



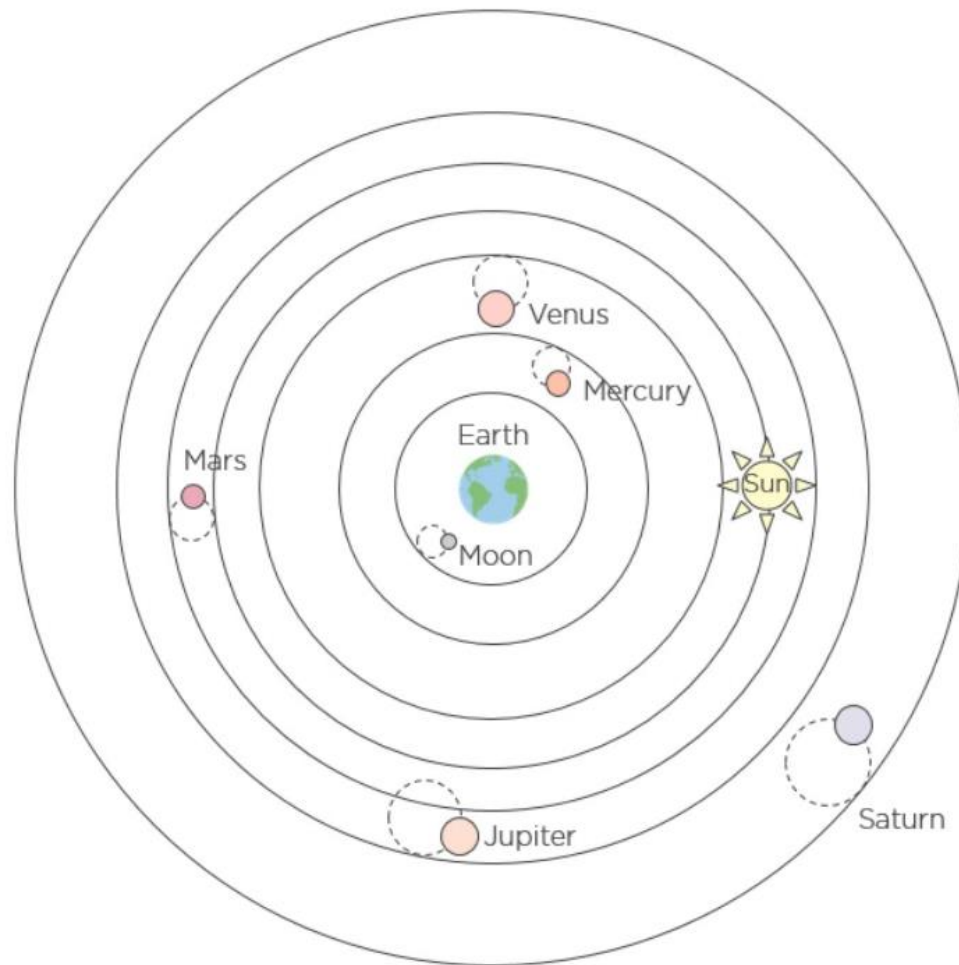
Проектирование ПО



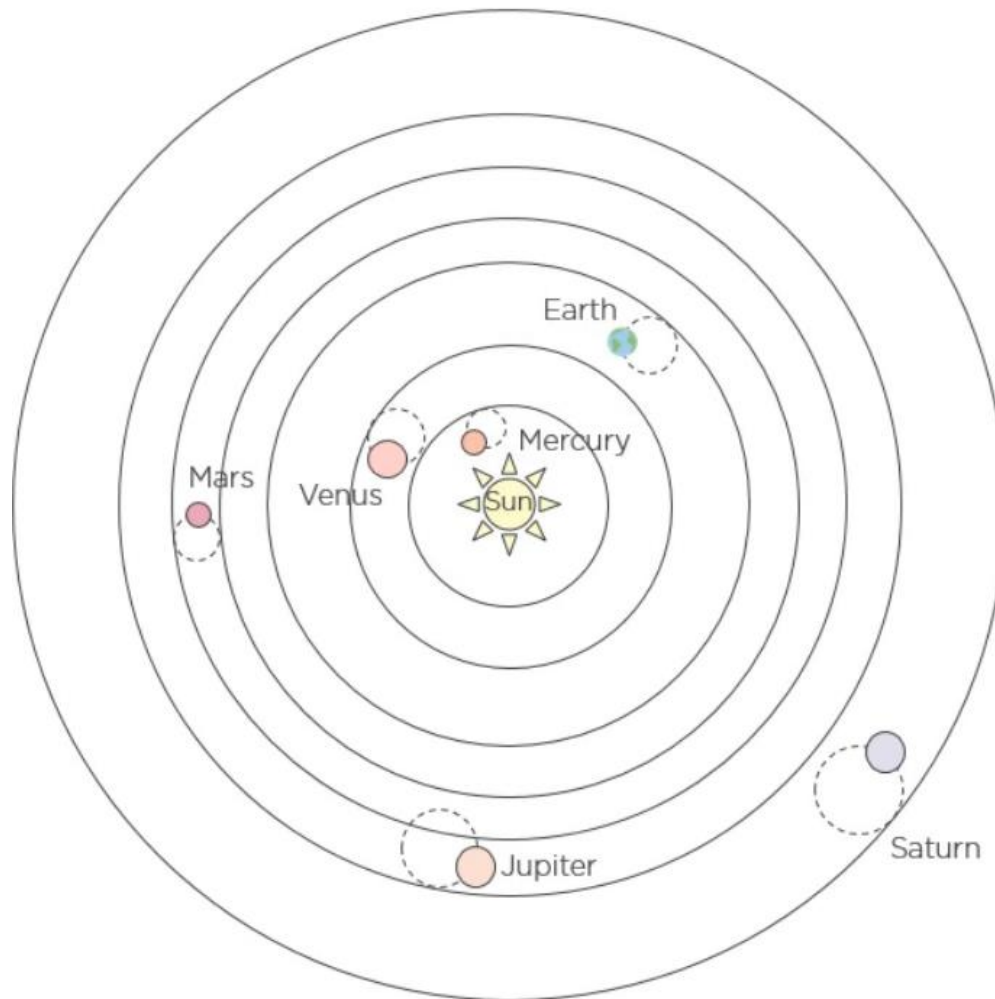
Главная цель проектирования

Проектирование (дизайн) — это процесс преобразования характеристик программного продукта таких как гибкость, расширяемость, выполнимость и безопасность в структурированное решение, отвечающее техническим и бизнес требованиям.

