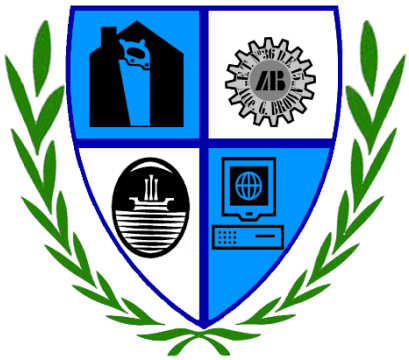
Escuela Técnica Nº36 D.E 15
“Almirante Guillermo Brown”



Análisis de Sistemas - 5to año

Profesores: De Couto, Anabella; Uzal, Alan
Materia: Análisis de Sistemas

Índice

1. Entrevistas – Relevamiento de Requisitos

En el proceso de análisis de sistemas, una de las actividades fundamentales es el relevamiento de requisitos, es decir, la recolección de información que permita conocer qué necesita realmente el cliente o los usuarios para el nuevo sistema. Dentro de las técnicas más utilizadas para este fin se encuentra la entrevista, que consiste en una conversación planificada entre el analista y uno o más actores involucrados en el sistema. A través de esta técnica se busca extraer información relevante sobre los procesos actuales, detectar problemas y entender las expectativas y necesidades funcionales y no funcionales que el nuevo sistema debería satisfacer.

La entrevista es una técnica cualitativa que permite al analista explorar en profundidad aspectos del trabajo diario de los usuarios, sus opiniones y experiencias. Este acercamiento directo ayuda a comprender no solo los aspectos formales del sistema, sino también las dinámicas informales, usos particulares y problemas que tal vez no se documentan, pero afectan al funcionamiento general.

El **objetivo principal de una entrevista** en este contexto es recolectar datos precisos y significativos sobre los procesos del negocio, los requisitos del sistema, las necesidades de los usuarios, así como aspectos subjetivos que pueden influir en el diseño del sistema, como expectativas, frustraciones y sugerencias. Además, la entrevista permite establecer una buena relación entre el analista y los usuarios clave, lo cual es esencial para el éxito del proyecto, ya que promueve la confianza y colaboración mutua.

Una buena entrevista debe contar con ciertas características clave. En primer lugar, debe estar bien planificada: es fundamental tener claridad sobre qué se quiere averiguar y a quién se va a entrevistar. También debe ser clara y comprensible, es decir, las preguntas deben estar formuladas de forma simple y accesible para el entrevistado, evitando tecnicismos innecesarios. A su vez, debe ser flexible, ya que muchas veces surgirán temas nuevos que no estaban contemplados, pero son de gran valor. Por último, es importante que esté documentada adecuadamente, mediante anotaciones, formularios o grabaciones (siempre con consentimiento), y que sea parte de un proceso iterativo, es decir, no siempre se obtiene toda la información en una sola sesión y puede ser necesario volver a entrevistar a la misma persona u otras para complementar los datos.

1.1. Tipos y enfoque de entrevistas

Existen diferentes **tipos de entrevistas**, que se clasifican según su estructura y la cantidad de participantes. Por un lado, según su nivel de

estructuración, podemos encontrar entrevistas estructuradas, en las que todas las preguntas están definidas de antemano y se hacen en un orden fijo; este tipo de entrevista es útil para comparar respuestas entre distintos entrevistados, pero puede resultar rígida. Luego están las entrevistas semiestructuradas, donde se parte de un guión básico de preguntas, pero se permite cierta libertad para profundizar o desviarse si surge algún tema relevante. Este tipo es muy usado en análisis de sistemas por su equilibrio entre planificación y flexibilidad. Finalmente, las no estructuradas son conversaciones abiertas, sin un guión definido, que permiten explorar en profundidad las ideas y opiniones del entrevistado, aunque presentan la desventaja de ser difíciles de sistematizar y analizar.

Tipo	**Descripción**	**Ventajas**	**Desventajas**
Estructurada	Preguntas fijas y en orden establecido.	Fácil de comparar respuestas.	Rígida, puede limitar la información espontánea.
Semiestructurada	Preguntas preparadas, pero con posibilidad de desviaciones o profundización.	Flexible, permite explorar más.	Requiere más experiencia del entrevistador.
No estructurada	Conversación libre, sin guión fijo.	Muy rica en información cualitativa.	Difícil de analizar, puede irse del tema.
Individual	Entrevista a una sola persona.	Permite profundidad y confianza.	Tiempo limitado, no hay interacción de ideas.
Grupal	Varias personas entrevistadas a la vez (también llamadas sesiones de grupo o focus group).	Diversidad de opiniones, puede generar ideas nuevas.	Puede haber dominancia de un participante, conflictos o dispersión.

Por otro lado, en cuanto a la cantidad de personas, las entrevistas pueden ser individuales, en las que se conversa con un solo entrevistado a la vez, lo que facilita la confidencialidad y permite obtener información detallada; o grupales, también conocidas como sesiones de grupo o focus group, donde se entrevista a varias personas simultáneamente, lo cual favorece el intercambio de ideas, pero

puede generar conflictos, dispersión o que algunas voces no se escuchen con claridad.

Una parte crucial del proceso de entrevistas es elegir adecuadamente a los entrevistados. Es recomendable incluir tanto a los usuarios finales del sistema (quienes lo utilizan en la práctica), como a los supervisores, jefes de área y gerentes, quienes pueden aportar una visión más global de los procesos. También se debe considerar entrevistar a técnicos o responsables de sistemas actuales, y en algunos casos, a clientes o beneficiarios externos del sistema.

El contenido de las entrevistas puede orientarse a extraer requisitos funcionales, que son las funciones y comportamientos que el sistema debe realizar (por ejemplo: registrar una venta, calcular un descuento, generar un informe), o bien requisitos no funcionales, que son características de calidad del sistema, como rendimiento, seguridad, disponibilidad, facilidad de uso, etc.

1.2. Las preguntas en los distintos tipos de entrevistas

En la **entrevista estructurada** se sigue un guion rígido, con preguntas definidas de antemano que se realizan en un orden específico. Es útil cuando se necesita comparar respuestas entre varios entrevistados o estandarizar la información.

Ejemplos de preguntas en una entrevista estructurada:

- ¿Con qué frecuencia utiliza el sistema actual?
- ¿Qué módulo del sistema utiliza con mayor regularidad?
- ¿El sistema le permite imprimir reportes fácilmente? (Sí/No)
- ¿Cuántos usuarios acceden al sistema simultáneamente?
- ¿Cada cuánto realiza una copia de seguridad de los datos?

Estas preguntas suelen ser cerradas, es decir, permiten respuestas limitadas como "sí" o "no", o bien con opciones específicas.

En la entrevista semiestructurada se combina preguntas preparadas con la posibilidad de realizar nuevas preguntas según el rumbo de la conversación. Es la más común en el análisis de sistemas porque permite explorar temas en profundidad sin perder el foco.

Ejemplos de preguntas en una entrevista semiestructurada:

- ¿Podría describirme cómo es el proceso completo desde que ingresa un pedido hasta que se entrega?
- ¿Qué problemas ha tenido recientemente con el sistema actual?
- ¿Qué información necesita ver de forma inmediata al iniciar sesión?
- ¿Qué mejoras cree que debería tener el sistema nuevo?

- ¿Le gustaría que el sistema funcione también desde el celular?

Estas preguntas suelen ser abiertas, permitiendo al entrevistado desarrollar su respuesta. También se pueden alternar con preguntas cerradas según sea necesario.

En la **entrevista no estructurada** no se tiene un guion definido, sino que se basa en una conversación libre. Es útil en etapas iniciales del proyecto, cuando se necesita explorar el entorno sin demasiada información previa.

Ejemplos de preguntas en una entrevista no estructurada:

- Cuénteme cómo es su día de trabajo y cómo el sistema lo ayuda (o no).
- ¿Hay alguna tarea que le resulte especialmente complicada al usar el sistema?
- ¿Qué aspectos del sistema actual le resultan más útiles y cuáles le resultan innecesarios?
- ¿Qué expectativas tiene del nuevo sistema?

Estas entrevistas permiten obtener información muy rica, pero requieren habilidad del entrevistador para guiar la conversación sin perder el objetivo.

Algunos ejemplos de preguntas útiles para **obtener requisitos funcionales** podrían ser:

- ¿Qué tareas realiza actualmente con el sistema?
- ¿Qué información necesita consultar con frecuencia?
- ¿Qué procesos considera prioritarios o más importantes?
- ¿Qué errores o problemas suele encontrar al usar el sistema?

En cuanto a los **requisitos no funcionales**, se podrían hacer preguntas como:

- ¿Con qué rapidez espera que el sistema responda?
- ¿Qué tan importante es que el sistema esté disponible todo el tiempo?
- ¿Qué medidas de seguridad considera necesarias?
- ¿Qué tipo de dispositivos suele utilizar para acceder al sistema?

1.3. Preguntas abiertas y cerradas

Además del tipo de entrevista, es importante diferenciar entre **preguntas abiertas y cerradas**, ya que el tipo de respuesta que se obtiene cambia significativamente.

Las **preguntas abiertas** permiten respuestas desarrolladas, con libertad de expresión. Son útiles para explorar necesidades, percepciones, ideas y experiencias del usuario.

Ejemplos:

- ¿Cómo se siente con el sistema actual?
- ¿Qué dificultades encuentra al cargar datos?
- ¿Qué haría usted para mejorar el sistema?
- ¿Qué funciones considera indispensables para su trabajo?
- ¿Cómo reaccionan sus compañeros ante los errores del sistema?

Estas preguntas fomentan el diálogo, ayudan a descubrir problemas ocultos y permiten profundizar según las respuestas.

Las **preguntas cerradas** ofrecen opciones limitadas de respuesta. Son útiles cuando se necesita recopilar datos concretos y medibles.

Ejemplos:

- ¿Utiliza el sistema todos los días? (Sí / No)
- ¿Cuántos informes genera por semana? (1–5 / 6–10 / más de 10)
- ¿Está conforme con el rendimiento del sistema? (Sí / No / Parcialmente)
- ¿Utiliza el sistema desde su celular? (Sí / No)
- ¿El sistema le avisa si hay errores de carga? (Sí / No)

Aunque son más rápidas de responder, no permiten conocer los motivos detrás de la respuesta, por lo que deben usarse en combinación con preguntas abiertas.

En una entrevista efectiva, se recomienda comenzar con preguntas abiertas para generar confianza y obtener una visión general, y luego pasar a preguntas cerradas para obtener datos más precisos. El uso combinado permite lograr un relevamiento completo y balanceado.

1.4. Efectividad de una entrevista

La efectividad de una entrevista en el contexto del análisis de sistemas se mide, principalmente, por la calidad, cantidad y relevancia de la información obtenida durante el encuentro. Una entrevista se considera efectiva si permite alcanzar los objetivos propuestos previamente, es decir, si logra recolectar datos útiles y precisos para comprender el funcionamiento del sistema actual, identificar sus problemas y detectar las necesidades reales de los usuarios.

