Metodologías de trabajo

Módulo 3. Desarrollo ágil de software.

Sesión 6. De requerimientos a User Stories

Requerimientos del sistema

Un requisito o requerimiento es un documento con una especificsción de una condición (o capacidad) de un software que necesita un usuario para resolver un problema o lograr un objetivo.

Requerimientos funcionales

Describen la funcionalidad o los servicios que se espera que el producto provea.

- Requerimientos de producto

Especifican el comportamiento del producto. Incluye lo referente a rendimiento del sistema y fiabilidad (tasa de fallos aceptable). (??)

Requerimientos organizacionales

Se derivan de políticas y procedimientos existentes en la organización del cliente y en la desarrollador.

Requerimientos no funcionales

Describen la fiabilidad, respuesta en el tiempo y capacidad de procesamiento que se espera del producto.

De requerimientos a User stories

Es la representación de un requerimiento por parte del cliente o usuario. Describe el comportamiento esperado del producto desde el punto de vista del usuario, y tiene como ventaja que divide una gran cantidad de funcionalidades en partes más pequeñas para facilitar la planificación e implementación. Se centran el valor que se obtiene al usar el producto.

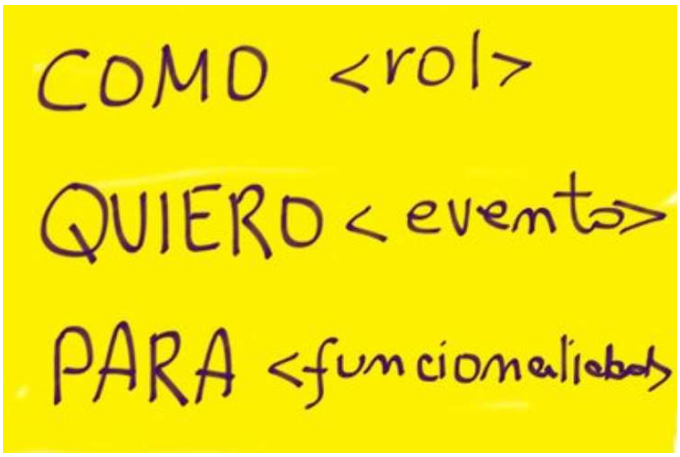
Las historias de usuario son cortas y transmiten la esencia principal y las características más importantes. Deben transmitirse de forma oral. Las escribe el Product Owner y son accesibles para el cliente, el equipo de desarrollo y todos los stakeholders. Por tanto, no deben estar escritas en lenguaje demasiado técnico.

Las User stories deben ser (modelo **Invest**):

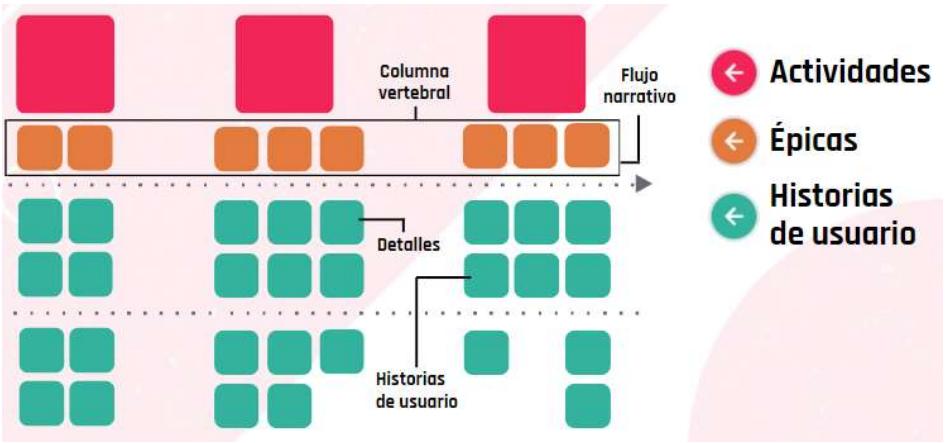
- Independientes. Deberían poder completarse en cualquier orden.

- Negociables
- Valiosas
- Estimables
- Pequeñas. Muchas historias en una iteración.
- Testeables

Con las historias de Usuario se genera el Product Backlog, incluyendo únicamente las descripciones de alto nivel del requerimiento.



