udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA Y CIENCIAS

Tutoría para Examen de Grado

Sesión 2: Modelamiento y Conceptos sobre Pruebas de SW

Docente: Mauricio Hidalgo Barrientos

Temario de la sesión

- Recordar UML y los diagramas de Modelamiento de Software
- Recordar aspectos relacionados con pruebas de Software

MODELAMIENTO DE SOFTWARE

Lenguaje Unificado de Modelado – UML

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico “de facto” para:

- Visualizar
- Especificar y
- Documentar

cada una de las partes que comprende el desarrollo de software.

¿Qué “cosas” podríamos modelar utilizando UML?

Lenguaje Unificado de Modelado – UML

¿Qué otorga (para que sirve) UML?

UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

¿Qué podemos Modelar con UML? (categorías de modelamiento)

- Diagramas estructurales

Son aquellos que muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción.

- Diagramas de comportamiento

Son aquellos que muestran el comportamiento dinámico de los objetos en el sistema.

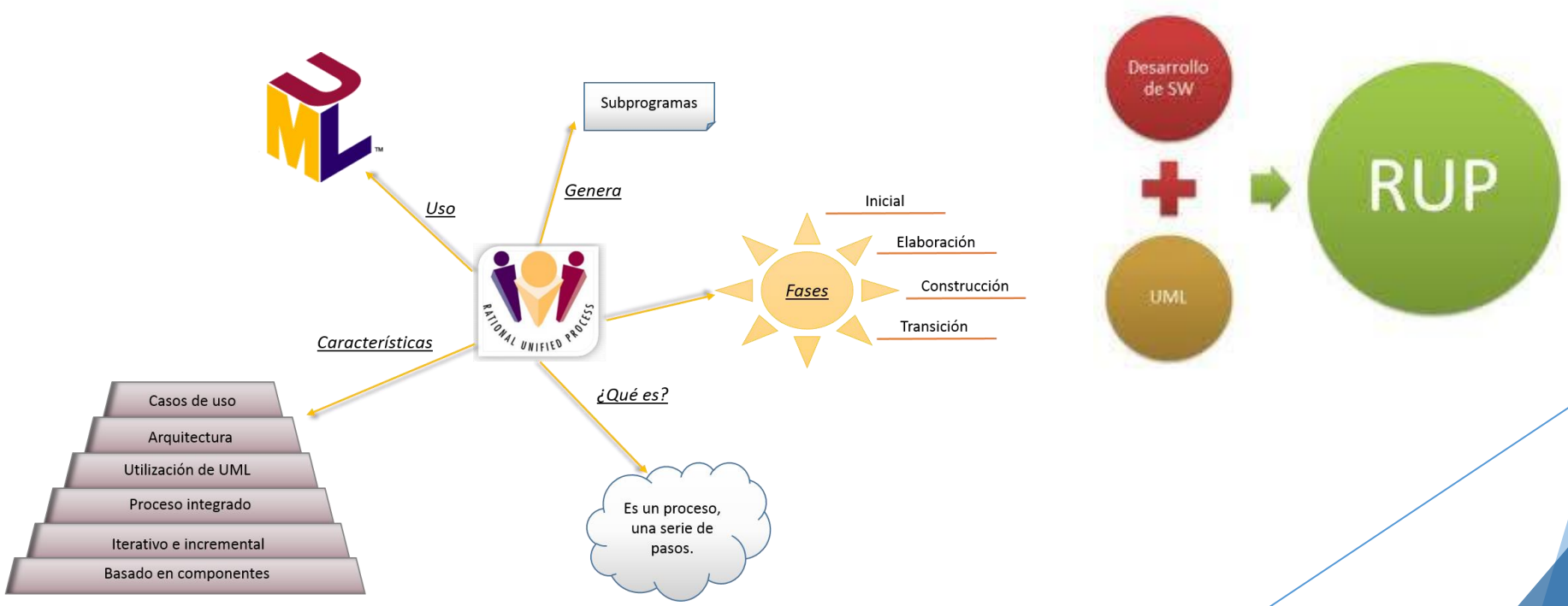
Fuente: Patricio Salinas y Nancy Histchfeld
DCC de la Universidad de Chile

Lenguaje Unificado de Modelado – UML

IMPORTANTE

UML no es un método de desarrollo por sí mismo; sin embargo, que fue diseñado para ser compatible con los métodos de desarrollo de software orientado a objetos (sobre todo con RUP).

¿Por qué UML es compatible con RUP?



Ventajas de UML

Ofrece una manera de visualizar planos de arquitectura de un sistema en un diagrama, que incluye elementos tales como:

- Cualquier actividad (usos)
- Los componentes individuales del sistema
- El cómo pueden interactuar con otros componentes de software
- Cómo funcionará el sistema en cuanto a usuario y otros softwares
- Cómo interactúan las entidades con otros (componentes e interfaces)

Discusión en Clase

¿UML sirve solo para Diseño de SW?

Tipos de Diagrama UML

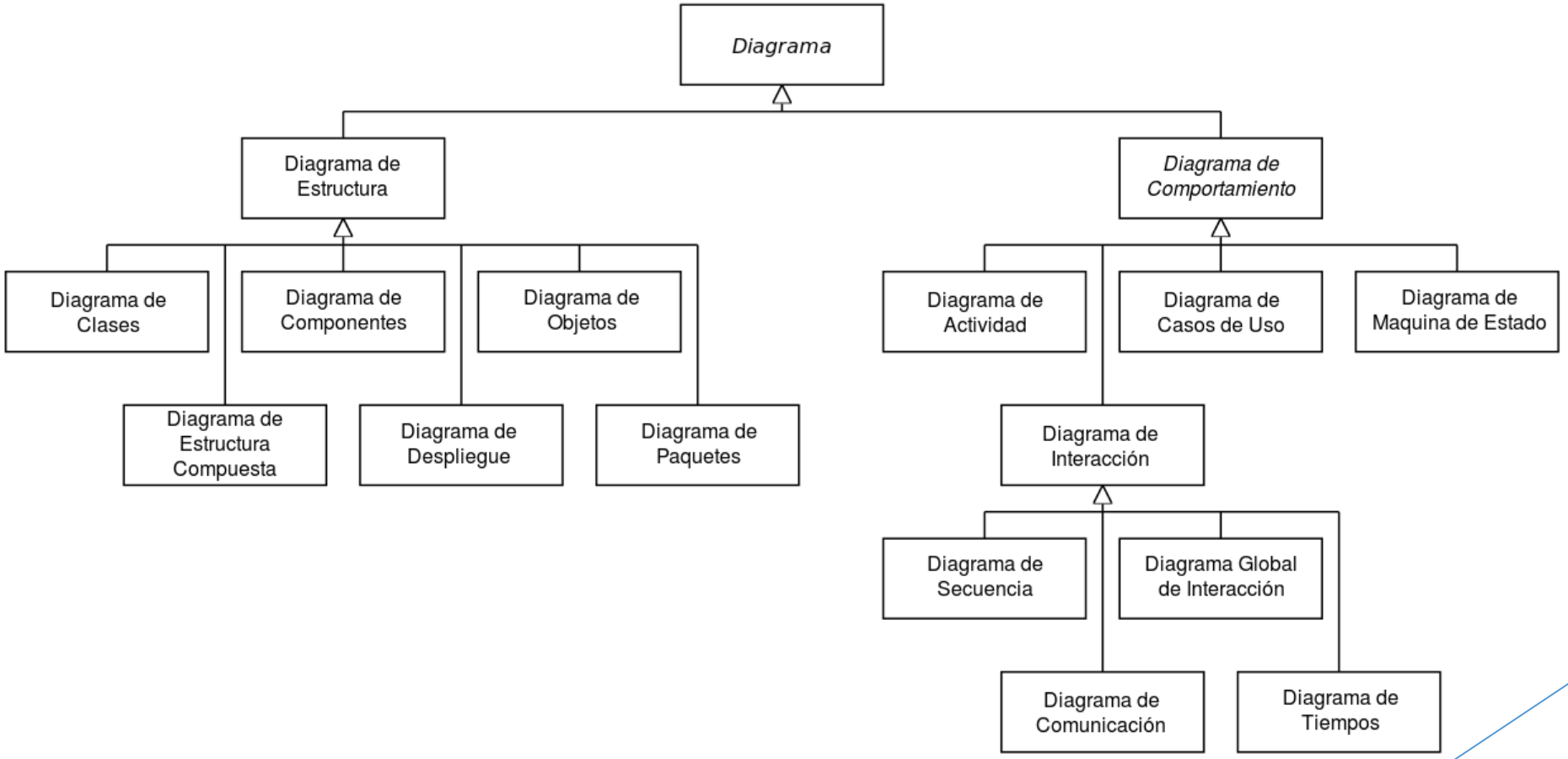
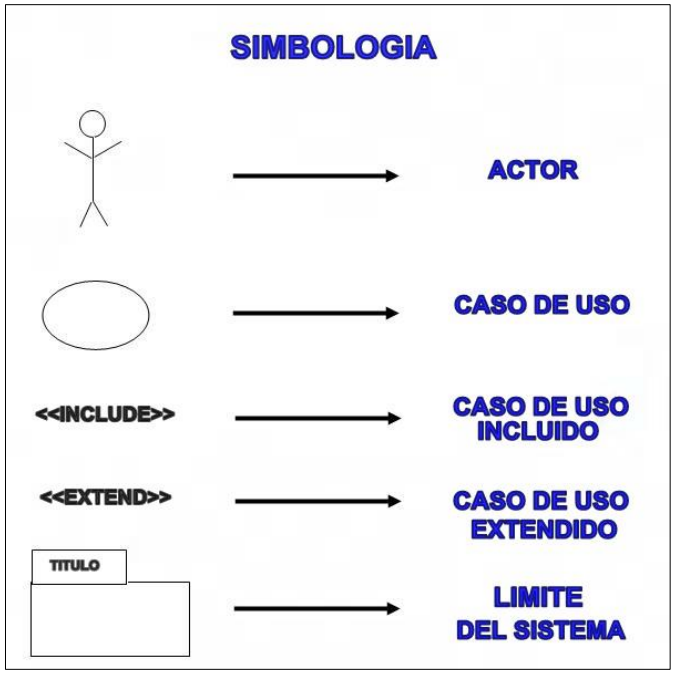