Uno de los aspectos clave a tener en cuenta es si las preguntas formuladas permitieron obtener respuestas claras, completas y coherentes. Es importante que, al finalizar la entrevista, el analista pueda reconstruir procesos, describir actividades, identificar actores, detectar errores o limitaciones del sistema actual, y vislumbrar posibles mejoras o funcionalidades necesarias. Si esto no se logra, probablemente haya fallas en el enfoque de la entrevista o en su desarrollo.

Otro indicador fundamental es la calidad de la información obtenida. No se trata únicamente de recopilar una gran cantidad de datos, sino de asegurarse de que estos sean relevantes, aplicables y verificables. La información debe ser lo suficientemente detallada como para utilizarla en etapas posteriores del análisis, como la elaboración de diagramas, modelos de procesos o la redacción de requerimientos.

También se debe tener en cuenta el nivel de colaboración del entrevistado. Una entrevista será más efectiva si se logra establecer un clima de confianza que favorezca el diálogo abierto, la participación activa y la disposición del entrevistado a compartir tanto información objetiva como percepciones o sugerencias personales. En este sentido, la actitud del entrevistador es fundamental: debe mostrarse profesional, respetuoso, atento y capaz de adaptar el lenguaje técnico a la comprensión del interlocutor.

La organización del tiempo durante la entrevista es otro aspecto a considerar. El entrevistador debe saber administrar los minutos disponibles para cubrir todos los temas importantes sin desviarse en asuntos irrelevantes. Una buena entrevista permite obtener una gran cantidad de información útil en un tiempo razonable, sin sobrecargar al entrevistado.

Como ejemplo práctico, supongamos que se desea mejorar un sistema de gestión de stock en una empresa. Una entrevista al encargado de almacén podría incluir preguntas como: ¿Cómo realiza actualmente el control del stock? ¿Cada cuánto se detectan faltantes o excesos? ¿Qué funcionalidades considera esenciales para un nuevo sistema? ¿Le gustaría recibir alertas automáticas de bajo stock?

1.5. Ejemplo Práctico

A continuación, se observa un ejemplo de entrevista (narrada). Nosotros en las clases nos enfocaremos en las preguntas, en su planificación y análisis.

> *Analista: Buen día, gracias por su tiempo. ¿Podría contarme brevemente cómo es el funcionamiento actual de la clínica?*
>
> *(Pregunta abierta – exploratoria, entrevista no estructurada)*
>
> *Cliente: Claro, actualmente atendemos de lunes a sábado, con varios veterinarios que trabajan en distintos turnos. Cuando una persona quiere traer a su mascota, tiene que llamarnos o venir directamente. Nosotros anotamos los datos del dueño y la mascota en una planilla Excel y agendamos el turno. A veces se generan confusiones porque no siempre tenemos la información actualizada.*

Analista: Entiendo. ¿Tienen algún sistema informático actualmente?

(Pregunta cerrada – entrevista estructurada)

Cliente: No, todo lo hacemos con planillas de cálculo, algunas en papel y otras en la computadora.

Analista: ¿Qué tipo de información suelen guardar sobre cada mascota?

(Pregunta abierta – específica, entrevista semi-estructurada)

Cliente: Guardamos el nombre, especie, raza, edad, el nombre del dueño, su teléfono y a veces algún detalle clínico si ya vino antes.

Analista: ¿Quisieran mantener un historial clínico en el nuevo sistema?

(Pregunta cerrada)

Cliente: Sí, eso sería ideal. Muchas veces vienen por tratamientos continuos y no tenemos a mano los datos de las consultas anteriores.

Analista: ¿Quiénes usarían el sistema en la clínica?

(Pregunta abierta)

Cliente: Principalmente las recepcionistas y los veterinarios. Las recepcionistas para cargar los datos y dar los turnos, y los veterinarios para ver los datos de los pacientes y registrar diagnósticos o tratamientos.

Analista: ¿Quieren que el sistema esté disponible en línea o solo en la red local de la clínica?

(Pregunta cerrada)

Cliente: En lo posible en línea, porque hay veterinarios que trabajan en más de una clínica y quieren acceder desde su casa o desde el celular.

Analista: ¿Qué problemas puntuales les gustaría evitar con el nuevo sistema?

(Pregunta abierta – entrevista semi-estructurada)

Cliente: Evitar turnos solapados, pérdida de datos, confusión con horarios y tener acceso rápido a la historia clínica. También nos gustaría que el sistema pueda enviar recordatorios por WhatsApp o email a los dueños de las mascotas.

Analista: ¿Cuántas personas deberían poder usar el sistema al mismo tiempo?

(Pregunta cerrada)

Cliente: Entre 5 y 8 personas normalmente.

Analista: Por último, ¿qué características considera que son indispensables para que el sistema les resulte útil y fácil de usar?

(Pregunta abierta – evaluación de requisitos no funcionales)

Cliente: Que sea rápido, fácil de aprender, y que funcione sin errores. No queremos algo que necesite capacitación compleja. También es importante que no se caiga, porque lo vamos a usar todo el día.

2. Requerimientos

Luego del proceso de entrevistas y recolección de información, el analista debe organizar y documentar lo obtenido en forma de requerimientos.

En el contexto del análisis de sistemas, los **requerimientos** (también llamados requisitos) representan el punto de partida fundamental para el desarrollo de cualquier sistema informático. Se trata de la descripción clara, precisa y verificable de lo que el sistema debe hacer y de las condiciones bajo las cuales debe funcionar. En otras palabras, los requerimientos especifican las necesidades del cliente o usuario que el sistema debe satisfacer.

Un **requerimiento puede surgir a partir de distintas fuentes**: entrevistas con usuarios, observación de tareas, análisis de documentos, encuestas, análisis de sistemas existentes o nuevas necesidades planteadas por la organización. Documentar correctamente los requerimientos es esencial para evitar malentendidos entre el equipo de desarrollo y el cliente, y para garantizar que el sistema final cumpla con las expectativas.

El **propósito** de definir requerimientos es establecer un acuerdo común entre los interesados sobre lo que debe realizar el sistema. Este acuerdo guía el diseño, el desarrollo, las pruebas y la validación del software. Si los requerimientos están mal definidos o son incompletos, es muy probable que el sistema final sea rechazado por los usuarios o requiera múltiples correcciones costosas.

Además, los requerimientos permiten identificar qué funcionalidades son prioritarias, qué condiciones deben respetarse, qué limitaciones existen, y qué indicadores permitirán evaluar el éxito del proyecto.

2.1. Requerimientos Funcionales

Los **requerimientos funcionales** describen qué debe hacer el sistema, es decir, las funciones, procesos o servicios específicos que debe cumplir.

Representan el comportamiento esperado del sistema ante ciertas entradas o situaciones, y suelen estar relacionados con acciones concretas que los usuarios podrán realizar.

Por ejemplo, en un sistema de gestión de biblioteca, algunos requerimientos funcionales podrían ser:

- El sistema debe permitir registrar nuevos libros en la base de datos.
- El sistema debe permitir a los usuarios buscar libros por título, autor o categoría.
- El sistema debe controlar el préstamo y la devolución de libros.
- El sistema debe enviar un aviso automático cuando un libro esté vencido.

Estos requerimientos están directamente relacionados con las funcionalidades visibles y accesibles del sistema. Generalmente surgen a partir de las tareas actuales del usuario que se desean informatizar o mejorar, y de las nuevas necesidades que el usuario plantea durante el relevamiento.

Es fundamental que los requerimientos funcionales sean claros, específicos y verificables. Deben evitarse las ambigüedades o expresiones vagas como "el sistema debe ser fácil de usar", ya que ese tipo de afirmaciones no describen una función concreta.

2.2. Requerimientos No Funcionales

Los **requerimientos no funcionales** establecen criterios de calidad, restricciones técnicas, condiciones operativas o requisitos generales del sistema. No definen una función específica, pero sí determinan cómo debe comportarse el sistema para ser considerado adecuado.

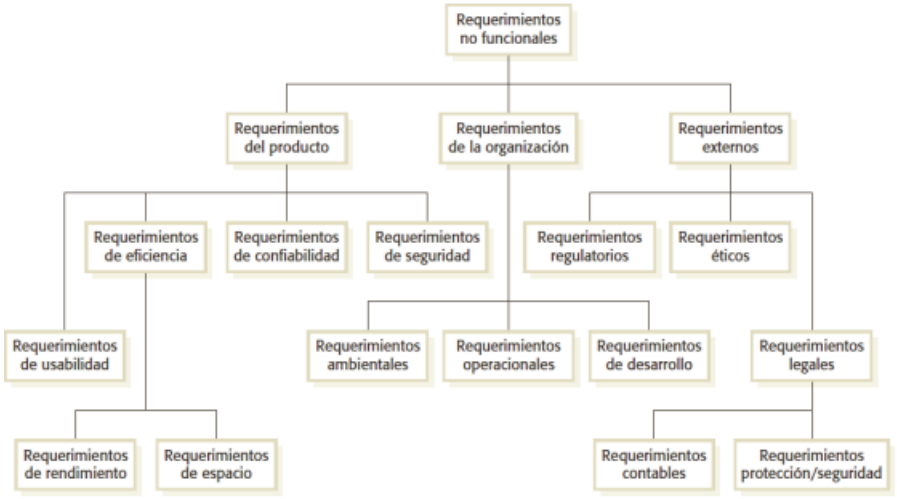
Algunos ejemplos:

- El sistema debe responder a cualquier solicitud del usuario en menos de 2 segundos.
- El sistema debe estar disponible 24 horas al día, 7 días a la semana.
- El sistema debe cumplir con la Ley de Protección de Datos Personales.
- El sistema debe ser compatible con navegadores modernos (Chrome, Firefox, Edge).
- La aplicación debe tener una interfaz intuitiva y accesible para personas con discapacidad visual.

Dentro de los requerimientos no funcionales también se pueden distinguir categorías como:

- Rendimiento: tiempo de respuesta, concurrencia, capacidad.
- Seguridad: control de accesos, cifrado, autenticación.

- Usabilidad: facilidad de uso, navegación, accesibilidad.
- Mantenibilidad: facilidad para corregir errores o hacer modificaciones.
- Portabilidad: posibilidad de ejecutar el sistema en diferentes entornos.
- Fiabilidad: estabilidad, manejo de errores, tolerancia a fallos.



Ambos tipos de requerimientos (funcionales y no funcionales) son fundamentales. Un sistema puede cumplir perfectamente con todas sus funciones, pero si es lento, difícil de usar o inseguro, no será aceptado por los usuarios. Por eso, el análisis debe contemplar ambos aspectos desde el principio.

