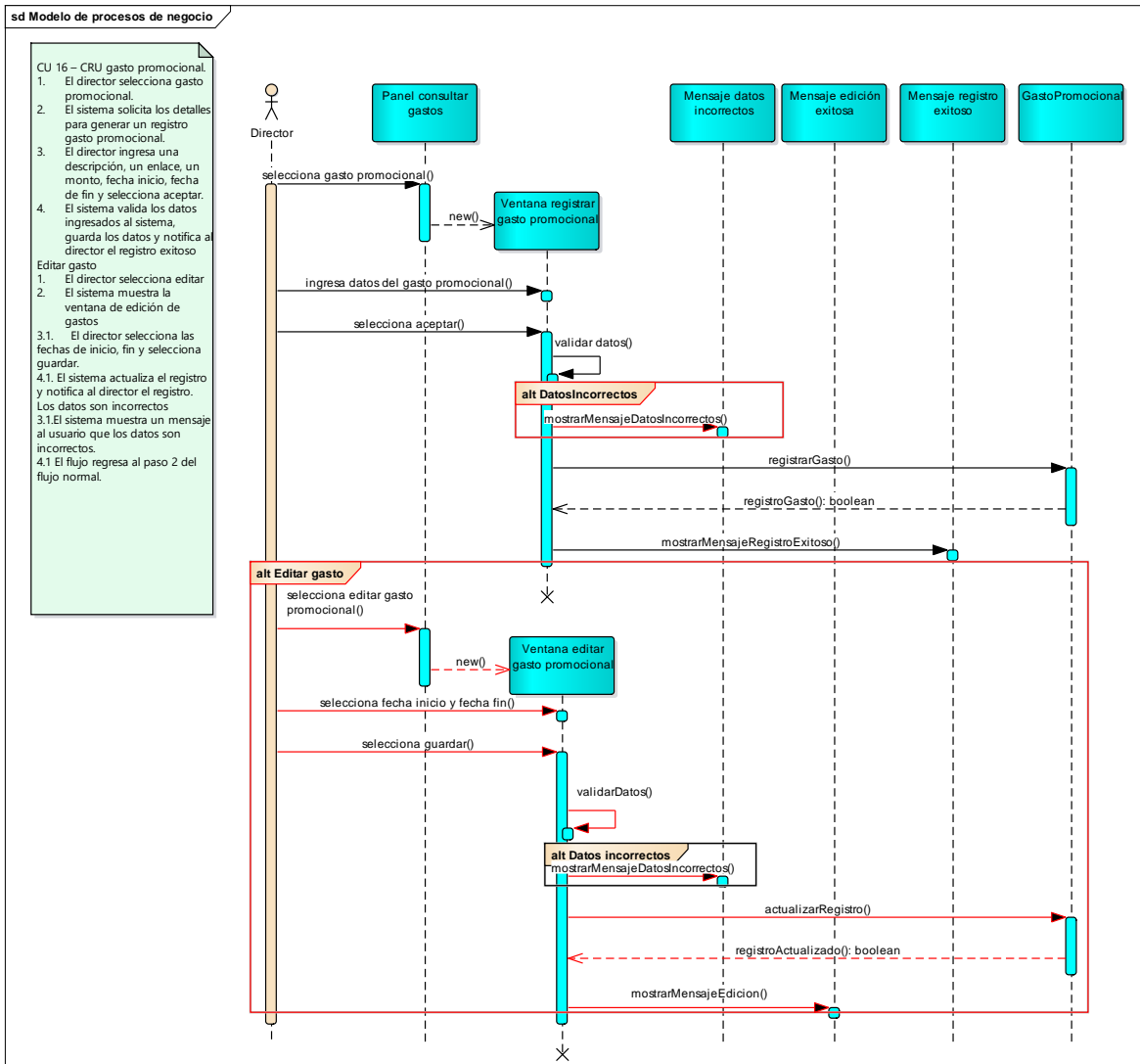
#### 1.4.7. CU 13 – Consultar historial de pago de profesores.



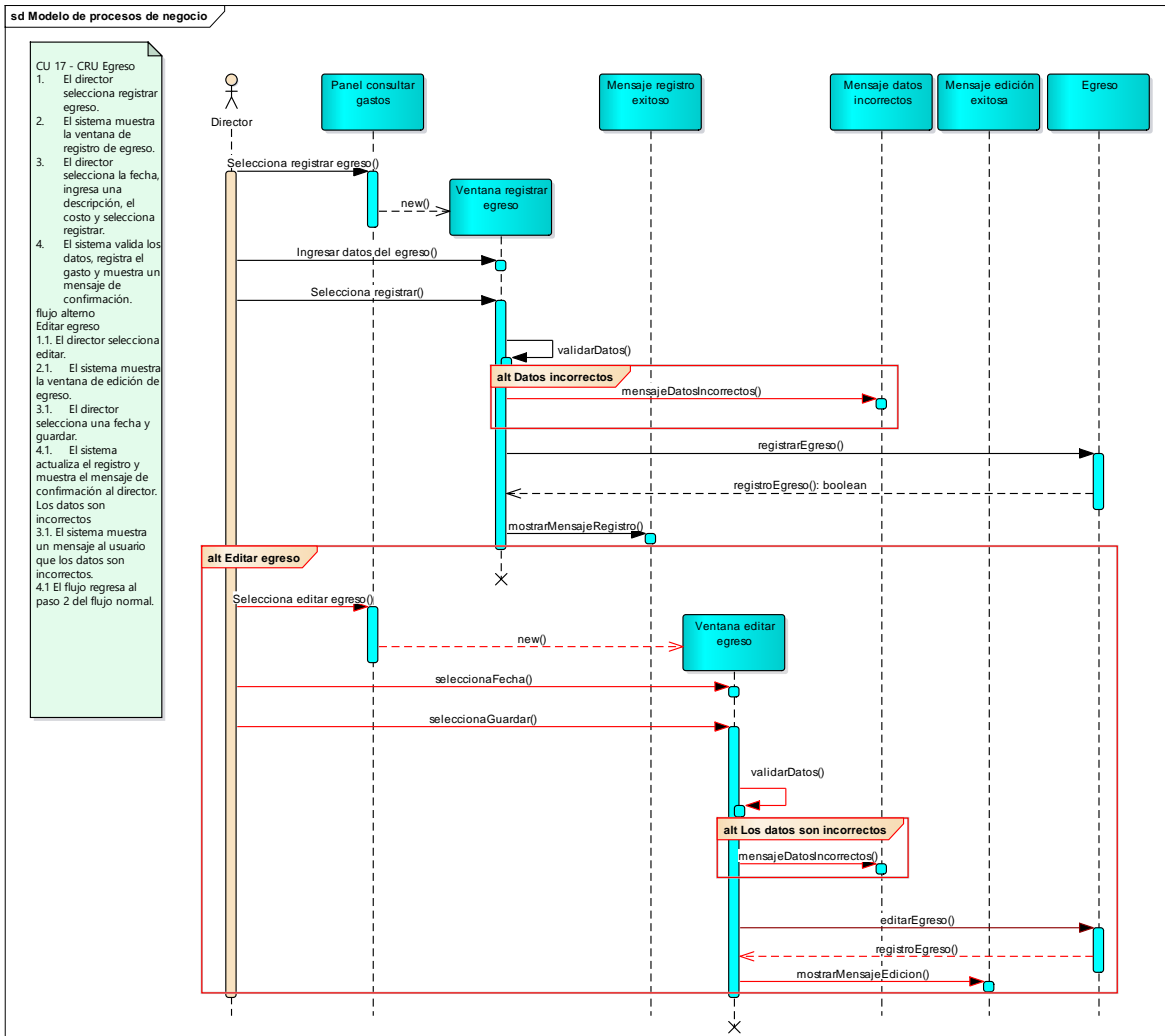
1.4.8. CU 14 – Registrar pago de profesor.