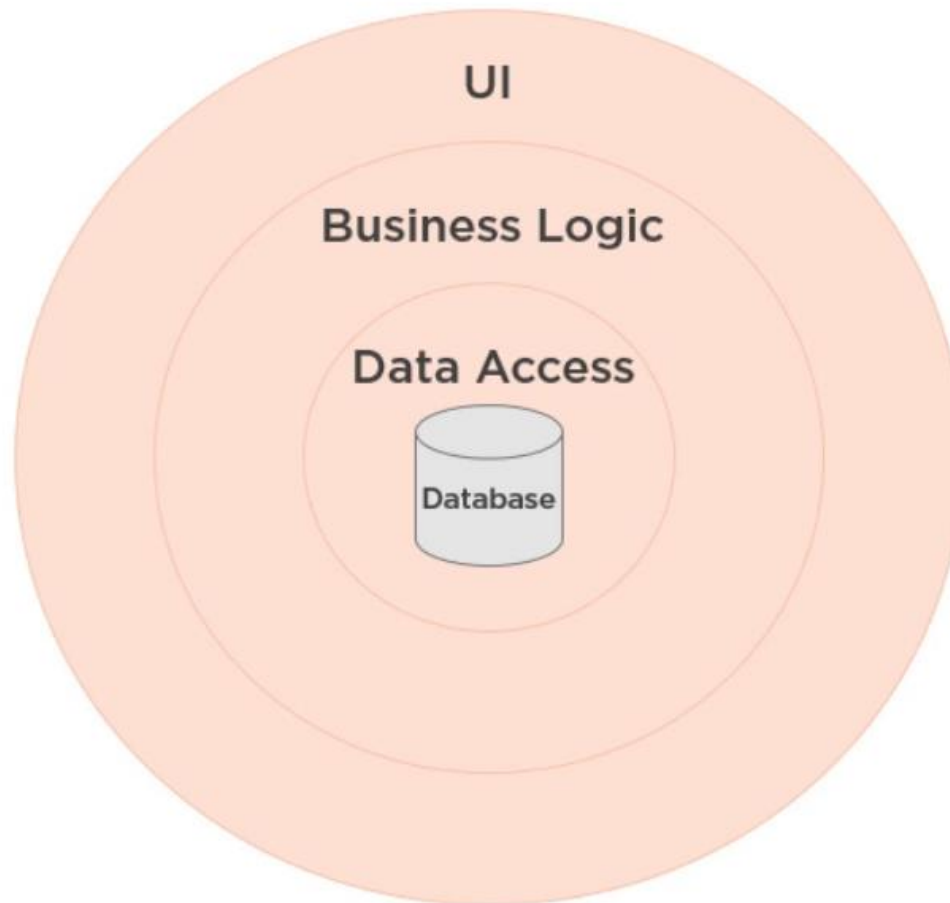
Объект архитектуры



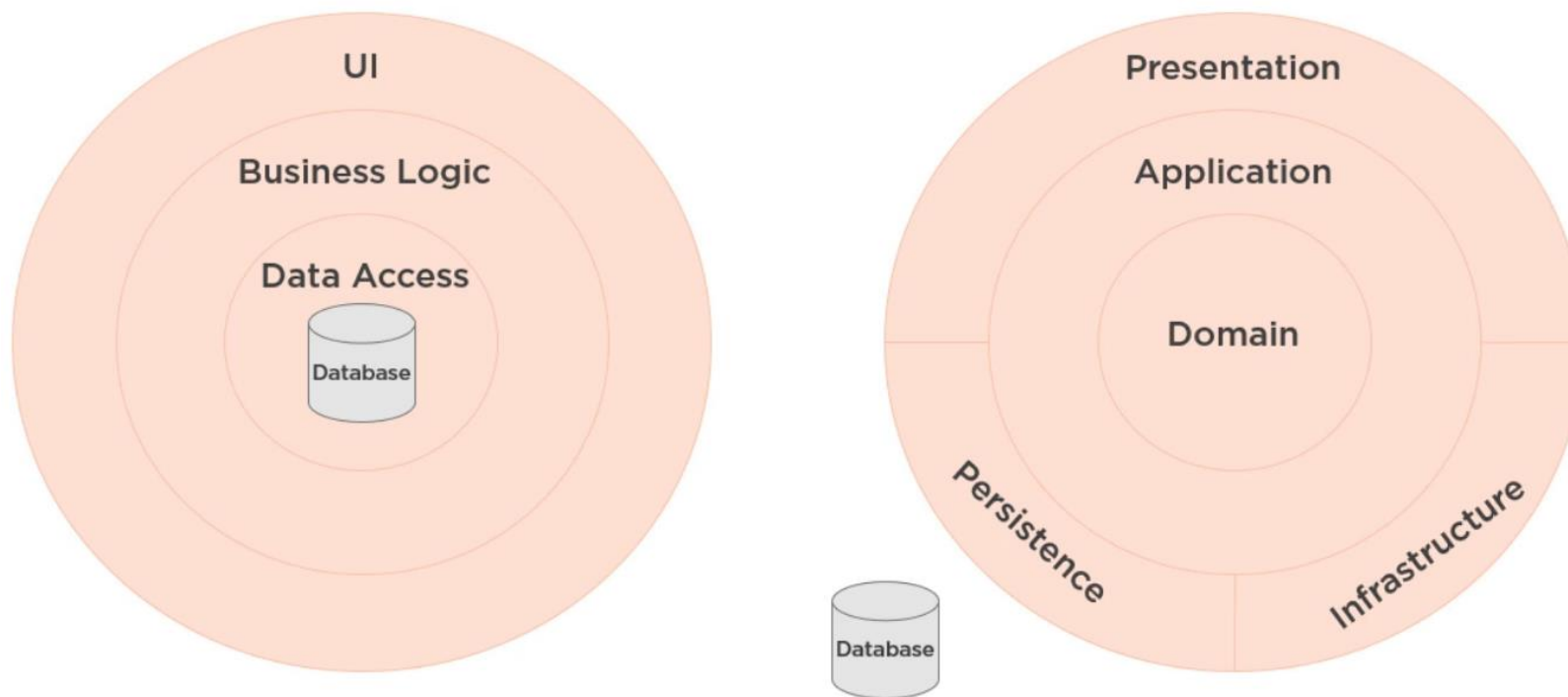
Объект архитектуры



Трёхслойная архитектура



Данные vs Домен



Важное vs второстепенное

- Размер помещений важен
- Удобство важно



Важное vs второстепенное

- Размер помещений важен
- Удобство важно
- Материалы второстепенны
- Убранство второстепенно



Важное vs второстепенное

- Предметная область важна
- Сценарии использования важны



Важное vs второстепенное

- Предметная область важна
- Сценарии использования важны
- Представление второстепенно
- Хранение второстепенно



