



# **UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO**

Facultad de ciencias de la ingeniería



## **ESTUDIANTE:**

CHICA VALFRE VALESKA SOFIA

LETURNE PLUAS JHON BYRON

## **CARRERA:**

Ingeniería de software

## **CURSO:**

7to Software "A"

## **ASIGNATURA:**

Aplicaciones distribuidas

## **DOCENTE:**

Ing. Guerrero Ulloa Gleiston Cicerón, MSc

## **TEMA:**

Desarrollo de Aplicaciones en Ambientes Distribuidos

## Contenido

1. Introducción.....	7
2. Diagramas UML .....	8
2.1. Fundamentos de Diagramas UML .....	8
2.1.1. Conceptos básicos.....	8
2.1.2. Elementos principales de UML .....	8
2.2. Modelamiento de sistemas distribuidos .....	9
2.2.1. Ventajas del modelamiento con UML en sistemas distribuidos .....	9
2.2.2. Consideraciones para el modelamiento de sistemas distribuidos .....	9
2.2.3. Ejemplos de modelamiento de sistemas distribuidos con UML .....	10
2.3. Herramientas y tecnologías para el modelamiento con UML en sistemas distribuidos .....	10
2.3.1. Herramientas de software para el modelamiento con UML .....	10
2.3.2. Integración de UML con tecnologías de sistemas distribuidos.....	11
2.3.3. Consideraciones de implementación y despliegue de modelos BPMN en sistemas distribuidos.....	11
2.3.4. Notación y símbolos utilizados en UML .....	12
2.4. Ejemplos de notaciones .....	15
2.4.1. Diagrama de clases .....	15
2.4.2. Diagrama de secuencia .....	17
2.4.3. Diagrama de secuencia .....	19
3. Diagramas BPMN.....	22
3.1. Fundamentos de Diagramas BPMN .....	22
3.1.1. Conceptos básicos.....	22
3.1.2. Elementos principales de BPMN .....	22
3.2. Modelamiento de sistemas distribuidos .....	23
3.2.1. Ventajas del modelamiento con BPMN en sistemas distribuidos.....	23
3.2.2. Consideraciones para el modelamiento de sistemas distribuidos .....	24
3.2.3. Ejemplos de modelamiento de sistemas distribuidos con BPMN .....	24
3.3. Herramientas y tecnologías para el modelamiento con BPMN en sistemas distribuidos.....	25
3.3.1. Herramientas de software para el modelamiento con BPMN.....	25
3.3.2. Integración de BPMN con tecnologías de sistemas distribuidos .....	25
3.3.3. Consideraciones de implementación y despliegue de modelos BPMN en sistemas distribuidos.....	25
3.3.4. Notación y símbolos utilizados en BPMN .....	26

3.4.	Ejemplos de notaciones .....	30
3.4.1.	Eventos.....	30
3.4.2.	Tareas .....	32
3.4.3.	Puertas de enlace.....	33
3.4.4.	Símbolo basado en eventos.....	34
3.4.5.	Artefactos.....	35
3.5.	Ejercicios.....	36
3.5.1.	Problema: Sistema de Control de Tráfico Urbano Inteligente .....	36
3.5.2.	Modelo .....	38
3.6.	Problema: Evaluación de solicitud de créditos .....	38
3.6.1.	Modelo .....	39
3.7.	Problema: gestión de reservas de un hotel .....	39
3.7.1.	Modelo .....	41
4.	Diagrama Petri.....	42
4.1.	Fundamentos de Diagramas Petri.....	42
4.1.1.	Conceptos básicos .....	42
4.1.2.	Elementos principales de Petri.....	42
4.2.	Modelamiento de sistemas distribuidos .....	43
4.2.1.	Ventajas del modelamiento con Petri en sistemas distribuidos.....	43
4.2.2.	Consideraciones para el modelamiento de sistemas distribuidos .....	43
4.2.3.	Ejemplos de modelamiento de sistemas distribuidos con Petri .....	44
4.3.	Herramientas y tecnologías para el modelamiento con Petri en sistemas distribuidos	45
4.3.1.	Herramientas de software para el modelamiento con Petri .....	45
4.3.2.	Integración de Petri con tecnologías de sistemas distribuidos.....	45
4.3.3.	Consideraciones de implementación y despliegue de modelos Petri en sistemas distribuidos .....	46
4.3.4.	Notación y símbolos utilizados en Petri.....	47
4.4.	Ejemplos de notaciones.....	47
4.4.1.	Lugares (Places).....	47
4.4.2.	Transiciones (Transitions).....	47
4.4.3.	Tokens .....	48
4.4.4.	Arcos (Arcs).....	48
5.	Desarrollo de Sistemas basados en IoT .....	49
5.1.	Metodología de Desarrollo.....	49

5.1.1.	Selección y aplicación de metodologías de desarrollo para proyectos basados en IoT.	49
5.1.2.	Consideración de aspectos como la adquisición de datos, el procesamiento en la nube y la interconexión de dispositivos.....	49
5.2.	Alternativas de Solución .....	50
5.2.1.	Exploración de alternativas de solución para la implementación de sistemas IoT	50
5.2.2.	Uso de plataformas y tecnologías como Arduino, Raspberry Pi, y servicios en la nube para IoT (por ejemplo, AWS IoT, Azure IoT, Google Cloud IoT).....	50
6.	Redes de Petri practica .....	51
6.1.	Problema: Evaluación de solicitud de créditos .....	51
6.1.1.	Modelo .....	51
6.1.2.	Solución .....	51
6.2.	Problema: proceso de fabricación de un producto genérico. ....	53
6.2.1.	Modelo .....	55
6.3.	Solución.....	55
6.4.	GitHub.....	58
7.	Conclusión .....	59
8.	Bibliografía.....	60

## Ilustraciones

Ilustración 1 Clases .....	15
Ilustración 2 Relaciones: Asociación .....	15
Ilustración 3 Relaciones: Agregación .....	16
Ilustración 4 Relaciones: Composición .....	16
Ilustración 5 Relaciones: Herencia .....	16
Ilustración 6 Relaciones: Dependencia .....	17
Ilustración 7 Atributos.....	17
Ilustración 8 Objetos .....	17
Ilustración 9 Líneas de vida .....	18
Ilustración 10 Mensaje .....	18
Ilustración 11 Barra de activación .....	19
Ilustración 12 Nodos de actividad: Actividad.....	19
Ilustración 13 Nodos de actividad: Transiciones .....	19
Ilustración 14 Nodos de control: Nodo inicial.....	20
Ilustración 15 Nodos de control: nodo final .....	20
Ilustración 16 Nodos de control: Nodo de decisión.....	20
Ilustración 17 Sincronización: Nodo de bifurcación .....	21
Ilustración 18 Sincronización: Nodo de unión.....	21
Ilustración 19 Objetos .....	21
Ilustración 20 Particiones.....	22
Ilustración 21 Evento de inicio de mensaje .....	30
Ilustración 22 Evento de envío y recepción de mensaje .....	31
Ilustración 23 Evento de inicio de temporizador .....	31
Ilustración 24 Evento intermedio de temporizador.....	31
Ilustración 25 Tarea genérica .....	32
Ilustración 26 Tarea de usuario .....	32
Ilustración 27 Tarea de servicio .....	32
Ilustración 28 Tarea de script.....	32
Ilustración 29 Tarea regla de negocio .....	33
Ilustración 30 Exclusivo divergente.....	33
Ilustración 31 Exclusivo convergente .....	33
Ilustración 32 Paralelo divergente.....	34
Ilustración 33 Paralelo convergente.....	34
Ilustración 34 Símbolo basado en eventos.....	35
Ilustración 35 Data store y Data object.....	35
Ilustración 36 Pool .....	36
Ilustración 37 Grupos.....	36
Ilustración 38 Diagrama BPMN de Sistema de Control de Tráfico Urbano Inteligente .....	38
Ilustración 39 Evaluación de solicitud de créditos .....	39
Ilustración 40 Gestión de reservas de un hotel .....	41
Ilustración 41 Lugares.....	47
Ilustración 42 Transiciones .....	48
Ilustración 43 Tokens.....	48
Ilustración 44 Arcos .....	48
Ilustración 45 Modelo de solicitud de creditos .....	51

Ilustración 46 Acciones primarias de solicitud de documentos .....	52
Ilustración 47 Bifurcación de entrega de documentos .....	52
Ilustración 48 Recepción de documento .....	52
Ilustración 49 Documentos no receptados .....	53
Ilustración 50 Agrupación de proveedores y bodega .....	56
Ilustración 51 Zona de sub-ensamble .....	56
Ilustración 52 Zona de producto terminado .....	57
Ilustración 53 Bodega producto ensamblado .....	57
Ilustración 54 Cambio de parámetros .....	58

## **Tablas**

Tabla 1 Notaciones para diagrama de clases.....	12
Tabla 2 Notaciones para diagrama de secuencia.....	13
Tabla 3 Notaciones diagrama BPMN .....	26
Tabla 4 Notaciones diagrama de petri.....	47
Tabla 5 Parámetros de la práctica de red de petri .....	54

## **1. Introducción**

El desarrollo de aplicaciones en ambientes distribuidos ha experimentado una transformación significativa en las últimas décadas, impulsada por avances revolucionarios en las tecnologías de la información [1]. Esta evolución ha sido fundamental para el surgimiento y la expansión de sistemas que son intrínsecamente complejos, requeridos para operar a través de diferentes plataformas y entornos geográficamente dispersos. La interconexión global y la capacidad de computación en la nube han remodelado no solo cómo diseñamos y desplegamos aplicaciones, sino también cómo interactuamos con la tecnología en nuestra vida diaria [2].

La computación en la nube, por ejemplo, ha proporcionado una base sólida para el desarrollo de aplicaciones en ambientes distribuidos, al ofrecer recursos de computación virtualmente ilimitados, escalabilidad y flexibilidad. Esto ha permitido a las organizaciones desplegar y escalar aplicaciones sin las limitaciones de la infraestructura física tradicional [1]. Los modelos de servicio como IaaS (Infraestructura como Servicio), PaaS (Plataforma como Servicio) y SaaS (Software como Servicio) han democratizado el acceso a la tecnología, permitiendo a empresas de todos los tamaños beneficiarse de las economías de escala y reducir los tiempos de lanzamiento al mercado [3].

La programación concurrente y paralela, por su parte, ha permitido aprovechar al máximo los recursos de hardware disponibles, mejorando significativamente el rendimiento de las aplicaciones distribuidas. Al permitir que múltiples procesos o hilos se ejecuten simultáneamente, estas técnicas de programación han sido fundamentales para desarrollar aplicaciones que pueden manejar grandes volúmenes de datos y solicitudes de usuarios de manera eficiente[2].

Sin embargo, el desarrollo en ambientes distribuidos también presenta desafíos notables, especialmente en términos de gestión de la consistencia de datos, latencia de red y seguridad. Asegurar la coherencia en sistemas distribuidos donde los datos se replican a través de múltiples nodos requiere estrategias complejas de sincronización y conciliación [1]. La latencia de red, por otro lado, puede afectar negativamente la experiencia del usuario y el rendimiento de la aplicación, lo que requiere un diseño cuidadoso para minimizar los retrasos. Además, la distribución de componentes de software aumenta la superficie de ataque, planteando desafíos significativos en la protección de datos y la prevención de accesos no autorizados [4].

Mirando hacia el futuro, es evidente que las tecnologías emergentes como la inteligencia artificial, el Internet de las cosas (IoT) y el blockchain continuarán impulsando la innovación

en el desarrollo de aplicaciones en ambientes distribuidos. Estas tecnologías no solo ofrecerán nuevas oportunidades para mejorar la eficiencia y la funcionalidad de las aplicaciones distribuidas, sino que también plantearán desafíos adicionales, especialmente en lo que respecta a la integración de sistemas, la gestión de la privacidad de los datos y la seguridad [3].

## **2. Diagramas UML**

### **2.1. Fundamentos de Diagramas UML**

#### **2.1.1. Conceptos básicos**

Los Diagramas UML (Lenguaje Unificado de Modelado, por sus siglas en inglés, Unified Modeling Language) son una herramienta estándar utilizada en la ingeniería de software para visualizar, especificar, construir y documentar los artefactos de sistemas de software. UML no es un lenguaje de programación, sino un lenguaje visual que sirve para modelar las estructuras y comportamientos de los sistemas de software, incluyendo sus componentes, relaciones, y cómo interactúan entre sí [5].

UML se compone de una variedad de tipos de diagramas, cada uno diseñado para representar diferentes aspectos y perspectivas de los sistemas de software. Estos diagramas se dividen principalmente en dos categorías: diagramas de estructura y diagramas de comportamiento [6].

#### **2.1.2. Elementos principales de UML**

Los diagramas de UML (Lenguaje Unificado de Modelado) son herramientas esenciales en el desarrollo de software, que permiten visualizar, especificar, construir y documentar los diversos aspectos de un sistema [5]. Entre los elementos principales de UML se encuentran los diagramas de clase, secuencia y actividad, cada uno abordando distintas facetas del modelado de sistemas [6]. Los diagramas de clase son fundamentales para representar la estructura estática del sistema, mostrando las clases, sus atributos, métodos y las relaciones entre ellas, como la herencia y las asociaciones. Esto permite una comprensión clara de cómo se organiza el sistema y cómo interactúan sus componentes a nivel de estructura [7].

Por otro lado, los diagramas de secuencia se centran en la interacción entre objetos a lo largo del tiempo, ilustrando la secuencia de mensajes y llamadas para realizar operaciones específicas [6]. Este tipo de diagrama es crucial para entender el flujo de comunicación y la dinámica de colaboración entre los distintos objetos del sistema, destacando la importancia del orden y el tiempo en el que ocurren las interacciones [7].



Finalmente, los diagramas de actividad se utilizan para modelar el flujo de control o el flujo de trabajo dentro de las operaciones del sistema, ofreciendo una vista similar a un diagrama de flujo que detalla la secuencia de actividades y acciones. Estos diagramas son especialmente útiles para representar procesos de negocio y flujos de trabajo, mostrando cómo se mueve el control a través de las actividades y destacando los puntos de decisión que afectan el flujo del proceso [5].

## **2.2. Modelamiento de sistemas distribuidos**

### **2.2.1. Ventajas del modelamiento con UML en sistemas distribuidos**

El modelado con UML en sistemas distribuidos ofrece significativas ventajas que son esenciales para el manejo eficiente de la complejidad inherente a estos sistemas [6]. Primero, proporciona un lenguaje visual común que mejora la comunicación y comprensión entre los miembros del equipo de desarrollo y otras partes interesadas, lo cual es crucial dada la naturaleza descentralizada y la complejidad de la arquitectura de los sistemas distribuidos. Esto permite una representación clara de la estructura y el comportamiento del sistema, facilitando la identificación y solución temprana de problemas potenciales [7].

Además, UML promueve un diseño estructurado y modular, vital para el desarrollo paralelo y la escalabilidad de sistemas distribuidos. También mejora la documentación y el mantenimiento del sistema, permitiendo actualizaciones claras y coherentes que son fundamentales para la integración de nuevas tecnologías y componentes a lo largo del tiempo. En resumen, el uso de UML en sistemas distribuidos simplifica la abstracción, visualización, y gestión de la complejidad, facilitando así el desarrollo, prueba, y evolución de estos sistemas [8].

### **2.2.2. Consideraciones para el modelamiento de sistemas distribuidos**

Al modelar sistemas distribuidos con UML, es esencial considerar varios aspectos clave para abordar su complejidad única. Primero, la definición clara de interfaces y la interacción entre componentes son cruciales para asegurar comunicaciones efectivas y colaboración entre los elementos del sistema. Esto implica un enfoque detallado en protocolos de comunicación y contratos de servicio[8].

La concurrencia y el manejo de estados también son vitales, requiriendo un modelado cuidadoso de la sincronización y políticas de consistencia para evitar problemas como condiciones de carrera. Además, la tolerancia a fallos y la recuperación ante errores deben ser contempladas para garantizar la alta disponibilidad y fiabilidad del sistema, incluyendo estrategias como redundancia de componentes y balanceo de carga [9].

Finalmente, la seguridad es una prioridad, con la necesidad de integrar mecanismos robustos para la protección de datos y comunicaciones, como cifrado y autenticación. Estas consideraciones son fundamentales para el éxito en el diseño y la implementación de sistemas distribuidos eficientes y seguros [8].

### **2.2.3. Ejemplos de modelamiento de sistemas distribuidos con UML**

El empleo de UML (Lenguaje Unificado de Modelado) en distintos sectores demuestra su capacidad para mejorar la arquitectura, interacción y mantenimiento de sistemas distribuidos complejos a través de ejemplos variados fuera del comercio electrónico, bancario y de salud [9].

En el ámbito de las telecomunicaciones, UML es fundamental para modelar sistemas que gestionan redes de comunicaciones, servicios de datos y telefonía. Al utilizar UML para diseñar la infraestructura y los flujos de trabajo, las empresas pueden optimizar la asignación de recursos, la gestión de fallos y la prestación de servicios a los clientes. Esto no solo mejora la eficiencia operativa sino también la calidad del servicio, asegurando una experiencia de usuario superior y una gestión efectiva de la red [10].

En la industria del software, especialmente en el desarrollo de aplicaciones móviles y en la nube, UML facilita la visualización de la estructura y el comportamiento de las aplicaciones a través de sus distintos componentes y servicios [9]. Modelar aplicaciones utilizando UML permite a los equipos de desarrollo identificar de manera proactiva problemas de diseño, dependencias entre componentes y optimizar la interacción usuario-aplicación. Esto resulta en aplicaciones más robustas, escalables y con una mejor experiencia de usuario final [10].