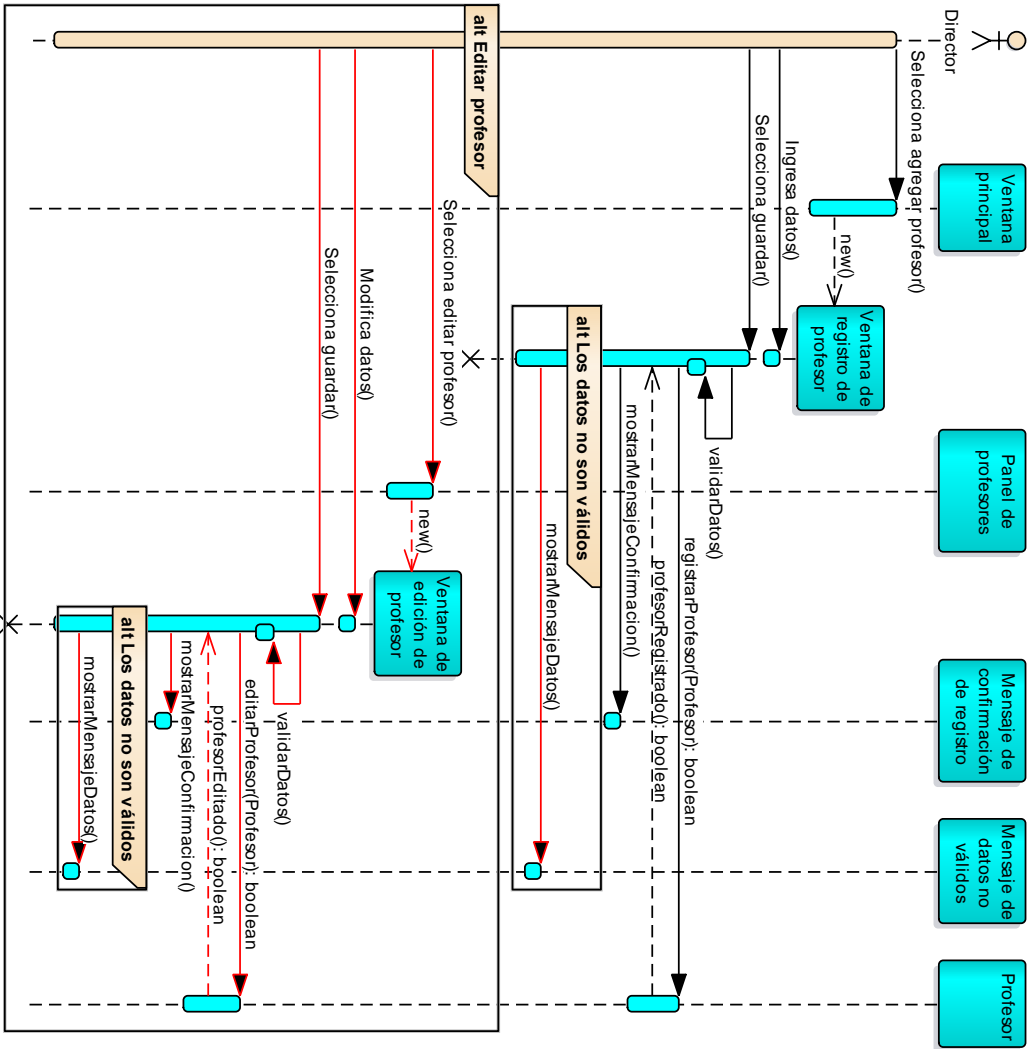
### 1.4.9. CU 16 – CRU Gasto promocional.



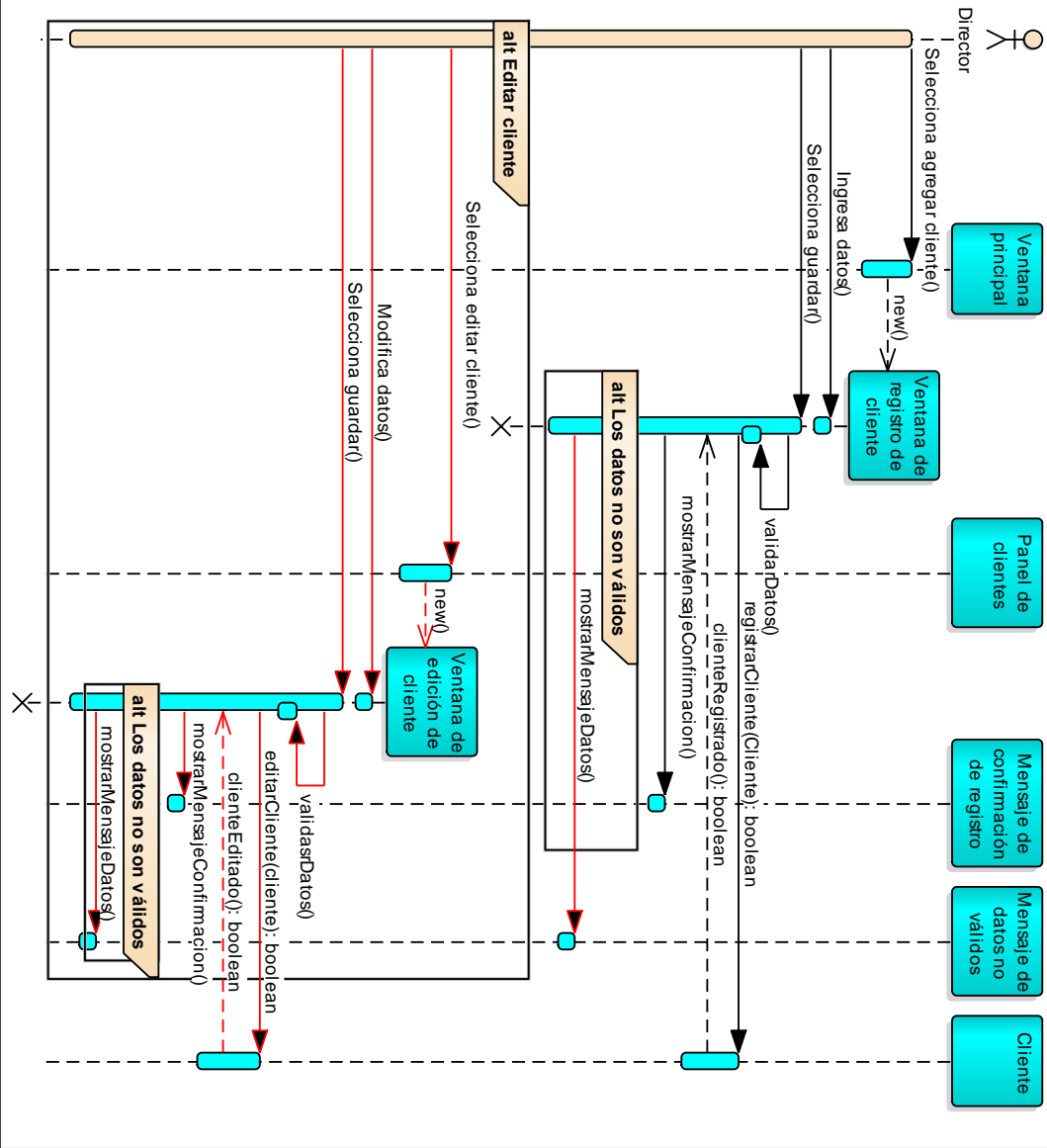
## 1.4.10. CU 17 – CRU Egreso.



