

# Escaleta-Teclado GC

# Especificación de realización de casos de uso

**Versión <1.0>**

# Historial de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| **13/09/2023** | **1.0** | **Primera versión** | **Wílmer E. León** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Prefacio**

El siguiente estudio de caso ha sido modificado con respecto a su contenido original. El estudio de caso está destinado a ser utilizado como punto de partida para ayudarle a comprender cómo utilizar el artefacto. Así, la información se ha reducido para evitar navegar en un documento enorme (en tamaño y páginas).

También puede consultar la plantilla relacionada (en formato HTML o WORD) en la Sección de Análisis y Diseño de Plantillas de Artefactos de UPEDU.

Saludos,

Equipo de Proceso Unificado para la Educación

# Tabla de contenidos

[Escaleta-Teclado GC 1](#_Toc177375514)

[Especificación de realización de casos de uso 1](#_Toc177375515)

[Historial de revisiones 2](#_Toc177375516)

[Tabla de contenidos 4](#_Toc177375517)

[Figuras 5](#_Toc177375518)

[Especificación de realización de casos de uso 6](#_Toc177375519)

[1. Introducción 6](#_Toc177375520)

[1.1. Propósito 6](#_Toc177375521)

[1.2. Alcance 6](#_Toc177375522)

[1.3. Definiciones, acrónimos y abreviaturas 6](#_Toc177375523)

[1.4. Referencias 6](#_Toc177375524)

[Documento 6](#_Toc177375525)

[Versión 6](#_Toc177375526)

[Fecha de versión 6](#_Toc177375527)

[Carta de proyecto Estimación de tiempo y recursos 6](#_Toc177375528)

[1.0 6](#_Toc177375529)

[13 de septiembre de 2024 6](#_Toc177375530)

[Especificación de requerimientos 7](#_Toc177375531)

[1.0 7](#_Toc177375532)

[10 de septiembre de 2024 7](#_Toc177375533)

[Formatos Plan de Mejoramiento 1- 2 - 3 7](#_Toc177375534)

[1.0. 7](#_Toc177375535)

[10 de septiembre de 2024 7](#_Toc177375536)

[Enunciado de creación del diagrama de casos de uso 7](#_Toc177375537)

[1.0 7](#_Toc177375538)

[12 de septiembre de 2024 7](#_Toc177375539)

[Documento de arquitectura de software 7](#_Toc177375540)

[1.0 7](#_Toc177375541)

[12 de septiembre de 2024 7](#_Toc177375542)

[1.5. Descripción general 7](#_Toc177375543)

[1. CASO DE USO <Ingresar al Módulo DCSHA> 7](#_Toc177375544)

[2. CASO DE USO <Seleccionar plantilla> 14](#_Toc177375545)

[3. CASO DE USO <Editar escaleta> 19](#_Toc177375546)

# Figuras

[Ilustración 1: Caso de uso 1 | Ingresar a módulo DCSHA 7](#_Toc177149418)

# Especificación de realización de casos de uso

## Introducción

## Propósito

Este documento proporciona una descripción general completa del sistema, utilizando varios diagramas diferentes para representar las funciones del sistema.

## Alcance

El presente documento tiene como finalidad establecer un marco claro y conciso para la especificación de los casos de uso del sistema *Escaleta-Teclado GC,* una herramienta innovadora diseñada para optimizar la creación y gestión de escaletas mediante un teclado virtual intuitivo. Este sistema busca mejorar la eficiencia y la experiencia del usuario en la producción de contenido, permitiendo a los usuarios interactuar de manera fluida y efectiva con la aplicación. En este contexto, el alcance del documento abarca la identificación y descripción detallada de los actores involucrados, así como los casos de uso que representan las interacciones entre estos actores y el sistema. Se abordarán los requisitos funcionales y no funcionales que el sistema debe cumplir, así como las limitaciones y exclusiones que definirán el perímetro del proyecto. Además, se incluirán consideraciones sobre la interfaz de usuario y el manejo de la confidencialidad de los datos, asegurando que el sistema no solo cumpla con las expectativas de funcionalidad, sino que también garantice la seguridad y la integridad de la información. Este enfoque integral permitirá al equipo de desarrollo y a las partes interesadas tener una comprensión compartida de los objetivos y requisitos del sistema, facilitando así un proceso de desarrollo más alineado y eficiente.

### Definiciones, acrónimos y abreviaturas

|  |  |
| --- | --- |
| Término / acrónimo | Definición |
| ERS | Especificación de requerimientos de ‘software’: Especificación de requerimientos de *software*, que puede ser sinónimo de ERS en algunos contextos. |
| UML | Unified Modeling Language: Lenguaje de modelado unificado utilizado para especificar, visualizar, construir y documentar los artefactos de un sistema de *software*. |
| ETgc | Escaleta-Teclado GC: Aplicativo de montaje de escaleta de tiempos y teclados de previsualización de contenidos gráficos con texto para transmisiones de programas en Máster. |
|  |  |
|  |  |

### Referencias

|  |  |  |
| --- | --- | --- |
| Documento | Versión | Fecha de versión |
| Carta de proyecto Estimación de tiempo y recursos | 1.0 | 13 de septiembre de 2024 |
| Especificación de requerimientos | 1.0 | 10 de septiembre de 2024 |
| Formatos Plan de Mejoramiento 1- 2 - 3 | 1.0. | 10 de septiembre de 2024 |
| Enunciado de creación del diagrama de casos de uso | 1.0 | 12 de septiembre de 2024 |
| Documento de arquitectura de software | 1.0 | 12 de septiembre de 2024 |

### Descripción general

Los casos de uso en el sistema *Escaleta-Teclado GC* son representaciones fundamentales que describen cómo los diferentes actores interactúan con la aplicación para lograr objetivos específicos relacionados con la creación y gestión de escaletas. Cada caso de uso captura una funcionalidad particular del sistema, proporcionando un marco claro para entender las expectativas y necesidades de los usuarios finales, así como los requisitos técnicos que deben ser implementados por el equipo de desarrollo.

En este sistema, los casos de uso abarcan una variedad de acciones que los usuarios pueden realizar, tales como la creación de nuevas escaletas, la edición de escaletas existentes, la eliminación de contenido no deseado y la visualización de escaletas en diferentes formatos. Además, se contemplan interacciones más complejas, como la integración de elementos multimedia y la colaboración entre múltiples usuarios, lo que permite un trabajo en equipo más eficiente.

Cada caso de uso se documentará con un enfoque en los flujos de trabajo típicos, así como en las excepciones y errores que pueden surgir durante la interacción con el sistema. Esto incluye la identificación de precondiciones necesarias para iniciar un caso de uso, el flujo principal de acciones que se llevan a cabo, y los flujos alternativos que describen cómo manejar situaciones imprevistas.

La documentación de los casos de uso no solo servirá como guía para el desarrollo del sistema, sino que también facilitará la comunicación entre los diferentes interesados, asegurando que todos tengan una comprensión clara de las funcionalidades que se están implementando. Al proporcionar una visión estructurada de las interacciones del usuario con el sistema, los casos de uso en *Escaleta-Teclado GC* son esenciales para garantizar que la solución final cumpla con las expectativas de los usuarios y se alinee con los objetivos del proyecto.

### CASO DE USO <Ingresar al Módulo DCSHA>

#### Breve descripción

El productor abre la aplicación e ingresa al módulo DCSHA.