En el sector de la logística y la gestión de cadenas de suministro, UML ayuda a modelar los complejos sistemas de seguimiento y entrega, integrando diversos procesos desde el inventario hasta la distribución [11]. Al aplicar UML, las empresas pueden diseñar sistemas que mejoren la trazabilidad, eficiencia y coordinación entre los diferentes eslabones de la cadena de suministro. Esto conduce a una gestión más eficaz de los recursos, reducción de tiempos de entrega y una mayor satisfacción del cliente al asegurar la entrega oportuna de productos [12].

## **2.3. Herramientas y tecnologías para el modelamiento con UML en sistemas distribuidos**

### **2.3.1. Herramientas de software para el modelamiento con UML**

El modelado UML se beneficia de una amplia gama de herramientas de software, que varían desde opciones básicas hasta soluciones empresariales completas. Estas herramientas ofrecen

una serie de funcionalidades críticas para el diseño de diagramas UML, permitiendo la visualización, diseño y documentación de sistemas de software complejos. Entre las herramientas más reconocidas en el ámbito de UML se encuentran Enterprise Architect, Visual Paradigm, Lucidchart, StarUML, y MagicDraw, cada una con sus particularidades y beneficios [13].

Estas plataformas proporcionan interfaces de usuario gráficas y amigables que simplifican la creación de diagramas UML de forma intuitiva. Además, muchas de estas herramientas están equipadas con funcionalidades para la colaboración en equipo, permitiendo a los desarrolladores trabajar conjuntamente en el diseño y refinamiento de la arquitectura de software. Funciones como la gestión de versiones y la capacidad de integrarse con otros sistemas y herramientas de desarrollo de software son también aspectos relevantes que pueden determinar la selección de una herramienta específica [11].

### **2.3.2. Integración de UML con tecnologías de sistemas distribuidos**

La integración de UML en el desarrollo de sistemas distribuidos facilita significativamente el proceso de diseño y comprensión de estos complejos sistemas [12]. Mediante el uso de diagramas de clases y componentes, UML permite a los desarrolladores mapear de forma clara la estructura y distribución de los sistemas, optimizando la arquitectura para microservicios y contenedores. Además, los diagramas de secuencia y de actividad proporcionan una visión detallada de las interacciones y flujos de proceso, esenciales para la eficiencia operativa y la gestión de la comunicación entre servicios [13].

Por otro lado, la adopción de UML mejora la colaboración dentro de los equipos de desarrollo y actúa como una herramienta de documentación vital, ofreciendo un lenguaje común para una comprensión unificada del diseño del sistema. Esto no solo beneficia el mantenimiento a largo plazo, sino que también facilita la escalabilidad y la adaptabilidad del sistema a nuevas demandas. La compatibilidad de UML con herramientas de desarrollo modernas promueve un enfoque ágil, permitiendo que los cambios en el diseño se integren rápidamente en el código, acelerando así el ciclo de desarrollo y manteniendo la alineación entre el modelo UML y la implementación final [11].

### **2.3.3. Consideraciones de implementación y despliegue de modelos BPMN en sistemas distribuidos**

Al implementar y desplegar modelos UML en sistemas distribuidos, es crucial mantener una alta fidelidad entre el modelo y el sistema implementado, asegurando que los cambios y ajustes

se reflejen constantemente en el modelo UML. La elección de tecnologías y herramientas adecuadas que complementen la arquitectura diseñada en UML es fundamental para una implementación efectiva, enfocándose en aspectos como la escalabilidad, disponibilidad y tolerancia a fallos [13].

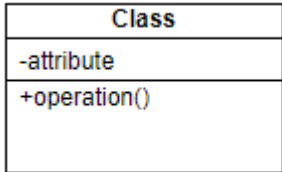


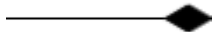
La automatización del despliegue y la gestión de configuraciones juegan un papel importante en la eficiencia y reducción de errores durante la implementación [14]. Herramientas de integración y despliegue continuo que se alineen con los modelos UML pueden facilitar estos procesos. Posteriormente, prácticas robustas de monitoreo y mantenimiento aseguran que el sistema funcione conforme a lo previsto, utilizando el modelo UML como guía para establecer métricas de rendimiento y puntos de monitoreo [15].



#### 2.3.4. Notación y símbolos utilizados en UML

La notación UML (Lenguaje Unificado de Modelado) se compone de varios símbolos y elementos gráficos diseñados para representar los diferentes aspectos de un sistema de software. UML incluye una amplia gama de diagramas, cada uno con su propio conjunto de símbolos, para modelar tanto la estructura estática como el comportamiento dinámico de los sistemas [14].

##### 2.3.4.1. Diagrama de clases

*Tabla 1 Notaciones para diagrama de clases*  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

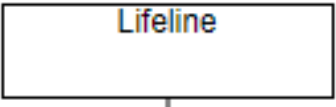



Tipo	Descripción	Notación
<b>Clases</b>	Se representan como rectángulos divididos en tres secciones	
<b>Relaciones</b>	<b>Asociación:</b> Representada por una línea simple entre dos clases.	
	<b>Agregación:</b> Esta relación se simboliza mediante un diamante vacío.	
	<b>Composición:</b> Representada por un diamante relleno en el extremo.	

	<b>Dependencia:</b> Se muestra como una línea punteada con una flecha que señala hacia la clase dependiente.	
	<b>Herencia:</b> Se simboliza con una línea que termina en un triángulo hueco apuntando hacia la clase base.	





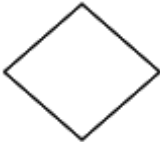
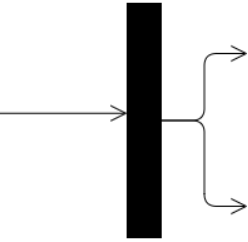
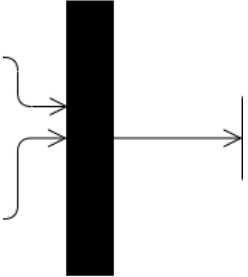
#### 2.3.4.2. Diagrama de secuencia

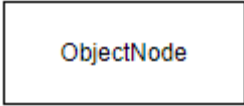

*Tabla 2 Notaciones para diagrama de secuencia*

*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

Tipo	Descripción	Notación
<b>Objetos</b>	Representados por rectángulos con el nombre del objeto, Se colocan en la parte superior del diagrama, indicando los participantes en la interacción.	
<b>Líneas de Vida</b>	Verticales, que se extienden hacia abajo desde cada objeto.	
<b>Mensajes</b>	Se muestran como flechas horizontales entre las líneas de vida de los objetos.	
<b>Activación</b>	Rectángulos delgados sobre una línea de vida que representan el período durante el cual un objeto está ejecutando una operación. Cuanto más larga es la barra de activación, mayor es el tiempo que el objeto pasa realizando esa operación.	

### 2.3.4.3. Diagrama de Actividades

Tipo	Descripción	Notación
<b>Nodos de Actividad</b>	Actividades: Representadas por rectángulos con esquinas redondeadas.	
	Transiciones: Flechas que conectan elementos en el diagrama.	
<b>Nodos de Control</b>	Inicio: Un círculo sólido, marca el comienzo del flujo de actividad.	
	Fin: Un círculo con un borde más grueso y un círculo más pequeño dentro, indica el fin del flujo de actividad.	
	Decisión: Un rombo, representa un punto de decisión basado en una condición.	
<b>Estructuras Avanzadas</b>	Nodo de bifurcación (Fork): Una barra de sincronización que divide un flujo en múltiples flujos concurrentes.	
	Nodo de unión (Join): Similar a la bifurcación pero inversa; une varios flujos concurrentes en uno solo.	

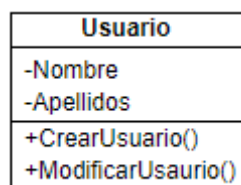
<b>Objetos</b>	Objetos: Representados por rectángulos, muestran los datos o entidades utilizadas o modificadas por las actividades.	
<b>Particiones (Swimlanes)</b>	Líneas verticales u horizontales que dividen el diagrama en secciones, representando diferentes responsabilidades o roles en el proceso modelado.	

## 2.4. Ejemplos de notaciones

### 2.4.1. Diagrama de clases

#### ❖ Clases

La parte superior contiene el nombre de la clase, la sección media enumera los atributos o propiedades, y la sección inferior muestra los métodos o funciones de la clase. Las clases abstractas pueden indicarse con el nombre en cursiva.



*Ilustración 1 Clases*

*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

#### ❖ Relaciones

Asociación: Representada por una línea simple que conecta dos clases, indica que existe una relación entre ellas, con opcionalmente flechas en uno o ambos extremos para mostrar la dirección de la relación y etiquetas para indicar los roles y la multiplicidad (por ejemplo, 1.. para indicar "uno a muchos").



*Ilustración 2 Relaciones: Asociación*

**Agregación:** Simbolizada por una línea que termina en un diamante hueco en el lado de la clase contenedora, representa una relación "todo-parte" donde las partes pueden existir independientemente del todo.



Ilustración 3 Relaciones: Agregación

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

**Composición:** Representada por una línea que termina en un diamante relleno en el lado de la clase contenedora, indica una relación "todo-parte" más fuerte que la agregación, donde las partes no pueden existir sin el todo.

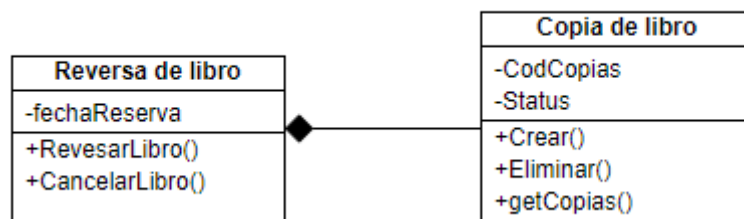


Ilustración 4 Relaciones: Composición

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

**Herencia (Generalización):** Una línea que termina en un triángulo hueco apuntando hacia la clase base, muestra que una clase hereda de otra. Se utiliza para representar la relación "es un" entre clases.

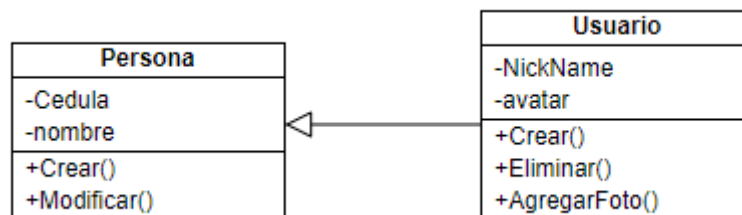


Ilustración 5 Relaciones: Herencia

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

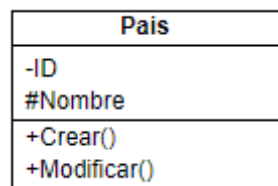


- ❖ **Dependencia:** Una línea punteada con una flecha que apunta hacia la clase de la que se depende, indica que una clase utiliza otra de manera ocasional o para un propósito específico dentro de un método.



*Ilustración 6 Relaciones: Dependencia*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

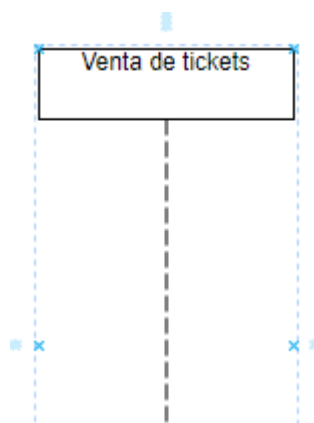
- ❖ **Atributos:** Se listan en la sección media del rectángulo de la clase, siguiendo el formato "nombre", y pueden incluir su visibilidad (pública [+], privada [-], protegida [#]).



*Ilustración 7 Atributos*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## 2.4.2. Diagrama de secuencia

- ❖ **Objetos:** Rectángulos situados en la parte superior del diagrama: Representan los participantes (objetos) en la interacción. El nombre del objeto y su clase pueden aparecer dentro del rectángulo, separados por dos puntos (por ejemplo, objeto:Clase).



*Ilustración 8 Objetos*

- ❖ **Líneas de Vida:** Líneas verticales discontinuas que se extienden hacia abajo desde cada objeto: Representan la existencia del objeto durante el tiempo que dura la interacción. La longitud de la línea no necesariamente refleja la duración temporal real.

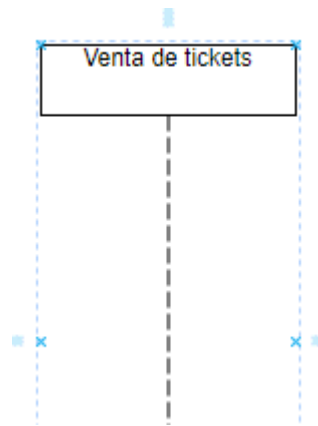


Ilustración 9 Líneas de vida

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

- ❖ **Mensajes:** Flechas horizontales entre las líneas de vida: Indican la comunicación entre objetos. La dirección de la flecha muestra el flujo del mensaje del emisor al receptor.

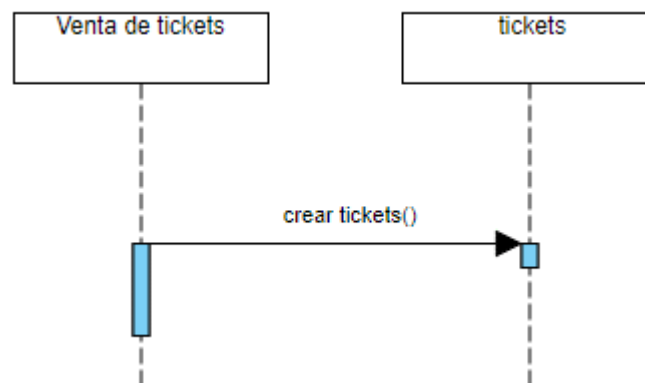
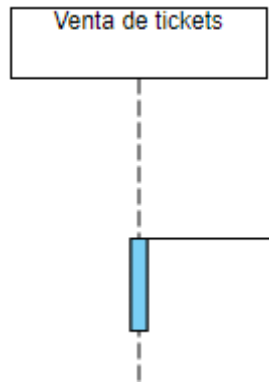


Ilustración 10 Mensaje

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

**Barras de Activación:** Rectángulos delgados sobre las líneas de vida: Indican el período durante el cual un objeto está activo o ejecutando un proceso.

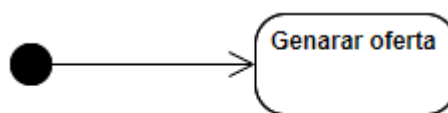


*Ilustración 11 Barra de activación*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### 2.4.3. Diagrama de secuencia

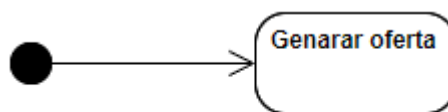
#### 2.4.3.1. Nodos de Actividad

- ❖ **Actividades:** Representadas por rectángulos con esquinas redondeadas, indican tareas o pasos en el proceso.



*Ilustración 12 Nodos de actividad: Actividad*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

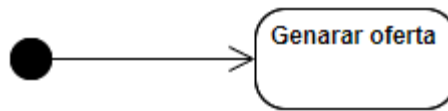
- ❖ **Transiciones:** Flechas que conectan los elementos, mostrando el flujo de una actividad a la siguiente.



*Ilustración 13 Nodos de actividad: Transiciones*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

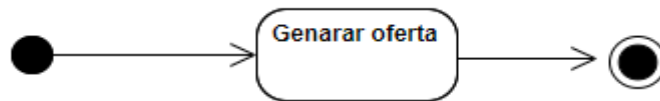
#### 2.4.3.2. Nodos de Control

- ❖ **Nodo Inicial:** Un círculo sólido, indica el punto de inicio del flujo de actividad.



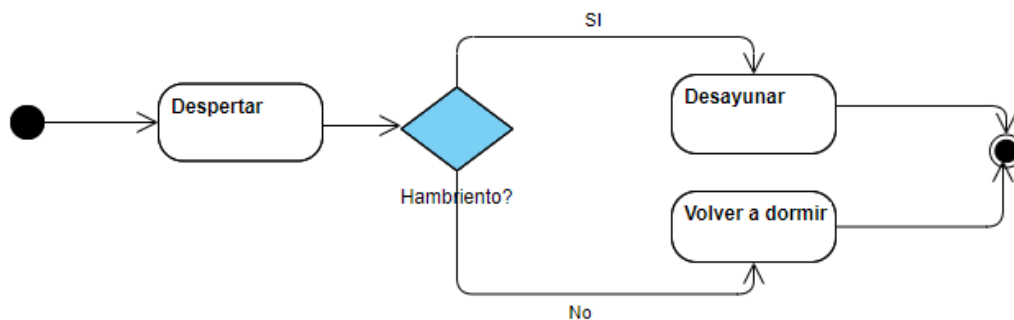
*Ilustración 14 Nodos de control: Nodo inicial*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

- ❖ **Nodo Final:** Un círculo con un borde más grueso y un círculo más pequeño en el centro, representa el fin de todo el flujo de actividad.



*Ilustración 15 Nodos de control: nodo final*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

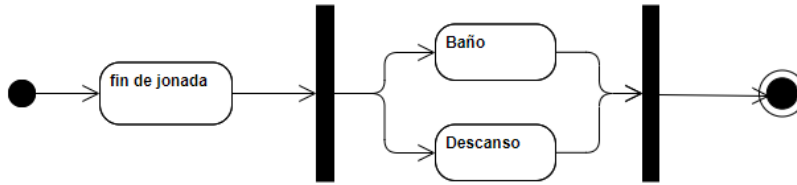
- ❖ **Nodo de Decisión:** Un rombo, se utiliza para representar una bifurcación en el flujo donde se toma una decisión basada en una condición.



*Ilustración 16 Nodos de control: Nodo de decisión*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

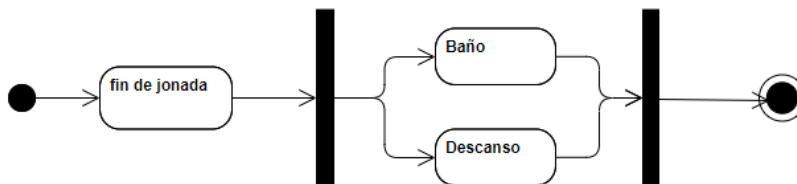
#### 2.4.3.3. Sincronización

- ❖ **Nodo de Bifurcación (Fork):** Una barra gruesa horizontal o vertical, representa el punto donde el flujo se divide en múltiples flujos paralelos.



