Implementação de API Avengers com SpringBoot + Kotlin

Apresentador

- Jether Rodrigues do Nascimento
  - Bacharel em Ciência da Computação
  - MBA em Tecnologia da Informação
  - Especialista em Desenvolvimento de Software
  - https://github.com/jether2011

Agenda - Conceitos

- Estilo Arquitetural REST
- Conceito de API First
- API RESTFul com Spring Boot
- Clean Architecture
  - Ports And Adapters
- Sistema Gerenciamento de Banco de Dados
  - NoSQL
  - Relacional
- Postman

Conceitos - REST

- REST
  - REST é um acrônimo de representação de transferência de estado (**RE**presentational **S**tate **T**ransfer);
  - REST é um estilo arquitetural apresentado por Roy Fielding (HTTP e REST)
  - Interoperável e agnóstico a linguagem ou tecnologia (meio de integração entre sistemas de qualquer stack tecnológica)

Conceitos - REST

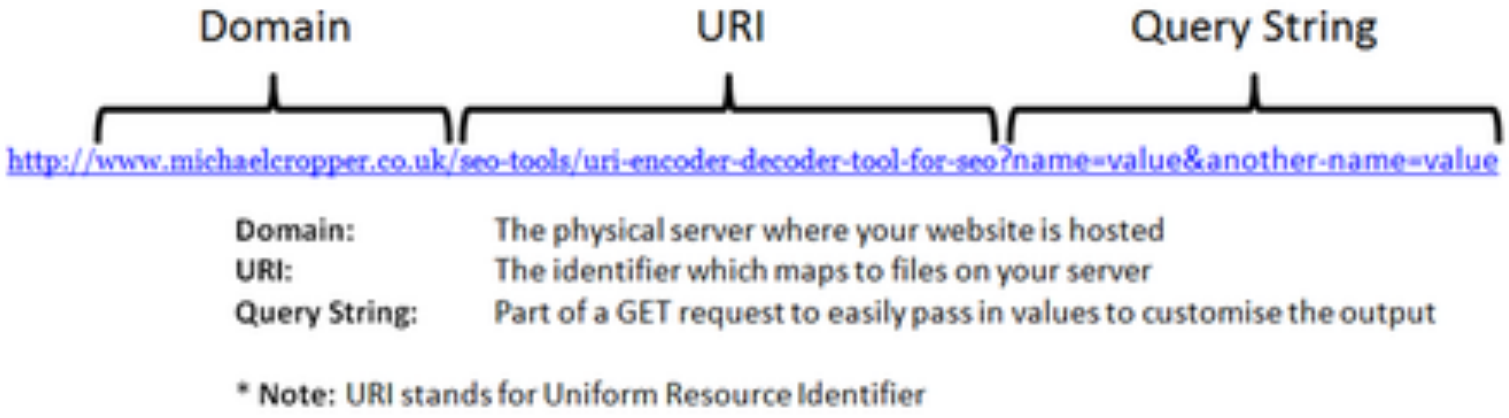
- Constraints
  - Interface Uniforme
    - Baseado em recursos, interfaces por onde o cliente se comunica com o servidor
  - Cliente-Servidor
    - Cliente e servidor podem existirem e serem desenvolvidos separadamente, são independentes
  - Sem estado
    - O servidor não guarda estado, sessão, necessário sempre, a cada requisição, caso necessário, informar o servidor de quem está requerendo o dado especifíco

Conceitos - REST

- Constraints
  - Cache
    - Funcionalidade que elimina, dado algumas configurações, voltar ao servidor, se a requisição se repetir dentro de um tempo. Cache pode ser gerenciando em modo client ou server side
  - Sistema em camadas
    - Por ser sem estado, permite distribuir as implantações e ou recursos, ex.: API em um servidor, data storage em outro e sistema de autenticação em um terceiro servidor
  - Código sobre demanda (opcional)
    - Possibilidade de retornar ”executáveis” via API para renderização de algum componente se necessário

