

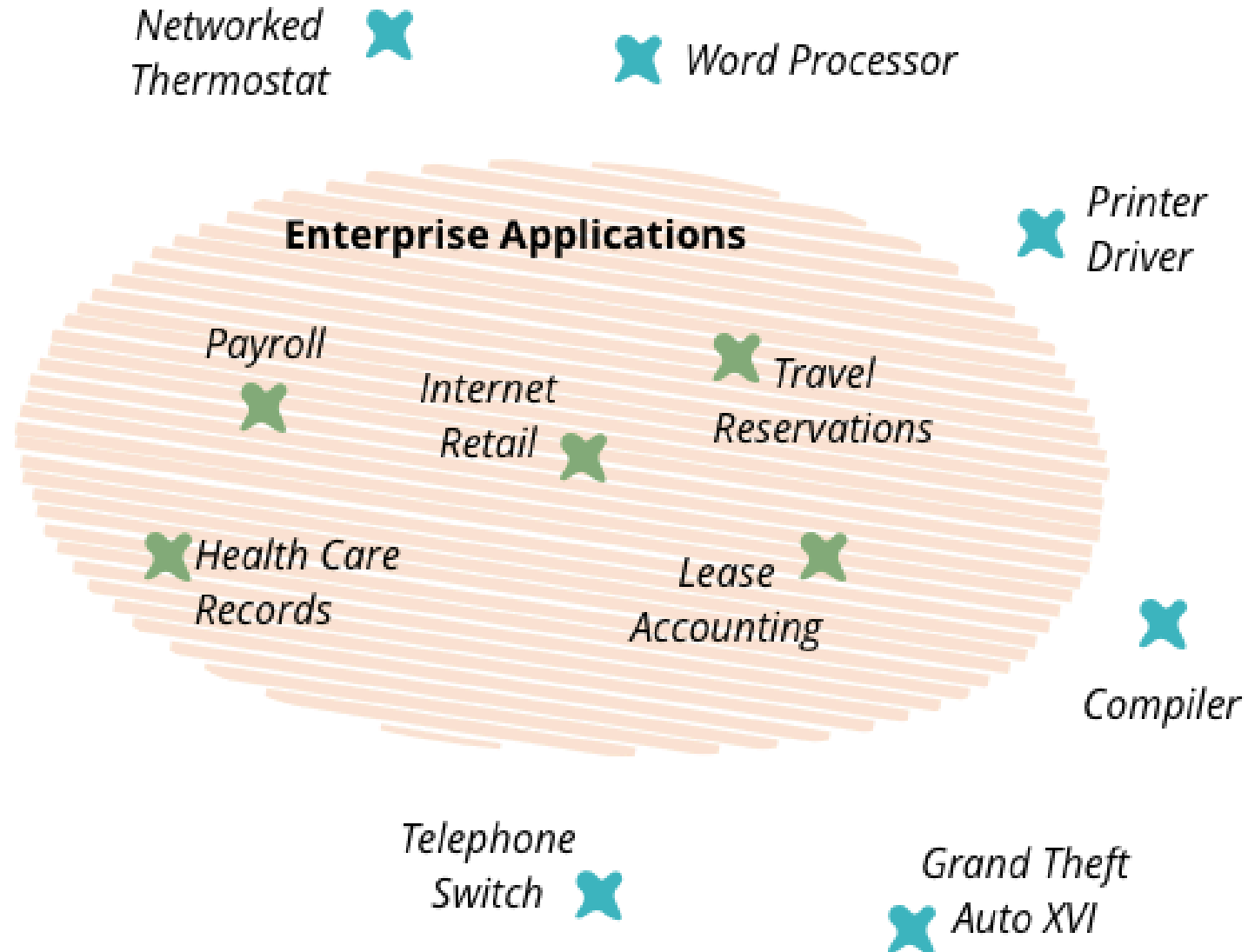
AULA 12 - ENTERPRISE SERVICE

Disciplina de Backend - Professor Ramon Venson - SATC 2024

Enterprise Applications

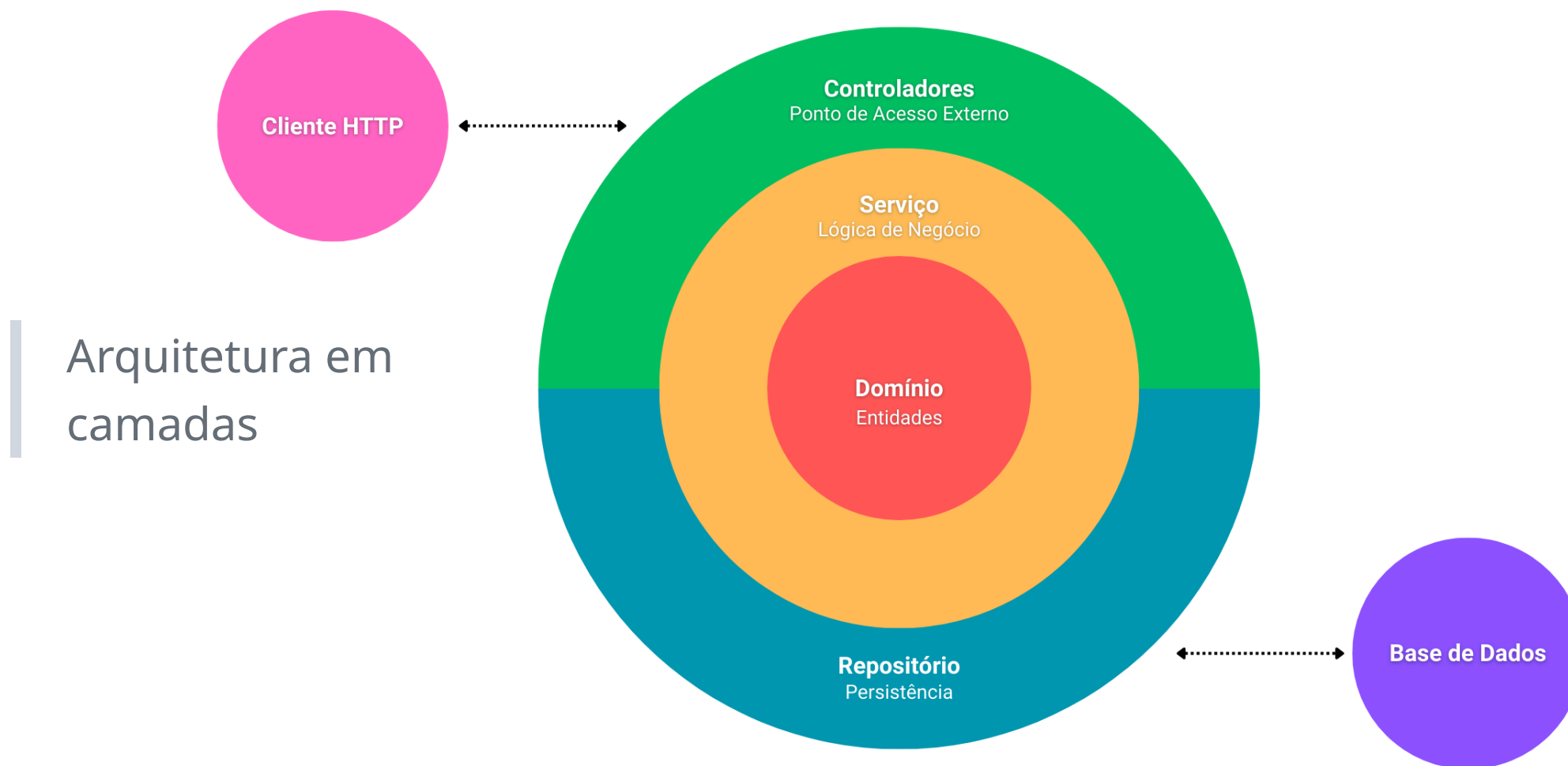
"Os aplicativos corporativos tratam da exibição, manipulação e armazenamento de grandes quantidades de dados muitas vezes complexos e do suporte ou automação de processos de negócios com esses dados." - Martin Fowler

- Multi-usuário e concorrência
- Alto padrão (Patterns e Arquitetura)
- Persistência, Validação, Integridade, Transações, Camadas...

