**REQUERIMIENTOS**

"la Ing. del software me provee las herramientas para q yo pueda hacer un software de calidad"

la Ing. del soft Es la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de software.

Stakeholders es alguien q se ve afectado por el software

Universo de información (universo de discurso o dominio de aplicación): es el dominio del problema, toda la información el vocabulario, el proceso de organización…. Toda la información q se utiliza para realizar el software

**Clasificación de requerimientos:**

**Requerimientos Funcionales (RF):** describen la funcionalidad o los servicios que se espera que el sistema proveerá. ◦ Cuando se expresan como requerimientos del usuario, se describen de forma general; mientras que los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etc.

**Requerimientos No Funcionales (RNF):** ◦ se refieren a las propiedades emergentes del sistema. ◦ definen las restricciones del sistema: capacidad de los dispositivos de entrada/salida, representación de datos que se utiliza en las interfaces, etc. Calidad: confiabilidad, disponibilidad, robustez... Factores humanos: facilidad de uso, simplicidad de las interfaces, Características de rendimiento: tiempos de respuesta, performance, ... Restricciones de hardware y software: compatibilidad con equipamiento y/o sistemas disponibles, ... Cambios y/o adaptaciones a nuevos requerimientos: adaptabilidad, reusó de componentes, ... Restricciones de seguridad.

**Requerimientos Inversos (RI):** definen cómo el software nunca se debe comportar.

Los requerimientos no funcionales se refieren al sistema como un todo más que a rasgos particulares del mismo; a menudo son más críticos que los requerimientos funcionales particulares. } Mientras que el incumplimiento de un requerimiento funcional degradará al sistema, una falla en un requerimiento no funcional del sistema lo inutiliza.

**CASOS DE USO**

**CASO DE USO PRIMARIO:** son ejecutados de forma directa o especifica por un actor. Son ejecutados en forma directa por un actor. Modelan el problema central. Representan alguna función fundamental del sistema  
**CASO DE USO SECUNDARIO:** Son invocados o ejecutados por otro caso de uso. Surgen luego de una “explotación” de los casos de uso primarios. Tienen mayor nivel de detalle mediante relaciones de “inclusión” o “extensión”. Representan alguna variación respecto de uno primario.

**CU TEMPORALES:** el tiempo dispara estos CU.

**COMPONENTE DE LOS CU:**

Límite del sistema: Define el alcance del sistema. Nombre.

Actores: Roles desempeñados por personas o elementos que utilizan el sistema. Se representan afuera del límite del sistema.

Casos de Uso: Describen interacciones de los actores con el sistema. Es una operación/tarea específica que se realiza tras una orden de algún agente externo

Relaciones: relaciones significativas entre actores y casos de uso, o entre casos de uso. Representada por una línea entre los casos de uso y/o actores relacionados

**Diferencia entre actor y usuario:**

Un actor es el rol o papel que juega un objeto externo en su relación con el sistema. Un usuario es una persona que, cuando usa el sistema, asume un rol. Un usuario puede acceder al sistema como distintos actores.

**INCLUDE**

**Relación de inclusión <>:** ◦ Los casos de uso pueden contener la funcionalidad de otro caso de uso como parte de su proceso normal. ◦ Los casos de uso pueden ser incluidos por uno o más casos de uso. ◦ Un caso de uso incluido se llamará cada vez que se ejecute una ruta básica. ◦ Ayuda a reducir el nivel de duplicación de la funcionalidad realizando un factoreo del comportamiento común en casos de uso que se usan muchas veces. La flecha va desde el CU que incluye hacia el CU incluido

**Relación de extensión <>:** ◦ Agrega nuevo comportamiento a un CU existente ◦ Refleja situaciones particulares en un CU que pueden ser tratadas (extendidas) por otro. ◦ En la descripción del caso de uso que es extendido debe haber una forma de indicar en que punto entra en juego el caso de uso que lo extiende (punto de extensión). La flecha va desde el CU que extiende al extendido

**Relación de generalización o herencia:** el actor hereda características del origen

**HISTORIA DE USUARIO**

**Métodos agiles:** Conjunto de métodos en el que las necesidades y soluciones evolucionan a través de la colaboración entre equipos multifunción y autoorganizados.

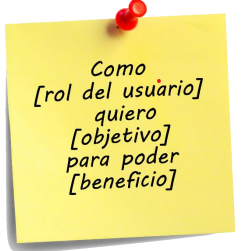
**Una historia de usuario (HU)** describe una funcionalidad que será valiosa para un usuario o cliente de un sistema o software.