Diagrama de Casos de Uso

Son una técnica de modelamiento que permiten representar

- La forma en como un Cliente (Actor) opera con el sistema en desarrollo
- El tipo y orden en como los elementos interactúan (operaciones o casos de uso)



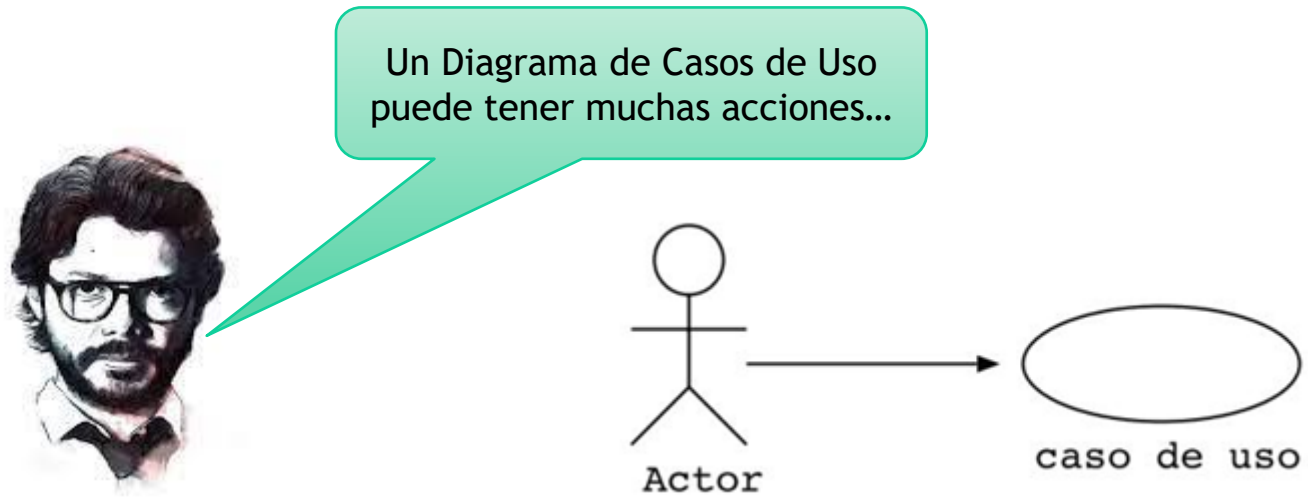
Diagramas de Casos de Uso: Componentes

- **Actor**

Se refiere a la persona (o sistema externo) que realiza acciones en el sistema. Se representa, usualmente, bajo el dibujo de una persona.

- **Caso de Uso**

Se refiere a una acción que se puede realizar en el sistema. Se representa como un óvalo con el nombre del Caso de Uso en su interior



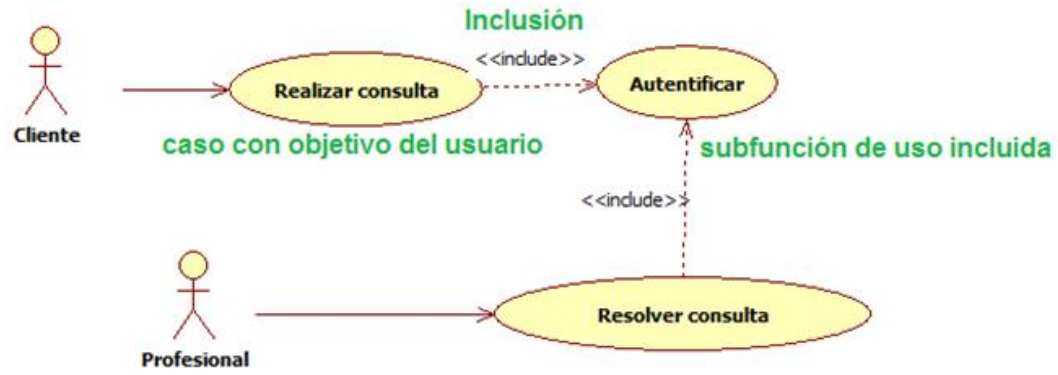
Diagramas de Casos de Uso: Relaciones

Relación directa

Es la forma de representar los comportamientos comunes. En simple, son acciones directas y se representan con una flecha de origen a fin.

Relación de inclusión

- Es una forma de interacción donde un caso de uso dado puede “depender" otro caso de uso. En simple, un caso de uso “requiere” de uno o más casos de uso.
- Se representa con una flecha desde el caso de uso “que ocupa” hacia los casos de uso “utilizados”.



