

Arquitecturas Software. Patrones y Arquitecturas Modernas

Escuela Técnica Superior de Ingeniería Informática

Grado en Informática

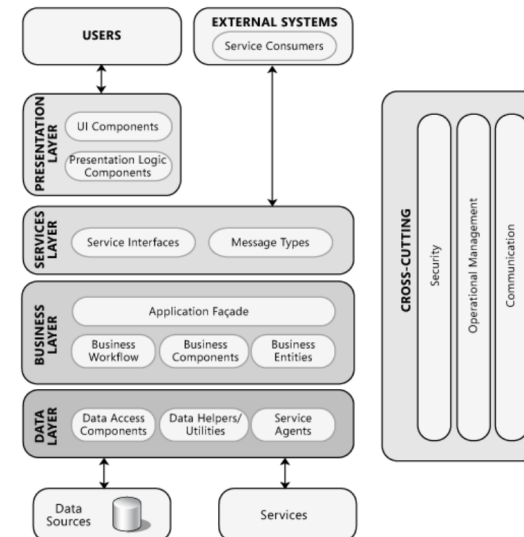
Contenido

- Arquitectura Software
- Patrones Arquitectónicos
- Arquitecturas Software Modernas
 - Características
 - 3-Niveles ...N-Niveles
 - Domain Centric Architecture (DDD)
 - Microservicios

Arquitectura Software

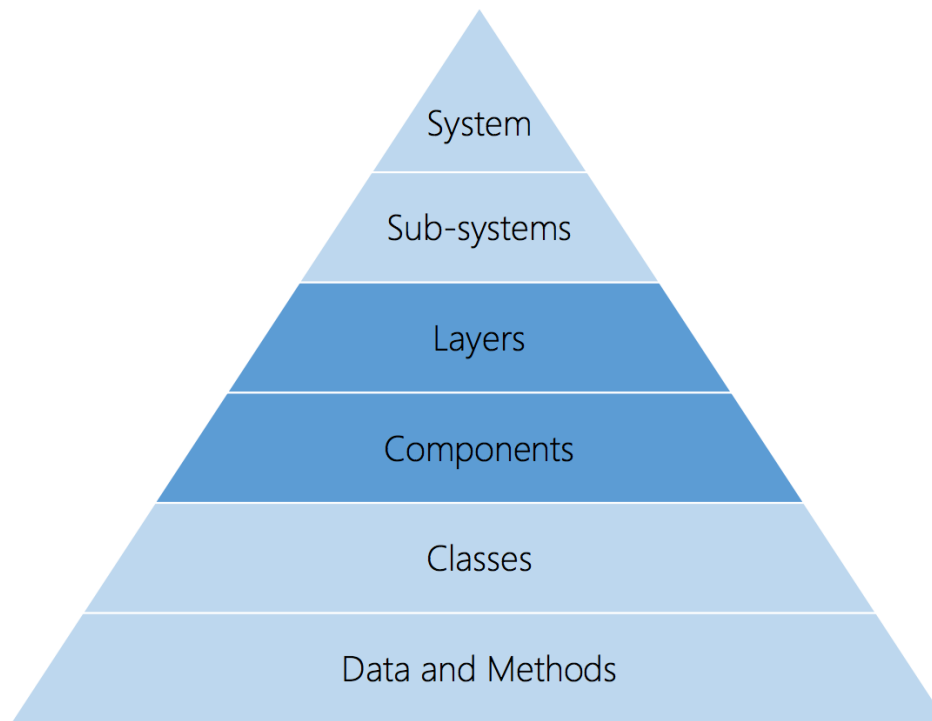
■ La **arquitectura de software** de un sistema es un conjunto de estructuras necesarias para razonar sobre el sistema, que comprenden elementos de software, las relaciones entre ellos, y las propiedades de ambos.

- Alto Nivel
- Estructura
- Niveles
- Componentes
- Relaciones



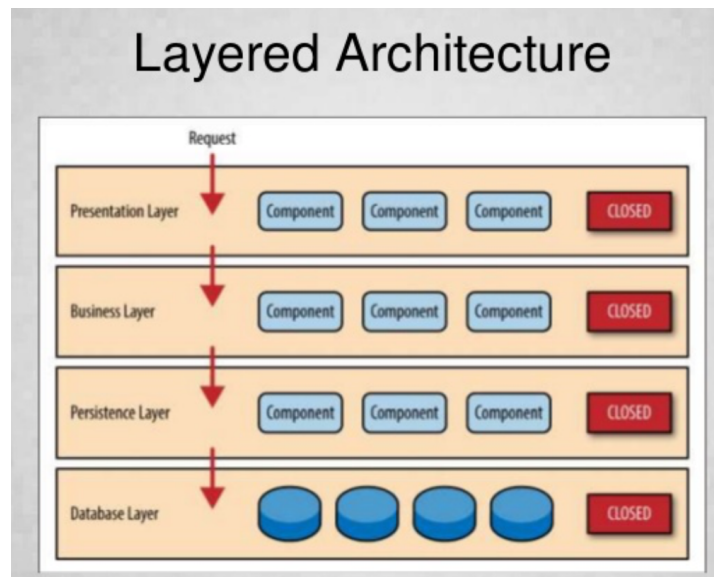
Arquitectura Software

Levels of Architectural Abstraction



Patrones Arquitectónicos

- Un **patrón arquitectónico** es un concepto que resuelve y esboza algunos **elementos esenciales** de una **arquitectura de software**.