User story mapping



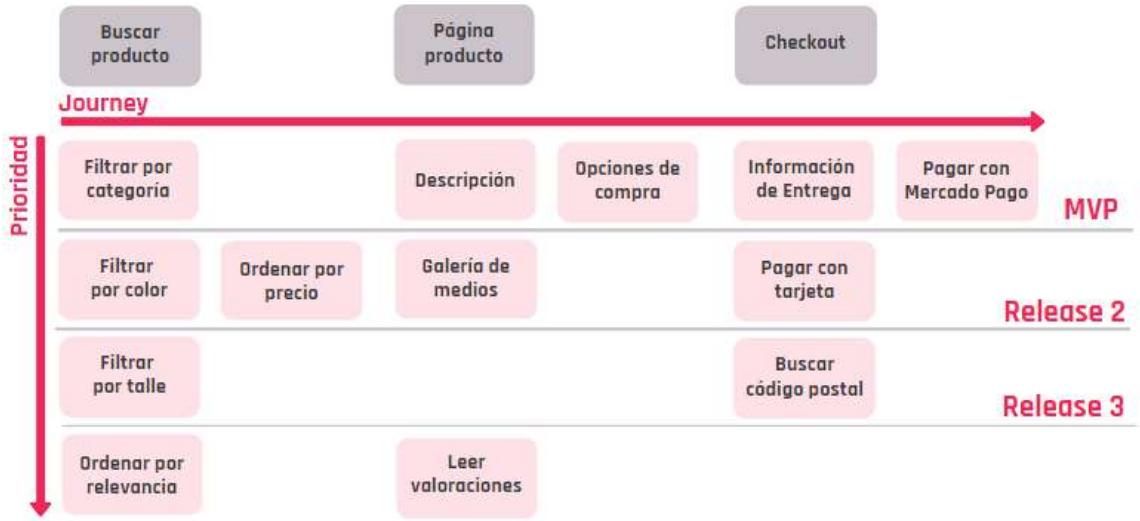
Descubrimiento de Producto:
- Necesidad del cliente
- Producto
- Funcionalidades
- Caracterísitcas
- Servicios

(Las últimas tres son definidas en el User Story Map)

Beneficios del User Story Map:
- Clarificar el scope y features
- Priorizar los componentes por valor de negocio
- Detectar dependencias entre componentes

- Identificar entregas o releases
- Ayudar a delimitar el alcance

Armar el User Story Map:
1. Identificar los procesos del negocio
2. Identificar las funcionalidades
3. Definir y detallar las funcionalidades
4. Identificar MVP y roadmap



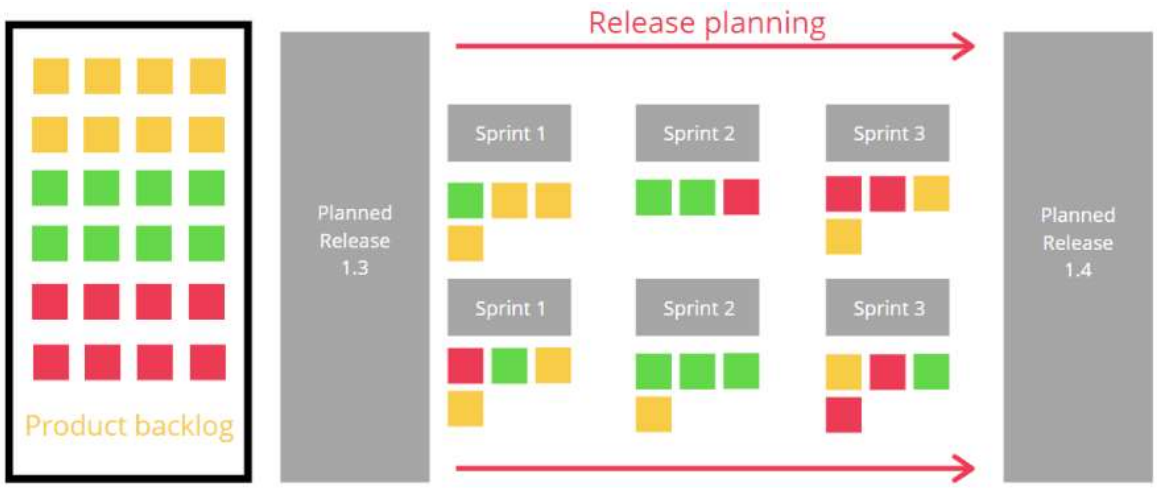
User Story Mapping y release plan

¿Qué es un Product Road Map?
Es un documento estratégico que permite establecer la dirección del desarrollo.

- Indica la visión y la dirección del producto.
- Ayuda a comprender cómo podría evolucionar el prodcuto.

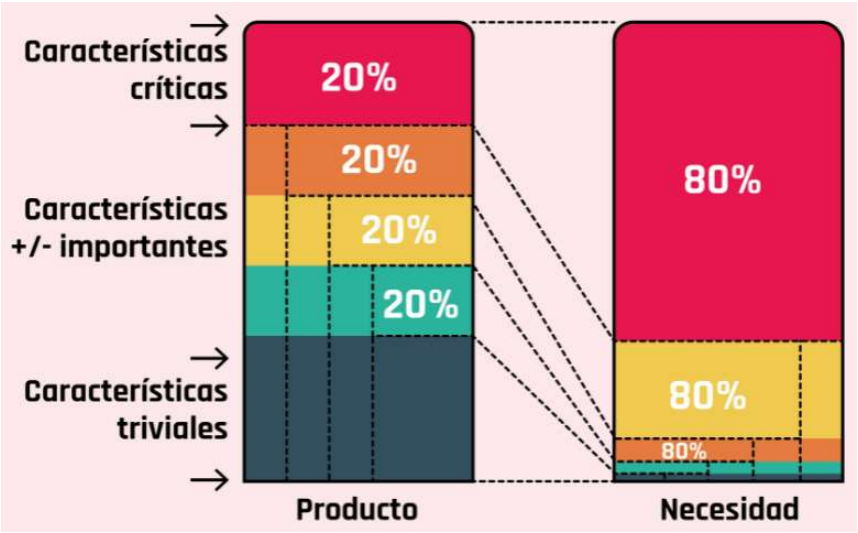
¿Qué es un release plan?
Un release plan es un conjunto de historias de usuario (normalmente épicas) agrupadas por versiones del producto —o releases— que se ponen a disposición de los usuarios incrementando el valor para estos respecto de la anterior.