Diagramas de Casos de Uso: Relaciones

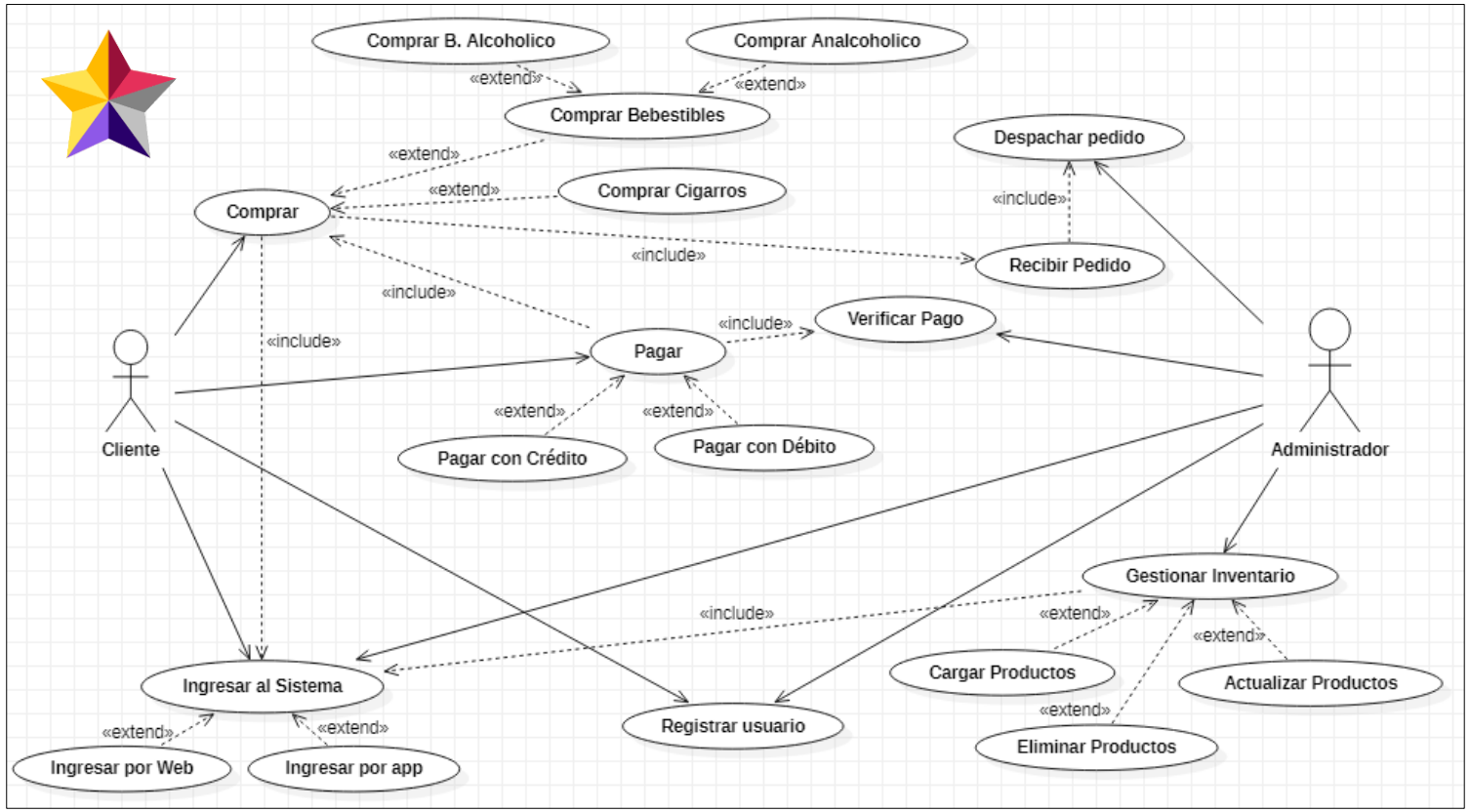
Relación de extensión

- La extensión, es el conjunto de objetos a los que se aplica un concepto.
- Los objetos de la extensión son los ejemplos o instancias de los conceptos. En simple, un caso de uso se puede "especializar" en otros casos de uso.
- Se representa con una flecha que comienza en el caso de uso "especializado" y termina en el caso de uso "general".

Fuente imágenes previas
https://www.seas.es/blog/wp-content/uploads/imagecache/screenshot184.png

Diagramas de Casos de Uso: Ejemplo

El siguiente Diagrama de Casos de Uso representa (en lo simple) una Botillería Virtual.



Nota al margen

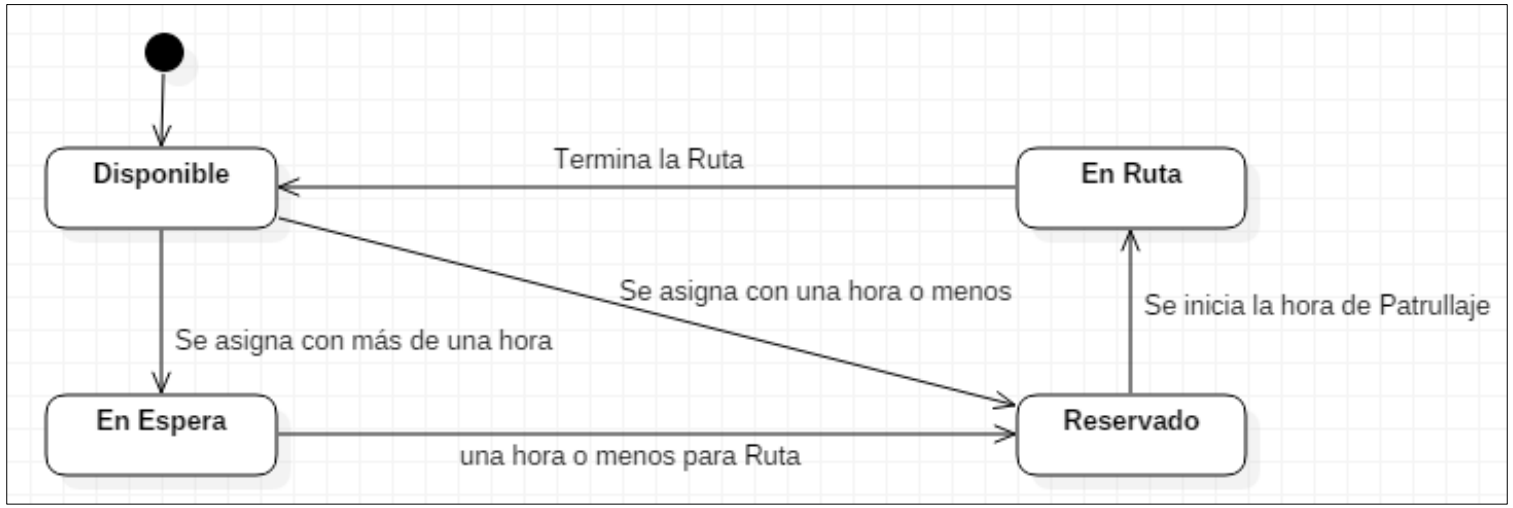
El Diagrama se realizó en un SW llamado StarUML

Diagramas de Estados

Es un diagrama que muestra la secuencia correcta de un caso de uso e indica los eventos para pasar de un estado a otro y las acciones que se genera con ello.

Ejemplo

Diagrama de Estados para un objeto “Patrullero”



Diagramas de Actividad

Es considerado una extensión del diagrama de estados y se utiliza para modelar una secuencia de acciones y condiciones dentro de un proceso. Adiciona al diagrama de estados:

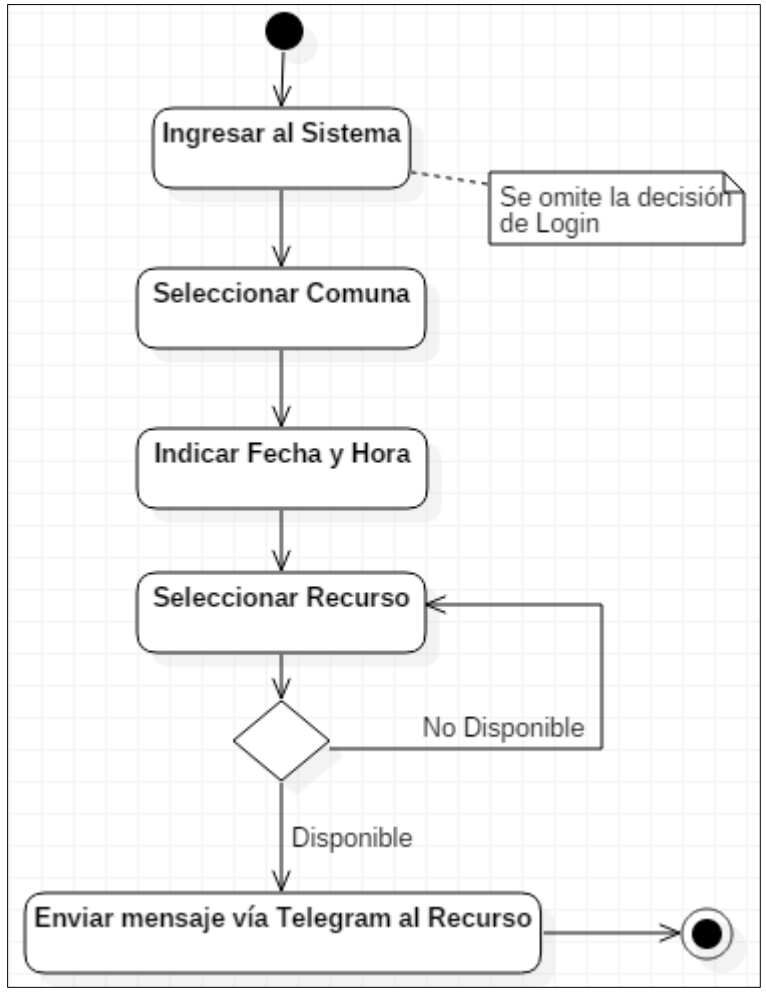
Branch

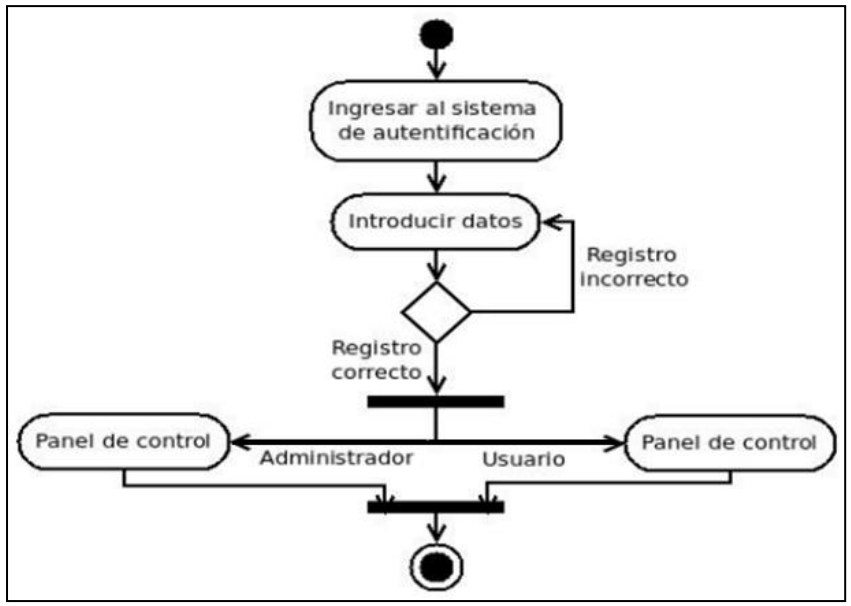
Se utiliza cuando existe la posibilidad de mas de una transición (similar a un if/else). Es representado por un rombo.

Join y Fork

- Join ⇒ Ocurre al fusionar dos o más transiciones. Se representa con una línea negra sólida, perpendicular a las líneas de transición.
- Fork ⇒ Representa una ramificación en dos o más transiciones obligadas. Se representa igual que un Join.

Diagramas de Actividad: Ejemplos





Diagramas de Clases

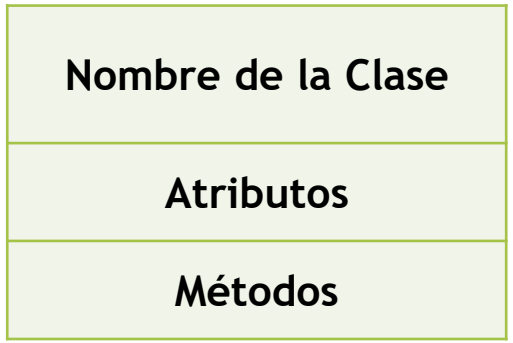
Es un diagrama estático de análisis y diseño que sirve para visualizar las relaciones entre las clases que involucran el sistema.

¿Qué muestran?

- Los atributos de una clase
- Los métodos de una clase
- Las restricciones a que se ven sujetas las clases según la forma en que se conectan.

¿Para qué sirven?

- Permite realizar una abstracción del dominio
- Permite formalizar el análisis de conceptos
- Permite definir una solución del diseño



Diagramas de Clases: Conceptos

Atributos

Son características que corresponden a un objeto. Por ejemplo: Color, material, cantidad, ubicación, entre otros.

Métodos

Son aquellas acciones (verbos) que se pueden realizar con y para un objeto. Por ejemplo: Abrir, cerrar, buscar, cancelar, acreditar, cargar, etc.

Interfaz

Conjunto de operaciones que permiten a un objeto comportarse de cierta manera. Define los requerimientos mínimos de un sistema.

Herencia

Es la posibilidad de crear un objeto “derivado” a través de un objeto “base”.

Diagramas de Clases: Ejemplo

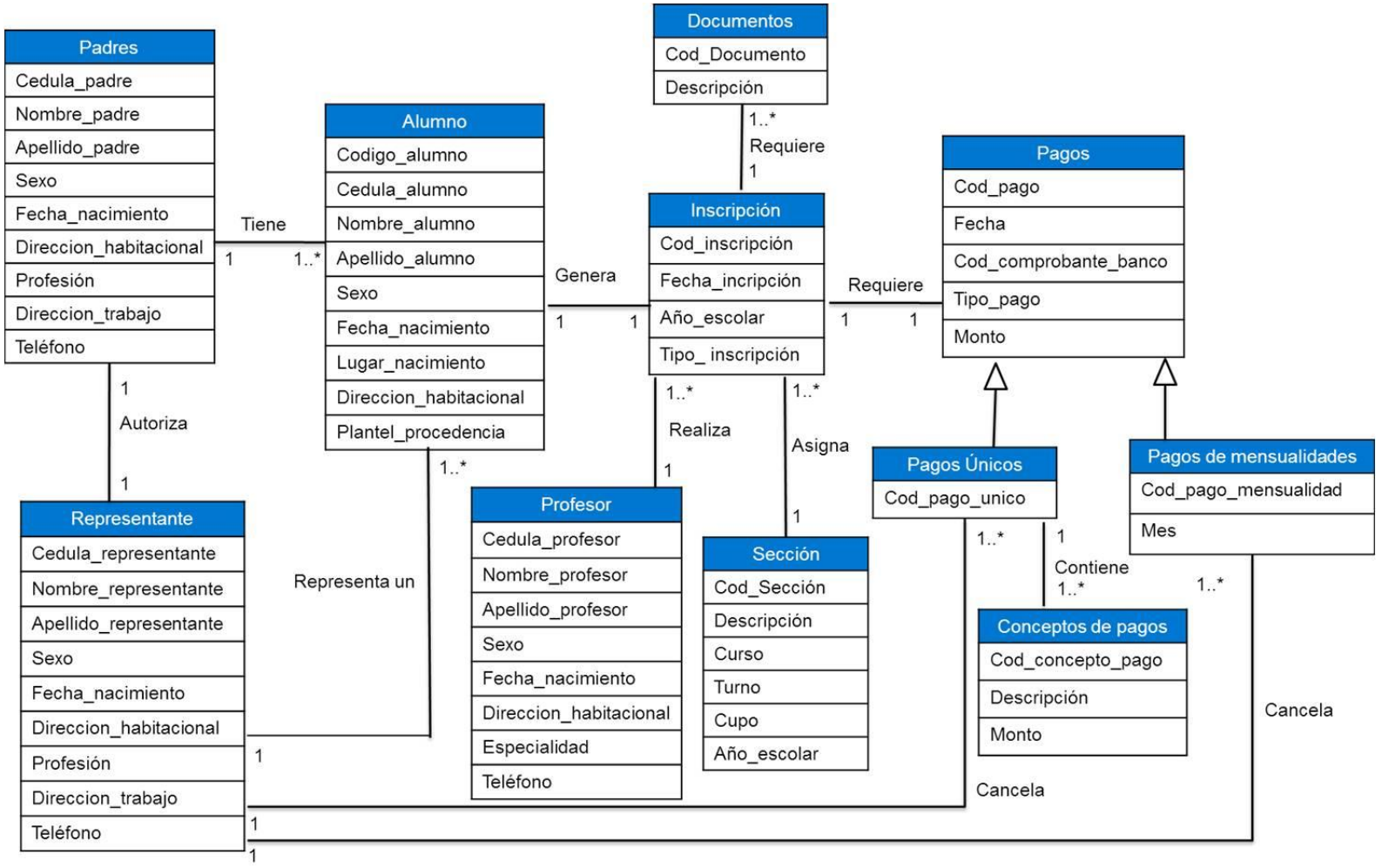


Diagrama de Secuencia

Es la “particularización” de cada caso de uso y se utiliza para modelar interacción entre objetos en un sistema ya que se basa en el “intercambio de mensajes” entre objetos.