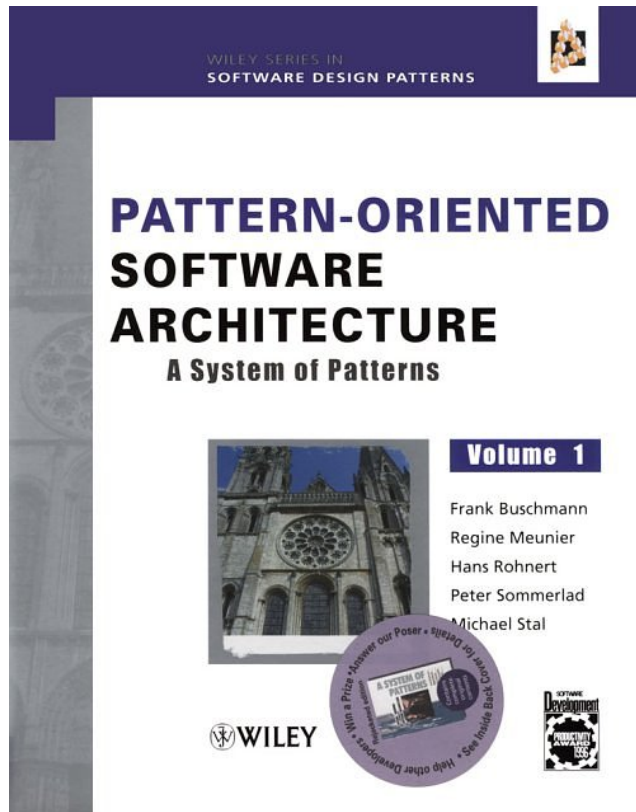
Patrones Arquitectónicos

- Los **patrones arquitectónicos** son un concepto clave en el campo de la **arquitectura de software**:
 - Ofrecen soluciones a los problemas arquitectónicos,
 - Ayudan a documentar las decisiones de diseño arquitectónico,
 - Facilitan la comunicación entre las partes interesadas a través de un vocabulario común,
 - Describen los atributos de calidad de un sistema de software como restricciones a cumplir.

Patrones Arquitectónicos

- Los **patrones arquitectónicos** representan la **organización estructural** básica de los sistemas software mediante:
 - Un conjunto de subsistemas predefinidos
 - Especifican responsabilidades de cada uno de los subsistemas y componentes
 - Reglas y Guías para organizar las relaciones entre los subsistemas

Patrones Arquitectónicos



- P. Avgeriou, U. Zdun, Architectural Patterns Revisited – A Pattern Language, the *European Pattern Languages of Programming* (EuroPLOP) 6–10 July 2005, Irsee, Germany.

Patrones Arquitectónicos

- Una **Vista Arquitectónica** es una representación de un sistema desde la perspectiva de un conjunto relacionado de aspectos.
- Un **patrón arquitectónico** define los tipos de elementos y relaciones que funcionan en conjunto para resolver un problema en particular desde una cierta **perspectiva/vista**.
- Vistas
 - Componentes
 - Conectores
 - Módulo
 - Distribución

Patrones Arquitectónicos. Vistas

1. La Vista **Layered** trata de cómo el sistema, como una entidad heterogénea compleja, se puede descomponer en partes que interactúan.
 - ▣ LAYERS
 - ▣ INDIRECTION LAYER (una variante de este patrón se llama “Virtual Machine”)
2. La Vista **Data Flow** trata con streams de datos que son sucesivamente procesados o transformados por componentes.
 - ▣ BATCH SEQUENTIAL
 - ▣ PIPES AND FILTERS
3. La Vista **Data-centered** es apropiada cuando se pretende que múltiples componentes accedan a un repositorio de datos central.
 - ▣ SHARED REPOSITORY
 - ▣ ACTIVE REPOSITORY
 - ▣ BLACKBOARD

Patrones Arquitectónicos. Vistas

4. La Vista Adaptation se ocupa de cómo el sistema se adapta durante la evolución.
 - ▣ MICROKERNEL
 - ▣ REFLECTION
 - ▣ INTERCEPTOR
5. La Vista Language Extension tiene que ver con cómo los sistemas ofrecen una capa de abstracción de la infraestructura de computación.
 - ▣ INTERPRETER
 - ▣ VIRTUAL MACHINE
 - ▣ RULE-BASED SYSTEM

Patrones Arquitectónicos. Vistas

6. La Vista User Interaction muestra la estructura en tiempo de ejecución de componentes que ofrecen una interfaz de usuario.
 - ❑ MODEL-VIEW-CONTROLLER
 - ❑ PRESENTATION-ABSTRACTION-CONTROL
 - ❑ C2
7. La Vista Component Interaction se centra en cómo componentes individuales intercambian mensajes pero manteniendo su autonomía.
 - ❑ EXPLICIT INVOCATION (llamado “communicating processes”)
 - ❑ IMPLICIT INVOCATION (llamado “event systems”)
 - ❑ CLIENT-SERVER
 - ❑ PEER-TO-PEER
 - ❑ PUBLISH-SUBSCRIBE (llamado “publisher-subscriber”)
8. La Vista Distribution trata aspectos sobre cómo diseminar/distribuir componentes en un entorno de red.
 - ❑ BROKER
 - ❑ REMOTE PROCEDURE CALLS (llamado “distributed objects”)
 - ❑ MESSAGE QUEUING (llamado “messaging”)