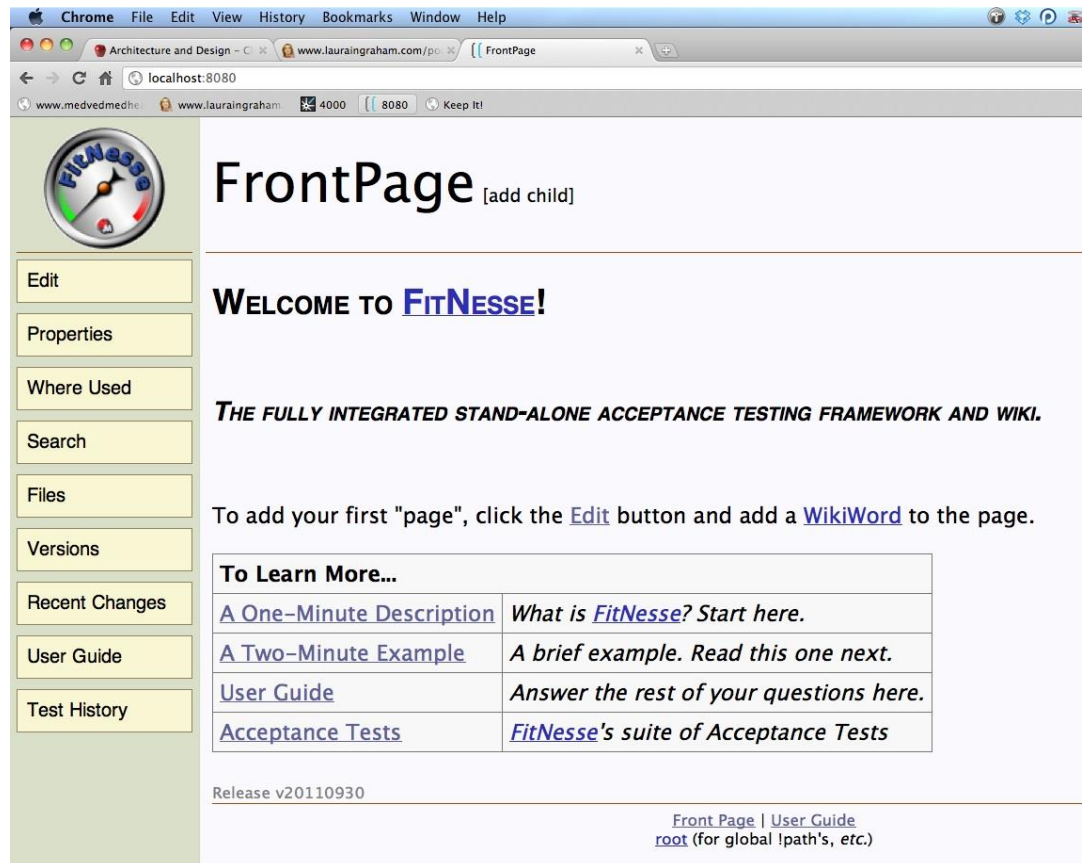


Чистая архитектура

Robert Martin (Clean Code, 2011)

Хорошая архитектура увеличивает число не принятых решений.

Архитектура Fitnesse

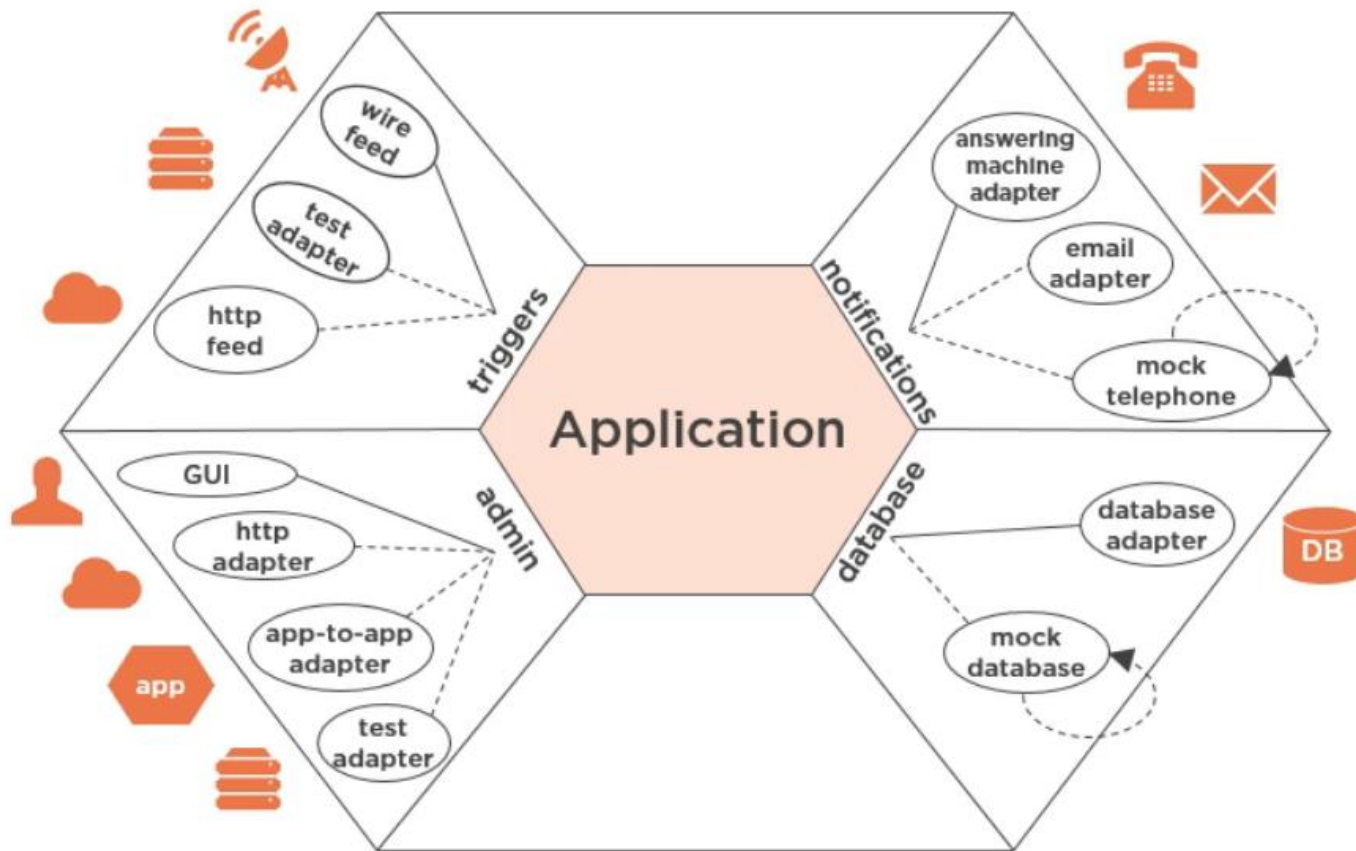


The screenshot shows a web browser window displaying the FitNesse FrontPage. The browser's address bar shows the URL `localhost:8080`. The page features a sidebar on the left with a **FitNesse** logo and buttons for **Edit**, **Properties**, **Where Used**, **Search**, **Files**, **Versions**, **Recent Changes**, **User Guide**, and **Test History**. The main content area is titled **FrontPage** [add child] and includes a **WELCOME TO [FitNesse!](#)** message. Below this, it states: **THE FULLY INTEGRATED STAND-ALONE ACCEPTANCE TESTING FRAMEWORK AND WIKI.** A paragraph follows: "To add your first 'page', click the [Edit](#) button and add a [WikiWord](#) to the page." A table titled **To Learn More...** provides links to various resources:

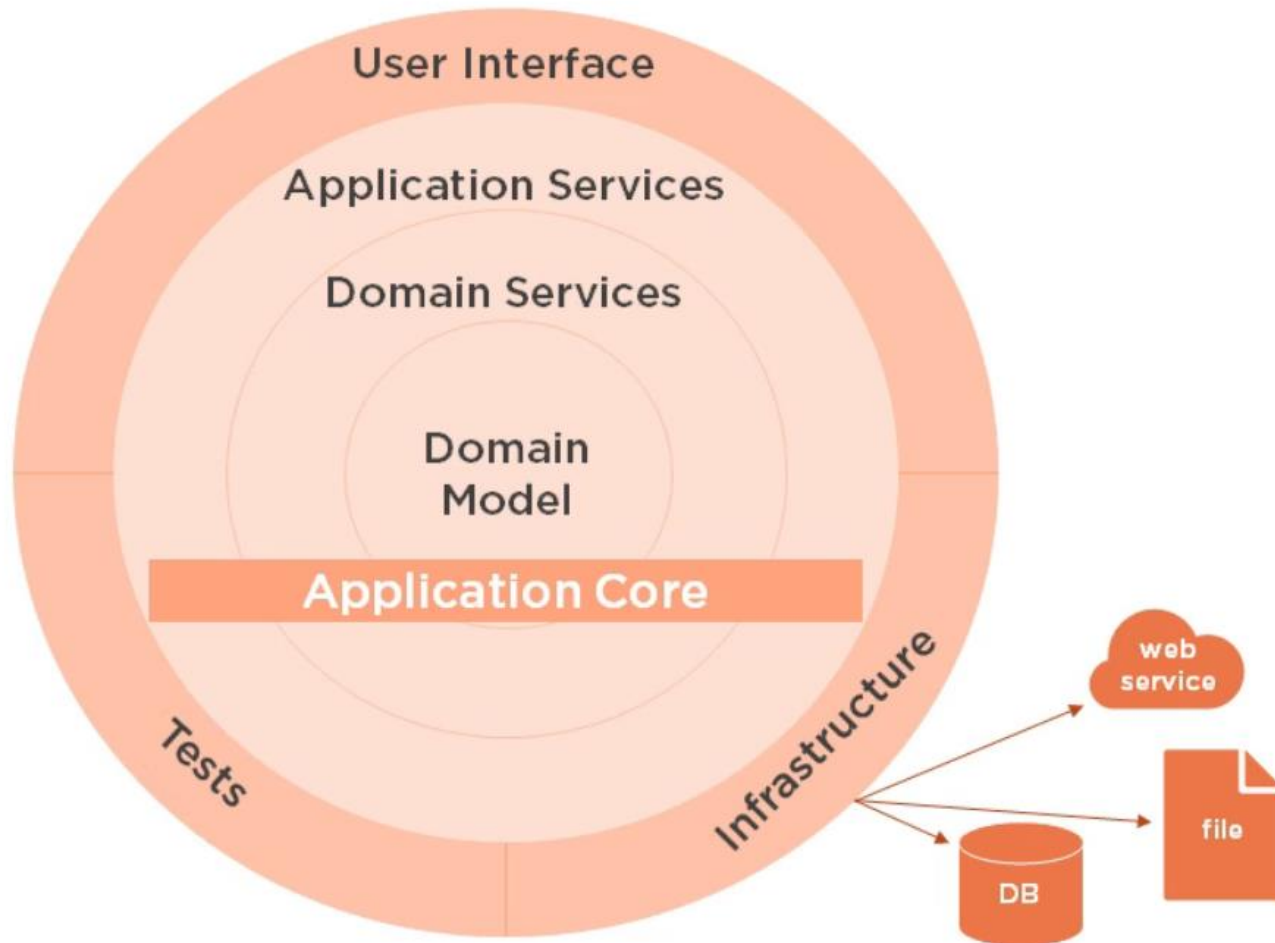
To Learn More...	
A One-Minute Description	<i>What is FitNesse? Start here.</i>
A Two-Minute Example	<i>A brief example. Read this one next.</i>
User Guide	<i>Answer the rest of your questions here.</i>
Acceptance Tests	<i>FitNesse's suite of Acceptance Tests</i>

At the bottom, the release version **Release v20110930** is noted, along with links for [Front Page](#), [User Guide](#), and [root](#) (for global lpath's, etc.).

