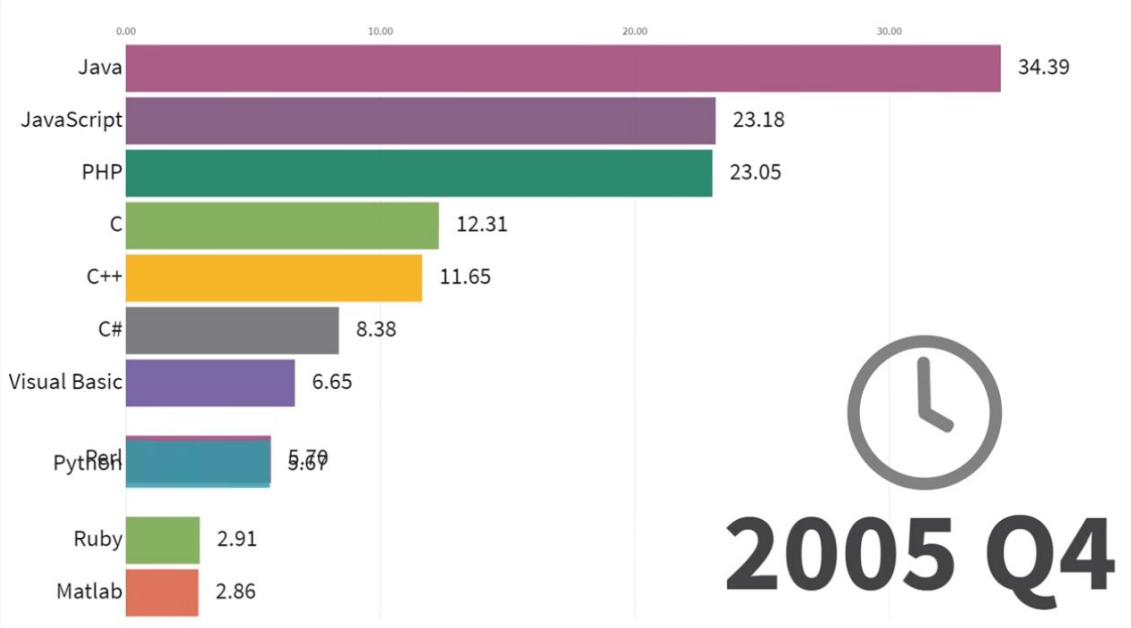
¿Por dónde empiezo en programación?

Carlos Yáñez

Evolución de los Lenguajes de Programación



https://www.youtube.com/watch?v=Og847HVwRSI

Open Source vs Propietario

Open Source VS Propietario

Open Source

- No significa que sea gratis
- El código está disponible para que cualquiera lo pueda modificar
- En principio, cualquiera lo puede modificar
- Generalmente, lo desarrolla una comunidad e incluso empresas
- Está en continua evolución -> Menos errores
- Existe un margen para adaptar el software a tus necesidades específicas
- Pequeñas y medianas empresas

Open Source VS Propietario

Propietario

- Es de pago
- Es caro
- Una única organización controla todo el software
- Necesitas a la empresa propietaria para hacer cualquier cambio
- Cualquier soporte o actualización se hará mediante un pago o suscripción
- Si aparece un error es la empresa propietaria la que lo tiene que solucionar
- Más “user friendly”
- Grandes empresas

Alto Nivel vs Bajo Nivel

Alto Nivel || Bajo Nivel

Alto Nivel

- Más parecido al lenguaje humano
- Más fácil de aprender
- Tiene capas de software que “traducen” LH -> LM
- Lenguajes modernos

Bajo Nivel

- Más parecido al lenguaje original de las máquinas
- Más difícil de aprender
- Funciona más cercano a la máquina
- Lenguajes más antiguos

Lenguajes de Programación

JavaScript

- Desarrollo Web
- Alto nivel
- Dar comportamiento a una web
- Se ejecuta en el navegador
- Inicialmente pensado para web, pero actualmente tiene muchos otros propósitos
- Frameworks:
  - Web: Angular, ReactJS, Vue.JS...
  - Servidores: Node.JS, Express.js
  - Móviles: React Native, Ionic...