Arquitecturas Software Modernas

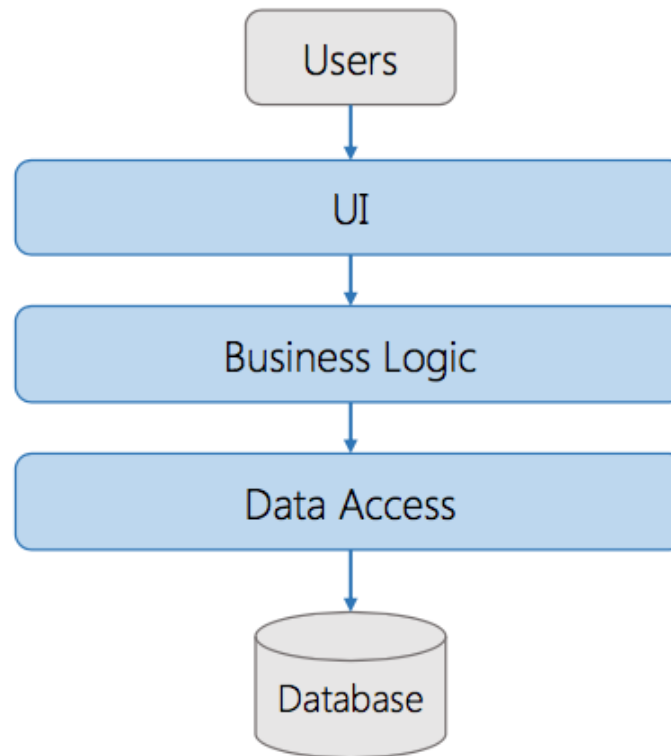
Incluyen, pero no están limitadas, las siguientes características:

- Son Orientadas a Servicios.
- Son Distribuidas y Altamente Escalables bajo demanda.
- Ofrecen APIs bien definidas.
- Se construyen para ser “cambiadas” en vez de “perdurar para siempre”.
- Proporcionan una clara separación de aspectos.

Arquitecturas Software Modernas

- Proporcionan un particionado de los datos que encaja con las necesidades arquitectónicas y no es monolítico (no almacena todos los datos en una única BD). Pueden existir muchos almacenes de datos diferentes (Relacional, NoSQL, in-memory, y muchos más).
- Son débilmente acopladas.
- Integran y se integran con gran cantidad de sistemas, aplicaciones, colas de mensajes, etc. Muchos son open-source.
- Ejecutan procesos síncronos y asíncronos.
- Se implementan usando una variedad amplia de tecnologías distintas.
- Se integran con servicios de la Computación en la Nube.