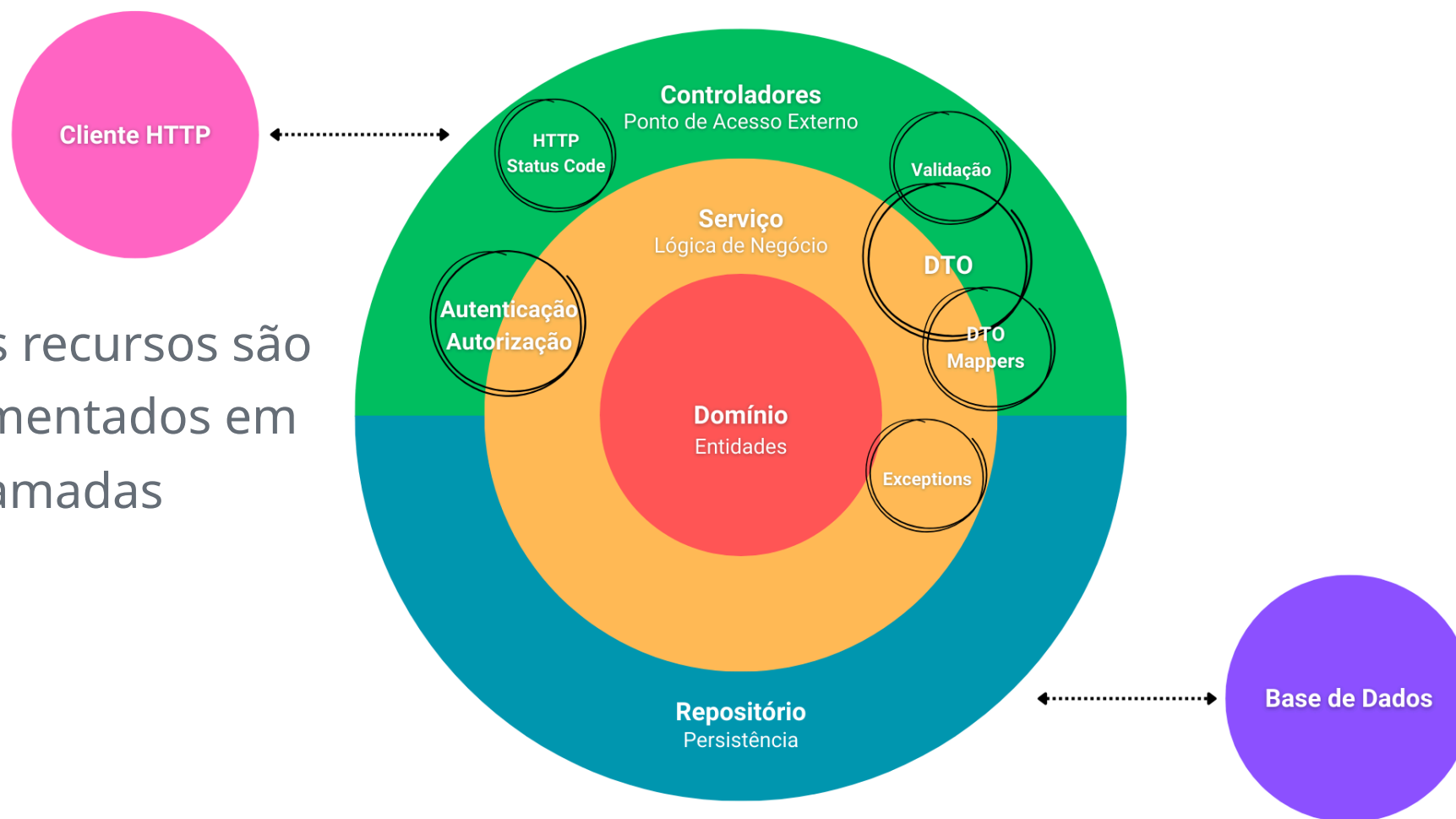


RESTful Services

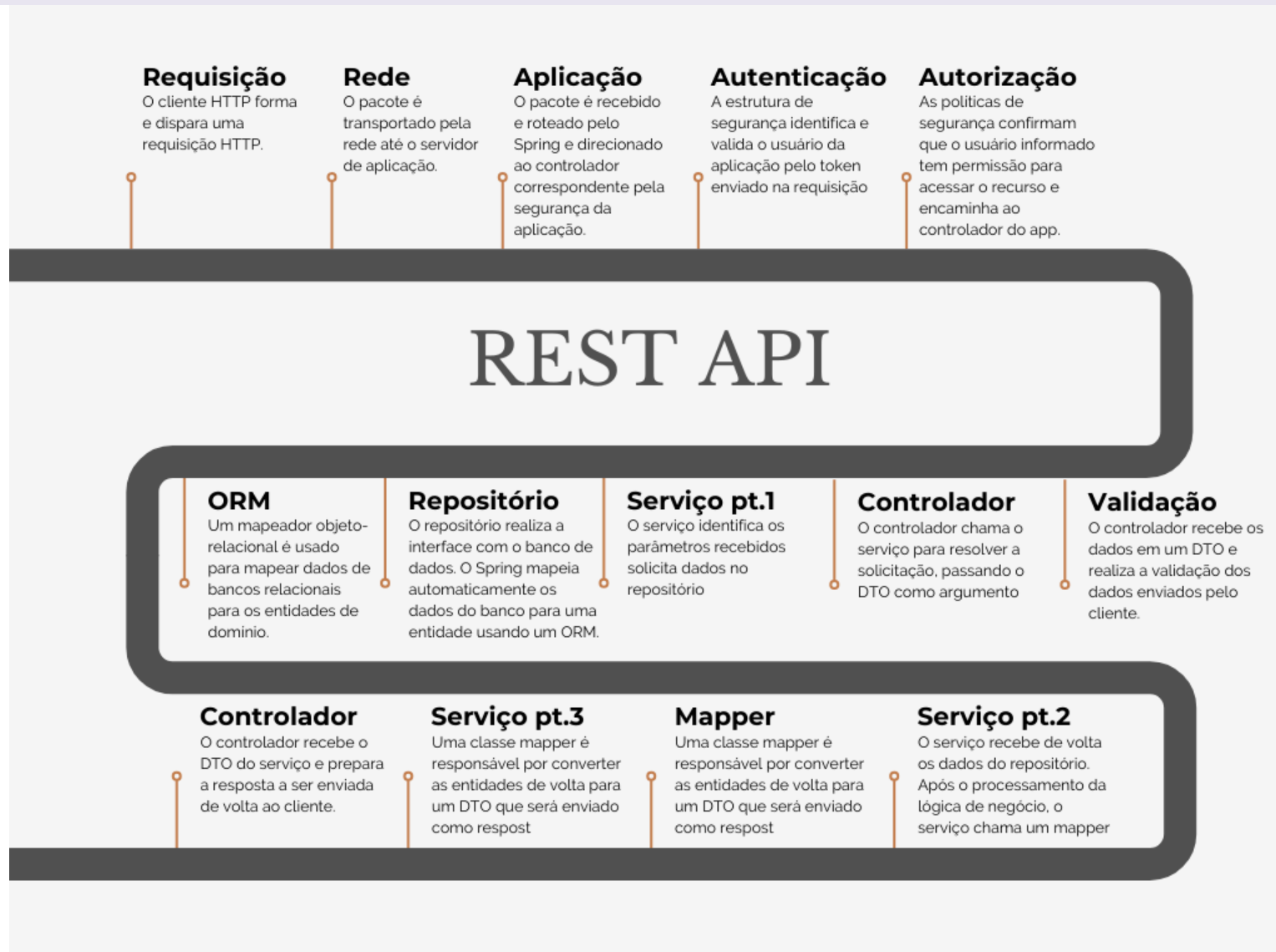
- Interface padronizada para troca de dados
- Baseado em **Recursos**
- Sistema de Camadas
- REST Design é aplicado nos controladores