Conceitos - REST

- **Resource Naming Guide**
  - Estratégia para expor recursos (plural ou singular)
  - Estrutura de nomes para recuperar recursos, armazenar, detalhe, nomes compostos, passar parametros (query param, path param)
  - URI (endereçamento para um determinado recurso)



Conceitos - REST

- **Caching**
  - Capacidade de manter cópias de um response armazenadas por um tempo determinado
  - Elementos:
    - Tempo de expiração
    - Cache-Control
    - E-tag
    - Last Modified
  - Cuidados

Conceitos - REST

- **Versionamento**
  - Permite evolução de uma API sem trazer quebra de contratos para os clientes (mudanças em requests e responses, remoção de API)
  - Tipos:
    - URI: http://api.example.com/v1, http://apiv1.example.com
    - Header Customizado: Accept-version: v1
    - Content Negotiation (Accept Header):
      - Accept: application/vnd.example.v1+json
      - Accept: application/vnd.example+json;version=1.0

Conceitos - REST

- **Verbos HTTP**
  - GET
  - POST
  - PUT
  - PATCH
  - DELETE

Conceitos - REST

- **Códigos HTTP**
  - 1xx - Informações
  - 2xx – Sucesso na requisição
  - 3xx - Redirecionamentos
  - 4xx – Erro no lado do cliente
  - 5xx – Erro no lado do servidor

- https://httpstatuses.com

Conceitos - REST

- **Idempotência**
  - Uma requisição idempotente é aquela que, independente da quantidade de vezes que se execute, o efeito é o mesmo que da primeira execução
  - POST – Altera estado no servidor, cria recurso, logo, não é idempotente
  - GET, PUT, DELETE, HEAD, OPTIONS e TRACE são idempotentes

Conceitos - REST

- **N+1 em REST APIs**

Conceitos – API FIRST

- https://swagger.io/resources/articles/adopting-an-api-first-approach
- https://editor.swagger.io

Agenda - API

- Pré-requisito
- Arquitetura hexagonal aplicado ao projeto
  - Application Layer
    - Configs
    - Controllers (conceito de fronteira)
      - DTOs (request, response)
      - Bean Validations
      - Init Binders (validações customizadas)
      - Error Handlers

Agenda - API

- Domain Layer
  - Entidades
  - Portas (interfaces)
  - Services
- Resource Layer
  - Spring data repository
  - Proxy repository (composição com spring data repository e implementação interface repository do domínio)
  - Entidades mapeadas com anotações de acordo com tecnologia de banco de dados escolhida para o projeto
- Testes
- https://start.spring.io

