ARQUITECTURAS MODERNAS

CONCLUSIÓN DEL CURSO

Daniel Blanco Calviño

ARQUITECTURAS CONVENCIONALES

- **Puntos clave.**
  - Monolitos y centradas en los datos.
  - Sin capas o tres capas típica. Presentación, lógica de negocio y acceso a datos.

Desarrollo rápido al inicio y fácil adaptación.

Difícil de mantener y evolucionar a largo plazo.

Proyectos con corto tiempo de vida.

DOMAIN DRIVEN DESIGN

- **Puntos clave.**
  - Centrado en el dominio del problema.
  - Lenguaje ubicuo, contextos acotados y mapeo de contextos.
  - Surge una nueva capa, la de dominio. Entidades + Servicios de Dominio.

Desarrollo mucho más rápido a largo plazo. Mantenible y evolucionable.

Lento al principio. Difícil adaptación. Necesario tener expertos de dominio.

Proyectos con largo tiempo de vida y con problemas con un dominio rico.

COMMAND QUERY RESPONSIBILITY SEGREGATION

- **Puntos clave.**
  - Tratamiento independiente de lecturas y escrituras.
  - Posibilidad de tener dos Bases de Datos.
  - Mejora con Event Sourcing.

Optimización de lecturas y escrituras.

Gran complejidad y sincronización.

Proyectos dónde el rendimiento sea algo crítico.

EVENT SOURCING

- **Puntos clave.**
  - Almacenamiento de eventos en lugar de estado actual.
  - El estado actual se consigue ejecutando todos los eventos.
  - Muy útil con CQRS.

Trazabilidad del estado del sistema en el tiempo.

Eficiencia en lecturas y mayor complejidad.

Proyectos dónde se use CQRS o se necesite un log de las acciones.

EVENT DRIVEN PROGRAMMING

- **Puntos clave.**
  - Interacción mediante eventos en lugar de llamadas directas a métodos.
  - Desacople entre la parte productora del evento y la consumidora.

Flexibilidad para añadir más consumidores del evento sin modificar nada.

Mayor complejidad general.

Cuando en el futuro se necesiten más consumidores. Procesos asíncronos.

MICROSERVICIOS

- **Puntos clave.**
  - División del dominio en subdominios. Cada uno será un microservicio.
  - Viven de forma independiente.
  - Agnósticos de la tecnología.

Sistemas más manejables y escalado independiente en cada microservicio.

Cooperación entre equipos e interfaz de usuario.

Sistemas en los que necesitemos una alta disponibilidad.

ARQUITECTURA HEXAGONAL

- **Puntos clave.**
  - Aislar aún más el core de nuestro sistema.
  - Puertos y adaptadores.

Fácil de testear y muy flexible.

Arquitectura pesada y confusa al aplicar con frameworks.

Incertidumbre en las tecnologías que se van a utilizar.

ARQUITECTURA MICROKERNEL

- **Puntos clave.**
  - Componente con funcionalidad básica, microkernel.
  - Componentes que ofrecen la funcionalidad, plug-ins.

Flexibilidad y personalización de nuestro sistema.

Necesidad de un gran análisis previo. No demasiado apto para web.

Sistemas en los que el concepto de plug-in sea de utilidad.

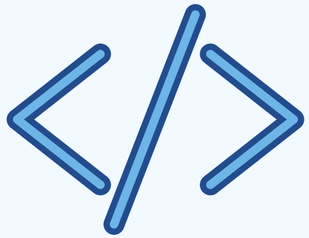
EL SOFTWARE NO ES UNA CIENCIA EXACTA

- El mayor profesor es la experiencia.
  - Prueba y error.

- **La arquitectura se debe adaptar al problema y al equipo**, no al revés.
  - No sigas al 100% la teoría aunque veas que no se ajusta del todo bien a tu caso particular. Toma los detalles que sí lo hagan.

- Lo más importante es ser **flexible y estar abierto al cambio** cuando sea necesario.

FIN DEL CURSO ARQUITECTURAS MODERNAS



Daniel Blanco Calviño

VALORA Y COMPARTE!

- Las valoraciones me ayudarán mucho a seguir creando cursos y a mejorar el material con el que trabajo.
- Si te ha gustado el curso, compártelo con otras personas a las que les pueda interesar!

¿TIENES DUDAS?

Pregunta, ¡sin miedo!

Estaré atento en **esta plataforma, en LinkedIn y en Twitter** para intentar resolver todas tus dudas!

Si tienes alguna sugerencia sobre el curso, o crees que me he equivocado en alguna cuestión, coméntamelo. De esta forma podré corregir el contenido y ayudará a muchas más personas.

Daniel Blanco Calviño

@DanielBlancoSWE