2.3. Redacción de requerimientos

Una parte fundamental del trabajo del analista de sistemas es redactar los requerimientos de forma clara, precisa y comprensible tanto para el cliente como para el equipo técnico. La manera en que se redactan influye directamente en la calidad del sistema final. Un requerimiento mal formulado puede generar malentendidos, errores de interpretación y costos innecesarios.

Un requerimiento bien redactado debe ser:

- Claro: fácil de entender, sin ambigüedades.
- Conciso: breve pero completo.
- Específico: detalla exactamente qué se espera.
- Verificable: debe poder comprobarse si se cumple o no.
- Alcanzable: debe ser realista, posible de implementar.
- Relevante: debe estar alineado con los objetivos del sistema.
- Trazable: debe poder vincularse con otras etapas del proceso (casos de uso, diseño, pruebas).

Recomendaciones para redactar correctamente:

- Usar un formato estándar: "El sistema debe [verbo en infinitivo] [detalle de la función o condición] [criterio de éxito o alcance]."
- Evitar palabras vagas: Evitar términos como “rápido”, “fácil”, “bueno”, “intuitivo”, “seguro” sin especificar qué significan concretamente.
- Usar cuantificadores cuando sea posible: Por ejemplo: tiempo de respuesta, cantidad de registros, porcentaje de éxito, número de clics.

A continuación, se presentan ejemplos de requerimientos comunes que están mal formulados, seguidos de una mejora posible:

Mal redactado	**Bien redactado**
"El sistema debe ser intuitivo."	"El sistema debe mostrar una guía interactiva al primer uso, explicando las funciones básicas mediante mensajes emergentes."
"Debe funcionar con cualquier sistema operativo."	"El sistema debe ser compatible con Windows 10, Windows 11 y Ubuntu 22.04."
"El sistema debe proteger los datos del usuario."	"Los datos personales del usuario deben ser almacenados cifrados con el algoritmo AES de 256 bits."
"El sistema debe guardar la información automáticamente."	"El sistema debe realizar un guardado automático de la información del formulario cada 60 segundos sin intervención del usuario."

2.4. Requerimientos del Usuario vs. Requerimientos del Sistema.

En el proceso de análisis y especificación de requerimientos, es fundamental distinguir entre los requerimientos del usuario y los requerimientos del sistema. Aunque están estrechamente relacionados, cumplen funciones diferentes y se redactan en niveles distintos de detalle.

Los **requerimientos del usuario** son descripciones generales y de alto nivel de lo que los usuarios necesitan que el sistema haga para cumplir con sus objetivos o resolver sus problemas. Este tipo de requerimientos se expresan, generalmente, en lenguaje natural y no técnico, ya que están destinados a ser entendidos por usuarios finales, clientes y otros interesados no especializados. Su función principal es establecer qué espera el usuario del sistema y por qué lo necesita.

Ejemplo:

"El sistema debe permitir que los empleados registren su asistencia diaria de manera sencilla desde cualquier dispositivo con conexión a internet."

Estos requerimientos ayudan a contextualizar el sistema desde la perspectiva del usuario y son clave para establecer las expectativas iniciales del proyecto. No incluyen detalles técnicos ni especificaciones de cómo se logrará la funcionalidad.

Los **requerimientos del sistema**, en cambio, son descripciones más técnicas, detalladas y precisas que derivan de los requerimientos del usuario. Definen cómo debe comportarse el sistema, qué funciones específicas debe cumplir, con qué restricciones técnicas, y cómo debe interactuar con otros sistemas o componentes. Están escritos en un lenguaje más técnico, pensado para desarrolladores, diseñadores y testers, ya que sirven como base para el diseño, la implementación y la validación del sistema.

Ejemplo:

"El sistema debe permitir a cada usuario autenticado registrar un evento de asistencia en la base de datos mediante un formulario accesible desde navegadores compatibles con HTML5, almacenando la fecha y hora del servidor junto con el ID del empleado."

Este tipo de requerimientos puede incluir reglas de negocio, validaciones, especificaciones de interfaz, rendimiento esperado, seguridad, y más. A menudo se derivan múltiples requerimientos del sistema a partir de un único requerimiento del usuario.

Podemos decir que los requerimientos del usuario son el punto de partida, y los del sistema son su traducción operativa. Para que el desarrollo sea exitoso, es necesario que exista una trazabilidad clara entre ellos: cada requerimiento del sistema debe satisfacer uno o más requerimientos del usuario, y cada requerimiento del usuario debe estar cubierto por al menos uno del sistema. Esta relación asegura que lo que se construye responde a lo que se necesita.

2.5. Ejemplo Práctico

A continuación, veremos un texto como ejemplo y un detalle de requerimientos funcionales y no funcionales que se observan en él:

La empresa desea implementar un sistema para gestionar las órdenes de compra de sus clientes. Actualmente, este proceso se realiza de forma manual a través de correos electrónicos, lo que provoca demoras, errores y pérdida de información. Los empleados deben revisar constantemente la bandeja de entrada, copiar los datos en una planilla de Excel y luego enviarla al sector de facturación. Esto no solo consume tiempo, sino que también genera duplicación de datos y retrabajo.

El nuevo sistema debe permitir que los clientes puedan ingresar sus pedidos directamente desde un portal web, seleccionando productos, cantidades y fechas de entrega. También debe enviar notificaciones automáticas por correo electrónico cuando se confirme un pedido o si ocurre algún problema, como falta de stock. Los administradores del sistema deben poder visualizar y modificar los pedidos, así como acceder a un historial detallado de operaciones.

Además, es importante que el sistema sea accesible desde cualquier dispositivo, tanto computadoras como celulares, y que cargue rápidamente. Dado que se manejará información sensible de los clientes, se requiere garantizar la seguridad de los datos. Por último, el sistema debe integrarse con el software de facturación que ya utiliza la empresa, para evitar trabajo duplicado.

Requisitos funcionales extraídos:

- El sistema debe permitir a los clientes registrar órdenes de compra desde un portal web.
- El sistema debe permitir seleccionar productos, cantidades y fechas de entrega.
- El sistema debe enviar notificaciones por correo electrónico a los clientes cuando se confirme un pedido.
- El sistema debe notificar al cliente si hay problemas con el pedido, como falta de stock.
- El sistema debe permitir a los administradores visualizar y modificar los pedidos registrados.
- El sistema debe registrar un historial de operaciones (pedidos realizados, modificados, cancelados, etc.).
- El sistema debe integrarse con el software de facturación existente.

Requerimientos no funcionales extraídos:

- El sistema debe ser accesible desde navegadores web de escritorio y dispositivos móviles.
- El sistema debe tener un tiempo de carga inferior a 3 segundos en conexiones estándar.
- El sistema debe garantizar la confidencialidad e integridad de los datos personales y comerciales de los clientes.
- El sistema debe contar con autenticación segura para clientes y administradores.
- La integración con el software de facturación debe realizarse mediante una API REST ya disponible en la empresa.
- La interfaz debe estar disponible en español y adaptarse automáticamente a distintos tamaños de pantalla (diseño responsivo).

3. Casos de Uso

Los diagramas de casos de uso son una herramienta fundamental dentro del análisis de sistemas, ya que permiten representar de forma visual y clara las funcionalidades que un sistema debe ofrecer y cómo los diferentes usuarios o sistemas externos (conocidos como actores) interactúan con esas funcionalidades. Estos diagramas se utilizan principalmente durante la fase de análisis de requisitos para definir el alcance y los límites del sistema, así como para facilitar la comunicación entre analistas, desarrolladores y usuarios finales.

Un caso de uso representa una funcionalidad específica que el sistema realiza para satisfacer una necesidad de un actor. Para facilitar su identificación y mantener un orden, los casos de uso suelen nombrarse en gerundio, utilizando verbos que reflejan acciones en proceso, por ejemplo: “Comprar producto”, “Registrar usuario” o “Consultar saldo”. Además, se acostumbra numerar estos casos de uso para mejorar su trazabilidad y organización, como “CU001 - Comprar producto”.

Los Casos de Uso describen tanto lo que hace el actor como lo que hace el sistema cuando se interactúa con él. Están acotados al uso de una determinada funcionalidad, claramente diferenciada, del sistema. Además, un caso de uso realiza cierto trabajo cuyo efecto es tangible para el actor.

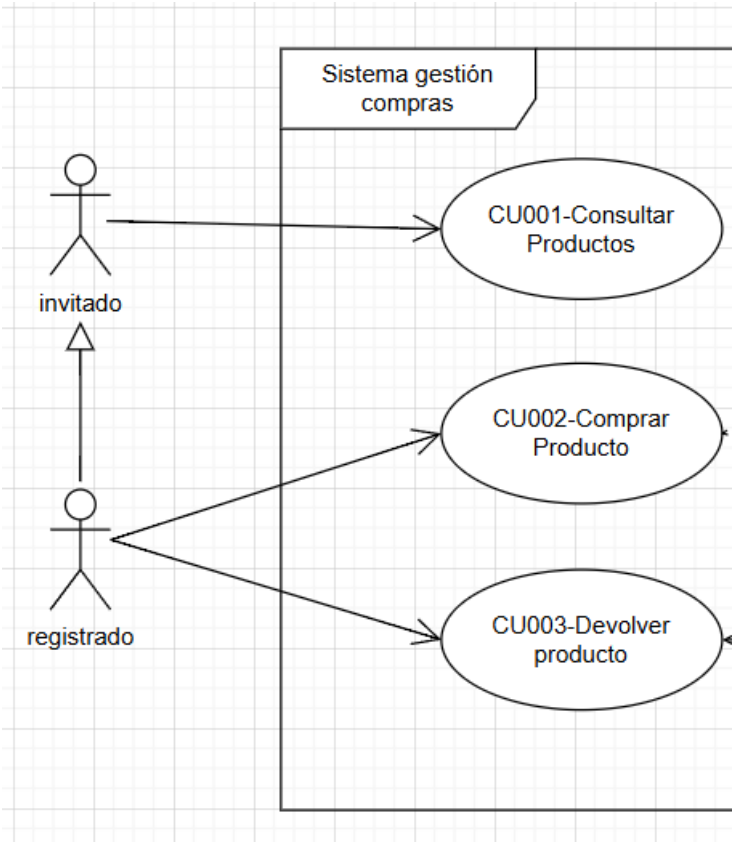
3.1. Actores

Los actores representan a todas las entidades externas que interactúan con el sistema. Estas entidades pueden ser personas, otros sistemas, dispositivos o incluso procesos externos que tienen algún tipo de comunicación con el sistema en cuestión. Los actores no forman parte del sistema mismo; por eso, siempre se colocan fuera del límite que define el sistema, y su función es iniciar o participar en las acciones que el sistema realiza.

