**UNIVERSIDAD ANDRES BELLO**



**"Proyecto Ingeniería de software II primer incremento para control de stock y cilindros para empresa Valgas e Hijo SPA"**

Profesor:

Paulo Luis Francisco Quinsacara Jofré

**Autores:**

Anakin Adolfo Benavides Romero

Tomas Andres Burgos Bottari

Diego Alonso Herrera Molina

Juan Pablo Olmedo Saavedra

Sebastián Ignacio Valdovinos Bravo

Cristóbal Alejandro Valenzuela Rojas

Tabla de contenido

1. Tabla de contenido

[2](#_heading=h.j5t9eia3hheb) Índice de Figuras 4

[3](#_heading=h.xbst0ljht5dn) Introducción 6

[4](#_heading=h.q9b6bcxdem0t) Propuesta de Solución e Incremento 6

[4.1](#_heading=h.sedrycy3tko9) Incremento 1 del Proyecto 7

[5](#_heading=h.dlmq7mvbbx57) Product Backlog 8

[6](#_heading=h.6o4a998if7ve) Sprint Backlog 9

[6.1](#_heading=h.7e5fkwvsgo8b) Casos de uso 9

[6.2](#_heading=h.evbjfw7umdnm) Planificación del Sprint 18

[7](#_heading=h.c0fun5mhcm7h) Metodología de desarrollo Scrum++ 19

[7.1](#_heading=h.ueav2sgodlq4) Vista lógica 19

[7.1.1](#_heading=h.bwljifghqmb4) Diagrama de clases 21

[7.1.2](#_heading=h.8l3vktunjbc) Modelo de Base de Datos 22

[7.2](#_heading=h.6g5lap2ch39u) Vista de Proceso 23

[7.2.1](#_heading=h.uybg3dirfymz) Diagrama de secuencias 24

[7.3](#_heading=h.rrfjqb12v0n) Vista de Desarrollo 33

[7.3.1](#_heading=h.udpdus4qcvcc) Diagrama de componentes 35

[7.4](#_heading=h.75wz0wpuc4kk) Vista Física 36

[7.4.1](#_heading=h.3qgom4cn0u4j) Diagrama de despliegue 37

[8](#_heading=h.9ow5tkwvwoiq) Árbol de Navegación 38

[9](#_heading=h.clesuyieu1um) Validación y evidencia de Casos de prueba 40

[9.1](#_heading=h.d7eq6v3rgbvv) Caso de Uso N°1 – Ingresando al sistema mediante Usuario y Contraseña 40

[9.2](#_heading=h.gykhzm1rr3z6) Caso de Uso N°2 – Definiendo de roles de usuario 42

[9.3](#_heading=h.5qykhronvi5o) Caso de Uso N°3 – Asignando permisos por módulo 44

[9.4](#_heading=h.ld3t9c39etbg) Caso de Uso N°4 – Recuperando la contraseña 45

[9.5](#_heading=h.tg4smpbk64xd) Caso de Uso N°5 – Cerrando automáticamente la sesión por inactividad 47

[9.6](#_heading=h.1703mw8182r5) Caso de Uso N°6 – Registrando cilindros nuevos 49

[9.7](#_heading=h.va7p0z7j0d46) Caso de Uso N°7 – Validando automáticamente los formatos al registrar cilindros 51

[9.8](#_heading=h.ssddb139v3mf) Caso de Uso N°8 – Modificando el estado del cilindro 52

[9.9](#_heading=h.vfm0okk733d0) Caso de Uso N°9 – Buscando cilindros por tipo 54

[9.10](#_heading=h.r7w64ait2a0f) Caso de Uso N°10 – Generando alerta de stock mínimo 56

[9.11](#_heading=h.mat3povt70vs) Caso de Uso N°11 – Registrando pérdidas o daños 56

[9.12](#_heading=h.d4uro0qc8i49) Caso de Uso N°12 – Generando reporte diario de stock 58

[9.13](#_heading=h.1cjjjg4g4l78) Caso de Uso N°13 – Consultando el historial de cambios en inventario 60

[10](#_heading=h.80ul57cdeool) Review Del Sprint 62

[11](#_heading=h.vhwkla6uecl4) Bibliografías 64

# Índice de Figuras

[**Índice de Figuras 3**](#_heading=h.ktdlodiss3j7)

[**1 Introducción 4**](#_heading=h.y07zaupfvwph)

[**2 Propuesta de Solución e Incremento 5**](#_heading=h.8h1agcjh0v45)

[2.1 Incremento 1 del Proyecto 6](#_heading=h.amf8l2arnxsq)

[**3 Product Backlog 7**](#_heading=h.fimq36dqfpv)

[**4 Sprint Backlog 8**](#_heading=h.yx0hflmpg9wh)

[4.1 Casos de uso 8](#_heading=h.lia91oup3bpr)

[4.2 Planificación del Sprint 17](#_heading=h.m9lhxb7pipe7)

[**5 Metodología de desarrollo Scrum++ 19**](#_heading=h.4uby5765ex9e)

[5.1 Vista lógica 19](#_heading=h.j1qj43p3s74v)

[5.1.1 Diagrama de clases 20](#_heading=h.fknqnruh7eyx)

[5.1.2 Modelo de Base de Datos 21](#_heading=h.h2e1jb4k441)

[5.2 Vista de Proceso 22](#_heading=h.r33xwxcmoqjl)

[5.2.1 Diagrama de secuencias 23](#_heading=h.emdj9paxvw2)

[**5.2.1.1 CU01 - Ingresando al sistema mediante usuario y contraseña: 23**](#_heading=)

[**5.2.1.2 CU02 - Definiendo los roles de usuario: 24**](#_heading=)

[**5.2.1.3 CU03 - Asignando permisos por módulo: 25**](#_heading=)

[**5.2.1.4 CU04.1 - Solicitando recuperación de contraseña: 26**](#_heading=)

[**5.2.1.4.1 CU04.2 – Restableciendo contraseña desde enlace 26**](#_heading=)

[**5.2.1.5 CU05 - Cerrando automáticamente la sesión por inactividad: 27**](#_heading=)

[**5.2.1.6 CU06 - Registrando cilindros nuevos: 28**](#_heading=)

[**5.2.1.7 CU07 - Validando automáticamente los formatos al registrar cilindros: 28**](#_heading=)

[**5.2.1.8 CU08 - Modificando el estado del cilindro: 29**](#_heading=)

[**5.2.1.9 CU09 - Buscando cilindros por tipo: 30**](#_heading=)

[**5.2.1.10 CU10 - Generando alerta de stock mínimo: 30**](#_heading=)

[**5.2.1.11 CU11 - Registrando pérdidas o daños: 31**](#_heading=)

[**5.2.1.12 CU12 - Generando reporte diario de stock: 32**](#_heading=)

[**5.2.1.13 CU13 - Consultando el historial de cambios en inventario: 32**](#_heading=)

[5.3 Vista de Desarrollo 33](#_heading=h.rcpvjgkmh89h)

[5.3.1 Diagrama de componentes 34](#_heading=h.ha5y5nf40fv9)

[5.4 Vista Física 35](#_heading=h.kduixpvm1d93)

[5.4.1 Diagrama de despliegue 36](#_heading=h.xi6gmpbme3fh)

[**6 Árbol de Navegación 37**](#_heading=h.8ewshq1i3x5h)

[**7 Validación y evidencia de Casos de prueba 39**](#_heading=h.6jwyy8czhmmc)

[7.1 Caso de Uso N°1 – Ingresando al sistema mediante Usuario y Contraseña 39](#_heading=h.tu9xabnn96d)

[7.2 Caso de Uso N°2 – Definiendo de roles de usuario 41](#_heading=h.qglhougcxn2t)

[7.3 Caso de Uso N°3 – Asignando permisos por módulo 43](#_heading=h.nymrfgfvn576)

[7.4 Caso de Uso N°4 – Recuperando la contraseña 44](#_heading=h.i9z6y5v37leq)

[7.5 Caso de Uso N°5 – Cerrando automáticamente la sesión por inactividad 46](#_heading=h.7oyoh412qhdw)

[7.6 Caso de Uso N°6 – Registrando cilindros nuevos 48](#_heading=h.7lj6kas4vbs)

[7.7 Caso de Uso N°7 – Validando automáticamente los formatos al registrar cilindros 50](#_heading=h.gqcql9coua90)

[7.8 Caso de Uso N°8 – Modificando el estado del cilindro 51](#_heading=h.ttc623l4u6ku)

[7.9 Caso de Uso N°9 – Buscando cilindros por tipo 53](#_heading=h.h98j3anmcf8a)

[7.10 Caso de Uso N°10 – Generando alerta de stock mínimo 55](#_heading=h.d5sv5c1xso95)

[7.11 Caso de Uso N°11 – Registrando pérdidas o daños 55](#_heading=h.1thbxwt9cchs)

[7.12 Caso de Uso N°12 – Generando reporte diario de stock 57](#_heading=h.vpq5e3p8o65z)

[7.13 Caso de Uso N°13 – Consultando el historial de cambios en inventario 59](#_heading=h.megbaq1ezjaw)

[**8 Review Del Sprint 61**](#_heading=)

[**9 Bibliografías 62**](#_heading=h.wwbl4dh9ouie)

# Introducción

El presente informe tiene como objetivo documentar el diseño, desarrollo y despliegue del sistema de gestión para la empresa **Valgas e Hijo SpA**, orientado al control de inventario, registro de ventas y administración de cilindros de gas. Esta solución tecnológica busca optimizar los procesos internos de la empresa, mejorar la trazabilidad de los productos y facilitar la toma de decisiones mediante reportes automatizados y visualizaciones en tiempo real.

Para estructurar el desarrollo de manera clara y modular, se ha utilizado la **metodología 4+1**, la cual permite representar el sistema desde distintas perspectivas arquitectónicas: la vista lógica, de proceso, de desarrollo, física y externa. Cada una de estas vistas describe aspectos fundamentales del sistema, permitiendo una comprensión integral tanto para desarrolladores como para partes interesadas.

Asimismo, se integró el enfoque ágil **Scrum++**, que combina los principios iterativos de Scrum con el modelo arquitectónico 4+1. Esta metodología permitió dividir el trabajo en **sprints independientes**, donde cada ciclo abordó un conjunto de casos de uso específicos (vista externa) y su correspondiente documentación técnica en las cuatro vistas restantes.

El documento incluye además los diagramas UML necesarios para visualizar el comportamiento y estructura del sistema, un árbol de navegación detallado, incorporando la interacción real con el sistema implementado.

# Propuesta de Solución e Incremento

La solución diseñada para Valgas e Hijo SpA consiste en una plataforma web centralizada que permite gestionar el ciclo completo de operación relacionado con cilindros de gas, desde su ingreso a inventario hasta su venta y trazabilidad, incluyendo control de usuarios, generación de reportes y alertas automatizadas.

El desarrollo del sistema se ejecuta bajo la metodología Scrum++, la cual combina el enfoque iterativo e incremental de Scrum con la estructura de documentación técnica del modelo 4+1. Esto garantiza entregas parciales funcionales acompañadas de sus respectivas vistas arquitectónicas.

## Incremento 1 del Proyecto

Este documento corresponde al **Incremento 1 del proyecto**, el cual contempla el diseño, implementación y documentación de las funcionalidades iniciales más críticas para la operación del sistema.

A continuación se presenta la tabla 2.1 la cual corresponde a una breve descripción del incremento 1

| **Elemento** | **Descripción** |
| --- | --- |
| **Incremento N°** | 1 |
| **Casos de uso abordados** | CU01 al CU13 |
| **Cantidad de casos de uso desarrollados** | 13 |
| **Total de casos de uso del sistema** | 63 |
| **Porcentaje parcial (sprint)** | **20.6%** (13 de 63) |
| **Porcentaje acumulado** | **20.6%** (primer incremento del sistema) |

*Tabla 2.1 Tabla de incremento 1*

# Product Backlog

El Product Backlog es una lista ordenada de los requerimientos funcionales del sistema, que representa todo lo necesario para construir y mantener el producto. Esta lista se prioriza según el valor que aporta al usuario y sirve como base para seleccionar los elementos a desarrollar en cada Sprint.

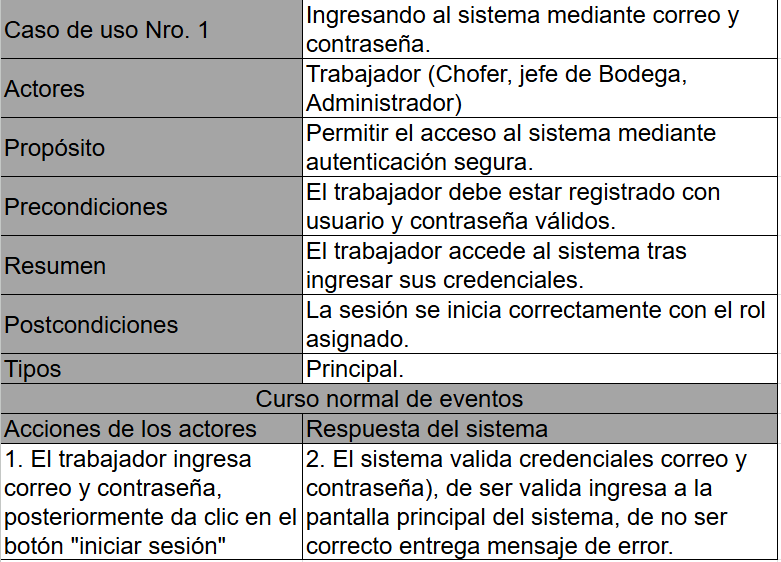
1. CU01 - 1.RF: Cada trabajador ingresará al sistema con un correo (formado por la primera letra del nombre + apellido + “.valgas@gmail.com”; en caso de duplicado, se añade la primera letra del segundo apellido. Ejemplo: Diego Herrera → dherrera.valgas@gmail.com) y una contraseña única.
2. CU02 - 2.RF: Los roles definidos en el sistema son: Chofer (consulta rutas y confirmación de entrega), Jefe de Bodega (gestiona inventario), y Administrador (acceso total).
3. CU03 - 3.RF: El administrador podrá asignar permisos (crear, leer, editar, eliminar) por módulo desde una interfaz gráfica.
4. CU04 - 4.RF: El sistema permitirá recuperar contraseñas mediante un enlace enviado al correo registrado del usuario (ej: botón “¿Olvidó su contraseña?”).
5. CU05 - 5.RF: Tras 9 minutos de inactividad, el sistema mostrará una advertencia. Al cumplir 10 minutos, se cerrará automáticamente la sesión.
6. CU06 - 6.RF: El sistema permitirá registrar cilindros nuevos con fecha (dd/mm/aaaa), tipo (5, 11, 15, 45 kg) y cantidad.
7. CU07 - 6.1.RNF: Se validará automáticamente los formatos ingresados, mostrando alertas en caso de error.
8. CU08 - 7.RF: Cada cilindro tendrá un estado (lleno/vacío) que podrá ser editado por el jefe de bodega.
9. CU09 - 8.RF: El sistema permitirá buscar cilindros filtrando por tipo y estado.
10. CU10 - 9.RF: Se generará una alerta automática si el stock de un tipo de cilindro es menor al mínimo definido.
11. CU11 - 10.RF: Se podrá registrar pérdidas o daños indicando motivo y fecha (dd/mm/aaaa).
12. CU12 - 11.RF: El sistema generará un informe diario de stock con cantidad, tipo y estado de los cilindros.
13. CU13 - 12.RF: Se podrá consultar el historial de cambios en inventario con usuario, fecha, hora y acción realizada.