Alguns recursos são implementados em intercamadas



Ciclo de Vida



Desenvolvimento

External Layer

- Validações
- HTTP Codes
- Controller REST

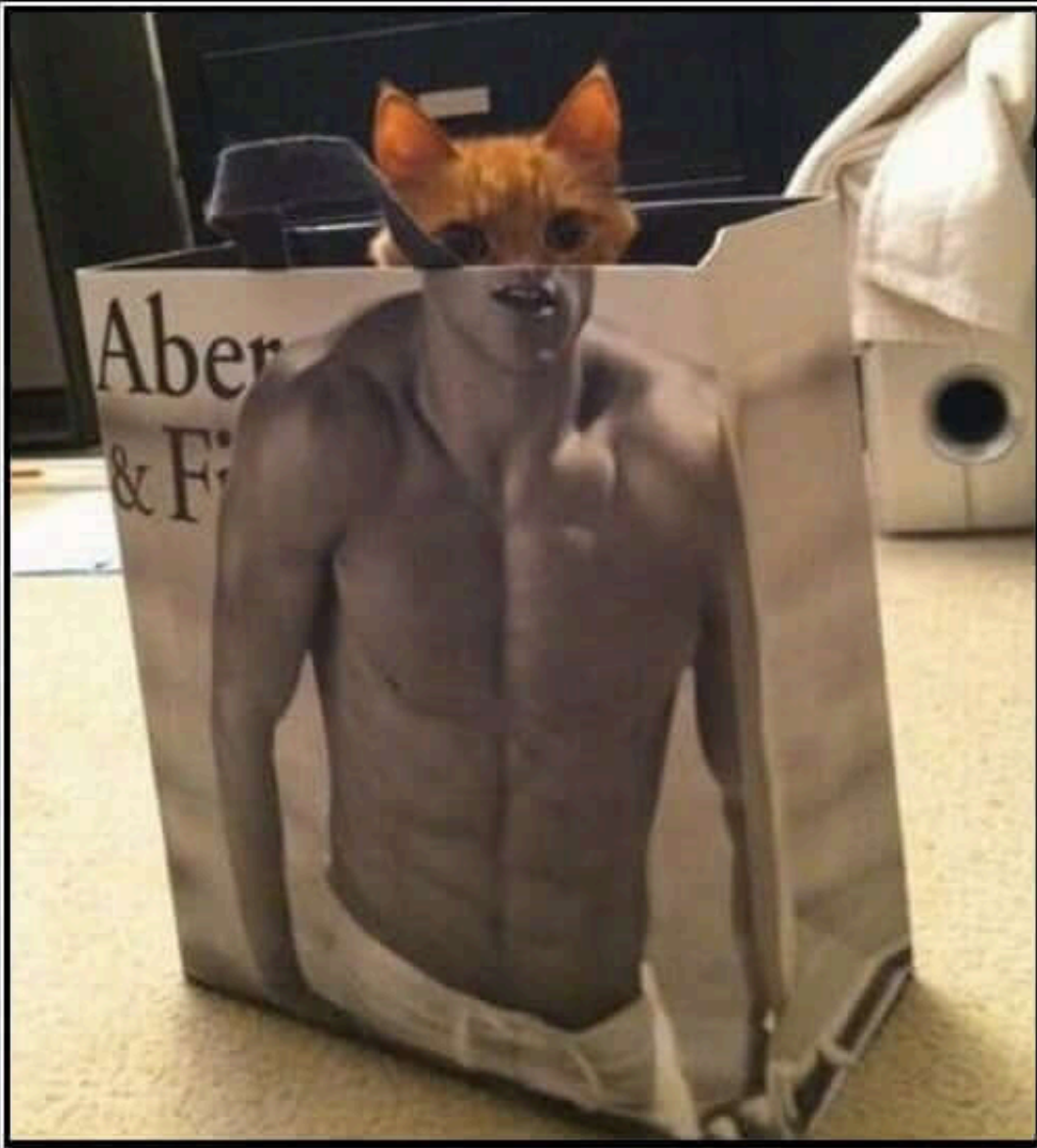
Validações

Validações para entidades e records garantindo integridade dos dados.