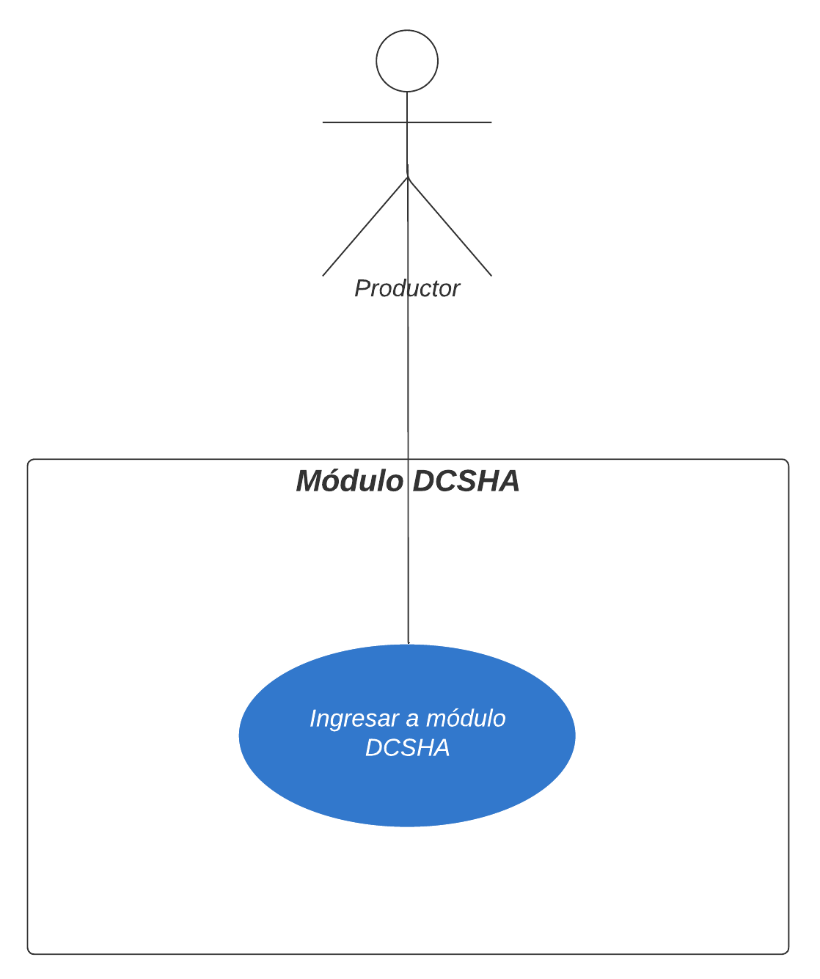


Ilustración 1: Caso de uso 1 | Ingresar a módulo DCSHA

#### Eventos de flujo

* El productor abre la aplicación.
* El productor selecciona el módulo DCSHA.
* El sistema verifica que el módulo DCSHA existe.
* El sistema muestra la interfaz del módulo DCSHA.

#### Diagramas de interacción

* **Actor:**

**Productor:** Representa al usuario que interactúa con la aplicación. En este contexto, el productor es quien inicia el proceso.

* **Participantes:**

**Aplicación (App):** Es el sistema que el productor utiliza. Se encarga de mostrar la interfaz y gestionar las interacciones del usuario.

**Sistema (Sys):** Es un componente que verifica la existencia del módulo solicitado por el productor.

* **Flujo de Interacciones:**

**Paso 1:** El productor inicia la interacción abriendo la aplicación (Productor -> App: Abrir aplicación).

**Paso 2:** La aplicación responde mostrando la interfaz al productor (App -> Productor: Mostrar interfaz).

**Paso 3:** El productor selecciona el módulo DCSHA en la aplicación (Productor -> App: Seleccionar módulo DCSHA).

**Paso 4:** La aplicación envía una solicitud al sistema para verificar si el módulo DCSHA existe (App -> Sys: Verificar existencia del módulo).

**Paso 5:** El sistema responde a la aplicación confirmando la existencia del módulo (Sys -> App: Confirmar existencia).

**Paso 6:** Finalmente, la aplicación muestra la interfaz del módulo DCSHA al productor (App -> Productor: Mostrar interfaz del módulo DCSHA).

* + 1. *Diagrama de secuencia*

El diagrama representa el proceso de un usuario (Productor) que interactúa con una aplicación para acceder a un módulo específico (DCSHA). Este tipo de diagrama es útil para visualizar cómo se llevan a cabo las interacciones en un sistema y cómo se gestionan las solicitudes y respuestas entre los diferentes componentes.