# Sprint Backlog

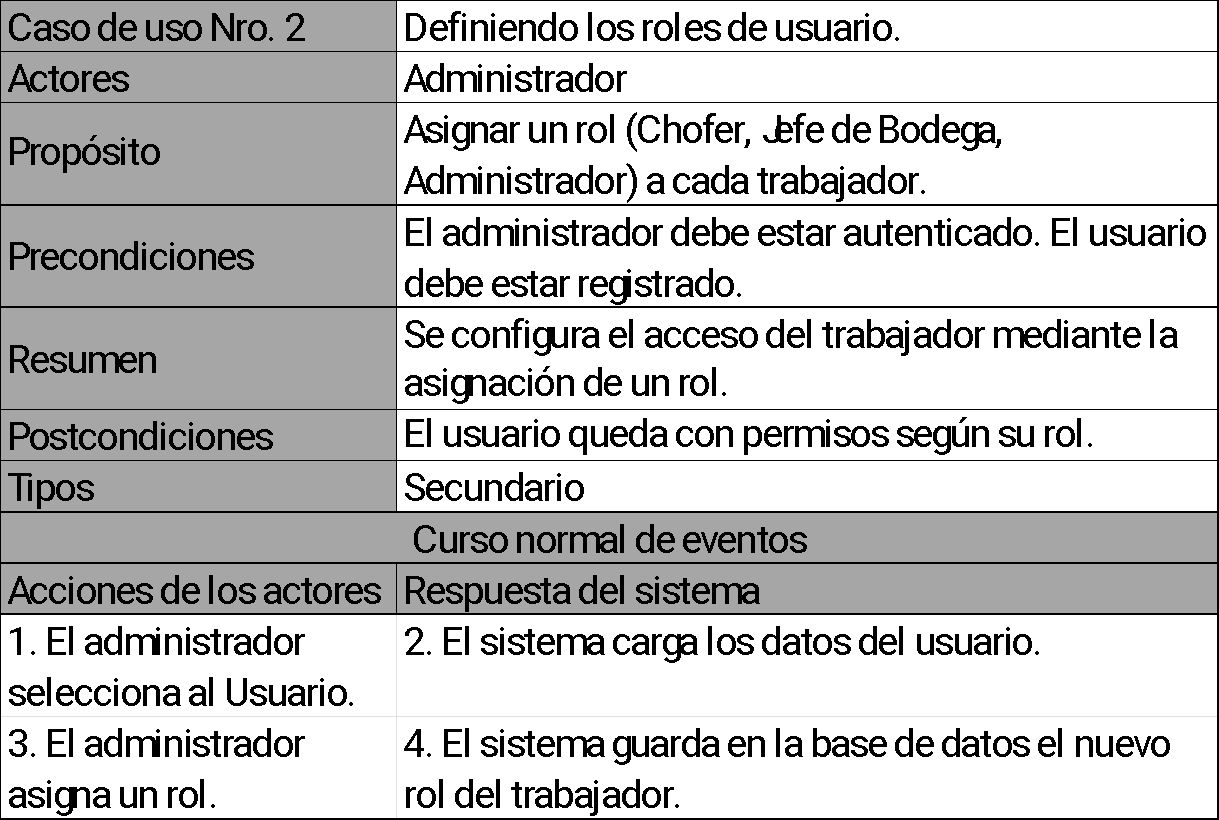
El Sprint Backlog reúne los requerimientos funcionales seleccionados del Product Backlog que serán desarrollados durante el Sprint actual. Estos se vinculan a casos de uso específicos que representan la vista externa del sistema, y permiten construir un incremento funcional y documentado conforme a la metodología 4+1.

## Casos de uso

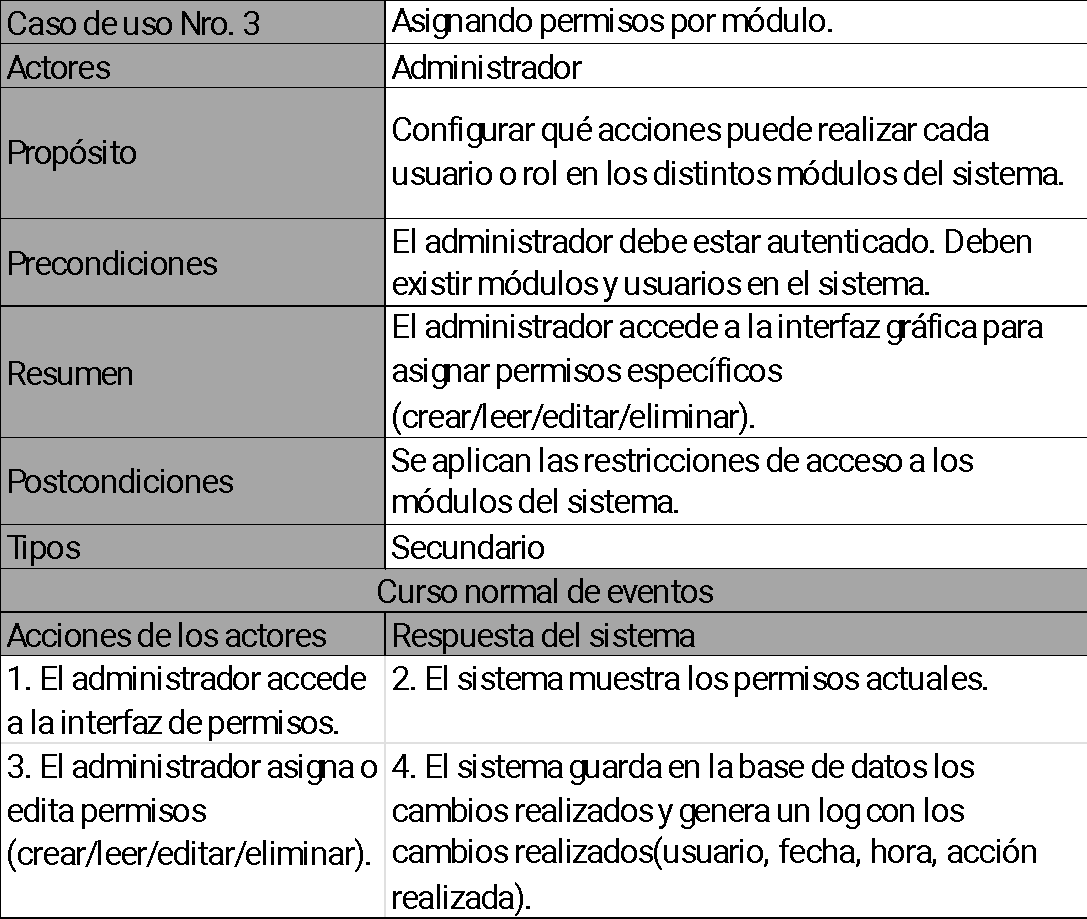
A continuación se muestran los casos de usos extendidos desarrollados en este primer incremento. Abarcando desde el CU1 hasta el CU13.