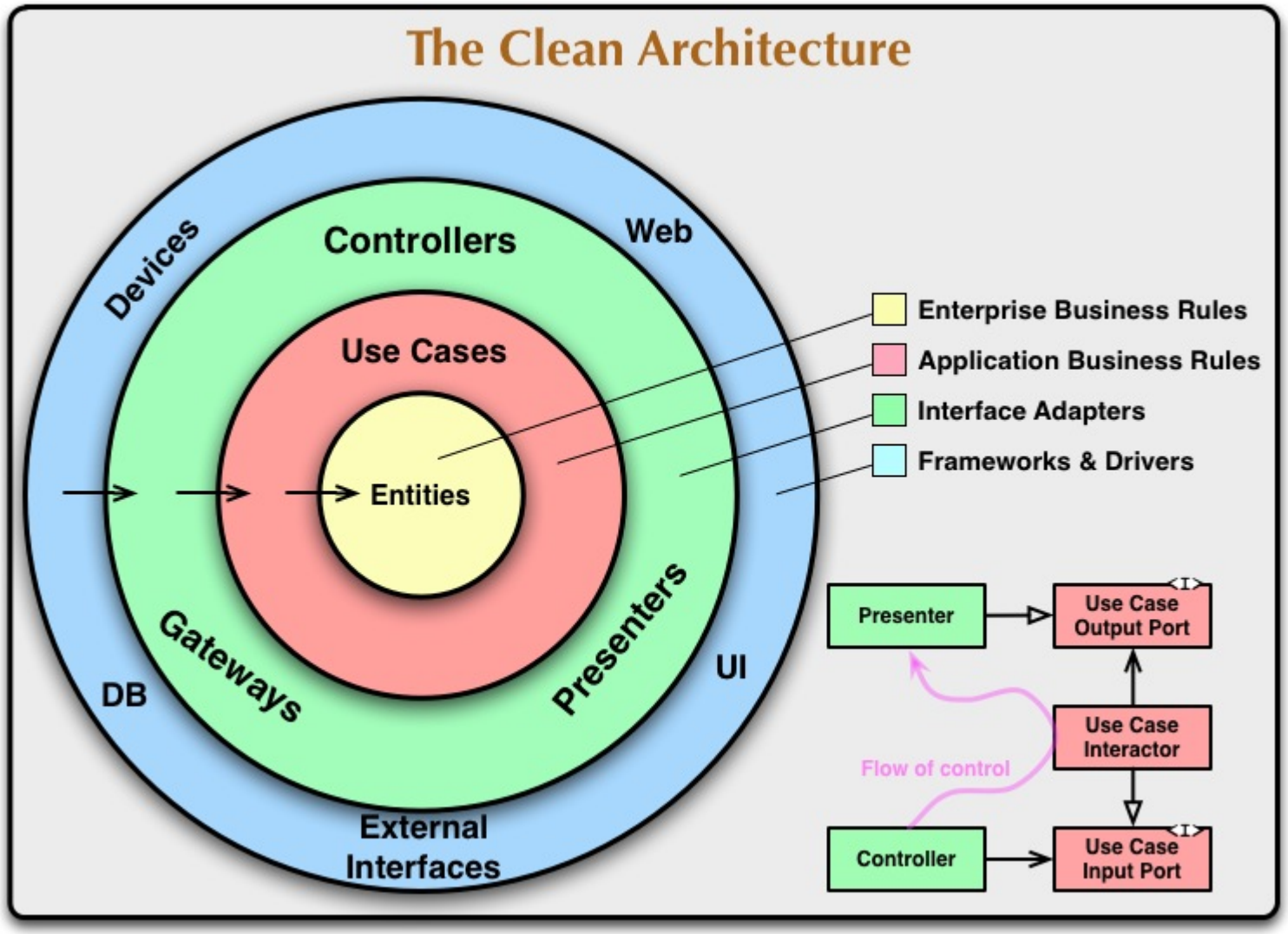
Agenda - API

- **Dockerização**
  - Script e configuração para criação de imagem
  - Compose yaml do serviço
  - Usando o docker-compose para subir o serviço
- **Heroku**
  - Criando uma app
  - Conectando com o github do projeto
  - Automatizando o pipeline de deploy