*Ilustración 17 Sincronización: Nodo de bifurcación*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

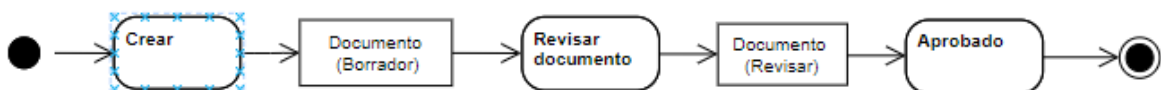
- ❖ **Nodo de Unión (Join):** Similar al nodo de bifurcación, pero se utiliza para sincronizar flujos paralelos en un solo flujo.



*Ilustración 18 Sincronización: Nodo de unión*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

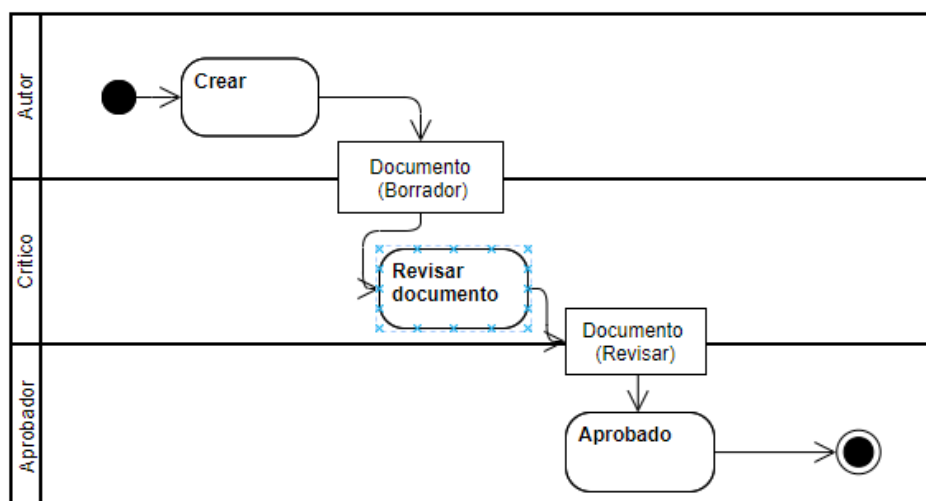
#### 2.4.3.4. Objetos

**Objetos:** Representados por rectángulos, indican los datos o entidades utilizadas o producidas por las actividades.



*Ilustración 19 Objetos*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

**Particiones:** Líneas que dividen el diagrama en columnas o filas, representan diferentes responsabilidades o roles en el proceso modelado.



*Ilustración 20 Particiones*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### 3. Diagramas BPMN

#### 3.1. Fundamentos de Diagramas BPMN

##### 3.1.1. Conceptos básicos

BPMN, conocido como Business Process Model and Notation, es un estándar global para la modelación de procesos de negocio. Proporciona un conjunto gráfico de símbolos y notaciones diseñadas para ser comprendidas por todos los participantes en el negocio [16]. Esto incluye desde los analistas de negocio, que crean los borradores de los procesos, hasta los técnicos que implementan las soluciones tecnológicas, y los gerentes encargados de supervisar los procesos. Su objetivo principal es facilitar la creación de una documentación de procesos de negocio coherente y fácil de entender, sirviendo como puente entre el diseño del proceso y su implementación práctica [17].

##### 3.1.2. Elementos principales de BPMN

Dentro de BPMN, los elementos se clasifican en cuatro categorías principales que son esenciales para la construcción de cualquier diagrama de proceso de negocio [16].

Los Objetos de Flujo incluyen tareas, eventos y puertas de enlace. Las tareas representan acciones individuales dentro de un proceso. Los eventos señalan algo que ocurre, como el inicio o fin de un proceso, y pueden ser clasificados según su posición (inicio, intermedio, fin). Las puertas de enlace dirigen la divergencia y convergencia del flujo, basándose en condiciones lógicas, permitiendo múltiples caminos dentro del proceso [17], [18].

Los Objetos de Conexión son los elementos que unen los objetos de flujo. La secuencia de flujos muestra el orden de las actividades, mientras que los mensajes y las asociaciones representan la interacción entre diferentes partes del proceso o con elementos externos. Estos objetos son cruciales para definir la secuencia y la comunicación dentro del proceso [17].

Las Piscinas y Carriles son estructuras que organizan las tareas y eventos. Una piscina representa a un participante completo en el proceso, y los carriles dividen esta piscina en roles o responsabilidades específicas. Facilitan la visualización de la interacción entre diferentes entidades o departamentos involucrados en el proceso [18].

Por último, los Artefactos ofrecen información adicional que puede ser necesaria para entender el proceso. Esto incluye datos, que muestran cómo se manejan y se transforman a lo largo del proceso, grupos, que permiten organizar elementos del proceso, y anotaciones, que proporcionan comentarios o explicaciones adicionales [19].

## **3.2. Modelamiento de sistemas distribuidos**

### **3.2.1. Ventajas del modelamiento con BPMN en sistemas distribuidos**

El uso de BPMN en sistemas distribuidos destaca por su capacidad para aportar claridad visual a los procesos complejos. Esta notación permite una comunicación efectiva entre todos los participantes del proyecto, desde analistas de negocio hasta desarrolladores y gestores. La visualización intuitiva de las interacciones entre componentes distribuidos facilita la comprensión común de los procesos y mejora la colaboración [20].

Como un estándar de la industria reconocido, BPMN promueve la interoperabilidad entre diferentes herramientas y plataformas. Esto evita el bloqueo por un proveedor específico y permite a las organizaciones adoptar las mejores prácticas de modelado de procesos. La flexibilidad y adaptabilidad son, por tanto, grandes ventajas de utilizar BPMN en el contexto de sistemas distribuidos [18].

Además, BPMN juega un papel crucial en la automatización de procesos. Permite la implementación de flujos de trabajo automatizados que operan eficientemente a través de distintos sistemas y servicios. Esta característica es especialmente valiosa para gestionar cambios, ya que los diagramas BPMN detallados pueden ayudar a identificar rápidamente áreas para adaptación o mejora [21].

### **3.2.2. Consideraciones para el modelamiento de sistemas distribuidos**

El modelado de sistemas distribuidos con BPMN requiere una planificación cuidadosa debido a la complejidad inherente de estos sistemas. Es vital asegurar una comunicación efectiva entre los componentes y gestionar adecuadamente la sincronización y los errores. Estos desafíos destacan la importancia de un diseño detallado y considerado [20].

El rendimiento y la escalabilidad son aspectos críticos en el modelado de sistemas distribuidos. Los modelos BPMN deben diseñarse teniendo en cuenta las cargas de trabajo esperadas y la capacidad de los componentes del sistema. Esto asegura que los sistemas no solo cumplen con los requisitos actuales, sino que también pueden adaptarse a futuras demandas [21].

La seguridad y la confiabilidad son fundamentales en cualquier sistema distribuido. Los modelos BPMN deben incluir consideraciones de seguridad para proteger los datos y asegurar la continuidad del servicio. Además, la interoperabilidad entre diversas tecnologías y plataformas es esencial para lograr una integración fluida en sistemas distribuidos complejos [21], [22].

### **3.2.3. Ejemplos de modelamiento de sistemas distribuidos con BPMN**

Los sistemas de comercio electrónico representan un excelente ejemplo de cómo BPMN puede facilitar la coordinación entre diversos servicios, como gestión de inventario, procesamiento de pagos y logística. El modelado con BPMN ayuda a optimizar estos procesos, mejorando la eficiencia operativa y la satisfacción del cliente [23].

En el sector bancario, el modelado BPMN es invaluable para visualizar y mejorar las transacciones, la seguridad y el procesamiento de datos entre sistemas bancarios. Esto no solo mejora la eficiencia, sino que también enriquece la experiencia del usuario, garantizando transacciones seguras y confiables [24].

Los sistemas de salud electrónica también se benefician enormemente del modelado con BPMN. Al visualizar los flujos de trabajo para la gestión de registros médicos, citas y tratamientos, BPMN facilita una atención coordinada entre hospitales, clínicas y otros proveedores de servicios de salud. Esto contribuye a una atención al paciente más eficaz y eficiente [22].

Estos ejemplos ilustran la versatilidad y el valor de BPMN en el modelado de sistemas distribuidos, demostrando cómo puede mejorar la comprensión, la automatización y la adaptabilidad en diversos sectores[24].



### **3.3. Herramientas y tecnologías para el modelamiento con BPMN en sistemas distribuidos**

#### **3.3.1. Herramientas de software para el modelamiento con BPMN**

El modelado BPMN se apoya en una variedad de herramientas de software, que van desde aplicaciones sencillas hasta soluciones empresariales avanzadas. Estas herramientas ofrecen funcionalidades que abarcan el diseño de diagramas, la simulación de procesos, y en algunos casos, la automatización y ejecución de estos procesos. Entre las herramientas más destacadas se encuentran Bizagi, Camunda, Signavio, y Bpm'online, cada una con sus propias características y ventajas [25].

Estas plataformas proporcionan interfaces gráficas que facilitan la creación de diagramas BPMN de manera intuitiva. Además, muchas de estas herramientas incluyen capacidades para la colaboración en equipo, lo que permite a los usuarios trabajar juntos en el diseño y mejora de los procesos de negocio. La gestión de versiones y la integración con sistemas de ejecución de procesos de negocio son otras características importantes que pueden influir en la elección de una herramienta específica [26].

#### **3.3.2. Integración de BPMN con tecnologías de sistemas distribuidos**

La integración de modelos BPMN con sistemas distribuidos es fundamental para la ejecución eficiente de los procesos de negocio. Esta integración se facilita mediante el uso de tecnologías como middleware de integración, servicios web, API REST, y protocolos de mensajería. Estas tecnologías permiten la comunicación entre distintos componentes y servicios, asegurando que los procesos modelados funcionen sin problemas a través de diversas plataformas y sistemas [25], [26].

Las herramientas de BPM y BPMN suelen ofrecer capacidades de integración o extensiones que permiten conectar los modelos de procesos con aplicaciones externas. Esto incluye sistemas como ERP, CRM y otras bases de datos y aplicaciones empresariales. La integración efectiva es crucial para automatizar los procesos de negocio modelados, permitiendo una operación fluida y coherente entre distintos sistemas [27].

#### **3.3.3. Consideraciones de implementación y despliegue de modelos BPMN en sistemas distribuidos**

Implementar y desplegar modelos BPMN en entornos distribuidos requiere una atención especial a varios factores clave. La escalabilidad es uno de estos factores, ya que los sistemas deben ser capaces de manejar incrementos en la demanda sin comprometer el rendimiento. La



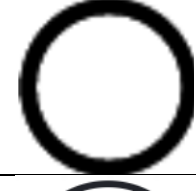


gestión de la carga y la seguridad son igualmente importantes, considerando los desafíos únicos que presentan los entornos distribuidos, como la dificultad para aislar y resolver fallos [28].





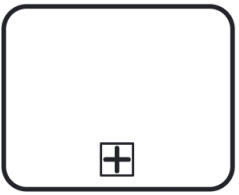
La interoperabilidad con tecnologías existentes también juega un papel crucial en la implementación exitosa de modelos BPMN. Asegurar que los nuevos modelos se integren sin problemas en el ecosistema tecnológico existente evita complicaciones y facilita una transición suave. Además, el monitoreo en tiempo real y la gestión de procesos son esenciales para mantener un control sobre el rendimiento del sistema, permitiendo ajustes proactivos y mejoras continuas [27].







### 3.3.4. Notación y símbolos utilizados en BPMN


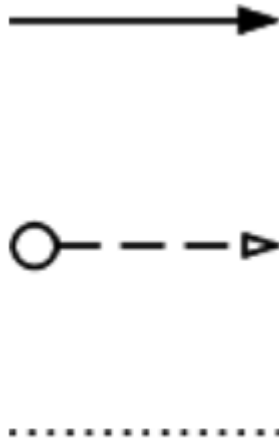


La notación BPMN se caracteriza por su conjunto estandarizado de símbolos gráficos, permitiendo la representación precisa y detallada de cada aspecto del proceso de negocio.



*Tabla 3 Notaciones diagrama BPMN*  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

Tipo	Descripción	Notación
<b>Eventos</b>	Indica el inicio de un proceso. Esta se representa mediante un círculo abierto con una línea final.	
	Este ocurre entre el evento de inicio y evento final.	
	Indica el final de un proceso	
	Evento de mensaje de recepción, se quiere o desea obtener el mensaje enviado por otro proceso.	
	Evento de envío de mensaje. Se activa cuando recibe un mensaje	

	Evento de inicio de mensaje.	
	Evento de temporizador, se utiliza con un símbolo de reloj. Se activa cuando la condición de tiempo especificada ocurre al inicio o intermedio de un proceso.	
		
<b>Actividades o tareas</b>	Representan el trabajo o proceso que se realiza dentro de una organización.	
	Subproceso, representan una tarea secundaria a la principalmente, es decir, las dos estas asociadas.	

	<p><b>Tareas de usuario:</b> Indica que la tarea debe ser realizada por un usuario con la ayuda de un sistema. Por ejemplo: registro de personal.</p> <p><b>Tarea de servicio:</b> Es realizada por un sistema de forma automática. Por ejemplo, envió de un correo de acuerdo a un tiempo establecido.</p> <p><b>Tarea de script:</b> Son actividades que ejecutan una expresión automatizada, por ejemplo: envío de correo electrónico personalizado.</p> <p><b>Tarea de regla de negocio:</b> Son definidas por tareas asociadas directamente a la organización. Por ejemplo, analizar los resultados de las encuestas.</p>	   
<p><b>Puertas de enlace</b></p>	<p><b>Símbolo exclusivo:</b> Evalúa según el estado del proceso y con ello toma una decisión. Dentro de lo que son compuerta exclusiva existen dos tipos las divergentes y convergentes.</p> <p><b>Divergente:</b> Es utilizada para crear caminos alternativos.</p> <p><b>Convergente:</b> Se utiliza para unir caminos alternativos.</p>	
	<p><b>Símbolo paralelo:</b> Es diferente porque a diferencia de otras compuertas esta no tiene dependencia de condiciones o eventos. En esta las tareas se activan u ocurren en el mismo instante,</p>	

	<p>pero tienen diferente orden de finalización ocasionando que la tarea final se ejecute cada vez que una tarea paralela termina. para ello se debe aplicar también los conceptos de divergencia y convergencia.</p>	
	<p><b>Símbolo basado en eventos:</b> Representa una acción o suceso del sistema en cualquier momento. Es similar a la compuerta exclusiva ya que ambas representan caminos o rutas. Pero esta pueda tener más de dos caminos, además si uno de los caminos o eventos es activado los demás se desactivan.</p>	
<p><b>Conexión</b></p>	<p><b>Flujo de secuencia:</b> Muestra el flujo normal de los procesos o actividades.</p> <p><b>Flujo de mensaje:</b> Se utiliza para enviar mensajes o información entre entidades o pool.</p> <p><b>Símbolo de asociación:</b> Se utiliza para asociar artefactos a una tarea o proceso.</p>	
<p><b>Carriles</b></p>	<p>Se utilizan para la organización de los procesos. Por ejemplo, dividir procesos de compras y ventas.</p>	
<p><b>Artefactos</b></p>	<p><b>Data store:</b> Representa guardado de datos de un proceso o extracción de información.</p>	

	<b>Data object:</b> Permite mostrar el flujo de información que está ocurriendo durante los procesos. Representa información como: correos, documentos o cartas.	
<b>Grupos</b>	Es utilizado para la agrupación de tareas o procesos que están desarrollándose sobre la misma categoría. Este grupo no afecta a los flujos de secuencias de los procesos.	

### 3.4. Ejemplos de notaciones

#### 3.4.1. Eventos

##### Evento de inicio de mensaje

En este ejemplo para que se inicie la actividad o proceso “A” primero tiene que haber llegado el documento.