Bean Validation fornece uma forma comum de validação por meio de declaração de restrição e metadados para aplicativos Java

```
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;

public record UserRequestDto(
    @NotBlank @Size(min=4, max=30) String name,
    @NotBlank @Email String email,
    @NotBlank @Min(5) String password
) {
}
```



206
Partial Content

HTTP Codes

Definição correta dos códigos de estado do HTTP

O código status 206 indica que a solicitação foi bem-sucedida e o corpo contém os intervalos de dados solicitados.

```
@RestController
@EnableAutoConfiguration
public class Example {

    public static void main(String[] args) {
        SpringApplication.run(Example.class, args);
    }

    @RequestMapping("/")
    public String home() {
        return "Hello World!";
    }
}
```

Controler REST

Definição de uma interface de comunicação padronizada e centrada em Recursos

REST rapidamente se tornou o padrão a construção de serviços web na web porque eles são fáceis de construir e consumir

Serviços

- **Data Transfer Objects**
- **Mappers**

Domínio

- **Models/Entities**

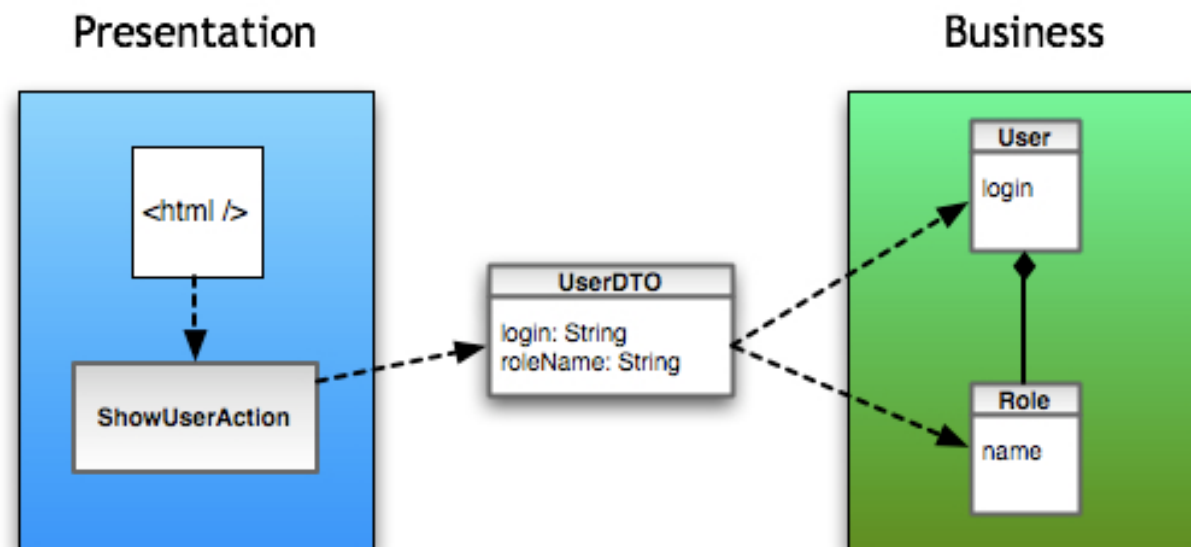
Persistência

- **Repositories**

Data Transfer Objects

Definir corretamente objetos para a transferência de dados entre cliente e servidor.

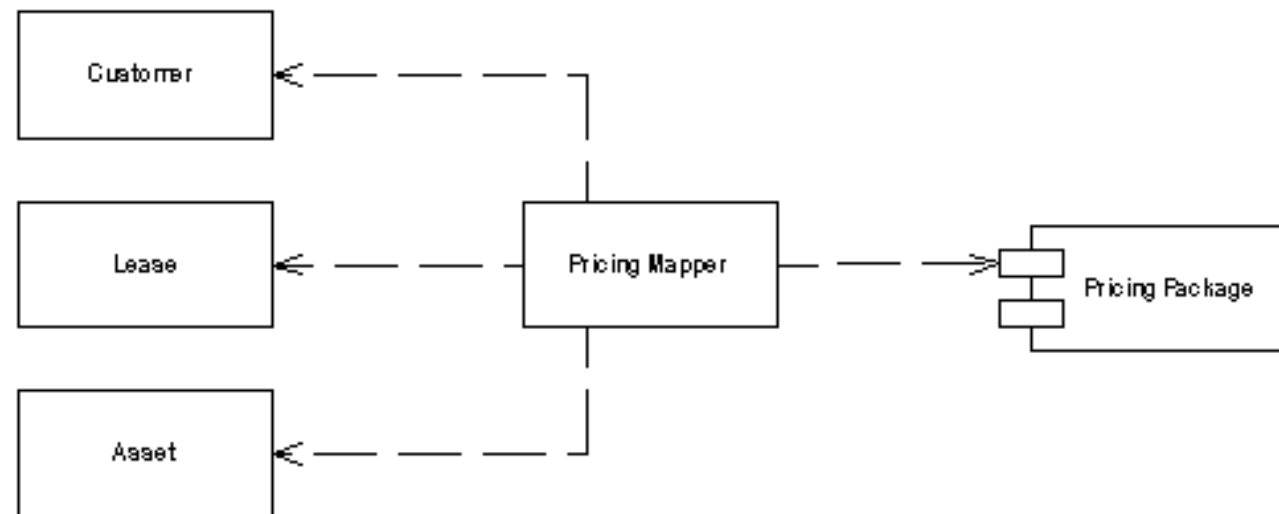
"DTOs mantêm lógica fora do resto do código e também fornecem um ponto claro para alterar a serialização, caso deseje" - Martin Fowler