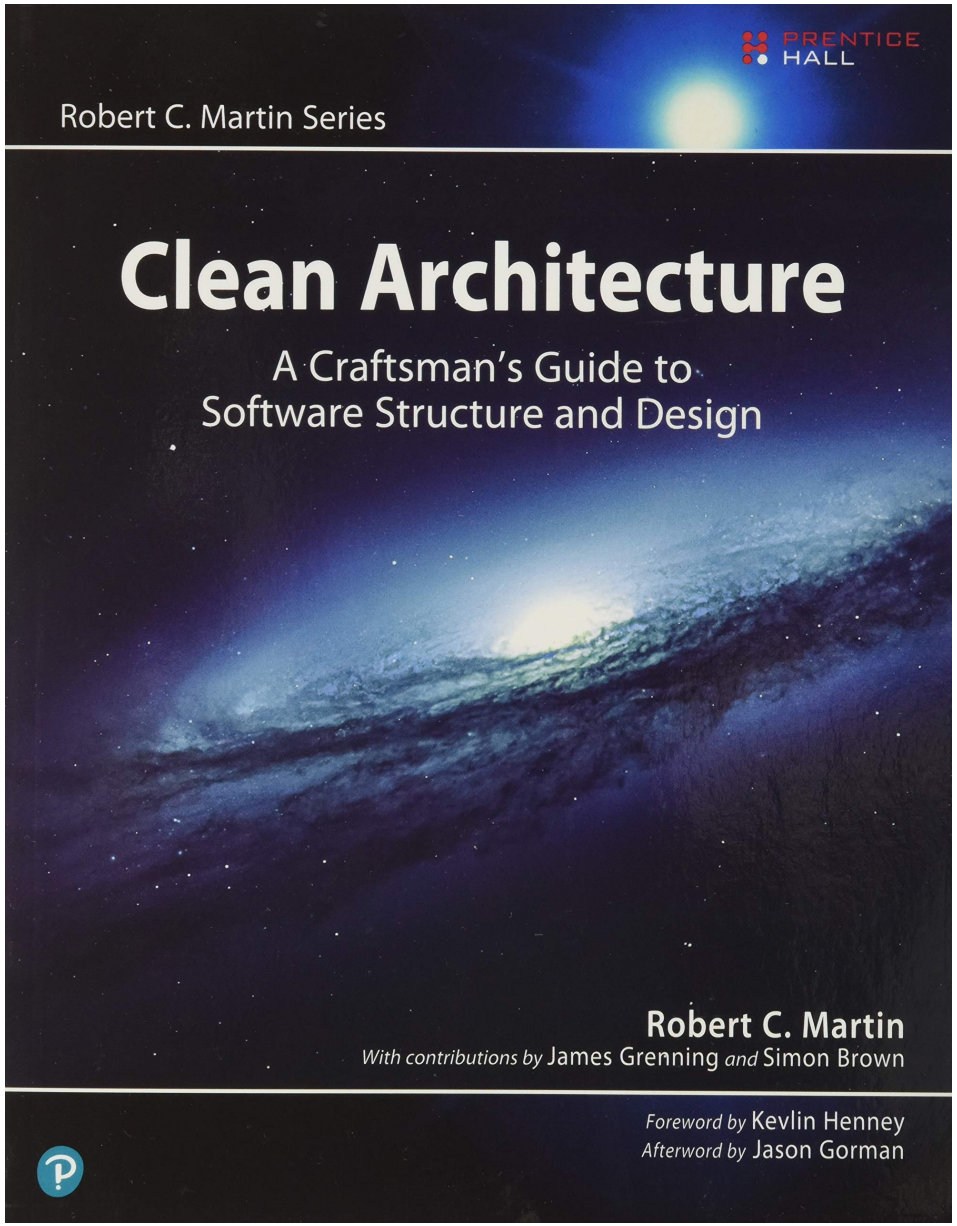
Agenda – Clean Architecture

- Principal objetivo de uso de ***Clean Architecture*** é fornecer aos desenvolvedores uma maneira de organizar o código de forma que encapsule a lógica de negócios, mas mantenha-o separado do mecanismo de entrega.
- **Vantagens**:
  - Independência de interfaces de usuário
  - Independência da stack tecnológica utilizada para acesso aos dados e ou expor entradas
  - Testável
  - Carga cognitiva de entendimento do projeto, regras de negócio
  - Facilita manutenabilidade