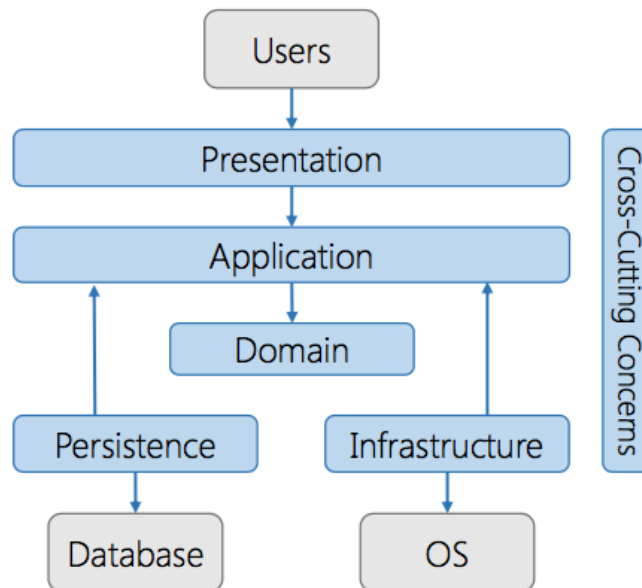
3-Niveles...N-Niveles



→ Dependency

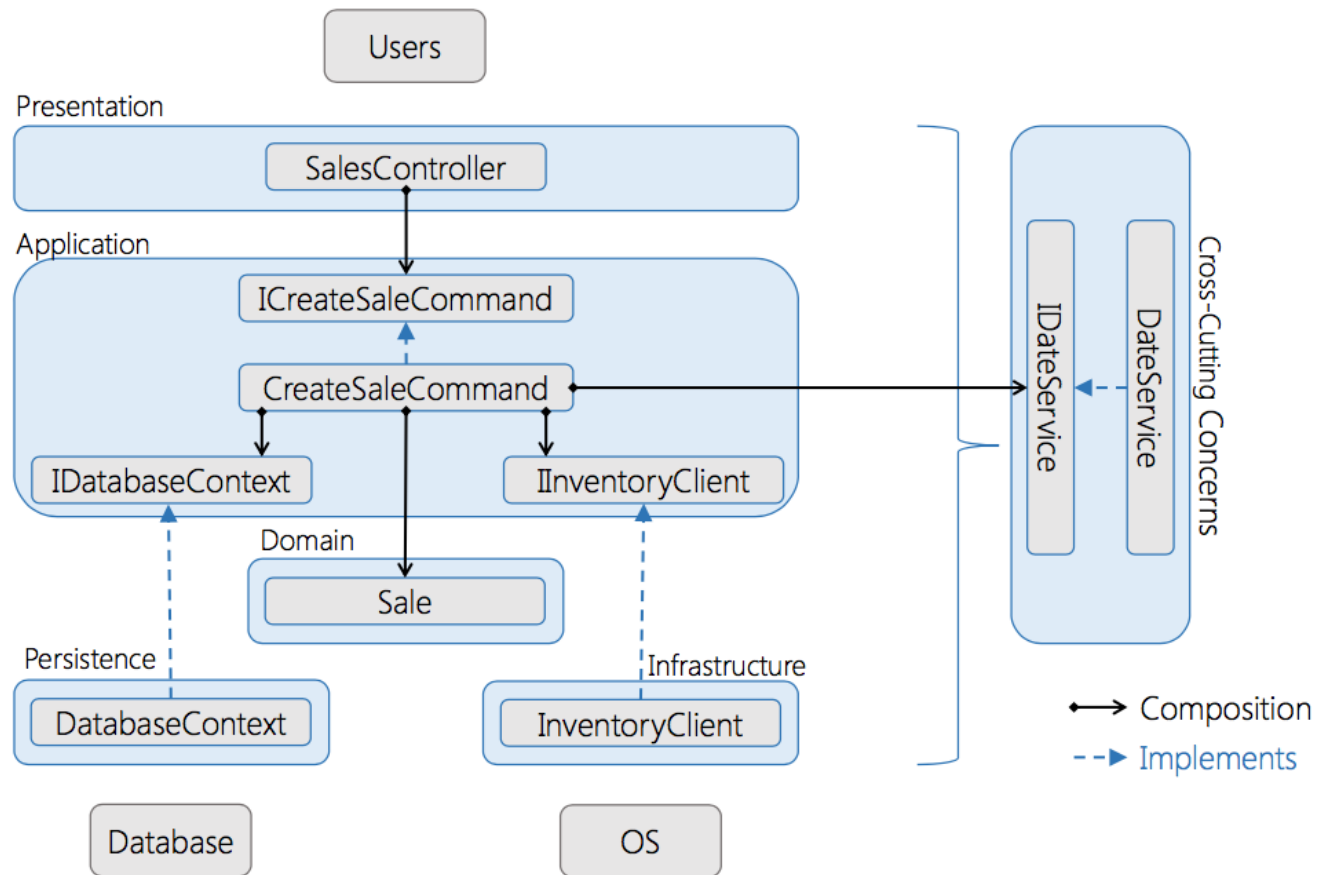
3-Niveles...N-Niveles

Modern 4-Layer Architecture



→ Dependency

3-Niveles...N-Niveles

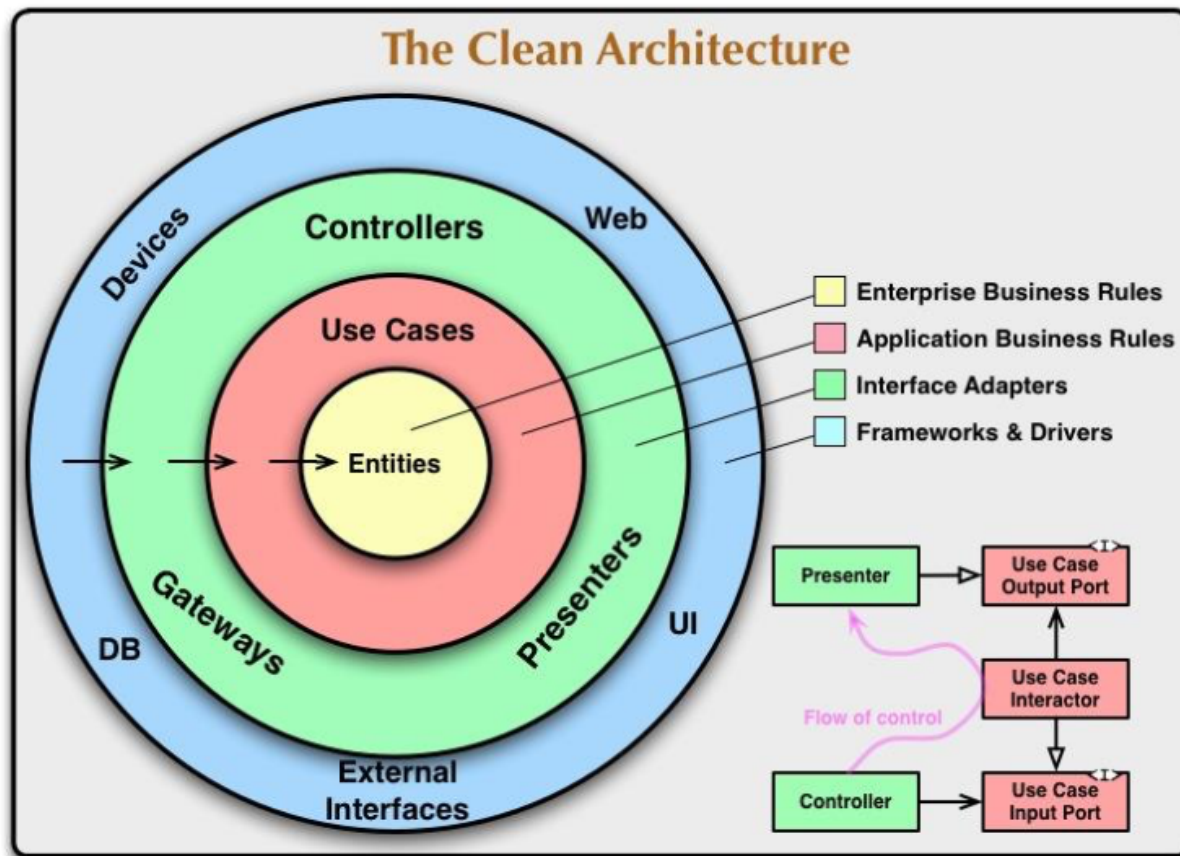


Domain Centric Architecture (DDD)

