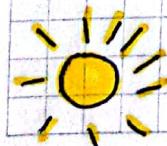


D
E
S
A
R
R
O
L
L
O

D
E

S
O
F
T
W
A
R
E
I



Marzo 8 dd 2023

INTRODUCCIÓN

- Tipos de apps: Móvil, Web, Escritorio, Embedded, Wearable.

- Evaluaciones: $\text{PA1} = 40\%$.

$$\text{PA2} = 55\%$$

$$\text{PA3} = 5\%$$

- Exámenes 30.03%.

- Quirós 10.03%.

- Tareas 9.86%.

- Exposiciones 10.02%.

- Proyecto 35.06%.

- Informes escritos 2.5%.

- Comunicación Oral 2.5%.

} 100%.

- Parcial → Tiene opcional

- Herramientas:

- Ambiente de trabajo: AV o MV

- Frontend: HTML5 + CSS3 + Javascript ↗ Angular, React, Etc. JQuery

- Back-end: Python
Django

- Bases de datos: PostgreSQL, Oracle.

- Herramientas de Gestión: Google Docs, Jira, Togita

- Manejador de versiones: GitHub, Bitbucket.

↳ Etiquetas

- IDE: Visual Studio Code.

- Servidor Web: Apache, Nginx

- Investigar Servicios de Amazon, Github, Azure por estudiante.



- Crisis del Software: Problemas de calidad, Sobrecostos, entregas tardías

- ¿Por qué?

- Problemas de Comunicación
- Falta de detalles **User**
- Intentar hacer un Software "flexible". **User**
- Yo terminé si estás desarrollando **Developer**
- Software no terminado, no evolucionó en calidad **Developer**
- No entregar monolíticos **Developer**
- Si: hay un buen hardware, todo estaría bien. **Admin**
- Retraso → Contrato más Developers → Profit **Admin**

- Tipología de Software

- Tiempo con múltiples roles
- Infinito crecimiento
- Reutilización para reducir costos

• Sistemas complejos

• Familias de Sistemas

• Requerimientos Mantenimiento

- Ciclo de vida del desarrollo de Software

A = Análisis

DI = Diseño

C = Codificación

P = Pruebas

DES = Despliegue

Entender problemas, necesidades

Disección o especificación de procesos, cómo se interactúan

Diseño o log. de programación.

Valor que el sistema finaure adecuadamente.

→ Antes era en meses, ahora es en semanas.

A DI C P DES

Sprint

A DI C P DES

A DI C P DES

Sprint

• Se va haciendo la aplicación por partes. Implementando funcionalidades en cada Sprint

• DevOps:

• Se automatizan algunos puntos: (Despliegue, Pruebas y parte de codificación)

1 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 |

Duración en días

- Pasos:

- Ideación: Uva de ideas → Proyectos
- Clasificación ↙
- Productos ↗
- Diseño (Caracterización de un sistema de Software)
- Despliegue (Colocar sistema en servidores)
- Monitoreo: Monita el rendimiento del Sistema para si necesita Actualización.

- Metodologías

- Colección de métodos aplicados a lo largo del ciclo de vida de desarrollo. Si bien un enfoque o filosofía.

- Tradicionales

- Ciclada (Cada etapa) RCP
- Murdo 15 de 2023

- Agiles

- (A) repetir) → Scrum, XP, etc.

REQUERIMIENTOS DE SOFTWARE

- Es una documentación que establece una característica o restricción deseada, funcional o no funcional de diseño de un producto o proceso.

- Ayuda a: Entender el diseño, estimación de costos y plan de desarrollo.

- Que cumple un requerimiento:

- Resuelve su necesidad.

- Satisface unas condiciones (Documento formal)

- Define lo que necesita

- Clasificación

- Funcionales (Que hace)

Actuar de determinada forma

- No funcionales (Que cumple)

Limitaciones en los servicios o funciones
(Click de 10 segs X)

- Requerimientos funcionales

- Describen el comportamiento del Sistema
 - Se describen por: lenguaje natural, de forma individual y a veces se enumeran.
- Ej: "El Sistema debe permitir Crear un usuario del sistema"

- Trascrito de requerimientos

- Fuente de problemas.

- Características

- Necesario, Correcto, No ambiguo, Conciso, Consistente, Completo, Alcanzable, Verificable.

- Recomendaciones

- Usar lenguaje de manejo consistente, no usar jerga informática, etc.

- Debe de tener una descripción y realizar la especificación de los detalles.