- 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.
1. El sistema muestra un mensaje indicando el error y como proceder.

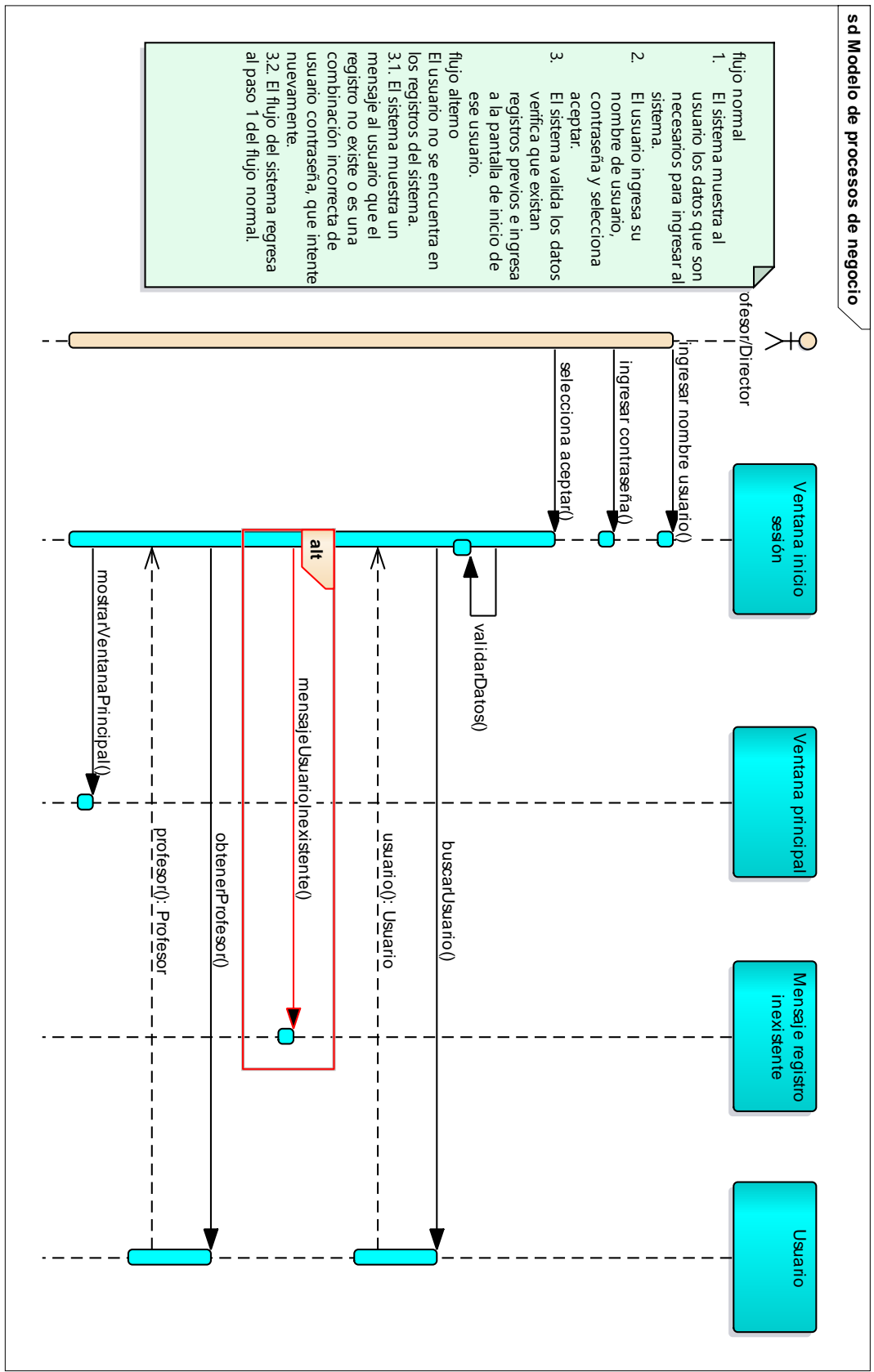


- Flujo normal:
1. El director selecciona agregar cliente.
  2. El sistema muestra todos los datos que deben ser cubiertos por el director para crear el registro.
  3. El director ingresa los datos del nuevo cliente como su nombre, dirección, teléfono, correo y selecciona guardar.
  4. El sistema verifica 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. El director selecciona editar cliente.
  - 2.1. El sistema muestra los datos del cliente seleccionado.
  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.