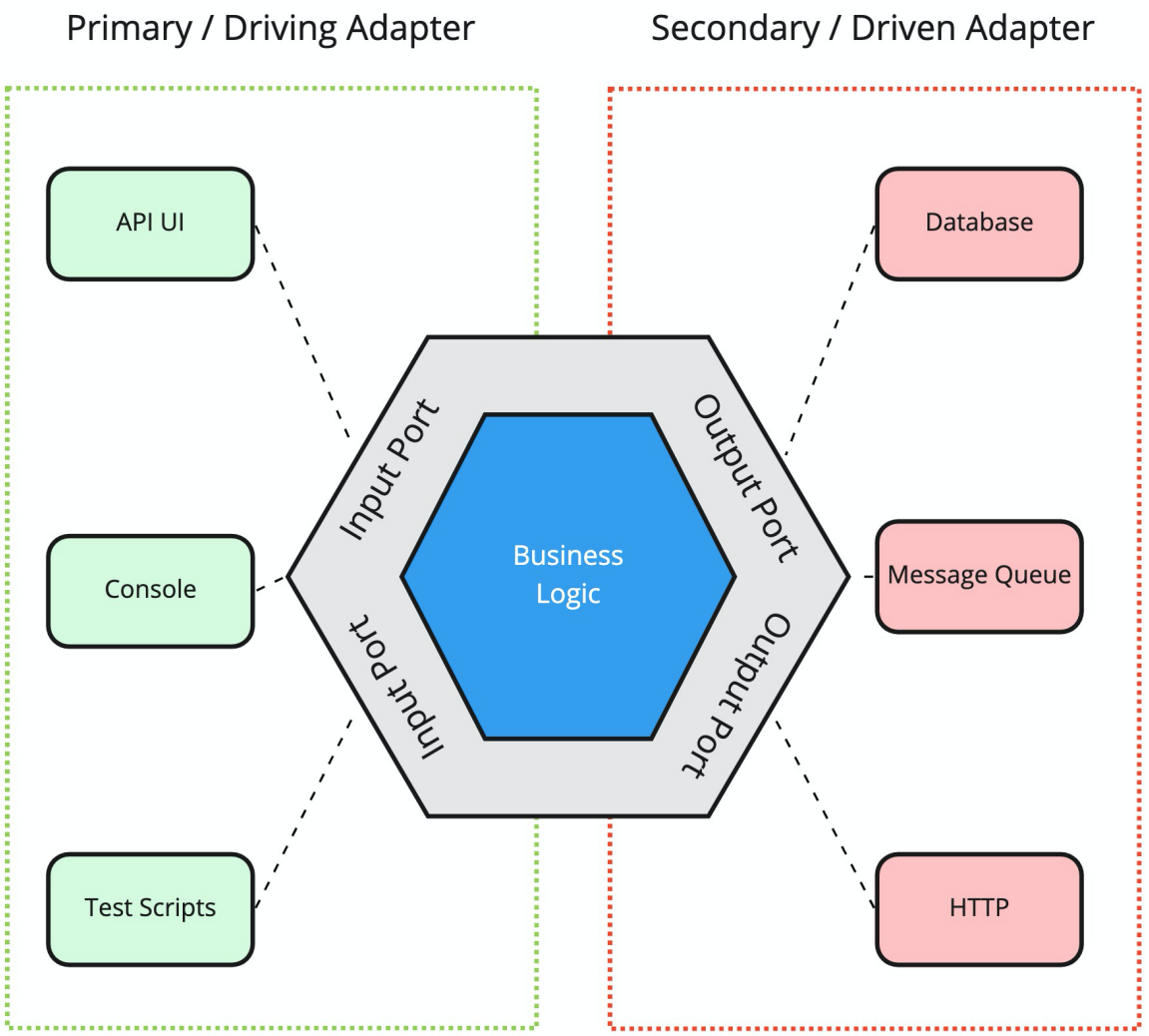
Agenda – Clean Architecture



Agenda – Clean Architecture