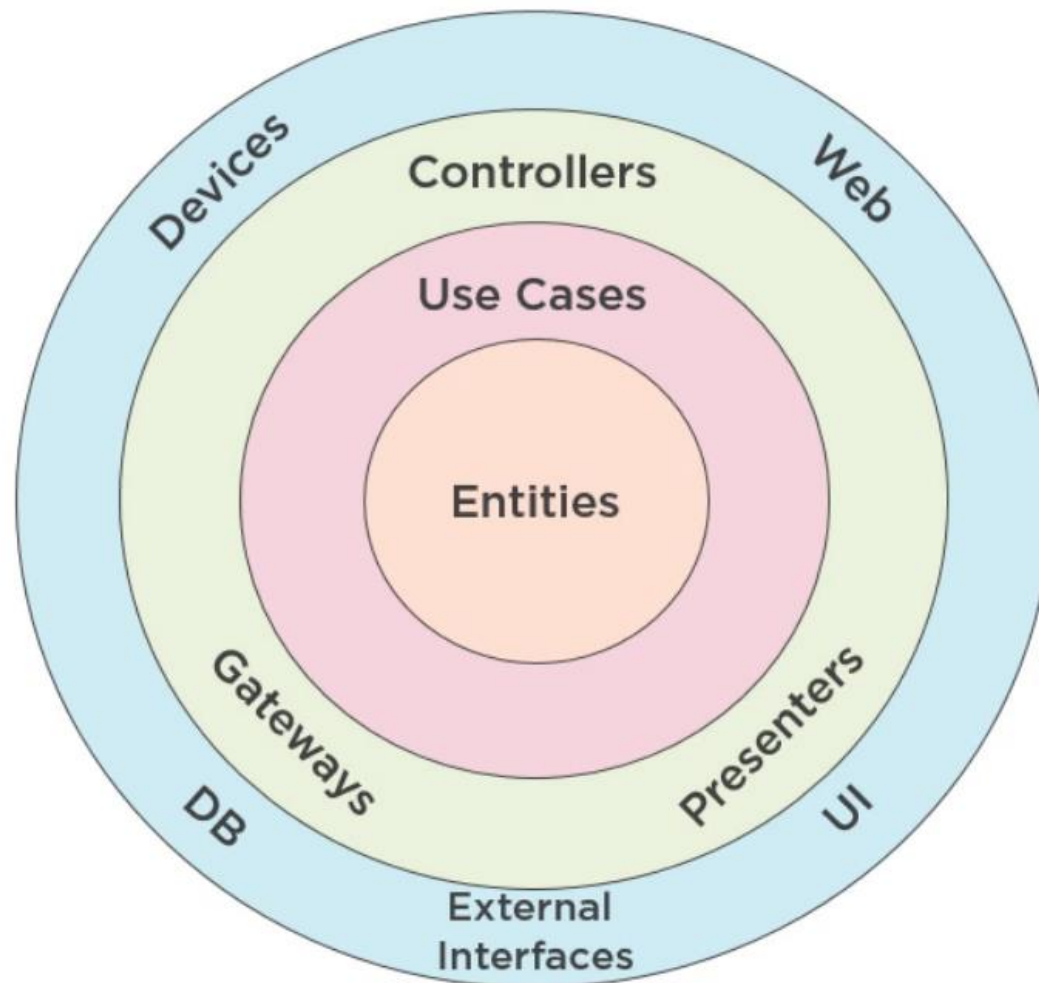
Шестиугольная архитектура



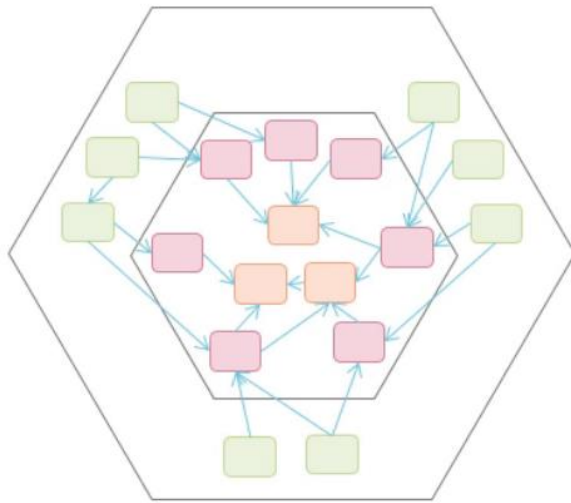
Луковая архитектура



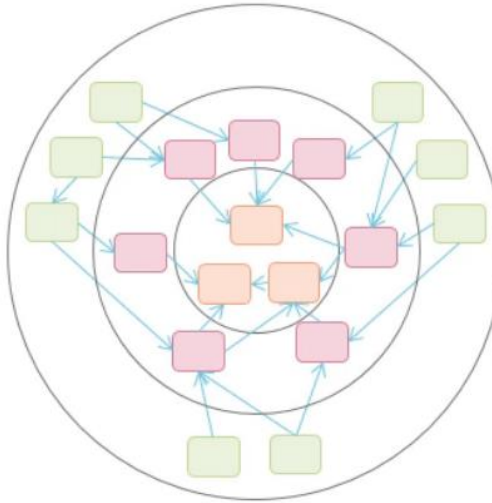
Чистая архитектура



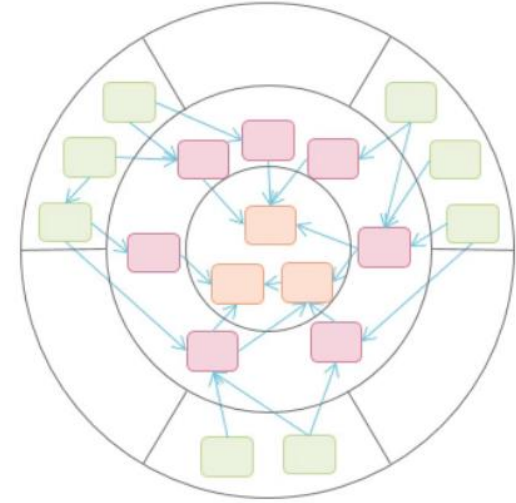
Сходство подходов



Hexagonal



Onion

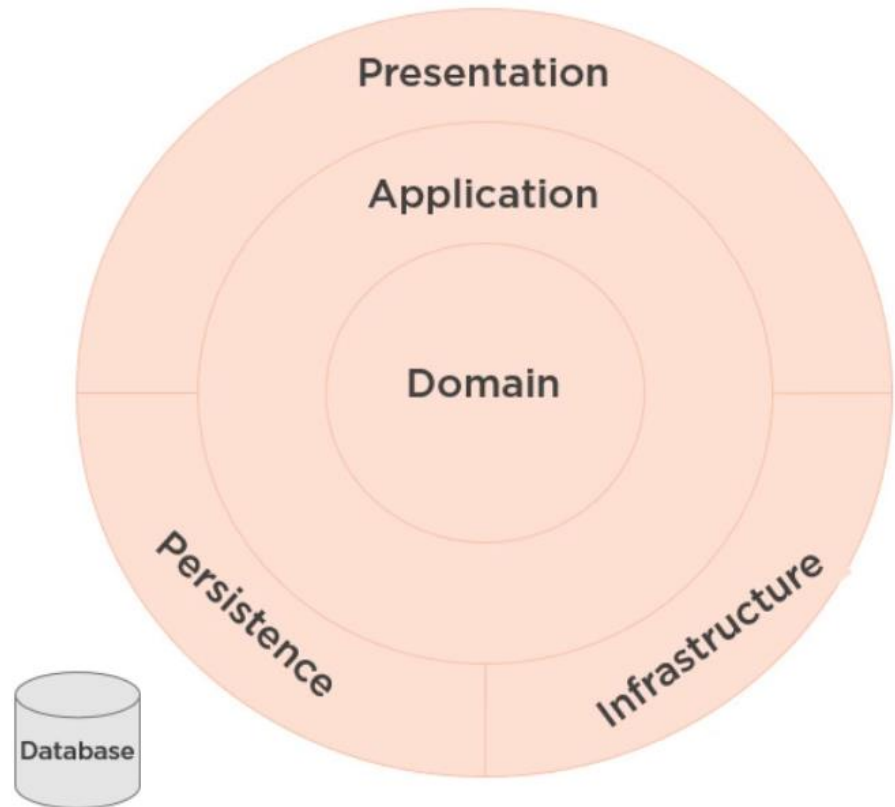


Clean

Проектирование вокруг домена

Преимущества

- Фокус на предметной области и сценариях использования
- Слабая связанность
- Использование практик DDD