Los mensajes se dibujan cronológicamente desde la parte superior hasta la inferior del diagrama y puede incluir mensajes sincrónicos y asincrónicos:

- Mensajes sincrónicos: Son llamadas a los métodos del objeto que recibe el mensaje. Se representan con flechas de cabeza rellena.
- Mensajes asincrónicos: Son hilos de ejecución en una secuencia. Se representan con flechas de cabeza abierta.

Diagrama de Secuencia: Ejemplo

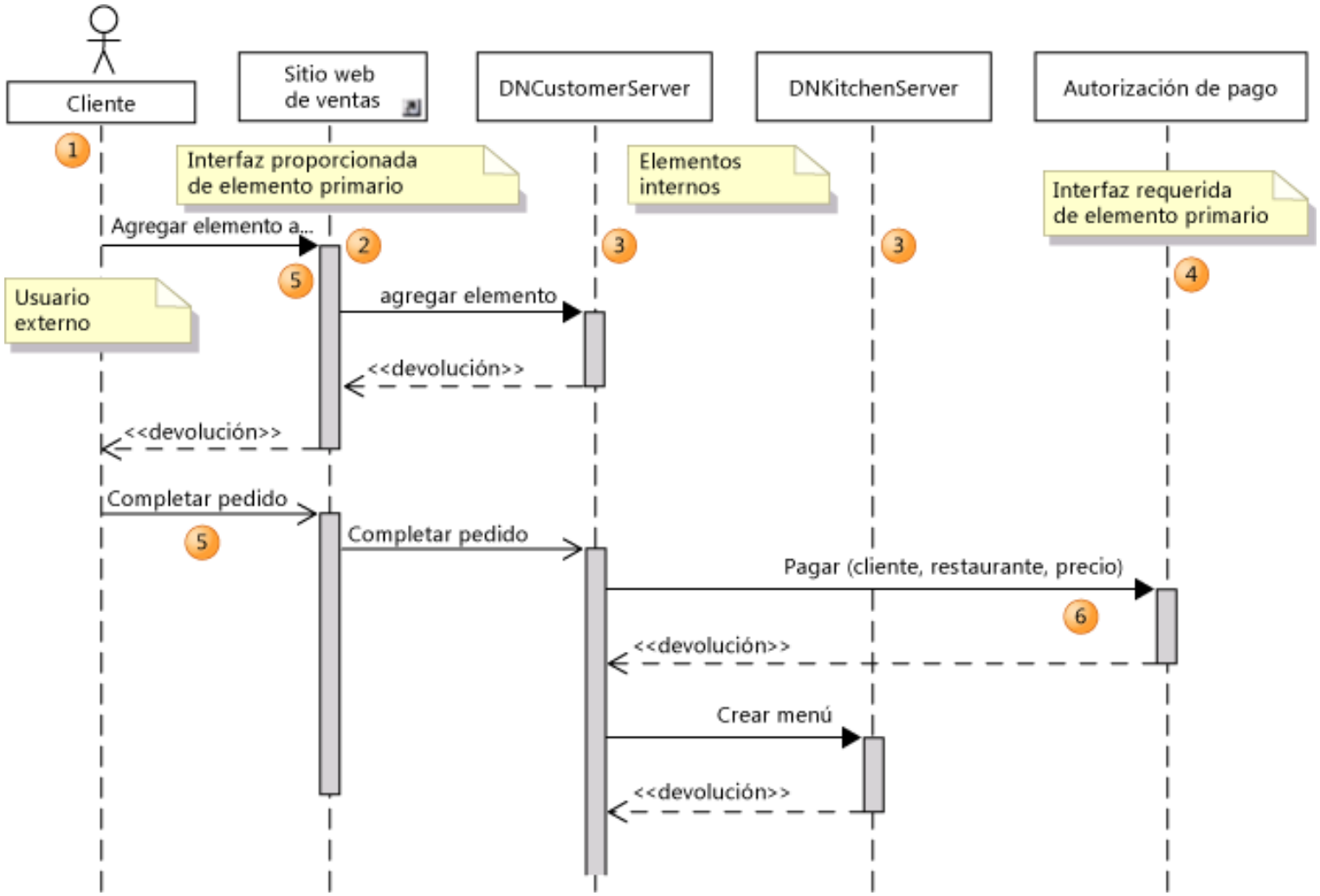


Diagrama de Paquetes

Es un diagrama estático que representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas (paquetes) y las dependencias entre esas agrupaciones.