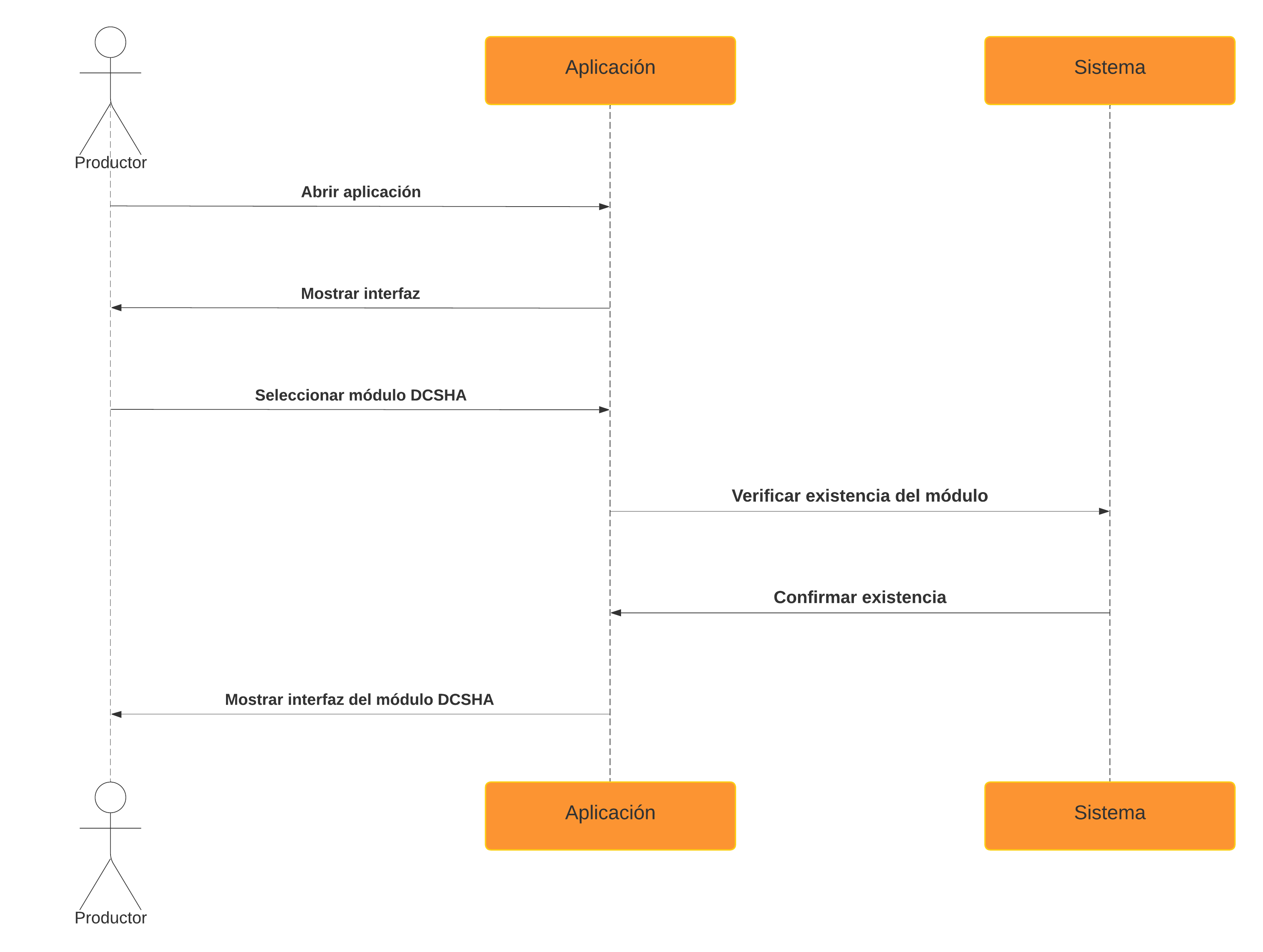


Ilustración 2: Diagrama de secuencia | Caso de uso 1

* **Inicio de la Interacción:**
  + **Productor -> App: Abrir aplicación**
    - El proceso comienza cuando el Productor envía un mensaje a la Aplicación para abrirla. Esto indica que el usuario desea iniciar su trabajo en el sistema.
* **Respuesta de la Aplicación:**
  + **App -> Productor: Mostrar interfaz**
    - La Aplicación responde mostrando su interfaz al Productor. Esto significa que la aplicación ha sido cargada y está lista para ser utilizada.
* **Selección del Módulo:**
  + **Productor -> App: Seleccionar módulo DCSHA**
    - Una vez que la interfaz está visible, el Productor selecciona un módulo específico llamado "DCSHA". Este paso implica que el usuario está interesado en acceder a funcionalidades o datos relacionados con ese módulo.
* **Verificación de Existencia:**
  + **App -> Sys: Verificar existencia del módulo**
    - La Aplicación envía una solicitud al Sistema para verificar si el módulo DCSHA está disponible. Este paso es crucial para asegurarse de que el módulo puede ser utilizado antes de proceder.
* **Confirmación del Sistema:**
  + **Sys -> App: Confirmar existencia**
    - El Sistema responde a la Aplicación confirmando que el módulo DCSHA efectivamente existe. Esta confirmación es necesaria para que la Aplicación pueda continuar con el flujo de trabajo.
* **Mostrar Interfaz del Módulo:**
  + **App -> Productor: Mostrar interfaz del módulo DCSHA**
    - Finalmente, la Aplicación muestra al Productor la interfaz del módulo DCSHA. Esto permite al usuario interactuar con las funcionalidades específicas de ese módulo.

*1.3.2 Diagrama de colaboración*

* Visualizar Interacciones: El diagrama muestra cómo los actores (en este caso, el “Productor”) y los componentes del sistema (la “Aplicación” y el “Sistema”) interactúan entre sí a lo largo del tiempo. Esto ayuda a entender el flujo de mensajes y la secuencia de acciones.
* Describir el Comportamiento del Sistema: Permite documentar y analizar el comportamiento del sistema en respuesta a las acciones del usuario. Es útil para identificar cómo se manejan las solicitudes y las respuestas.
* Facilitar la Comunicación: Sirve como una herramienta de comunicación entre los desarrolladores, diseñadores y otros interesados, proporcionando una representación clara de cómo se espera que funcione el sistema.

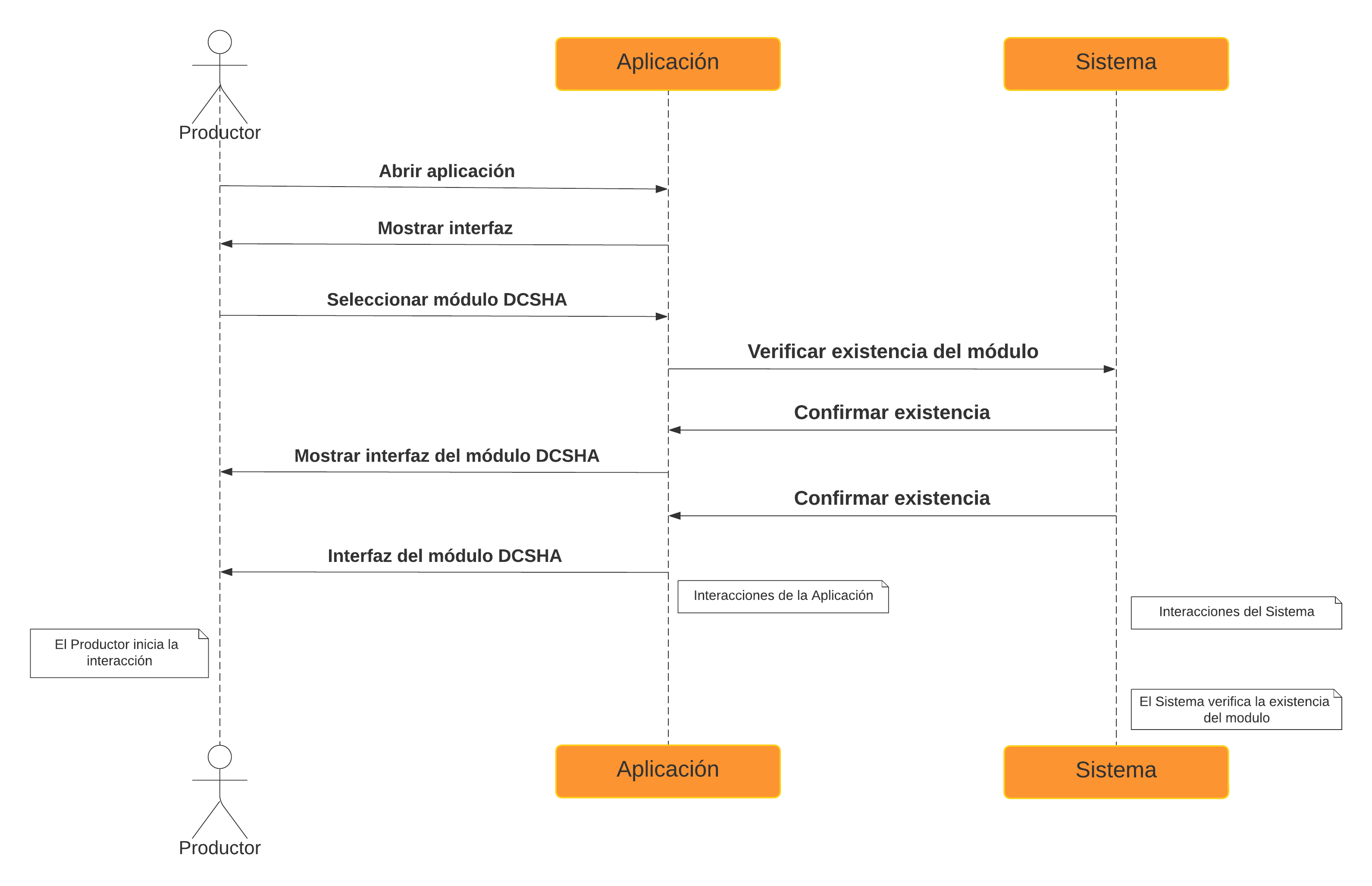


Ilustración 3: Diagrama de colaboración | Caso de uso 1

#### Objetos participantes

* Productor.
* Aplicación.
* Sistema.

#### 1.5 Diagrama de objetos

Este diagrama de objetos ilustra cómo se organiza un sistema en capas, donde cada capa tiene responsabilidades específicas. La **Capa de Presentación** se encarga de la interfaz de usuario, la **Capa de Control** gestiona la lógica de control y la **Capa de Negocio** maneja la lógica de negocio y los datos. Las relaciones entre las clases muestran cómo interactúan y se comunican entre sí, lo que facilita la comprensión de la arquitectura del sistema y su funcionamiento.