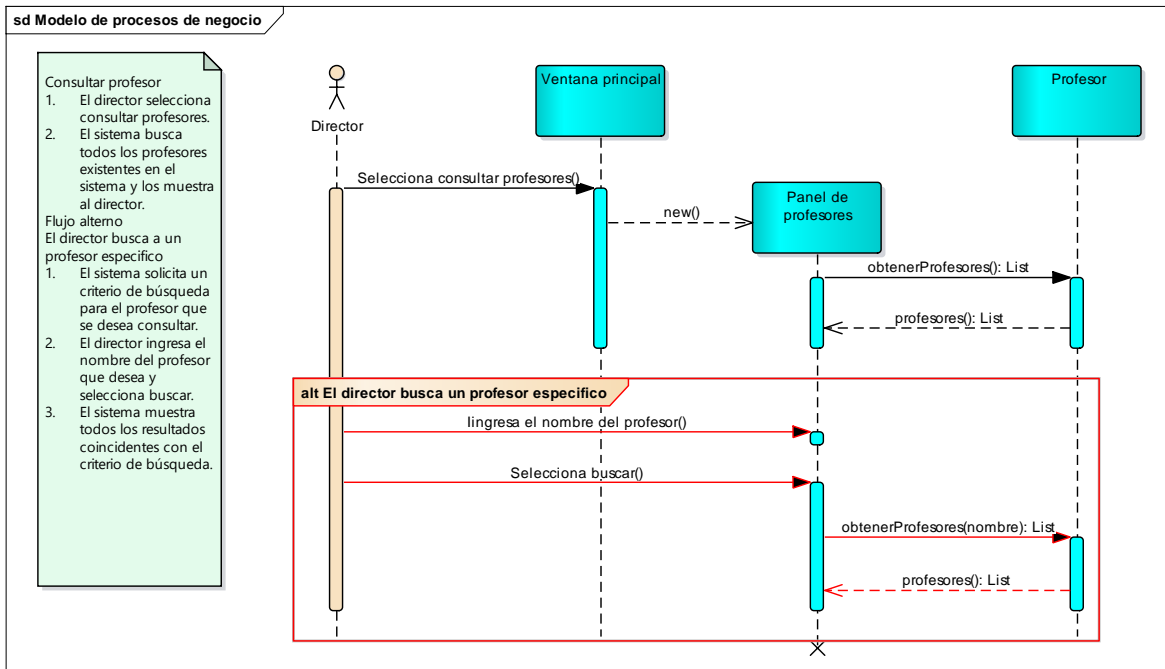




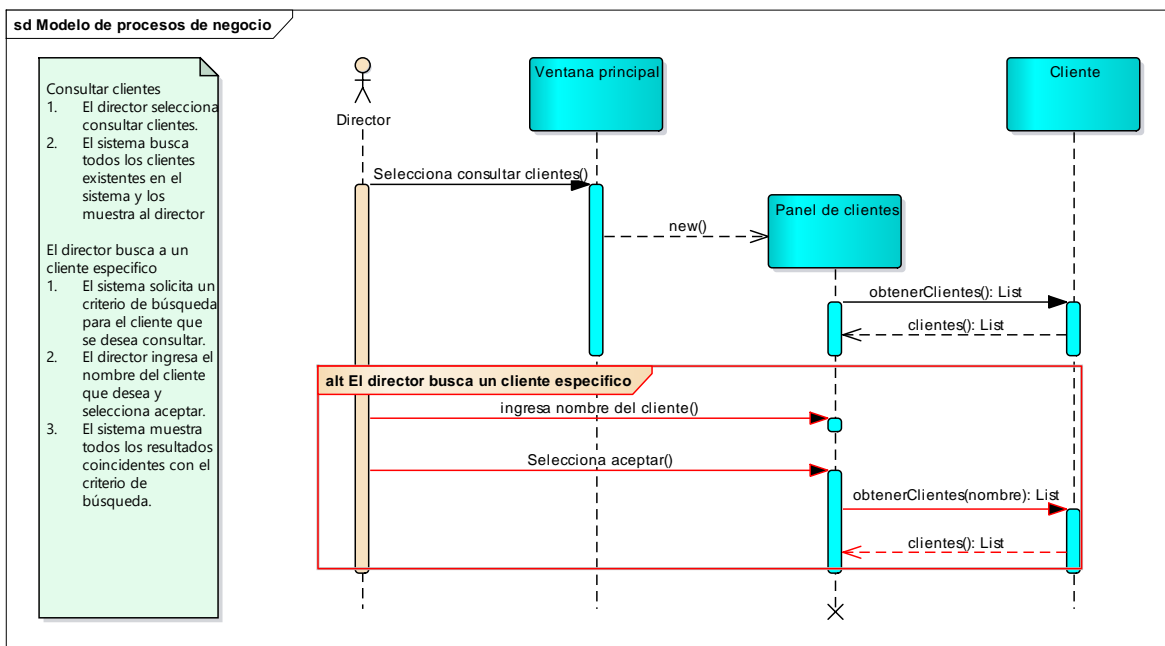
1.4.13. CU 22 – Iniciar sesión.



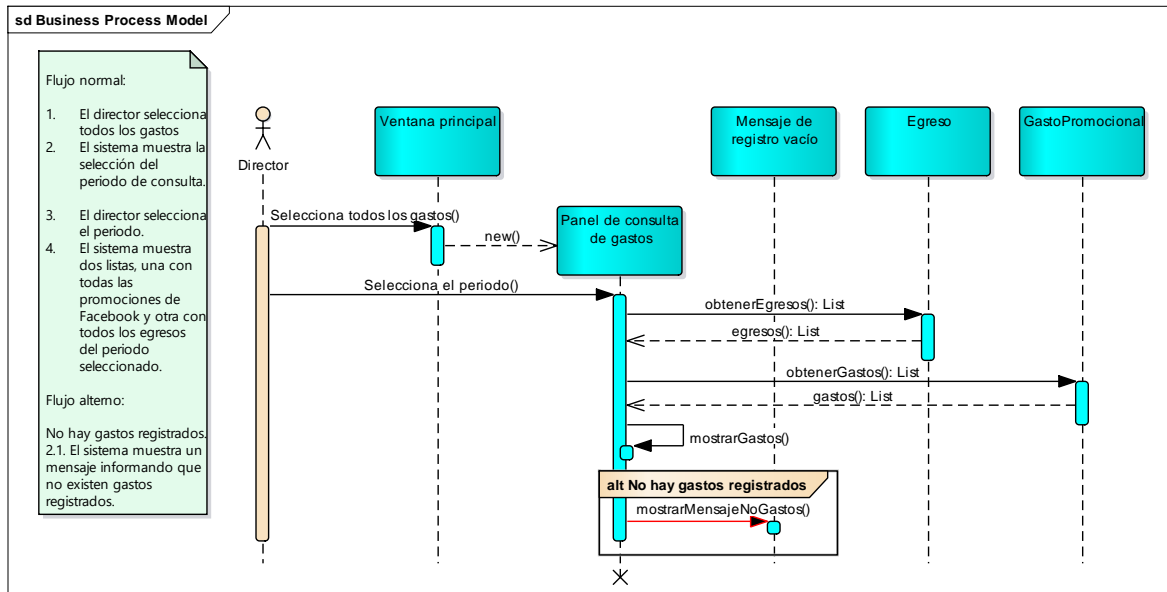
#### 1.4.14. CU 23 – Consultar profesores.