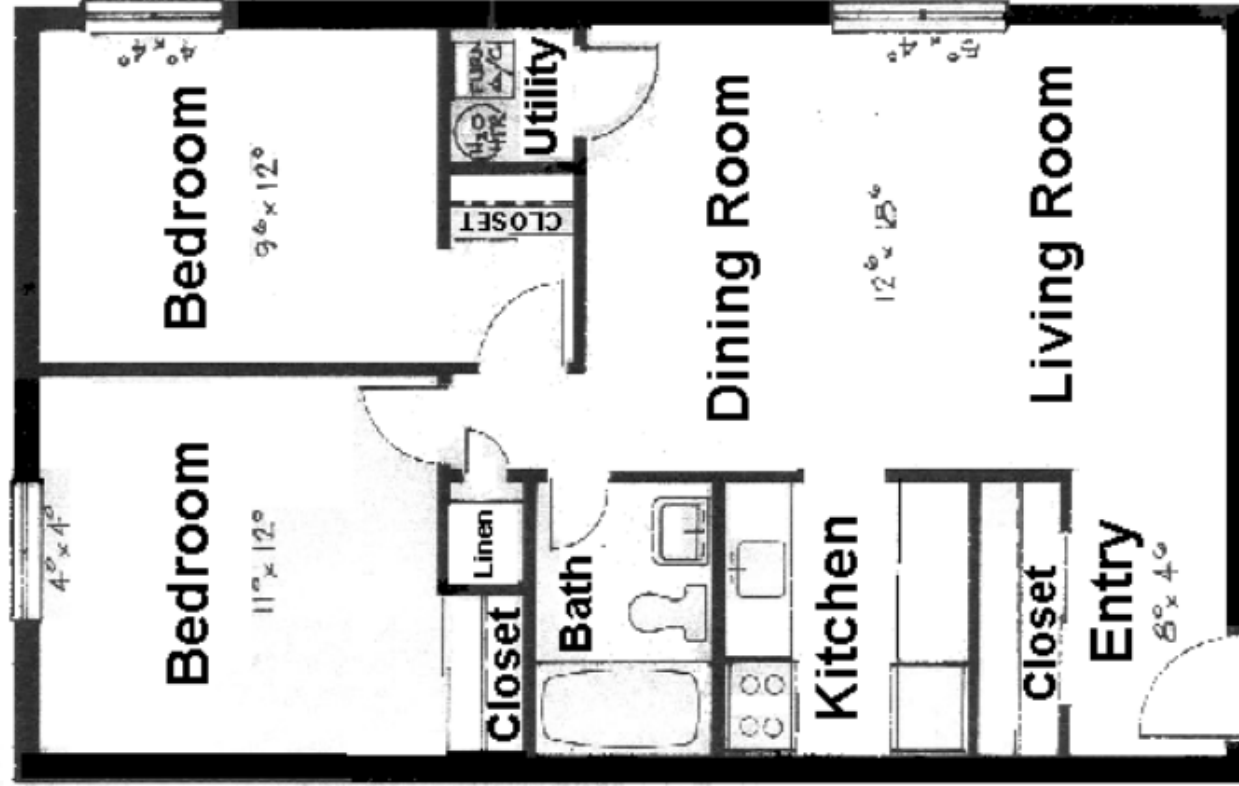
Database- vs. Domain-centric Architecture



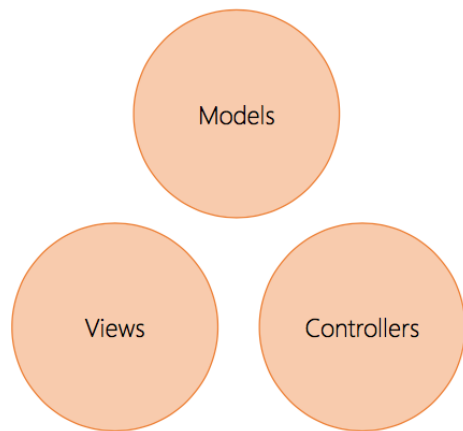
Domain Centric Architecture (DDD)



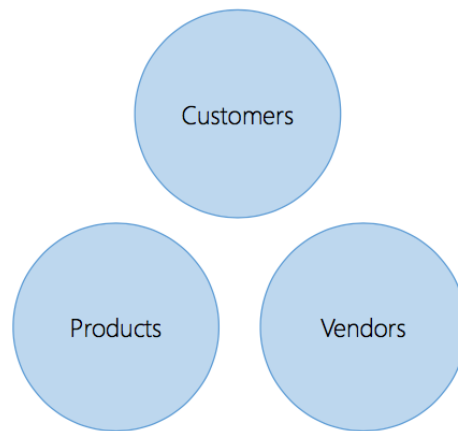
Organización Funcional



Organización Funcional



VS



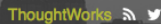
- 📁 Content
- 📁 Controllers
- 📁 Models
- 📁 Scripts
- 📁 Views

vs

- 📁 Customers
- 📁 Employees
- 📁 Products
- 📁 Sales
- 📁 Vendors

Microservicios

MARTIN FOWLER.COM

Intro Videos Design Agile Refactoring FAQ About Me All Sections 

Microservices

a definition of this new architectural term

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014



James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James'

interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.



Martin Fowler

Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the

problem of how to componentize software systems, having heard more vague claims than he's happy with. He hopes that microservices will live up to the early promise its advocates have found.