El product roadmap comunica la descripción general de alto nivel de la estrategia de un producto, mientras que un release plan es un documento táctico diseñado para capturar y rastrear las funciones planificadas para un lanzamiento próximo.



Priorización del backlog
El backlog es una lista priorizada y posiblemente estimada de las user stories del proyecto. Es básicamente un listado de ítems (PBIs) o características del producto a construir, mantenido y priorizado por el product owner.

Principio de Pareto
El 20% del esfuerzo produce el 80% de los resultados.



Técnicas de priorización de User Stories

- MoSCoW
- Puntos de valor de negocio
- Matriz de priorización

Refinamiento de historias
Acotar el alcance, eliminar ambigüedades, reducir el riesgo y simplificar las historias de usuario para entregar la mayor cantidad de valor, de forma independiente, en el menor tiempo posible.

Cartas de Slicing
Slicing o Splitting es una práctica que permite reducir la complejidad y esfuerzo requerido de las Historias de Usuario, y que se aplica a todos los ítems sobre los cuales trabaja el equipo.

Sincrónico

¿Cuál es el problema de la documentación?

- No maneja bien los cambios.
- El software se construye en base a la especificación en lugar de en base a lo que el cliente quiere.
- Toman supuestos falsos.
- Lleva mucho tiempo.

¿Qué prácticas proponen soluciones a este problema?

Prácticas ágiles de desarrollo: Scrum & Peer Programming

- Generar la documentación necesaria y justo a tiempo
- Refactorizar el código constantemente, manteniendo siempre la calidad alta
- Para el ingreso de nuevos desarrolladores, la documentación a consultar está entre líneas
- Incentivar comunicación cara a cara para evacuar dudas.

User Stories:
- Es una "promesa d euna conversación"
- Tiene un sentido de negocio (no técnicas)
- Abarcan todos los componentes del sistema.

Beneficios:
- Permiten planificar e implementar más ágilmente.
- Permiten alinear el negocio con el equipo de desarrollo.
- Permiten ampliar el abanico de soluciones.

Épica: Una historia de usuario enorme.

Ejercitación:

Actividad 1

User stories

Definir al menos tres stories según la siguiente visión de producto:

"Construir un sitio web que permita a los clientes personalizar zapatos y comprar pares de zapatos únicos."

10 minutos de trabajo en equipo.
5 minutos de puesta en común.

Técnicas de priorización

- Bussines Value Board: Puntuar por valor de negocio y esfuerzo
- MoSCoW
- Matriz de priorización

User Story mapping

…

Responder

- ¿Cuántas épicas y user stories me quedaron?
- ¿Cuántas user stories por versiones tengo?
- ¿Cuántas incluye mi MVP?
- ¿Cuántas user stories puedo hacer por semana?
- ¿Cuánto va a tardar el primer release?

Sesión 7. Métricas Agile

Julio 13 de 2021

Estimación y Capacity

Estimar: Pronosticar el valor de algo basándose en supuestos, y teniendo en cuenta un nivel de incertidumbre sobre variables que exceden nuestro control. Durante la Reunión de Planificación del Scrum se estiman las Historias de Usuario, mediante Puntos de Historia que se basan en la Sucesión de Fibonacci.

En caso de que haya errores en la planificación, se hace una Reunión de Refinamiento.

Técnicas de Estimación

T-Shirt Sizes.
Asignar tallas de camiseta a las tareas del Product Backlog.

Bucket System
Utiliza una secuencia numérica. Los números de la secuencia se disponen de forma ordenada como si fueran cubos o canastos. A continuación, cada elemento del Producto Backlog se deposita en un cubo.

Dot voting
Cada miembro del equipo tiene un número de etiquetas o fichas que puede repartir entre los elementos del Product Backlog.

Planning Poker

1. Preparar los materiales:
Cada participante debe tener una baraja de Planning Poker (un juego de tarjetas marcadas con números así:

- 1/2: Tareas muy pequeñas
- 1, 2, 3: Tareas pequeñas
- 5, 6, 13: Tareas medianas
- 20, 40: Tareas grandes
- 100: Tareas muy grandes
- ?: Tareas inestimables
- (Infinito): Tareas enormes
- Hora de una pausa

2. ¿En qué unidades estimar?
Definir si se estima en semanas ideales (e, para épicas) o en días ideales.

3. Los números se basan en la Sucesión de Fibonacci y representan el esfuerzo requerido para llevar a cabo la tarea. El cliente explica un requerimiento y los miembros del equipo asignan una tarjeta a esa tarea. En caso de desacuerdos granes, se puede discutir y repetir la asignación de tarjetas.