Java

- Lenguaje multipropósito
- Alto Nivel
- Está en todas partes: ordenadores, móviles, webs, servidores, hardware…
- Su importancia inicial era que el código se podía escribir independientemente de dónde fuese a ser ejecutado
- Es el lenguaje que más aspectos abarca
- Infinidad de librerías, APIs y frameworks
- Frameworks: Springboot, Hibernate, Play...

Python

- Uno de los lenguajes que más están creciendo
- Alto nivel
- Cada vez abarca más ramas
- Un lenguaje centrado en la simplicidad y sencillez de uso
- Multipropósito: Web, Data Science, Machine Learning... y cada vez más aspectos
- Cuenta con multitud de librerías, dedicadas a diferentes aspectos, que potencia mucho a Python
- Frameworks:
    - Web: Flask, Django, Pyramid
    - Data Science: Scrapy

C

- Lenguaje muy antiguo
- Está resurgiendo después de unos años de capa caída
- Bajo nivel: dependiente del hardware
- Permite darle comportamiento a máquinas y dispositivos
- Internet of Things
- Se trabaja un poco “a mano”

C#

- Pertenece a la plataforma .NET de Microsoft
- Alto Nivel
- Tecnologías del ecosistema Microsoft: C#, ASP.NET, Visual Basic, Azure, TypeScript, SQLServer...
- Normalmente para grandes empresas
- Algunas versiones de las herramientas son de pago
- Abarca muchos ámbitos: desde web hasta HoloLens

PHP

- Antiguamente muy usado
- Alto nivel
- Muy de capa caída, no ha sabido reinventarse
- Desarrollo web, servidor
- Hoy en día suele usarse para añadir mejoras a proyectos que ya existen
- Tiene CMS basados en el: Magento, Prestashop, Wordpress que generan webs de muy baja calidad, lentas y anticuadas
- Frameworks:
    - Web: Symfony, Laravel, CodeIgniter

Librería vs API vs Framework

¿Qué es una Librería?

- Un conjunto de código que ya está hecho
- Nos da hecho parte del trabajo, una parte que suele ser básica o muy común
- Por lo general, el código es visible para los programadores
- Diferentes programadores o empresas lo desarrollan y lo comparten para todos
- Suelen tener un propósito determinado (más o menos amplio)
- Simplifican, automatizan o aceleran el desarrollo
- Escribes, en principio, menos código -> Desarrollas más rápido
- Automatizan en un paso código que te llevaría muchos pasos desarrollar
- En un proyecto puedes usar muchas librerías diferentes
- Todos los lenguajes cuentan con librerías
- Bootstrap

¿Qué es una API?

- Application Programming Interface
- Se puede ver como una aplicación independiente que interactúa con la nuestra
- Es como una librería, pero no se tiene en cuenta el código que compone la librería, sino sólo su uso
- El código no es visible para los programadores
- El resto de principios de la librería, también se aplican
- Pueden servir para interactuar con otros sistemas: Whatsapp, Google Maps...

¿Qué es un framework?

- Va más allá que una librería o una API, no es solo algo que te dan hecho, es un marco nuevo de trabajo
- Tiene una estructura ya definida
- Todo lo que se desarrolla, se hace dentro del framework, siguiendo sus reglas
- Es un marco de trabajo completo, con una series de librerías ya listas para poder utilizarlas
- No tiene necesidad de utilizar librerías externas, aunque existen
- Normalmente están basados en lenguajes mainstream: Java, JS, …
- Angular y React de JS, Spring de Java, Django de Python...