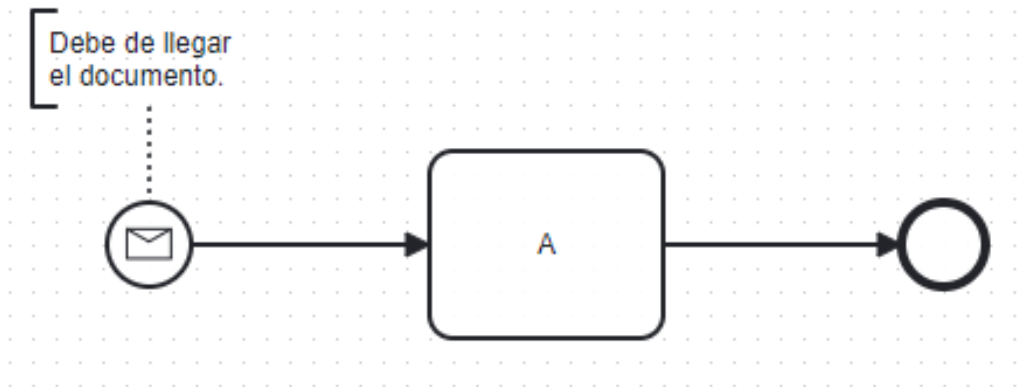
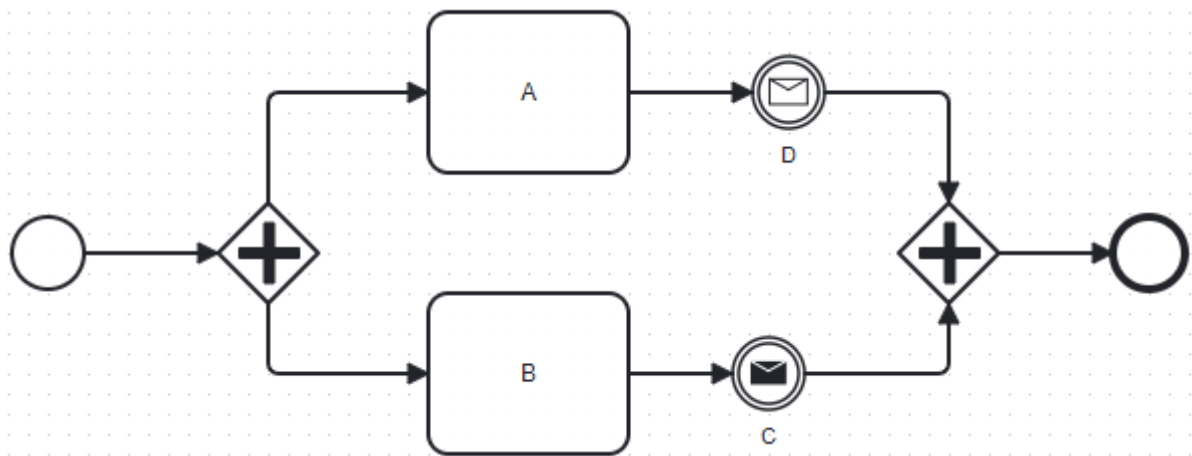


Ilustración 21 Evento de inicio de mensaje  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

##### Evento de envío y recepción de mensaje

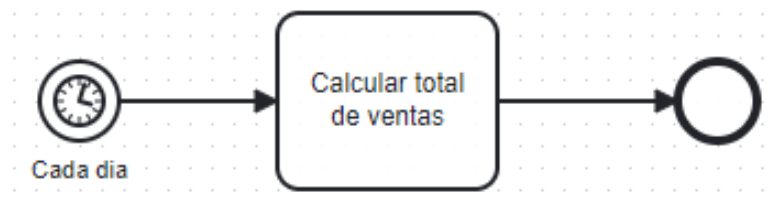
En este caso se utilizó actividades en paralelo con esto cuando el flujo llegué al evento “D” se quedará esperando a que el evento “C” envíe la información que D necesita para avanzar.



*Ilustración 22 Evento de envío y recepción de mensaje  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### **Evento de inicio de temporizador**

En este caso para que inicie la actividad o el proceso debe haber pasado el ciclo de tiempo que se especifica en el temporizador.



*Ilustración 23 Evento de inicio de temporizador  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### **Evento intermedio de temporizador**

Una vez el flujo llega al temporizador debe pasar el tiempo especificado para que pasa a la actividad o proceso siguiente "B".



*Ilustración 24 Evento intermedio de temporizador  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### 3.4.2. Tareas

#### Tarea generica

En este caso la tarea es genérica toma como definición según la especificación del proceso.

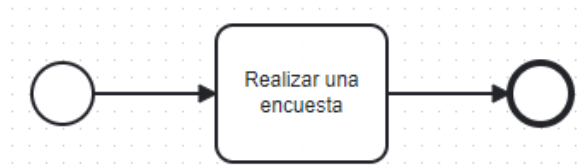


Ilustración 25 Tarea genérica

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

#### Tarea de usuario

En este caso la tarea es realizada por un actor del sistema que se está llevando a cabo.

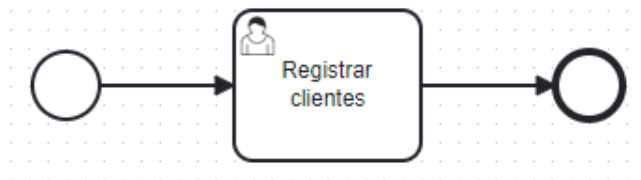


Ilustración 26 Tarea de usuario

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

#### Tarea de servicio

En este caso la tarea se realiza de forma automática dentro del flujo del sistema.



Ilustración 27 Tarea de servicio

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

#### Tarea de script

La tarea esta definida como un proceso automático para el sistema. Como se puede observar la tarea ya no la tiene que realizar un usuario ya se automatizo esta acción por ende el sistema con su plantilla de correo ya puede enviar los correos de forma más rápida a los usuarios.

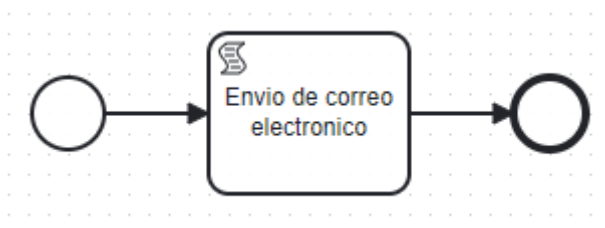


Ilustración 28 Tarea de script

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA



### Tarea de regla de negocio

Tarea que condiciona una acción o regla de la organización. En este caso debe cumplirse el descuento del 15% para que la tarea pueda validarse.



Ilustración 29 Tarea regla de negocio

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

### 3.4.3. Puertas de enlace

#### Exclusivo

##### Exclusivo Divergente

En este caso la compuerta divergente actúa como la condicional if en lenguajes de programación con dos salidas verdadera o falsa. En este caso se cumplirá si la edad del usuario es mayor o igual a 18 años caso contrario no se cumplirá la condición.

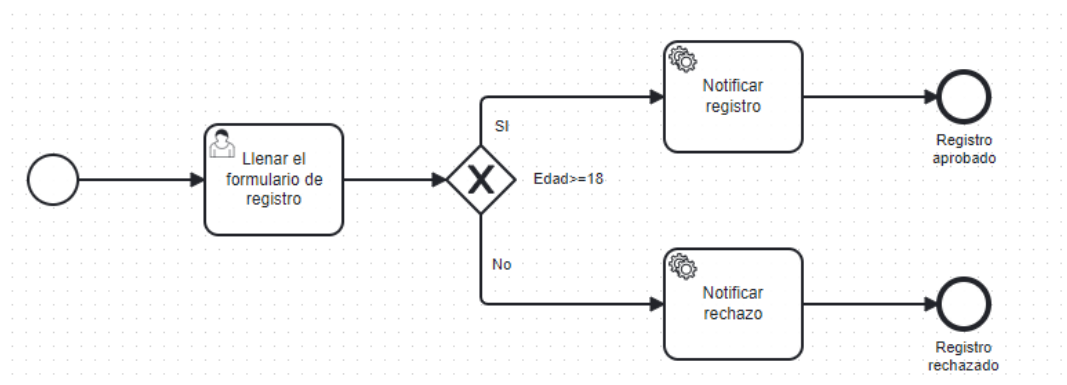


Ilustración 30 Exclusivo divergente

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

##### Exclusivo Convergente

En este caso la compuerta actúa como la unión de los caminos para la tarea "Corrección". Es decir no solo se utiliza para condicionales.

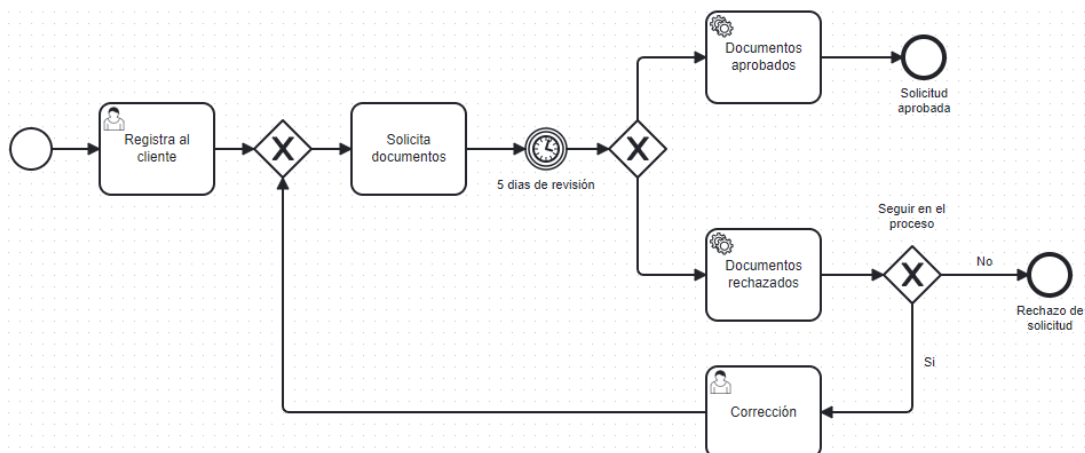


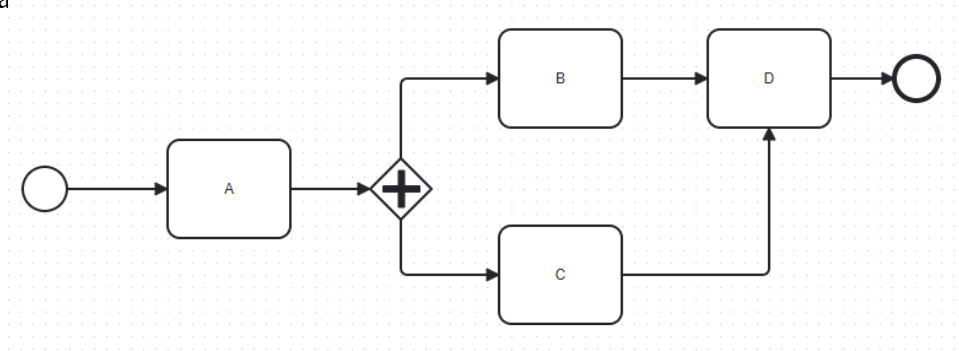
Ilustración 31 Exclusivo convergente

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## Paralelo

### Divergente

Error ya que la actividad C se ejecutará cada vez que termine la tarea, las tareas B y C terminaran en tiempo diferentes lo cual harán que la tarea D se ejecuten cada vez que finalicen las tareas anteriores

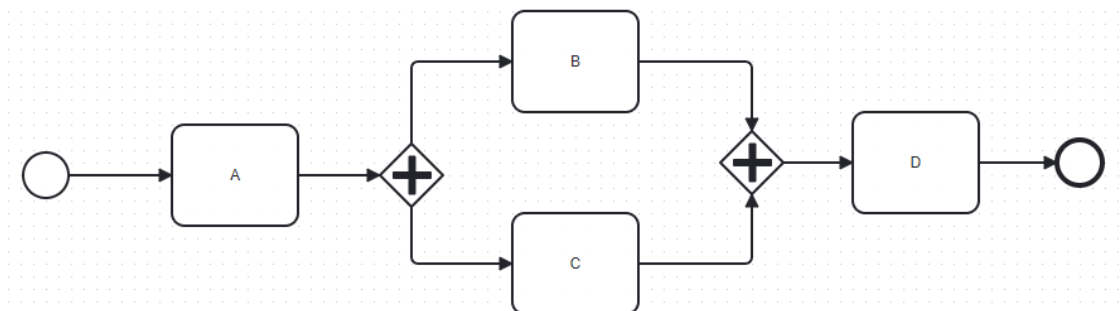


*Ilustración 32 Paralelo divergente*

*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### Convergente

La acción convergente hará que la tarea D solo se ejecute una vez.

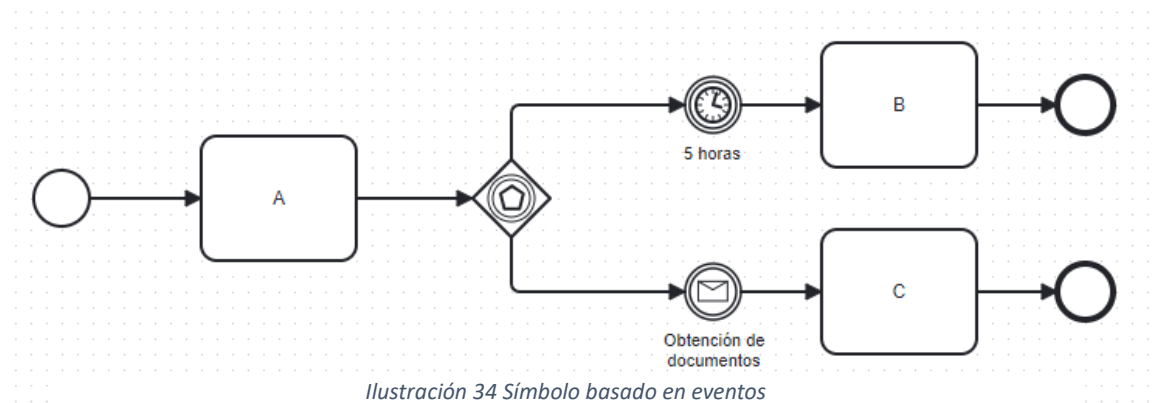


*Ilustración 33 Paralelo convergente*

*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

### 3.4.4. Símbolo basado en eventos

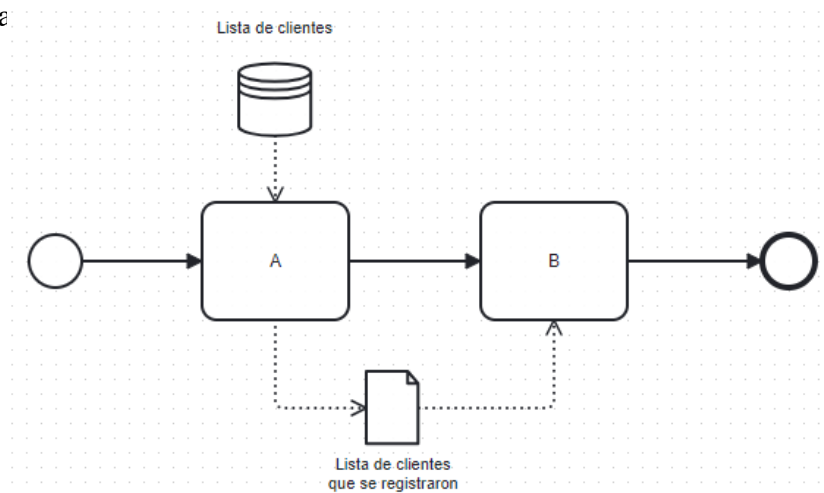
En este ejemplo se puede observar que ser tiene el evento de tiempo y de mensaje en este caso cuando se ejecute el de tiempo la acción o evento de mensaje se desactivara por lo cual hará que solo un evento se ejecute.



### 3.4.5. Artefactos

#### Data store y Data object

En este caso la actividad o proceso “A” debe consumir de un servicio la lista del cliente para luego que envíe:



#### Pool

En esta ocasión los procesos fueron divididos en dos Pool los cuales el primero se lo asocio a ventas y el segundo a productos en las cuales la Actividad “A” solicitara los clientes de la base de datos en la cual esta mismo será procesada y solo recolectara en un objeto los clientes que realizaron compras. En la cual la actividad B con esos clientes solicitara al servicio de los productos una lista de sus insumos para con ello saber los productos que cada cliente compro.

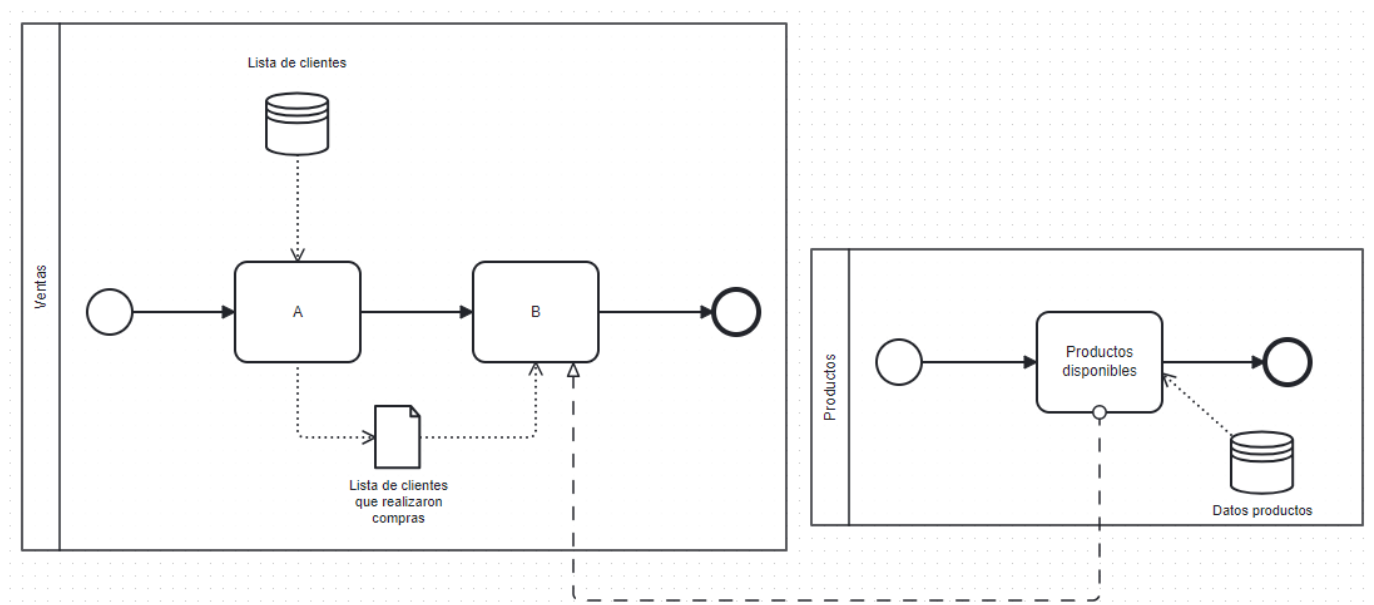


Ilustración 36 Pool  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## Grupos

En este caso la actividad A y B están desarrollándose dentro de una misma categoría por ende se las agrupa.