Métricas en Kanban

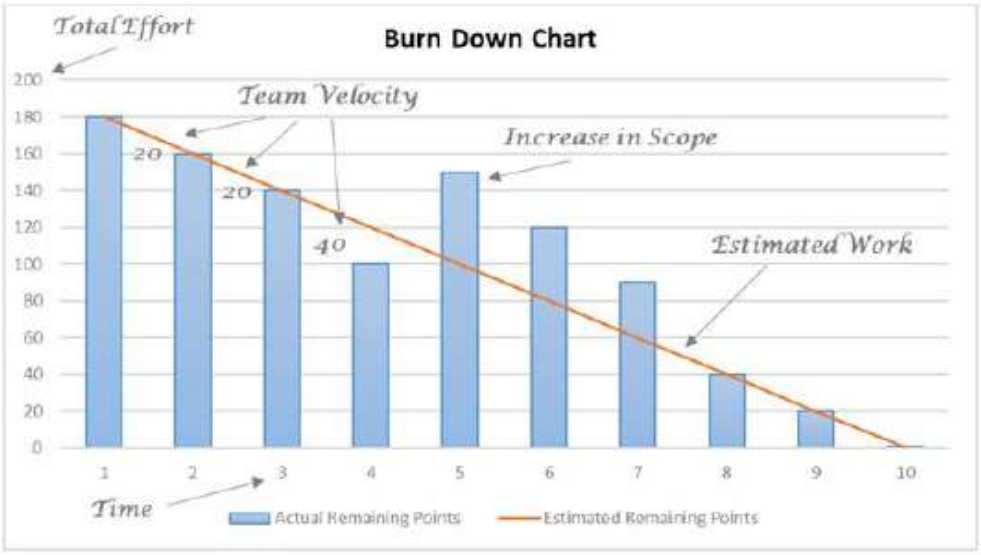
Lead time: Tiempo de entrega y tiempo de ciclo
Tiempo de entrega (Lead time): Cantidad de entrega desde que se solicita un elemento hasta que se entrega.
Tiempo de ciclo (Cycle time): Periodo durante el cual se trabaja activamente en un elemento.
Para medirlos se puede utilizar un Diagrama de Flujo Acumulativo (Cumulative Flow Diagram, CFD)

Trabajo en curso
Esta métrica permite supervisar y analizar la Capacidad del flujo de trabajo. Se puede utilizar un Gráfico de Trabajo en Curso Envejecido.

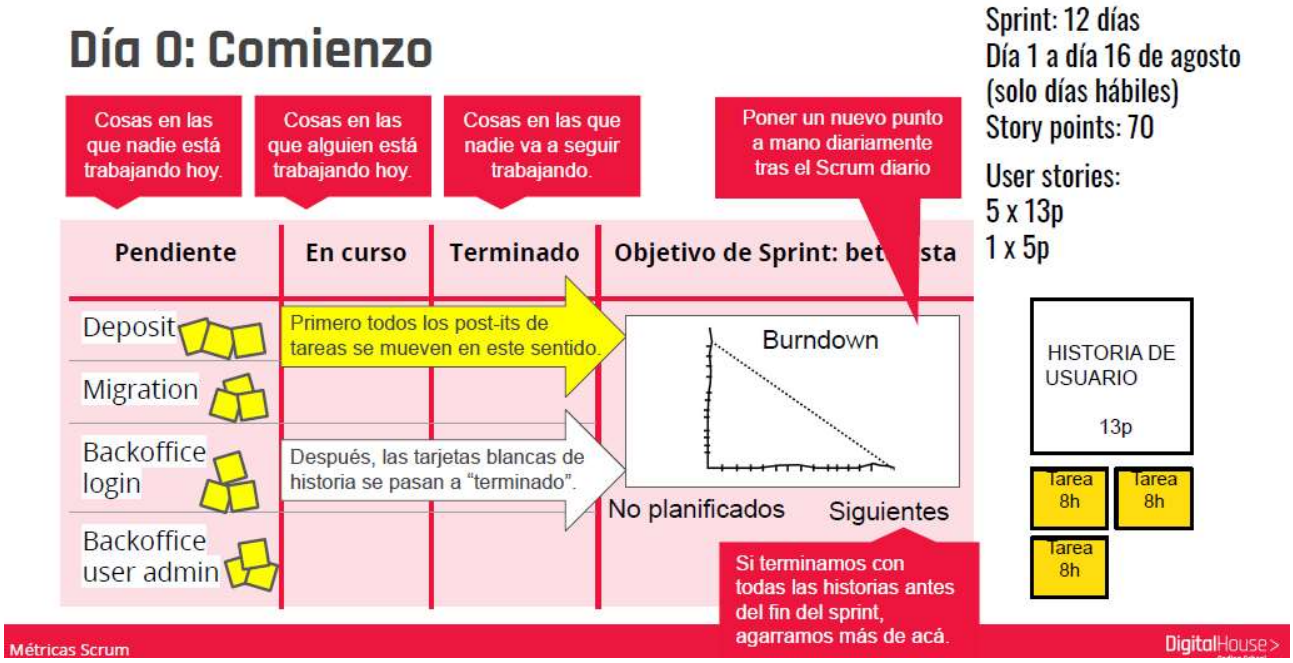
Throughput: Rendimiento
Número de tareas finalizadas por unidad de tiempo. El Rendimiento es una métrica de Productividad del Equipo. Se puede utilizar un Histograma de Rendimiento.

Métricas Scrum

Burn Down Chart (Cuenta Regresiva)
Muestra los User Story Point pendientes por realizar en función de los días del sprint.



Velocity
Mide la velocidad del equipo. Cantidad de trabajo que un equipo puede completar satisfactoriamente en un sprint.