Niveles Profesionales

DEVELOPER LEVELS

JUNIOR

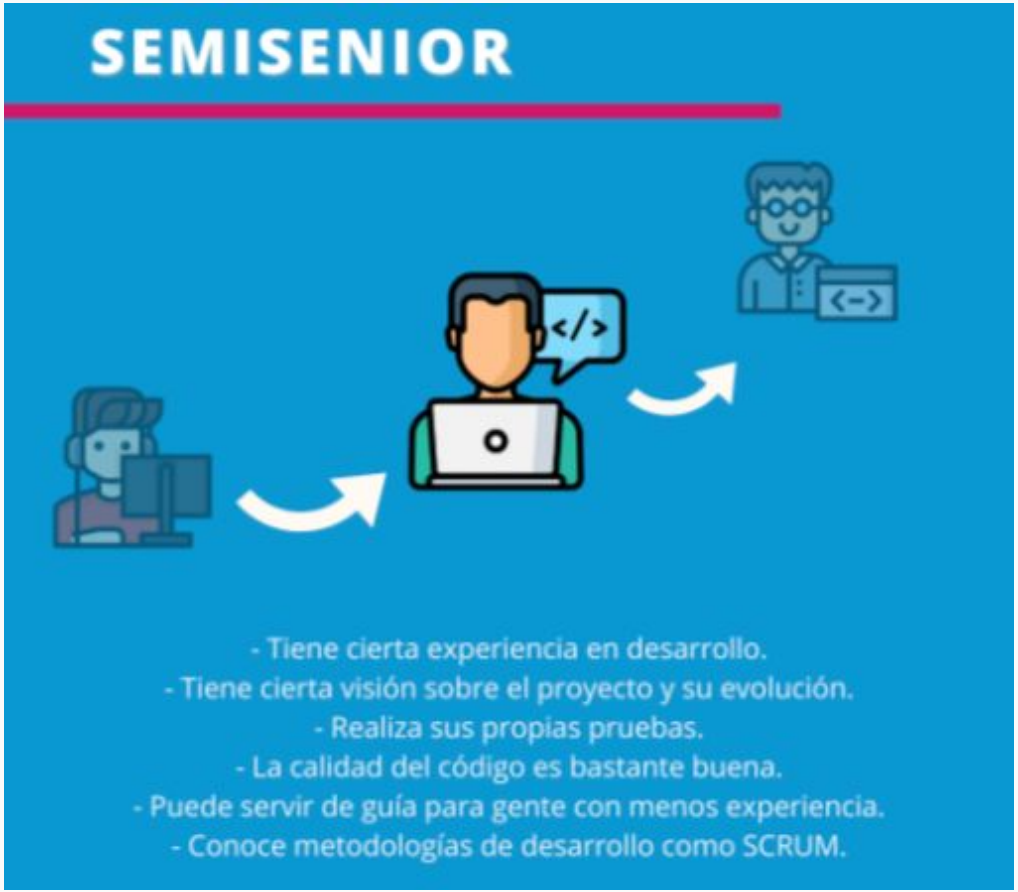
SEMI SENIOR

SENIOR

Jr

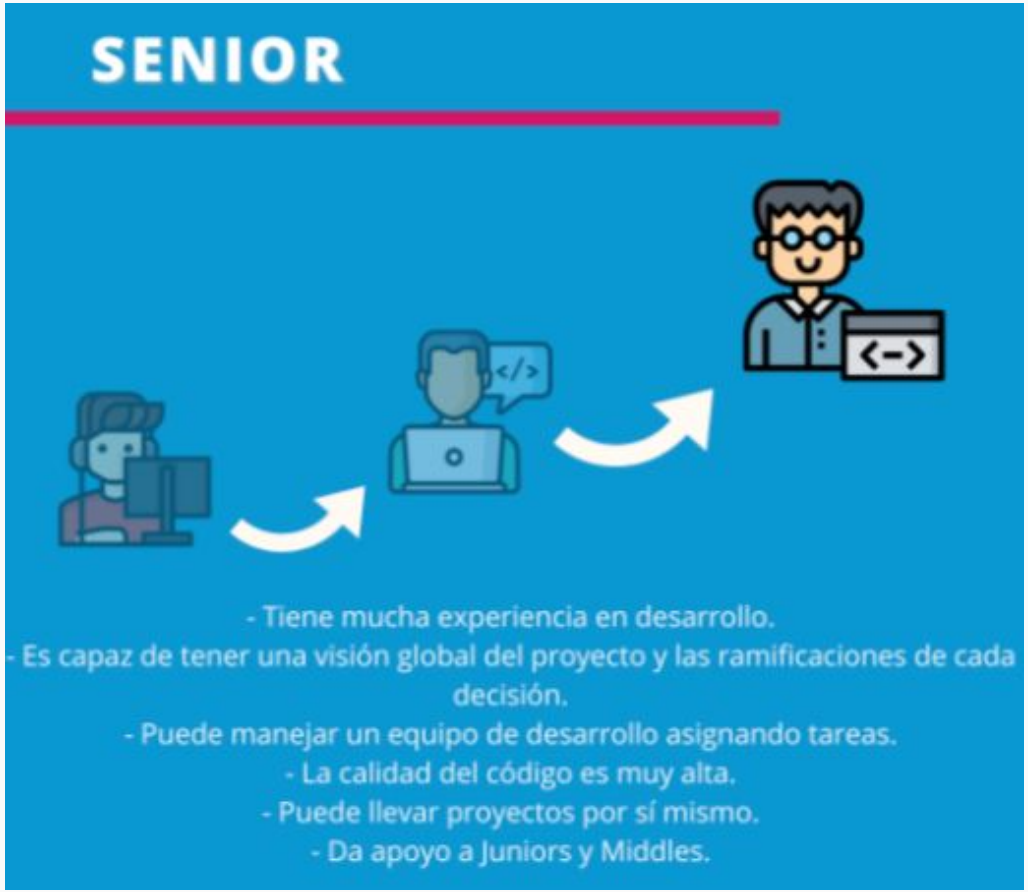

JUNIOR
- Tiene los conceptos básicos de un lenguaje de programación.
- Le falta soltura en el desarrollo.
- Se le dan tareas para su nivel.
- Participa en otras tareas con otra persona con más experiencia que le supervise.