Q. Sistemas
Diseñarlos

• Detalles de los requerimientos se tienen al desarrollador (Datos de entrada, Proceso, Datos de salida, Resultados esperados, Origen, Dirigido a, Prioridad, requerimientos cumplidos).

- Cada empresa define su propio formato y nivel de detalle.

- Proceso para especificar requerimientos

Otros.
• Historia: Captura, descripción y adhesión de requerimientos (Toda información de otros y particularidades) → lo que el sistema haga

- Análisis de requerimientos.

- Especificación: Producción de documentos de requerimientos

- Validación.

- Determinación de requerimientos

- Otros: Entrevista con el usuario, talleres (lluvia de ideas), BPM

- Documentación: SRS (Tiene todos los requerimientos)

- Validación: Comprobar las características de los requerimientos.

- Entrevista

- Diálogo formal donde se busca respuesta a un conjunto de preguntas planificadas.
- Identificar el propósito de la entrevista.
- Identificar posibles entrevistados.
- Establecer el problema planteado.
- Familiarizarse con el vocabulario del negocio.
- Tomar apuntes, ó Grabar!

- Críticas

- Criticar, No dejar terminar, No manifestar interés, Escuchar poco, lenguaje muy técnico o informal, Falta de preparación, Corregir, Falta de cortesía, Tiempo o lugar inadecuado, presentación personal.

- Uso de ideas

- Identificar posibles soluciones a problemas o aplicaciones sin antecedentes.
- Tomar plan: dividir problema en partes pequeñas.
- Proporcionar alternativas, Combinar ideas
- Tener un moderador y límite de tiempo.
- Tomar apuntes, o Grabar!

- BPM (Nr)

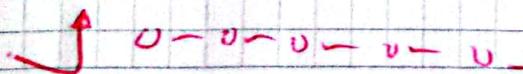
- Diagrama de flujo (Hay de varios tipos)

- Técnica para validación de resultados

1. Validación de expertos (Normativamente impar)

2. Prototipado de interfaz de usuario (Costoso y demorado)

3. Recorrido de BPM. Valerlo o reírse



- Requerimientos no formales

- Propiedades emergentes del Sistema (Tiempo, fiabilidad, etc.)
- También define restricciones

- Tipos:

- Eficiencia, Seguridad de lógica y de datos, Usabilidad (Manual de usuario), Tecnología

- Requerimientos en módulos ógiles

- Basado en procesos ógiles.

(HU)

- Historia de Usuario; Escribir la "historia" del Usuario y oportuna responder a lo que el Usuario necesita.

- ¿Como actuar?

- Mismos métodos de lo metodológico tradicional.

- Características dirigidas al HU?

- Módulo INVEST.

- Ej.: "Como estudiante deseo registrarme en un curso para matricularme en la U."
Actor: Usuario del que usa el sistema. / Tarea: / Propósito:
Comu <Actor> desea <Tarea> para <Propósito>

- Suficientemente Completo

- Centrarse en el Usuario

Se subdivide en HU.

- Si no hay precisión, Historia épica (Grande) (Génerica)

- Spyke → Investigar cosas que no se conocen.

- Dividir historia en tareas sencillas

- Identificar funcionalidades

- Descomponer HU

- Descomponer en tareas técnicas (No pequeñas). Programar tareas con duración de horas para mostrar avances, Buscar si hay componentes ya hechas reutilizables

- Detalles de una historia de usuario

- La HU tiene una estructura.

Tarea: Casos técnicos

Ej: "Mantener la BD".

- Criterios de aceptación de una HU.

- DOD (Definition of Done)

- Product Backlog

- Lista de items, Casos que hoy que hacer.

- Requerimientos (No Funcionales)

- "El Cuadro de búsqueda debe de devolver valores alfanuméricos"

- Validación

- Los mismos del anterior.

○ ~ ○ ~ ○ ~ ○ ~ ○ ~ ○ ~ ○ ~ ○

- Desarrollo Tarea

1. Tarea: "Como tomar requerimientos | 5 minutos para una tarea de rellenamiento efectivo"

- Hola Mundo

- Request For Proposal (RFP)

- Documento extenso que define el Software que necesita el cliente.

2. Revisión Presencial

- Revisión cara a cara.

- Permite presentarse ante el cliente, Conversación fluida y líman aspergazos de la comunicación.

2. Que el cliente describe el problema

- ¿Qué problema o solución? → Para tomar decisión correcta en caso de algún error. Sin llamar constantemente al cliente.

