**Tribu de Microservicios - Diccionario para Novatos**

**Luis Lázaro Hoz - Ivan Vozmediano Martínez**

# Introducción:

Bienvenido a la tribu de microservicios. Tanto si te lo crees como si no, ahora, formas parte del equipo (¡Enhorabuena colega!).

Es posible (prácticamente seguro) que pienses “¿Qué hago yo aquí? ¡SI YO NO SOY INFORMÁTICO!”, eso importa más bien poco. Vengas de donde vengas (mecánica, economía, matemáticas, etc.), estás aquí por lo que puedes llegar a hacer.

Las próximas semanas las dedicarás a cursos de programación, de tecnologías, filosofías, métodos, herramientas y un sinfín de cosas nuevas. Bien, al principio no te enterarás de la mitad (y de la otra mitad, seamos sinceros, muy poco). Es normal, estás aprendiendo un nuevo idioma en toda regla. Algunas cosas las entenderás por el contexto, otras te las tendrán que explicar 7 veces y otras... Bueno, otras siguen siendo un misterio en la oficina.

Para todo lo demás se ha hecho este diccionario. Empecemos primero con unos conceptos básicos:

1º- Casi toda la terminología está en inglés (ellos lo inventan, ellos le ponen el nombre) y podrás encontrarla en el buscador de Google y páginas esenciales como <https://stackoverflow.com/>, <https://mvnrepository.com/> o <https://www.findjar.com>, entre muchas otras que serán tus fuentes de soluciones y referencias más habituales, pero hay muchísimas más.

2º- Trata de organizar tu entorno de trabajo (físico y virtual) para que te sea mas cómodo (color y brillo de la pantalla, asiento, teclado, organización de los archivos, etc.). Cuanto más fácil te sea trabajar, menos te molestarán las eventualidades que puedan aparecer.

3º- Si tienes dudas pregunta sin miedo, puede que algunas de tus dudas revelen detalles o soluciones que no se habían planteado.

Y lo más importante: acuérdate de que los miércoles salimos todos a comer juntos.

**\*Nota: he metido términos pero quizá es importante el orden (por ejemplo: hablar de Lombok sin explicar primero qué es Spring quizá tiene poco sentido)**

# Terminología Técnica

**- Consola (CMD, GIT Bash, Cygwin, etc.)**

Las consolas de comandos, o interpretes de comandos, son programas que permiten ejecutar sentencias,o comandos, y tareas:

· Ejecutar comandos de Frameworks,

· Gestionar conexiones de Red e Internet

· Instalar paquetes y utilidades.

· Acceder a Servidores en la Nube.

· Gestionar archivos y carpetas.

· Ejecutar funciones en el Sistema Operativo

A traves de ellas realizamos

**- Entorno de de desarrollo (IDE):**

Es el “programa” en el que escribes el código, lo ejecutas y te indica los errores (si, los que te van a desesperar).

Los mas utilizados por el equipo son [Eclipse](https://www.eclipse.org/), [Intellij](https://www.jetbrains.com/idea/) y [SpringToolsSuite](https://spring.io/tools)

Estos entornos son para programa Java, esencialmente todos funcionan de la misma manera, pero con ligeras diferencias entre si (Estética, complementos, comandos particulares, etc.)

Te parecerán muy complejos, indescifrables incluso, pero a medida que vayas trabajando, veas a los compañeros e investigues un poco iras consiguiendo un truco de poco en poco (Dar cera, pulir cera).

**- Java (El idioma)**

Se trata de un lenguaje de programación, es decir, lo utilizas para comunicarte con la maquina para que esta haga lo que le indiques.

Se trata de un [lenguaje orientado a objetos](https://es.m.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), un lenguaje en el que se definen entidades, u Objetos, que recogen métodos, atributos e identidades. Definiendo estos elementos y las relaciones entre si generamos programas y servicios.

La principal ventaja de Java sobre otros lenguajes es su capacidad de trabajo sobre múltiples plataformas (ordenadores, servidores, etc), dejando a un lado las diferencias de hardware entre ellos, salvo por sensibles diferencias, un código generado en un ordenador, si lo ejecutamos en cualquier otra plataforma deberla funcionar sin mayores complicaciones en cualquier otra plataforma.