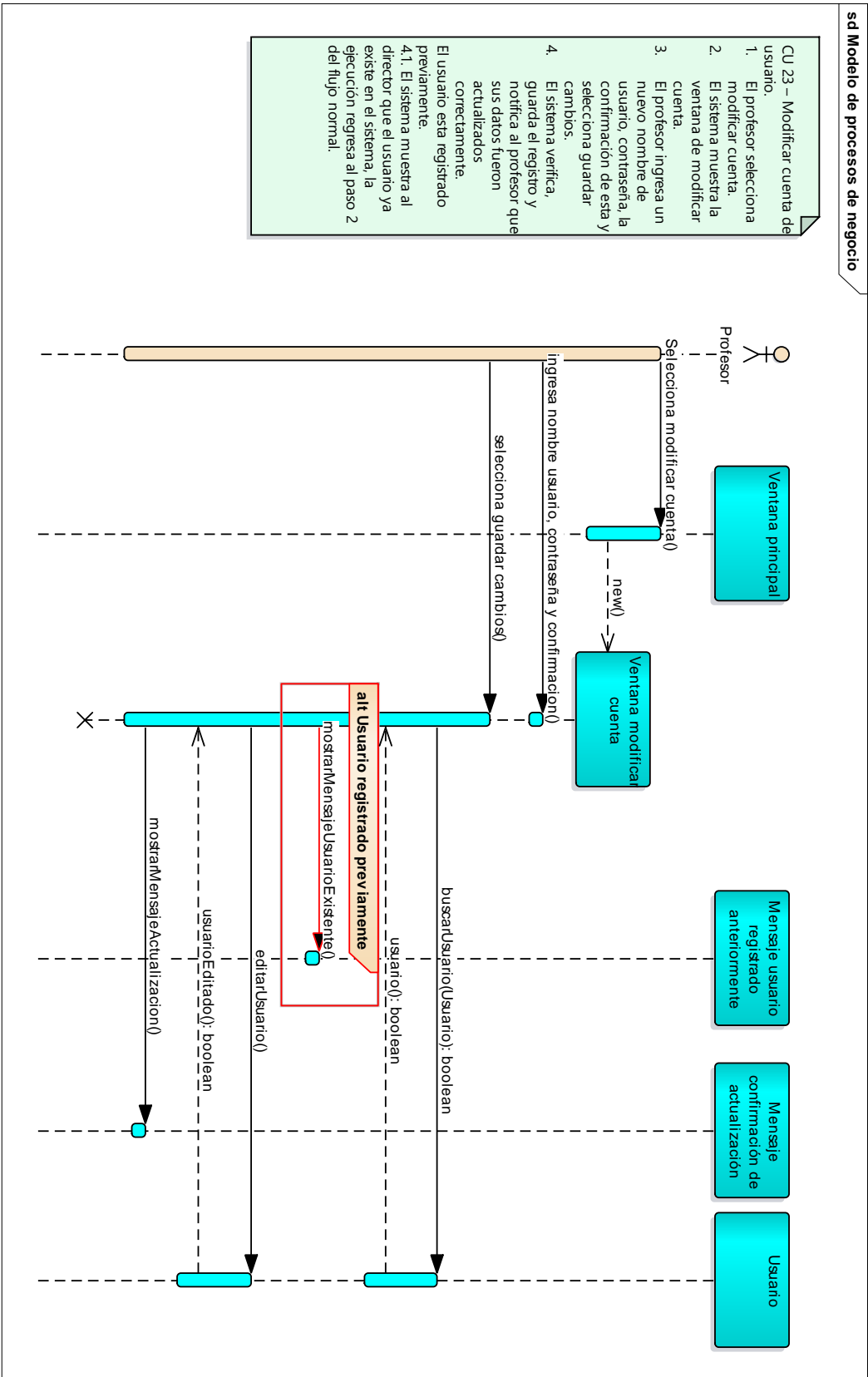
#### 1.4.15. CU 24 – Consultar clientes.



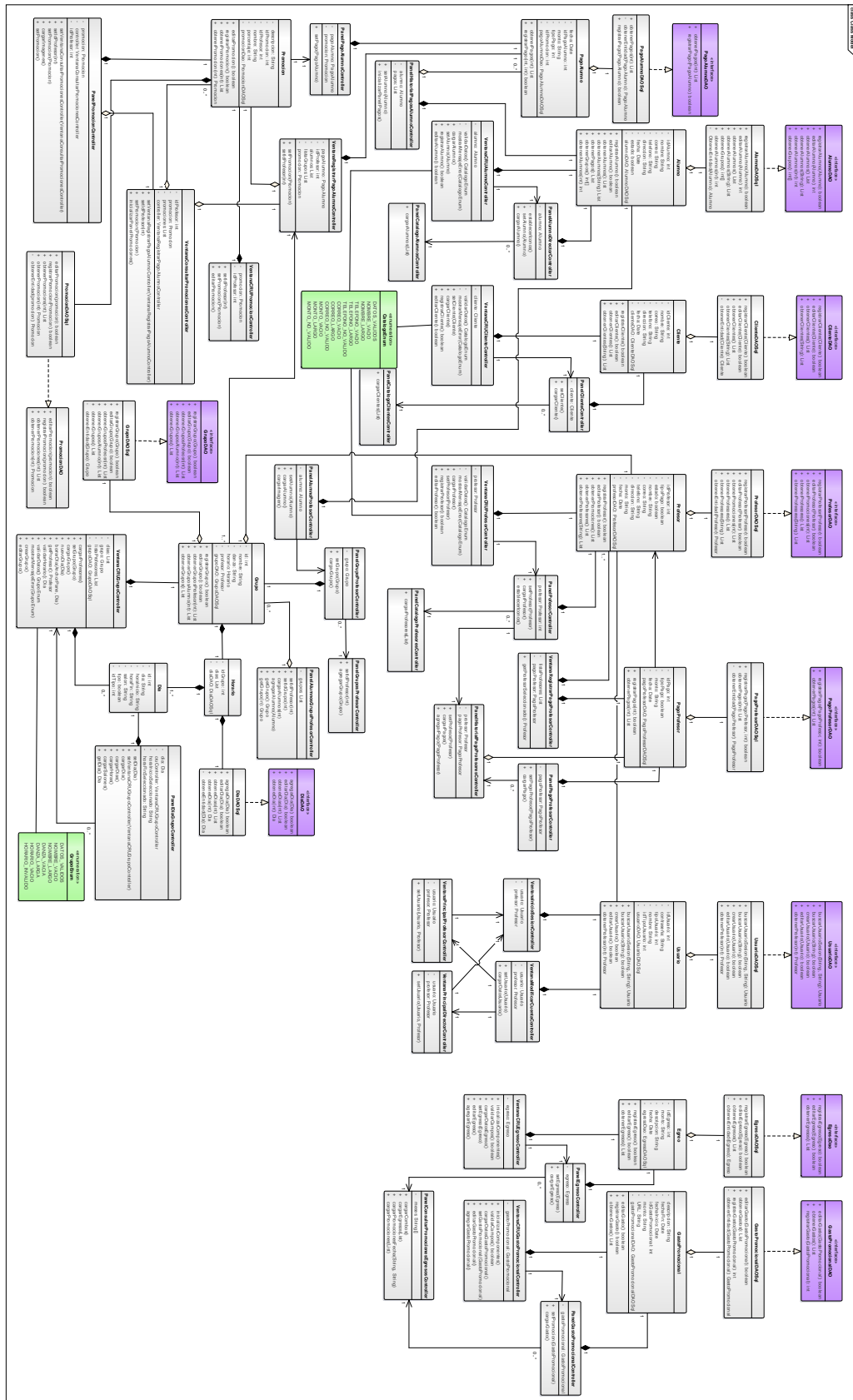
#### 1.4.16. CU 25 – Consultar gastos.