Проектирование вокруг домена

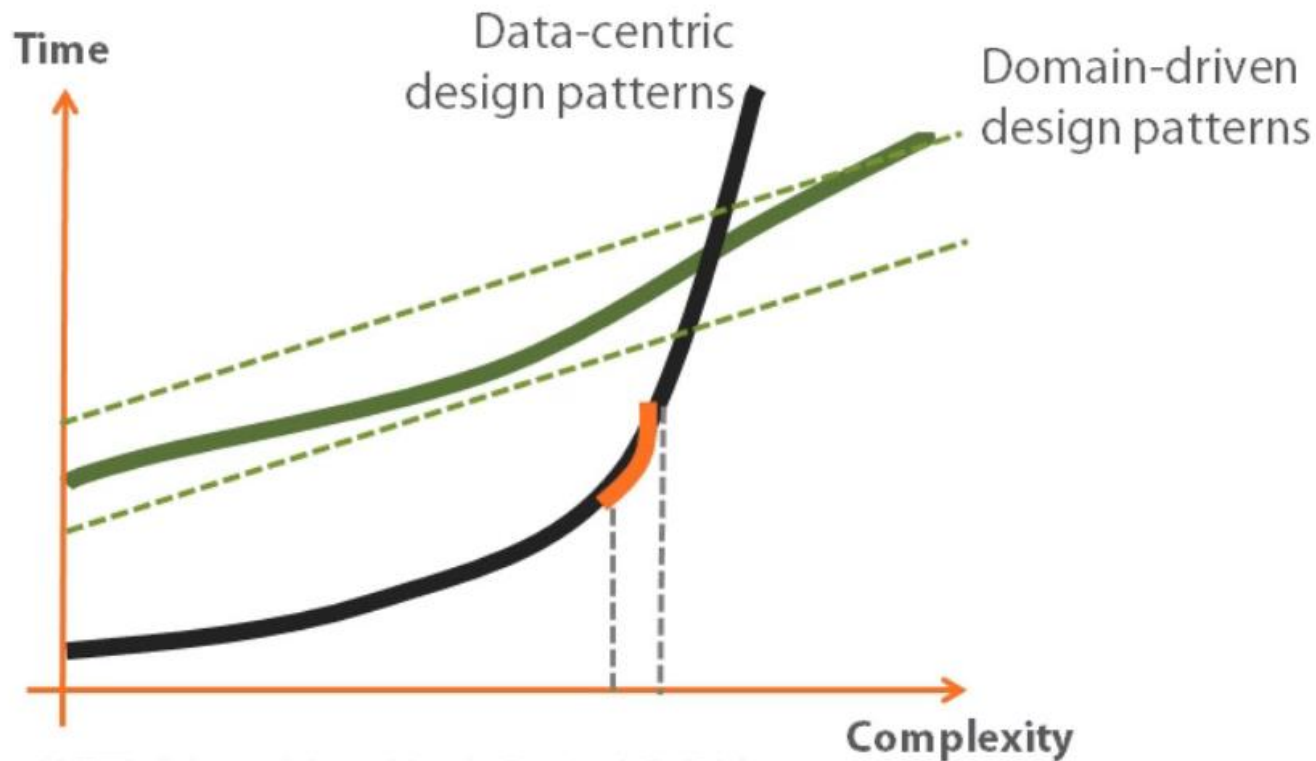
Преимущества

- Фокус на предметной области и сценариях использования
- Слабая связанность
- Использование практик DDD

Недостатки

- Сложность в освоении
- Требуется большей осмысленности
- Высокие затраты на ранних этапах

Данные vs Домен



NOTE: Adapted from Martin Fowler's PoEAA



Command-Query Separation

Command

- Делает изменение
- Модифицирует состояние системы
- Не возвращает значения

Query

- Отвечает на вопрос
- Не модифицирует состояние системы
- Возвращает значение

Bertrand Meyer (1997)



Исключения из CQS

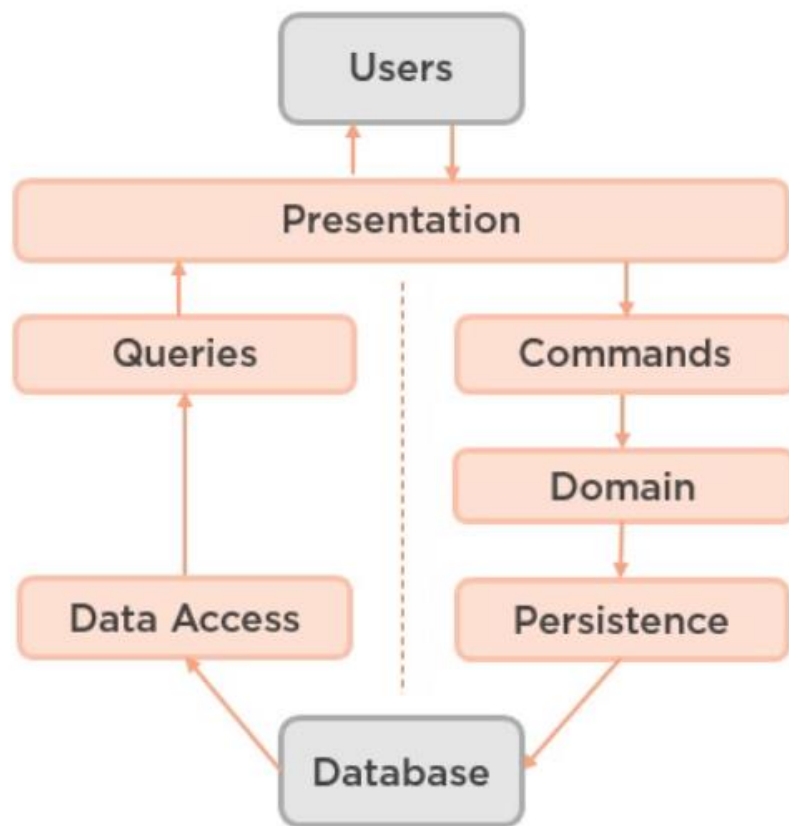
Выталкивание элемента из стэка

- Удаляет элемент (command)
- Возвращает верхний элемент (query)