A medida que vayas integrandote mas y mas empezaras a ser consciente de las infinidad de posibilidades que puede haber.

**Framework**

**Interfaz de programación de aplicaciones (API)**

**GIT (meter algún comandillo o enlace a alguna web con comandos)**

**Consola (CMD, GIT Bash, Cygwin, etc.)**

**Docker (igual que GIT)**

**Maven**

**Anotación (@)**

**Lombok**

**Spring**

**REST**

**Postman**

**Base de datos**

**Entorno desarrollo de Base de datos (DBeaver, SQL Developer, etc.)**

**SQL**

**Historias de Usuario (HdU)**

En las reuniones con el cliente, o su representantes, se definen en un lenguaje claro y sencillo los deseos y necesidades del clientes

De las historias, una vez bien definidas, los integrantes del equipo las analizan y extraen las tareas precisas para poder obtener los deseos del cliente.

Este elemento, aparentemente simple, es uno de los mayores quebraderos de cabeza para todo el equipo, una HdU bien o mal definida y/o analizada puede ahorrar mucho tiempo y malentendidos al equipo, o tirar por tierra todo el trabajo realizado (con el consiguiente cabreo).

Ejemplo “Quiero las paredes de la casa pintadas, interiores de rojo y las de la fachada de azul”

De esta historia se pueden extraer las siguientes tareas, por orden de prioridad:

- Identificar que paredes son de fachada y cuales de interior.

- Averiguar que clase de rojo y azul quiere el cliente

- Calcular cuanta de pintura es necesaria.

- Comprar los elementos necesarios.

- Pintar

# 2- Personal

**Tu Responsable (tu jefe Jefe)**

Es quien te ha dado la bienvenida a esta empresa, quien te ha proporcionado, practicamente todo (Fue la persona cuyo “Si” hicieron que te ficharan).

Sus funciones son bastante

**Cliente (Aquel del que nunca hablamos)**

Como la misma palabra indica, es a quien le prestamos un servicio, a contratado a nuestra empresa ,y por ende a nosotros, para que le proporcionemos una solución a una de sus necesidades.

**Product Owner (El del otro lado de la mesa)**

Es el representante del cliente, no pertenece al equipo perse pero, dependiendo del proyecto, como si lo fuera. Normalmente solo debería tratar con el Jefe de proyecto o el Scrum Master. Indica las necesidades del cliente al jefe de proyecto, controla que se cumplan los tiempos de trabajo, analiza los resultados etc.

INCISO: te puede dar la sensación de que es tu “jefe”, y que si se dirige directamente a ti tendrás que atender a sus peticiones. Más bien no, cualquier cosa que te pida debe estar aprobado por el jefe de proyecto y el equipo, cualquier detalle que te solicite, comunícaselo a tu jefe de proyecto.

**Jefe de Proyecto (El capitán)**

El cabeza de proyecto, quien se pasa casi todo el día reunido con el cliente para saber que es lo que quiere, traducírselo al equipo para saber que pasos hay que seguir.

Aunque te parezca que su trabajo es mas “sencillo” que el tuyo, nada mas lejos de la realidad, son los que menos duermen del equipo y que hasta mas tarde se quedan, de ellos dependen directamente el trabajo del resto del equipo.

**Scrum Master (El Contramaestre)**

Es el que se encarga de que todo funcione como debe, que el equipo te funcione bien, explicar a todo el mundo si hubiera algún malentendido y mediar entre los integrantes del equipo de haber algún conflicto.

No confundir con el jefe de proyecto, el Scrum Master alivia la carga del jefe de proyecto centrándose en la gestión del equipo.

Para mas detalles revisar Filosofia Agile

**Devops**

**UX**

# 3- Proyecto

**Front**

**Backend**

**Sprint**

**Una vez se han definido las HdU a desarrollar, y las tareas a desarrollar a partir de estas.**

**P**

**De**

**Sprint Review**

**Restrospectiva**

**Refinamiento**