Contents

- Characteristics of a Microservice Architecture
 - Componentization via Services
 - Organized around Business Capabilities
 - Products not Projects
 - Smart endpoints and dumb pipes
 - Decentralized Governance
 - Decentralized Data Management
 - Infrastructure Automation
 - Design for failure
 - Evolutionary Design

Are Microservices the Future?

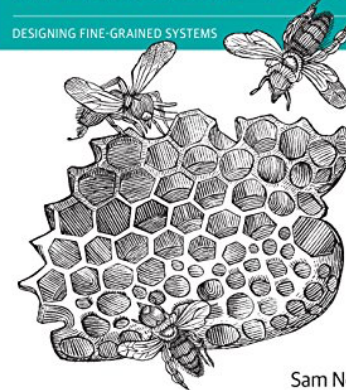
Sidebars

- How big is a microservice?
- Microservices and SOA
- Many languages, many options
- Battle-tested standards and enforced standards
- Make it easy to do the right thing
- The circuit breaker and production ready code
- Synchronous calls considered harmful

O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



Sam Newman

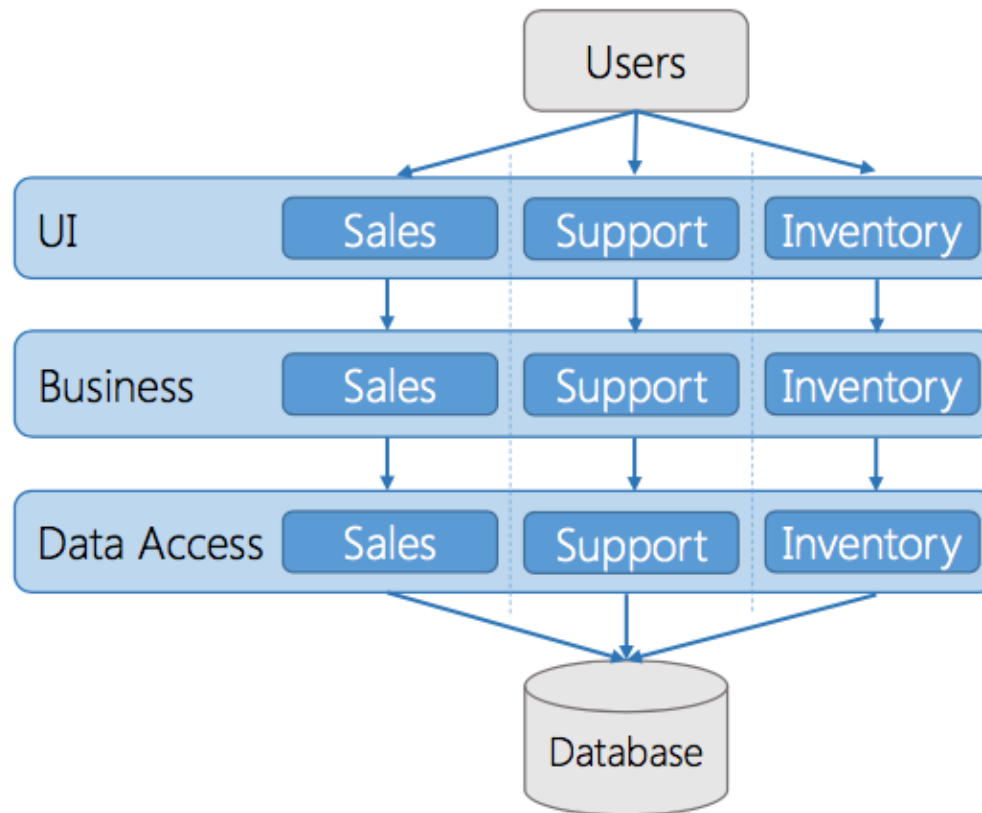
<http://microservices.io>

<https://martinfowler.com/articles/microservices.html>

Microservicios

- El **estilo arquitectónico** de los **microservicios** es una aproximación para el desarrollo de una aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose entre ellos mediante mecanismos lightweight, normalmente como una API de recursos HTTP.
- Estos servicios se construyen alrededor de capacidades de negocio y se despliegan de manera independiente mediante herramientas de despliegue automático.
- Hay muy poca gestión centralizada en este tipo de servicios. Los servicios pueden implementarse en diferentes lenguajes de programación y pueden usar tecnologías de almacenamiento de datos distintas.