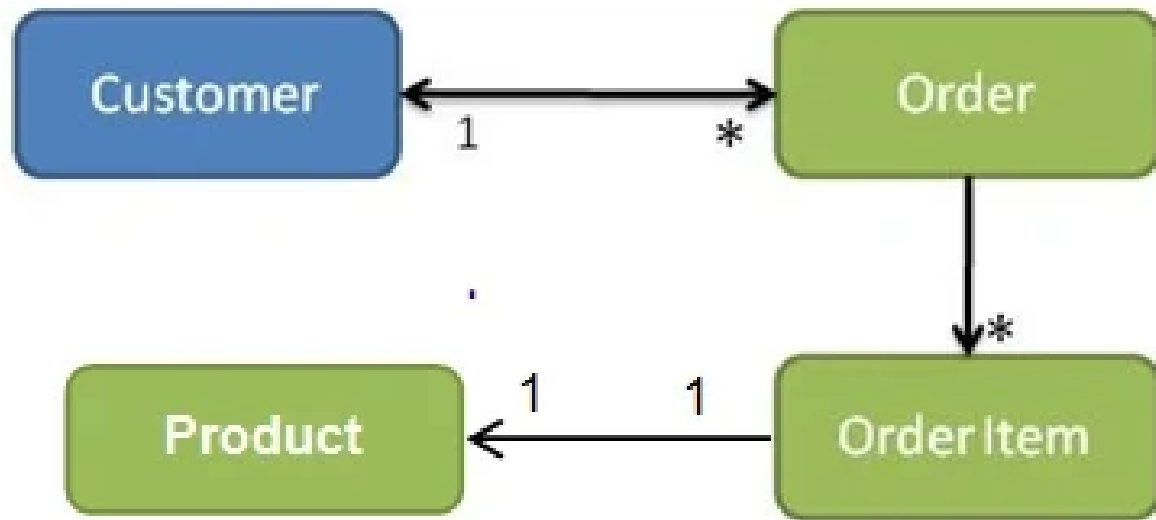


Mappers

Mapeamento de objetos para objetos

"Às vezes você precisa configurar comunicações entre dois subsistemas que ainda precisam permanecer ignorantes um do outro." - Martin Fowler





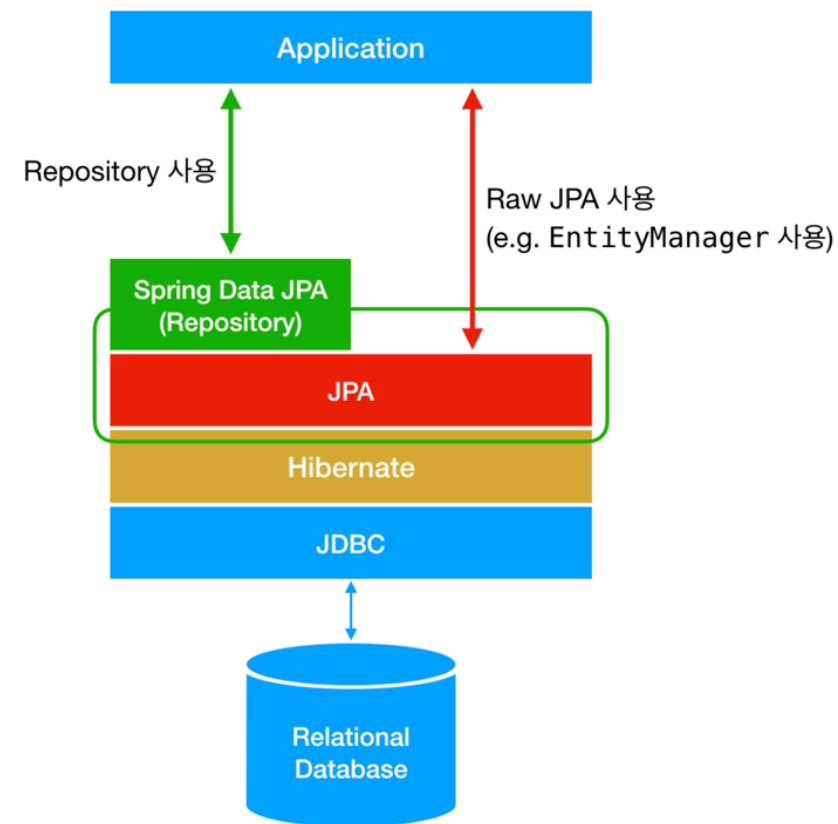
Models

Modelagem dos objetos que serão centro da aplicação, contendo todos os dados necessários para o funcionamento