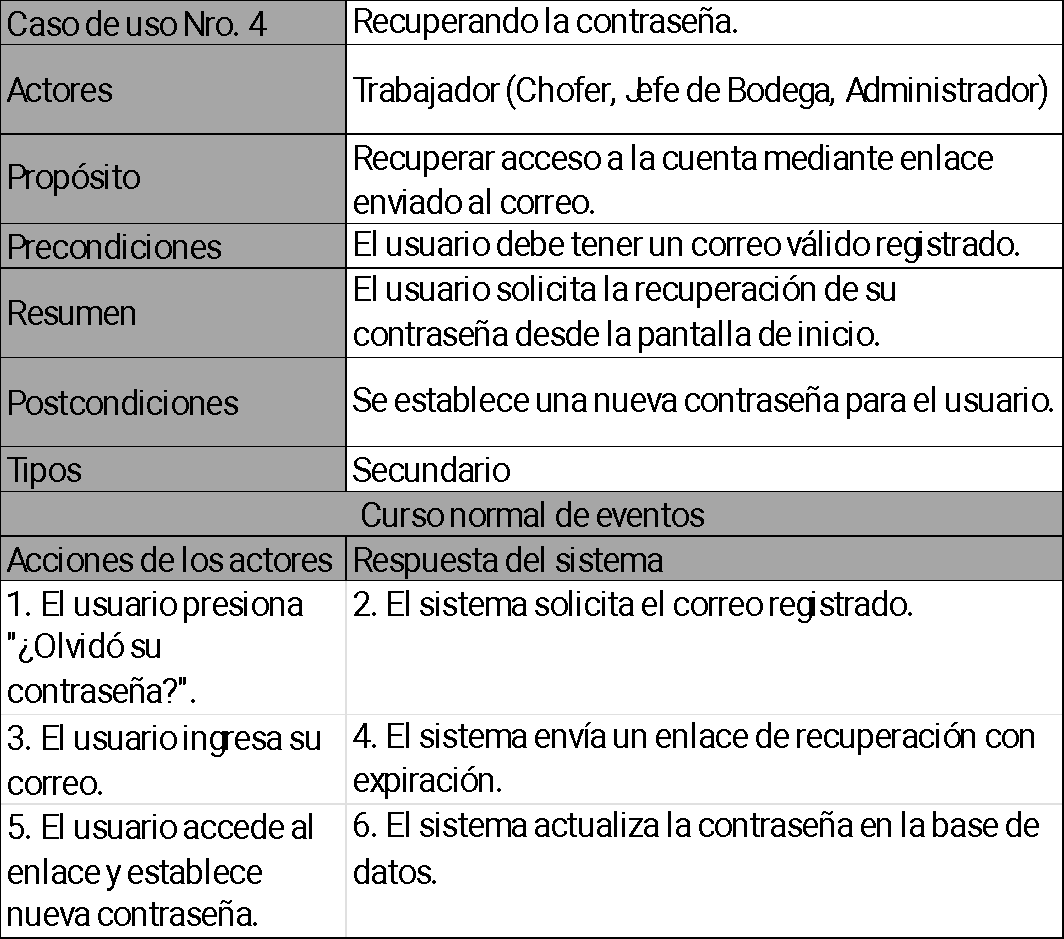
*Tabla 4.1 CU01*



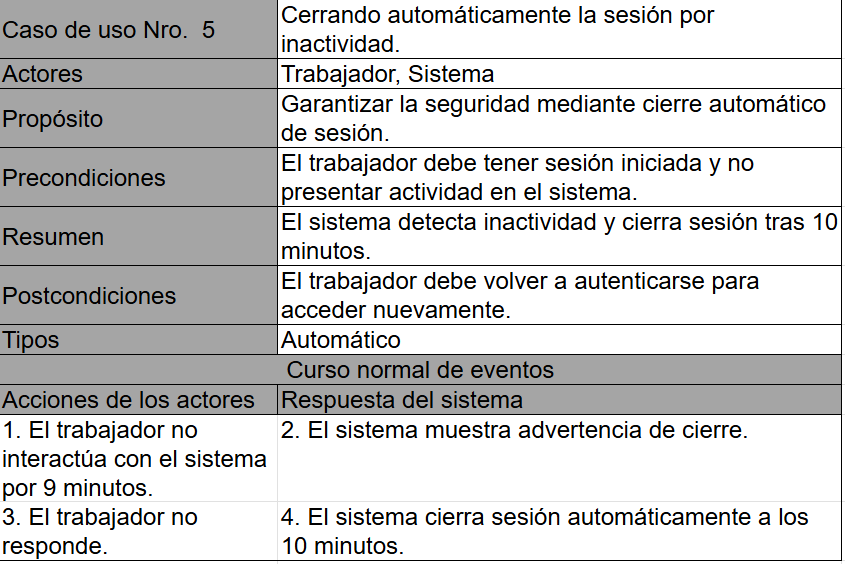
*Tabla 4.2 CU02*



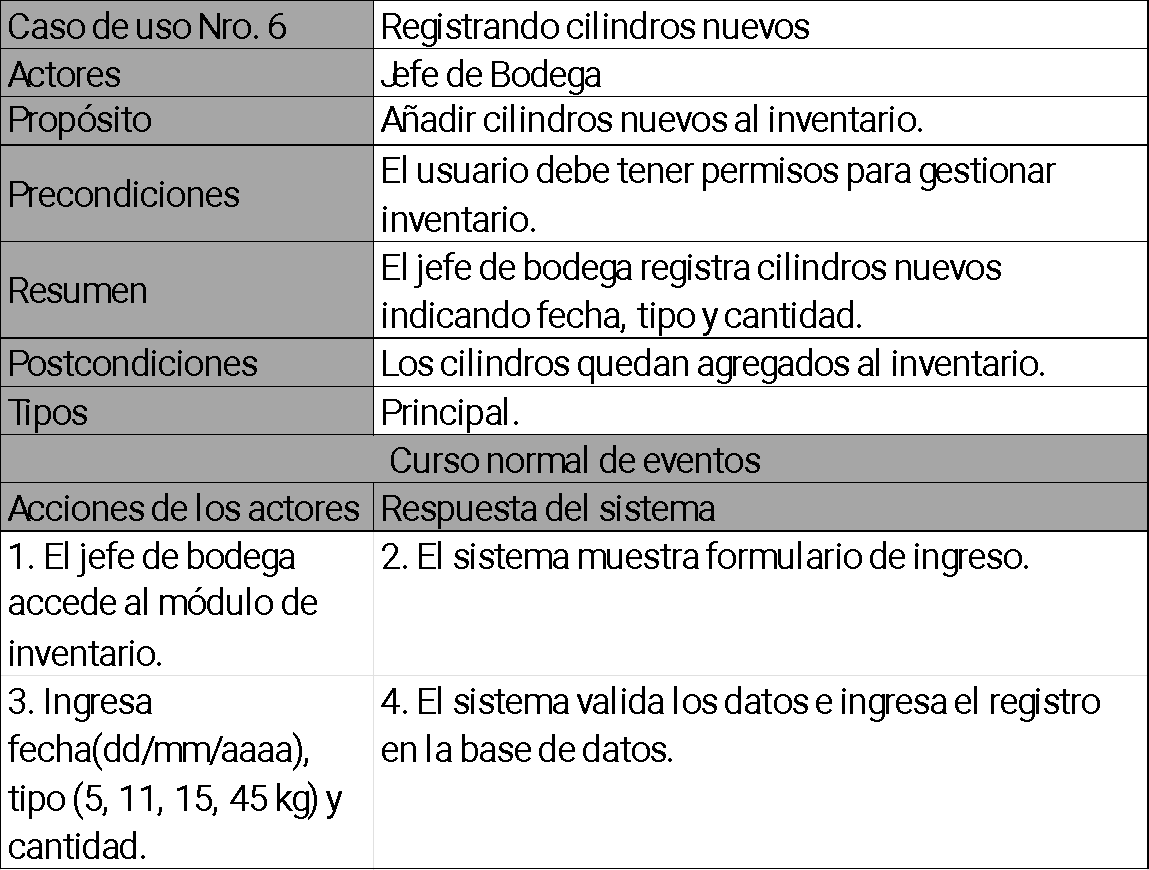
*Tabla 4.3 CU03*



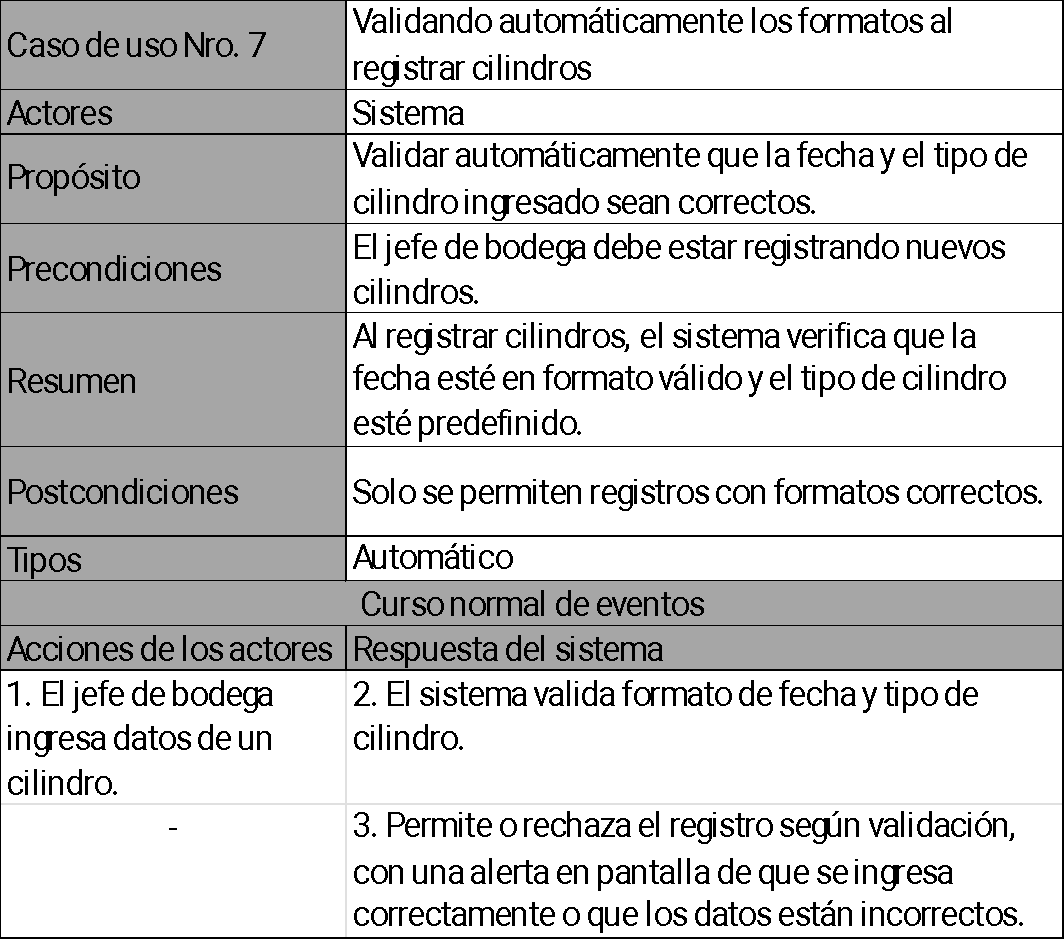
*Tabla 4.4 CU04*



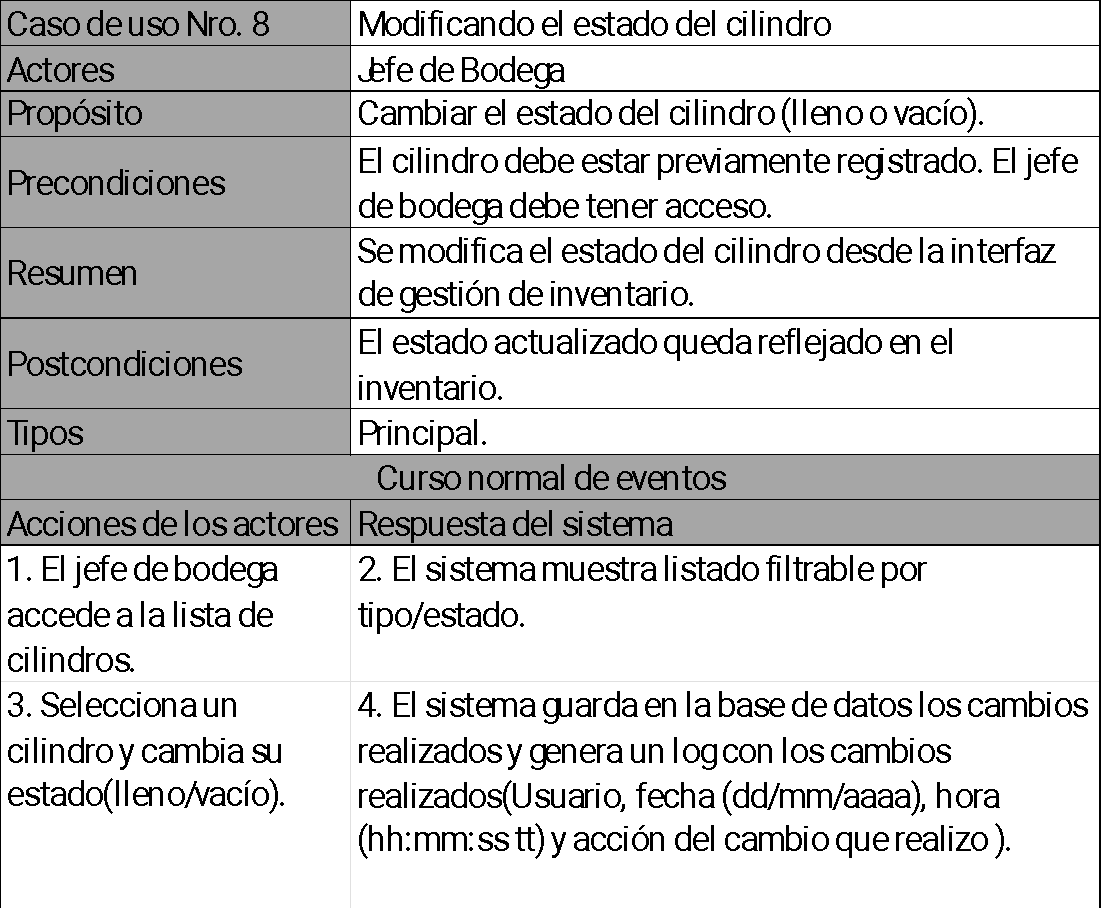
*Tabla 4.5 CU05*



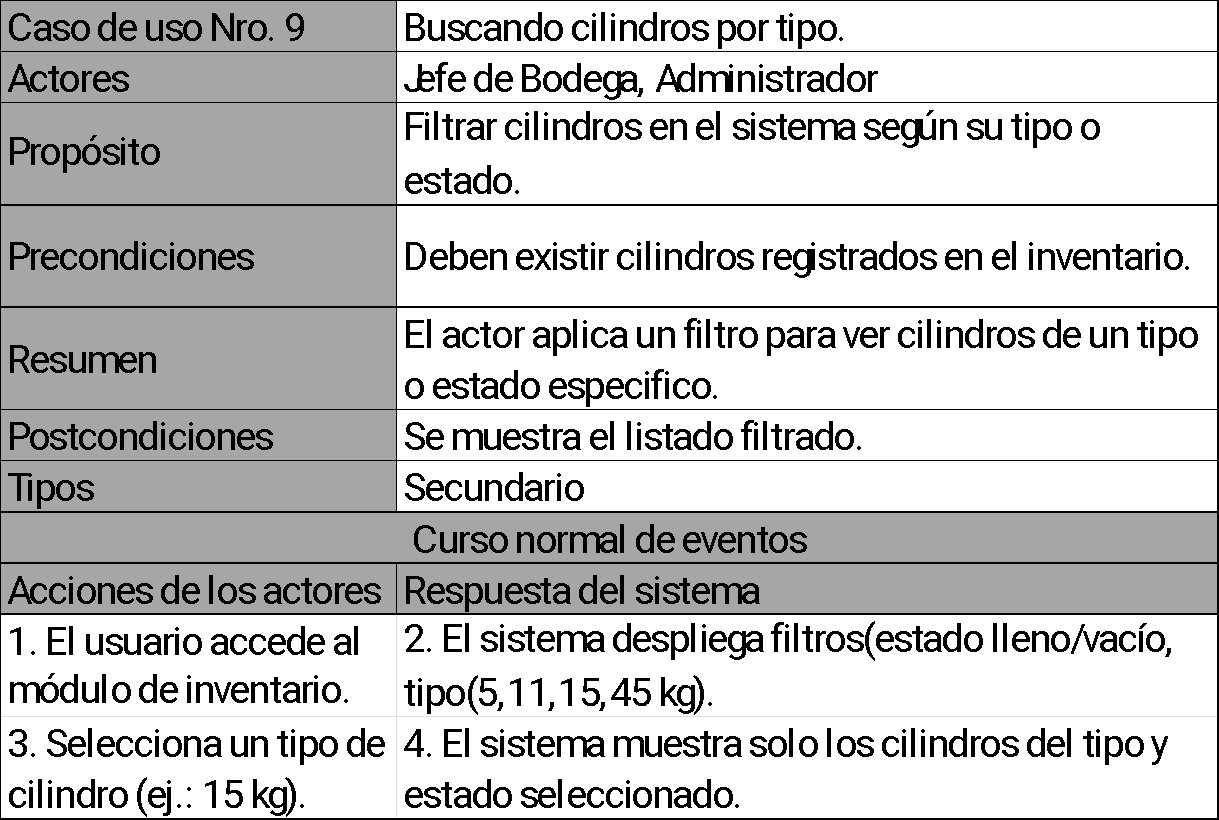
*Tabla 4.6 CU06*



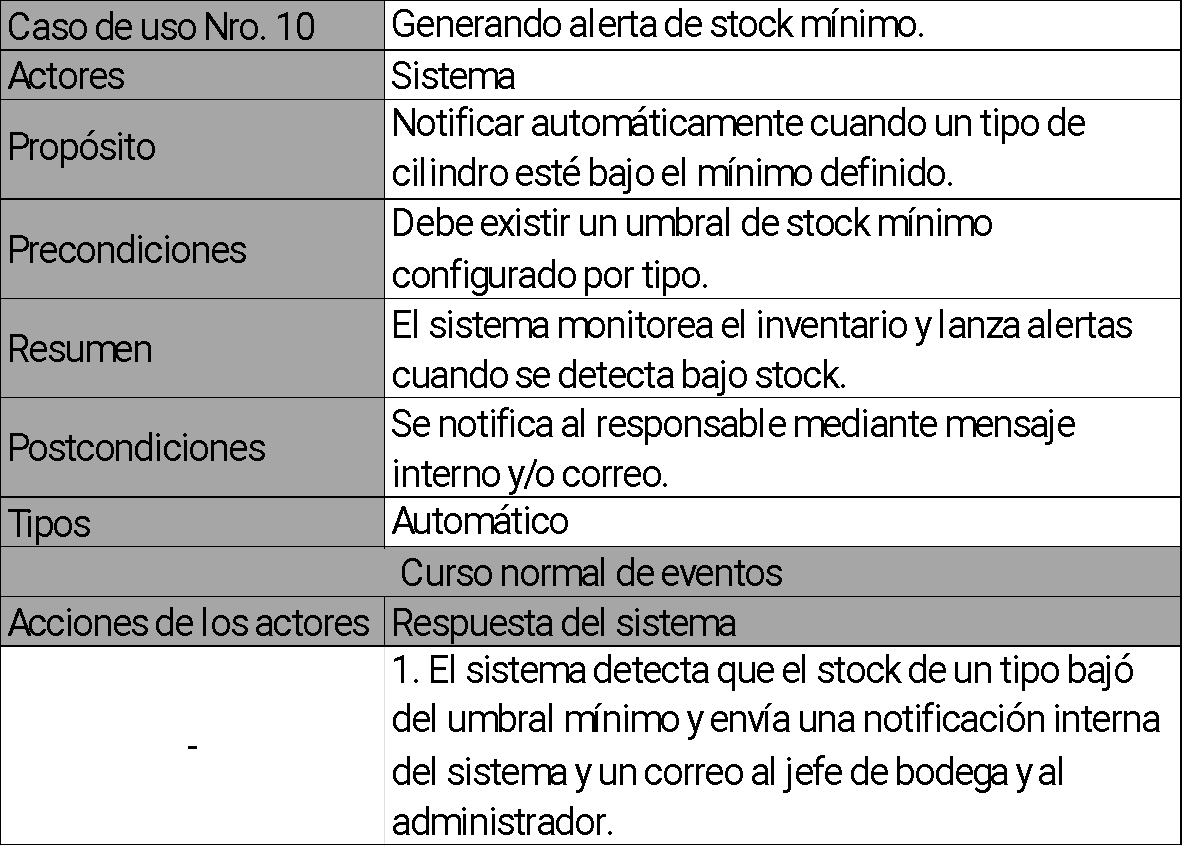
*Tabla 4.7 CU07*



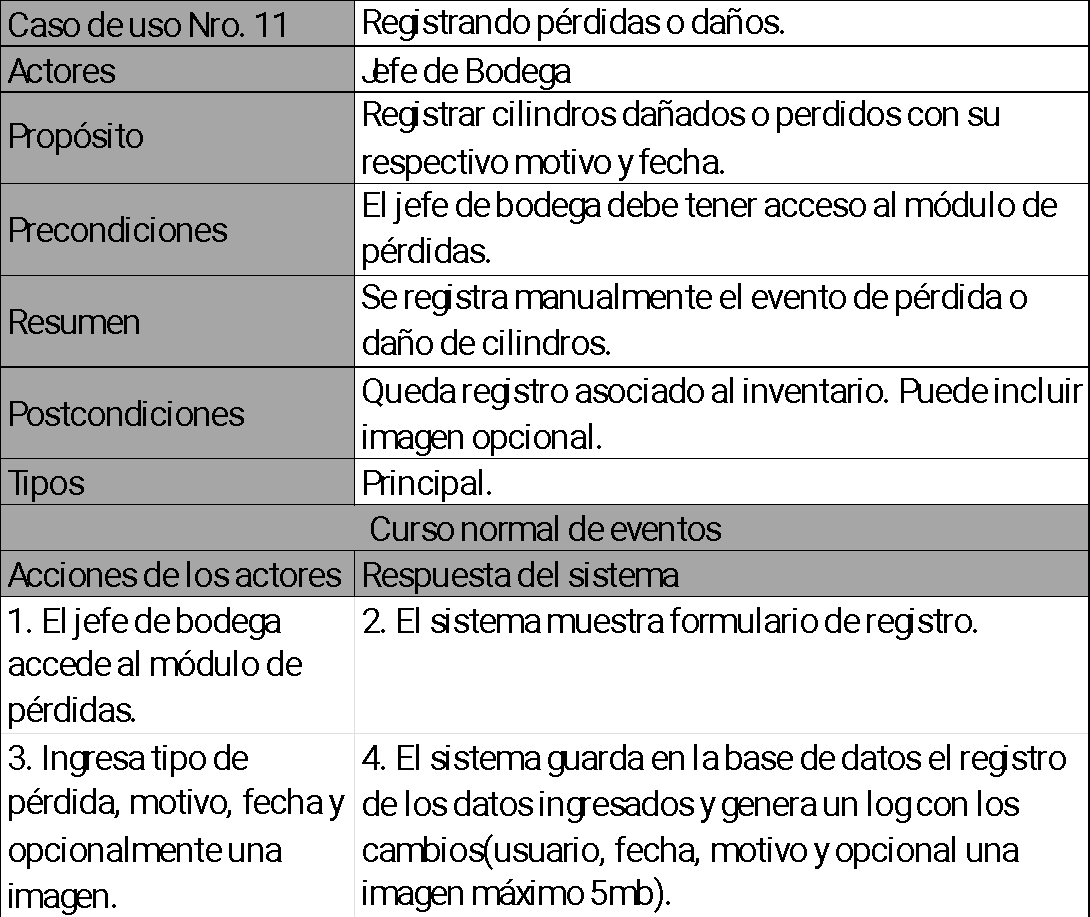
*Tabla 4.8 CU08*



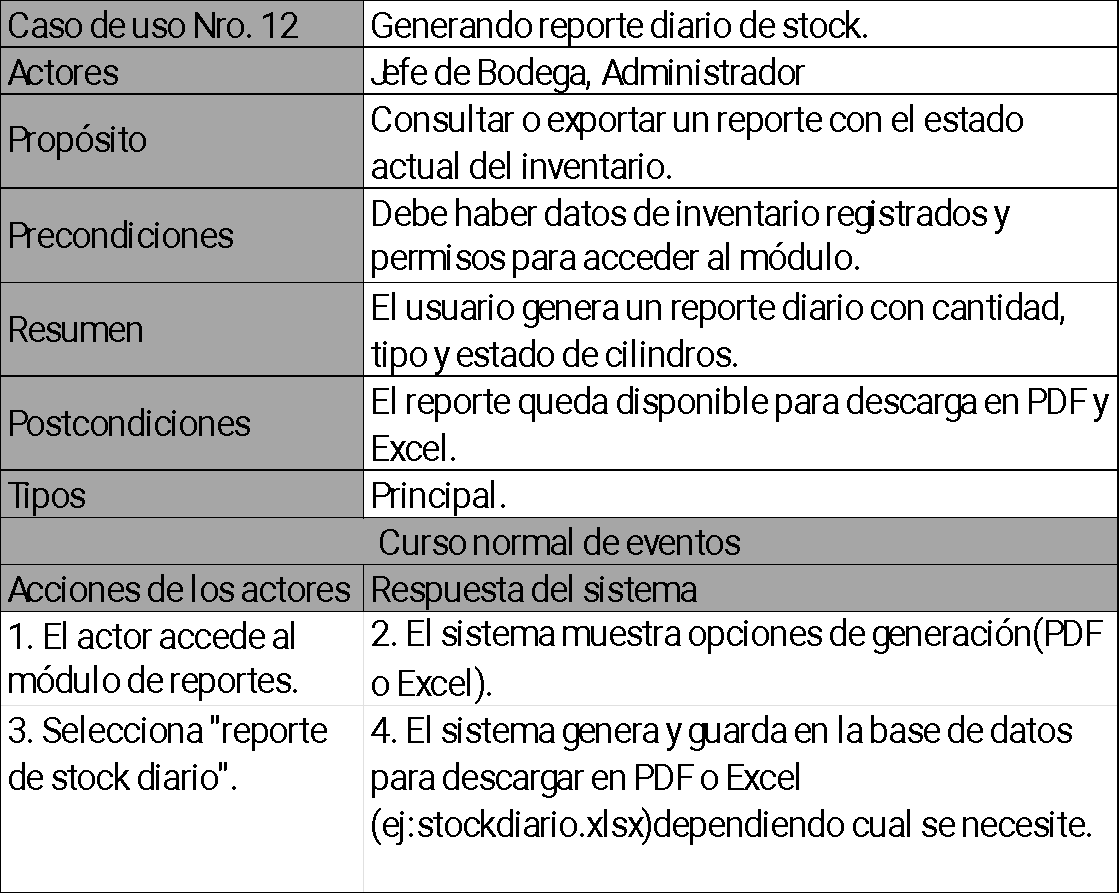
*Tabla 4.9 CU09*



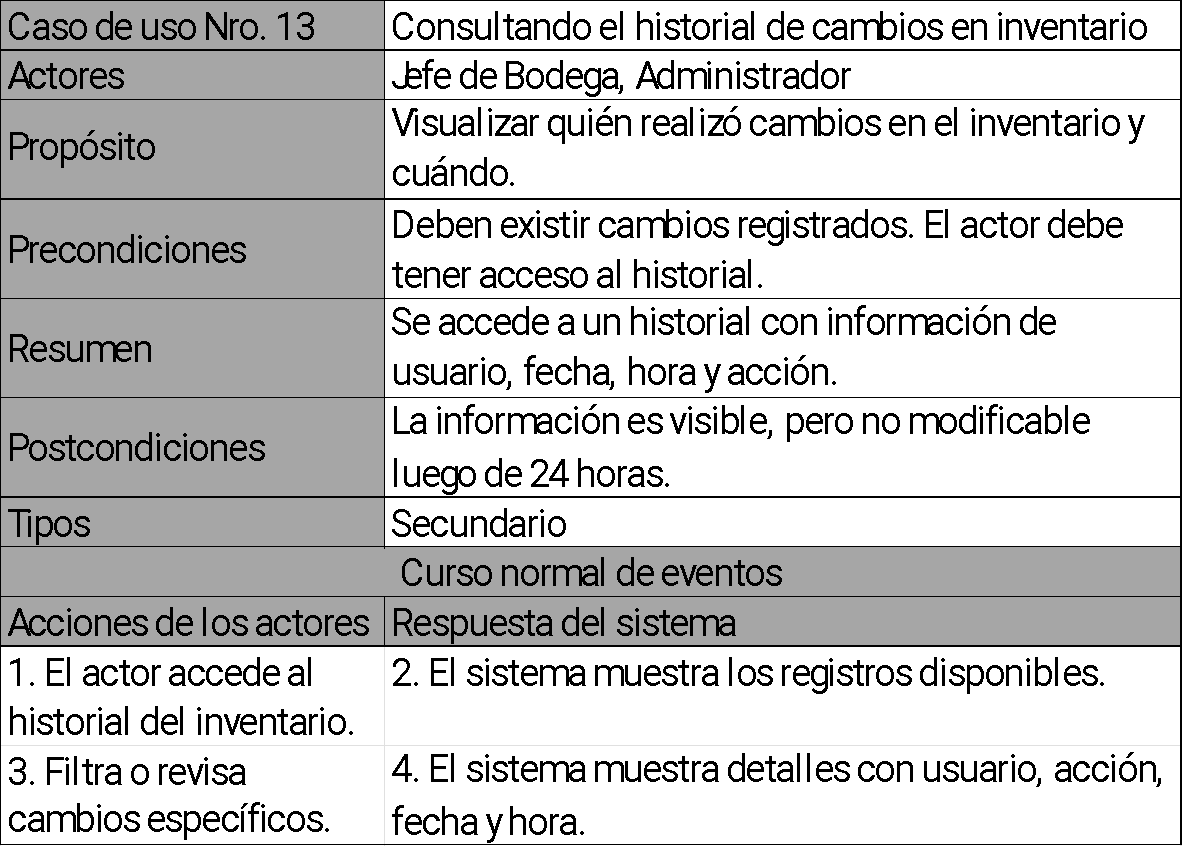
*Tabla 4.10 CU10*



*Tabla 4.11 CU11*



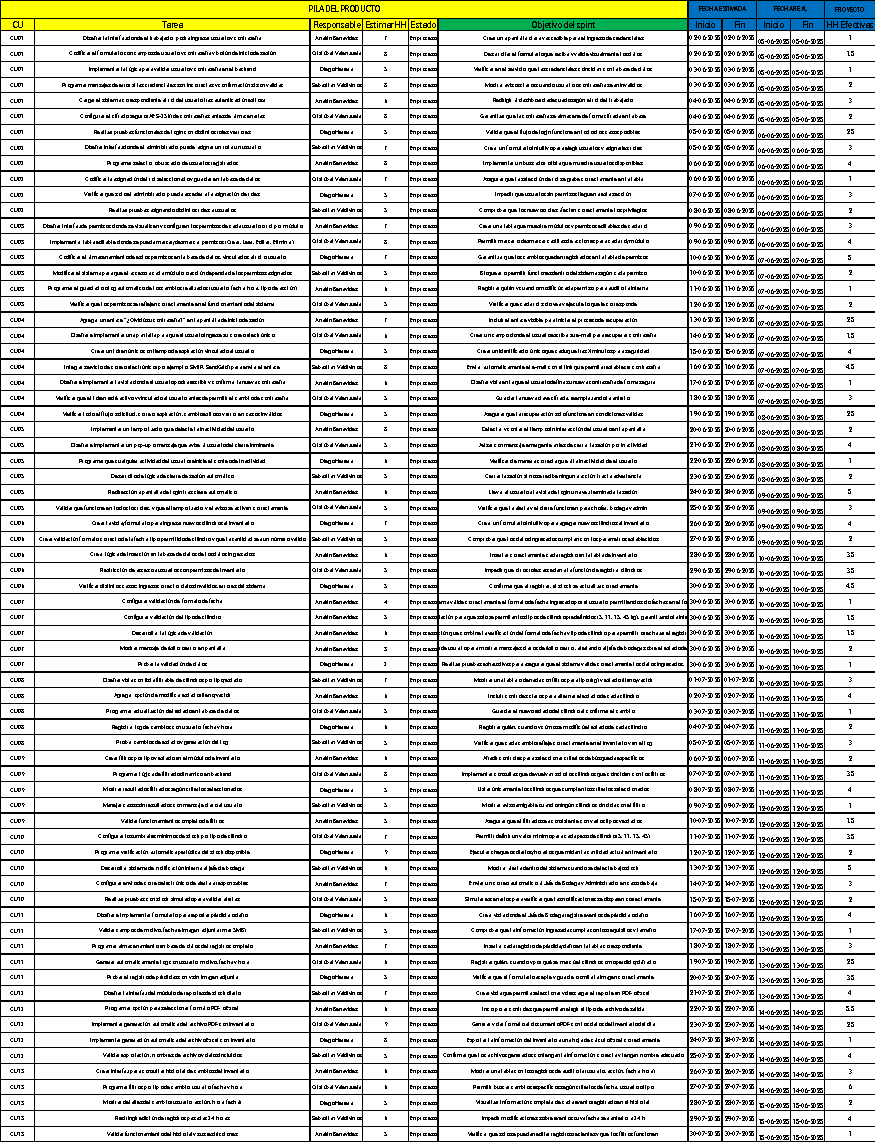
*Tabla 4.12 CU12*



*Tabla 4.13 CU13*

## Planificación del Sprint

A continuación se muestra la figura 4.14 la cual corresponde a la planificación del sprint



*Tabla 4.14 Planificación del sprint*

La imagen muestra la planificación del primer sprint, enmarcada dentro del enfoque ágil Scrum++. Esta planificación se representa mediante una tabla que detalla las tareas asociadas a los primeros 13 casos de uso seleccionados para el incremento 1 del proyecto.

Cada fila de la tabla corresponde a una tarea específica (derivada de los casos de uso), identificada con su código (CU), nombre de la tarea, responsable, esfuerzo estimado en horas-hombre (HH), y el estado actual de avance. Además, se incluye información del objetivo del sprint, fechas estimadas de inicio y término, tanto reales como planificadas, y el esfuerzo efectivamente utilizado.

Esta planificación permite organizar, visualizar y controlar el progreso de cada tarea durante la iteración, asegurando el cumplimiento de los entregables priorizados en la pila del producto (Product Backlog) y facilitando la coordinación del equipo de trabajo.

# Metodología de desarrollo Scrum++

## Vista lógica

La Vista Lógica corresponde a la representación estructural del sistema desde una perspectiva orientada a objetos. En la Figura se presenta el diagrama de clases del sistema de gestión de cilindros desarrollado para Valgas e Hijo SpA, el cual detalla las entidades clave, sus atributos, métodos y relaciones.