- Realiza código de baja calidad.

Mid

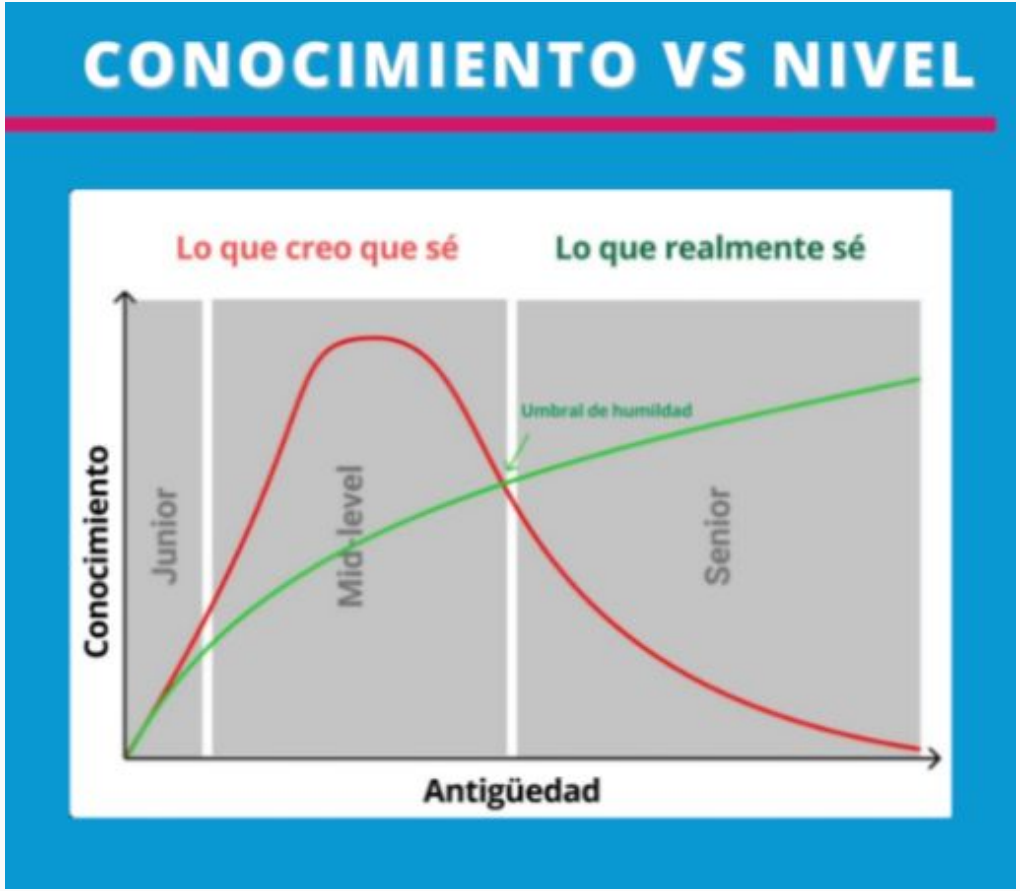


- Tiene cierta experiencia en desarrollo.
- Tiene cierta visión sobre el proyecto y su evolución.
- Realiza sus propias pruebas.
- La calidad del código es bastante buena.
- Puede servir de guía para gente con menos experiencia.
- Conoce metodologías de desarrollo como SCRUM.