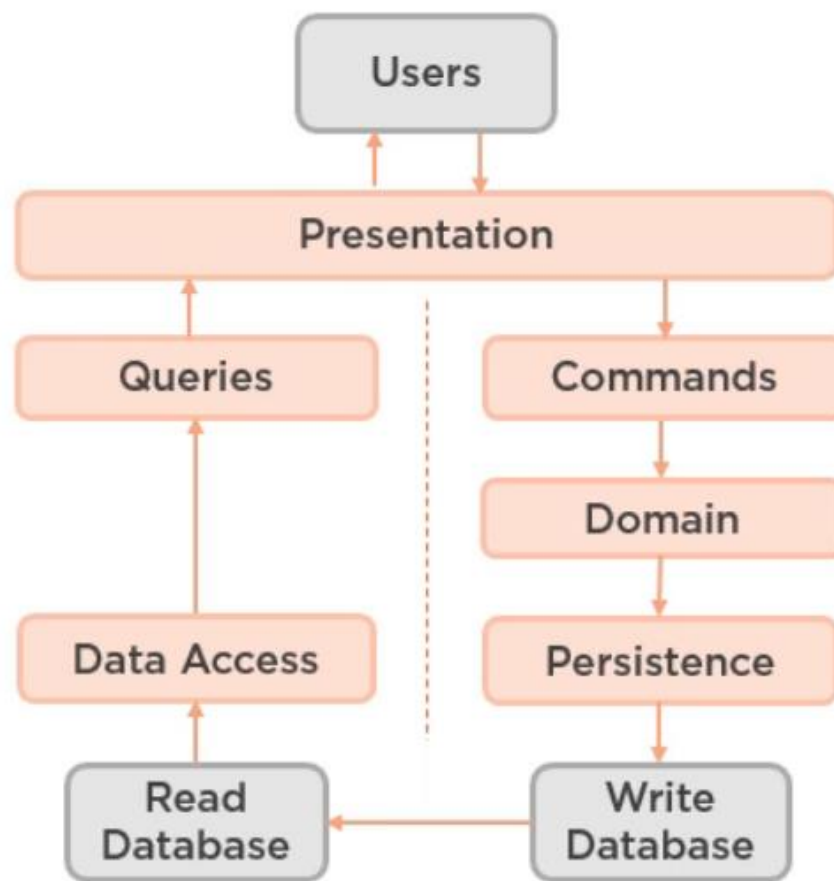
Создание новой записи

- Создает запись (command)
- Возвращает новый ID (query)

CQRS архитектура

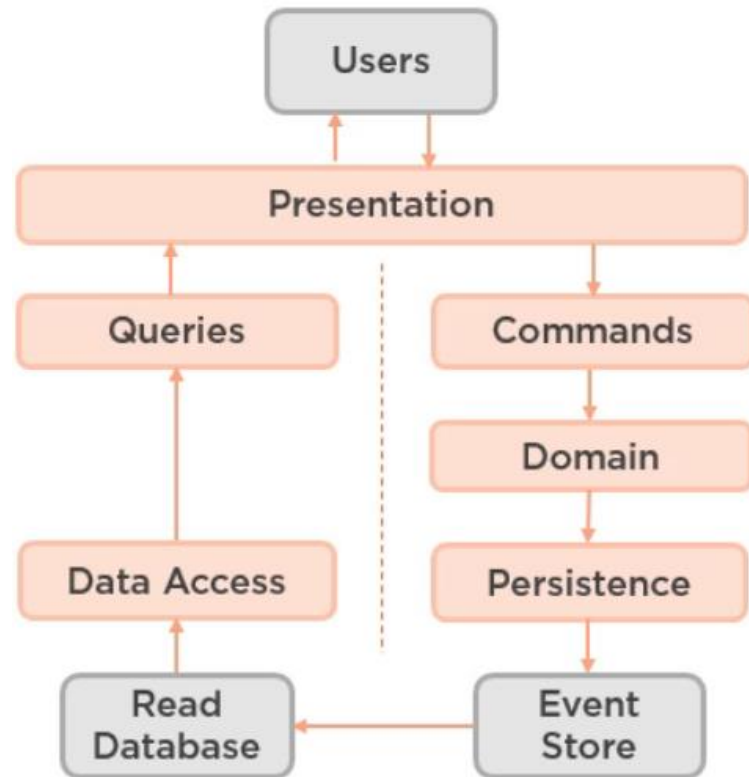


CQRS с раздельными БД



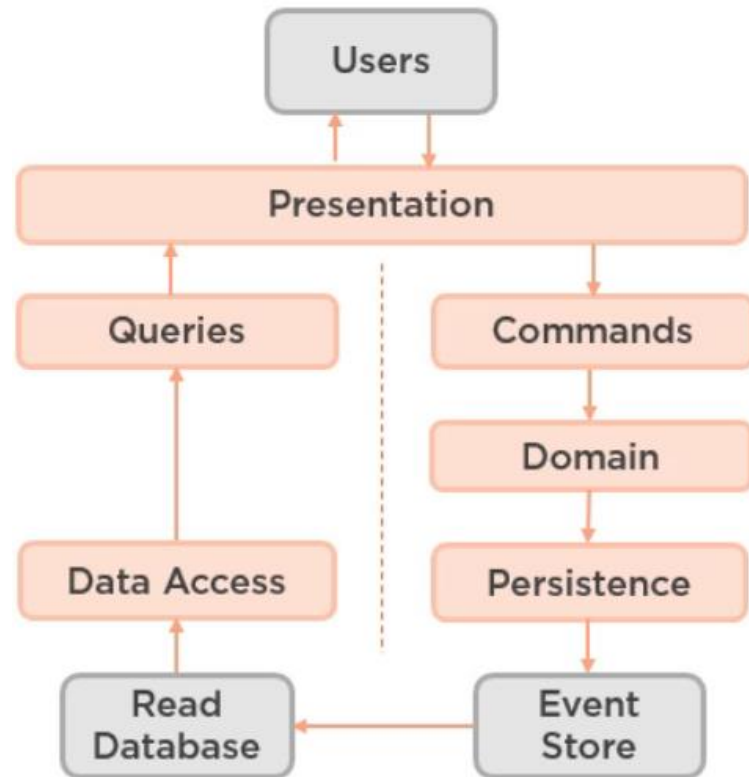
Event Sourcing CQRS

- Хранение событий
- Воспроизведение событий
- Изменение сущности
- Сохранение нового события
- Обновление БД для чтения



Event Sourcing CQRS

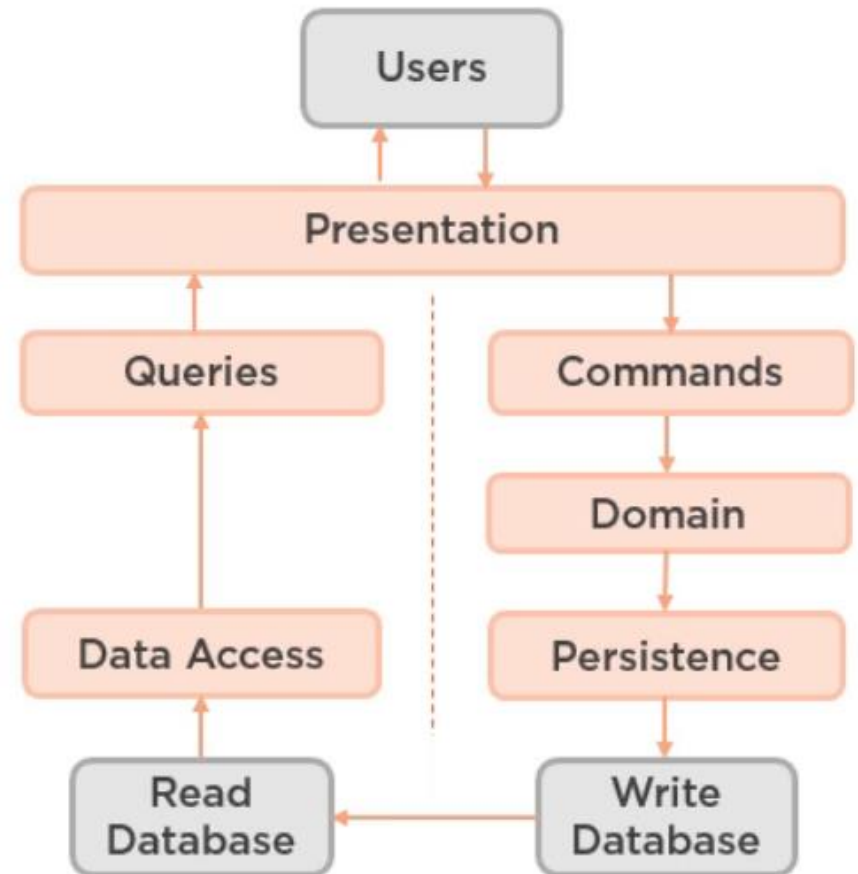
- Аудит событий
- Восстановление контекста
- Воспроизведение событий
- Использование нескольких БД для чтения