3. Que el cliente describa la experiencia

- ¿Qué software ha estado usando el cliente? → Acá podemos aportar ideas y llegar en algo en común.

4. Apoyarse con Wireframes.

- Debes de anotar la conversación y sus detalles.
- Usar **Wireframe**. (Es casi una "mockup") para mostrar y creando el diseño de app.

5. Considera infraestructura como parte del proyecto.

- Considerar todo lo relacionado para desplegar la app.

6. Firmar el contrato

- Documento redactado → firmar → Agregar un "fee" por el producto final y horas.
- Agregar tipos de horas extra para opciones extras pedidas por el cliente.
- También agregar ideas para agregar otras del cliente.

2.RT = - Requirements Gathering Techniques - Samra Malik

- A useful technique is determined by its need and the kind of advantages it offers in a particular project.

1. **Brainstorming**: For gathering a good number of ideas. It's used for identifying all possible solutions to problems.

2. **Document Analysis**: Evaluating the documentation of a present system can suggest when making AS-IS process documents.

↳ Improving an existing software

3. **Focus group**: People who are customers or representatives in order to gain its feedback (Needs, problems, etc.)

4. **Interface Analysis**: Analysing the touch points with another system from the user for not overlooking requirements.

5. Interview: Pw

6. Observations: Study of Users in its natural habitat (Pain points, awkward steps, etc.). It can be active or passive.

7. Prototyping: Fast sketches. For gathering feedback.

8. Requirements Workshop: Collaborative Workshops between the Client and the developer. More organized and structured.

9. Reverse Engineering: When there is no documentation.

10. Survey: For many clients. For characterizing the market.

3.PU = **Como <Actor> deseó <función> para <propósito>**

- Como Cliente deseo redimir botellas para obtener cupones.
- Como proveedor deseo obtener botellas para reciclarlas.
- Como cliente deseo tener soporte para solventar

Martes 22 de 2022

METODOLOGIAS SCRUM

• Práctica: Forma específica de hacer algo

• Existe un Manifiesto Ágil (4 Valores - 12 Principios)

• Valores

• Individuos e interacciones

• Software Funcionando

• Colaboración con el Cliente

• Respuesta ante el cambio

- Principios

1. Entrega Temprana y de Calidad

2. Los requisitos pueden cambiar

3. Software funcional frente código 2 Semanas

4. Requerimientos y desarrollo en trío/un jefe

5. Motivar al equipo frente al proyecto.

6. Comunicación: Cara a Cara.

7. Software Andando → Medir de progreso.

8. Promover desarrollo sostenible (Avances Uniformes)

9. Buon diseño y excelencia técnica.

10. Hacer cenas prácticas y precisas.

11. Fuerzas reunidas organizadas.

12. Reflejarse sobre intervalos regulares.

- SCRUM

- ¿Quién y ¿Qué?: Identificación de roles.
- ¿Dónde y Cuándo?: Sprint (Iteración)
- ¿Por qué y Cómo?: Herramientas

- Roles

- **Project Owner:** Persona que identifica los requerimientos. Se comunica con el Cliente.
- **Product Backlog:** Lista de requerimientos
- **Sprint Planning meeting:** ¿Qué se va a hacer?
- **Sprint Backlog:** Funcionalidades para el Sprint.
- **Team:** Equipo desarrollador.
- **Scrum master:** Persona experta en la metodología.
- **Burndown:** Página entrega de Software (Trenzado de Software)
- **Burndown:** Ver como nos fue. Medir de forma decreciente el avance.
- **Sprint review:** Presentación del avance al equipo y clientes.

- Priorización: Ver que tareas cuales son las más importantes. Se pueden clasificar.

- **Criterios:** Necesidad del Cliente, Añade valor al Sistema, Solida oportunidad de mercado, Requeridos por otras historias, riesgo que implica desarrollar.
- **Técnicas:** Clasificación de lista, de 1 hasta n → Prioridad de orden. Cualquier se hace primero.
- **Priorización:** Alto, Medio, Bajo. El orden para implementar serían los altos.
- **M.S.C.o.W:** Valor Semántico (Must, Should, Could, Won't)

• También se mide la estimación de uno HV por puntos. **Ej:** 1 punto = 'X' horas. # P

Planning Poker:		HV	Tarea 1	1
Historia base	1	2	1.5	
	2	3	2.5	
	3	4	3.5	
	4	5	4.5	
	5	6	5.5	
	6	+ 3		

11 horas → 1 punto ☺ → Muy grande ? → No le queda claro.