Métricas: Estimación vs Realidad

Permiten medir:
- Velocidad del equipo
- Esfuerzo
- Estatus
- Avance por unidad de tiempo

Velocity Chart
¿Cuántos puntos de historia logró terminar el equipo durante el sprint?

BurnDown Chart
Muestra la cantidad de Story Points que se van resolviendo en el sprint en curso. Es una herramienta de actualización diaria.

BurnUp Chart
Muestra en el eje y la cantidad total de Story Points del proyecto, y en el eje x los sprints o iteraciones.

Scrum y OKR

Resultados clave: Son los principales hitos o métricas que buscamos alcanzar para lograr el objetivo. Se mantienen en la parte superior del tablero y se inicia con ellos cada reunión.

Cada tarea del Sprint Backlog pertenece a un objetivo clave.

Sincrónico

El planning Poker lo realiza el equipo de desarrollo. El Scrum Master y el Product Owner no puntúan.
Las cartas permanecen ocultas hasta la votción, con el fin de eliminar el sesgo y la dependencia.

https://www.planitpoker.com/authentication/

Mural:
https://app.mural.co/t/lopaworkspace7627/m/lopaworkspace7627/1625600224567/d45c7d8f947d1862eedb5003815c1476afd0db71?sender=b569d3a3-50e5-4f6d-83d9-55d451c545be

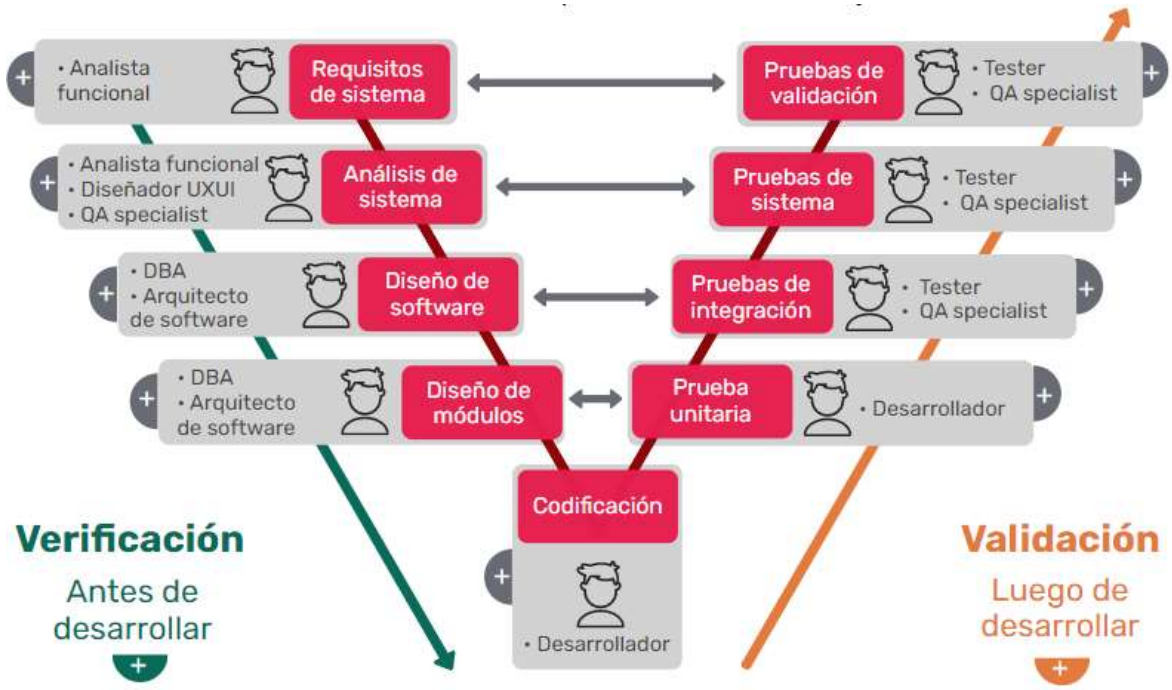
Sesión 8. Arquitectura ágil

Julio 20 de 2021

Ciclo de vida de desarrollo del Software

- Comunicación
- Recolección de solicitudes. Los requisitos se contemplan y agrupan en requisitos del usuario, funcionales y del sistema.
- Estudio de viabilidad
- Análisis del sistema
- Diseño de software
- Codificación
- Pruebas
- Integración. Integración del software con las entidades del mundo exterior.
- Implementación. El software se evalúa por su adaptabilidad y su portabilidad.
- Mantenimiento y funcionamiento.
- Disposición

Modelo en V



Ciclo de vida en relación con los roles (Modelo en V):
Requisitos de sistema – Análisis de sistema – Diseño de software – Diseño de módulos – Codificación – Prueba unitaria – Pruebas de integración – Pruebas de sistema – Pruebas de validación

Roles

Analista Funcional - Bussiness Analyst
- Planifica el ciclo óptimo de desarrollo del software
- Prepara y mantiene la documentación
- Traduce los requisitos en especificación de software

Diseñador UX/UI – UX/UI Designer
- Garantiza que el producto resuelva el problema del usuario final

Especialista en Control de Calidad – QA Specialist
- Se encarga de la calidad en los procesos de desarrollo de software

Administrador de Base de Datos – DBA – Database administrator
- Manejo, mantenimiento, desempeño y confiabilidad de la base de datos.

