Ruta de aprendizaje para ser arquitecto de software

Elaborada por Manuel Zapata

manuelzapata.co

Áreas principales

Estas son las 4 partes en que dividiremos la ruta de aprendizaje.

Crecer como desarrollador

Conocimientos

Habilidades blandas

Amplitud técnica

Crecer como desarrollador

Para ser un buen arquitecto/a de software es fundamental tener experiencia desarrollando.

Y no solo se trata de saber desarrollar, sino también ser capaz de escribir BUEN CÓDIGO: entendible, fácil de mantener, organizado y siguiendo buenas prácticas.

Temas que te recomiendo para que seas un buen desarrollador →

- Refactorización
- Principios de diseño
- Pruebas Unitarias
- Patrones de diseño

Refactorización

Refactoring en inglés. Son un conjunto de técnicas para cambiar nuestro código, identificando "malos olores" en este (*code smells*).

Aprender a identificar estos problemas y las mejores formas de resolverlo, te permitirán mejorar muchísimo tu código.

Recursos recomendados

Refactoring, de Martin Fowler.

Principios de diseño

Los principios de diseño son un conjunto de prácticas que nos ayudan a mejorar nuestras aplicaciones. A diferencia de las técnicas de refactoring, no son tan detalladas. Algunas pueden aplicar tanto a alto nivel de arquitectura como a nivel de código.

Entre los más conocidos, están los de diseño orientado a objetos, SOLID, GRASP y STUPID.

Recursos recomendados

Clean Architecture, de Robert Martin.

Principios de Diseño SOLID.
Mini Curso Gratuito.

Programación Profesional con Objetos.
Mini Curso Gratuito.

Pruebas Unitarias

Consisten en escribir código para verificar que nuestro código funciona. Es una herramienta fundamental para que lo que hagamos sea fácil de mantener.

Para dominar este tema, es importante estudiar conceptos como TDD, cobertura de código y dobles de pruebas.

Recursos recomendados

- Test Driven Development: By Example, de Kent Beck.
- Mocks aren't Stubs, por Martin Fowler.
- Pruebas Unitarias y Test-Driven Development.

Patrones de diseño

Los patrones de diseño son soluciones creadas por expertos, a problemas recurrentes que podamos tener en nuestro código. Un ejemplo es el patrón Observer, que nos ayuda a notificarle a ciertos objetos cuando algo ocurra en un objeto de interés.

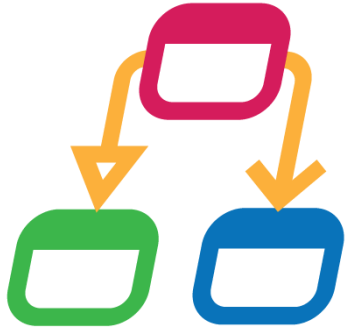
Existen muchos patrones de diseño, pero los más comunes son los 23 originales, publicados en 1994.

Recursos recomendados

Head First Design Patterns, de E. Freeman y E. Robson.

Patrones de Diseño. Refactoring.Guru.

Introducción a los Patrones de Diseño.

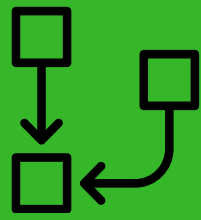


CURSO PRÁCTICO DE PATRONES DE DISEÑO

Te presento mi curso premium 100% dedicado a patrones de diseño. Puedes conocer todos los detalles aquí:

https://bit.ly/3nsLIqf

Conocimientos

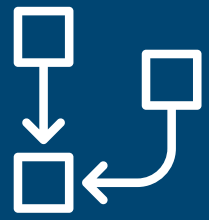


Hacer arquitectura de software implica conocer a fondo temas que suelen escaparse de día a día de un desarrollador.

Temas básicos que debes dominar →

- Atributos de calidad
- Patrones de arquitectura
- Diagramación

Atributos de calidad



Son propiedades medibles o verificables de un sistema, por las cuales un arquitecto debe velar.

Algunos atributos son: disponibilidad, rendimiento, seguridad y mantenibilidad.

También se les conoce como requerimientos no funcionales o características de arquitectura.

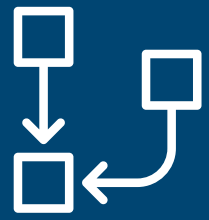
Recursos recomendados

Software Architecture in Practice, de L. Bass, P. Clements y R. Kazman.

High Scalability.

Atributos de calidad.

Patrones de arquitectura



Definen la estructura de más alto nivel de nuestro sistema, y nos permiten tomar decisiones que afectarán significativamente el desarrollo del proyecto.

Entre los más comunes tenemos capas, microservicios, microkernel y arquitecturas orientadas a eventos.

Recursos recomendados

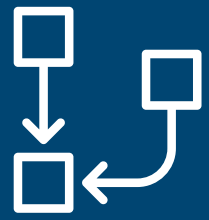
Fundamentals of Software Architecture, de M. Richards y N. Ford.