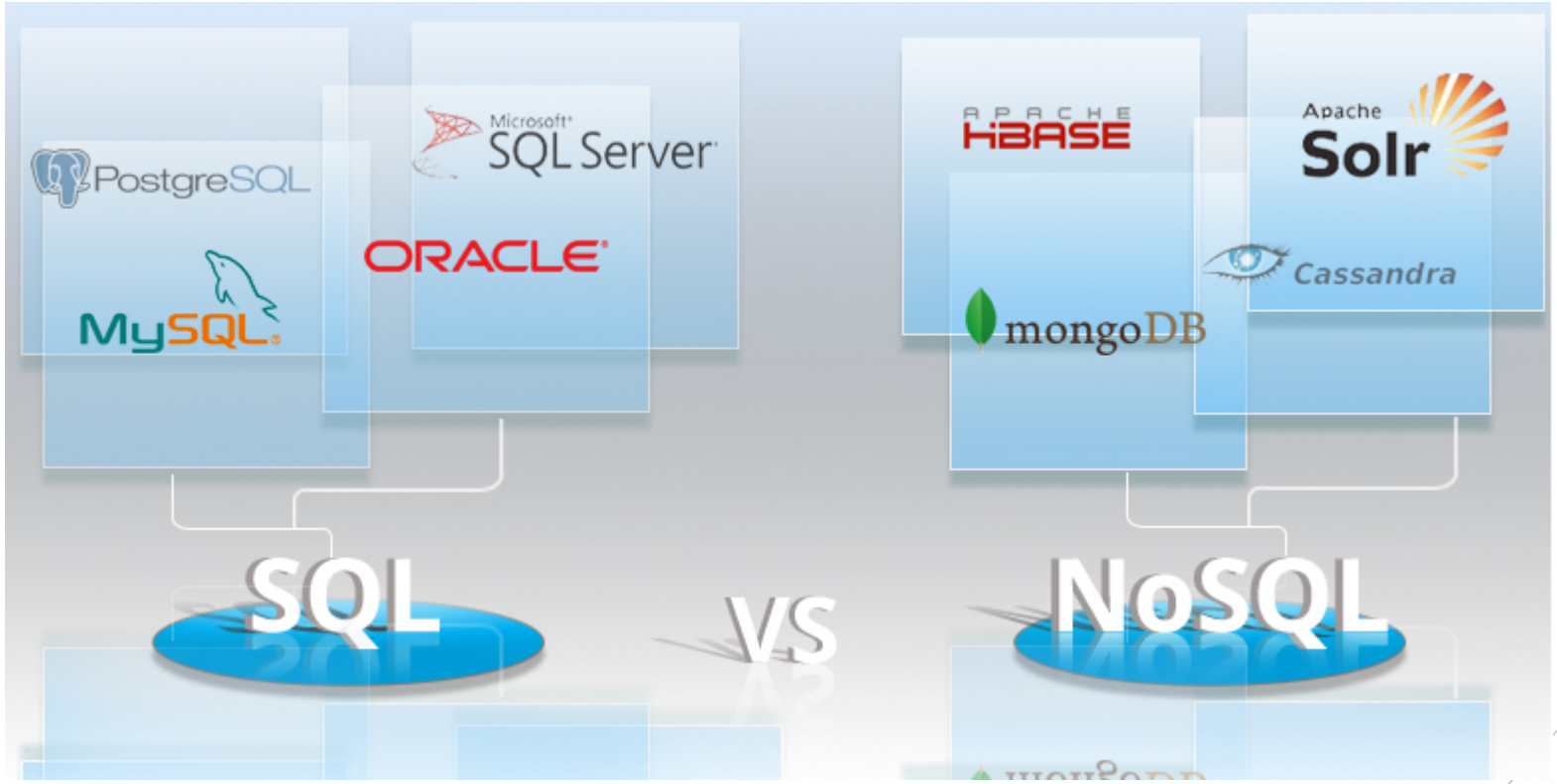


Agenda – Clean Architecture – Ports And Adapter

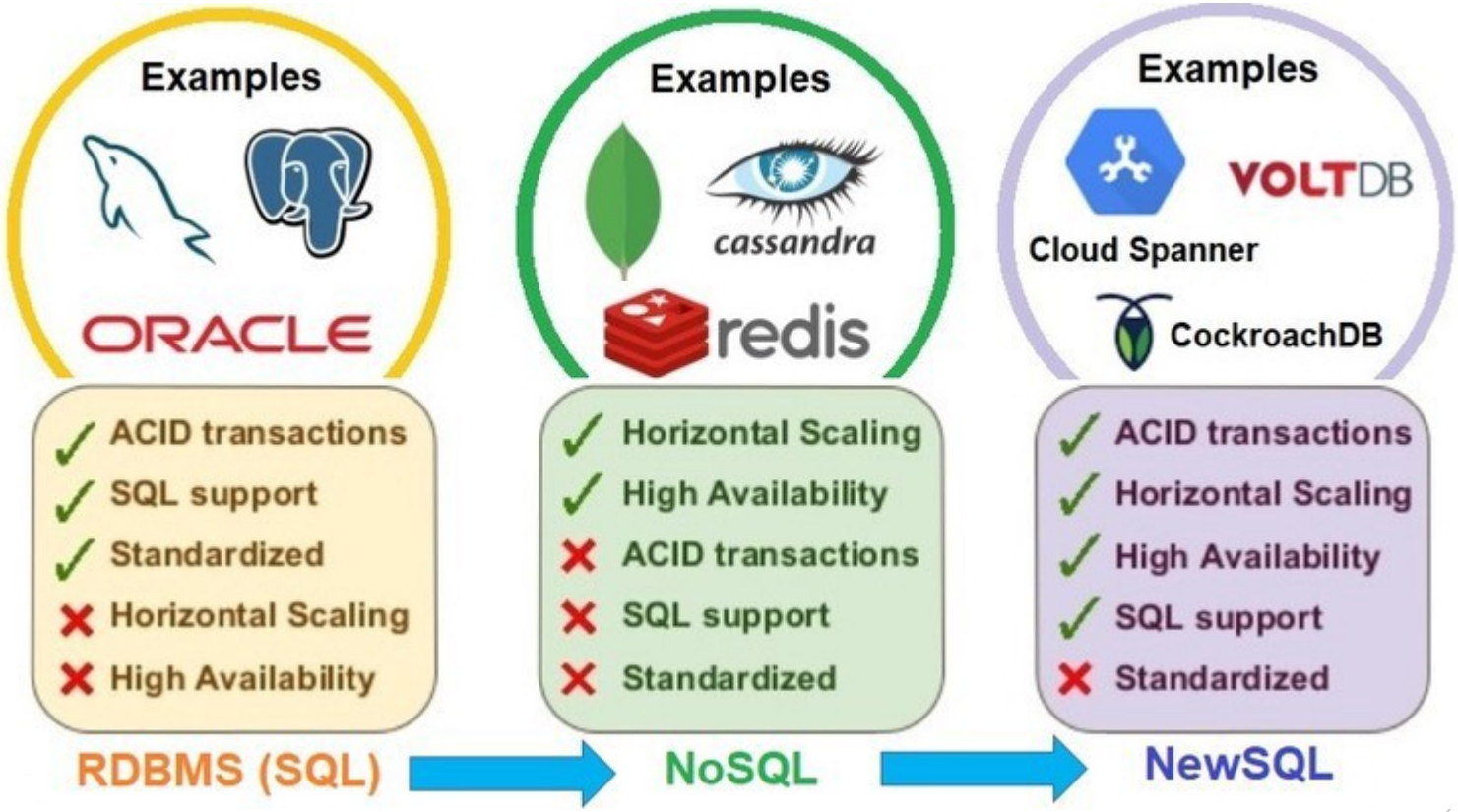