↑ → Salir o pensar

Con los puntos se estima la duración a partir de la duración de uno HV y se sustra un punto y se escoge la media.

- Poco al principio usaremos creación-plotación
 - Si se acuerda usar metáforas (?)

Sprint 1: Intervalo de tiempo en el que se hace un grupo de tareas.

- Retoque plan: Vision de producto disponible, para el usuario.

Mon 7.9.2023

METODOLOGIA XP

→ Anotaciones

- Projects: Daily meeting, Sprint review, etc. ≠ • Methodologies: Scrum, XP, etc.

→ XP (eXtreme Programming)

- Roles: Cliente, Programador, Verificador, Consultor técnico, Coach o mentor, Seguimiento de resultados (tracker) y Un Gran Jefe.

- **Programación en paralelo:** Para mejorar la productividad y calidad del código. Piloto/Copiloto o Intercambios regulares.

- Diáscus, Simus, Glorius, Tigris (Spyke), Funeritudo extra (Solo para lo necesario), Refutator, Codificación, Proverbios.

→ Criterio de Análisis: SMART (Ver Júpito)

- ## • Técnica de extracción

- i) Como se responde?

- Sprint 0: Preparación, avances al desarrollo. (Ver retroalimentación / esquema)

- Se debesa kira la BD ya creeda (?)

- ? Où se trouve en Côte d'Ivoire?

- **Poss Web** de aplicaciones: Para lo general en dos contenedores, uno para el objetivo y otro para la BD.

- BPMN

- Sistemas de punto el mecanismo de procesos de negocios (Prosesos para hacer una venta)

- Cada actividad () es una HU.

1. Se

1. See Jeffun pictures

2.

April 12 Jd 2023

CONCEPTOS BÁSICOS EN EL DESARROLLO DE APPS

- **Actor:** Quién interactúa
 - **Tel:** Papel de una persona en el sistema.
 - **Perfil:** Permisos que tiene acceso por determinado rol.

- **Interfaces de Usuario**

 - Representación fluida visual de una aplicación de software.
 - Maqueta del sistema, Navegación entre pantallas, Se obtiene retroalimentación de los clicos, Son un complemento, etc.

- **Partes: Widgets**

 - Botones, Campos de texto, Selectores, etc.
 - Herramientas para diseño: Figma, Miro, Mockplus, etc. También puedes usar plantillas.

Para el proyecto usamos una plantilla :D

- **Reportes**

 - Presenta informaciones disponibles en una App que puede ser útil para los usuarios o clientes (Ex: Consulta a una TSD)

- **DataTable**

 - Una tabla (x), con un reporte se puede mostrar en uno gráfico.

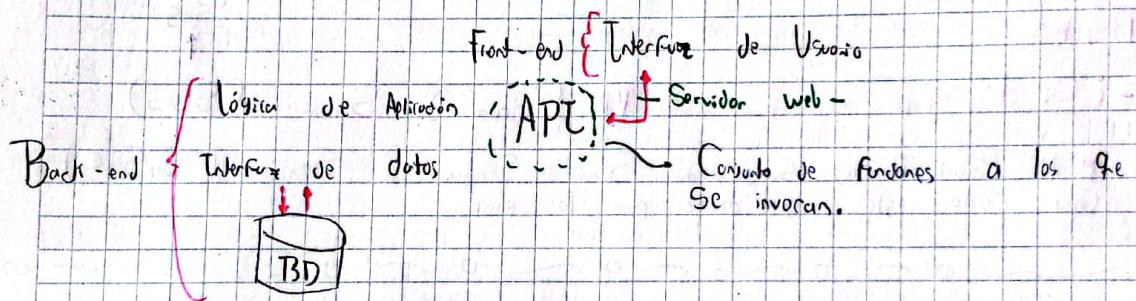
- **Dashboard**

 - Visualizar datos de manera resumida.

Arquitecturas de Software

- Atributos de calidad: Testabilidad, Mantenibilidad, Despliegabilidad, Escalabilidad, Evolución, (Grado de principio) Rendimiento, Interoperabilidad, Seguridad.
- En una aplicación web describe componentes e interacciones entre el frontend y el back-end.
- Son dos códigos: Del lado del Cliente y del Servidor.
"Servidor: Recibe peticiones"
"Cliente: Navegar web"
- Arquitectura para aplicaciones integradas (monolíticas)

Arquitecturas para aplicaciones desacopladas en API



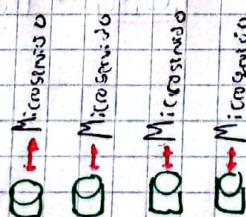
Arquitectura de microservicios.