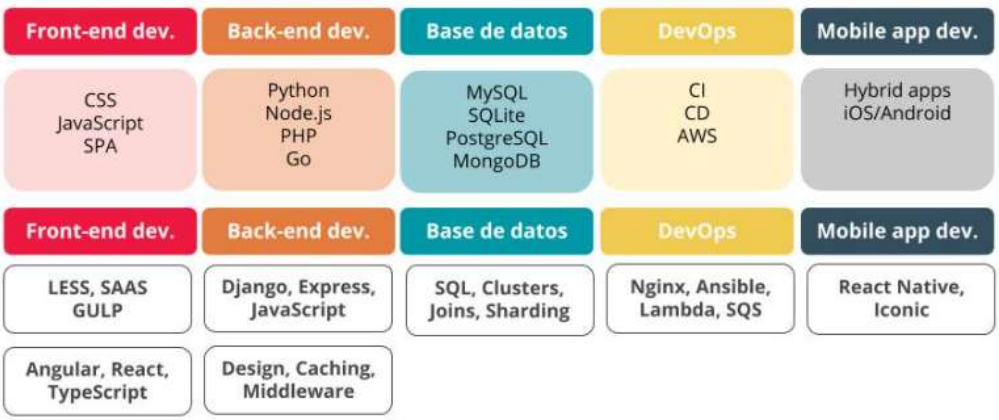
Arquitecto de software – Líder técnico
- Define el mejor modelo de estructura arquitectónica del software para el proyecto. Son responsables de todas las decisiones de diseño de alto nivel.

Desarrollador – Developer
- Traduce los requisitos del software a un lenguaje de programación

Tester
- Comprueba que el producto se está desarrollando correctamente.

Director de Proyecto – Project Manager
- Responsable de organizar y dirigir el equipo. Asigna tareas, realiza el seguimiento de las mismas, elimina obstáculos, gestiona riesgos, facilita las reuniones y gestiona la comunicación dentro del equipo y con la alta dirección.

Tech stak
= Stack de soluciones = Ecosistema de datos. lista de todos los servicios tecnológicos utilizados para construir y ejecutar una sola aplicación. Ej LAMP (Linux + Apache + MySQL + PHP).



Arquitectura de la solución

Arquitectura de sofware es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación. Es un grupo de principios y restricciones sobre cómo las soluciones de un sistema de software deben ser construidas dentro de un ámbito dado.

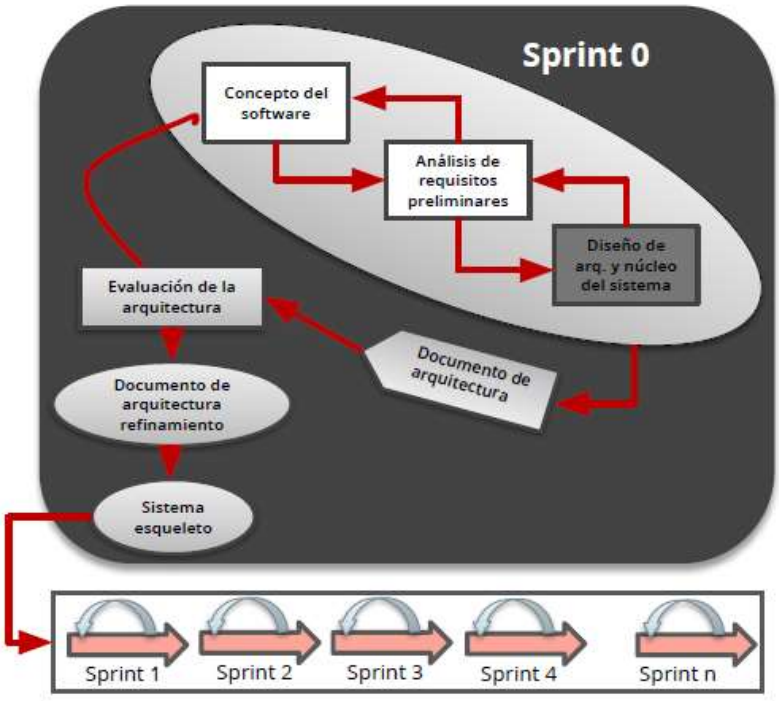
- Define la forma de trabajo
- Define construcción de módulos
- Deja intuir el tipo de aplicación

La arquitectura de software refleja:
- La estructura de módulos y carpetas, sus responsabilidades e interacciones.
- El control y flujo de datos.
- Los protocolos de interacción y comunicación.
- Ubicación en el hardware y comportamiento.
==> Validar la coherencia antes de empezar a construir.

Atributos de calidad:

Requisitos no funcionales:
- Alta disponibilidad
- Renidmiento
- Modificable
- Testeable
- Seguridad

Arquitectura de software y Equipos ágiles…



Diagramas UML

Diagrama de componentes:
Componentes, interfaces y relaciones.
Los componentes pueden ser: frameworks, librerías, APIs.