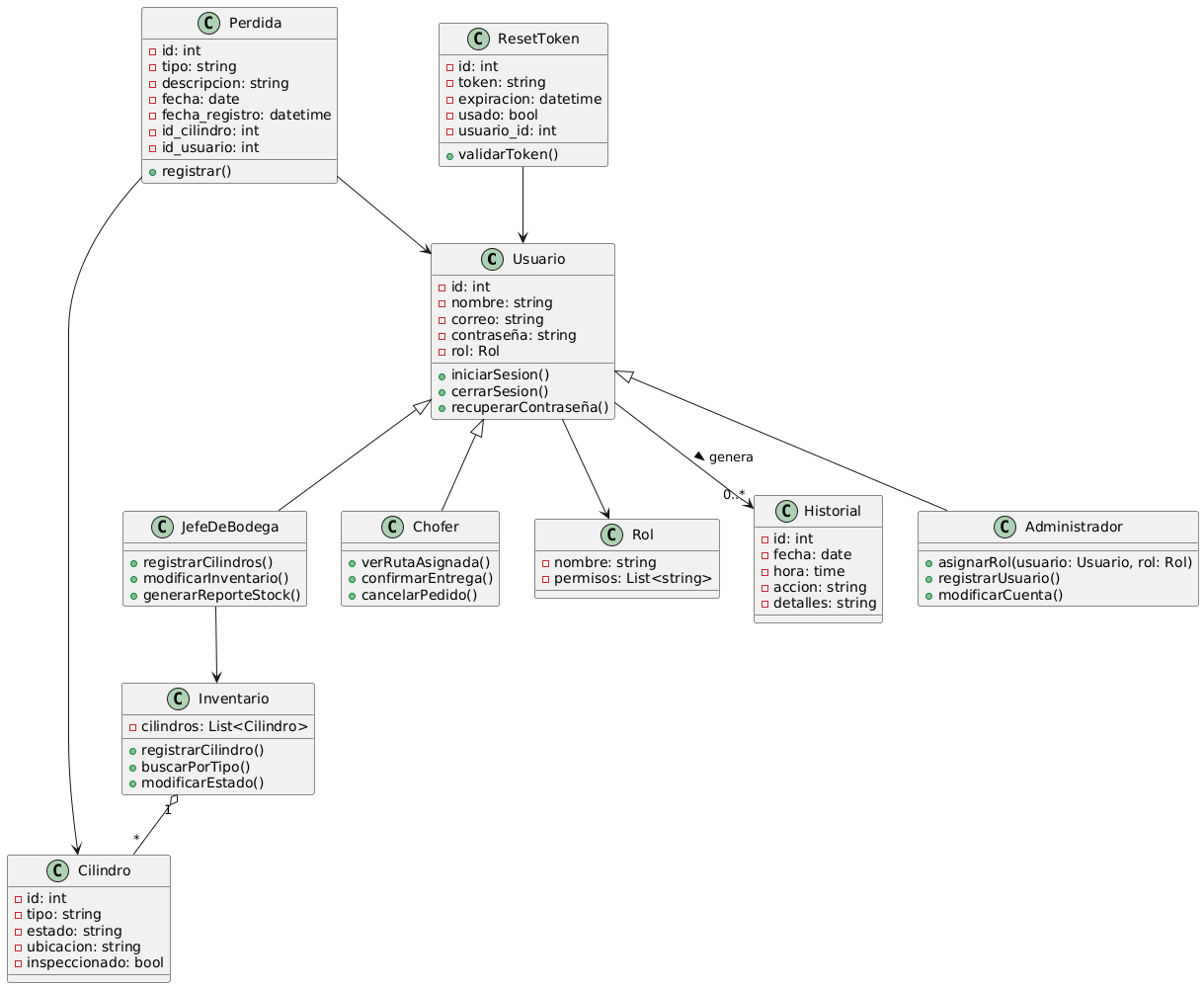
La clase central del sistema es Usuario, la cual se especializa en los tres roles principales: Administrador, Jefe De Bodega y Chofer. Cada uno posee operaciones específicas que reflejan sus responsabilidades dentro del sistema. Por ejemplo, el Administrador puede asignar roles y gestionar usuarios; el Jefe De Bodega administra el inventario, registra cilindros y genera reportes; mientras que el Chofer interactúa con las rutas y entregas asignadas.

El modelo incluye también entidades fundamentales como Rol, que define los permisos del usuario; Inventario, el cual contiene una lista de Cilindros registrados con atributos como tipo, estado, ubicación e inspección; y la clase Historial, que permite registrar todas las acciones ejecutadas en el sistema, garantizando la trazabilidad mediante fecha, hora, usuario y detalles de cada operación.

Otras clases relevantes incorporadas en este primer incremento son Pérdida, utilizada para documentar eventos de daño o pérdida de cilindros; y ResetToken, responsable del control de tokens temporales asociados a la recuperación de contraseñas de usuario.

### Diagrama de clases

A continuación se muestra la figura 5.1 que corresponde al diagrama de clases de la base de datos.



*Ilustración 5.1 Diagrama de Clases*

**Usuario y roles**Usuario representa a cualquier persona con acceso al sistema. Tiene atributos como id, nombre, correo, contraseña y un rol.

Se conecta con Rol, que define los permisos que tiene cada tipo de usuario.

Usuario es la clase base para Administrador, Jefe De Bodega y Chofer, los cuales heredan su comportamiento.

**Seguridad y autenticación**ResetToken: permite gestionar la recuperación de contraseñas. Contiene un token, fecha de expiración, estado de uso y una conexión con Usuario

**Historial**: registra cuándo y qué acción realizó un usuario.

**Gestión de cilindros e inventario  
Cilindro**: representa cada cilindro de gas con datos como tipo, estado, ubicación e inspección.