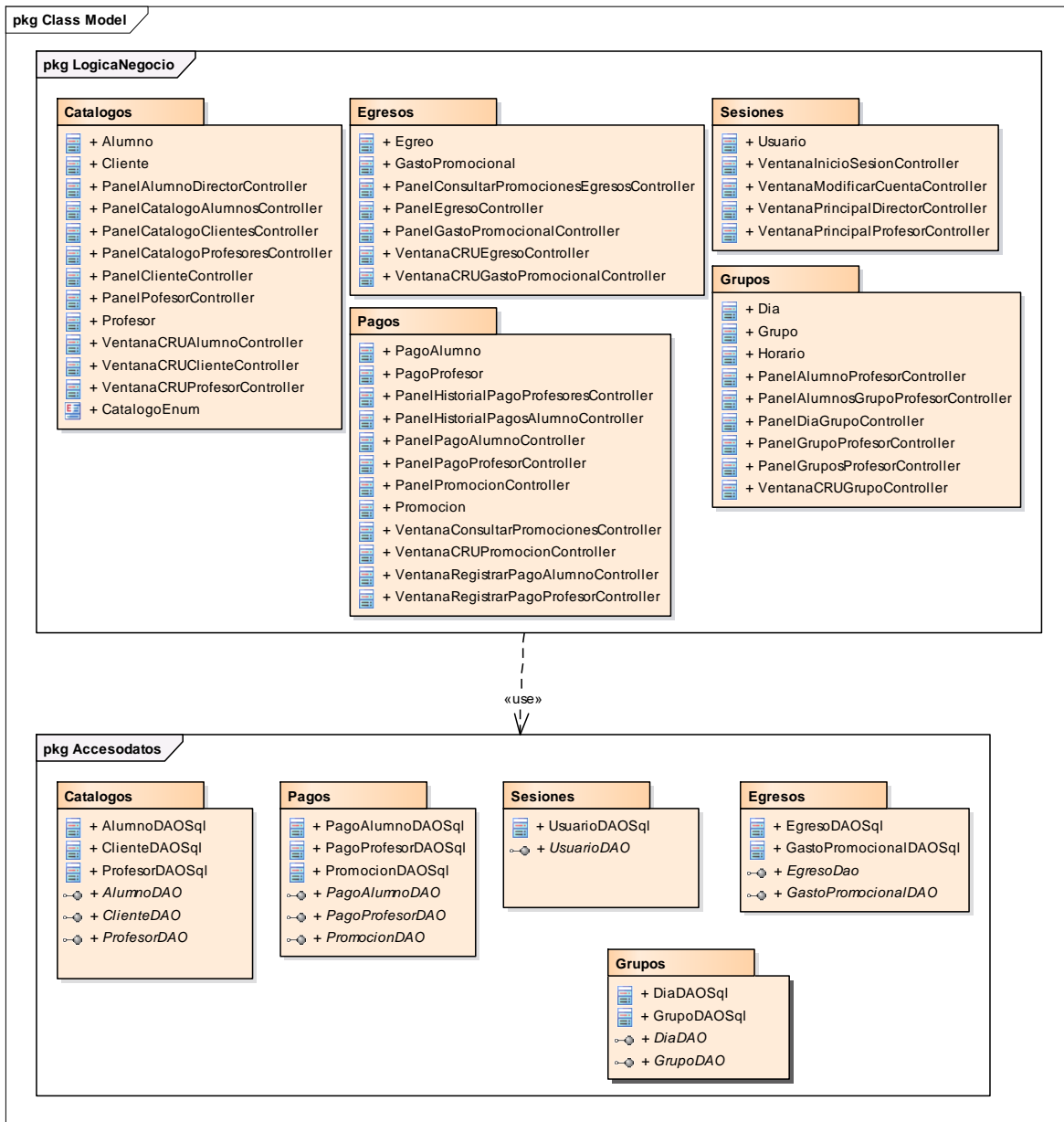
#### 1.4.17. CU 26 – Modificar cuenta de usuario.



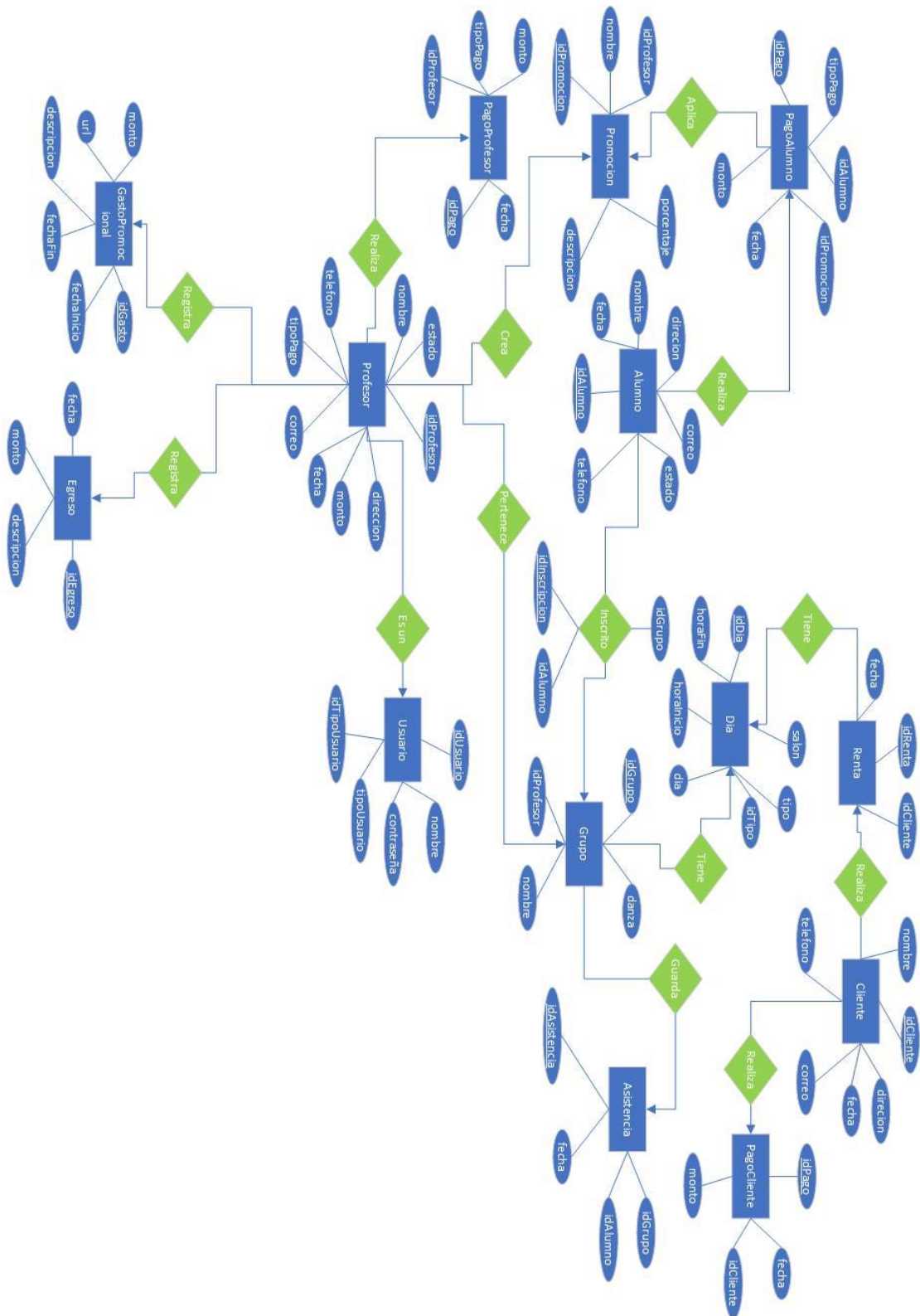
### 1.5. Diagrama de clases.