Microservicios



Microservicios

Problem Domain

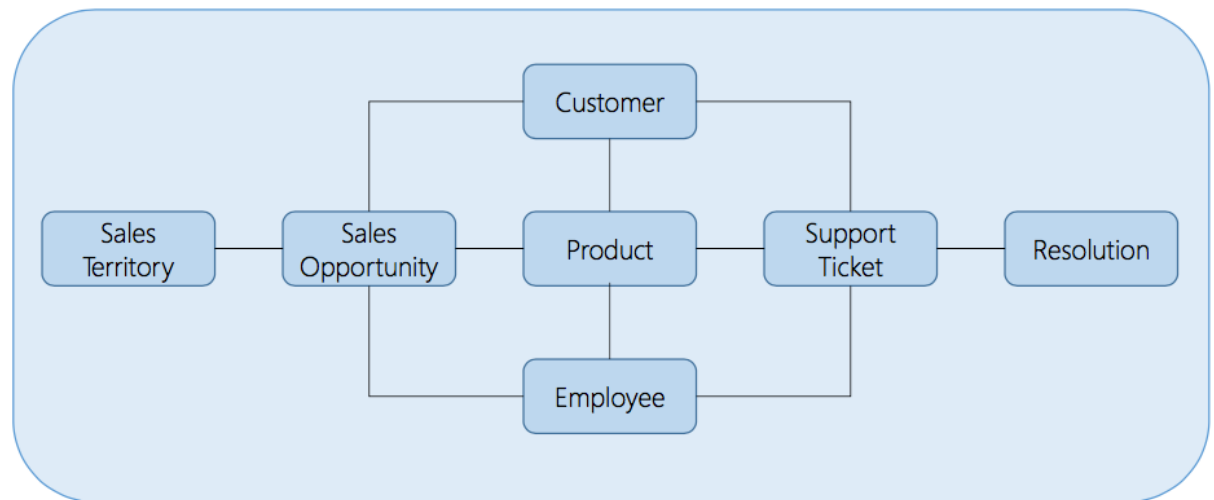
Sales

Sales Opportunity
Contact
Sales Person
Product
Sales Territory

Support

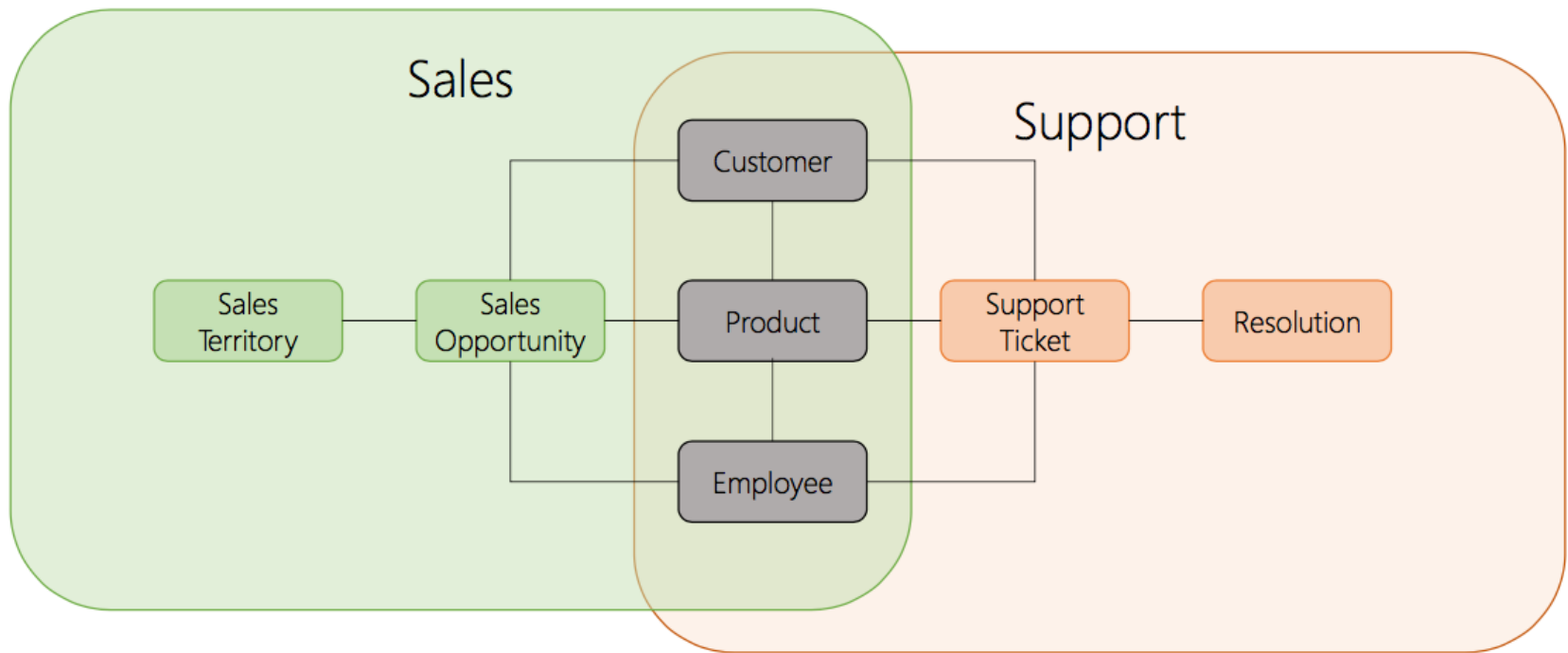
Support Ticket
Customer
Support Person
Product
Resolution