Existen diferentes tipos de actores, principalmente dos grandes categorías: actores primarios y actores secundarios. Los actores primarios son aquellos que inician una interacción con el sistema porque necesitan obtener un servicio o resultado. Por ejemplo, en un sistema bancario, el “Cliente” es un actor primario porque inicia acciones como “Realizar un depósito” o “Consultar saldo”. En cambio, los actores secundarios colaboran con el sistema para que las acciones puedan llevarse a cabo, pero no son quienes las solicitan directamente. Siguiendo con el ejemplo bancario, el “Sistema de verificación de identidad” podría ser un actor secundario que interviene para validar la información proporcionada por el cliente.

Los actores también pueden establecer relaciones de generalización o herencia entre sí. Esto sucede cuando un actor especializado hereda todas las interacciones de otro actor más general, pero además añade características o funcionalidades específicas. Por ejemplo, en un sistema de comercio electrónico, se podría tener un actor genérico “Usuario invitado” y un actor más especializado “Usuario registrado”. El “Usuario registrado” hereda todas las capacidades del “Usuario invitado” (como navegar y consultar productos), pero además puede realizar acciones exclusivas como “Agregar productos al carrito” o “Realizar pagos”.

Estar atentos a que la flecha que los relaciones tiene la punta vacía.



Además, un mismo actor puede participar en múltiples casos de uso, y varios actores pueden estar involucrados en un mismo caso de uso si se requiere colaboración o intercambio entre ellos. Por ejemplo, en un sistema de atención médica, tanto el “Paciente” como el “Médico” pueden participar en el caso de uso “Agendar cita”.

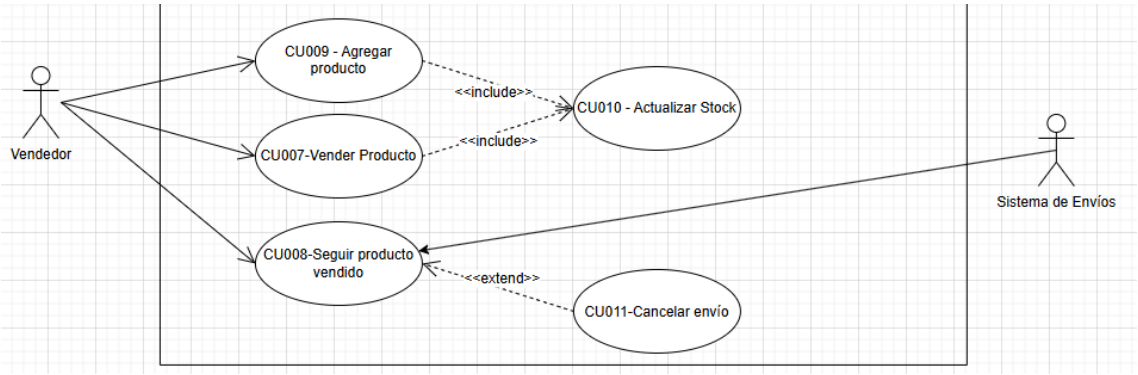
En la elaboración de casos de uso, los **actores secundarios** son aquellos elementos externos al sistema que intervienen de manera indirecta o

complementaria en el desarrollo del caso, pero no inician el proceso principal. Su participación es esencial para que el flujo del caso de uso se complete correctamente, aunque no sean los protagonistas del escenario principal. Dentro del diagrama de Caso de Uso se suelen ubicar del lado derecho del mismo, pero queda a criterio del analista y/u la organización.

A diferencia del actor principal, que es quien inicia la interacción con el sistema para lograr un objetivo (por ejemplo, un cliente que desea comprar un producto), el actor secundario suele representar a otro sistema, base de datos, servicio externo o incluso a otra persona que colabora o proporciona un servicio necesario para completar el caso de uso. En muchos casos, los actores secundarios responden a solicitudes iniciadas por el actor principal o por el propio sistema.

Por ejemplo, en un sistema de compras en línea, cuando un usuario registrado compra un producto, el actor principal es el Usuario Registrado, ya que es quien inicia la operación. Sin embargo, para procesar el pago, el sistema debe interactuar con un sistema de pago externo (como un proveedor de tarjetas de crédito o billetera virtual). En este caso, ese proveedor actúa como actor secundario, ya que participa del proceso, pero no lo inicia.

A continuación, podemos ver un ejemplo de un actor secundario como lo es el Sistema de envíos en un sistema de compra y venta de productos en una plataforma:



Ejemplos de actores secundarios:

- Sistema de pagos (MercadoPago, PayPal, etc.)
- Sistema de inventario
- Sistema bancario
- Administrador del sistema (cuando valida una acción iniciada por otro)
- Proveedor externo
- Servicio de mensajería o logística

Los **actores secundarios** también pueden representar sistemas internos del entorno organizacional, como un sistema de inventario, un sistema de facturación o una base de datos de usuarios. Su correcta identificación es fundamental para reflejar todas las interacciones reales que requiere el sistema y entender sus dependencias externas.

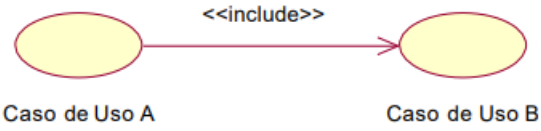
Reconocer e incluir actores secundarios en los diagramas y especificaciones de casos de uso permite:

- Comprender todas las dependencias del sistema.
- Diseñar soluciones más completas y realistas.
- Estimar mejor los riesgos y las necesidades de integración.
- Asegurar que las interfaces y protocolos de comunicación estén contemplados desde el análisis.

3.2. Relaciones entre casos de uso

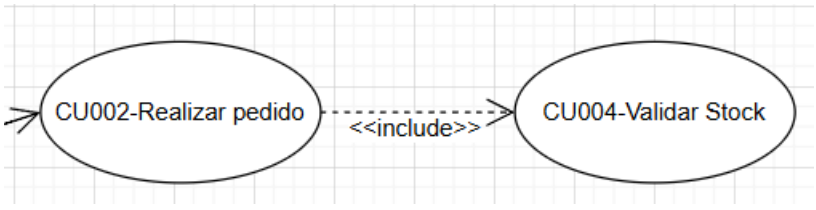
Para organizar y modelar adecuadamente las funcionalidades, los casos de uso pueden relacionarse entre sí mediante diferentes tipos de relaciones. Estas relaciones permiten evitar la duplicación de información, facilitar la reutilización y mostrar variaciones en el comportamiento del sistema.

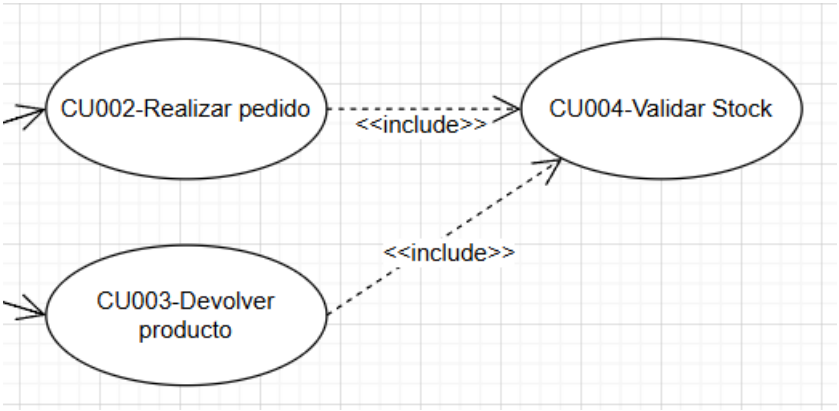
Una de estas relaciones es la **relación de include**. Esta relación indica que un caso de uso siempre incorpora, de forma obligatoria, el comportamiento definido en otro caso de uso. Es decir, cada vez que se ejecuta el caso de uso "principal", automáticamente se ejecuta el caso de uso "incluido". Esta relación se usa para extraer funcionalidades comunes que pueden ser reutilizadas en varios casos de uso, ayudando a evitar la repetición.



El caso de uso A (caso de uso base) incluye en su funcionalidad el caso uso B

Por ejemplo, en un sistema de ventas, el caso de uso "Realizar pedido" puede incluir el caso de uso "Validar stock". Esto significa que cada vez que se realiza un pedido, siempre debe validarse que el stock esté disponible, por lo que "Validar stock" es un comportamiento obligatorio y reutilizado.

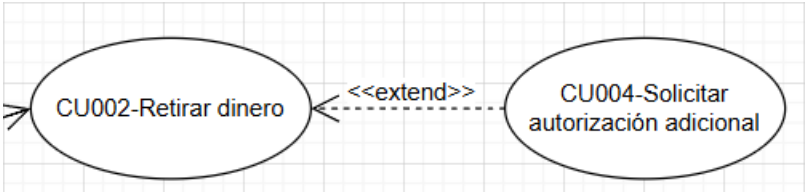




La relación include se representa en los diagramas con una flecha punteada que apunta desde el caso de uso que incluye hacia el caso de uso incluido, acompañada de la etiqueta «<<include>>».

La **relación extend** es utilizada para modelar comportamientos opcionales o excepcionales que amplían el comportamiento de un caso de uso base bajo ciertas condiciones. A diferencia de include, la ejecución del caso de uso extendido no es obligatoria siempre, solo se activa cuando se cumple una condición particular.

Por ejemplo, en un sistema bancario, el caso de uso “Retirar dinero” puede extenderse con “Solicitar autorización adicional” cuando el monto supera un límite predefinido. En este caso, el flujo normal del retiro no requiere autorización adicional, pero bajo ciertas circunstancias, el sistema ofrece esa funcionalidad extra.



En el diagrama, esta relación se dibuja con una flecha punteada desde el caso de uso extendido hacia el caso de uso base, con la etiqueta «<<extend>>».