Agenda - Conceitos

- Sistema Gerenciamento de Banco de Dados

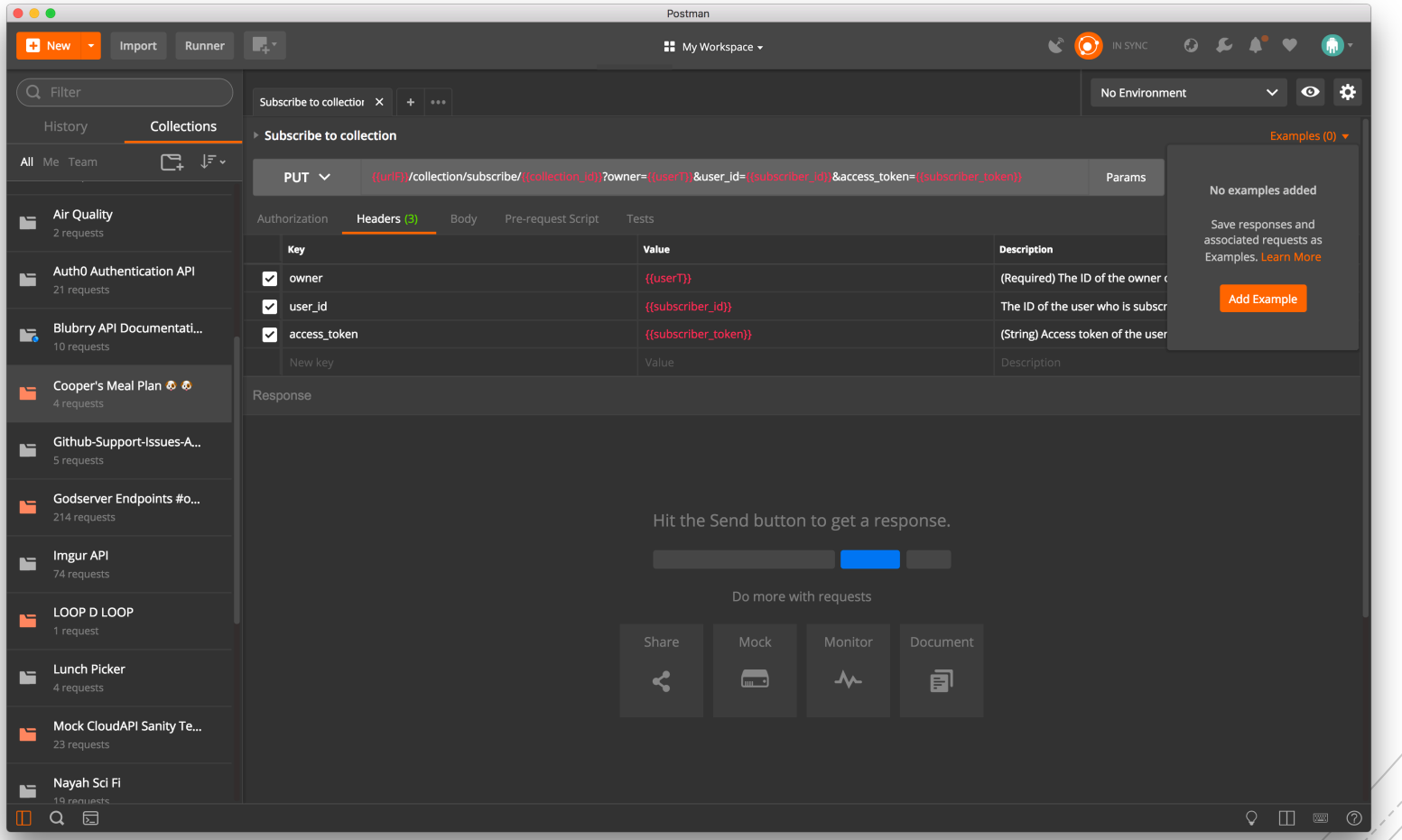


Agenda - Conceitos



Conceitos - Ferramenta

- Postman



Agradecimento

- Obrigado pela atenção
- Dúvidas????