Modelo de Dominio



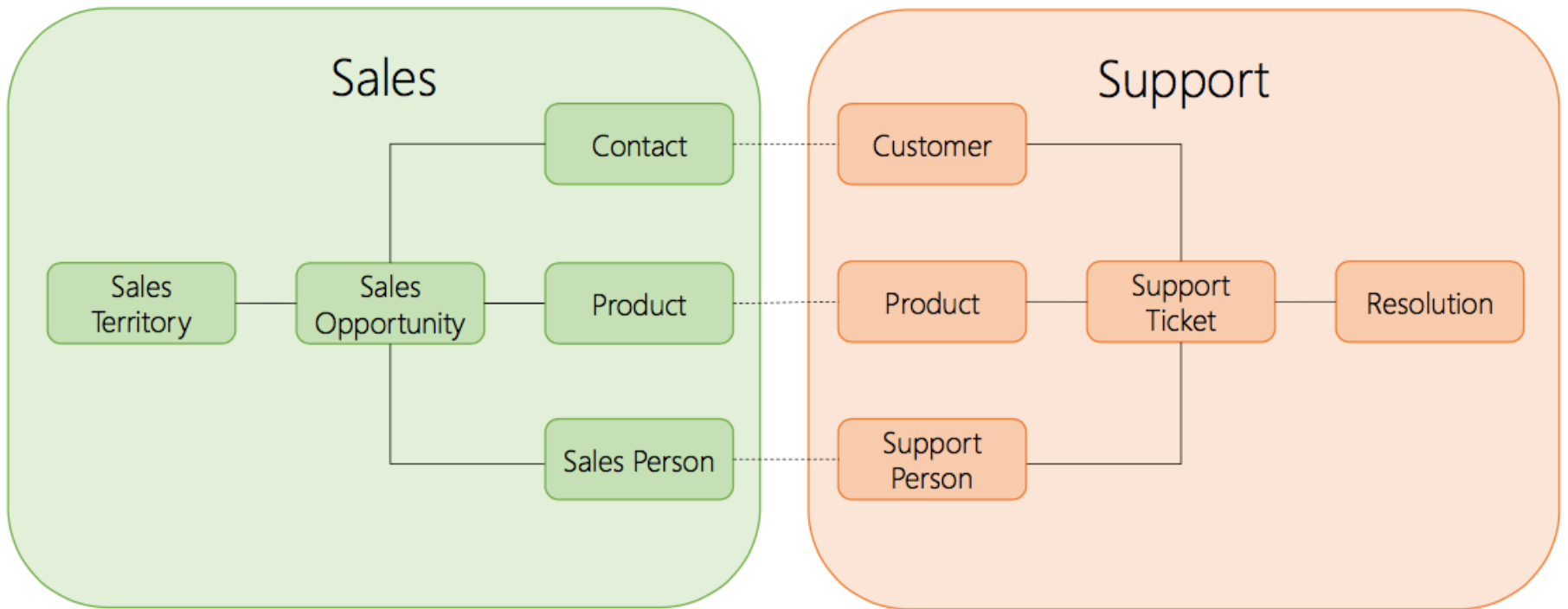
Microservicios

Contextos Encerrados



Microservicios

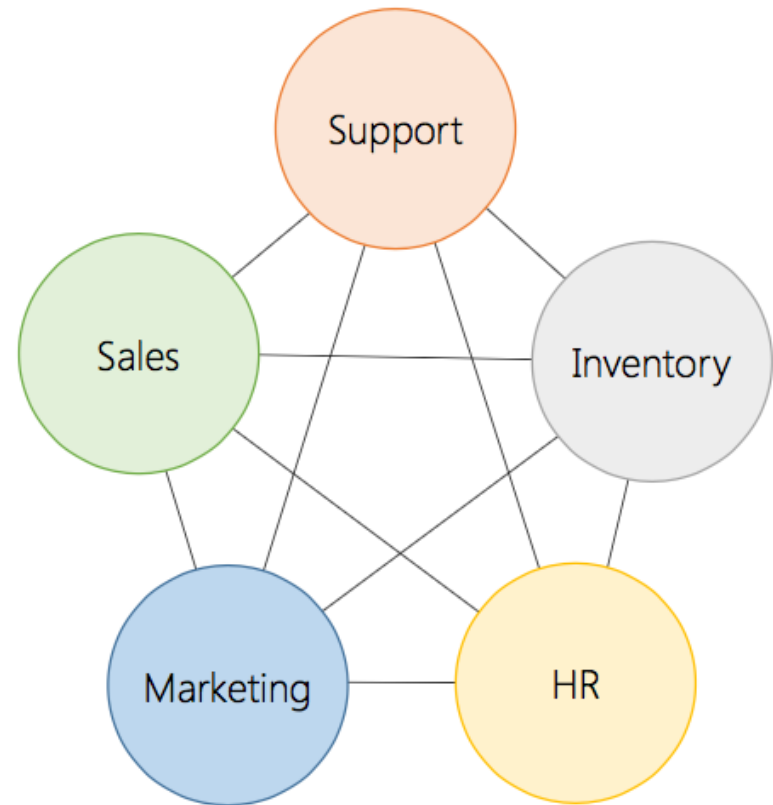
Contextos Encerrados



Microservicios

MICROSERVICIOS

Subdividir el Sistema
Contextos Encerrados
Equipos Pequeños

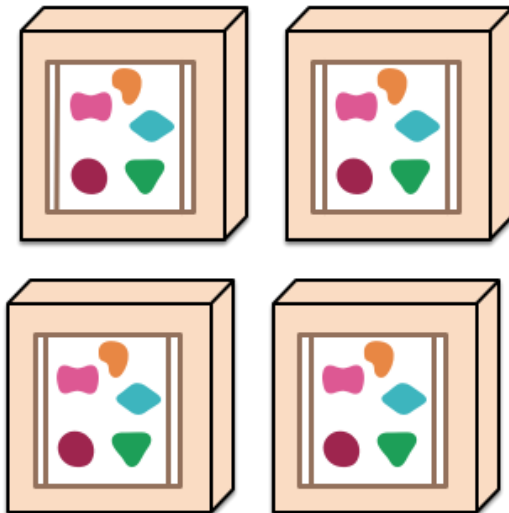


Monolitos y Microservicios

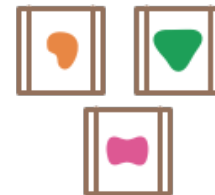
A monolithic application puts all its functionality into a single process...



... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.