Repositories

Interface da aplicação com persistência de dados

O Repositorio fornece funcionalidade CRUD sofisticada para a classe de entidade que está sendo gerenciada.



Segurança

- Autenticação
- Autorização

Infraestrutura

- Docker
- Database (PostgreSQL)
- Implantação (Deploy)
- Testes Unitários *
- Documentação *

Authentication



Confirms users
are who they say they are.

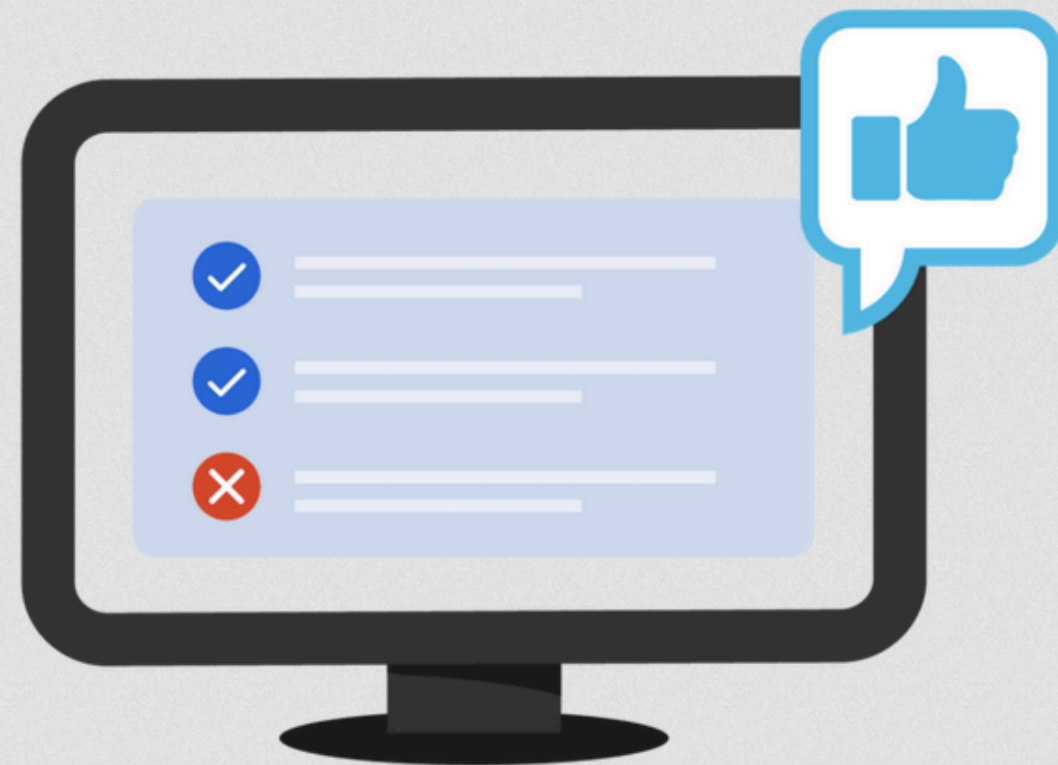
Autenticação

Processo de reconhecimento de
um cliente/usuário identificado.