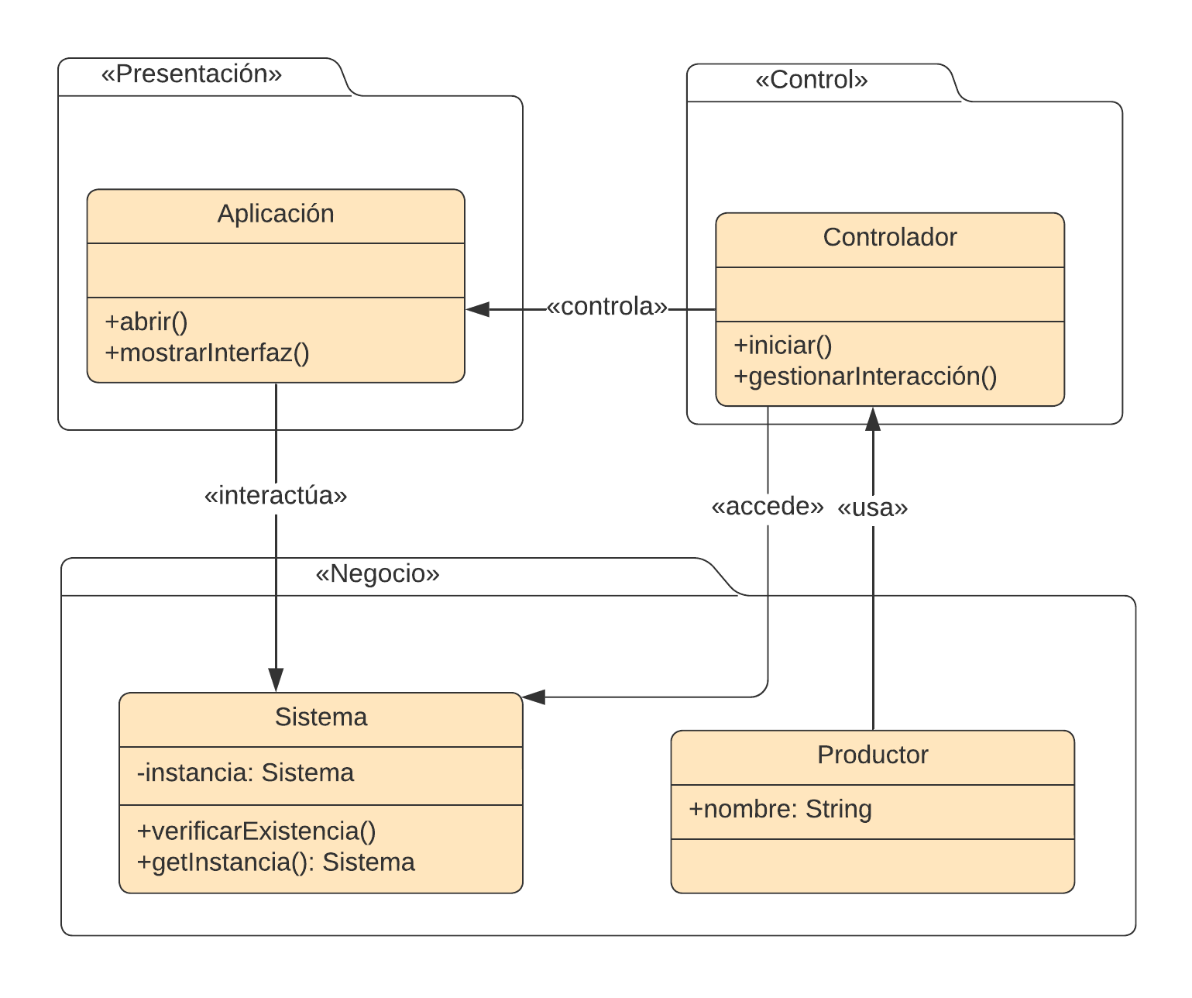


Ilustración 4: Diagrama de objetos | Caso de uso 1

* **Capa de Presentación:**
  + **Clase Aplicación:**
    - **Métodos:**
      * +abrir(): Método público que se utiliza para iniciar la aplicación.
      * +mostrarInterfaz(): Método público que se encarga de mostrar la interfaz de usuario al usuario final.
    - **Función:** Esta clase es responsable de la interacción con el usuario. Se encarga de presentar la información y recibir las entradas del usuario.
* **Capa de Control:**
  + **Clase Controlador:**
    - **Métodos:**
      * +iniciar(): Método público que inicia el proceso de control, inicializando la aplicación o configurando el entorno.
      * +gestionarInteracción(): Método público que maneja las interacciones del usuario, actuando como intermediario entre la interfaz de usuario y la lógica de negocio.
    - **Función:** Esta clase actúa como un intermediario entre la capa de presentación y la capa de negocio. Su responsabilidad principal es gestionar la lógica de control y coordinar las acciones entre las diferentes capas.
* **Capa de Negocio:**
  + **Clase Sistema:**
    - **Métodos:**
      * +verificarExistencia(): Método público que probablemente se utiliza para verificar la existencia de un elemento o entidad en el sistema.
      * -instancia: Sistema: Atributo privado que sugiere que esta clase puede estar implementando el patrón *Singleton*, donde solo puede existir una instancia de Sistema.
      * +getInstancia(): Sistema: Método público que devuelve la única instancia de Sistema.
    - **Función:** Esta clase contiene la lógica de negocio del sistema. Se encarga de las operaciones fundamentales y de la gestión de datos.
  + **Clase Productor:**
    - **Atributos:**
      * +nombre: String: Atributo público que representa el nombre del productor.
    - **Función:** Esta clase representa una entidad del negocio, que puede ser utilizada por el sistema para realizar operaciones relacionadas con los productores.

**Relaciones entre Clases**

* **Productor --> Controlador: <<usa>>:**
  + Indica que la clase Controlador utiliza la clase Productor. Esto sugiere que el controlador necesita acceder a la información o los métodos de Productor para realizar su función.
* **Controlador --> Aplicación: <<controla>>:**
  + Indica que el Controlador controla la Aplicación. Esto significa que el controlador gestiona la lógica de la interfaz de usuario y coordina las acciones que se deben realizar en respuesta a las interacciones del usuario.
* **Controlador --> Sistema: <<accede>>:**
  + Indica que el Controlador accede a la clase Sistema. Esto implica que el controlador puede invocar métodos de Sistema para realizar operaciones de negocio.
* **Aplicación --> Sistema: <<interactúa>>:**
  + Indica que la Aplicación interactúa con Sistema. Esto sugiere que la interfaz de usuario puede llamar a métodos de Sistema para obtener datos o realizar acciones.

### CASO DE USO <Seleccionar plantilla>

#### Breve descripción

El productor selecciona una plantilla base, una plantilla del día anterior o importa de Histórico. Este caso de uso permite al productor elegir la plantilla adecuada para su trabajo.

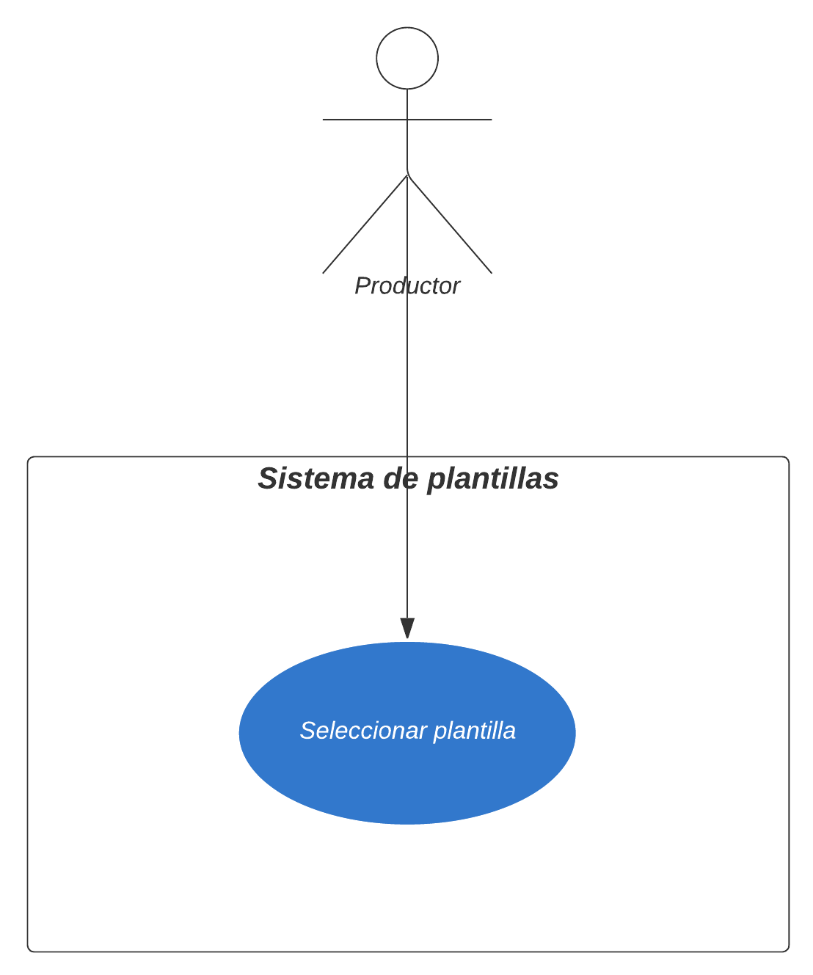


Ilustración 5: Caso de uso 2 | Seleccionar plantilla

#### Eventos de flujo

* El productor selecciona la opción de plantilla base, plantilla del día anterior o importar de Histórico.
* El sistema verifica la disponibilidad de la opción seleccionada.
* El sistema carga la plantilla seleccionada.