## 1.6. Diagrama de paquetes.



## 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}.

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

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

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}.

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

PagoCliente {idPago, idCliente, fecha, monto}.

## 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.

<b>Función</b>	registrarEgreso: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
<ul style="list-style-type: none"><li>- Monto: 1200</li><li>- Descripción: Una escoba.</li><li>- Fecha: 2012-12-12</li></ul>	Registro valido. true	N/A.	Registro valido de un nuevo egreso.	Registro valido. true
<ul style="list-style-type: none"><li>- Monto: monto invalido</li><li>- Descripción: Una escoba.</li><li>- Fecha: 2012-12-12</li></ul>	Registro no valido. false	N/A.	Notificación de un registro invalido.	Registro no valido. false

<b>Función</b>	editarEgreso: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
<ul style="list-style-type: none"><li>- IdCosto: 3</li><li>- Monto: 1200</li><li>- Descripción: Una escoba.</li><li>- Fecha: 2012-12-12</li></ul>	Edición valida: true	N/A.	Edición correcta.	Edición valida: true
<ul style="list-style-type: none"><li>- IdCosto: 500</li><li>- Monto: monto</li><li>- Descripción: Una escoba.</li><li>- Fecha: 2012-12-12</li></ul>	Edición no valida: false	N/A.	Edición incorrecta.	Edición no valida: false

<b>Función</b>	obtenerEgresos: list			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>

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

### 3.2.1.2. Clase *GastoPromocional*DAOSql.

<b>Función</b>	editarGasto: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
<ul style="list-style-type: none"> <li>- Descripción: módulo de prueba. Fecha inicio:2012-12-12,</li> <li>- fecha fin: 2012-12-12</li> <li>- monto: "1200"</li> <li>- URL: "www.uv.mx"</li> </ul>	Edición de gasto valido	N/A.	Una edición de gasto valida.	Edición de gasto valido
<ul style="list-style-type: none"> <li>- New Gasto();</li> </ul>	Gasto invalido	N/A	Confirmación de que no puede registrarse el gasto.	gasto invalido.

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

<b>Función</b>	registrarGasto: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
<ul style="list-style-type: none"> <li>- Descripción: módulo de prueba. Fecha inicio:2012-12-12,</li> <li>- fecha fin: 2012-12-12</li> <li>- monto: "1200"</li> </ul>	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.

<b>Función</b>	crearUsuario: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
<ul style="list-style-type: none"> <li>- contraseña: 1234</li> <li>- nombre: Antonio</li> <li>- idTipoUsuario: 1</li> <li>- idUsuario: 1</li> <li>- tipoUsuario: 1</li> </ul>	true	N/A.	Un registro exitoso de un nuevo usuario.	true.

<b>Función</b>	obtenerProfesor: Profesor			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
<ul style="list-style-type: none"> <li>- contraseña: nemesis</li> <li>- nombre: Gabriela</li> </ul>	Profesor valido.	N/A.	Un registro de un profesor existente en el sistema.	Profesor valido.
<ul style="list-style-type: none"> <li>- idTipoUsuario: -1</li> </ul>	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.

<b>Función</b>	obtenerPagos(int idProfesor): List<PagoProfesor>			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
idProfesor: 1.	Ninguna excepción relativa a la persistencia.	No hay excepciones.	<ul style="list-style-type: none"> <li>- El profesor con el id '1' está registrado.</li> <li>- Existen o no pagos del profesor registrados.</li> </ul>	Lista válida.
idProfesor: -1.	NullPointerException	NullPointerException	<ul style="list-style-type: none"> <li>- El profesor con el id '-1' no existe.</li> </ul>	Lista vacía.

<b>Función</b>	registrarPago(PagoProfesor pago, int idProfesor): boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
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.

<b>Función</b>	editarPromocion(Promocion promocion): boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
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.

Promoción: - Descripción: promoción de prueba. - idPromoción: 1 - idProfesor: 0 - Nombre: promoción de descuento 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 - idPromoción: 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 - idPromoción: 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	NullPointerException.	N/A.	Se espera que se lance una excepción sobre un id de profesor que no existe.	NullPointerException.

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	NullPointerException.	N/A.	Para que esta prueba funcione no debe existir en base de datos una promoción con ese id.	NullPointerException.

### 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.

<ul style="list-style-type: none"> <li>- Pago alumno: fecha: new Date(). idPago: 0. Monto: "no es un pago" idTipoPago: 1</li> <li>- idAlumno: 1</li> <li>- idPromocion: 1</li> </ul>	False.	N/A.	Se espera que no se registre un pago de alumno con una cantidad no valida o con incoherencia de datos.	False.
<ul style="list-style-type: none"> <li>- Pago alumno: fecha: new Date(). idPago: 0. Monto: "500" idTipoPago: 1</li> <li>- idAlumno: 1</li> <li>- TidPromocion: 0</li> </ul>	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
<ul style="list-style-type: none"> <li>- idAlumno: 1</li> </ul>	List <PagoAlumno>	N/A.	Se espera una lista de pagos de un alumno.	List <PagoAlumno>
<ul style="list-style-type: none"> <li>- idAlumno: 0</li> </ul>	NullPointerException.	N/A.	Se espera que el proceso arroje una excepción de tipo null debido a la no existencia del alumno.	NullPointerException.

### 3.2.4. Modulo catálogos.

#### 3.2.4.1. Clase AlumnoDAOsql.

Función registrarAlumno: boolean				
Entradas	Salidas esperadas	Salidas obtenidas	Condiciones de salida	Resultado
Alumno:	N/A.	N/A.	Alumno con todas sus	True.



<ul style="list-style-type: none"> <li>- <b>Nombre:</b> Alumno de prueba.</li> <li>- <b>Correo:</b> <a href="mailto:alumno@prueba.clase">alumno@prueba.clase</a></li> <li>- <b>Teléfono:</b> 2281917762.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Estado:</b> true.</li> </ul>			propiedades establecidas correctamente.	
Alumno: <ul style="list-style-type: none"> <li>- <b>Nombre:</b> Alumno de prueba.</li> <li>- <b>Correo:</b> @com.gmail</li> <li>- <b>Teléfono:</b> 2281917762.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Estado:</b> true.</li> </ul>	N/A.	N/A.	Alumno con todas sus propiedades establecidas correctamente, pero un correo no permitido.	False.
Alumno: <ul style="list-style-type: none"> <li>- <b>Nombre:</b> Alumno de prueba.</li> <li>- <b>Correo:</b> <a href="mailto:alumno@prueba.clase">alumno@prueba.clase</a></li> <li>- <b>Teléfono:</b> 228262z221.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Estado:</b> true.</li> </ul>	N/A.	N/A.	Alumno con todas sus propiedades establecidas correctamente, pero un teléfono no permitido.	False.