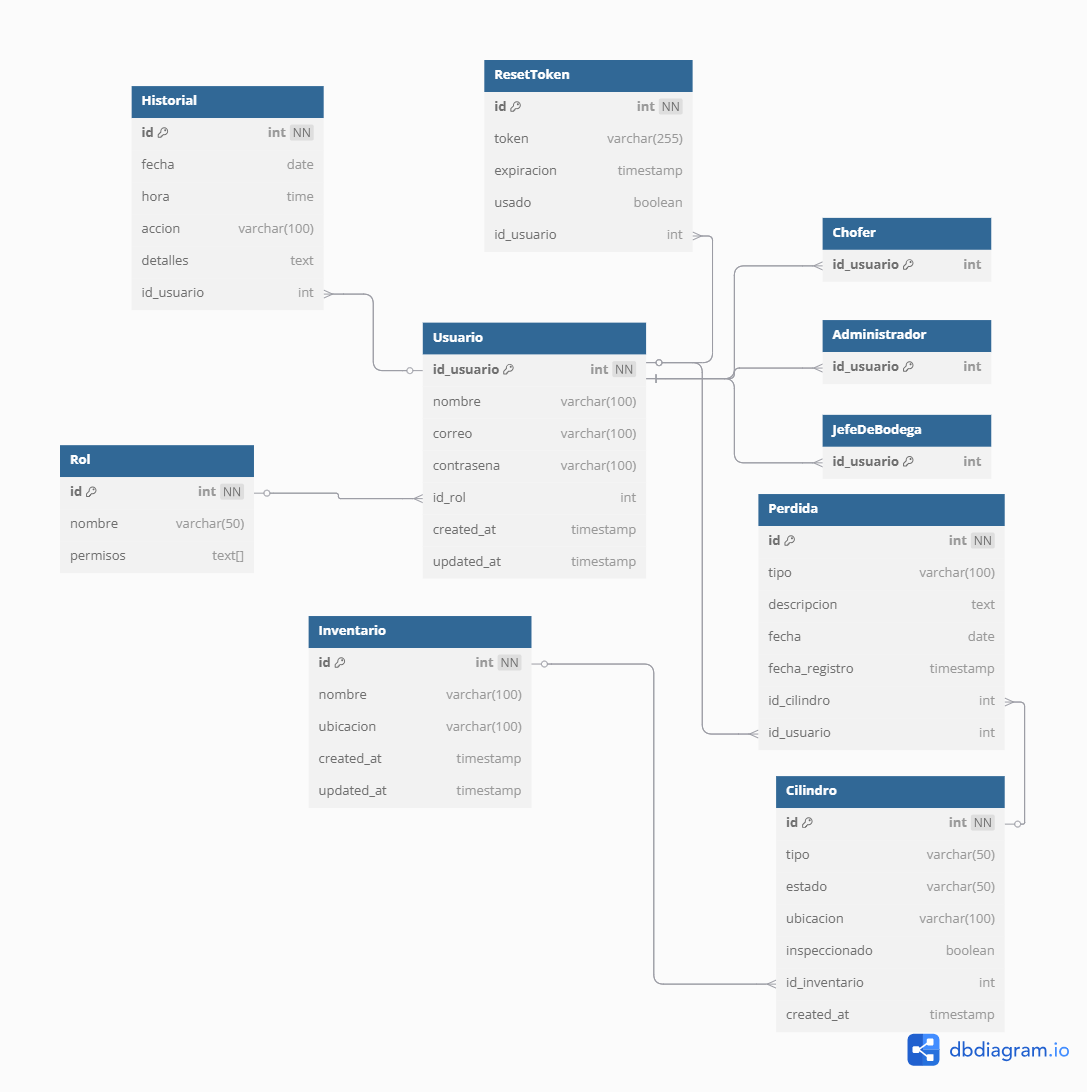
**Inventario**: agrupa cilindros y ofrece métodos para registrar, buscar y modificar su estado.

**Pérdida**: registra cilindros dañados o perdidos, con tipo, descripción, fecha, y se conecta tanto a Usuario (quien registra la pérdida) como a Cilindro (el objeto perdido).

**Jefe De Bodega**: es un tipo de usuario con permisos para gestionar inventario, generar reportes y registrar cilindros.

### Modelo de Base de Datos

A continuación, se presenta la Figura 5.2, correspondiente al diagrama entidad-relación de la base de datos.



*Ilustración 5.2 Modelo de base de datos*

**Usuarios y Roles**

La tabla Usuario almacena la información básica de los usuarios: nombre, correo, contraseña y su rol.

Cada usuario se vincula con un Rol, que define su nombre (por ejemplo, "Administrador") y sus permisos (como un arreglo de strings).

Existen subtablas específicas: Administrador, Chofer y JefeDeBodega, cada una relacionada 1:1 con Usuario si se desea extender su comportamiento.

**Seguridad**

ResetToken: almacena tokens temporales para recuperación de contraseña. Está ligado a un usuario y tiene fecha de expiración y estado (usado o no).

**Historial**

Registra acciones realizadas por los usuarios, como inicios de sesión o modificaciones, con fecha, hora y detalles.

**Inventario y Cilindros**

Inventario: representa un punto de almacenamiento (como una bodega), con nombre y ubicación.

**Cilindro**

Cada cilindro tiene un tipo, estado (por ejemplo, “lleno” o “vacío”), ubicación interna, y si fue inspeccionado. Está vinculado a un inventario.

**Gestión de Pérdidas**

Pérdida: registra eventos en que un cilindro fue dañado o extraviado. Incluye tipo, descripción, fecha y se relaciona tanto con el cilindro afectado como con el usuario que lo reportó.

## Vista de Proceso

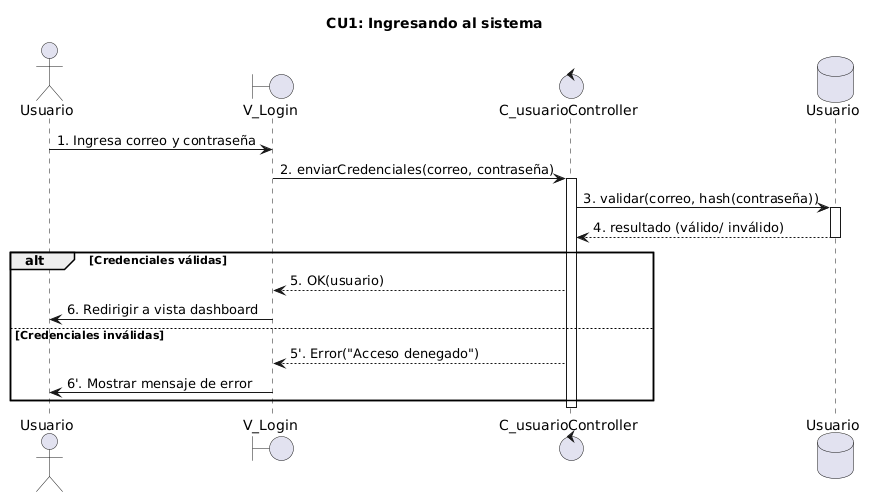
La Vista de Proceso forma parte de la arquitectura 4+1 y tiene como objetivo representar la interacción entre los distintos objetos del sistema en tiempo de ejecución. Esta vista se expresa a través de diagramas de secuencia, los cuales permiten visualizar el flujo de mensajes entre los actores, las interfaces gráficas (vistas), los controladores y la base de datos.

A continuación, se presentan los diagramas de secuencia correspondientes a los primeros casos de uso implementados en el Incremento 1. Cada diagrama refleja el comportamiento dinámico del sistema ante una funcionalidad específica, destacando el orden de las operaciones y la comunicación entre los distintos componentes de la arquitectura.

### Diagrama de secuencias

A continuación se muestra el diagrama de secuencias respecto a los casos de uso.

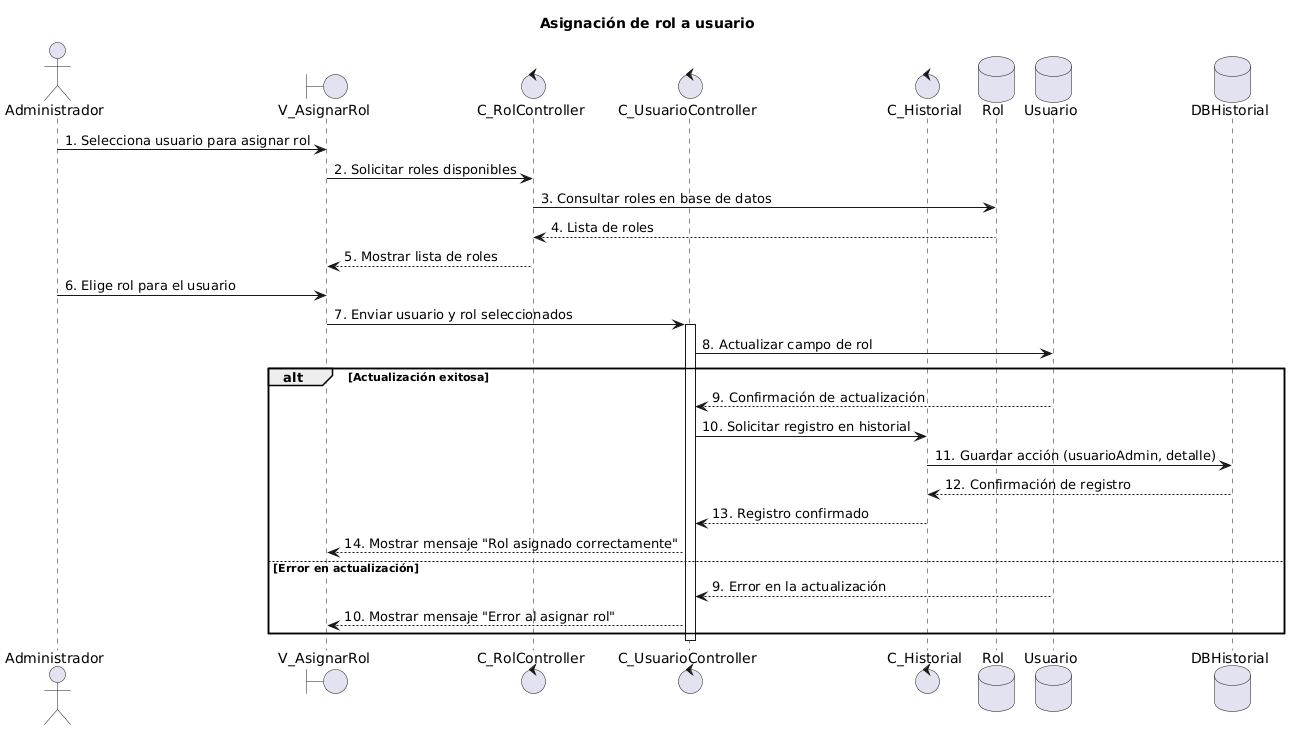
#### CU01 - Ingresando al sistema mediante usuario y contraseña:



*Ilustración 5.3 Diagrama de secuencias CU01*

El usuario ingresa su correo y contraseña desde **V\_Login**, que envía estos datos al controlador **C\_usuarioController**. Este realiza la validación consultando la información en la tabla **Usuario**. Si las credenciales son válidas, se devuelve una confirmación y el sistema redirige al usuario hacia el dashboard o vista principal. En caso contrario, se retorna un error y se muestra un mensaje indicando que el acceso ha sido denegado.

#### CU02 - Definiendo los roles de usuario:



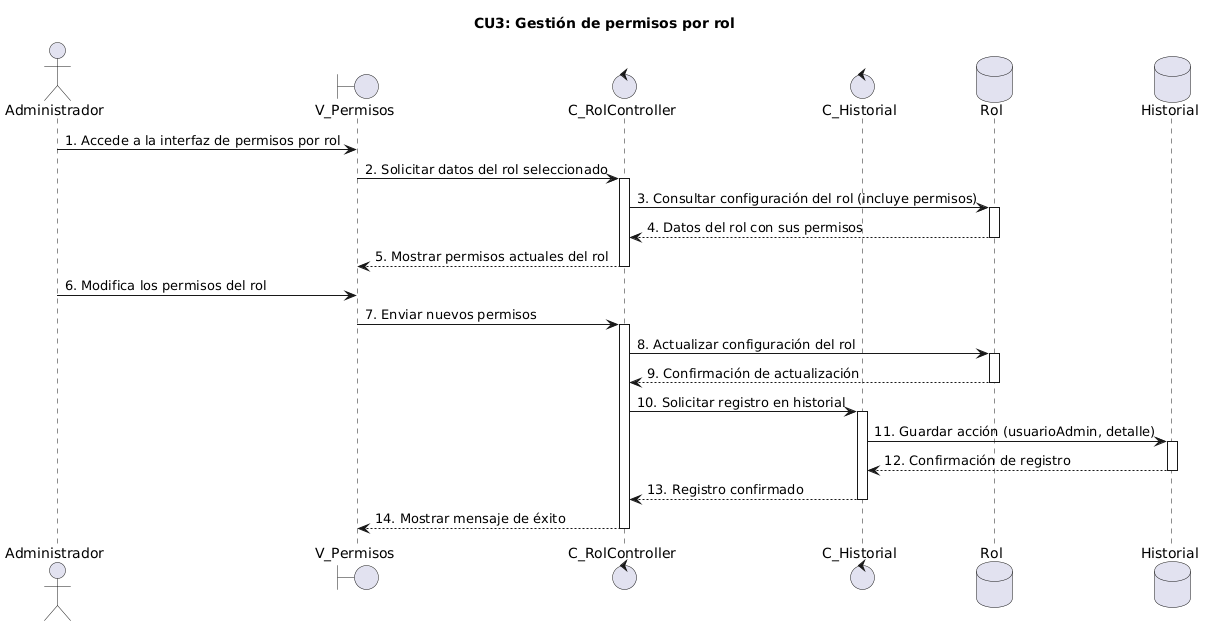
*Ilustración 5.4 Diagrama de secuencias CU02*

Desde la vista **V\_AsignarRol**, el administrador solicita la lista de roles disponibles al controlador **C\_rolController**, que consulta la tabla Rol y devuelve la lista. Tras elegir un rol, la asignación se envía al **C\_usuarioController**, el cual actualiza el campo correspondiente en la tabla Usuario.

Si la actualización es exitosa, el sistema registra la acción en el historial a través de **C\_Historial**, guardando quién realizó la asignación, la acción ejecutada y el momento en que ocurrió. Finalmente, se muestra un mensaje de confirmación.

En caso de que ocurra un error en la actualización, se muestra un mensaje de error al administrador.

#### CU03 - Asignando permisos por módulo:



*Ilustración 5.5 Diagrama de secuencias CU03*

El administrador accede a la interfaz de **V\_Permisos** para gestionar los permisos por rol. Desde allí se solicitan los datos del rol seleccionado, que son obtenidos por medio del **C\_RolController** al consultar la configuración correspondiente en la base de datos **Rol**. Una vez recuperada la información, se muestran los permisos actuales del rol en la interfaz.

El administrador modifica los permisos y envía la nueva configuración al **C\_RolController**, el cual actualiza la información en la base de datos. Tras la confirmación de actualización, el sistema solicita el registro de la acción en el **C\_Historial**, donde se almacena el detalle de la operación junto con el usuario que la ejecutó. Finalmente, se confirma el registro y en la interfaz **V\_Permisos** se muestra un mensaje de éxito al administrador.

#### CU04.1 - Solicitando recuperación de contraseña:



*Ilustración 5.6 Diagrama de secuencias CU04.1*

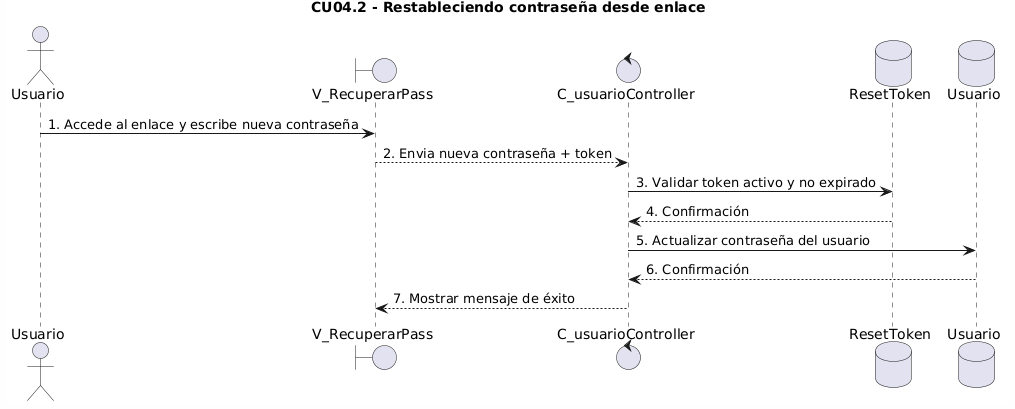
El **Usuario** selecciona la opción *“¿Olvidó su contraseña?”* en la interfaz de **V\_Login**. La vista solicita al usuario que ingrese su correo electrónico y envía esta información al **C\_usuarioController**.