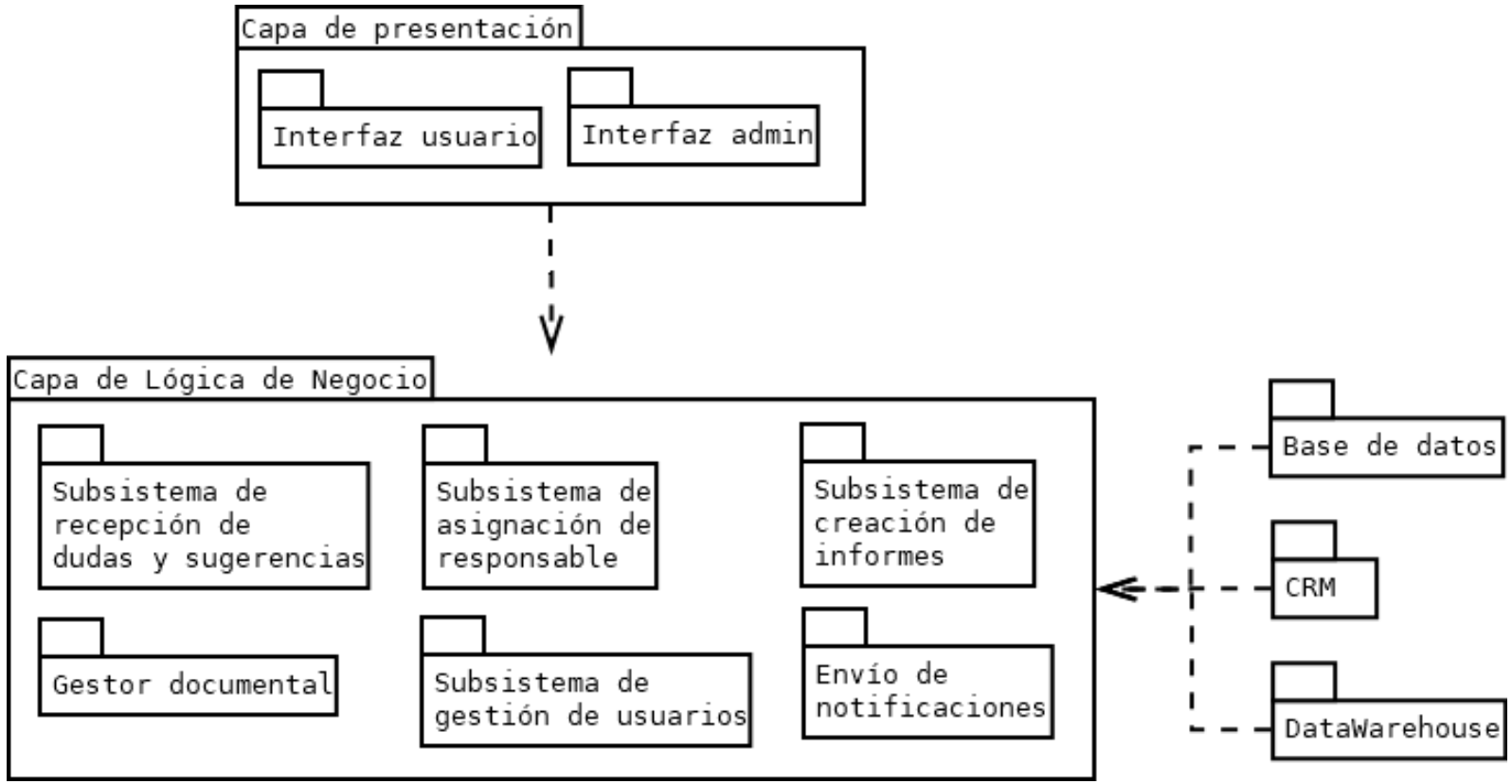
- ***Paquete***

Es un espacio de nombres encapsulado dentro del cual todos los nombres deben ser únicos. Visualmente se representa como una carpeta.

- ***Dependencia***

Indica que un elemento de un paquete requiere a otro de un paquete distinto. Visualmente se representa mediante una flecha con inicio en el paquete que depende del otro, es decir, la flecha parte del elemento de origen y apunta hacia el elemento destino.

Diagrama de Paquetes: Ejemplo



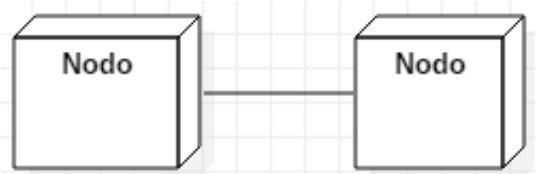
Ejemplo obtenido de: https://diagramasuml.com/paquetes/

Diagrama de Despliegue

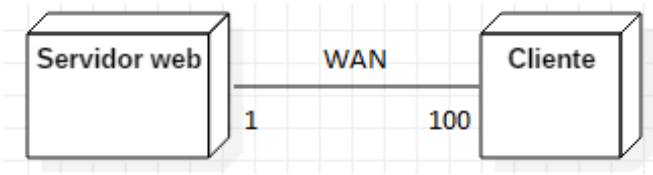
Es un diagrama estático que representa la disposición física de los artefactos del sistema. Esto incluye tanto el Hardware como el Software. En general, implica modelar el Hardware sobre el cuál se ejecutará el software.

Elementos

Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones.