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:	N/A.	N/A.	Alumno ya registrado en base, pero con	False.

- Id: 0.			un id inexistente.	
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.

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

<b>Función</b>	obtenerAlumno: List<Alumno>			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
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.

<b>Función</b>	List<Alumno> obtenerAlumnos(int idGrupo)			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
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.

<b>Función</b>	registrarCliente: boolean
----------------	---------------------------

Entradas	Salidas esperadas	Salidas obtenidas	Condiciones de salida	Resultado
Cliente: - <b>Nombre:</b> Cliente de prueba. - <b>Correo:</b> <a href="mailto:cliente@prueba.clase">cliente@prueba.clase</a> - <b>Teléfono:</b> 2289365356. - <b>Dirección:</b> Dirección de prueba.	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente.	True.
Cliente: - <b>Nombre:</b> Cliente de prueba. - <b>Correo:</b> @com.gmail - <b>Teléfono:</b> 2289365356. - <b>Dirección:</b> Dirección de prueba.	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente, pero un correo no permitido.	False.
Cliente: - <b>Nombre:</b> Alumno de prueba. - <b>Correo:</b> <a href="mailto:cliente@prueba.clase">cliente@prueba.clase</a> - <b>Teléfono:</b> 228262z221. - <b>Dirección:</b> 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.

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

<b>Función</b>	obtenerCientes: List<Cliente>			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
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.

<b>Función</b>	registrarProfesor: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
Profesor:  - <b>Nombre:</b> Profesor de prueba.	N/A.	N/A.	Profesor con todas sus propiedades establecidas correctamente.	True.

<ul style="list-style-type: none"> <li>- <b>Correo:</b> profe@sor.clase</li> <li>- <b>Teléfono:</b> 2281916654.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Monto:</b> 1500.</li> </ul>				
Profesor: <ul style="list-style-type: none"> <li>- <b>Nombre:</b> Profesor de prueba.</li> <li>- <b>Correo:</b> lkjhlkj</li> <li>- <b>Teléfono:</b> 2281916654.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Monto:</b> 1500.</li> </ul>	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente, pero un correo no permitido.	False.
Cliente: <ul style="list-style-type: none"> <li>- <b>Nombre:</b> Profesor de prueba.</li> <li>- <b>Correo:</b> profe@sor.clase</li> <li>- <b>Teléfono:</b> lkjhlkj.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Monto:</b> 1500.</li> </ul>	N/A.	N/A.	Cliente con todas sus propiedades establecidas correctamente, pero un teléfono no permitido.	False.
Profesor: <ul style="list-style-type: none"> <li>- <b>Nombre:</b> Profesor de prueba.</li> <li>- <b>Correo:</b> profe@sor.clase</li> <li>- <b>Teléfono:</b> 2281916654.</li> <li>- <b>Dirección:</b> Dirección de prueba.</li> <li>- <b>Monto:</b> 1iuy.</li> </ul>	N/A.	N/A.	Profesor con todas sus propiedades establecidas correctamente, pero un monto no permitido.	False.

<b>Función</b>	editarProfesor: boolean			
<b>Entradas</b>	<b>Salidas esperadas</b>	<b>Salidas obtenidas</b>	<b>Condiciones de salida</b>	<b>Resultado</b>
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.

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

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

			profesores registrados.	
--	--	--	----------------------------	--

### 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. - HoraInicio: 19:00. - Id: 0. - idTipo: 5. - Salón: Y. - Tipo: true.	True.	True.	El día tienes establecidos todos sus atributos correctamente.	True.

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.	False.	False.	El día con el id 0 no existe.	False.

<ul style="list-style-type: none"> <li>- HoraFin: 09:30.</li> <li>- HoraInicio: 09:00.</li> <li>- Id: 0.</li> <li>- idTipo: 18.</li> <li>- Salón: X.</li> <li>- Tipo: true.</li> </ul>				
--	--	--	--	--

### 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: <ul style="list-style-type: none"> <li>- Nombre: Soy un grupo.</li> <li>- Danza: Un tipo de danza.</li> <li>- Id: 0.</li> <li>- Profesor: id: 1.</li> <li>- Dia: <ul style="list-style-type: none"> <li>o Dia: Miércoles.</li> <li>o Id: 0.</li> <li>o Inicio: 09:30.</li> <li>o Fin: 11:00.</li> <li>o Salón: X.</li> <li>o Tipo: true.</li> <li>o idTipo: 0.</li> </ul> </li> </ul>	True.	True.	<ul style="list-style-type: none"> <li>- El grupo tiene sus atributos asignados correctamente</li> <li>- El día marcado tiene sus atributos correctos y no choca con otro horario en la base de datos.</li> <li>- El profesor con id 1 existe en base de datos.</li> </ul>	True.
Grupo: <ul style="list-style-type: none"> <li>- Nombre: Soy un grupo.</li> <li>- Danza: Un tipo de danza.</li> <li>- Id: 0.</li> <li>- Profesor: id: 0.</li> </ul>	False.	False.	<ul style="list-style-type: none"> <li>- El grupo tiene sus atributos asignados correctamente</li> <li>- El día marcado tiene sus atributos</li> </ul>	False



<ul style="list-style-type: none"> <li>- Dia: <ul style="list-style-type: none"> <li>o Dia: Miércoles.</li> <li>o Id: 0.</li> <li>o Inicio: 09:30.</li> <li>o Fin: 11:00.</li> <li>o Salón: X.</li> <li>o Tipo: true.</li> <li>o idTipo: 0.</li> </ul> </li> </ul>			<p>correctos y no choca con otro horario en la base de datos.</p> <ul style="list-style-type: none"> <li>- El profesor con id 0 no existe en base de datos.</li> </ul>	
<p>Grupo:</p> <ul style="list-style-type: none"> <li>- Nombre: Soy un grupo.</li> <li>- Danza: Un tipo de danza.</li> <li>- Id: 0.</li> <li>- Profesor: id: 0.</li> <li>- Dia: <ul style="list-style-type: none"> <li>o Dia: Lunes.</li> <li>o Id: 0.</li> <li>o Inicio: 04:00.</li> <li>o Fin: 15:00.</li> <li>o Salón: X.</li> <li>o Tipo: true.</li> <li>o idTipo: 0.</li> </ul> </li> </ul>	Horario Exception.	Horario Exception.	<ul style="list-style-type: none"> <li>- El grupo tiene sus atributos asignados correctamente</li> <li>- El día marcado choca con otro horario en la base de datos.</li> <li>- El profesor con id 1 existe en base de datos.</li> </ul>	Horario Exception.

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.
----------	--	-----------------------	--	---------------