El controlador verifica en la base de datos **Usuario** si el correo existe y, tras la confirmación, procede a generar un **token de recuperación con expiración**, que se guarda en la base de datos **ResetToken**. Una vez confirmado el registro del

token, el **C\_mailController** envía al correo electrónico del usuario un enlace de recuperación asociado al token.

Finalmente, el usuario recibe en su bandeja de entrada el correo con el enlace para restablecer su contraseña.

##### CU04.2 – Restableciendo contraseña desde enlace



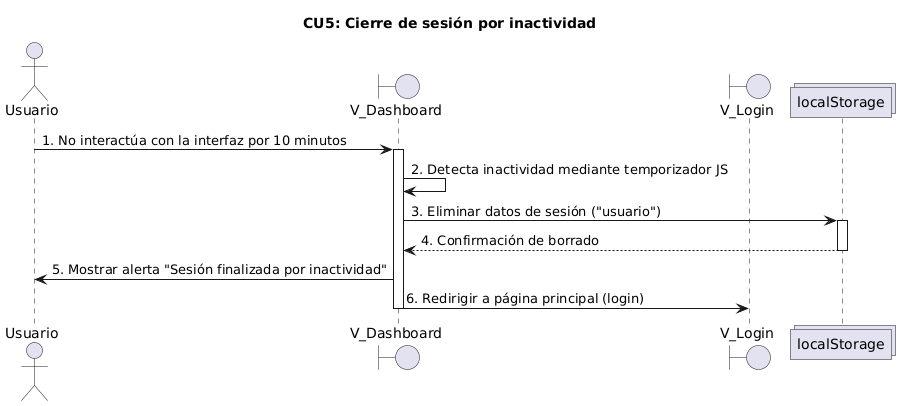
*Ilustración 5.7 Diagrama de secuencias CU04.2*

El **Usuario** accede al enlace de recuperación y escribe su nueva contraseña en la vista **V\_RecuperarPass**. La interfaz envía la nueva contraseña junto con el **token** al **C\_usuarioController**, que se encarga de validar que el token esté activo y no haya expirado.

Tras la confirmación de validez, el controlador actualiza la contraseña en la base de datos **Usuario** y recibe la confirmación de que el cambio fue realizado exitosamente.

Finalmente, la vista muestra un mensaje de éxito notificando al usuario que su contraseña ha sido restablecida correctamente.

#### CU05 - Cerrando automáticamente la sesión por inactividad:

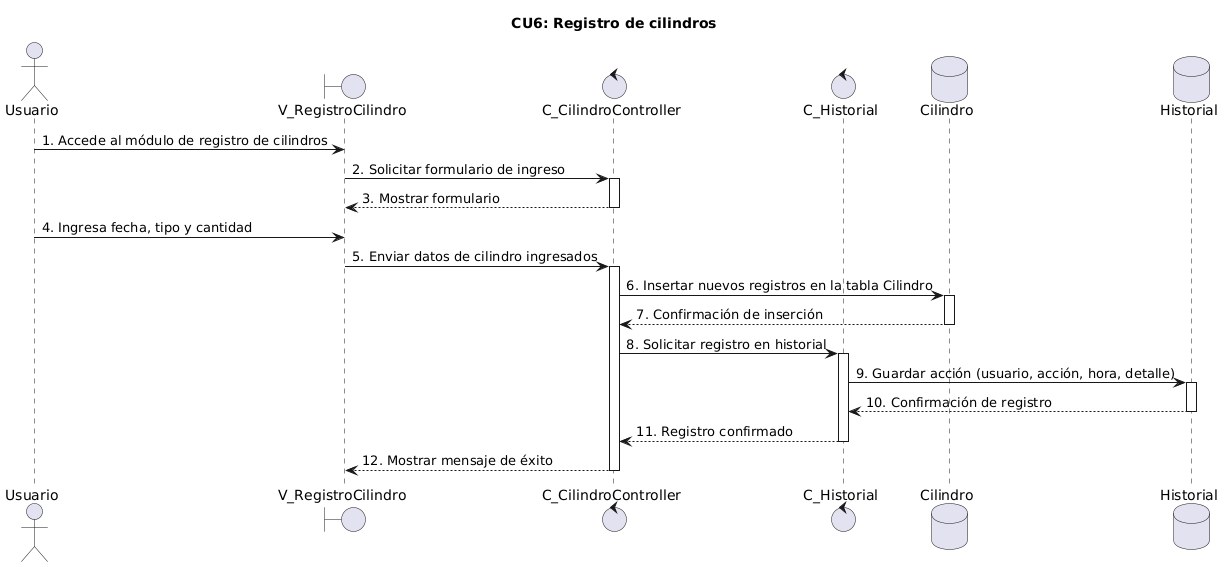


*Ilustración 5.8 Diagrama de secuencias CU05*

El usuario permanece inactivo en la interfaz de **V\_Dashboard** durante 10 minutos. Pasado este tiempo, el sistema detecta la inactividad mediante un temporizador en **JavaScript**. Como respuesta, se procede a eliminar los datos de sesión almacenados en el **localStorage**, confirmando posteriormente el borrado.

Una vez finalizado este proceso, la interfaz muestra una alerta notificando que la sesión ha sido finalizada por inactividad. Finalmente, el sistema redirige al usuario hacia la página principal de inicio de sesión (**V\_Login**).

#### CU06 - Registrando cilindros nuevos:



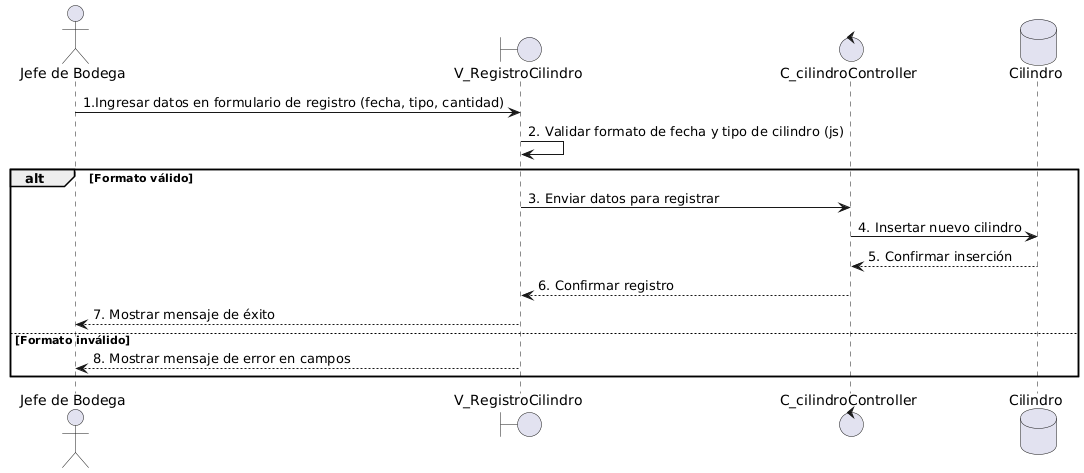
*Ilustración 5.9 Diagrama de secuencias CU06*

El **Usuario** accede al módulo de **V\_RegistroCilindro** para registrar nuevos cilindros. La vista solicita el formulario de ingreso al **C\_CilindroController**, que lo devuelve para su visualización. Posteriormente, el usuario completa los campos correspondientes con la fecha, tipo y cantidad, y envía la información registrada.

El **C\_CilindroController** procesa los datos e inserta los nuevos registros en la base de datos **Cilindro**, recibiendo una confirmación de inserción. Luego, solicita al **C\_Historial** registrar la acción, indicando usuario, acción, hora y detalle, lo que se guarda en la base de datos **Historial** y se confirma exitosamente.

Finalmente, se muestra en la interfaz un mensaje de éxito notificando al usuario que el registro de los cilindros fue realizado correctamente.

#### CU07 - Validando automáticamente los formatos al registrar cilindros:

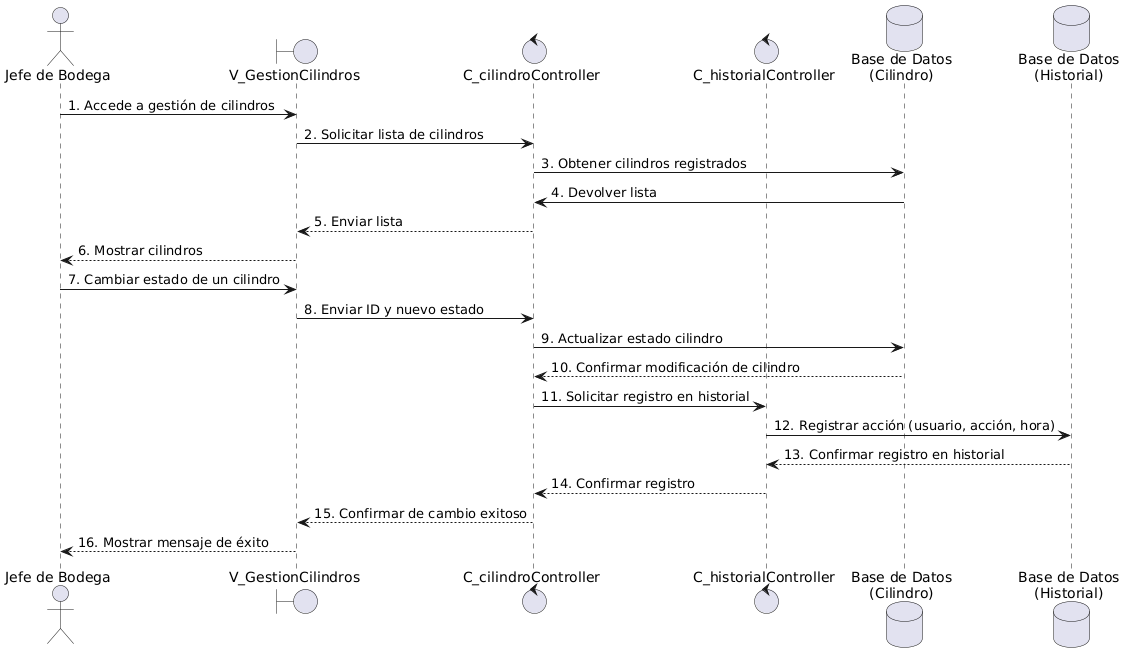


*Ilustración 5.10 Diagrama de secuencias CU07*

El **Jefe de Bodega** ingresa los datos en el formulario de **V\_RegistroCilindro**, especificando la fecha, tipo y cantidad. La interfaz valida el formato de los datos y el tipo de cilindro. Si el formato es correcto, se envían los datos al **C\_cilindroController**, que inserta el nuevo registro en la base de datos **Cilindro** y confirma la inserción. Posteriormente, se confirma el registro y se muestra un mensaje de éxito en la interfaz.

En caso de que el formato de los datos sea inválido, el sistema responde mostrando un mensaje de error en los campos correspondientes.

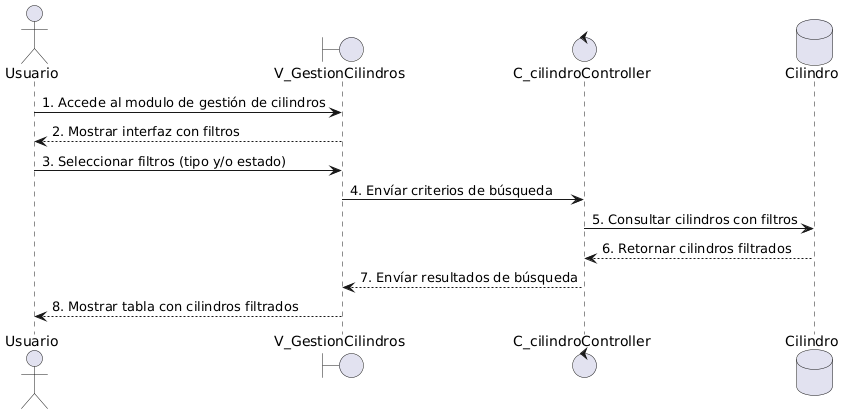
#### CU08 - Modificando el estado del cilindro:



*Ilustración 5.11 Diagrama de secuencias CU08*

El Jefe de Bodega accede a la vista **V\_GestionCilindros**, donde solicita la lista de cilindros. Esta es obtenida por **C\_cilindroController** desde la Base de Datos de **Cilindros** y mostrada al usuario. Al seleccionar un cilindro y cambiar su estado, el controlador actualiza el registro en la base de datos. Luego, **C\_historialController** registra la acción (usuario, acción, hora) en la Base de Datos de **Historial**. Finalmente, se notifica al usuario que el cambio fue exitoso.

#### CU09 - Buscando cilindros por tipo:



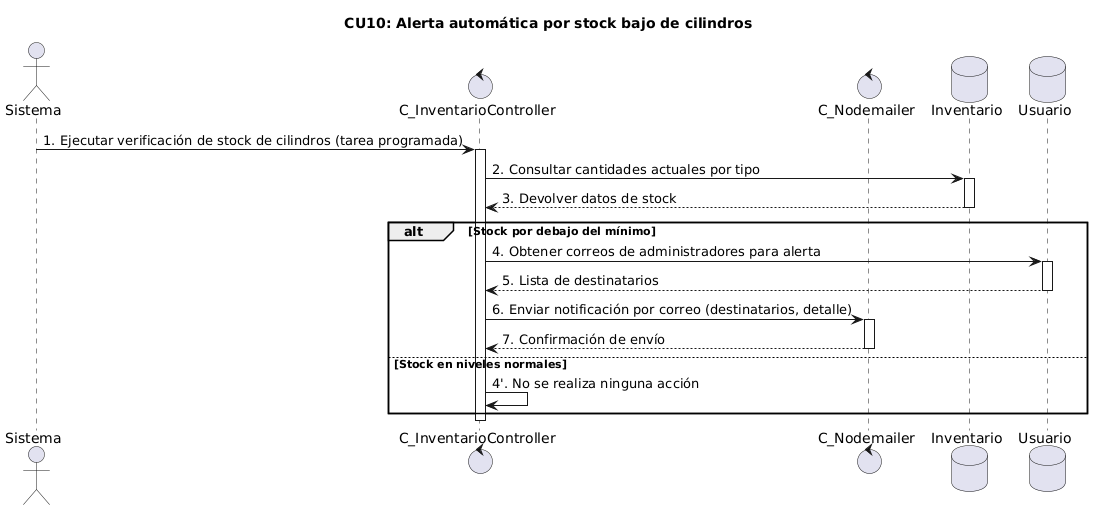
*Ilustración 5.12 Diagrama de secuencias CU09*

El **Usuario** accede al módulo de **V\_GestionCilindros** para realizar la búsqueda de cilindros. La vista muestra una interfaz con filtros, donde el usuario selecciona las opciones correspondientes (tipo y/o estado).

Los criterios de búsqueda son enviados al **C\_cilindroController**, que consulta en la base de datos **Cilindro** los registros que cumplen con las condiciones establecidas. Una vez obtenidos, se devuelven los resultados filtrados a la vista.

Finalmente, la interfaz muestra una tabla con los cilindros filtrados, permitiendo al usuario visualizar únicamente aquellos que cumplen con los parámetros seleccionados.