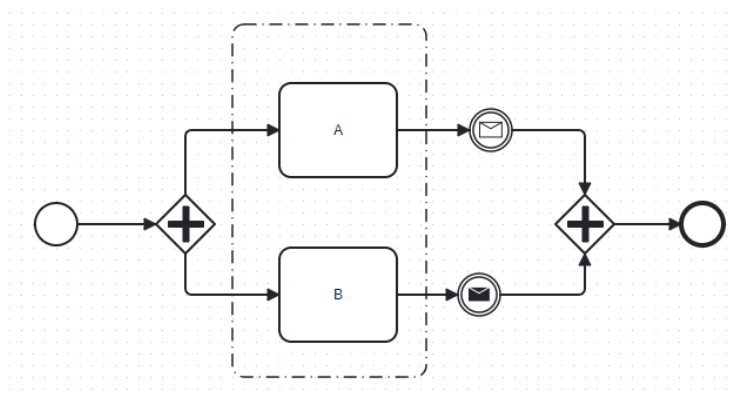


Ilustración 37 Grupos  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## 3.5. Ejercicios

### 3.5.1. Problema: Sistema de Control de Tráfico Urbano Inteligente

**Descripción:** Imaginen una ciudad con un tráfico intenso y complejo, donde se busca mejorar la eficiencia y la seguridad mediante un Sistema de Control de Tráfico Urbano Inteligente (SCTUI). Este sistema debe abordar la coordinación de semáforos, el monitoreo del flujo vehicular y la gestión de emergencias.

**Requerimientos:**

### **1. Sincronización de Semáforos:**

- ❖ Los semáforos deben ser sincronizados dinámicamente para optimizar el flujo de tráfico en tiempo real.
- ❖ Los cambios en las condiciones del tráfico, como congestiones o eventos especiales, deben reflejarse en la sincronización de los semáforos.

### **2. Detección de Flujo Vehicular:**

- ❖ Se requiere un sistema de detección de vehículos mediante cámaras y sensores para monitorear el flujo vehicular en intersecciones.
- ❖ El sistema debe ser capaz de analizar patrones de tráfico y ajustar la sincronización de semáforos en consecuencia.

### **3. Gestión de Emergencias:**

- ❖ Se deben implementar protocolos para priorizar vehículos de emergencia (ambulancias, bomberos, policía) y garantizar su paso seguro a través de las intersecciones.
- ❖ El sistema debe tener la capacidad de identificar y responder rápidamente a situaciones de emergencia.

### **4. Integración con Sistemas Externos:**

- ❖ El SCTUI debe integrarse con sistemas externos, como bases de datos de eventos especiales, servicios meteorológicos y mapas de tráfico en tiempo real.
- ❖ La información de estos sistemas externos debe utilizarse para tomar decisiones informadas sobre la sincronización de semáforos.

### **5. Visualización en Tiempo Real:**

- ❖ Se requiere una interfaz de usuario para visualizar el estado del tráfico y la sincronización de semáforos en tiempo real.
- ❖ La interfaz debe ser accesible para operadores y autoridades de tráfico para tomar decisiones y realizar ajustes según sea necesario.

### 3.5.2. Modelo

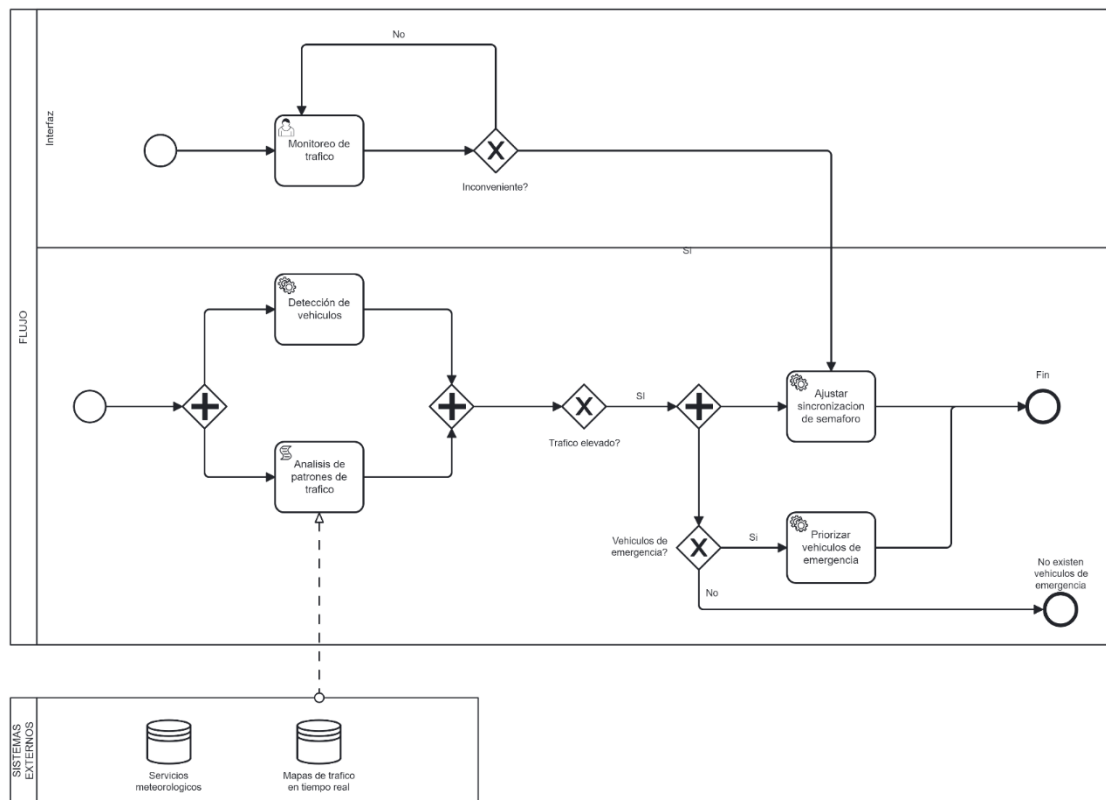
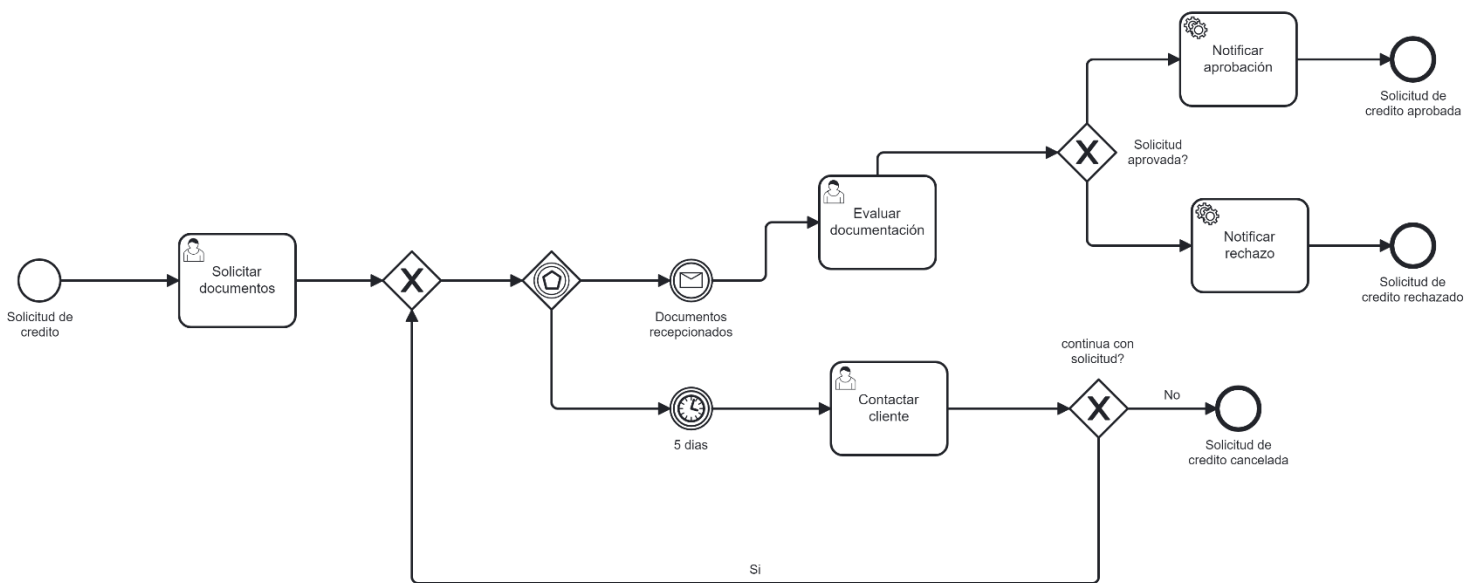


Ilustración 38 Diagrama BPMN de Sistema de Control de Tráfico Urbano Inteligente  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

### 3.6. Problema: Evaluación de solicitud de créditos

Cuando el banco recibe la solicitud de crédito de un cliente procede a solicitar algunos documentos. Si el cliente no trae los documentos dentro de los siguientes 5 días es necesario contactarlo. Si el cliente no continua con la solicitud no se deben esperar los documentos y el proceso debe terminar caso contrario el banco esperará los documentos del cliente.

Por otro lado, si el cliente envía los documentos, no es necesario contactarlo y se procede a evaluar la documentación, si la solicitud de crédito no es aprobada se notifica rechazo caso contrario se notifica aprobación solicitud de créditos.



*Ilustración 39 Evaluación de solicitud de créditos*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

### 3.7. Problema: gestión de reservas de un hotel

1. Los clientes pueden efectuar reservar anticipadas. El hotel admite tantas reservas como habitaciones libres tenga. Las reservas telefónicas tienen que estar respaldadas por un número de tarjeta de crédito. Si en la fecha de reserva no se presenta el cliente, se genera una factura que se envía a la compañía de tarjetas de crédito.
2. Hay dos tipos de clientes: los individuales y los que pertenecen a empresas. Para los clientes de empresa no es necesario garantizar las reservas mediante una tarjeta de crédito.
3. Cuando un cliente llega al hotel su reserva es procesada, comprobándose la misma con los detalles que proporciona el cliente.
4. Hay clientes que solicitan una habitación en el mostrador del hotel.
5. Algunos clientes solicitan habitaciones para no fumadores.
6. Las habitaciones se pueden alquilar para dormir únicamente, con media pensión o con pensión completa.
7. Cuando los clientes abandonan el hotel, un empleado comprueba los detalles de ocupación (llamadas telefónicas, servicio de bar, etc) y genera una factura para el cliente.
8. Hay clientes, que pertenecen a empresas, que no abonan la factura en ese momento. A final de mes se envía una factura única a la empresa.
9. El sistema tendrá tres tipos de usuarios: los empleados de mostrador o recepción, el gerente y un administrador. El gerente se encargará de gestionar las cuentas de empresas: tipo de descuento por habitación, apertura de cuenta y cierre de cuenta. El administrador se

encargará de efectuar un mantenimiento sobre la información que se almacena en el sistema. Por último, los empleados de mostrador se encargan de la gestión de clientes.



### 3.7.1. Modelo

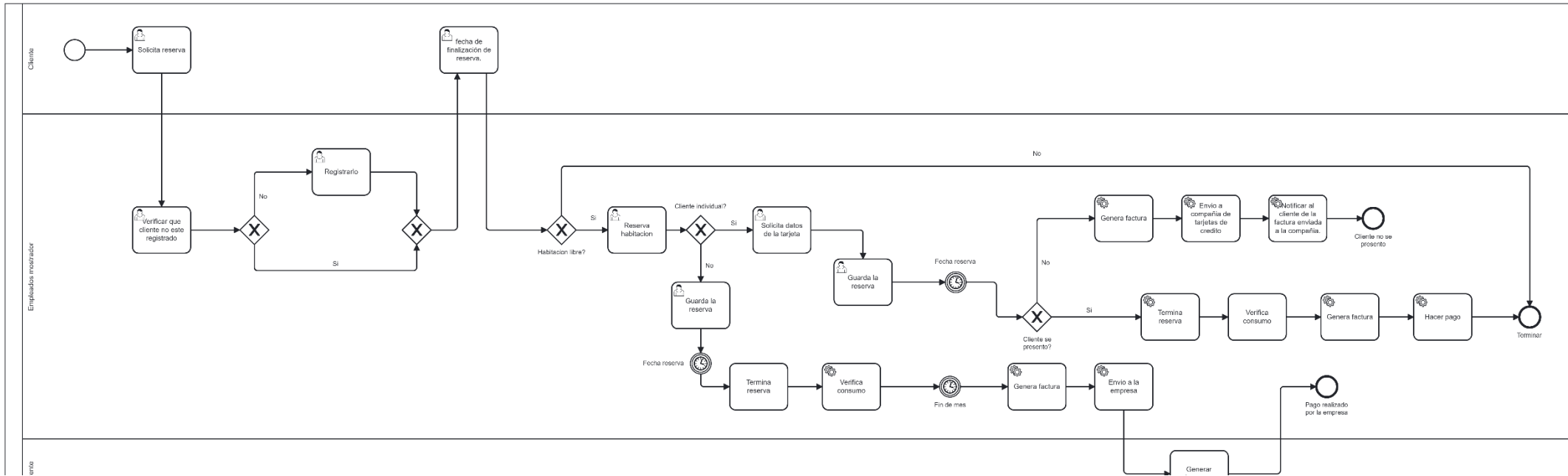


Ilustración 40 Gestión de reservas de un hotel  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## **4. Diagrama Petri**

### **4.1. Fundamentos de Diagramas Petri**

#### **4.1.1. Conceptos básicos**

Un diagrama de Petri es un modelo matemático y gráfico utilizado para describir y analizar sistemas concurrentes y distribuidos. Se caracteriza por su capacidad para modelar la sincronización, la paralelización y la dependencia mutua entre eventos o procesos. Los elementos fundamentales de un diagrama de Petri son los lugares (representados por círculos), las transiciones (representadas por barras o rectángulos) y los tokens (indicados por puntos dentro de los lugares) [29].

Los lugares pueden interpretarse como condiciones o estados, las transiciones como eventos que pueden cambiar esos estados, y los tokens como indicadores del estado actual del sistema [30]. Un token se mueve de un lugar a otro a través de las transiciones, modelando así el flujo de control o la ejecución de operaciones en el sistema. La activación de una transición ocurre cuando todos los lugares de entrada tienen el número necesario de tokens, lo que permite la simulación de la dinámica y la concurrencia dentro de los sistemas modelados [29], [30].

Este modelo es ampliamente utilizado en diversas áreas como la informática, la ingeniería, la robótica y la gestión de procesos de negocio, proporcionando una herramienta poderosa para el análisis y diseño de sistemas complejos que involucren operaciones concurrentes y recursos compartidos [29].

#### **4.1.2. Elementos principales de Petri**

Los diagramas de Petri se componen de tres elementos clave que juntos facilitan la modelación de procesos dinámicos y concurrentes en sistemas. El primer elemento, los lugares (Places), representados por círculos, simbolizan estados o condiciones dentro del sistema. Los lugares pueden albergar tokens, que indican la presencia de una condición o la disponibilidad de recursos, reflejando el estado actual del sistema a medida que los tokens se mueven o cambian [30].

El segundo elemento esencial son las transiciones (Transitions), ilustradas mediante barras o rectángulos. Estas representan eventos o acciones que pueden cambiar el estado del sistema. Para que una transición se active, es necesario que todos sus lugares de entrada contengan el número adecuado de tokens requeridos, lo que permite la simulación de eventos que ocurren bajo ciertas condiciones. Al dispararse, las transiciones consumen tokens de sus lugares de

entrada y generan tokens en los lugares de salida, modificando así la distribución de tokens en el sistema y, por ende, su estado [31].

Finalmente, los arcos (Arcs), representados por flechas, conectan lugares con transiciones y viceversa, definiendo el flujo de tokens entre estos elementos. Los arcos direccionan el movimiento de los tokens, con arcos de entrada que apuntan hacia las transiciones desde los lugares, y arcos de salida que van desde las transiciones hacia los lugares [32]. Los pesos en los arcos determinan la cantidad de tokens necesarios para activar una transición o los que se producen tras la activación, permitiendo modelar con precisión la lógica del sistema modelado. Estos componentes trabajan en conjunto para proporcionar una poderosa herramienta de modelado para sistemas que involucran complejas interacciones y procesos concurrentes [31].

## **4.2. Modelamiento de sistemas distribuidos**

### **4.2.1. Ventajas del modelamiento con Petri en sistemas distribuidos**

El modelado con redes de Petri en sistemas distribuidos brinda una representación clara y detallada de los procesos concurrentes que son fundamentales en estos sistemas [33]. A través de su estructura única, que facilita la visualización de múltiples actividades ocurriendo simultáneamente, las redes de Petri destacan por su capacidad para modelar la concurrencia de una manera que otros métodos encuentran desafiante. Esta claridad en la representación permite a los diseñadores y analistas de sistemas comprender mejor las interacciones complejas y la simultaneidad inherente a los sistemas distribuidos, mejorando así el diseño y la implementación del sistema [32].

Además, las redes de Petri ofrecen un análisis formal y matemático del comportamiento de los sistemas distribuidos, proporcionando herramientas para verificar propiedades críticas como la vivacidad, la seguridad y la ausencia de condiciones de carrera [31]. Esta capacidad de análisis formal es una ventaja significativa, ya que permite identificar problemas potenciales en las fases tempranas del diseño, evitando costosas correcciones en etapas posteriores del desarrollo. El fundamento matemático de las redes de Petri no solo asegura la precisión en el modelado, sino que también contribuye a la creación de sistemas más robustos y confiables [33].

### **4.2.2. Consideraciones para el modelamiento de sistemas distribuidos**

Al modelar sistemas distribuidos con redes de Petri, es clave identificar con precisión los componentes del sistema, como las actividades concurrentes y los recursos compartidos, para capturar efectivamente la concurrencia y las interacciones [32]. Manejar la complejidad del sistema es otro aspecto crucial, donde técnicas como la abstracción y el modularidad resultan

esenciales para simplificar el modelo sin perder detalle. Esto implica descomponer el sistema en subcomponentes o emplear variantes de redes de Petri para representar diversos procesos de manera más manejable [33].