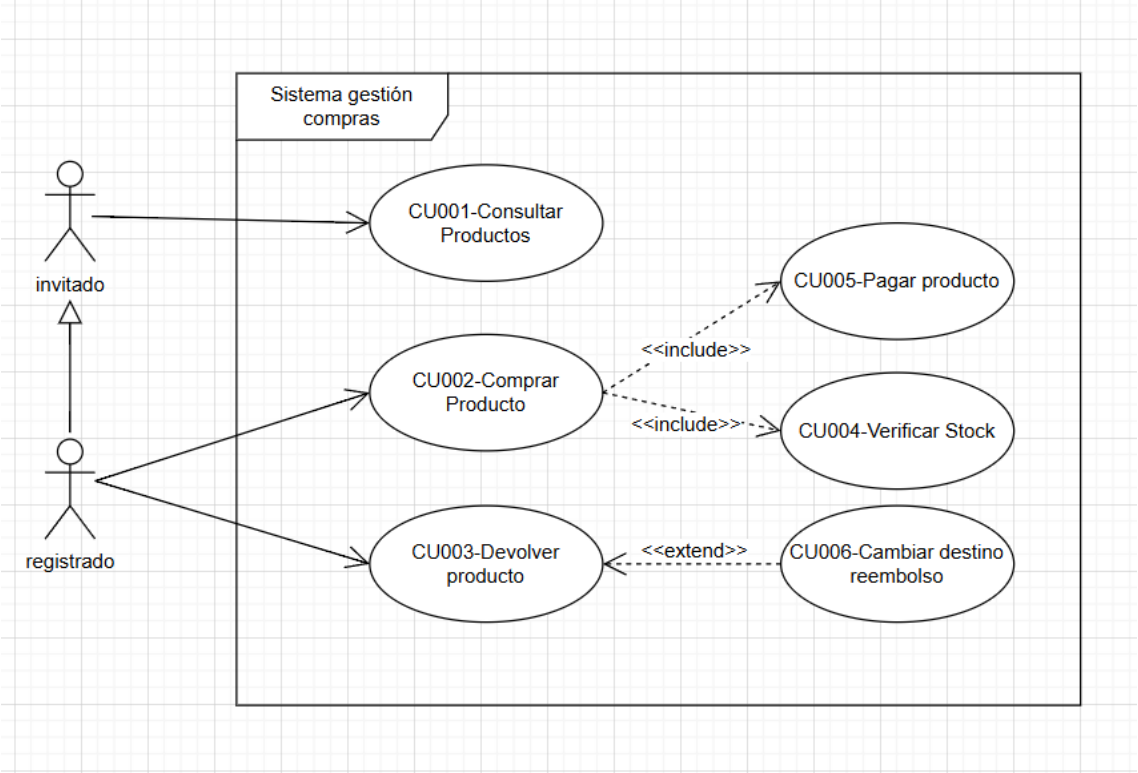
3.3. Límites del sistema

El límite del sistema es una representación visual y conceptual que indica hasta dónde llega la funcionalidad del sistema que se está analizando. Este límite se representa gráficamente mediante un rectángulo que encierra todos los casos de uso, y dentro de este rectángulo solo se incluyen aquellos elementos que forman parte del sistema, es decir, las funcionalidades que el sistema debe ofrecer.

Todo lo que se encuentra fuera del límite del sistema pertenece al entorno externo: usuarios, sistemas externos, dispositivos o cualquier otra entidad que interactúe con el sistema, pero que no forma parte de él. Esta distinción es fundamental para mantener la claridad sobre qué responsabilidades le corresponden al sistema y cuáles dependen de factores externos. Por ejemplo, si estamos modelando un sistema de gestión de biblioteca, los casos de uso como "Prestar libro" o "Registrar devolución" estarán dentro del límite del sistema, mientras que actores como "Usuario" o "Bibliotecario" estarán afuera, ya que son entidades que interactúan con el sistema, pero no lo componen.

El uso correcto del límite del sistema ayuda a definir el alcance del desarrollo y facilita la comprensión para los usuarios, analistas y desarrolladores, ya que aclara qué funciones deben ser implementadas y qué aspectos quedan fuera del sistema.

A continuación, vemos como los casos de uso están incluidos en un rectángulo que define los límites del sistema:



3.4. Especificación de Caso de Uso

Una especificación de caso de uso es un documento o sección dentro de un documento de análisis y diseño de sistemas que describe de forma detallada el comportamiento esperado del sistema desde la perspectiva del usuario. Es parte fundamental de la ingeniería de requisitos, y su propósito es definir claramente qué debe hacer el sistema ante una interacción específica con uno o varios actores.

Es una descripción textual estructurada que acompaña al diagrama de casos de uso. Mientras el diagrama proporciona una vista general de los actores y sus interacciones con el sistema, la especificación profundiza en los detalles, aclarando el flujo de eventos, las condiciones necesarias, las alternativas, los posibles errores y los resultados finales.

Esta especificación funciona como guía tanto para los desarrolladores (quienes deben programar el comportamiento) como para los analistas, testers y usuarios (quienes validan que lo que se construye coincide con lo que se necesita).

La especificación es importante ya que:

- Evita ambigüedades: al detallar cómo debe comportarse el sistema, todos los involucrados entienden lo mismo.
- Guía el desarrollo y las pruebas: sirve como referencia para construir y validar el sistema.
- Documenta decisiones y funcionalidades: útil para mantenimiento, actualizaciones o nuevas versiones.

A continuación, se visualiza la estructura de una especificación de caso de uso y se explica cada punto:

ID y Nombre: Es un identificador único y un nombre descriptivo para el caso de uso. • El ID suele tener una codificación estándar (ej.: CU008). • El nombre debe estar en gerundio (acción continua), indicando lo que hace el usuario o sistema (ej.: Seguir producto vendido). Este punto sirve para: Para facilitar la referencia rápida del caso de uso en documentación, reuniones y trazabilidad.
Descripción: Un resumen breve pero claro de lo que trata el caso de uso. Debe describir el propósito principal de la funcionalidad desde el punto de vista del usuario. Ejemplo: "Permitir al vendedor visualizar el estado del envío de un producto previamente vendido y, si corresponde, iniciar la cancelación del mismo."
Actor Principal: El rol o usuario que inicia la interacción con el sistema. Se menciona el nombre del actor según el modelo (por ejemplo: Cliente, Administrador, Vendedor). Se utiliza para identificar quién es el principal beneficiario o ejecutor de la funcionalidad.
Actor Secundario: Otro sistema o actor que interviene indirectamente o colabora en el caso de uso. Un sistema de pagos, una API de envíos, o un Encargado que valida una acción.

Precondición: Son las condiciones que deben cumplirse antes de iniciar el caso de uso. Se describe un estado necesario del sistema o datos ya existentes. Ejemplo: "El cliente debe estar registrado en el sistema", o "El producto debe haber sido vendido".
Puntos de extensión: Hace referencia a otros casos de uso que se extienden desde este caso. Se citan los casos de uso relacionados mediante extend, que se activan bajo ciertas condiciones o acciones del usuario. Se puede extender a mas de un caso de uso. Ejemplo: CU009 – Cancelar Envío; CU-REC-40 – Modificar Cliente
Condiciones: Se descompone en dos subpartes importantes: el escenario principal y el flujo alternativo
Escenario principal: El flujo normal, paso a paso, que sigue el actor principal desde el inicio hasta el final del caso de uso. Con frases simples y numeradas o con letras. Cada línea representa una acción del usuario o una respuesta del sistema. Importante: Debe mantenerse coherencia cronológica y debe ser completo pero claro.
Flujo alternativo: Son caminos alternativos al flujo principal, que pueden surgir ante decisiones del usuario o errores del sistema. Se identifican con etiquetas como F1, F2, etc. y explican qué ocurre en esos casos. Ejemplo: • F1: Si el cliente no tiene saldo suficiente, el sistema muestra un mensaje de error. • F2: Si el producto no tiene stock, se informa al cliente.
Postcondiciones: Es el resultado final esperado luego de ejecutar el caso de uso, si se completa correctamente. Describe el nuevo estado del sistema o de los datos. Ejemplo: "El pedido fue registrado exitosamente en la base de datos", o "El estado del producto fue actualizado".

Como ejemplo podemos observar la siguiente especificación de Caso de Uso:

ID y Nombre: CU015 - Comprar Producto
Descripción: Permitir que el usuario registrado pueda comprar un producto disponible en el catálogo. La acción incluye el pago del producto y la verificación del stock. En algunos casos, se extiende al seguimiento del producto vendido.
Actor Principal: Usuario Registrado
Actor Secundario: Sistema de pago, Sistema de inventario

Precondición: El usuario debe estar autenticado y haber seleccionado un producto del catálogo.
Puntos de extensión: CU016 - Seguir Producto Vendido
Condiciones: Se descompone en dos subpartes importantes: el escenario principal y el flujo alternativo
Escenario principal: a) El usuario accede a la opción “Comprar producto”. b) El sistema solicita confirmación de la compra. c) El sistema incluye el CU011 - Verificar Stock. d) El sistema muestra el resultado de la verificación de stock. e) El sistema solicita los datos de pago. f) El sistema incluye el CU009 - Pagar Producto. g) El sistema confirma la compra y emite una orden. h) El sistema extiende al CU010 - Seguir Producto Vendido.
Flujo alternativo: F1: Stock no disponible. Si en el paso c el sistema detecta que no hay stock, muestra un mensaje de error y cancela la compra. F2: Pago fallido. Si en el paso f el pago no se aprueba, el sistema notifica al usuario y no genera la orden de compra.
Postcondiciones: La orden de compra queda registrada en el sistema. Si el usuario lo desea, puede hacer seguimiento del envío del producto.

3.5. Ejemplo Práctico

Dado el siguiente enunciado podemos ver cómo se puede obtener un diagrama de Caso de Uso simple y una especificación de un caso de uso seleccionado:

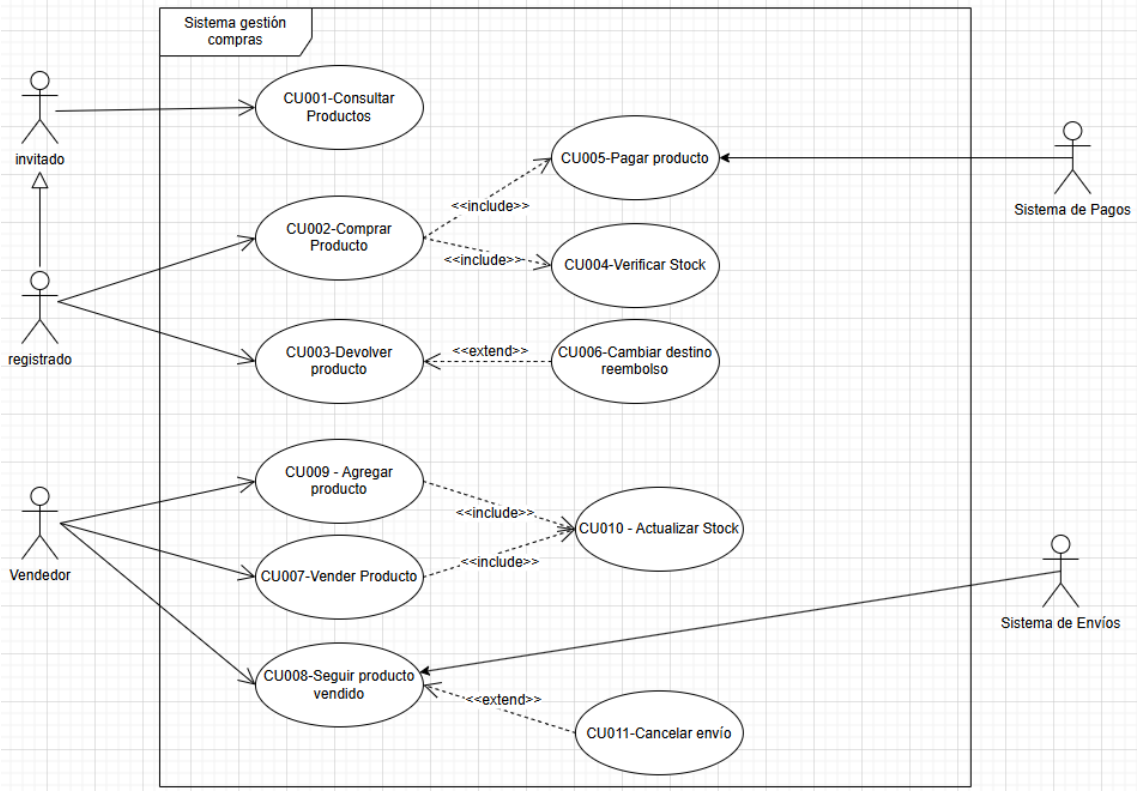
Una tienda en línea permite a los usuarios consultar y comprar productos. Cualquier persona puede ingresar al sitio como invitado para consultar productos disponibles. Para poder realizar una compra o devolución, el usuario debe estar registrado.