#### CU10 - Generando alerta de stock mínimo:



*Ilustración 5.13 Diagrama de secuencias CU10*

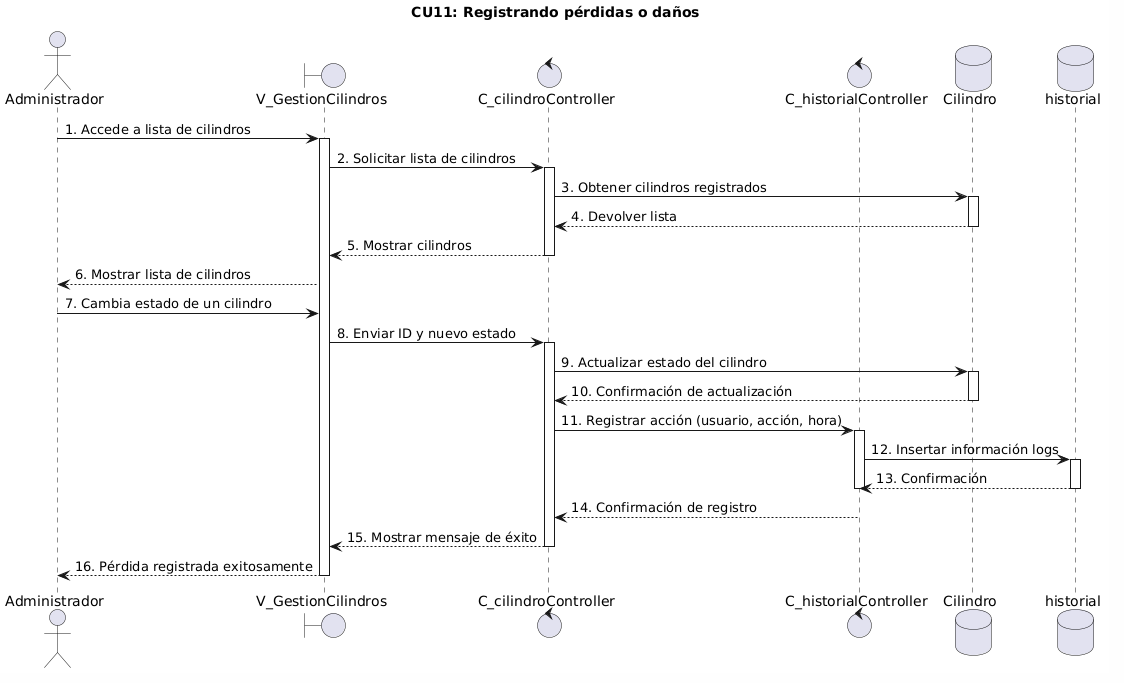
El **Sistema** ejecuta una tarea programada para verificar el stock de cilindros, solicitando al **C\_InventarioController** las cantidades actuales registradas. El controlador consulta en la base de datos **Inventario** y devuelve los datos correspondientes.

El proceso contempla dos escenarios alternativos:

* **Stock por debajo del mínimo:** el controlador obtiene los correos de los administradores desde la base de datos **Usuario** y, con la lista de destinatarios, solicita al **C\_Nodemailer** el envío de una notificación por correo electrónico con los detalles de la alerta. Una vez confirmada la entrega, se asegura que la notificación fue enviada exitosamente.
* **Stock en niveles normales:** no se ejecuta ninguna acción adicional.

De esta manera, el caso de uso permite automatizar la supervisión de stock de cilindros, garantizando que los administradores reciban alertas oportunas en caso de niveles críticos.

#### CU11 - Registrando pérdidas o daños:



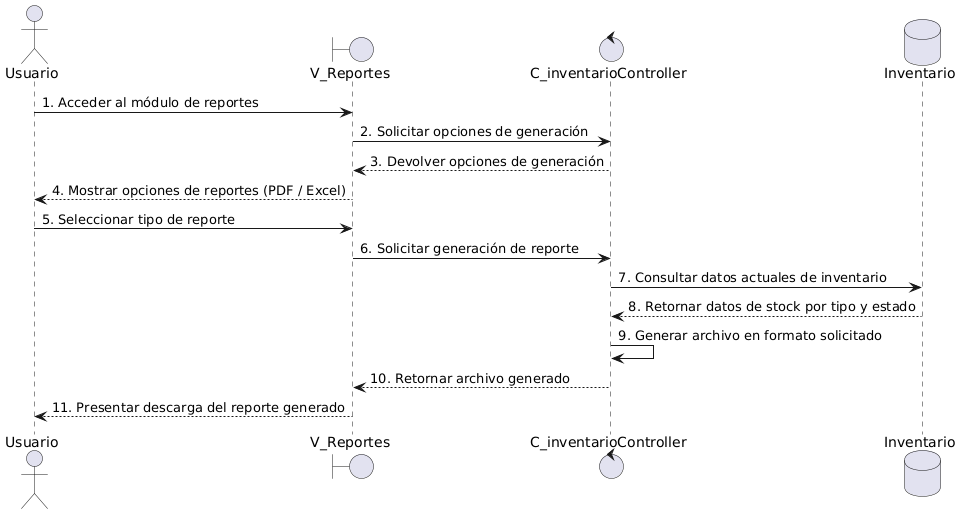
*Ilustración 5.14 Diagrama de secuencias CU11*

El **Administrador** accede a la interfaz de **V\_GestionCilindros** para consultar la lista de cilindros. La vista solicita los datos al **C\_cilindroController**, que obtiene la información registrada en la base de datos **Cilindro** y la devuelve para mostrarla en la interfaz.

Posteriormente, el administrador selecciona un cilindro y cambia su estado, enviando el ID y el nuevo valor al **C\_cilindroController**, el cual actualiza la información en la base de datos. Tras la confirmación de la actualización, se solicita al **C\_historialController** registrar la acción realizada (usuario, acción y hora), insertando la información en la base de datos **Historial** y confirmando el registro.

Finalmente, la interfaz muestra un mensaje de éxito indicando que la pérdida o daño ha sido registrado correctamente.

#### CU12 - Generando reporte diario de stock:



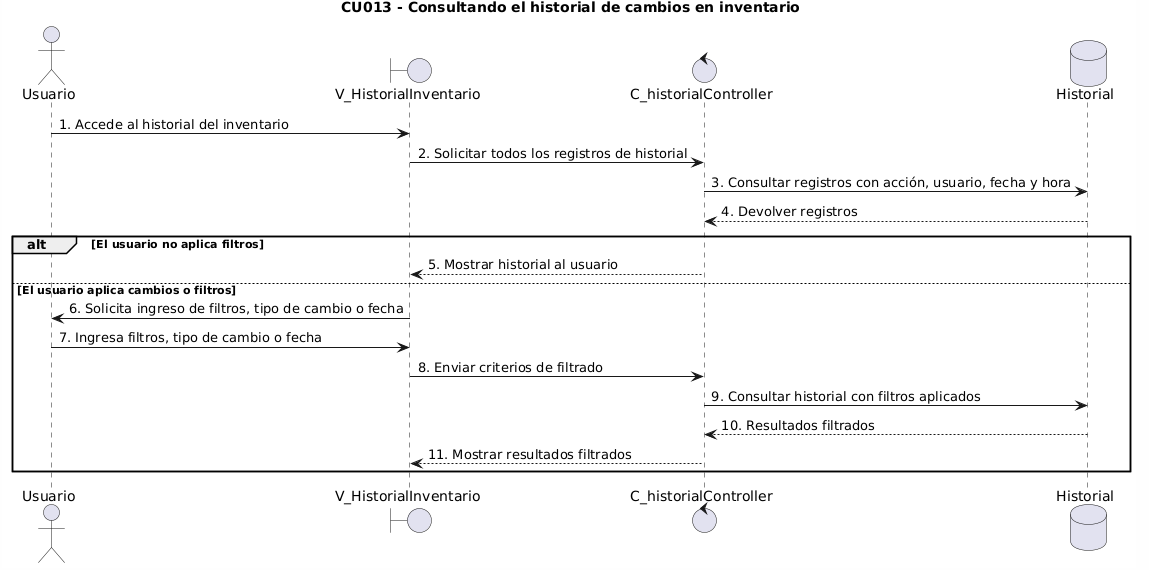
*Ilustración 5.15 Diagrama de secuencias CU12*

El **Usuario** accede al módulo de **V\_Reportes** para generar un informe de inventario. La vista solicita al **C\_inventarioController** las opciones disponibles de generación, que son devueltas y mostradas al usuario (por ejemplo, PDF o Excel).

El usuario selecciona el tipo de reporte que desea y la vista envía la solicitud al controlador, el cual consulta los datos actuales en la base de datos **Inventario**. Tras obtener la información de stock por tipo y estado, el controlador genera el archivo en el formato solicitado.

Finalmente, el sistema retorna el archivo generado a la vista y se presenta al usuario la opción de descargar el reporte.

#### CU13 - Consultando el historial de cambios en inventario:



*Ilustración 5.16 Diagrama de secuencias CU13*

En esta secuencia, el usuario accede a la vista V\_HistorialInventario para consultar los cambios realizados en el inventario. La vista solicita al C\_historialController todos los registros almacenados en la tabla historial, la cual contiene detalles como la acción realizada, el usuario que la ejecutó, la fecha y la hora. Además, el usuario puede aplicar filtros (por fecha, usuario o tipo de acción), los cuales se envían al mismo controlador para refinar los resultados. La vista finalmente muestra los datos solicitados de forma detallada.

## Vista de Desarrollo

El sistema fue construido utilizando una arquitectura en capas cliente-servidor, que organiza la solución en Frontend, Backend y Base de Datos, cada uno con responsabilidades bien definidas.

1. Frontend (Capa Visual)

Corresponde a la capa visual del sistema, encargada de interactuar directamente con el usuario (trabajadores como administrador, jefe de bodega o chofer). Está compuesta por:

* Archivos HTML que definen la estructura de las vistas.
* Hojas de estilo CSS que dan formato y diseño visual a la interfaz.
* Archivos JavaScript (JS) que permiten la validación y lógica en el navegador.
* Un componente de Solicitudes HTTP/JSON que se comunica con el backend para enviar o recibir datos.

2. Backend (Capa de Controladores)

Esta es la capa de controladores y procesamiento lógico del sistema. Desarrollada en Node.js con el framework Express.js, está dividida en varios módulos:

* Una API REST que recibe y responde solicitudes desde el frontend.
* Un middleware de autenticación (AuthMiddleware.js) que valida accesos.
* Múltiples controladores (UsuarioController.js, RolController.js, etc.), cada uno encargado de una funcionalidad específica (usuarios, cilindros, pedidos, reportes, etc.).
* Una integración con el servicio externo Nodemailer.js, utilizado para enviar correos de alerta o recuperación de contraseña.

3. Base de Datos

La capa de persistencia del sistema se implementa con PostgreSQL. Aquí se almacenan de forma estructurada las entidades como usuarios, roles, inventario, pedidos, reportes y tokens. Los controladores del backend se conectan directamente a esta base de datos para realizar operaciones de lectura y escritura.

### Diagrama de componentes

A continuación se muestra la figura 5.17 la cual corresponde al diagrama de componentes utilizado en este proyecto

Diagrama

Descripción generada automáticamente

*Ilustración 5.17 Diagrama de Componentes*

En esta imagen se representa el Diagrama de Componentes que describe la arquitectura modular del sistema:

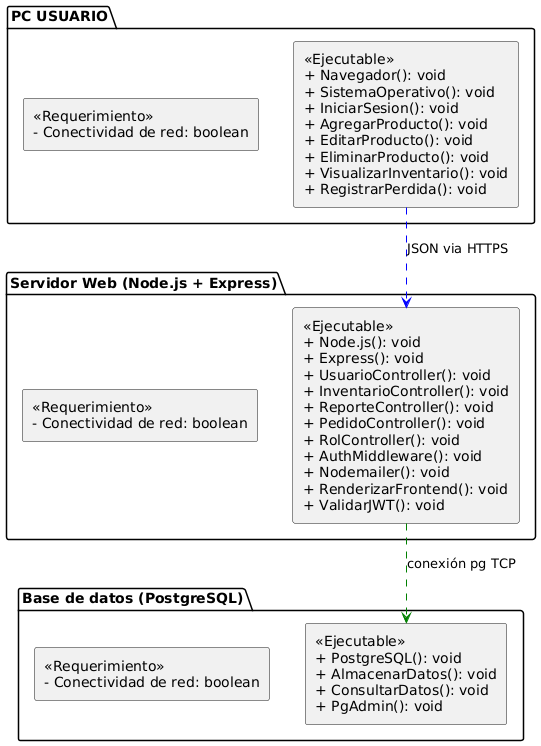
* A la izquierda se encuentra el Frontend, compuesto por la capa visual que contiene los archivos HTML, CSS y JavaScript, agrupados bajo el bloque "Vistas". Esta capa se comunica con el backend enviando solicitudes HTTP/JSON.
* Al centro, el Backend contiene la API REST (basada en Node.js y Express), el middleware de autenticación y el conjunto de Controladores, que procesan la lógica del negocio. También se incluye el servicio externo Nodemailer.js utilizado para el envío de correos.
* A la derecha inferior, se ubica la base de datos PostgreSQL, que es accedida directamente por los controladores para almacenar y recuperar información.

## Vista Física

La vista física representa cómo los componentes del sistema se despliegan sobre la infraestructura física y las conexiones entre ellos. En este caso, se detalla el entorno de ejecución de la solución propuesta, que se basa en una arquitectura cliente-servidor.

### Diagrama de despliegue

En la figura “5.18”, se muestra el diagrama de arquitectura del sistema, el cual describe la interacción entre el PC del Usuario, el Servidor Web desarrollado con Node.js y Express, y la base de datos PostgreSQL. El flujo de información se realiza mediante el envío y recepción de datos en formato JSON a través del protocolo HTTPS entre el cliente y el servidor, y mediante conexión TCP entre el servidor y la base de datos. Este diseño asegura que todas las operaciones - como iniciar sesión, gestionar inventario, registrar pérdidas y generar reportes - se realicen de forma centralizada, segura y con requerimiento de conectividad de red en cada capa del sistema.



*Ilustración 5.18 Diagrama de Despliegue*

El diagrama de despliegue representa la arquitectura física del sistema web de gestión de cilindros, detallando los nodos involucrados y su interacción.

PC Usuario: representa a los actores del sistema (Administrador, Jefe de Bodega), quienes acceden mediante navegador web. Desde aquí se realizan acciones como iniciar sesión, agregar o editar productos, registrar pérdidas, entre otras.

Servidor Web (Node.js + Express): alberga el backend y la lógica de negocio. Gestiona autenticación con tokens JWT, renderiza el frontend, y organiza la lógica mediante controladores y middleware.

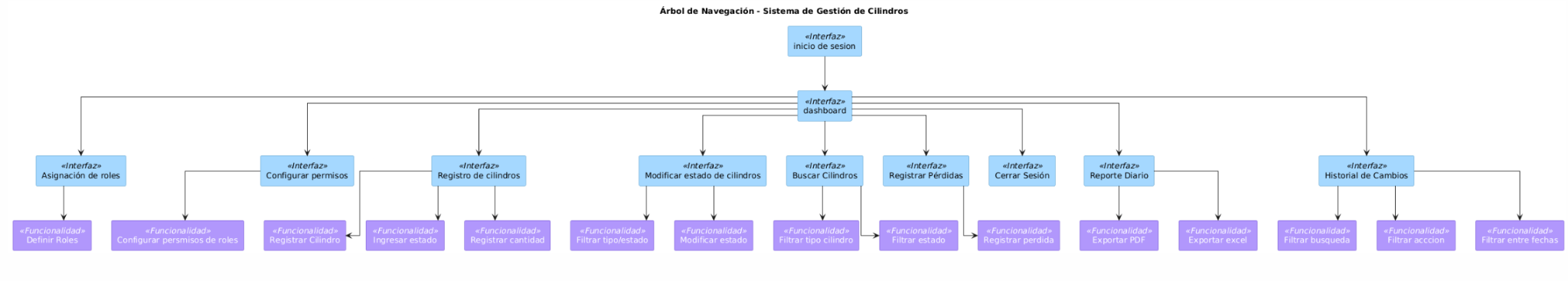
Base de Datos (PostgreSQL): almacena la información del sistema, incluyendo usuarios, productos, inventario, historial y seguridad. Se accede desde el servidor mediante conexión TCP.

Las conexiones se realizan mediante JSON sobre HTTPS entre el cliente y el servidor, y por TCP (librería pg) entre el backend y la base de datos. Todos los nodos requieren conectividad de red.