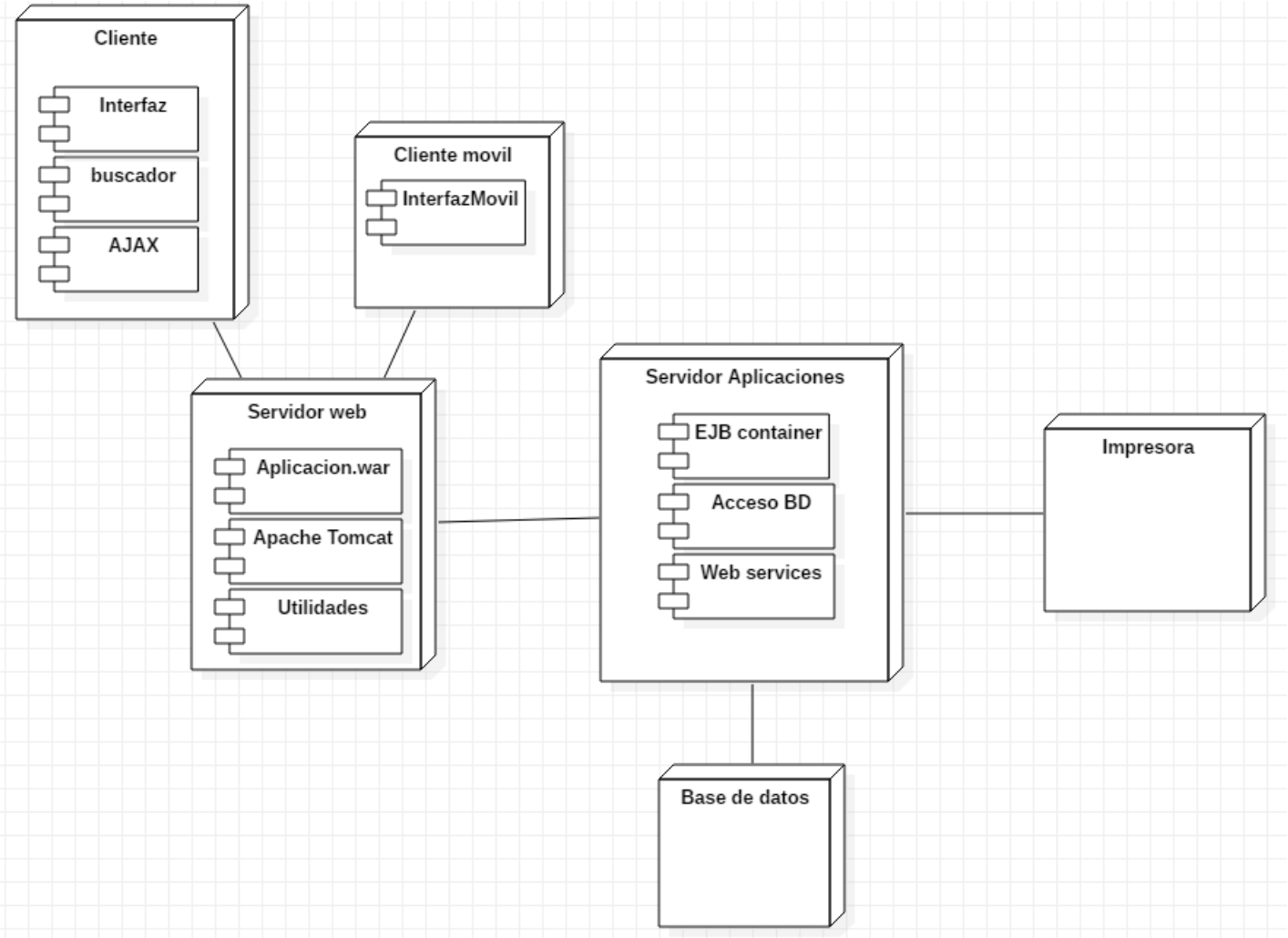
Notación de una conexión



Notación de conexión Cliente-Servidor

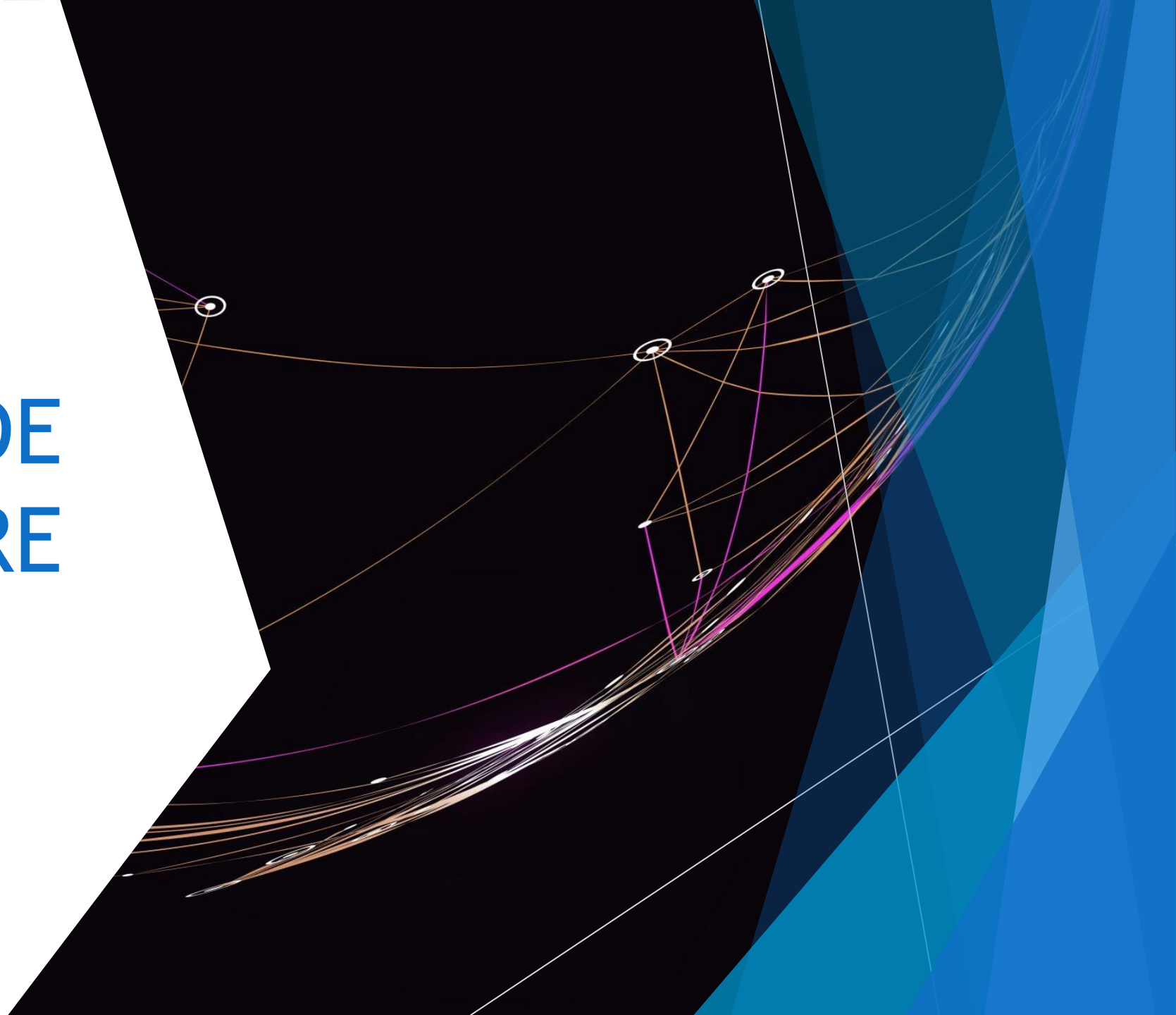
Imágenes obtenidas de: https://diagramasuml.com/paquetes/

Diagrama de Despliegue: Ejemplo



Ejemplo obtenido de: https://diagramasuml.com/paquetes/

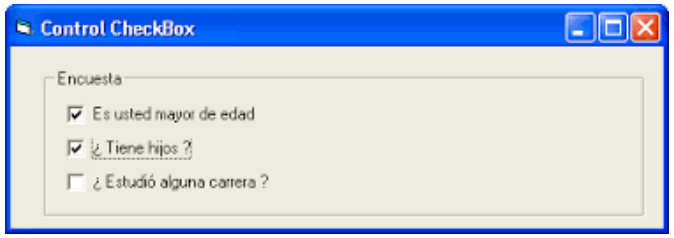
PRUEBAS DE SOFTWARE



Aseguramiento de Calidad: Conceptos

Verificación

En el sentido más general, la verificación es la comprobación de algún evento, proceso, material u otro.



Validación

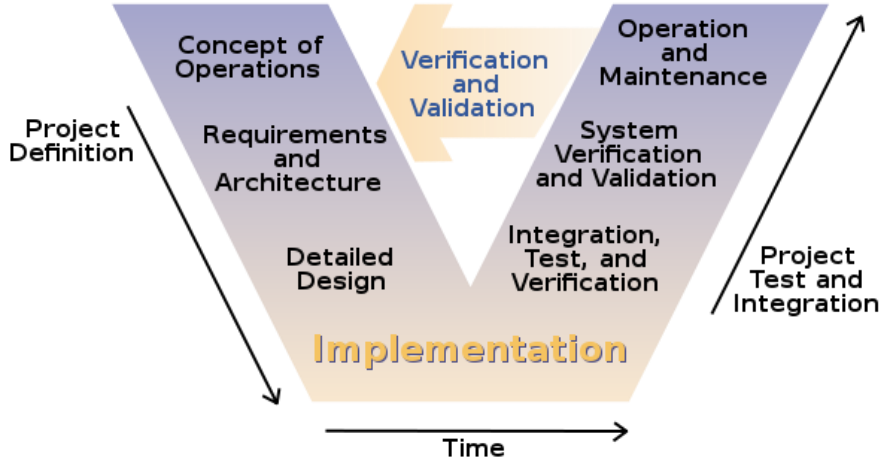
Es la acción de dar fuerza o firmeza a aquello que tiene un peso legal o que es rígido y subsistente

Pruebas de Validación

Las pruebas de validación son el proceso de revisión al que se somete un programa informático para comprobar que cumple con sus especificaciones. El mismo, que suele tener lugar al final de la etapa de desarrollo, se realiza principalmente con la intención de confirmar que la aplicación permita llevar a cabo las tareas que sus potenciales usuarios esperan de ella.

Verificar → ¿Estamos construyendo correctamente el producto?

Validación → ¿Estamos construyendo el producto correcto?



V & V en las etapas de Modelamiento

Nivel de Modelización	Verificación	Validación
Modelo Conceptual		¿Contiene el modelo todos los elementos, sucesos, y relaciones relevantes? ¿Podrá el modelo responder a las cuestiones planteadas?
Modelo Lógico	¿Están los eventos representados correctamente? ¿Son las fórmulas matemáticas y las relaciones correctas? ¿Están las medidas estadísticas formuladas correctamente?	¿Incluye el modelo todos los elementos considerados en el modelo conceptual? ¿Contiene todas las relaciones del modelo conceptual?
Modelo de ordenador/Modelo de Simulación	¿Contiene el código todos los aspectos del modelo lógico? ¿Están las estadísticas y las fórmulas calculadas correctamente? ¿Contiene el modelo errores de codificación?	¿Es el modelo una representación válida del sistema real? ¿Puede el modelo duplicar el comportamiento del sistema real? ¿Es creible el modelo para los expertos del sistema?

Técnicas de Testing como V&V

El Testing consiste en realizar una serie de pruebas controladas sobre el sistema o conjunto de programa. Los dos tipos de prueba más comunes son:

Ascendentes

En estos se comienza por comprobar los módulos básicos y luego se pasa a probar las interrelaciones entre ellos hasta que el modelo ha sido probado como un único sistema.

Descendentes

Las pruebas comienzan con el módulo principal y van bajando hasta los módulos básicos.

Técnicas de Validación

Comparación de salidas entre lo modelado y lo real

- Se podrá aplicar en aquellos casos en los que el sistema exista y se pueda experimentar con él de forma que se obtengan datos de salida del mismo.
- Este método consiste en ejecutar el modelo y obtener una serie de datos de salida y comparar éstos, mediante algún método estadístico, con resultados que se tengan del sistema.

Método de Turing

- Alan Turing sugirió este método como un test de inteligencia artificial.
- En este test, a un experto, o grupo de expertos, se le presentan resúmenes o informes de resultados de ejecución del sistema y del modelo, a los que se les ha dado el mismo formato.
- Estos informes se reparten aleatoriamente a los ingenieros y administradores del sistema, para ver si son capaces de discernir cuáles son los reales del sistema y cuales la imitación resultado de la simulación.
- Si los expertos no son capaces de distinguir entre ambos, se puede concluir que no hay evidencias para considerar inadecuado al modelo. Si descubren diferencias las respuestas sobre lo que encuentran inconsistente se puede utilizar para realizar mejoras en el modelo.