#### Diagramas de interacción

* Actor:
  + Productor: Representa a un usuario o entidad externa que interactúa con el sistema. En este contexto, el productor es quien selecciona y utiliza las plantillas disponibles en el sistema.
* Participante:
  + Sistema de Plantillas (Plantillas): Representa el sistema que gestiona las plantillas. Este sistema es responsable de ofrecer opciones al productor y cargar la plantilla seleccionada.

Interacciones en el Diagrama

* Productor -> Plantillas: Seleccionar plantilla:
  + El productor inicia la interacción al seleccionar una plantilla del sistema. Este es el primer paso en el proceso, donde el productor expresa su intención de trabajar con una plantilla específica.
* Plantillas -> Productor: Mostrar opciones:
  + En respuesta a la selección del productor, el sistema de plantillas muestra las opciones disponibles. Esto implica que el sistema presenta al productor una lista o un conjunto de plantillas que puede elegir.
* Productor -> Plantillas: Elegir opción:
  + Después de revisar las opciones, el productor elige una de las plantillas disponibles. Este paso indica que el productor ha tomado una decisión sobre qué plantilla utilizar.
* Plantillas -> Productor: Cargar plantilla:
  + Finalmente, el sistema de plantillas carga la plantilla seleccionada y la presenta al productor. Este paso completa el proceso, permitiendo al productor acceder a la plantilla elegida para su uso.
    1. *Diagrama de secuencia*

En este diagrama se ilustra la interacción entre un actor (en este caso, el “Productor”) y un sistema (denominado “Sistema de Plantillas”). Este tipo de diagrama se utiliza para mostrar cómo los objetos interactúan en un sistema a lo largo del tiempo, destacando el orden de los mensajes intercambiados.

