¿Qué es desarrollar software?

Desarrollar software es el proceso iterativo (* por qué iterativo) en el cual definimos instrucciones precisas (definir instrucciones no necesariamente programando, por config) para ser ejecutadas por el sistema intérprete encargado de representar funcionalmente el universo que da existencia a esas instrucciones.

La aplicación(/el software) no son las instrucciones lógicas, si no, la representación de las mismas. Y entre la instrucción y la representación hay una gran cantidad de objetos y procesos que la permiten -a la representación-. La representación del vidrio no es posible sin la arena, el calor para la fundición (la máquina que produzca ese calor (los elementos con los que la máquina están creados (la máquina que creó esos elementos (los elementos con los que se creó esa máquina(los materiales con los que se crearon esos elementos(...(...(...(...)))))))))).
De la misma manera podríamos listar los elementos y procesos necesarios, casi infinitamente, que son necesarios para representar nuestras instrucciones lógicas. Y de la misma manera que compraríamos una máquina de fundición, en vez de construirla, para fundir vidrio, utilizamos herramientas ya existentes para aplicar nuestras reglas específicas. Si desarrollamos una aplicación web, no desarrollaremos el programa que recibe peticiones HTTP por el puerto 80, las direcciona a un programa que la interpreta y genera una respuesta, y envía la respuesta a quien inició la petición, simplemente instalaríamos apache, o nginx.

Este proceso es iterativo por la propia definición del concepto ‘software’, etimológicamente “producto blando” (soft+ware), un objeto maleable, no estricto, modificable y adaptable. El software permite la opción de cambiar un sistema de uso específico tanto como sea necesario, a contraposición de los sistemas físicos, por ejemplo, un disco de corte para una amoladora no serviría si quisiéramos pulir, deberíamos cambiar completamente el objeto.

Estos cambios son los requisitos que el software debe cumplir, su motivo de existencia, las reglas que existan con o sin el software, son nombrados “reglas de negocio”. Por ejemplo, en un sistema de ventas, el concepto de realizar una transacción de pago y registrarla, va a existir cualquiera sea el sistema que lo represente (una persona, un posnet, una aplicación móvil).
La implementación de estos cambios no debe ser considerada como una novedad o una nueva parte del sistema, si no, como una tarea más a llevarse a cabo en el iterativo proceso de implementar y modificar requisitos en cuanto a que lo que el software representa lo necesite.

Para incluir o implementar estos cambios (o adiciones) en los requisitos, es necesario traducirlos al entorno de quien va a hacerlos cumplir, nuestro software. La traducción no es -tan solo- ‘escribir en lenguaje máquina’, sino, desmenuzar los requisitos en instrucciones lógicas precisas que puedan ser escritas en un idioma que el desarrollador y la computadora puedan comprender.

Cosmologicamente hablando, podemos pensar a las instrucciones lógicas como leyes del universo, y el universo como todo lo que existe que da sentido y funcionamiento a estas leyes.
El universo en el desarrollo del software es lo que va a ejecutar las instrucciones, el entorno, el software donde se va a ejecutar nuestro software. Definir la existencia de un usuario en nuestra aplicación no sería real si no hubiera un entorno donde el usuario exista (una base de datos, la memoria ram, el campo 'nombre de usuario' en un formulario), y mucho menos sentido si no existiera una aplicación.

Desarrollar software es, entonces, identificar los requisitos que dan sentido a la existencia del software, encontrar qué herramientas y objetos son necesarios para su representación y traducir los requisitos en instrucciones precisas aplicables a las herramientas y objetos que hemos elegido. Todo esto, con una precedencia al cambio, y cada vez que los requisitos cambien.

Desarrollar versus programar

Desarrollar no necesariamente es programar, pero programar si es desarrollar. La programación, la escritura de código, es tan solo una parte en el proceso iterativo del desarrollo de software. Hay partes de este mismo proceso que no incluyen ni una línea de código, ni una computadora siquiera, como la definición de requisitos, el flujo accionar de un usuario dentro de nuestra aplicación, etc.

La definición del universo, la infraestructura

El software donde va a correr nuestro software. El sistema operativo que va a tener el servidor, cuantos servidores, que cosa va a hacer cada uno, el lenguaje de programación en el que va a estar escrita la aplicación (o los lenguajes), el intérprete o compilador de ese lenguaje, el motor de la base de datos, el tipo de protocolo de intercambio de datos que más se adecúe a nuestro caso de uso. Todas estas decisiones conforman la definición del universo donde vivirá nuestra aplicación, este universo es correctamente llamado "infraestructura".
La definición de la infraestructura debe ser un proceso muy meticuloso y personalizado al caso de uso con el que estemos trabajando. Incorrectamente, hoy en día se cree que 'la nube' es la infraestructura *más mejor* para todo tipo de casos de uso.
¿Qué es la famosa nube? La famosa nube es una colección de servicios que ofrecen los proveedores de infraestructura online. Este servicio es 'on demand'. Si hoy necesitamos 2 GBs de RAM y mañana 4, contratando un servicio en nube, no es necesario cambiar de servidor o contratar otro. Si necesitamos una base de datos, no necesitamos instalarla y configurarla, tan solo la "solicitamos". Pero la nube no es muy distinta a cualquier servidor que cualquiera pueda contratar y configurar, tan sólo, ya está configurado. Suena a bueno y

mejor, pero de la misma manera, es muchísimo más costoso. El costo es parte de la infraestructura.
La definición de la infraestructura es un trabajo arquitectural, de diseño y de análisis e investigación.
Es de los procesos más importantes, ya que, como se mencionó antes, los requisitos cambian, y estos cambios de requisitos pueden traducirse a cambios de infraestructura. Con una infraestructura bien diseñada y modularizada, ningún cambio sería catastrófico. Si la infraestructura está 'atada con alambre', cualquier cambio que hagamos, tiraría abajo todo el edificio.

Desarrollo web / no web

Sorprendentemente, no hay muchas diferencias entre el desarrollo de una aplicación web y una aplicación no web. Hoy en día, pueden hasta no variar ni en el lenguaje de programación.
La web es un modelo de infraestructura, donde desde un comienzo está definido el protocolo de intercambio de datos (HTTP, Hyper Text Transfer Protocol), el motor gráfico (el navegador) y una limitación en la selección de lenguajes y librerías.
Una aplicación 'no web', podría estar de igual manera estar escrita en un lenguaje de programación que usualmente se use para la programación web, podría también usar el protocolo HTTP, pero no usar el navegador para mostrarse.
¿Qué hace que una aplicación sea web? La utilización del protocolo HTTP, ser renderizada en el navegador web y ser accesible desde el internet. Pero la aplicación puede ser de cualquier manera y cumplir cualquier funcionalidad. También una aplicación no web puede ser convertida a web, o ejecutada en un ambiente web, y viceversa.
En sintesis, no hay mucha diferencia entre una aplicación web y no web, personalmente, no utilizaría tal concepto para diferenciar aplicaciones.
La diferenciación más apropiada en mi opinión, pasa por la funcionalidad de la aplicación, si es una aplicación orientada a usuarios, a desarrolladores, o a la misma computadora.