- El Backend se divide en aplicaciones

Front-end { Interfaz de usuario

Servidor Web

Capa de agregación (REST, etc.)



Arquitectura de Micro Front-ends.



Arquitectura Hexagonal: (firma de estructura código)

Arquitectura basada en puertos:

Abi 19 del 2023

ESTIMACIONES

- Estimación de tiempo y costos

- A partir de release plan → Tiempo
- Costo: lo que cuesta hacer algo "Unidad"
- Precio: cantidad de dinero que la empresa espera recibir (Costo + Ganancias)
- Valor: lo que el cliente está dispuesto a pagar.

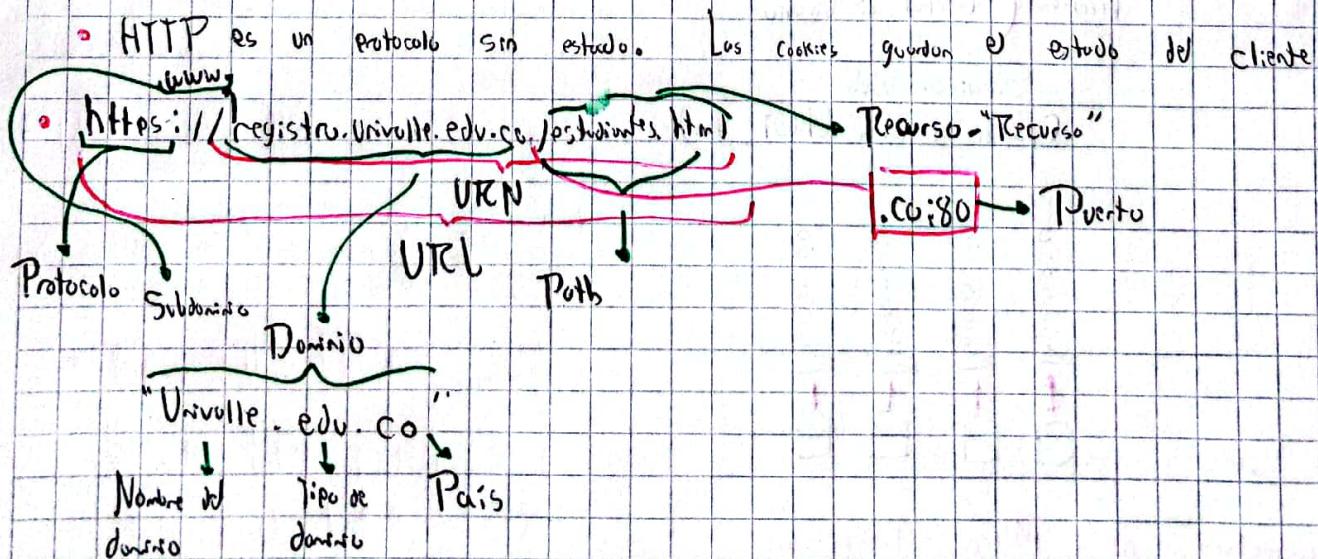
- Preguntas

- ¿Si el equipo es muy grande? TLL = Scrum de Scrums (SOS)

• El Project Backlog si es grande Scrum de Scrums de Scrums (SoSoS)
Unív Solo, este se divide por los equipos

APLICACIONES WEB

- Tipos



— o ~ o ~ o ~ o ~
April 26 2023

MODELO DE DESPLIEGUE

- Definir cuáles servicios se necesitan.
 - Servicios físicos → Servidores virtuales → Contenedores → Serverless
 - X Hardware
 - X OS
 - X Language Runtime
 - X App Container.
 - (1) (2) (3) (4)
 - Pruebas Staging Producción ← "Ambiente de pruebas"
 - ← "Ambiente de producción - con datos reales"
 - El despliegue "4"
 - T Repositorio
 - Desarrollo ← "Ambiente de desarrollo"
 - TSTD
 - Por lo general se hace un nodo para la TSTD y el otro para la aplicación.