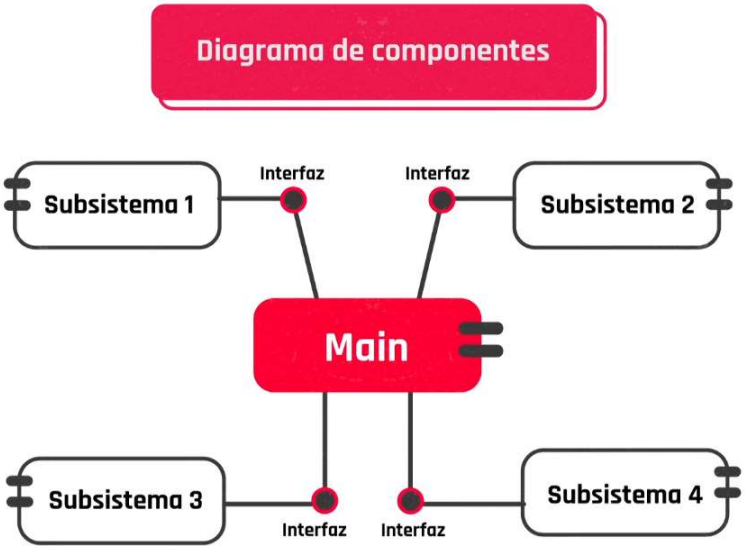
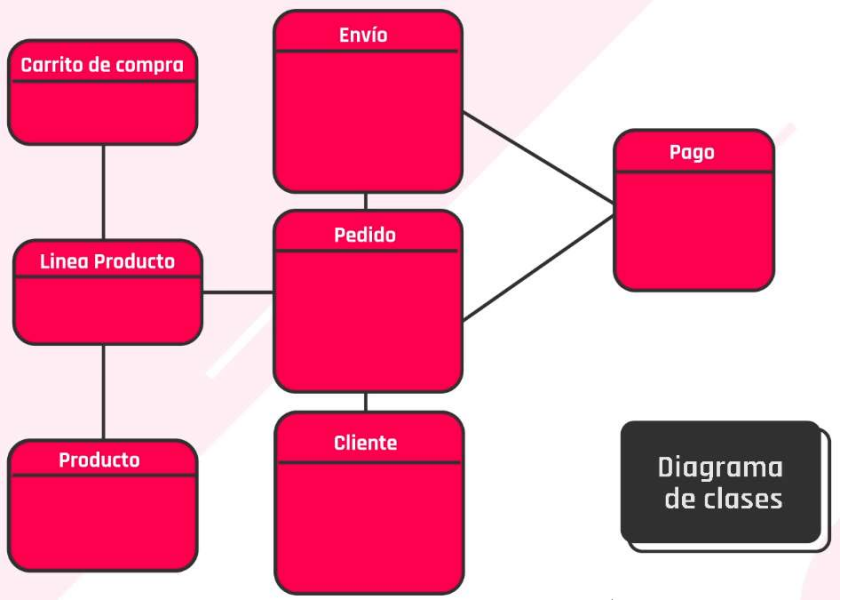


Diagrama de clases:
Objetos, atributos y métodos.
Muestra los elementos que componen el sistema en el modelo de programación orientada a objetos.



Ejemplo:

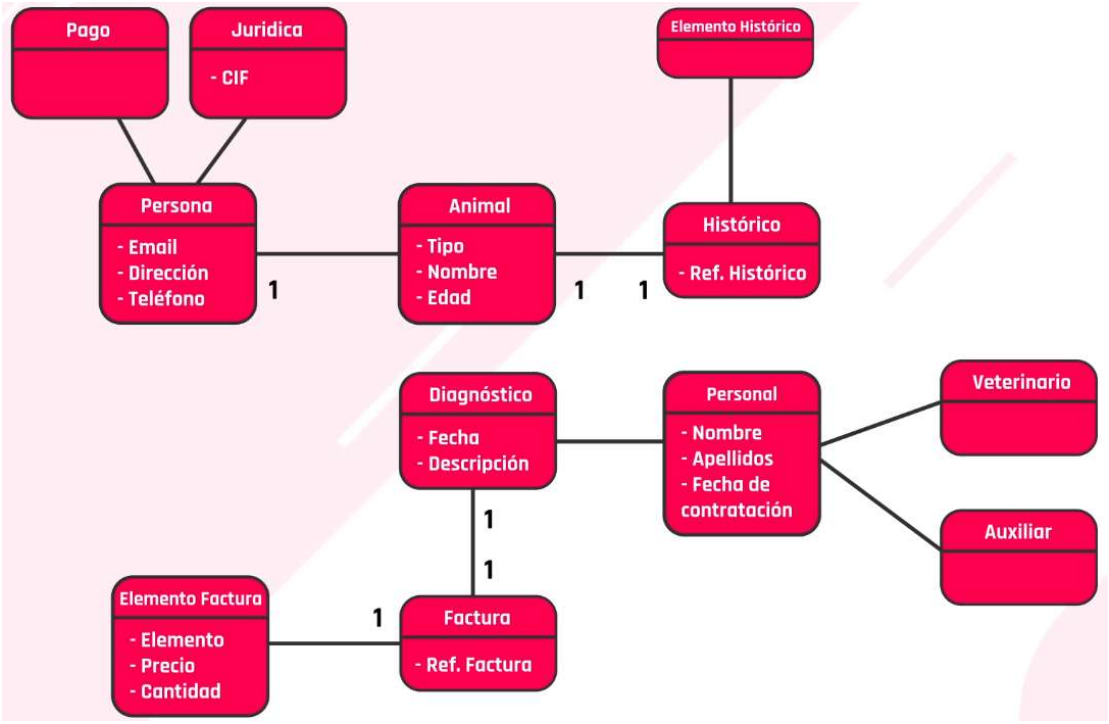
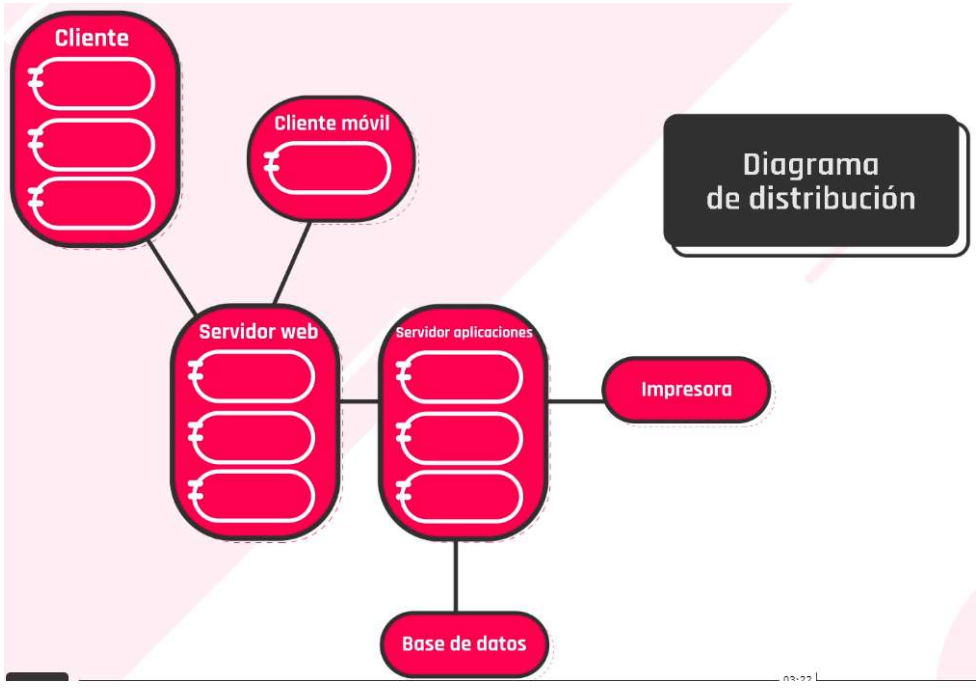


Diagrama de distribución o despliegue:
Representa la distribución física o estática de los componentes software en los distintos nodos físicos de la red hardware.



Arquitectura por capas
- Cada capa oculta las capas inferiores de las siguientes superiores

Beneficios:
- Se entiende una capa como una unidad, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas
- Minimiza las dependencias entre capas.

- Posibilita la estandarización de servicios.
- Las capas pueden ser reutilizadas por varios servicios de mayor nivel.
- Facilita la escalabilidad y el mantenimiento de un sistema complejo

Introducción al modelado de Software

UML: Lenguaje de modelado unificado.
Fue creado para forjar un lenguaje de modelado visual común, semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. Es una herramienta que permite a quienes crean los sistemas generar diseños que capturen ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas.

Diagramas de comportamiento (dinámicos):
- Diagramas de actividades / flujo
- Diagrama de estado
- Diagrama de casos de uso

Diagramas de estructuras:
- Diagrama de componentes
- Diagrama de clases/objetos
- Diagrama de distribución

Elementos de los diagramas UML:
- Diagramas
- Elementos
- Relaciones
- Reglas

Herramientas para hacer diagramas:

- PlantUML
http://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000

- Miró:
https://miro.com/app/dashboard/

- Visio:
https://www.microsoft.com/es-co/microsoft-365/visio/flowchart-software

- Diagrams:
https://app.diagrams.net/

- Figma:
https://www.figma.com/files/recent?fuid=918309065052543480

Sincrónico

UML no se utiliza hoy de forma tan ‘’burocrática’ como antes.
Hoy se usan más Historias de usuario que Casos de Uso.

En cambio, sí se utilizan los diagramas de estado.

Calidad

Inyectar calidad desde el momento cero.
Las metodologías permiten gestionar la calidad en el desarrollo del producto de software.

- Evitar costos de corrección y de retrabajo
- Mejoran satisfacción del cliente
- Controlan riesgos
- Reducen mantenimiento

¿Cómo aseguramos la calidad?
“Primero hacer las cosas bien, luego hacerlas rápido”

- Usando metodologías probadas
- Asegurando alto nivel de consenso
- Haciendo revisiones cruzadas en etapas tempranas
- Priorizando la previsión más que la corrección
- Midiendo la calidad

DoD: Definition of Done
DoR: Definition of Ready

- El tester puede comenzar a plantear sus casos de prueba desde las User Stories.

Mural:
https://app.mural.co/t/lopaworkspace7627/m/lopaworkspace7627/1626795772742/265d4f4b9915d30e7e95363c34de016314376bb6?sender=8ca740c7-a56a-428f-bd7c-f14312b59fea

Examen:

https://docs.google.com/spreadsheets/d/1I4cX3gEwMhmyX98m3eM5GvHiXEk3mrLd/edit#gid=751832467