Técnicas de Validación

Método Delphi

- Se utiliza cuando no se tienen datos o se dispone de muy pocos, acerca del sistema que se está considerando.
- En este método, se selecciona un grupo de expertos los cuales deben llegar a un consenso en las respuestas que den acerca de una serie de preguntas que se les plantean. Su proceso consiste en:
  - *Se envía un cuestionario a cada miembro del grupo. Para la validación de un modelo de simulación, las cuestiones podrían tratar sobre las respuestas del sistema real ante ciertas entradas o cambios en su estructura.*
  - *Basándose en las respuestas dadas a las cuestiones planteadas, se elaboran nuevos cuestionarios que van centrándose en temas más específicos.*
  - *Los nuevos cuestionarios se envían al grupo junto con las respuestas obtenidas a las cuestiones en rondas anteriores.*

Estos tres pasos se repiten hasta que el analista consiga de los expertos una predicción de la respuesta de sistema.

Técnicas de Validación

Comportamientos en casos extremos (test de estrés)

- En algunas ocasiones se puede observar el comportamiento del sistema bajo condiciones extremas.
- Esta es una situación ideal para recoger datos de las medidas de ejecución del sistema real de forma que luego se puedan comparar con los resultados de la simulación, una vez que se ejecute el modelo bajo situaciones similares.
- También es posible que los expertos del sistema puedan predecir el comportamiento del sistema bajo condiciones extremas y utilizar estas predicciones para validar el modelo.

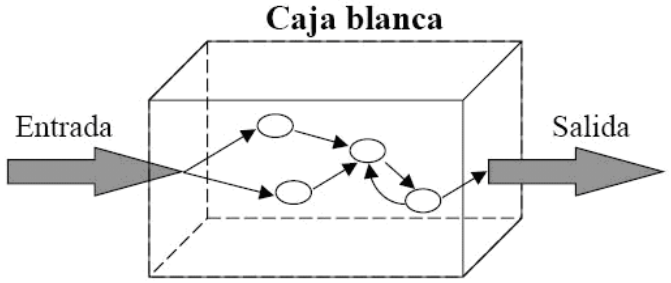
Pruebas de Software

Pruebas unitarias

Es una forma de comprobar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

Para que una prueba unitaria tenga la calidad suficiente se deben cumplir los siguientes requisitos:

- Automatizable → Sin intervención humana (o mínima).
- Completas → Deben cubrir la mayor cantidad de código.
- Repetibles o Reciclables → Las pruebas deben poder ser ejecutadas varias veces.
- Independientes → La ejecución de una prueba no debe afectar a la ejecución de otra.
- Profesionales → Se deben documentar.



Pruebas de Software

Pruebas de integración

- Son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias.
- Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez.
- Sirven para verificar que un gran conjunto de partes de software funcionan juntos.

Pruebas Modulares

- Estas pruebas nos permiten determinar si un modulo del programa esta listo y correctamente terminado.

Pruebas de Aceptación

- Las pruebas de aceptación son pruebas formales que verifican si un sistema satisface los requisitos empresariales.
- Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios.
- Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

Definición de un Plan de Pruebas de SW

El plan de pruebas de software se elabora para atender los objetivos de calidad en un desarrollo de sistemas, encargándose de definir aspectos como por ejemplo los módulos o funcionalidades sujeto de verificación, tipos de pruebas, entornos, recursos asignados, entre otros aspectos.

PASO 1: Analizar los requerimientos de desarrollo de software

- Para elaborar un plan de pruebas de software lo primero es comprender los requerimientos de usuario que componen la iteración (o sistema), que son el sujeto de la verificación de calidad que se va a realizar.
- Se debe analizar toda la información de la ingeniería de requisitos, incluyendo la matriz de trazabilidad (más detalle en **este enlace**), especificaciones y diseño funcional, requisitos no funcionales, casos de uso, historias de usuario (si estás trabajando con metodologías ágiles), entre otra documentación.
- También es muy importante realizar entrevistas con el equipo encargado de la ingeniería de requisitos para aclarar dudas y ampliar la información que sea necesaria.

Definición de un Plan de Pruebas de SW

PASO 2: Identificar las funcionalidades nuevas a probar

- A partir de la documentación del análisis de requisitos y de las entrevistas con el equipo de ingeniería de requisito y desarrollo, debes identificar e incluir en el plan de pruebas de software en la lista de las funcionalidades (si el sistema es nuevo, todo es nuevo)
- En el caso de desarrollos de software integrados a un sistema existente es necesario revisar las funcionalidades que forman parte del desarrollo de software, en todas las capas de la arquitectura.

PASO 3: Identificar las funcionalidades de sistemas existentes que se deben probar

- Se debe identificar las funcionalidades existentes que estén siendo impactadas por el desarrollo de alguna forma, considerando todos los componentes afectados en todas las capas de la arquitectura de software.

Definición de un Plan de Pruebas de SW

Existen dos situaciones que se puede encontrar al identificar estas funcionalidades:

- Funcionalidades modificadas de cara al usuario

Por ejemplo, si una funcionalidad está siendo modificada agregando más pantallas o cambios a su flujo de proceso, debe ser incluida en el plan de pruebas de software.

- Funcionalidades modificadas en sus componentes internos

Son funcionalidades no modificadas de cara al usuario, manteniendo la misma interfaz gráfica y flujo de procesos, sin embargo, si se modifican componentes internos que comparten con otras funcionalidades del sistema, en las capas de lógica de negocio o acceso a datos. Estas se deben incluir en el plan de pruebas de software para determinar a partir de ellas pruebas de regresión a realizar.

Definición de un Plan de Pruebas de SW

PASO 4: Definir la estrategia de pruebas

- Consiste básicamente en seleccionar cuáles son los tipos de pruebas de software que se deben realizar.

PASO 5: Definir criterios de inicio, aceptación y suspensión de pruebas

Criterios de aceptación o rechazo

- Para definir los criterios de aceptación o rechazo, es necesario definir el nivel de tolerancia a fallos de calidad.
- Si la tolerancia a fallos es muy baja se puede definir como criterio de aceptación que el 100% de los casos de prueba estén sin incidencias.
- Lograr este margen en todos los casos de prueba principales y casos bordes será muy difícil, y podría comprometer los plazos del proyecto (incrementa los riesgos), pero asegura la calidad del producto.
- Por otra parte, puede ser que la intención sea realizar un Soft Launch, o un mínimo producto viable, en ese caso se podría definir como criterio de aceptación el 100% de los casos de prueba principales (considerados clave) y 20% de casos de prueba no principales (casos borde).

Una vez logradas las condiciones, se darán por aceptadas las pruebas y el desarrollo de software.

Definición de un Plan de Pruebas de SW

Criterios de inicio o reanudación

- Definen las condiciones que se deben cumplir para dar inicio o reanudar las pruebas.
- Por ejemplo, en el caso de inicio la condición podría ser la instalación de los componentes de software en el ambiente y que los casos de pruebas de verificación de ambiente sean exitosos.
- Para el caso de la reanudación las condiciones están relacionadas, se determina a partir de cuales criterios de suspensión se presentaron para detener las pruebas. Una vez que estás condiciones ya no existan (sean solventadas) se procede con la reanudación.

Criterios de suspensión

- Las condiciones van a depender de los acuerdos de nivel de servicio (SLA) internos de la organización y también de los acuerdos establecidos en cada proyecto individual.
- Por ejemplo, si se tiene un equipo de pruebas que comparte su esfuerzo entre varios proyectos, se puede definir un criterio de suspensión exigente, un determinado porcentaje de casos fallidos que resulten en incidencias. Si la condición se cumple, se detienen las pruebas y se dedica el personal a otras actividades.
- Por otra parte si se tiene un equipo de pruebas con personal dedicado, el criterio de suspensión puede ser poco exigente, por ejemplo solo ocurriendo si se bloquean por incidencia todos los casos de prueba.

Cierre de la sesión