- Por lo general se hace un nodo para la BD y otro para la aplicación.
- Deploy
 - Ejecutables o Script, librerías, Archivos de datos y BD con drivers de conexión
- Estrategias para actualizar
 - Pausar → Cambiar → Prender.
 - Rolling Upgrade: Actualizando servidores uno por uno.
 - Blue/Green: Duplicar infraestructura, llevar actualización, Reemplazar.
 - Canary Deployment: Despliegue con algunos servidores, y consigo usuarios.

ARQUITECTURAS DE SOFTWARE

- Documento: Documento entre 4 y 6 págs.

- Contiene los resultados de las prácticas ágiles

- Resistirse a elefante.

Mayo 03 de 2023

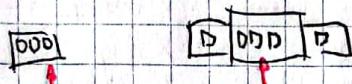
UNA APP WEB - ACUERDOS INICIALES

- Frontend (Frontend)

• DOM: Document Object Model

Es una API que representa un HTML o XML como un árbol de nodos.

• Podemos seleccionar elementos con métodos. Ej: getElementById()



AJAX

• SPA VS MPA

- SPA

• Funcionalidades en una sola página. La lógica se ejecuta en el navegador.

- MPA

• Múltiples páginas cuando el usuario interactúa.

- Backend

• Procesar solicitudes entrantes, generar y enviar la respuesta al Cliente.

• Son numerosos

• Existen algunos criterios para seleccionar.

~ Bases de datos

• Relacional, No SQL and NewSQL

• Pip install block

block {file.py}

• Close virtual ↔ 8 días



O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O

TABLEAU KANBAN

- Busca mejorar los cuellos de botella. Método "Pull", Apesar se recibe la orden, se suple la demanda.
- 5 partes → Señales visuales, columnas, límites del trabajo en curso, Punto de compromiso y punto de entrega.
 - Cuentes trabajos por columna
- Herramientas: Trello o Jira.

O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O ~ O

Chrome Dev Tools

- Herramientas para desarrolladores web.
- Se pueden seleccionar elementos del DOM.
- Ver Javascript: Source > Buscar ".js". Desde aquí se pueden usar breakpoints para el debugger.
- Performance: Inspector > Performance
- Lighthouse. También para rendimiento. Inspector > Lighthouse

Mayo 31 del 2023

TIPOS DE APLICACIONES WEB

- Orientadas a transacciones

- * Hoy dominio de negocio sobre transacciones.
- ~ Busesca en Almacenes de trabajo (Workflows)

- Vía por etapas. Ex: Ulenor formulario → Agregar → Realizar compra



• B2C : Business To Consumer

- Aplicaciones Colaborativas.

~~~~~

### - Orientadas a portales

- Contenido de más de una fuente para crear un servicio único.

### - Scraping

- Consultar automáticamente páginas Web y extraer de ellas información
- AI Software se le llama "Bot" Spider o Crawler.

### - Interoperabilidad

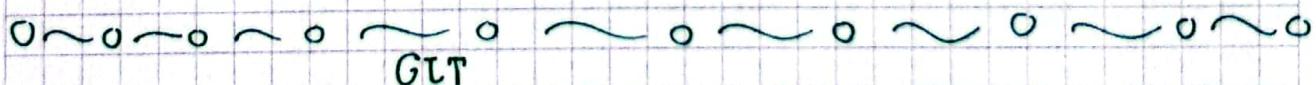
- Apoyados en Web Services (Permiten que aplicaciones intercambien datos)
- Serialización: Convertir datos a texto (XML, JSON, YAML, Etc.)

### - Ubicuos

- Usar extractos basado en Ontologías. Ex: <news> m </news>

## VERSIONES

- Contar la evolución de un Software
  - Por estabilidad:
    - Alpha: Versión instable, para encontrar errores, para los desarrolladores
    - Beta: Una poco más estable, producto en su totalidad, la ponen a prueba los usuarios.
    - Release (RC): Versión final y estable. Último tope fino. "Technical Candidate"
  - Versiónado por números
    - "X.Y.Z.P"
    - X = Versión mayor
    - Y = Versión menor
    - Z = Revisión del código
    - P = Parche



- Git tag <filename>
  - Git flow: Flujo de trabajo para trabajo mejor con Git.
    - Master: Código de Producción. Nunca se toca directamente.
    - Hotfix: Corregir errores críticos
    - Develop Branch: Para crear nuevas versiones del código
    - Feature: De esta rama nace Feature.
    - Feature: Para añadir implementaciones.

- Trunk-based Flow
- Muster-only flow 100