Un usuario registrado puede comprar productos, lo cual implica primero verificar el stock disponible y luego pagar el producto. Parte del proceso del pago es realizada en conjunto con el Sistema de Pago de la entidad bancaria. Si el producto no está en stock, el sistema no permite continuar la compra.

También, los usuarios registrados tienen la posibilidad de devolver un producto previamente comprado. En algunos casos, durante el proceso de devolución, el cliente puede optar por cambiar el destino del reembolso, por ejemplo, redirigir el monto a una cuenta distinta o solicitar un crédito interno en lugar de un reintegro directo. Esta acción no siempre ocurre, solo si el usuario lo solicita, por lo tanto, es un comportamiento opcional.

Por otro lado, los vendedores registrados tienen acceso a funciones administrativas. Pueden agregar nuevos productos al sistema, y al hacerlo, se debe actualizar el stock automáticamente para reflejar la cantidad inicial del producto ingresado. También pueden vender productos directamente a clientes mediante otras vías (como ventas presenciales), lo cual también incluye actualizar el stock.

Una vez realizada una venta, el vendedor puede seguir el estado de los productos vendidos, visualizando su estado de envío, una información que es provista por el Sistema de Envíos. En algunos casos excepcionales, si el cliente lo solicita o si hay un error, el vendedor puede cancelar el envío del producto, aunque esta opción solo está disponible bajo ciertas condiciones.



A continuación, tomamos 3 casos de uso y realizamos la especificación de los mismos.

ID y Nombre: CU002-Comprar Producto
Descripción: Permitir al usuario registrado realizar la compra de un producto, incluyendo la verificación de stock y el proceso de pago.
Actor Principal: Usuario Registrado
Actor Secundario: -
Precondición: El usuario debe estar autenticado en el sistema.
Puntos de extensión: -
Condiciones:
Escenario principal:

a) El usuario accede a la sección de productos. b) El sistema muestra el catálogo de productos disponibles. c) El usuario selecciona un producto y la cantidad deseada. d) El sistema verifica el stock disponible para el producto seleccionado. e) Si hay stock suficiente, el sistema permite continuar con la compra. f) El sistema incluye al CU-COM-02 – Pagar Producto. g) El sistema registra la compra realizada y genera una confirmación.
Flujo alternativo: F1: Si no hay stock suficiente, el sistema notifica al usuario que no puede continuar con la compra. F2: Si ocurre un error durante el proceso de pago, se muestra un mensaje de error y se permite reintentar.
Postcondiciones: La compra queda registrada en el sistema y el stock se actualiza.

ID y Nombre: CU005-Pagar Producto
Descripción: Permitir al usuario registrado realizar el pago del producto seleccionado mediante los medios habilitados.
Actor Principal: Usuario Registrado
Actor Secundario: Sistema de Pagos (solo a modo de ejemplo)
Precondición: El usuario debe estar autenticado en el sistema.
Puntos de extensión: -
Condiciones:
Escenario principal: a) El sistema muestra las opciones de medios de pago disponibles (tarjeta de crédito, débito, transferencia, etc.). b) El usuario selecciona un medio de pago y proporciona la información correspondiente. c) El sistema valida los datos ingresados. d) El sistema envía la solicitud de pago al sistema de pagos externo. e) El sistema recibe la confirmación del pago aprobado. f) El sistema muestra un comprobante de la transacción al usuario.
Flujo alternativo: F1: Si el sistema de pagos rechaza la transacción, se informa al usuario y se ofrece intentar con otro medio de pago. F2: Si se produce una interrupción en la conexión con el sistema de pagos, se permite reintentar o cancelar la operación.
Postcondiciones: El pago queda registrado y se asocia a la orden de compra correspondiente.

ID y Nombre: CU008-Seguir Producto Vendido
Descripción: Permitir al vendedor visualizar el estado del envío de un producto previamente vendido y, si corresponde, iniciar la cancelación del mismo.
Actor Principal: Vendedor
Actor Secundario: Sistema de Gestión de Envíos (solo a modo de ejemplo)

Precondición: El producto debe haber sido previamente vendido y registrado en el sistema.
Puntos de extensión: CU011-Cancelar Envío
Condiciones:
Escenario principal: a) El vendedor accede al menú de productos vendidos. b) El sistema muestra la lista de productos vendidos por ese usuario. c) El vendedor selecciona un producto de la lista para hacer el seguimiento. d) El sistema consulta el estado del envío en tiempo real. e) El sistema muestra el estado actual del envío (por ejemplo: en preparación, enviado, en tránsito, entregado). Si el estado del envío lo permite, el vendedor puede elegir cancelar el envío. f) El sistema extiende al CU009 – Cancelar Envío.
Flujo alternativo: F1: Si el envío ya fue entregado, el sistema notifica que no se puede cancelar. F2: Si hay un error al consultar el estado del envío, se muestra un mensaje de error y se ofrece reintentar.
Postcondiciones: El estado del envío fue consultado correctamente.

4. Diagrama de Clases (UML)

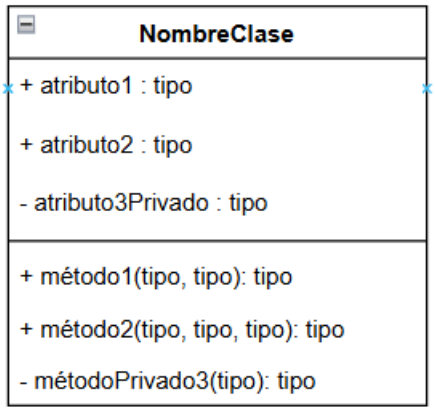
El **diagrama de clases** es una herramienta fundamental dentro del lenguaje de modelado **UML** (Unified Modeling Language). Representa la estructura estática de un sistema orientado a objetos, mostrando las clases, sus atributos, métodos, y las relaciones entre ellas.

Se utiliza en las fases de análisis y diseño del desarrollo de software, ya que permite representar visualmente la arquitectura del sistema desde una perspectiva lógica.

Los objetivos del diagrama de clases son:

- Describir las clases que conforman un sistema.
- Especificar los atributos (datos) y métodos (comportamientos) de cada clase.
- Representar las relaciones entre clases: asociaciones, herencia, agregaciones, dependencias, etc.
- Establecer la estructura básica del sistema antes de codificar.
- Servir como base para la implementación en un lenguaje de programación orientado a objetos.

4.1. Estructura de clase



Como se puede observar a la izquierda, tenemos la estructura de la clase donde en su encabezado colocamos el **nombre** de la misma. El nombre de la clase se coloca en negrita, debe comenzar en mayúscula y debe escribirse en **sustantivo singular**.

En la primera sección se encuentran todos los **atributos**, nombre y tipo, de dicha clase.

Luego, en la sección inferior se colocan los **métodos**, con los tipos de los parámetros y el tipo que retorna el propio método. En caso que el método reciba más de un parámetro se debe poner tantos "tipos" como parámetros reciba.

Tanto para los atributos como para los métodos se debe aclarar su **visibilidad**, es decir, si es público (+) o privado (-) antes del nombre de los mismos.

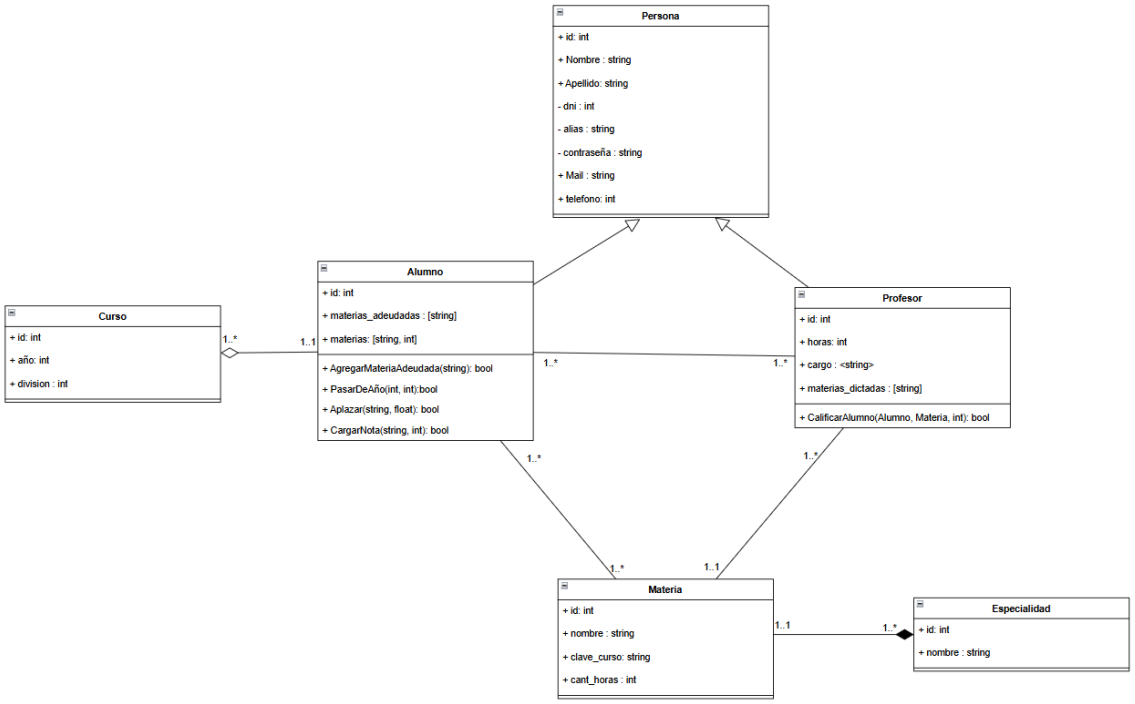
4.2. Relaciones entre clases

- Asociación: relación lógica o funcional entre dos clases. Se representa con una línea simple. Se representa: *Profesor* ——— *Alumno*
- Agregación: (todo-parte débil): contiene a las partes, pero pueden existir independientemente. Se representa: *Alumno* ——◇ *Curso*
- Composición: Similar a la agregación, pero más fuerte. El ciclo de vida de las partes depende del ciclo de vida del todo. Se representa: *Habitación* ——◆ *Casa*
- Herencia o generalización: Indica que una clase es una especialización de otra. La subclase hereda atributos y métodos de la superclase. Se representa: *Gato* ——▷ *Mascota*
- Dependencia: Indica que una clase usa o depende temporalmente de otra. Se representa: *Factura* - - - -> *Cliente*

4.3. Multiplicidad

- 1: Exactamente 1. De uno a uno
- 1.. *: De uno a muchos
- 0..1: De 0 a uno
- 0.. *: De 0 a muchos
- *: Muchos a muchos.