CQRS

Преимущества

- Удобный дизайн
- Высокая производительность
- Преимущества Event sourcing





CQRS

Преимущества

- Удобный дизайн
- Высокая производительность
- Преимущества Event sourcing

Недостатки

- Несовместимый подход к commands и queries
- Сложность разработки и поддержки



Организация кода

Robert Martin (Clean Code, 2011)

Архитектура должна кричать о назначении системы.

Организация кода



Content



Controllers



Models



Scripts



Views

Организация кода



Content



Controllers



Models



Scripts



Views

vs



Customers



Employees



Products



Sales



Vendors

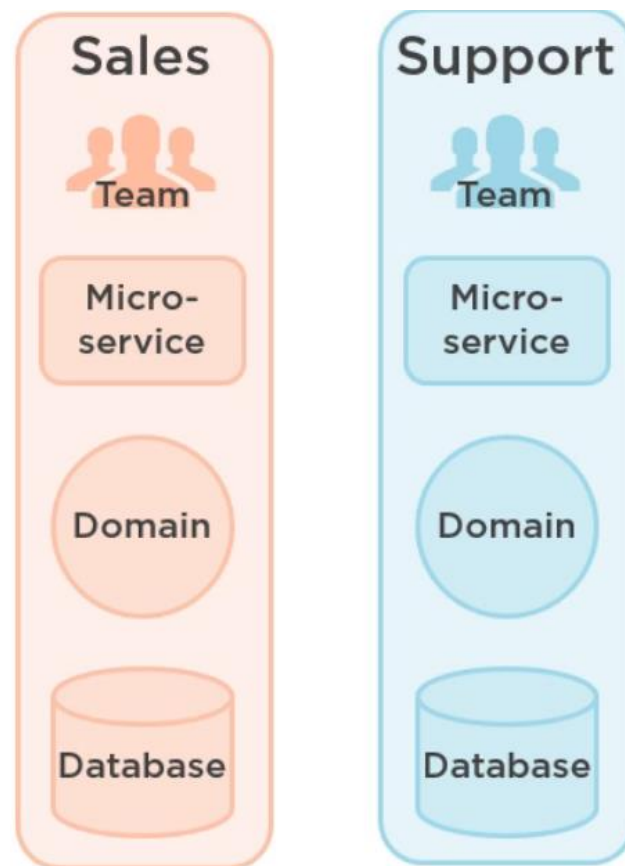
Микросервисная архитектура

- Разделение на под-системы
- Понятные интерфейсы
- Небольшие команды
- Независимость разработки и развертывания



Микросервисная архитектура

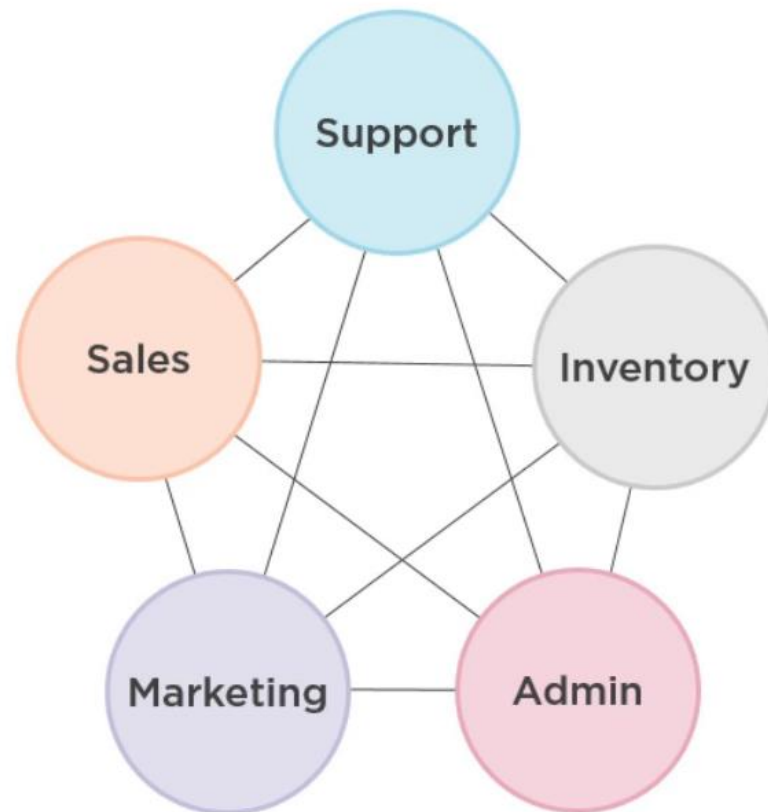
- Разделение контекстов
- Высокая связность и слабая связанность
- Фокус на одном контексте
- Независимость подходов и инструментов



Микросервисная архитектура

Преимущества

- Простота расширения системы
- Высокая связность и слабая связанность
- Независимость





Микросервисная архитектура

Преимущества

- Простота расширения системы
- Высокая связность и слабая связанность
- Независимость

Недостатки

- Издержки в начале разработки
- Закон Конвея
- Издержки распределенных систем

Микросервисная архитектура



Плохая



Слоистая



Микросервисная

Адаптируемая архитектура

- Снижение рисков на ранних этапах



Адаптируемая архитектура

- Снижение рисков на ранних этапах
- Технологии могут измениться



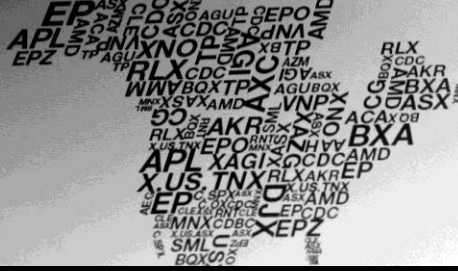
Адаптируемая архитектура

- Снижение рисков на ранних этапах
- Технологии могут измениться
- Ситуация на рынке может измениться





Q&A



Жаров Алексей

alexeyzh@cqg.com