Además, validar el modelo contra el comportamiento real del sistema es fundamental. Esto se realiza a través de simulaciones y análisis formales para asegurar que el modelo cumpla con los requisitos y refleje fielmente el sistema diseñado. Estas consideraciones garantizan que el modelo de Petri sea una herramienta eficaz para el diseño y análisis de sistemas distribuidos, facilitando la identificación y corrección de problemas potenciales desde las etapas iniciales del desarrollo [34].

#### **4.2.3. Ejemplos de modelamiento de sistemas distribuidos con Petri**

El modelado de sistemas de control de tráfico mediante redes de Petri ilustra cómo gestionar de manera eficiente la concurrencia y la sincronización de señales y vehículos en intersecciones [35]. En este contexto, las redes de Petri pueden representar los semáforos como transiciones y los vehículos como tokens, moviéndose a través de lugares que representan diferentes segmentos de la carretera o intersecciones. Esto permite analizar el flujo de tráfico, identificar posibles congestiones y optimizar los tiempos de los semáforos para mejorar el flujo vehicular. Al modelar estos sistemas, se pueden evaluar diversas estrategias de control de tráfico bajo distintas condiciones, facilitando la implementación de soluciones que minimicen los tiempos de espera y maximicen la eficiencia del sistema de tránsito [34].

En sistemas de manufactura distribuida, las redes de Petri ayudan a modelar los procesos de producción, desde la adquisición de materiales hasta la entrega de productos finales, pasando por diversas estaciones de trabajo [36]. Cada estación de trabajo puede ser representada por lugares que indican la disponibilidad de recursos y transiciones que simbolizan las operaciones de manufactura. Los tokens pueden representar tanto los materiales en proceso como los productos finales. Este enfoque permite visualizar y optimizar el flujo de producción, identificar cuellos de botella y asegurar una asignación eficiente de recursos[37]. Al aplicar redes de Petri, se facilita la simulación y mejora de procesos de manufactura, contribuyendo a la eficiencia operativa y la reducción de tiempos de producción en entornos distribuidos [36].

### **4.3. Herramientas y tecnologías para el modelamiento con Petri en sistemas distribuidos**

#### **4.3.1. Herramientas de software para el modelamiento con Petri**

Para el modelado con redes de Petri, existen varias herramientas de software especializadas que facilitan la creación, simulación y análisis de estos modelos. Estas herramientas varían en funcionalidades, desde opciones básicas para dibujar y simular hasta plataformas avanzadas que ofrecen análisis formal y verificación de propiedades [34]. Entre las más destacadas se encuentran WoPeD (Workflow Petri Net Designer), que se enfoca en la facilidad de uso y es ideal para principiantes que desean aprender los fundamentos de las redes de Petri aplicadas a los flujos de trabajo. CPN Tools es otra herramienta poderosa, que permite modelar sistemas complejos utilizando redes de Petri coloreadas, soportando simulaciones detalladas y análisis de comportamiento [35].

Por otro lado, PIPE (Platform Independent Petri net Editor) ofrece un entorno gráfico para la creación y análisis de redes de Petri, destacándose por su capacidad de ser utilizado en diferentes plataformas. Además, TINA (TIme Petri Net Analyzer) es reconocida por su capacidad para analizar redes de Petri temporizadas, brindando soporte para el estudio de sistemas donde el tiempo es un factor crítico. Estas herramientas proporcionan a los diseñadores y analistas de sistemas una gama de opciones para modelar con precisión sistemas distribuidos, desde el diseño conceptual hasta el análisis detallado y la verificación de propiedades específicas del sistema [36].

#### **4.3.2. Integración de Petri con tecnologías de sistemas distribuidos**

La integración de redes de Petri con tecnologías de sistemas distribuidos ofrece un enfoque robusto para el diseño, análisis y optimización de estos sistemas complejos. Las redes de Petri, con su capacidad para modelar la concurrencia, la sincronización y los recursos compartidos de manera precisa, complementan las tecnologías de sistemas distribuidos al proporcionar un marco formal para entender y mejorar su comportamiento [37].

Una de las principales ventajas de esta integración es la posibilidad de simular y analizar el comportamiento de sistemas distribuidos antes de su implementación. Esto permite identificar y resolver problemas potenciales como deadlocks, condiciones de carrera y cuellos de botella en el flujo de procesos. La capacidad de modelar sistemas distribuidos de manera detallada también facilita la tarea de optimizar el rendimiento y la eficiencia, permitiendo a los

diseñadores experimentar con diferentes configuraciones y estrategias de gestión de recursos [38].

Además, las redes de Petri pueden integrarse con herramientas y plataformas de desarrollo de software para sistemas distribuidos, permitiendo una transición fluida del modelo al código. Esta integración puede mejorar significativamente el proceso de desarrollo de software, asegurando que la implementación refleje fielmente el diseño y las especificaciones del modelo [39]. La utilización de redes de Petri en conjunto con tecnologías de sistemas distribuidos no solo mejora la calidad y fiabilidad de los sistemas diseñados, sino que también contribuye a reducir los tiempos de desarrollo y los costos asociados a la detección y corrección de errores en etapas avanzadas [40].

#### **4.3.3. Consideraciones de implementación y despliegue de modelos Petri en sistemas distribuidos**

La implementación y despliegue de modelos de redes de Petri en sistemas distribuidos requieren una serie de consideraciones cuidadosas para asegurar la efectividad del sistema diseñado [38]. Primero, es fundamental garantizar la correspondencia precisa entre el modelo de Petri y la lógica de negocio o los requisitos del sistema distribuido. Esto implica una validación rigurosa del modelo para confirmar que todas las operaciones, sincronizaciones y dependencias están correctamente representadas y que el modelo refleja con exactitud el comportamiento deseado del sistema en el mundo real [40].



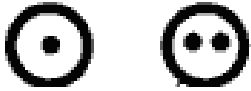
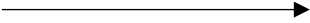
Una segunda consideración importante es la elección de herramientas y plataformas que soporten eficientemente la transición del modelo de Petri al entorno de sistema distribuido. Esto incluye herramientas que faciliten la simulación detallada del modelo, así como plataformas de desarrollo que permitan una implementación efectiva del modelo en código ejecutable. La compatibilidad y la capacidad de integración entre estas herramientas y plataformas son cruciales para minimizar los problemas durante la implementación [39].

Además, al desplegar modelos de Petri en sistemas distribuidos, se debe prestar especial atención a la gestión de recursos y la planificación de la capacidad. Los modelos de Petri pueden ayudar a identificar los requisitos de recursos y a optimizar la asignación de estos, pero es esencial planificar adecuadamente para asegurar que el sistema pueda escalar y manejar la carga de trabajo prevista sin comprometer el rendimiento [40].

#### 4.3.4. Notación y símbolos utilizados en Petri

La notación y los símbolos utilizados en las redes de Petri son fundamentales para modelar sistemas que involucran procesos concurrentes y comunicación entre componentes [39].

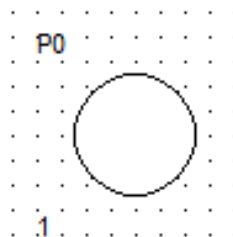
*Tabla 4 Notaciones diagrama de petri*  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

Tipo	Descripción	Notación
<b>Lugares (Places)</b>	Representados por círculos.	
<b>Transiciones (Transitions)</b>	Ilustradas por barras o rectángulos.	
<b>Tokens</b>	Dibujados como puntos o pequeños círculos dentro de los lugares.	
<b>Arcos (Arcs)</b>	Flechas que conectan lugares con transiciones y viceversa.	

#### 4.4. Ejemplos de notaciones

##### 4.4.1. Lugares (Places)

Simbolizan los estados, condiciones o recursos disponibles en el sistema. Los lugares pueden contener tokens, que representan la existencia o cantidad de una determinada condición o recurso.



*Ilustración 41 Lugares*  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

##### 4.4.2. Transiciones (Transitions)

Representan los eventos o acciones que causan cambios en el estado del sistema. Una transición se activa cuando todas las condiciones para su activación se cumplen, lo cual generalmente significa que todos los lugares de entrada tienen el número adecuado de tokens.

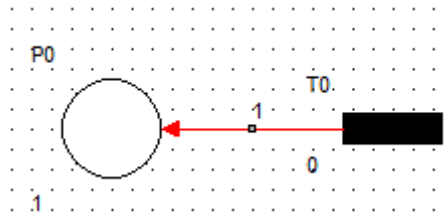


Ilustración 42 Transiciones

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

#### 4.4.3. Tokens

Indican la presencia de una condición o la cantidad de un recurso. El movimiento de tokens entre lugares, a través de las transiciones, modela el cambio de estado en el sistema.

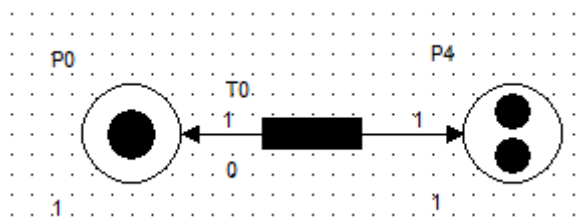


Ilustración 43 Tokens Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

#### 4.4.4. Arcos (Arcs)

Indican la dirección del flujo entre lugares y transiciones. Los arcos de entrada llevan tokens hacia las transiciones, mientras que los arcos de salida mueven tokens desde las transiciones a los lugares. Los arcos pueden tener asociados pesos, los cuales especifican el número de tokens requeridos o producidos por la activación de una transición.

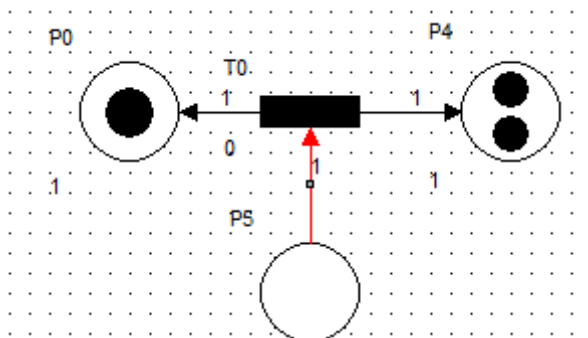


Ilustración 44 Arcos

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA



## **5. Desarrollo de Sistemas basados en IoT**

### **5.1. Metodología de Desarrollo**

#### **5.1.1. Selección y aplicación de metodologías de desarrollo para proyectos basados en IoT.**

La selección de metodologías de desarrollo adecuadas es crucial para el éxito de proyectos basados en Internet de las Cosas (IoT) debido a su inherente complejidad y la necesidad de integrar hardware, software y conectividad. Estos proyectos demandan un enfoque holístico que abarque desde el diseño inicial del dispositivo hasta su implementación final en la nube, soportando iteraciones rápidas, prototipación y pruebas continuas [41].

El desarrollo ágil se presenta como una metodología especialmente adecuada para el IoT, promoviendo la adaptabilidad, la colaboración entre diferentes funciones del equipo y una rápida respuesta a los cambios. Esta flexibilidad es esencial para abordar la naturaleza cambiante de los requisitos en proyectos de IoT, permitiendo una gestión eficaz de la incertidumbre y la complejidad del sistema [42].

Complementando al desarrollo ágil, las prácticas de integración y entrega continuas (CI/CD) facilitan la automatización del proceso de testing y despliegue, mejorando significativamente la calidad y la eficiencia del ciclo de desarrollo. Por otro lado, la adopción de metodologías Lean, con su enfoque en maximizar el valor para el cliente y minimizar el desperdicio, guía el desarrollo hacia características que ofrecen un valor real desde la perspectiva del usuario [41].

#### **5.1.2. Consideración de aspectos como la adquisición de datos, el procesamiento en la nube y la interconexión de dispositivos**

En proyectos basados en IoT, es crucial considerar la adquisición de datos, el procesamiento en la nube y la interconexión de dispositivos como aspectos fundamentales del diseño y desarrollo del sistema. La adquisición de datos implica la recolección eficiente y precisa de información desde una variedad de sensores y dispositivos, lo que requiere un enfoque detallado en la selección de hardware y la implementación de protocolos de comunicación seguros y eficientes [43].

Por otro lado, el procesamiento en la nube es esencial para analizar y almacenar grandes volúmenes de datos generados por dispositivos IoT. Esto no solo implica la elección de una plataforma de nube adecuada, sino también la implementación de soluciones escalables que puedan adaptarse al crecimiento del sistema. La interconexión de dispositivos, que asegura la comunicación fluida entre componentes del sistema IoT, es vital para el rendimiento general y

la funcionalidad del proyecto, demandando atención especial a la compatibilidad y la gestión de la red [42].

Estos aspectos son pilares en el desarrollo de soluciones IoT, dictando la arquitectura del sistema y las decisiones tecnológicas, y por lo tanto deben ser considerados cuidadosamente para garantizar el éxito del proyecto [43].

## **5.2. Alternativas de Solución**

### **5.2.1. Exploración de alternativas de solución para la implementación de sistemas IoT**

Al implementar sistemas IoT, es esencial evaluar diversas opciones tecnológicas para cumplir con los requisitos específicos del proyecto. La selección del hardware adecuado, que varía desde microcontroladores hasta dispositivos avanzados, depende de las necesidades de procesamiento de datos y conectividad. En cuanto al software, la elección de sistemas operativos, lenguajes de programación y plataformas de desarrollo debe facilitar la integración y escalabilidad del sistema, priorizando la seguridad y la privacidad en todas las etapas [44].

Además, la experiencia del usuario final juega un papel crucial en el diseño de interfaces que permitan una interacción intuitiva con el sistema IoT. Considerar todas estas opciones no solo ayuda a construir sistemas IoT eficientes y seguros, sino que también asegura su aceptación y efectividad en el entorno previsto [45].

### **5.2.2. Uso de plataformas y tecnologías como Arduino, Raspberry Pi, y servicios en la nube para IoT (por ejemplo, AWS IoT, Azure IoT, Google Cloud IoT).**

El desarrollo de soluciones IoT se ve notablemente favorecido por el uso de plataformas como Arduino y Raspberry Pi. Arduino, con su simplicidad y facilidad de uso, es perfecto para proyectos básicos que necesitan control de dispositivos y recolección de datos. Por su parte, Raspberry Pi ofrece una mayor capacidad de procesamiento, ideal para proyectos IoT más complejos que demandan procesamiento de datos en el dispositivo o la ejecución de sistemas operativos completos [46].

Los servicios en la nube como AWS IoT, Azure IoT y Google Cloud IoT juegan un papel complementario esencial al proporcionar herramientas avanzadas para la gestión de dispositivos, análisis de datos y medidas de seguridad. Estos servicios facilitan el escalado de proyectos IoT desde simples prototipos hasta soluciones integrales y robustas. Esta sinergia entre hardware accesible y servicios en la nube potentes abre un amplio abanico de posibilidades para los desarrolladores, permitiéndoles implementar soluciones IoT innovadoras y efectivas de manera rápida y eficiente [47].

## 6. Redes de Petri practica

Para la práctica de redes de Petri, se empleó un software denominado "HPSim". Este programa, desarrollado en lenguajes C y C++, facilita la creación y simulación de modelos de redes de Petri.

### 6.1. Problema: Evaluación de solicitud de créditos

Cuando el banco recibe la solicitud de crédito de un cliente procede a solicitar algunos documentos. Si el cliente no trae los documentos dentro de los siguientes 5 días es necesario contactarlo. Si el cliente no continua con la solicitud no se deben esperar los documentos y el proceso debe terminar caso contrario el banco esperará los documentos del cliente.

Por otro lado, si el cliente envía los documentos, no es necesario contactarlo y se procede a evaluar la documentación, si la solicitud de crédito no es aprobada se notifica rechazo caso contrario se notifica aprobación solicitud de créditos.

#### 6.1.1. Modelo

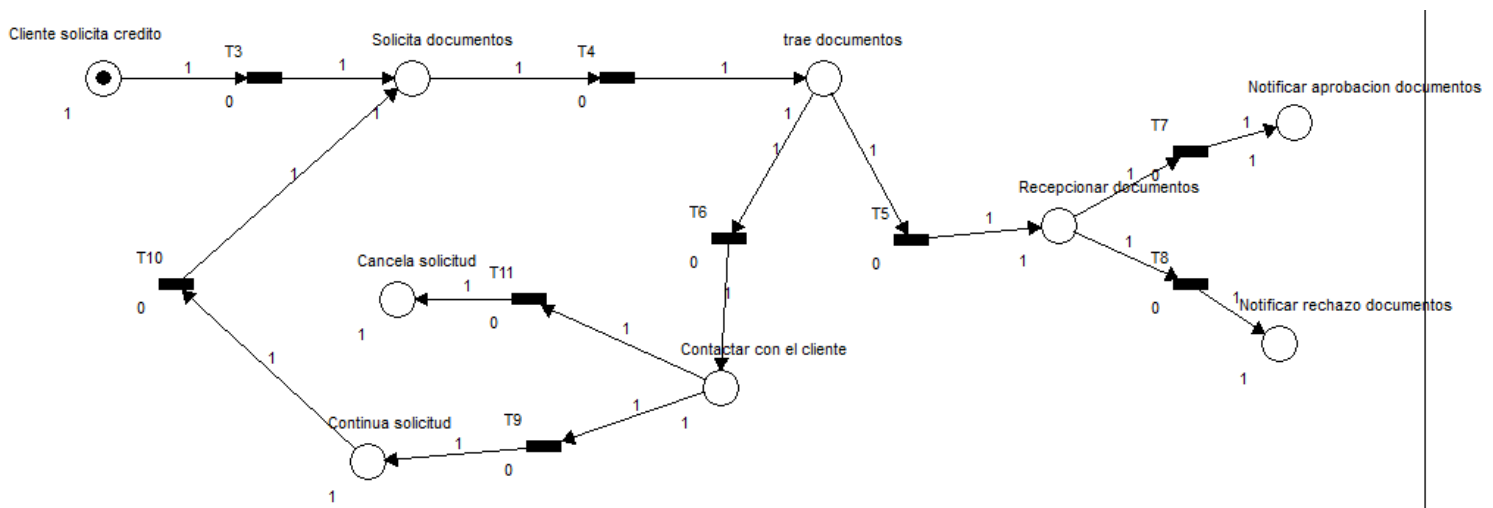


Ilustración 45 Modelo de solicitud de creditos  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

#### 6.1.2. Solución

Para la solución de este caso se implementaría un sistema que ayude a controlar de mejor manera la revisión de documentos para la solicitud de crédito. Además de que este mismo notifique a los clientes si los documentos fueron aprobados o no. Además de que el cliente pueda recibir seguimiento por parte de los trabajadores de la entidad donde desea solicitar el crédito. Esto ocasionaría que el sistema fuera más viable y mas practica tanto para clientes como para los trabajadores.