4.4. Aplicación práctica



5. Diccionario de datos

En el análisis de sistemas, el Diccionario de Datos es una herramienta fundamental que permite registrar, organizar y definir todos los elementos de datos utilizados dentro de un sistema.

Su objetivo principal es describir detalladamente la información que se manipula, para garantizar una comprensión común entre analistas, diseñadores, programadores y usuarios.

El Diccionario de Datos (DD) es un repositorio o catálogo estructurado de información sobre los datos del sistema.

Contiene definiciones, descripciones, estructuras, orígenes, usos, formatos y relaciones de todos los datos que se procesan. El objetivo de este es:

- **Unificar criterios:** asegura que todos los participantes del proyecto comprendan los datos del mismo modo.
- **Evitar ambigüedades y redundancias:** previene inconsistencias en los nombres o significados de los datos.
- **Documentar el sistema:** actúa como fuente de documentación técnica.
- **Facilitar el diseño lógico y físico:** sirve de base para el diseño de bases de datos o estructuras de archivos.

- **Controlar los cambios:** ayuda a gestionar modificaciones en los datos a lo largo del ciclo de vida del sistema.

Dentro del diccionario, los datos pueden clasificarse en distintos niveles:

- Datos elementales: son los más básicos (ejemplo: nombre, edad, precio).
- Datos compuestos: se forman a partir de varios datos elementales (ejemplo: dirección = calle + número + ciudad).
- Registros: conjunto de datos relacionados que representan una entidad (ejemplo: Cliente).
- Archivos o tablas: colección de registros del mismo tipo (ejemplo: Archivo de Clientes).

5.1. Contenido del Diccionario de Datos

Cada entrada del diccionario puede incluir los siguientes campos (según el nivel de detalle):

Elemento	**Descripción**
Nombre del dato	Identificador único (por ejemplo: "Código_Cliente")
Tipo de dato	Numérico, alfanumérico, fecha, lógico, etc.
Longitud en UI	Tamaño máximo del dato (por ejemplo: 10 caracteres)
Longitud en BD	Tamaño máximo del dato en la BD (por ejemplo: 50 caracteres)
Formato	Estructura que debe seguir el dato (ej: DD/MM/AAAA)
Valores	Los valores que puede poseer ese campo

5.2. Ejemplo Práctico

Nombre	Tipo	Long UI	Long BD	Formato	Valores
Código_cli	entéro	6	6	######	000001-999999
password	string	20	150	#...#	[a-z],[A-Z],[0-9][%,&,*..._]
Fecha_nac	date	10	10	dd/mm/aaaa	(01/01/1900-actualidad)

6. Diagrama de secuencia de sistema – DSS

El Diagrama de Secuencia es uno de los diagramas de interacción de la notación UML (Unified Modeling Language).

Su propósito es mostrar cómo interactúan los objetos o clases del sistema a lo largo del tiempo para llevar a cabo un determinado Caso de Uso o funcionalidad específica.

Representa la secuencia de mensajes que se envían entre los objetos, en qué orden, y qué actor o componente los inicia.

Permite visualizar y analizar:

- **El flujo dinámico de un proceso** dentro del sistema.
- **Cómo colaboran los objetos** entre sí para cumplir una tarea.
- **El orden cronológico** de los mensajes y llamadas.
- **El comportamiento interno** de un Caso de Uso o parte del sistema.

En síntesis, **muestra el "cómo" sucede algo en el sistema.**

6.1. Elementos principales

Elemento	**Descripción**
Actor	Representa a la persona, sistema o dispositivo externo que inicia la interacción. Se ubica a la izquierda del diagrama.
Objeto / Clase	Entidades internas del sistema que participan en la interacción. Se representan con un rectángulo con el nombre subrayado (ej: :Usuario, :ControladorLogin).
Línea de vida (Lifeline)	Línea vertical que desciende desde cada objeto. Representa la existencia del objeto durante la interacción.
Mensaje	Flecha horizontal que representa una comunicación o invocación de método entre objetos. Se lee de arriba hacia abajo, siguiendo el orden temporal.
Mensaje síncrono	Representa una llamada que **espera respuesta** antes de continuar (flecha con punta llena).
Mensaje asíncrono	Representa una llamada **que no espera respuesta** (flecha con punta abierta).
Respuesta / Retorno	Flecha punteada que muestra la respuesta de un mensaje.
Bloque de activación	Rectángulo fino sobre la línea de vida que muestra el tiempo durante el cual el objeto está activo (ejecutando una acción).
Condiciones / Bifurcaciones	Se pueden usar notas o fragmentos combinados (alt, opt, loop) para mostrar decisiones, alternativas o repeticiones.

También disponemos de elementos combinados como:

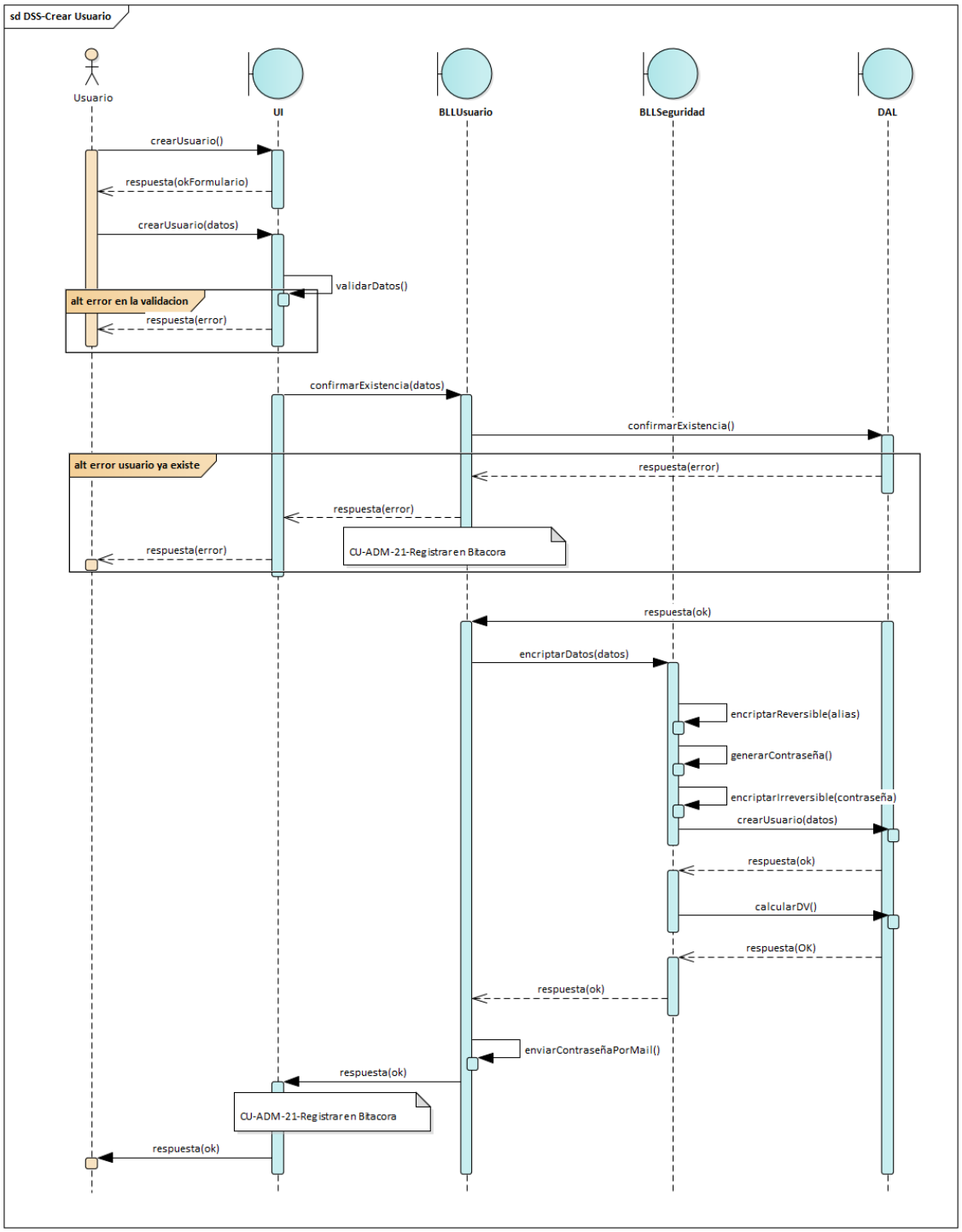
Fragmento	**Uso**	**Ejemplo**
alt	Alternativa (if/else)	Validar usuario correcto o incorrecto
opt	Opción (if con una sola rama)	Mostrar mensaje de error
loop	Repetición	Intentos de ingreso hasta 3 veces
par	Paralelismo	Procesos que ocurren al mismo tiempo

6.2. Ejemplo Práctico

Los dos siguientes ejemplos parten de un CU previo en la que podemos tener en cuenta sus pasos a nivel general.