Sr


SENIOR
- Tiene mucha experiencia en desarrollo.
- Es capaz de tener una visión global del proyecto y las ramificaciones de cada decisión.
- Puede manejar un equipo de desarrollo asignando tareas.
- La calidad del código es muy alta.
- Puede llevar proyectos por sí mismo.
- Da apoyo a Juniors y Middles.

Resumen


CONOCIMIENTO VS NIVEL
Lo que creo que sé
Lo que realmente sé
Umbral de humildad
Conocimiento
Junior
Mid-level
Senior
Antigüedad

Tipos de Programadores

Programador (Jr - Mid - Sr)

- Escribe el código
- Hace sus tests
- Va a requerir de ayuda
- No debe tomar muchas decisiones al principio
- Suele tener lagunas en el aprendizaje
- Ninguna o muy poca experiencia

Analista (Mid - Sr)

- Tiene experiencia en proyectos
- Algo de Experiencia
- Se reúne con los clientes
- Plantea las líneas generales del proyecto
- Reparte el trabajo entre los programadores
- Es el responsable del proyecto o de varios de sus aspectos
- Analista-programador: hace ambos papeles
- Analista funcional: Se encarga de estudiar y planificar la implementación de las funcionalidades del sistema: qué es lo que el sistema hace

Jefe de Proyecto (Sr)

- Visión global
- No programa, hace PowerPoints
- Conoce a los miembros del equipo
- Suele establecer el orden de trabajo en el proyecto
- Los grupos de trabajo están a su cargo
- Mucha experiencia

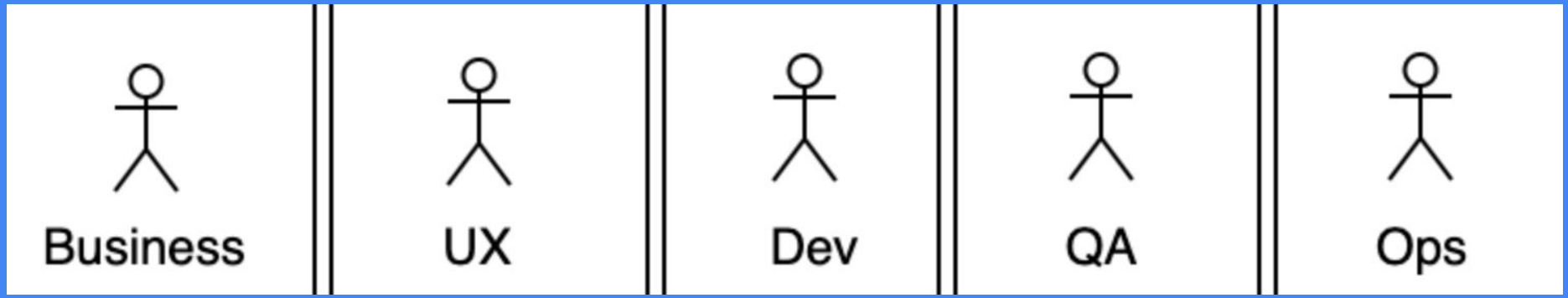
Consultor (Sr)