## Pasos

1. Primero se tienen que crear las acciones o transacciones de solicitud e entrega los documentos ya que con ello iniciara nuestro proceso.

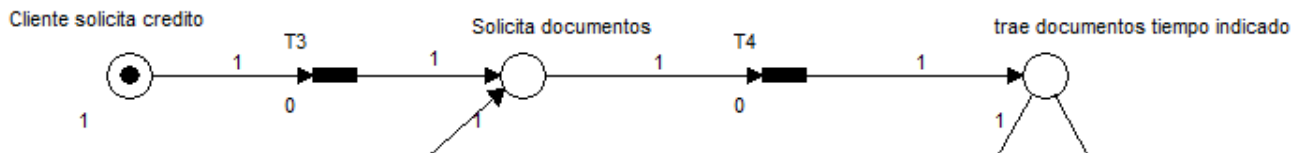


Ilustración 46 Acciones primarias de solicitud de documentos

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

2. Luego cuando los documentos los trae el cliente se bifurcan en dos acciones tanto si la recepción como no los trae en el tiempo indicado o no.

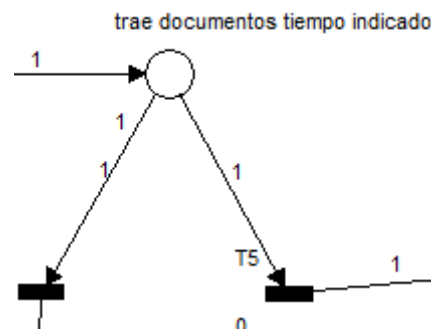


Ilustración 47 Bifurcación de entrega de documentos

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

3. Si el cliente los trae en el tiempo indicado se reciben los documentos y posteriormente la notificación si fueron aprobados o no.

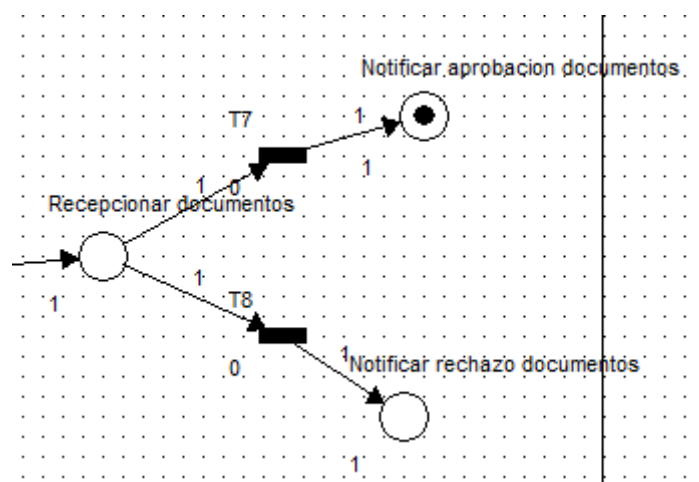


Ilustración 48 Recepción de documento

Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

4. Pero si los documento no los entrega en el tiempo indicado pasa a contactar al cliente y se bifurca en dos transacciones si se cancela la solicitud o no. Además, si el cliente no cancela la solicitud se le vuelven a solicitar los documentos iniciando de nuevo esa transacción.

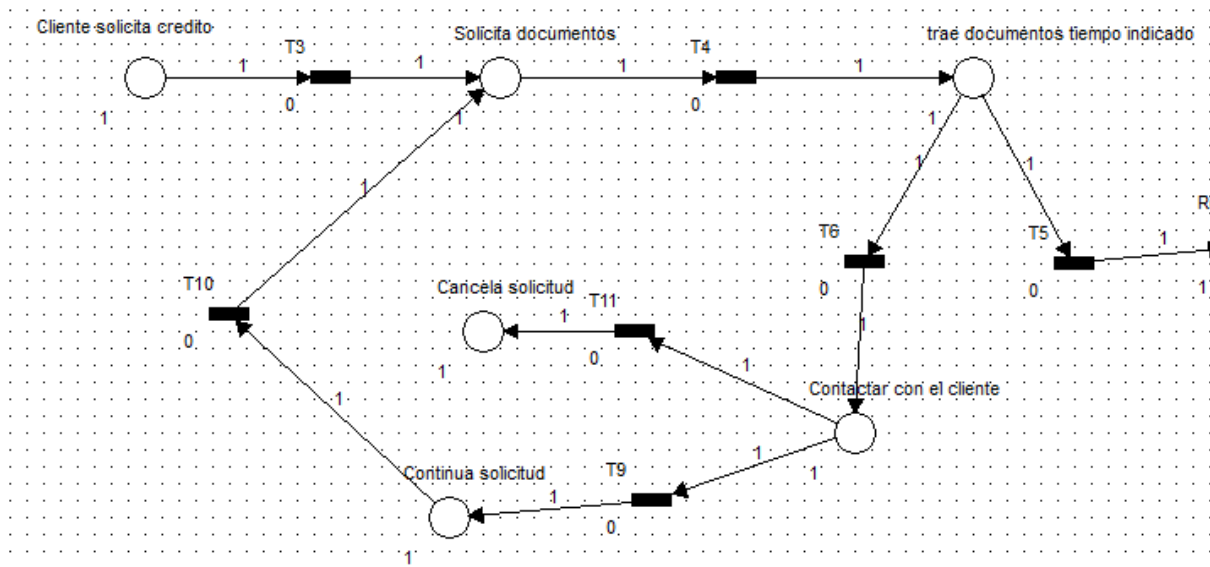
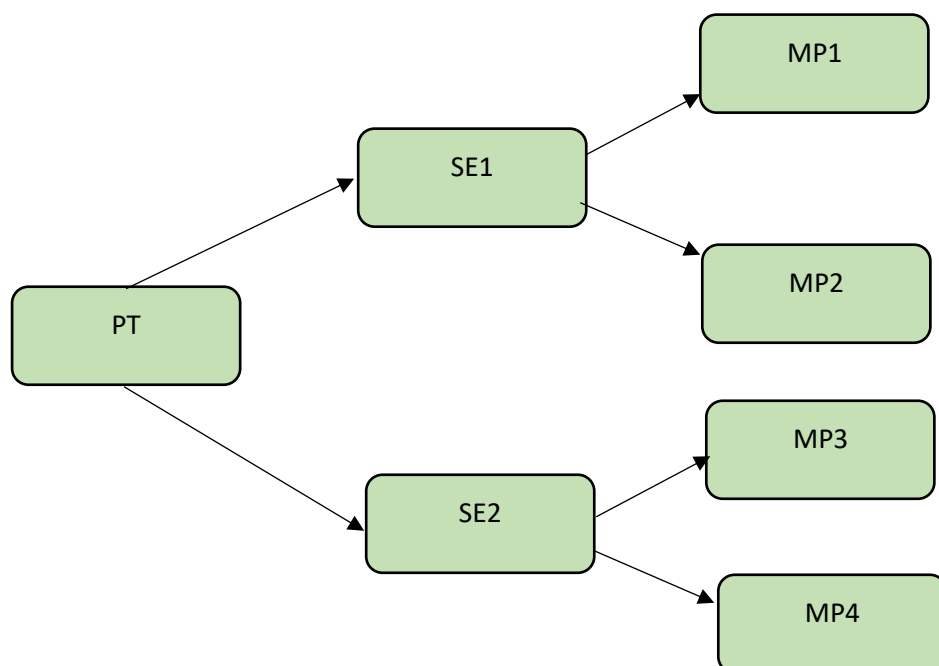


Ilustración 49 Documentos no receptados  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## 6.2. Problema: proceso de fabricación de un producto genérico.

Modelar y simular, el siguiente proceso de fabricación de un producto genérico (PT), teniendo en cuenta el siguiente diagrama de materiales (BOM – Bill Of Materials). Se identifican en el diagrama la materia prima (MP), los sub-ensambles (SE), el producto terminado y las cantidades requeridas (entre paréntesis) por unidad de PT.



Las existencias de materiales y capacidades de almacenamiento se muestran a continuación.

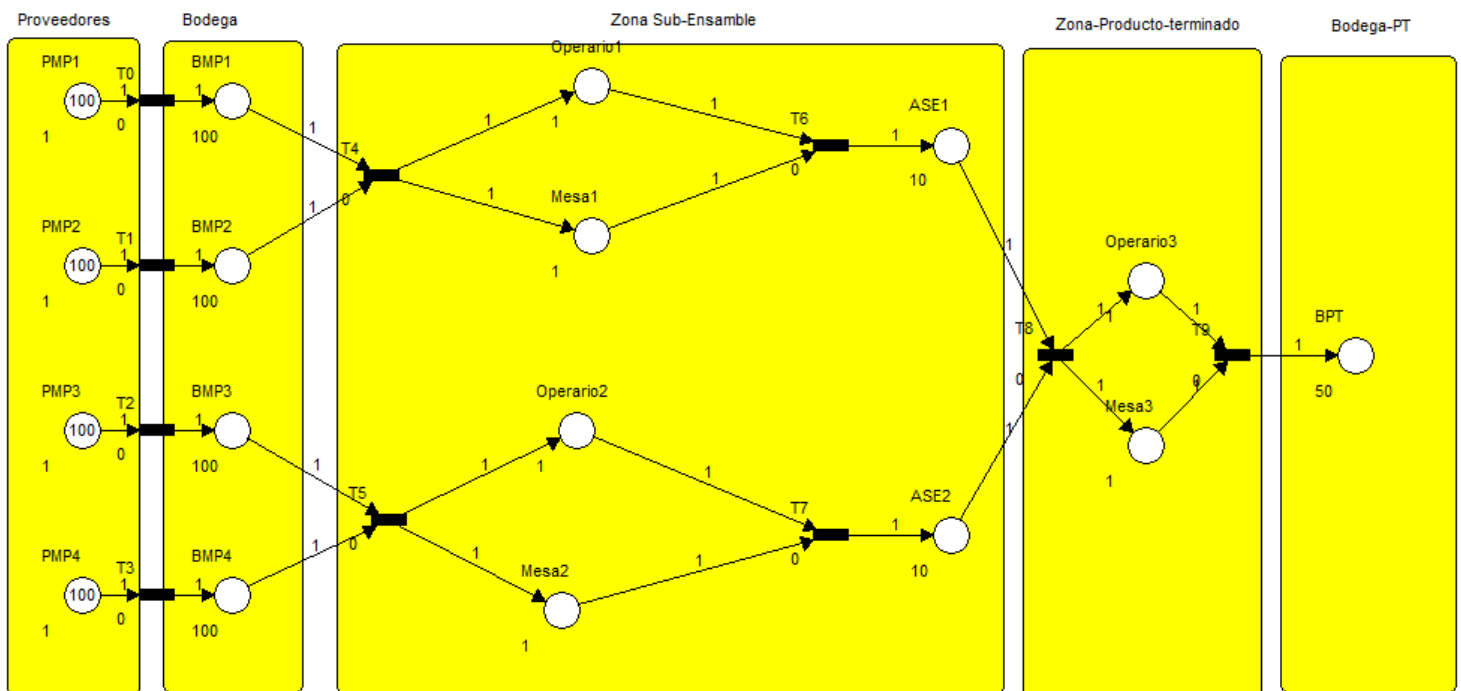
*Tabla 5 Parámetros de la práctica de red de petri*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

<b>Materiales</b>	<b>Existentes</b>	<b>Capacidades</b>
PT	4	50
SE1	3	10
SE2	5	10
MP1	5	100
MP2	2	100
MP3	0	100
MP4	3	100

En este caso, se omitirán los transportes, y solo se requerirán de tres (3) operarios distribuidos en la elaboración de PT, SE1 y SE2. Se pide disponer de 50 PT, teniendo en cuenta las capacidades y existencias del sistema.

1. Tener en cuenta, representar los trabajadores (en conjunto a las operaciones) con la siguiente estructura.
2. Programe las capacidades de las zonas de almacenamiento, las existencias y los pedidos para cada proveedor de materia prima (MP). Suponga que cada MP es entregado por un proveedor distinto.
3. Corra la simulación, y ajuste el modelo para producir las unidades faltantes, hasta completar las 50 unidades requeridas, sin dejar en existencias unidades de MP y SE.

### 6.2.1. Modelo

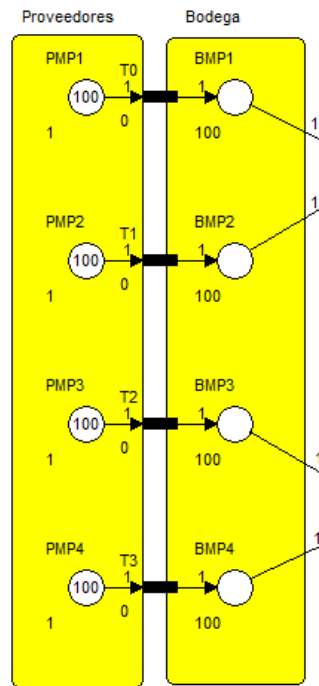


### 6.3. Solución

La solución para este caso sería automatizar mejor las tareas tanto del personal para que de esta manera se optimice el tiempo que transcurre cuando se ensambla y se lleva a bodega el producto. Esa pequeña diferencia de tiempo ocasionaría que el sistema sea más factible en cuestión de ganancia de tiempo y dinero. Además de tener un historial de problemas, ensambles y productos en bodega ante cualquier error que se pueda cometer.

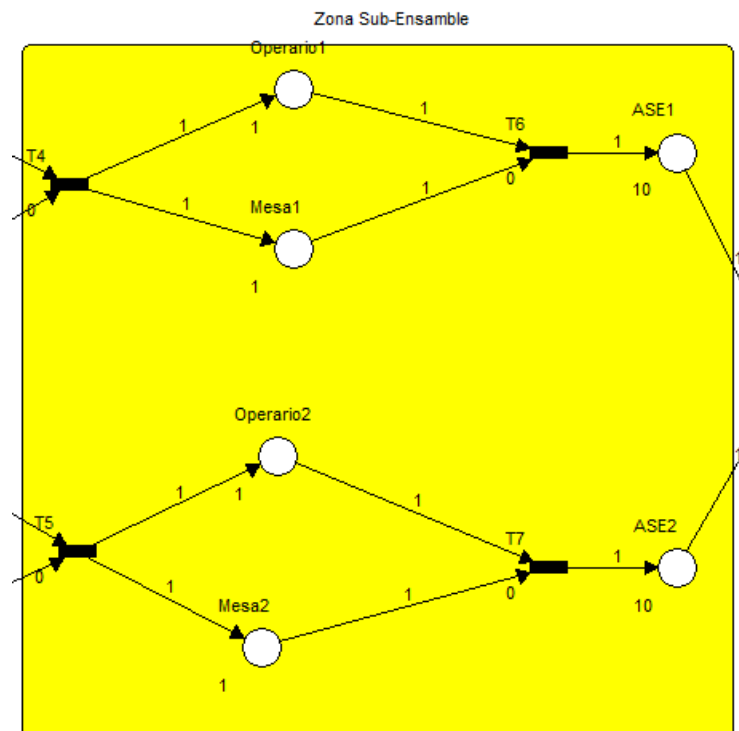
#### Pasos

1. Se tienen que crear dos agrupaciones en la cual en cada una se tienen que agregar cuatro “places” y cuatro transiciones para interconectar los places. Además, en la agrupación de proveedores en cada “place” cambiar la inicialización de cada uno a 100 como lo indica la tabla mostrada en el problema inicial.



*Ilustración 50 Agrupación de proveedores y bodega*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

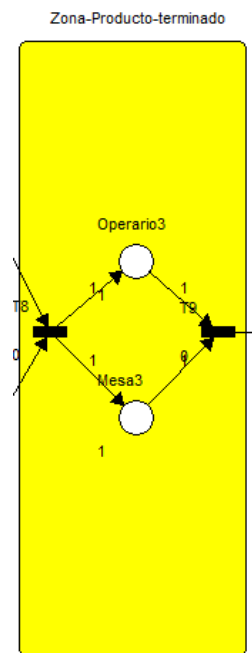
2. Luego en los places de Bodega se unen a una transacción del área de en la cual se bifurcan o se dividen indicando dos “Places” el de operario como para meza todo esto ocurriendo de forma paralela. Esto porque cada individuo tendrá la acción de ensamblar un producto. Hay que destacar que cada ensamble “ASE” tendrá que inicializarse con los parámetros que indica en la tabla inicial del problema.



*Ilustración 51 Zona de sub-ensamble*  
 Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

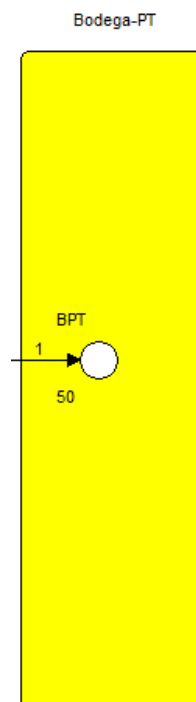


3. Luego en el área de producto terminado se realiza la misma acción de bifurcación que se realizó en la zona de ensamble del producto.