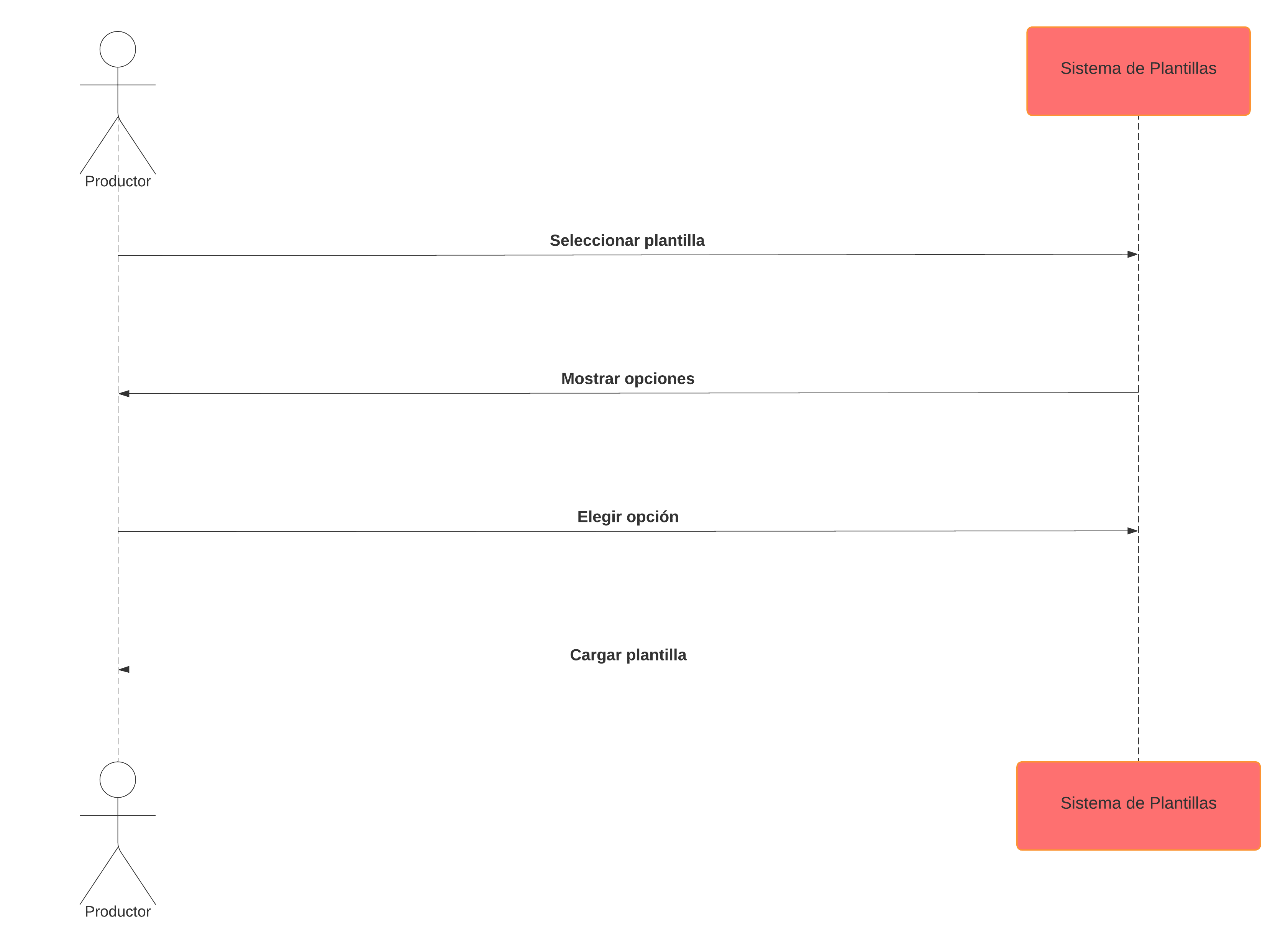


Ilustración 6: Diagrama de secuencia | Caso de uso 2

#### Objetos participantes

* Productor.
* Sistema de plantillas

#### 2.5 Diagrama de objetos

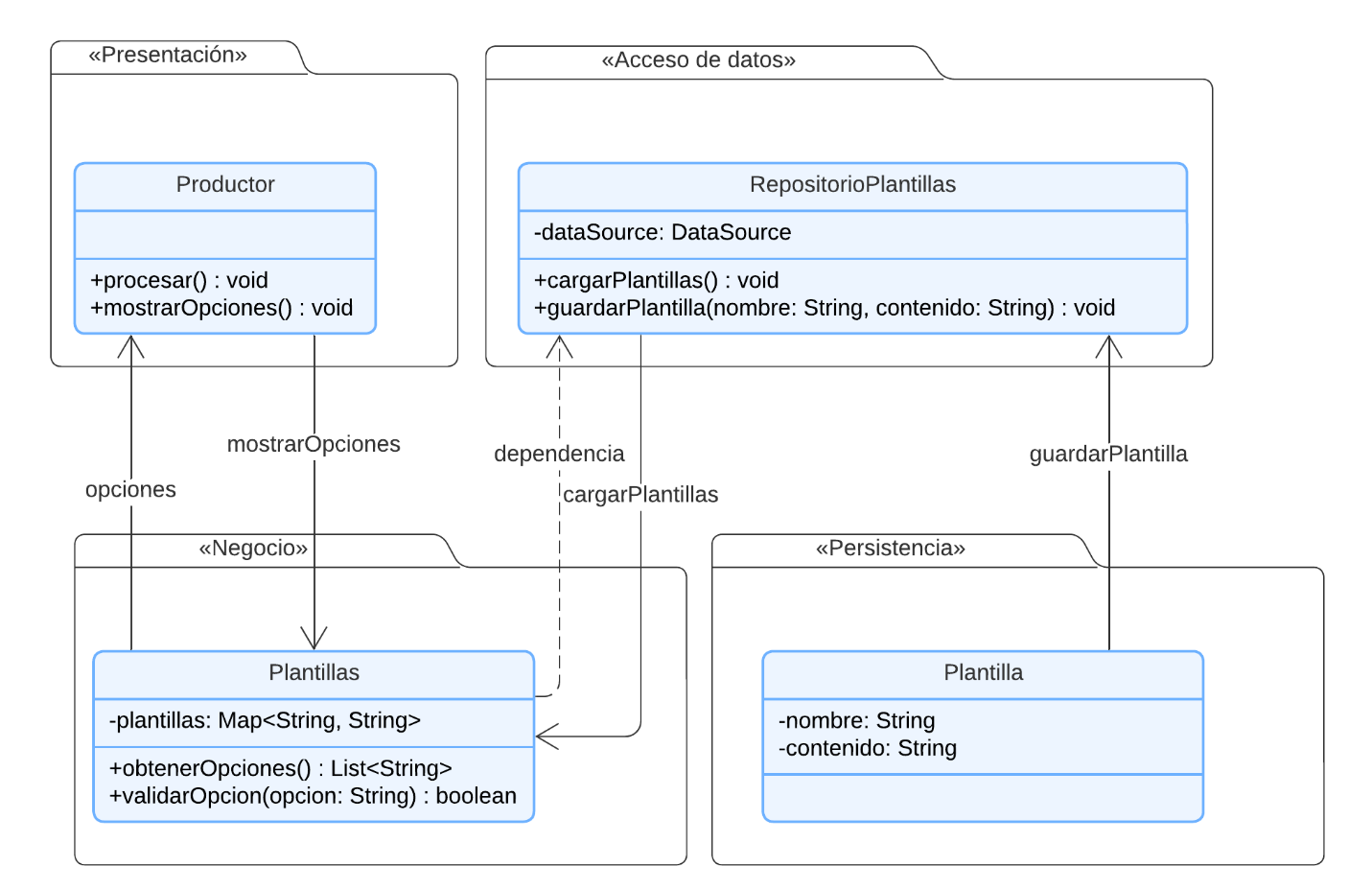


Ilustración 8: Diagrama de objetos | Caso de uso 2

* **Capa de Presentación:**
  + **Productor:**
    - Métodos:
      * procesar(): void: Método que inicia el proceso de selección de plantillas.
      * mostrarOpciones(): void: Método que permite al productor ver las opciones de plantillas disponibles.
* **Capa de Negocio:**
  + **Plantillas:**
    - Atributos:
      * plantillas: Map <String, String>: Atributo que llama al entorno de plantillas disponibles.
    - Métodos:
      * seleccionar(): void: Método que permite al productor seleccionar una plantilla.
      * obtenerOpciones(): List<String>: Método que devuelve una lista de opciones de plantillas disponibles para el productor.
      * validarOpcion(opción: String): boolean: Método que valida la selección de la plantilla.
* **Capa de Acceso a Datos:**
  + **RepositorioPlantillas:**
    - Atributos:
      * dataSource: DataSource: Atributo que llama la conexión a la base de datos del contenido de las plantillas disponibles.
    - Métodos:
      * obtenerPlantillas(): List<String>: Método que obtiene la lista de plantillas desde la base de datos o fuente de datos.
      * guardarPlantilla(plantilla: String): void: Método que guarda una plantilla en la base de datos o fuente de datos.
* **Capa de Persistencia:**
  + **Plantilla:**
    - Atributos:
      * nombre: String: Atributo que permite el almacenamiento del nombre de la plantilla.
      * contenido: String: Atributo que permite el almacenamiento del contenido de la plantilla.

**Relaciones entre Clases**

* **Relación entre Productor y Plantillas:**

**Tipo de relación: Asociación**

**Descripción:**El Productor inicia la interacción con la clase Plantillas al llamar al método obtenerOpciones(). Esta relación indica que el Productor depende de la clase Plantillas para realizar su tarea de selección de plantillas. En términos de diseño, esto significa que el Productor tiene conocimiento de la existencia de Plantillas y puede invocar sus métodos para interactuar con el sistema de plantillas.

* **Relación entre Plantillas y RepositorioPlantillas:**

**Tipo de relación: Asociación**

**Descripción:**La clase Plantillas se comunica con la clase RepositorioPlantillas/cargarPlantillas(): void para obtener la lista de plantillas disponibles mediante el método obtenerOpciones(). Esta relación indica que Plantillas necesita acceder a los datos almacenados en RepositorioPlantillas para poder mostrar las opciones al Productor. Aquí, Plantillas actúa como un intermediario que utiliza los datos proporcionados por RepositorioPlantillas para cumplir con su funcionalidad.

**Detalle de las Relaciones**

* **Asociación:**

La asociación es una relación estructural que indica que una clase utiliza o tiene una referencia a otra clase. En este caso, tanto Productor como Plantillas están asociados, ya que el Productor necesita interactuar con Plantillas para seleccionar y cargar plantillas. Asimismo, Plantillas está asociada con RepositorioPlantillas para acceder a los datos necesarios.

* **Flujo de Interacción**

El Productor inicia la selección:

Cuando el Productor llama al método mostrarOpciones(), se establece la interacción con la clase Plantillas. Esto puede implicar que el Productor solicita ver las opciones de plantillas disponibles.

* **Plantillas obtiene las opciones:**

La clase Plantillas invoca el método obtenerOpciones() de RepositorioPlantillas para recuperar la lista de plantillas. Esta acción es crucial, ya que Plantillas no tiene los datos por sí misma; depende de RepositorioPlantillas para obtener la información necesaria.

* **Mostrar opciones al Productor:**

Una vez que Plantillas tiene la lista de plantillas, utiliza el método mostrarOpciones() para presentar estas opciones al Productor. Esto completa la interacción inicial.

* **Selección y carga de la plantilla:**

Después de que el Productor elige una opción, llama al método validarOpciones() en Plantillas, que a su vez puede implicar que Plantillas interactúe nuevamente con RepositorioPlantillas para cargar la plantilla seleccionada, utilizando el método cargarPlantillas() y; luego, la función de salvado va almacenando los datos en plantilla, en la capa de Persistencia.

### CASO DE USO <Editar escaleta>

#### Breve descripción

El productor edita una escaleta con una estructura similar a Excel. Este caso de uso permite al productor modificar la escaleta según sus necesidades.