# Árbol de Navegación

A continuación se presenta la figura 6.1, corresponde al Árbol de Navegación, representa la organización jerárquica de las interfaces y funcionalidades del sistema de gestión de cilindros. A partir del análisis de los casos de uso definidos en las distintas vistas (principalmente la vista lógica, de proceso y externa), se identifican las pantallas principales del sistema y las acciones asociadas a cada una.

Este diagrama permite visualizar cómo se estructura la navegación del sistema desde el inicio de sesión hacia las distintas secciones funcionales, facilitando el diseño de la interfaz de usuario y la implementación coherente de las tareas asociadas.



*Ilustración 6.1 Árbol de Navegación*

En la parte superior se encuentran las interfaces iniciales del sistema: **Inicio de sesión** y **Dashboard**, que actúan como punto de acceso al resto de los módulos. Desde el *dashboard* se despliegan las distintas interfaces del sistema, cada una con sus funcionalidades específicas:

* **Asignación de roles:** permite definir y configurar los roles de los usuarios.
* **Configurar permisos:** posibilita asignar y gestionar permisos asociados a los roles.
* **Registro de cilindros:** habilita funciones para registrar cilindros, cantidades y estados.
* **Modificar estado de cilindros:** ofrece opciones para filtrar y actualizar el estado de los cilindros.
* **Buscar cilindros:** incluye filtros por tipo y estado para facilitar la búsqueda.
* **Registrar pérdidas:** permite registrar pérdidas de cilindros aplicando filtros de estado.
* **Cerrar sesión:** proporciona la funcionalidad de salida del sistema.
* **Reporte diario:** genera informes exportables en formato **PDF** o **Excel**.
* **Historial de cambios:** ofrece filtros por búsqueda, acción o fechas para consultar los registros de modificaciones realizadas en el sistema.

Este árbol facilita la visualización de la estructura de navegación, evidenciando cómo se agrupan las interfaces y las funcionalidades que permiten una administración integral del sistema.

# Validación y evidencia de Casos de prueba

## Caso de Uso N°1 – Ingresando al sistema mediante Correo y Contraseña

**Casos de Prueba:**

1. Verificar que se despliegue el formulario de ingreso.
2. Ingresar credenciales válidas y acceder al sistema.
3. Ingresar credenciales inválidas y comprobar la respuesta del sistema.

**Resultados Esperados:**

1. El formulario de inicio de sesión aparece correctamente.
2. El sistema permite el acceso y redirige al panel correspondiente.
3. El sistema muestra un mensaje de error por credenciales incorrectas.

**Resultados Obtenidos:**

A continuación se presentan las figuras 7.1, 7.2, 7.3, que corresponden a la interfaz del caso de uso n°1

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 7.1 Evidencia CU01.1*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 7.2 Evidencia CU01.2*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 7.3 Evidencia CU01.3*

## Caso de Uso N°2 – Definiendo de roles de usuario

**Casos de Prueba:**

1. Ingresar a la vista de usuarios.
2. Seleccionar un usuario y asignar un rol.
3. Confirmar que el rol asignado se guarde correctamente.

**Resultados Esperados:**

1. Se visualiza la lista de usuarios disponibles.
2. El administrador puede asignar un rol desde el sistema.
3. El rol queda registrado correctamente en la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 7.4 Evidencia CU02.1*

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Ilustración 7.5 Evidencia CU02.2*

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Ilustración 7.6 Evidencia CU02.3*

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Ilustración 7.7 Evidencia CU02.4*

## Caso de Uso N°3 – Asignando permisos por módulo

**Casos de Prueba:**

1. Acceder a la interfaz de permisos.
2. Modificar permisos de un rol específico.
3. Verificar que los cambios se apliquen y se almacenen.

**Resultados Esperados:**

1. La interfaz permite la visualización de los permisos actuales.
2. El administrador modifica permisos y los guarda.
3. El sistema actualiza correctamente los permisos en la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 7.8 Evidencia CU03.1*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Ilustración 7.9 Evidencia CU03.2*

## Caso de Uso N°4 – Recuperando la contraseña

**Casos de Prueba:**

1. Acceder a la opción “¿Olvidó su contraseña?”.
2. Ingresar el correo registrado.
3. Establecer una nueva contraseña a través del enlace.

**Resultados Esperados:**

1. El sistema solicita correo válido.
2. Se envía un enlace con tiempo de expiración.
3. La nueva contraseña queda registrada y permite el acceso.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

*Ilustración 7.10 Evidencia CU04.1*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 7.11 Evidencia CU04.2*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 7.12 Evidencia CU04.3*

## Caso de Uso N°5 – Cerrando automáticamente la sesión por inactividad

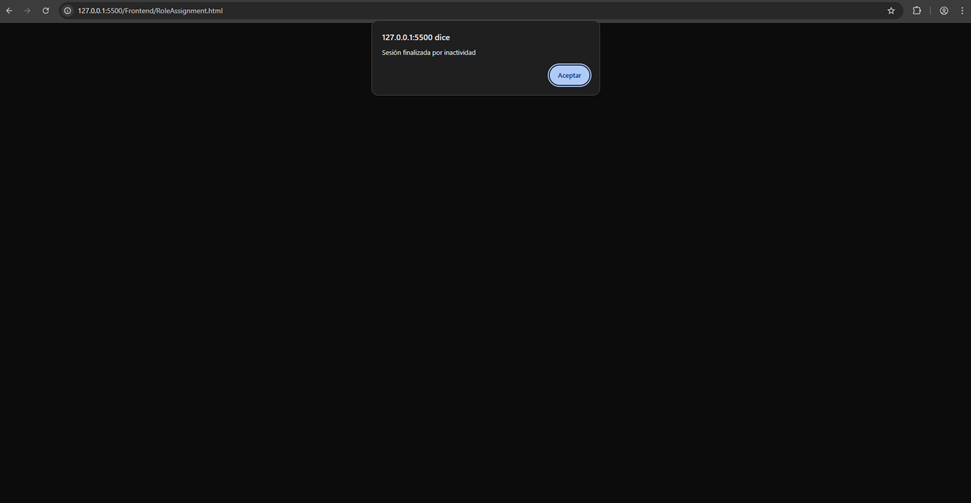
**Casos de Prueba:**

1. Iniciar sesión y permanecer inactivo por más de 9 minutos.
2. Comprobar si se muestra advertencia previa al cierre.
3. Validar cierre automático tras 10 minutos sin respuesta.

**Resultados Esperados:**

1. El sistema muestra advertencia de cierre tras 9 minutos.
2. Si no hay actividad, la sesión se cierra automáticamente.
3. El usuario debe volver a ingresar sus credenciales para acceder.





*Ilustración 7.13 Evidencia CU05*

## Caso de Uso N°6 – Registrando cilindros nuevos

**Casos de Prueba:**

1. Ingresar al módulo de inventario.
2. Completar formulario con tipo, fecha y cantidad de cilindros.
3. Confirmar el registro.

**Resultados Esperados:**

1. El sistema despliega el formulario correctamente.
2. Los datos se ingresan sin errores.
3. Los cilindros quedan almacenados en el inventario.



Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Ilustración 7.14 Evidencia CU06.1*

Tabla

Descripción generada automáticamente

*Ilustración 7.15 Evidencia CU06.2*

## Caso de Uso N°7 – Validando automáticamente los formatos al registrar cilindros

**Casos de Prueba:**

1. Registrar un cilindro con fecha en formato incorrecto.
2. Registrar un tipo de cilindro inexistente.
3. Verificar la validación del sistema.

**Resultados Esperados:**

1. El sistema muestra error si la fecha no es válida.
2. El sistema bloquea tipos de cilindros no definidos.
3. Solo se permite continuar si los formatos son correctos.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Ilustración 7.16 Evidencia CU07*

## Caso de Uso N°8 – Modificando el estado del cilindro

**Casos de Prueba:**

1. Ingresar a la lista de cilindros registrados.
2. Seleccionar un cilindro y cambiar su estado (lleno/vacío).
3. Confirmar el cambio.

**Resultados Esperados:**

1. El sistema permite filtrar cilindros por estado.
2. El usuario cambia correctamente el estado.
3. El sistema guarda el cambio y genera log con datos del usuario.

Tabla

Descripción generada automáticamente

*Ilustración 7.17 Evidencia CU08.1*

Tabla

Descripción generada automáticamente

*Ilustración 7.18 Evidencia CU08.2*

Tabla

Descripción generada automáticamente

*Ilustración 7.19 Evidencia CU08.3*

## Caso de Uso N°9 – Buscando cilindros por tipo

**Casos de Prueba:**

1. Acceder al módulo de inventario.
2. Seleccionar tipo y estado de cilindro desde filtros.
3. Visualizar listado filtrado.

**Resultados Esperados:**

1. El sistema muestra opciones de filtro (tipo y estado).
2. Se seleccionan filtros correctamente.
3. Se despliega listado de cilindros que cumplen los criterios.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 7.20 Evidencia CU09*

## Caso de Uso N°10 – Generando alerta de stock mínimo

**Casos de Prueba:**

1. Simular una baja de stock por debajo del umbral mínimo.
2. Verificar la generación de alertas.
3. Confirmar recepción de mensaje interno o correo.

**Resultados Esperados:**

1. El sistema detecta automáticamente que el stock es bajo.
2. Se genera alerta visible en la interfaz.
3. Se envía notificación interna y correo al responsable.



*Ilustración 7.21 Evidencia CU10*

## Caso de Uso N°11 – Registrando pérdidas o daños

**Casos de Prueba:**

1. Acceder al módulo de pérdidas.
2. Ingresar tipo de pérdida, fecha, motivo e imagen opcional.
3. Confirmar el registro.

**Resultados Esperados:**

1. El formulario se muestra correctamente.
2. Los datos se validan y se guarda el registro.
3. Se genera log con la información y archivo adjunto si aplica.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 7.22 Evidencia CU11.1*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 7.23 Evidencia CU11.2*

## Caso de Uso N°12 – Generando reporte diario de stock

**Casos de Prueba:**

1. Ingresar al módulo de reportes.
2. Seleccionar la opción “reporte diario de stock”.
3. Descargar reporte en formato PDF o Excel.

**Resultados Esperados:**

1. El sistema muestra opciones de descarga.
2. Se genera archivo con datos del día (tipo, cantidad, estado).
3. El archivo se descarga correctamente en el formato elegido.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

*Ilustración 7.24 Evidencia CU12.1*

Texto

Descripción generada automáticamente

*Ilustración 7.25 Evidencia CU12.2*

Imagen que contiene Tabla

Descripción generada automáticamente

*Ilustración 7.26 Evidencia CU12.3*

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

*Ilustración 7.27 Evidencia CU12.4*

## Caso de Uso N°13 – Consultando el historial de cambios en inventario

**Casos de Prueba:**

1. Ingresar al historial de inventario.
2. Filtrar por usuario, fecha o tipo de acción.
3. Ver detalles del cambio.

**Resultados Esperados:**

1. El historial es visible y accesible según permisos.
2. Se aplican filtros correctamente.
3. Se visualizan detalles con usuario, fecha y tipo de acción.

Interfaz de usuario gráfica

Descripción generada automáticamente

*Ilustración 7.28 Evidencia CU13.1*

Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente

*Ilustración 7.29 Evidencia CU13.2*

# Review Del Sprint

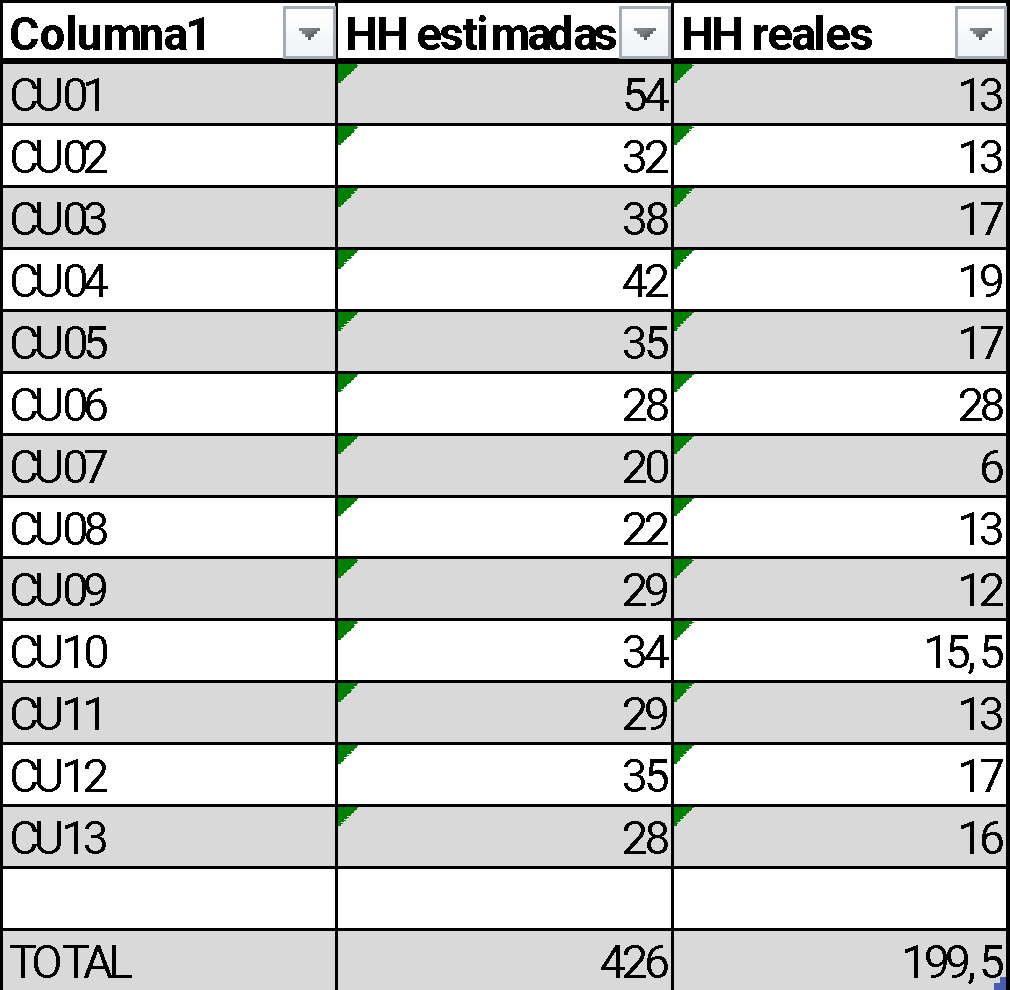
A continuación se muestra la figura 8.1 la cual es una ilustración de ejemplo para la revisión del sprint. (en la carpeta de entrega está la planificación y sprint final realizado.)

Tabla

Descripción generada automáticamente

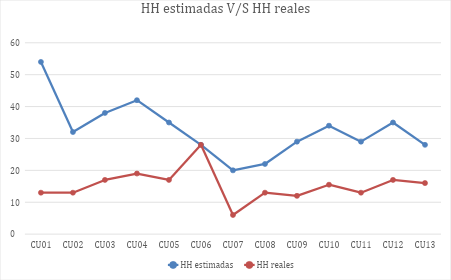
*Ilustración 8.1 Tabla Sprint FInal*

En la Figura 8.2 se observa la comparación entre las horas hombre estimadas y las reales para cada caso de uso, mostrando una diferencia considerable entre ambos valores.



*Ilustración 8.2 Tabla de horas estimadas vs reales*

En la Figura 8.3 se presenta el gráfico comparativo entre las horas hombre estimadas y las reales, el cual permite visualizar de manera más clara la diferencia entre ambos valores.



*Ilustración 8.3 Gráfico de la tabla de las horas estimada*

# Bibliografías

* Pichler, R. (2019). *Agile Product Management with Scrum: Creating Products That Customers Love*. Addison-Wesley Professional.
* Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería de software: Un enfoque práctico* (9.ª ed.). McGraw-Hill.
* Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org. https://scrumguides.org/scrum-guide.html
* Sommerville, I. (2016). *Ingeniería del software* (10.ª ed.). Pearson Educación.
* Universidad Andrés Bello. (2025). *Guía metodológica para el desarrollo de proyectos con Scrum++* [Material docente no publicado]. Facultad de Ingeniería, Universidad Andrés Bello.