La HU busca cambiar el enfoque de “escribir” a “fomentar la conversación” sobre requerimientos y características. }Se debe describir el rol, la funcionalidad y el resultado de la aplicación en una frase corta. } Se incorporan criterios de aceptación, no más de 4 por historia, incluyendo contexto, el evento y el comportamiento esperado frente a ese evento.

**Se componen de tres aspectos:**

Una descripción escrita de la historia: se utiliza para la planificación y como recordatorio

Conversaciones sobre la historia: Sirven para concretar los detalles de la misma

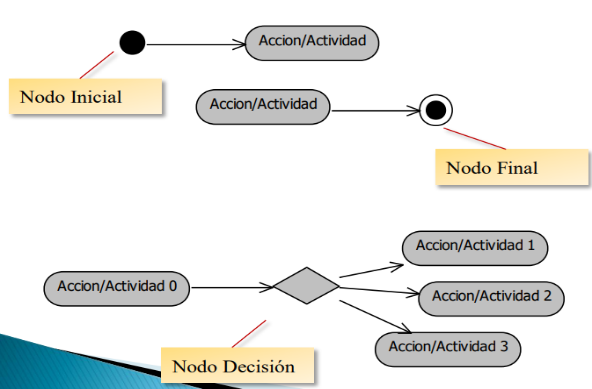
Pruebas que transmiten y documentan los detalles: Pueden utilizarse para determinar cuándo se ha completado una historia

**PUEDEN SER INCOMPLETAS.**

EJ: **Como** cliente del banco   
**Quiero** retirar dinero de mi cuenta  
**Para** hacer compras con efectivo

Un **«release plan» o plan de proyecto** es un conjunto de historias de usuario (normalmente épicas) agrupadas por «releases» o versiones del producto que se ponen a los disposición de usuarios incrementando el valor para estos respecto de la anterior

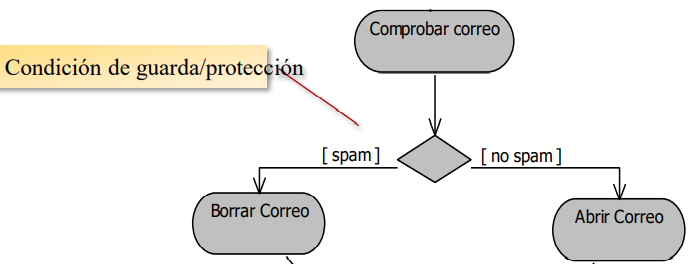
**DIAGRAMA DE ACTIVIDADES**

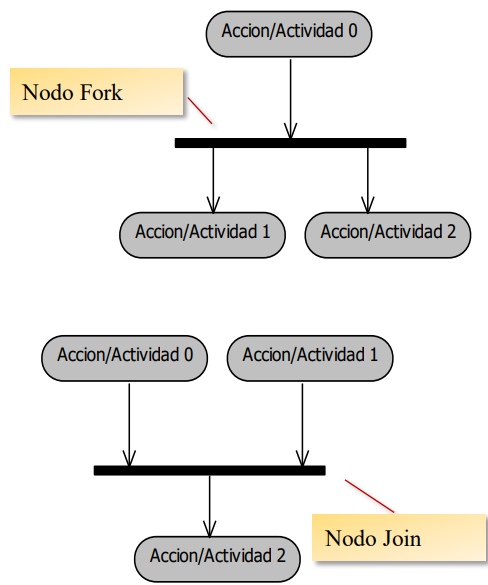


**Nodo inicial:** Indica dónde empieza el flujo cuando se invoca una actividad

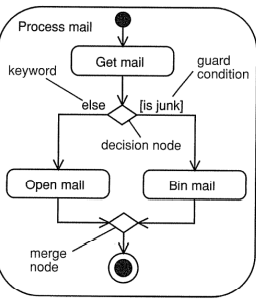
**Nodo final:** Termina una actividad

**Nodo decisión:** Posee un extremo de entrada y dos o más de salida. Un token que llega al extremo de entrada se ofrecerá a todos los extremos de salida, atravesando uno. Cada uno de los extremos condición de protección (guarda), que debe evaluar a verdadera.



**Nodo fork:** Divide el flujo en múltiples flujos concurrentes ◦ Tiene un extremo de entrada y dos o más de salida paralelos

**Nodo Join:** Múltiples extremos entrantes y uno solo de salida ◦ Sincroniza flujos

**Nodo Merge:** Posee dos o más arcos de entrada y un único arco de salida. Fusionan los flujos de entrada en un flujo de salida simple