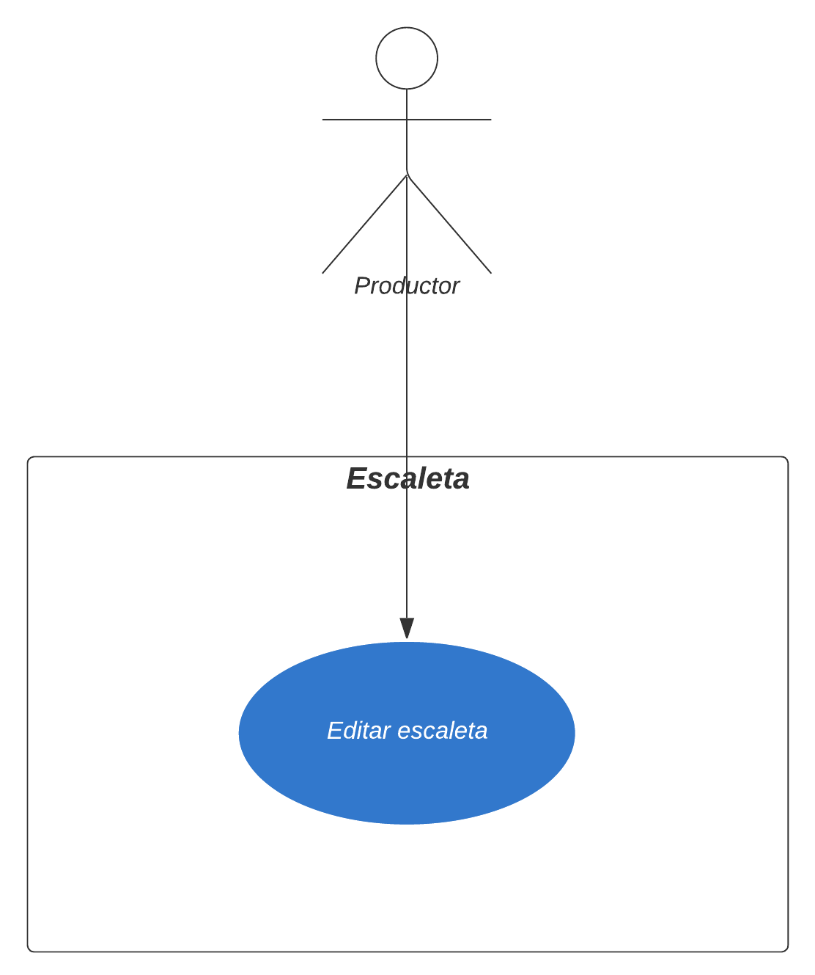


Ilustración 5: Caso de uso 3 | Editar escaleta

#### Eventos de flujo

* El productor selecciona la opción de plantilla base, plantilla del día anterior o importar de Histórico.
* El sistema verifica la disponibilidad de la opción seleccionada.
* El sistema carga la plantilla seleccionada.

#### Diagramas de interacción

* Actor:
  + Productor: Representa a un usuario o entidad externa que interactúa con el sistema. En este contexto, el productor es quien selecciona y utiliza las plantillas disponibles en el sistema.
* Participante:
  + Sistema de Plantillas (Plantillas): Representa el sistema que gestiona las plantillas. Este sistema es responsable de ofrecer opciones al productor y cargar la plantilla seleccionada.

Interacciones en el Diagrama

* Productor -> Plantillas: Seleccionar plantilla:
  + El productor inicia la interacción al seleccionar una plantilla del sistema. Este es el primer paso en el proceso, donde el productor expresa su intención de trabajar con una plantilla específica.
* Plantillas -> Productor: Mostrar opciones:
  + En respuesta a la selección del productor, el sistema de plantillas muestra las opciones disponibles. Esto implica que el sistema presenta al productor una lista o un conjunto de plantillas que puede elegir.
* Productor -> Plantillas: Elegir opción:
  + Después de revisar las opciones, el productor elige una de las plantillas disponibles. Este paso indica que el productor ha tomado una decisión sobre qué plantilla utilizar.
* Plantillas -> Productor: Cargar plantilla:
  + Finalmente, el sistema de plantillas carga la plantilla seleccionada y la presenta al productor. Este paso completa el proceso, permitiendo al productor acceder a la plantilla elegida para su uso.
    1. *Diagrama de secuencia*

En este diagrama se ilustra la interacción entre un actor (en este caso, el “Productor”) y un sistema (denominado “Sistema de Plantillas”). Este tipo de diagrama se utiliza para mostrar cómo los objetos interactúan en un sistema a lo largo del tiempo, destacando el orden de los mensajes intercambiados.