- Persona externa
- Viene, aconseja y se va
- Suelen ser unos máquinas

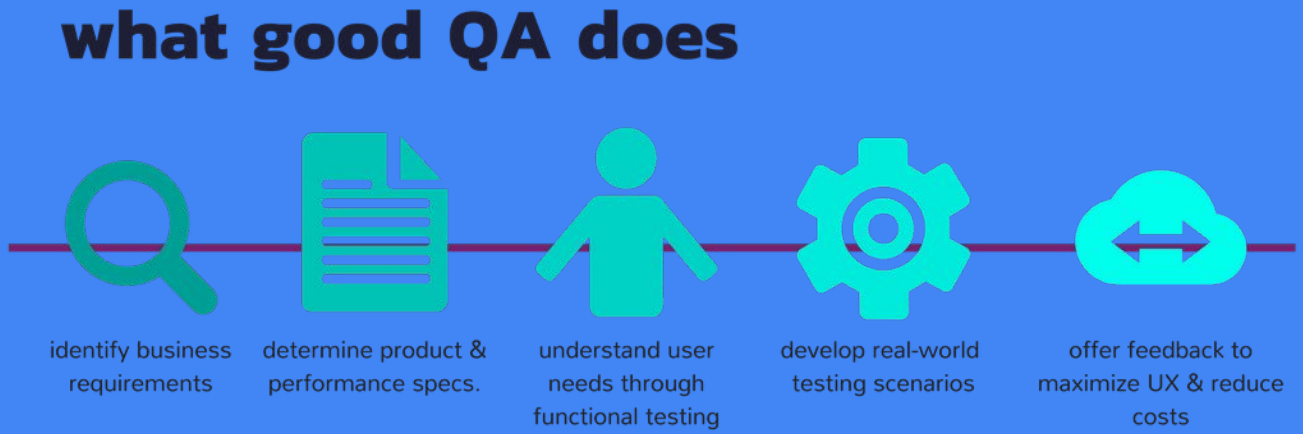
Otros Títulos

- Analista - Programador
- Jefe de Equipo
- Arquitecto
- Technical Lead
- Técnico de Sistemas / Responsable de sistemas: es el encargado de que la infraestructura funcione. Redes, PCs, Servers…
- Administrador (de la Red, de la Base de Datos…)

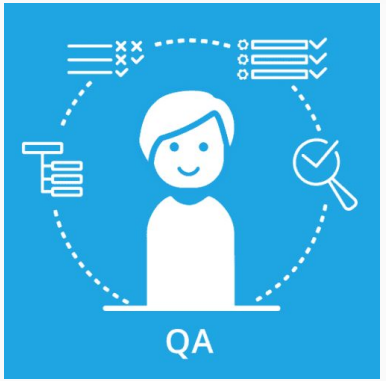
Profesiones Tangenciales al Desarrollo



¿Qué es QA?

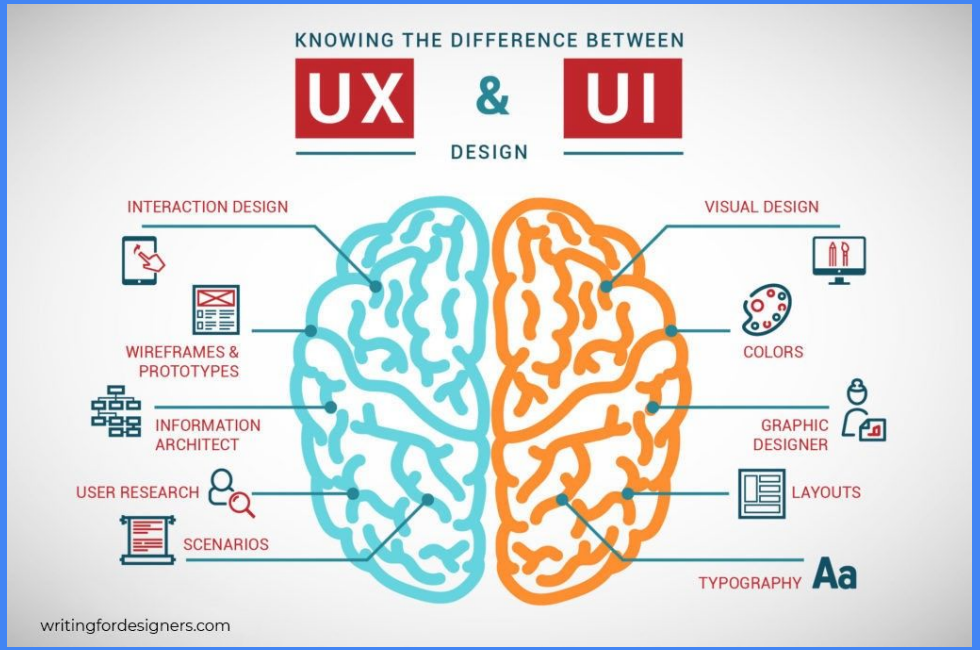

what good QA does
identify business requirements
determine product & performance specs.
understand user needs through functional testing
develop real-world testing scenarios
offer feedback to maximize UX & reduce costs

QA



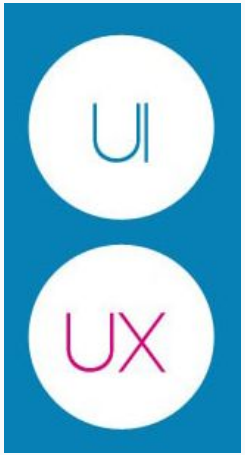
- Hace pruebas en el código
- Busca (y siempre encuentra) errores en un software
- Verifica que el código hace lo que debe hacer
- Verifica que el código no hace lo que no debe hacer
- Crea el plan de pruebas
- Existen QAs *muggles* y QA desarrolladores
- Existen los mismos niveles que para los programadores (Jr - Mid - Sr)
- Un QA no tiene que ser informático, y un informático no tiene que saber QA
- Rama en crecimiento dentro de la informática. Nuevos departamentos.