*Ilustración 52 Zona de producto terminado*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

4. Finalmente, en el área de bodega se almacenarán los productos. Hay que tener en cuenta del límite que tiene la bodega por lo cual hay que configurar correctamente este parámetro.



*Ilustración 53 Bodega producto ensamblado*  
*Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA*

## 5. Cambio de parámetros.

Hay que tener en cuenta que para cambiar los parámetros de los controles que estamos utilizando le tendremos que dar clic y con ello a la derecha del software “HPSim” nos desplegara una ventana con todas las propiedades de dicho control y las pondremos cambiar a nuestro gusto.

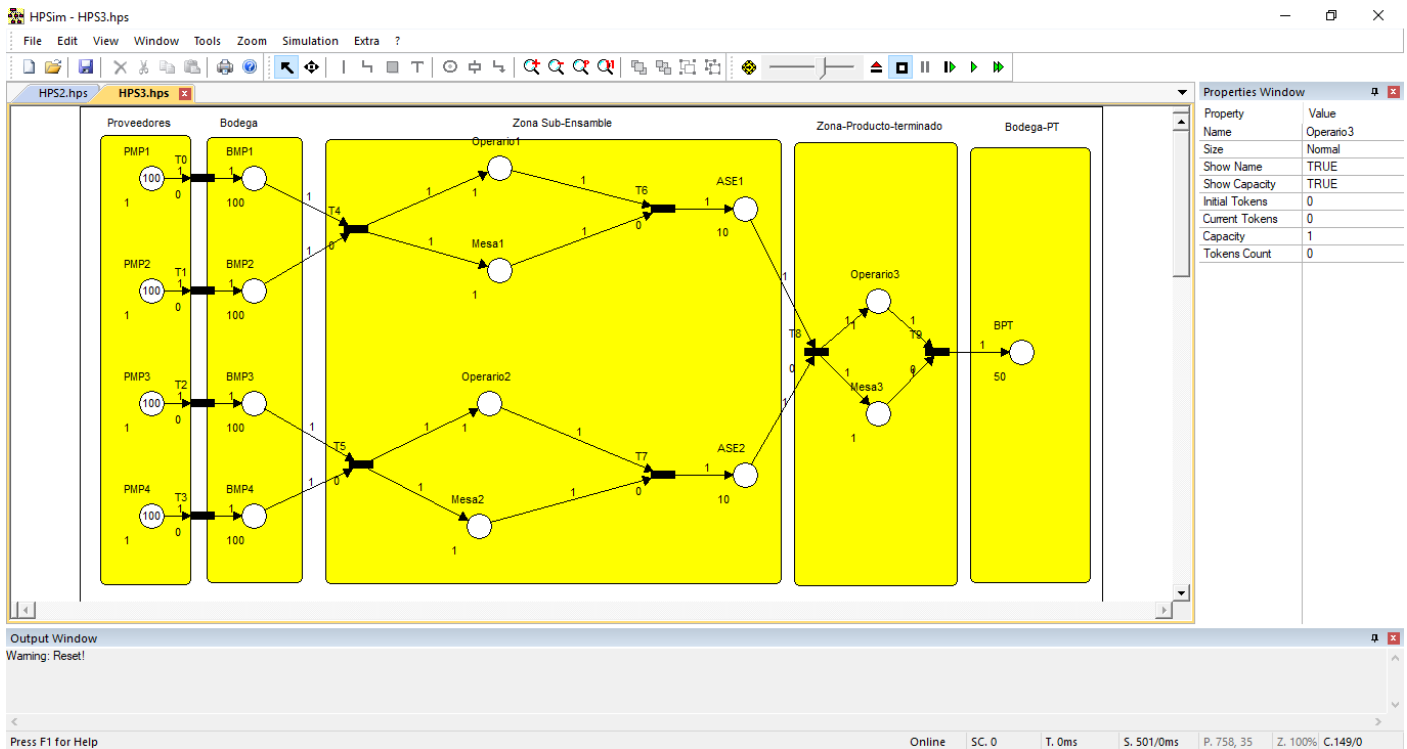


Ilustración 54 Cambio de parámetros  
Realizado por LETURNE PLUAS JHON BYRON & CHICA VALFRE VALESKA SOFIA

## 6.4. GitHub

Para descarga de los ejemplos creados anteriormente puede acceder al siguiente GitHub:

<https://github.com/jhonleturne192005/practicapetri.git>

## **7. Conclusión**

Cada uno de los diagramas nos permiten representar problemas complejos ya sea para tener un mejor análisis del problema, para en futuro conocer la arquitectura de implementación del sistema o para encontrar posibles falencias en el mismo. Pero hay que tener en cuenta que hay diagramas que se enfocan en fines diferentes como creación de base de datos o para conocer estados del sistema. Por lo cual hay que conocer la funcionalidad de cada uno de ellos.

En ejemplo muy sencillo podremos observar que el caso de uso se enfoca mas a los procesos que tendrá el sistema, pero no podremos ver un flujo normal como lo haría un diagrama BPMN o de Petri con lo cual tendríamos un mejor análisis tanto en corrección de errores e implementación del sistema que estamos creando.

Por lo cual observamos que todos son importantes, pero algunos facilitarían el uso para fines que nosotros mismos como creadores de un sistema requerimos. De tal forma es necesario conocer su funcionamiento e implementación para casos donde tengamos que proponer mejores soluciones y así el sistema quede mejor implementado.

## 8. Bibliografía

- [1] I. Benyahia y M. Hilali, “An adaptive framework for distributed complex applications development”, en *Proceedings. 34th International Conference on Technology of Object-Oriented Languages and Systems - TOOLS 34*, IEEE Comput. Soc, pp. 339–349. doi: 10.1109/TOOLS.2000.868984.
- [2] M. S. BABA y T. HERAWAN, “A Large Scale Distributed Virtual Environment Architecture”, *Studies in Informatics and Control*, vol. 24, núm. 2, jun. 2015, doi: 10.24846/v24i2y201504.
- [3] S. Daubrenet y S. Pettifer, “A unifying model for the composition and simulation of behaviors in distributed virtual environments”, en *Proceedings of Theory and Practice of Computer Graphics, 2003.*, IEEE Comput. Soc, pp. 201–208. doi: 10.1109/TPCG.2003.1206949.
- [4] N. Erdogan, Y. E. Selcuk, y O. Sahingoz, “A distributed execution environment for shared java objects”, *Inf Softw Technol*, vol. 46, núm. 7, pp. 445–455, jun. 2004, doi: 10.1016/j.infsof.2003.09.017.
- [5] C. Zhang, B. Li, T. N. Sainath, T. Strohman, y S.-Y. Chang, “UML: A Universal Monolingual Output Layer For Multilingual Asr”, en *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, jun. 2023, pp. 1–5. doi: 10.1109/ICASSP49357.2023.10096228.
- [6] S. Sundaramoorthy, *UML Diagramming*. Boca Raton: Auerbach Publications, 2022. doi: 10.1201/9781003287124.
- [7] J. Farinha y A. R. da Silva, “UML Templates Distilled”, *IEEE Access*, vol. 10, pp. 8709–8727, 2022, doi: 10.1109/ACCESS.2022.3143898.
- [8] I. Häring, “Semi-Formal Modeling of Multi-technological Systems I: UML”, en *Technical Safety, Reliability and Resilience*, Singapore: Springer Singapore, 2021, pp. 227–263. doi: 10.1007/978-981-33-4272-9\_13.
- [9] D. N. H. Weerasinghe, K. A. T. Thiwanka, H. B. C. Jayasith, P. A. D. Onella Natalie, U. U. Samantha Rajapaksha, y A. Karunasena, “Smart UML - Assignment Management Tool for UML Diagrams”, en *2022 4th International Conference on Advancements in Computing (ICAC)*, IEEE, dic. 2022, pp. 114–119. doi: 10.1109/ICAC57685.2022.10025080.
- [10] K. Żyła, A. Ulidowski, J. Wrzos, B. Włodarczyk, K. Kroc, y P. Drozd, “UML – a survey on technical university students in Lublin”, *Journal of Computer Sciences Institute*, vol. 13, pp. 279–282, dic. 2019, doi: 10.35784/jcsi.825.
- [11] D. Torre, Y. Labiche, M. Genero, M. Elaasar, y C. Menghi, “UML Consistency Rules”, en *Proceedings of the 8th International Conference on Formal Methods in Software Engineering*, New York, NY, USA: ACM, oct. 2020, pp. 130–140. doi: 10.1145/3372020.3391554.
- [12] A. Lopes, I. Steinmacher, y T. Conte, “UML Acceptance”, en *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, New York, NY, USA: ACM, sep. 2019, pp. 264–272. doi: 10.1145/3350768.3352575.

- [13] H. Koç, A. M. Erdoğan, Y. Barjakly, y S. Peker, “UML Diagrams in Software Engineering Research: A Systematic Literature Review”, en *The 7th International Management Information Systems Conference*, Basel Switzerland: MDPI, mar. 2021, p. 13. doi: 10.3390/proceedings2021074013.
- [14] G. Vanwormhoudt, M. Allon, O. Caron, y B. Carré, “Template based model engineering in UML”, en *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, New York, NY, USA: ACM, oct. 2020, pp. 47–56. doi: 10.1145/3365438.3410988.
- [15] Y. El Ahmar, S. Gerard, C. Dumoulin, y X. Le Pallec, “Enhancing the communication value of UML models with graphical layers”, en *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, IEEE, sep. 2015, pp. 64–69. doi: 10.1109/MODELS.2015.7338236.
- [16] G. Salaun, “Quantifying the Similarity of BPMN Processes”, en *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, dic. 2022, pp. 377–386. doi: 10.1109/APSEC57359.2022.00050.
- [17] H. Cheng, G. Kang, J. Liu, Y. Wen, B. Cao, y Z. Wang, “BPMN++: Comprehensive Business Process Modeling for Industrial Internet Application”, en *2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCLOUD/SocialCom/SustainCom)*, IEEE, dic. 2022, pp. 548–555. doi: 10.1109/ISPA-BDCLOUD-SocialCom-SustainCom57177.2022.00076.
- [18] I. N. Fatimah *et al.*, “USESPEC to BPMN: Web generator program for use case specification to BPMN”, 2023, p. 040008. doi: 10.1063/5.0103694.
- [19] S. Ipek-Ugay, T. Herrmann, y E. Siegeris, “BPMN-CREATOR – Ein innovatives Tool zur vollautomatischen Digitalisierung zuvor manuell erstellter Geschäftsprozessmodelle”, en *Angewandte Forschung in der Wirtschaftsinformatik 2022*, GITO mbH Verlag, 2022, pp. 305–315. doi: 10.30844/AKWI\_2022\_20.
- [20] M. Zhang, H. Li, Z. Huang, Y. Huang, y F. Bao, “Analyzing Realizability of BPMN Choreographies”, en *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCOM/CyberSciTech)*, IEEE, sep. 2022, pp. 1–6. doi: 10.1109/DASC/PiCom/CBDCOM/Cy55231.2022.9927990.
- [21] Y. Falcone, G. Salaün, y A. Zuo, “Probabilistic Model Checking of BPMN Processes at Runtime”, 2022, pp. 191–208. doi: 10.1007/978-3-031-07727-2\_11.
- [22] A. Contreras, Y. Falcone, G. Salaün, y A. Zuo, “WEASY: A Tool for Modelling Optimised BPMN Processes”, 2022, pp. 110–118. doi: 10.1007/978-3-031-20872-0\_7.
- [23] M. de O. Nunes, R. M. Pillat, y T. C. de Oliveira, “Identifying Support for Knowledge-Intensive Processes in BPMN and its Extensions”, en *Proceedings of the XIX Brazilian Symposium on Information Systems*, New York, NY, USA: ACM, may 2023, pp. 451–458. doi: 10.1145/3592813.3592937.

- [24] P. von Olberg y L. Strey, “Approach to Generating Functional Test Cases from BPMN Process Diagrams”, en *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, IEEE, ago. 2022, pp. 185–189. doi: 10.1109/REW56159.2022.00042.
- [25] Y. Kirikkayis, F. Gallik, y M. Reichert, “Towards a Comprehensive BPMN Extension for Modeling IoT-Aware Processes in Business Process Models”, 2022, pp. 711–718. doi: 10.1007/978-3-031-05760-1\_47.
- [26] E. Díaz, J. I. Panach, S. Rueda, y J. Vanderdonckt, “An empirical study of rules for mapping BPMN models to graphical user interfaces”, *Multimed Tools Appl*, vol. 80, núm. 7, pp. 9813–9848, mar. 2021, doi: 10.1007/s11042-020-09651-6.
- [27] P. Bocciarelli, A. D’Ambrogio, A. Giglio, y E. Paglia, “BPMN-Based Business Process Modeling and Simulation”, en *2019 Winter Simulation Conference (WSC)*, IEEE, dic. 2019, pp. 1439–1453. doi: 10.1109/WSC40007.2019.9004960.
- [28] B. Boonmepit y T. Suwannasart, “Test Case Generation from BPMN with DMN”, en *Proceedings of the 2019 3rd International Conference on Software and e-Business*, New York, NY, USA: ACM, dic. 2019, pp. 92–96. doi: 10.1145/3374549.3374582.
- [29] A. Choquet-Geniet y P. Richard, “Petri Nets: A Graphical Tool for System Modelling and Analysis”, en *Software Specification Methods*, London: Springer London, 2001, pp. 241–257. doi: 10.1007/978-1-4471-0701-9\_14.
- [30] G. S. Hura, “Petri nets (abstract)”, en *Proceedings of the 1986 ACM fourteenth annual conference on Computer science - CSC ’86*, New York, New York, USA: ACM Press, 1986, p. 495. doi: 10.1145/324634.325331.
- [31] B. Samanta y B. Sarkar, “Application of Petri nets for systems modelling and analysis”, *OPSEARCH*, vol. 49, núm. 4, pp. 334–347, dic. 2012, doi: 10.1007/s12597-012-0083-4.
- [32] P. B. Thomas, “The Petri net as a modeling tool”, en *Proceedings of the 14th annual Southeast regional conference on - ACM-SE 14*, New York, New York, USA: ACM Press, 1976, p. 172. doi: 10.1145/503561.503597.
- [33] S. Pujari y S. Mukhopadhyay, “Petri Net: A Tool for Modeling and Analyze Multi-agent Oriented Systems”, *International Journal of Intelligent Systems and Applications*, vol. 4, núm. 10, pp. 103–112, sep. 2012, doi: 10.5815/ijisa.2012.10.11.
- [34] M. Barad, “Petri Nets—A Versatile Modeling Structure”, *Appl Math (Irvine)*, vol. 07, núm. 09, pp. 829–839, 2016, doi: 10.4236/am.2016.79074.
- [35] M. Rawson y M. G. Rawson, “Petri Nets for Concurrent Programming”, en *2022 IEEE/ACM Fifth Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware (IPDRM)*, IEEE, nov. 2022, pp. 17–24. doi: 10.1109/IPDRM56689.2022.00008.
- [36] J.-P. Signoret y A. Leroy, “Petri Net Modelling”, 2021, pp. 587–660. doi: 10.1007/978-3-030-64708-7\_33.

- [37] D. Bedok, “Application of Petri-nets in object-oriented environment”, en *2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI)*, IEEE, nov. 2016, pp. 000117–000122. doi: 10.1109/CINTI.2016.7846389.
- [38] W. J. Thong y M. A. Ameen, “A Survey of Petri Net Tools”, 2015, pp. 537–551. doi: 10.1007/978-3-319-07674-4\_51.
- [39] M. A. Blätke, C. Rohr, M. Heiner, y W. Marwan, “A Petri-Net-Based Framework for Biomodel Engineering”, 2014, pp. 317–366. doi: 10.1007/978-3-319-08437-4\_6.
- [40] S. Böhm y M. Běhálek, “Usage of petri nets for high performance computing”, en *Proceedings of the 1st ACM SIGPLAN workshop on Functional high-performance computing*, New York, NY, USA: ACM, sep. 2012, pp. 37–48. doi: 10.1145/2364474.2364481.
- [41] S. Matuska, J. Machaj, M. Hutar, y P. Brida, “A Development of an IoT-Based Connected University System: Progress Report”, *Sensors*, vol. 23, núm. 6, p. 2875, mar. 2023, doi: 10.3390/s23062875.
- [42] M. G. Do Nascimento, J. M. N. David, M. A. R. Dantas, R. M. M. B. Villela, V. S. de Andrade Menezes, y F. A. B. Colugnati, “An Architecture to Support the Development of Collaborative Systems in IoT Context”, en *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, may 2023, pp. 1722–1727. doi: 10.1109/CSCWD57460.2023.10152672.
- [43] G. Guerrero-Ulloa, C. Rodríguez-Domínguez, y M. J. Hornos, “Agile Methodologies Applied to the Development of Internet of Things (IoT)-Based Systems: A Review”, *Sensors*, vol. 23, núm. 2, p. 790, ene. 2023, doi: 10.3390/s23020790.
- [44] R. Paesani, G. Paolone, P. Di Felice, D. Iachetti, y M. Marinelli, “iFogSim Simulations on IoT Computational Alternatives”, en *ASEC 2022*, Basel Switzerland: MDPI, dic. 2022, p. 44. doi: 10.3390/ASEC2022-13857.
- [45] A. Tripathy y B. Singh, “A Study of AES Software Implementation for IoT Systems”, en *2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, IEEE, nov. 2022, pp. 1–4. doi: 10.1109/ICICT55121.2022.10064507.
- [46] N. K. Jangid y M. K. Gupta, “Protecting software design in cloud using AWS IoT”, en *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, New York, NY, USA: ACM, jun. 2022, pp. 561–562. doi: 10.1145/3498361.3538784.
- [47] S. Mallick, T. Singh, y S. Podder, “IoT based Smart Mirror Using Raspberry PI”, *Int J Res Appl Sci Eng Technol*, vol. 11, núm. 1, pp. 460–467, ene. 2023, doi: 10.22214/ijraset.2023.48583.