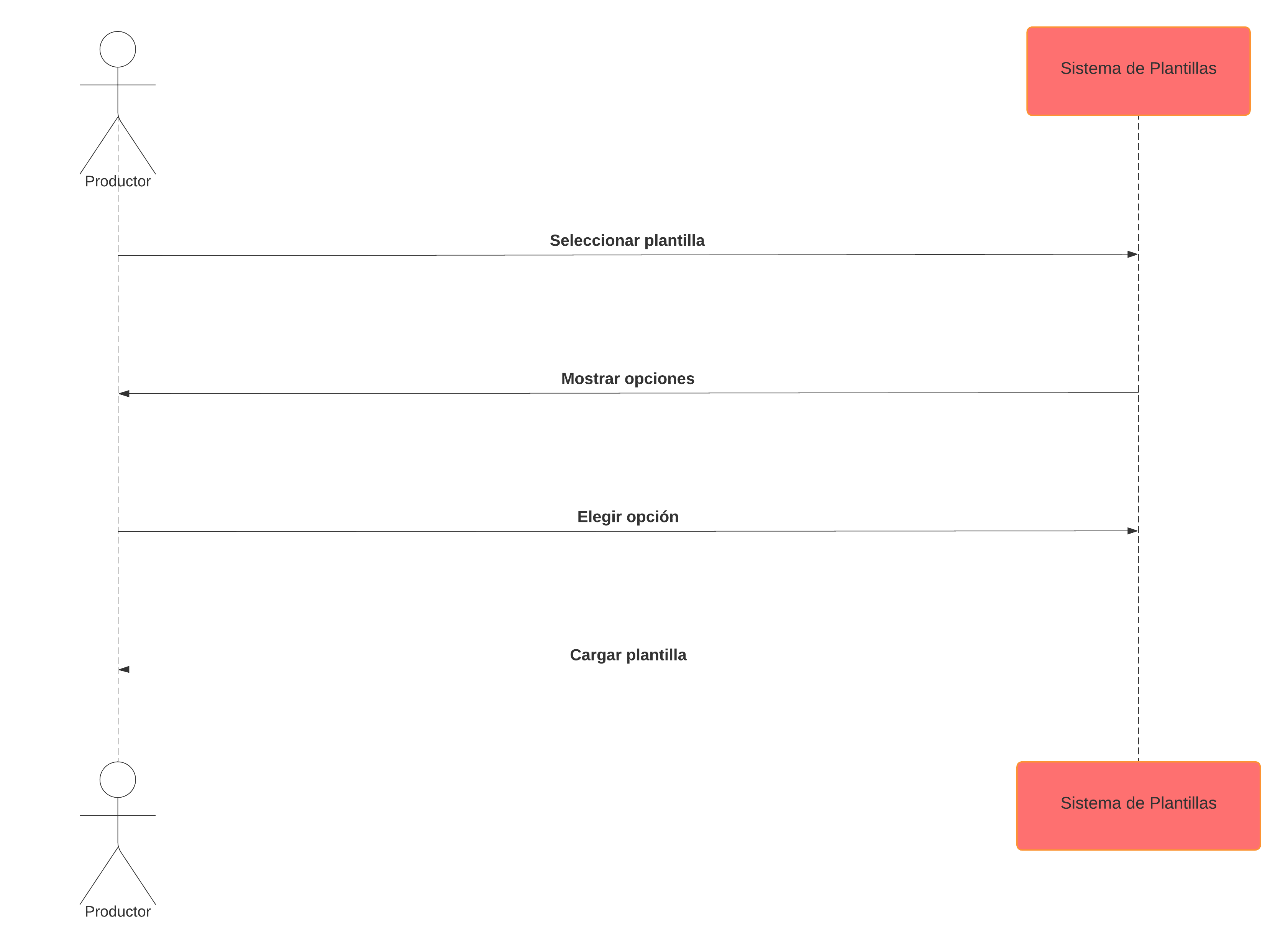


Ilustración 6: Diagrama de secuencia | Caso de uso 2

#### Objetos participantes

* Productor.
* Sistema de plantillas

#### Diagrama de objetos

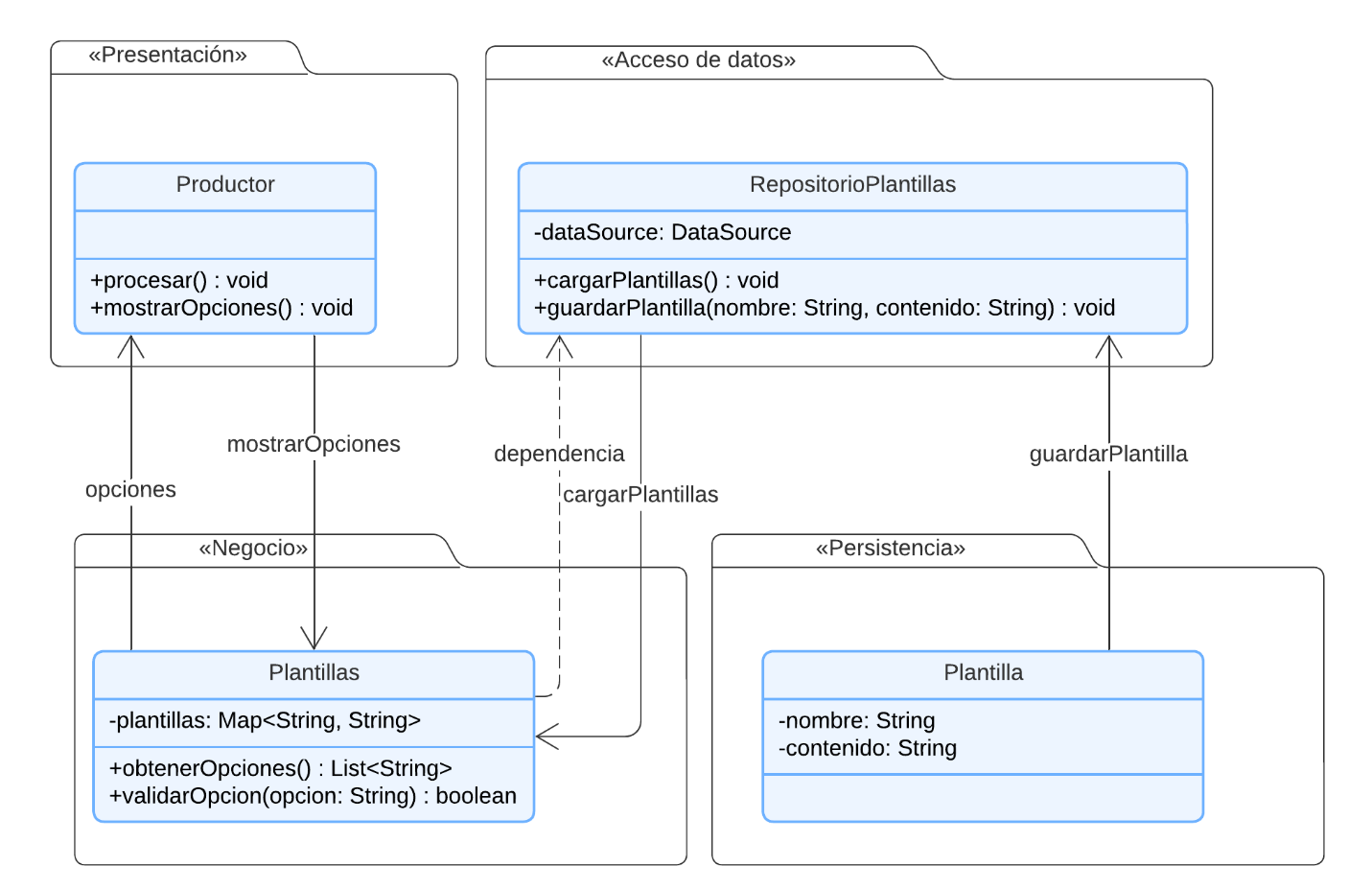


Ilustración 8: Diagrama de objetos | Caso de uso 2

* **Capa de Presentación:**
  + **Productor:**
    - Métodos:
      * procesar(): void: Método que inicia el proceso de selección de plantillas.
      * mostrarOpciones(): void: Método que permite al productor ver las opciones de plantillas disponibles.
* **Capa de Negocio:**
  + **Plantillas:**
    - Atributos:
      * plantillas: Map <String, String>: Atributo que llama al entorno de plantillas disponibles.
    - Métodos:
      * seleccionar(): void: Método que permite al productor seleccionar una plantilla.
      * obtenerOpciones(): List<String>: Método que devuelve una lista de opciones de plantillas disponibles para el productor.
      * validarOpcion(opción: String): boolean: Método que valida la selección de la plantilla.
* **Capa de Acceso a Datos:**
  + **RepositorioPlantillas:**
    - Atributos:
      * dataSource: DataSource: Atributo que llama la conexión a la base de datos del contenido de las plantillas disponibles.
    - Métodos:
      * obtenerPlantillas(): List<String>: Método que obtiene la lista de plantillas desde la base de datos o fuente de datos.
      * guardarPlantilla(plantilla: String): void: Método que guarda una plantilla en la base de datos o fuente de datos.

**Relaciones entre Clases**

* **Relación entre Productor y Plantillas:**

**Tipo de relación: Asociación**

**Descripción:**El Productor inicia la interacción con la clase Plantillas al llamar al método obtenerOpciones(). Esta relación indica que el Productor depende de la clase Plantillas para realizar su tarea de selección de plantillas. En términos de diseño, esto significa que el Productor tiene conocimiento de la existencia de Plantillas y puede invocar sus métodos para interactuar con el sistema de plantillas.

* **Relación entre Plantillas y RepositorioPlantillas:**

**Tipo de relación: Asociación**

**Descripción:**La clase Plantillas se comunica con la clase RepositorioPlantillas/cargarPlantillas(): void para obtener la lista de plantillas disponibles mediante el método obtenerOpciones(). Esta relación indica que Plantillas necesita acceder a los datos almacenados en RepositorioPlantillas para poder mostrar las opciones al Productor. Aquí, Plantillas actúa como un intermediario que utiliza los datos proporcionados por RepositorioPlantillas para cumplir con su funcionalidad.

**Detalle de las Relaciones**

* **Asociación:**

La asociación es una relación estructural que indica que una clase utiliza o tiene una referencia a otra clase. En este caso, tanto Productor como Plantillas están asociados, ya que el Productor necesita interactuar con Plantillas para seleccionar y cargar plantillas. Asimismo, Plantillas está asociada con RepositorioPlantillas para acceder a los datos necesarios.

* **Flujo de Interacción**

El Productor inicia la selección:

Cuando el Productor llama al método mostrarOpciones(), se establece la interacción con la clase Plantillas. Esto puede implicar que el Productor solicita ver las opciones de plantillas disponibles.

* **Plantillas obtiene las opciones:**

La clase Plantillas invoca el método obtenerOpciones() de RepositorioPlantillas para recuperar la lista de plantillas. Esta acción es crucial, ya que Plantillas no tiene los datos por sí misma; depende de RepositorioPlantillas para obtener la información necesaria.

* **Mostrar opciones al Productor:**

Una vez que Plantillas tiene la lista de plantillas, utiliza el método mostrarOpciones() para presentar estas opciones al Productor. Esto completa la interacción inicial.

* **Selección y carga de la plantilla:**

Después de que el Productor elige una opción, llama al método validarOpciones() en Plantillas, que a su vez puede implicar que Plantillas interactúe nuevamente con RepositorioPlantillas para cargar la plantilla seleccionada, utilizando el método cargarPlantillas() y; luego, la función de salvado va almacenando los datos en plantilla, en la capa de Persistencia.