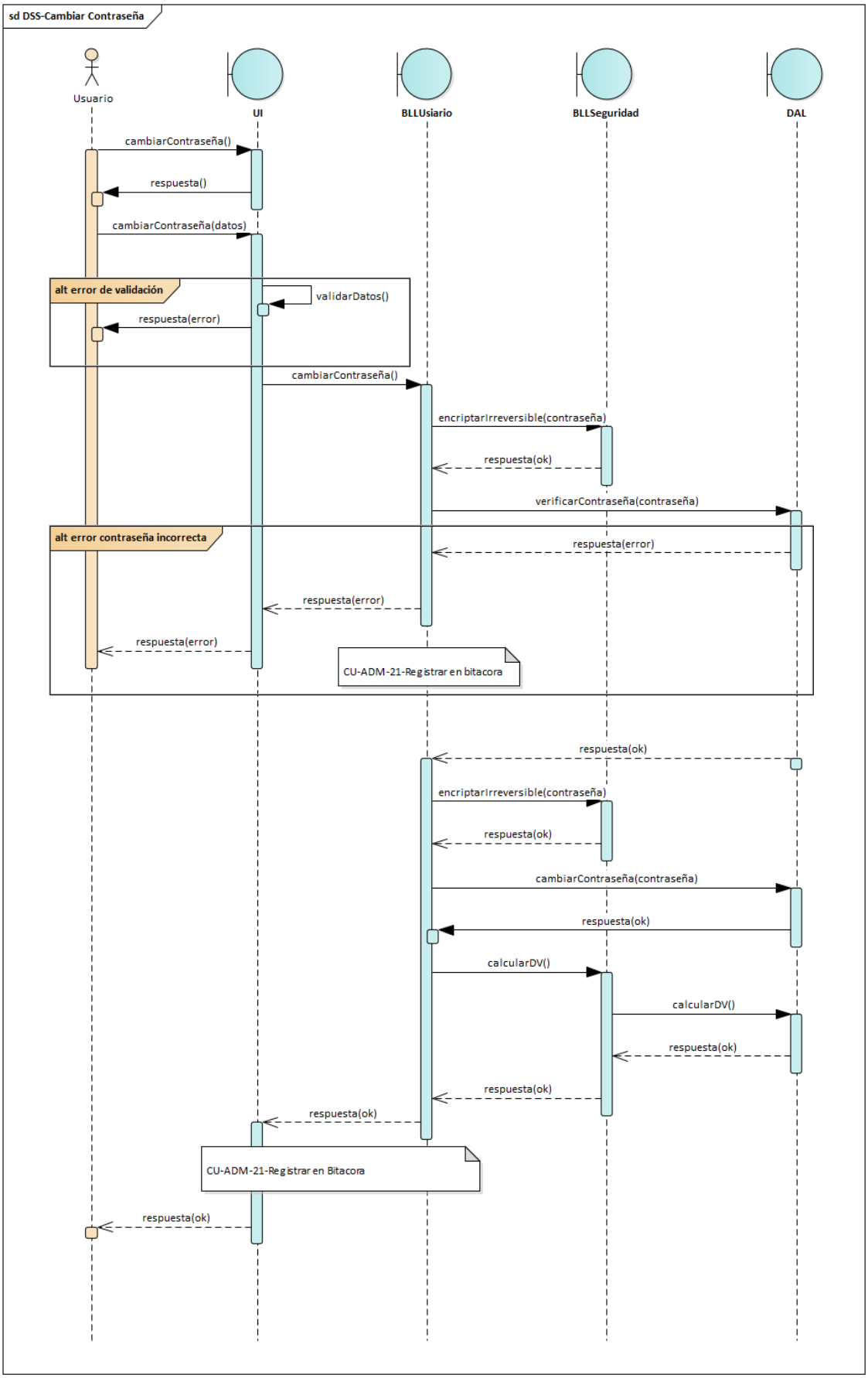
ID y Nombre: CU-ADM-10-Crear Nuevo Usuario
Descripción: Este caso de uso permite al administrador crear un nuevo usuario en el sistema.
Actor Principal: Administrador
Actor Secundario:
Precondición: El administrador debe estar autenticado y tener permisos para crear usuarios.
Puntos de extensión: -
Condiciones: -
Escenario principal: a) El administrador ingresa a la sección de creación de usuarios. b) El administrador llena el formulario de registro (nombre, apellido, correo, alias). c) El administrador confirma el ingreso de datos. d) El sistema verifica que todos los campos estén completos. e) El sistema asegura que el correo ni el alias no esté registrado en otro usuario. f) El sistema genera una contraseña y la envía por mail al usuario g) El sistema cifra la contraseña de manera segura con encriptación no reversible. h) El sistema guarda el nuevo usuario en la base de datos. i) El sistema calcula el DV y actualiza la tabla de Usuarios. j) El sistema registra la operación en la bitácora. k) El sistema informa que se creó correctamente el usuario.
Flujo alternativo: D1. El sistema, en caso de ingresar datos incorrectos, informa que los datos fueron ingresados de manera incorrecta para el usuario vuelva a ingresar.

E1. El sistema, en caso de encontrar registrado el mail o ya estar usado el alias, le informa al usuario.
Postcondiciones: Se registró un nuevo usuario en el sistema correctamente.



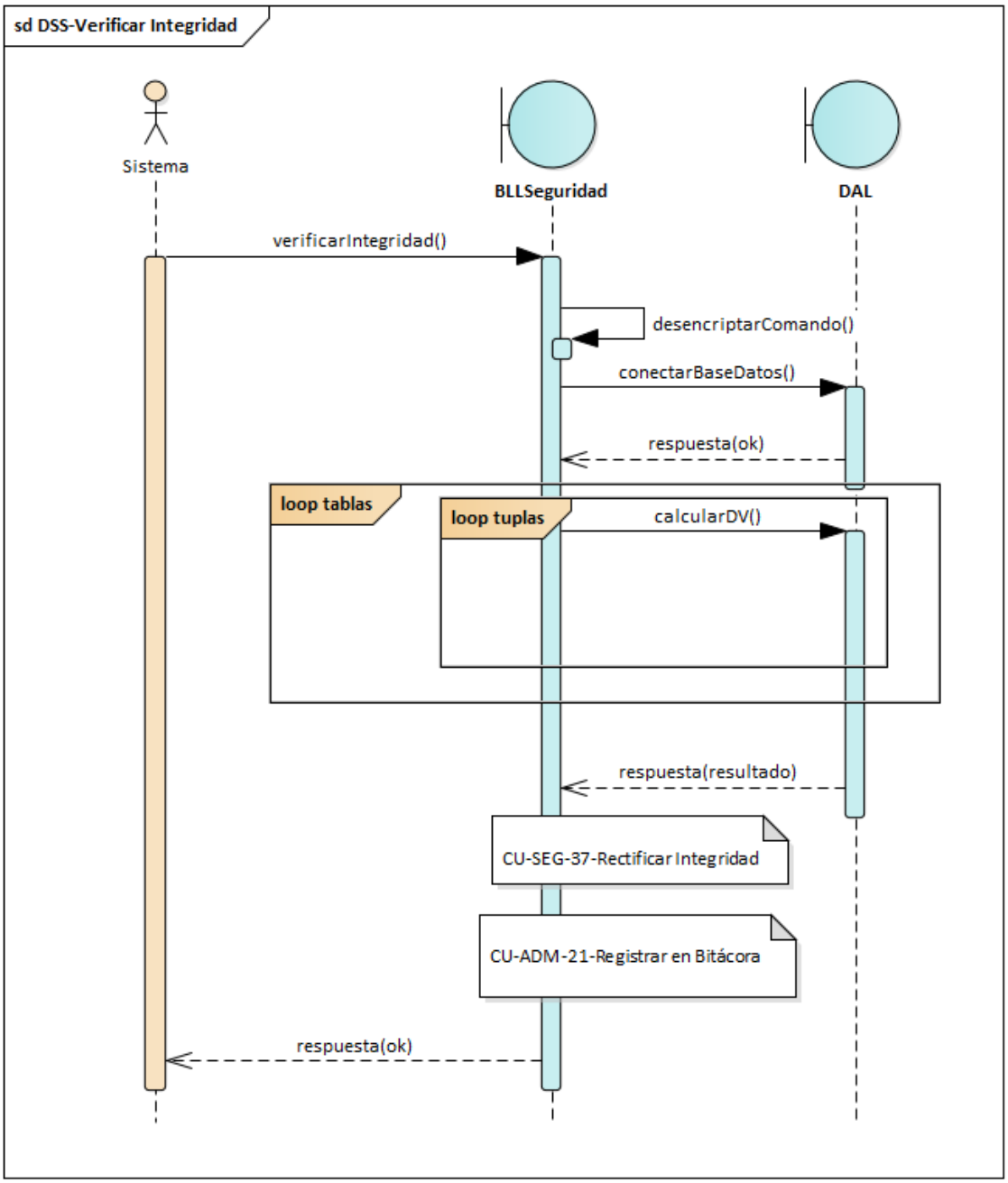
ID y Nombre: CU-USU-15-Cambiar Contraseña
Descripción: Permitir al usuario cambiar su contraseña de acceso al sistema.
Actor Principal: Usuario
Actor Secundario:
Precondición: El usuario selección la opción para cambiar Contraseña.

Puntos de extensión
Condiciones
Escenario principal: a) El usuario selecciona la opción "Cambiar Contraseña" desde el menú de configuración. b) El sistema muestra un formulario para cambiar la contraseña, incluyendo los campos "Contraseña Actual", "Nueva Contraseña" y "Confirmar Nueva Contraseña". c) El usuario completa el formulario con su contraseña actual y la nueva contraseña. d) El sistema valida que la contraseña actual ingresada sea correcta. e) El sistema valida que la nueva contraseña cumpla con las políticas de seguridad (longitud mínima, uso de caracteres especiales, etc.). f) El sistema valida que la nueva contraseña y la confirmación coincidan. g) El sistema Encripta la contraseña de manera irreversible. h) El sistema verifica que la contraseña actual coincida con la almacenada en la base de datos. i) El sistema actualiza la contraseña del usuario en la base de datos. j) El sistema recalcula el dígito verificador y actualiza la tabla de usuarios. k) El sistema registra el cambio de contraseña en la bitácora. l) El sistema muestra un mensaje confirmando que la contraseña se ha cambiado exitosamente.
Flujo alternativo: D1. Si los datos no están ingresados o no respetan las reglas definidas D2. Se vuelve al paso B. F1. Si las contraseñas nueva y nueva de confirmación no coinciden el sistema muestra un mensaje de error. F2. Se vuelve al paso B H1. Si la contraseña actual, al encriptarse reversiblemente, no se encuentra en la base de datos se emite un error. H2. Se registra en la bitácora el evento
Postcondiciones: La nueva contraseña del usuario se actualiza correctamente en la base de datos del sistema.



ID y Nombre: CU-SEG-38-Verificar Integridad

Descripción: El sistema verifica la integridad de la base de datos.
Actor Principal: Sistema
Actor Secundario:
Precondición:
Puntos de extensión: CU-SEG-37-Rectificar Integridad
Condiciones:
Escenario principal: a) El sistema accede al archivo que contiene el comando (cadena) de conexión. b) El sistema descifra el comando de conexión para obtener los datos de conexión. c) Se establece la conexión con la base de datos de manera exitosa. d) El sistema verifica los dígitos verificadores horizontales y luego los verticales. e) El sistema extiende al CU-SEG-37-Rectificar Integridad f) Se registra el evento en la bitácora. g) El sistema confirma la verificación
Flujo alternativo:
Postcondiciones: Se validó la integridad de la base de datos y se registró la transacción en la bitácora del sistema.


sd DSS-Verificar Integridad
Sistema
BLLSeguridad
DAL
verificarIntegridad()
desencriptarComando()
conectarBaseDatos()
respuesta(ok)
loop tablas
loop tuplas
calcularDV()
respuesta(resultado)
CU-SEG-37-Rectificar Integridad
CU-ADM-21-Registrar en Bitácora
respuesta(ok)