CURSO PRÁCTICO DE PATRONES DE ARQUITECTURA

Si deseas profundizar en estos patrones, mi curso premium sobre el tema puede ayudarte. Consulta los detalles:

https://bit.ly/3se1wB2

Diagramación



Los diagramas son un elemento fundamental para comunicar una arquitectura a un equipo de desarrollo, e incluso con personal no técnico.

Algunas notaciones para modelar incluyen: C4 (mi favorito), vistas 4+1, UML y ArchiMate.

Recursos recomendados

- Software Architecture for Developers Vol. 2, de Simon Brown.
- UML Distilled, de Martin Fowler.
- Modelo C4.

Habilidades blandas

Los conocimientos técnicos no son suficientes para ser un buen arquitecto de software. Las habilidades blandas (*soft skills* o *people skills*) juegan un rol fundamental. Muchas veces los proyectos y aplicaciones fracasan por aspectos no técnicos.

Habilidades importantes a dominar: negociación, comunicación, visión, liderzgo, capacidad de abstracción y mentoría.

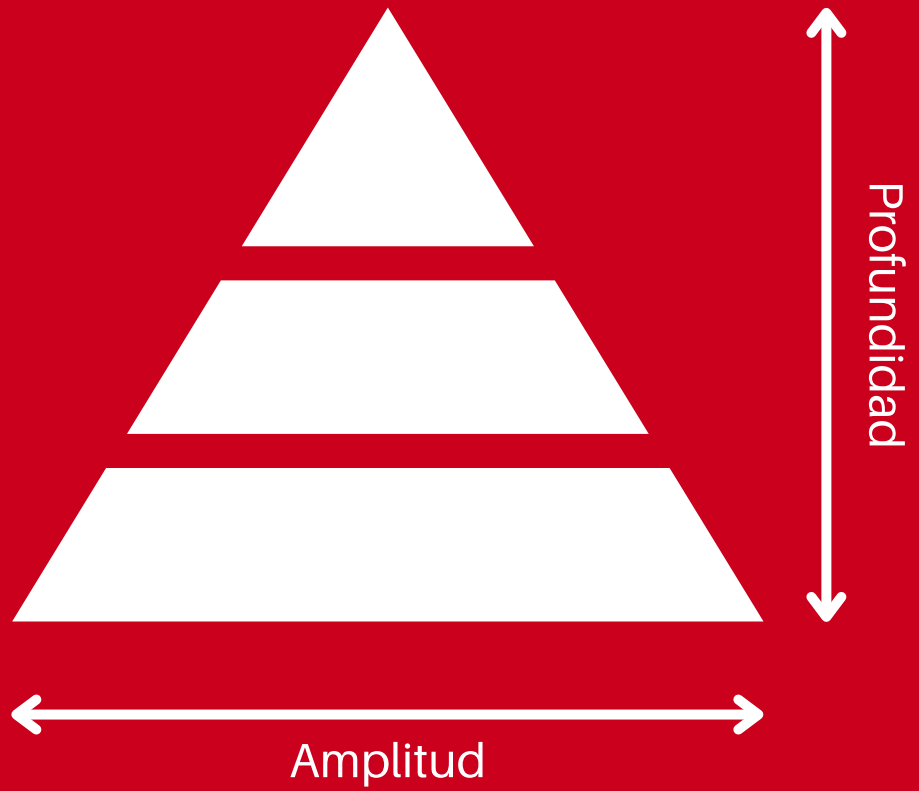
Recursos recomendados

- 12 Essential Skills for Software Architects, de D. Hendricksen.
- 97 things every software architect should know. Múltiples autores.
- 7 Soft Skills cruciales de un arquitecto de software.

Amplitud técnica

Cuando estás creciendo profesionalmente como desarrollador, necesitas PROFUNDIDAD TÉCNICA. Es decir, debes hacerte experto en una tecnología. Esto te permitirá ganar mejor y volverte un referente.

En arquitectura de software, es crucial la amplitud técnica. Más que ser un experto en una sola cosa, necesitas conocer de muchos temas, a menor detalle, para poder tomar decisiones.



Áreas para crear amplitud técnica

Estos son algunos temas importantes que deberían estar en el conocimiento de un arquitecto:

DevOps. Actividades como integración y entrega continua son vitales para habilitar ciertas arquitecturas y mantener buena calidad en el código.

Computación en la nube. Plataformas *cloud* como AWS, Azure y GCP facilitan ciertos atributos de calidad y son una opción que se debe evaluar.

Entendimiento de tecnologías y tendencias. Conocer opciones de almacenamiento, frameworks, clientes y tendencias en cada área es vital para estar actualizado.

Conocimiento de la industria. Conocer el sector donde se mueve tu empresa es vital para garantizar que la arquitectura sigue siendo relevante.

Espero que esta guía haya sido de ayuda. Cualquier comentario me lo puedes enviar al correo contacto@manuelzapata.co.

Saludos,

Manuel.