¿Qué es UX/UI?

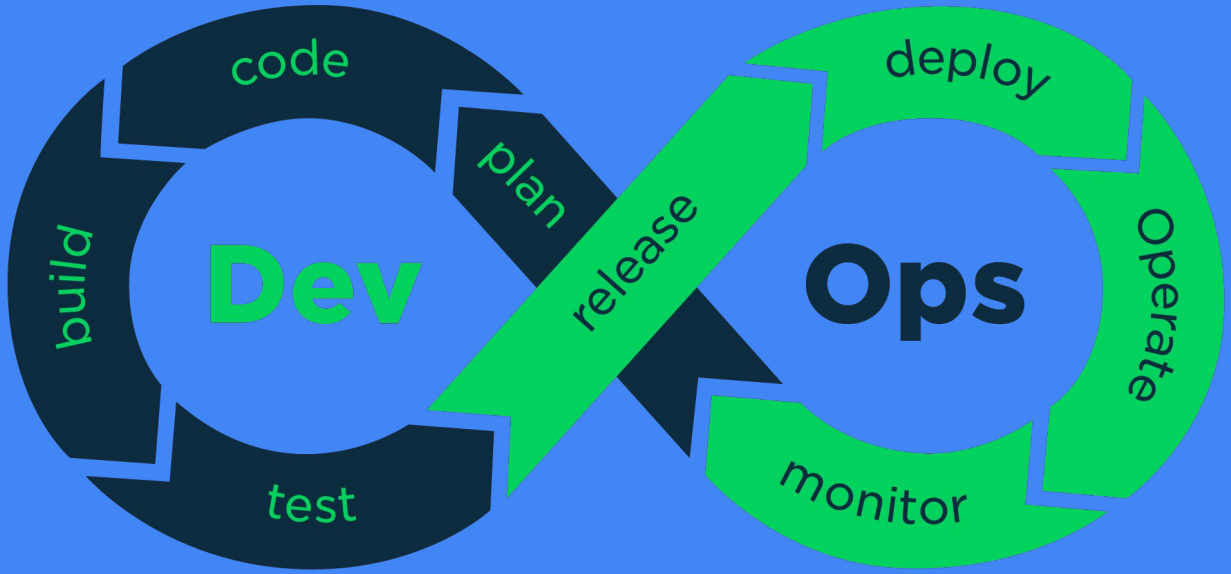

KNOWING THE DIFFERENCE BETWEEN
UX
&
UI
DESIGN
INTERACTION DESIGN
WIREFRAMES & PROTOTYPES
INFORMATION ARCHITECT
USER RESEARCH
SCENARIOS
VISUAL DESIGN
COLORS
GRAPHIC DESIGNER
LAYOUTS
TYPOGRAPHY
Aa
writingfordesigners.com

UX/UI

- User Experience: Cómo interactúa el usuario con nuestro sistema. Entender las necesidades de los usuarios.
- User Interface: Hacer el producto atractivo y visual
- Normalmente se trabaja en ambos aspectos
- Diseñar la interfaz de usuario
- Prototipar o, incluso mejor, crear el front sabiendo programación
- Es una de las partes más importantes del desarrollo
- Colores, formas, situación de la información en la pantalla, la navegación dentro de una web o app...
- Un informático no sabe UX/UI



¿Qué es DevOps?



DevOps

- Development Operations
- Es todo lo relacionado con llevar a cabo el desarrollo del software
- Intervienen muchas herramientas para automatizar o facilitar todo lo posible el desarrollo de software y su puesta en marcha
- Los objetivos principales son:
  - Integración continua (Continuous Integration): Gestionar todos los cambios que los programadores hacen sobre el código
  - Entrega continua (Continuous Delivery): Trabajar con el objetivo de liberar una versión del software cada poco tiempo