Autorização

Processo de autorização de acesso
à um usuário/cliente já autenticado

Authorization

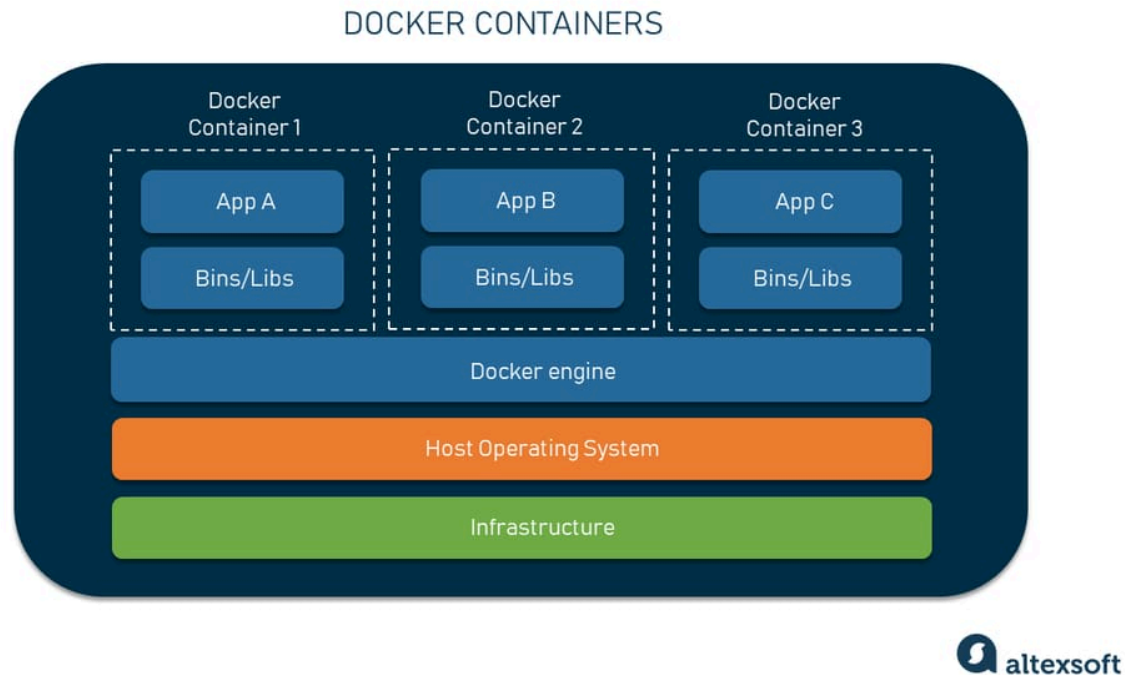


Gives users permission
to access a resource.

Docker

Encapsulamento de bibliotecas e ambientes em **containers padronizados**

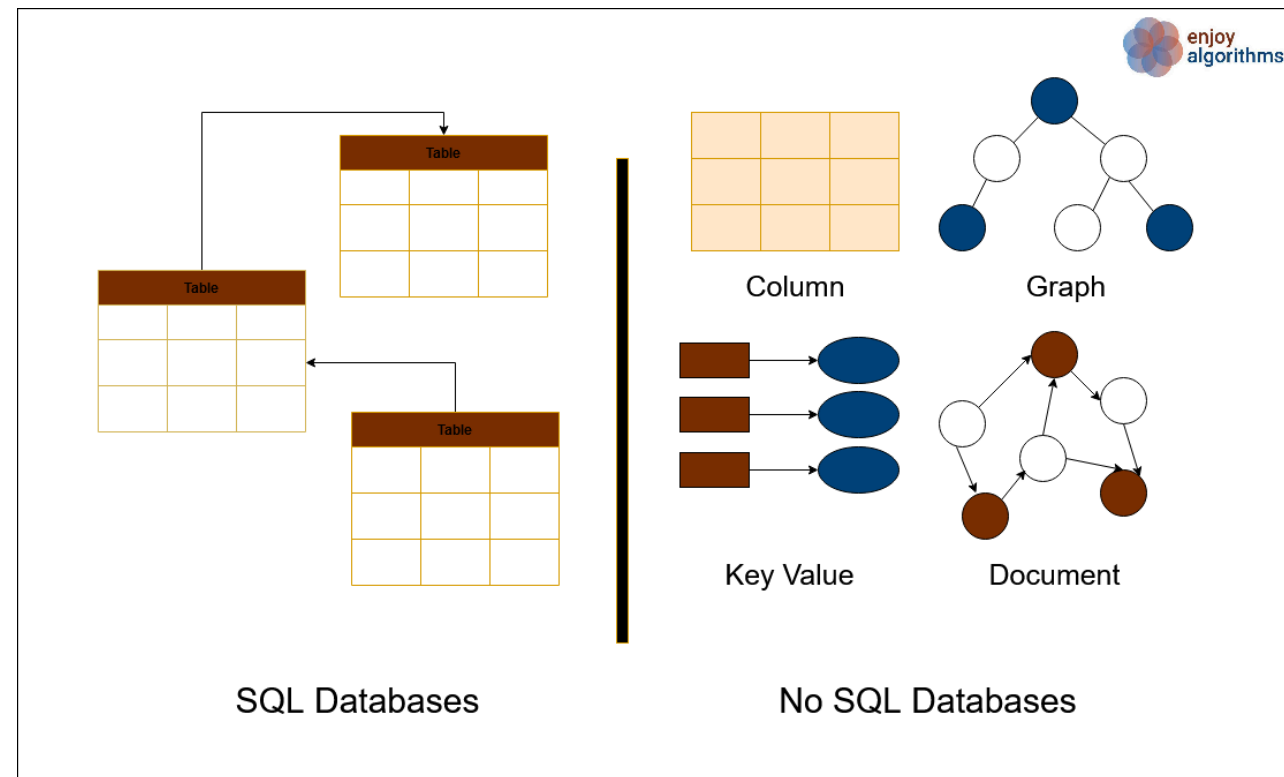
Com o Docker, os aplicativos e seus ambientes são virtualizados e isolados uns dos outros em um sistema operacional compartilhado do computador host.

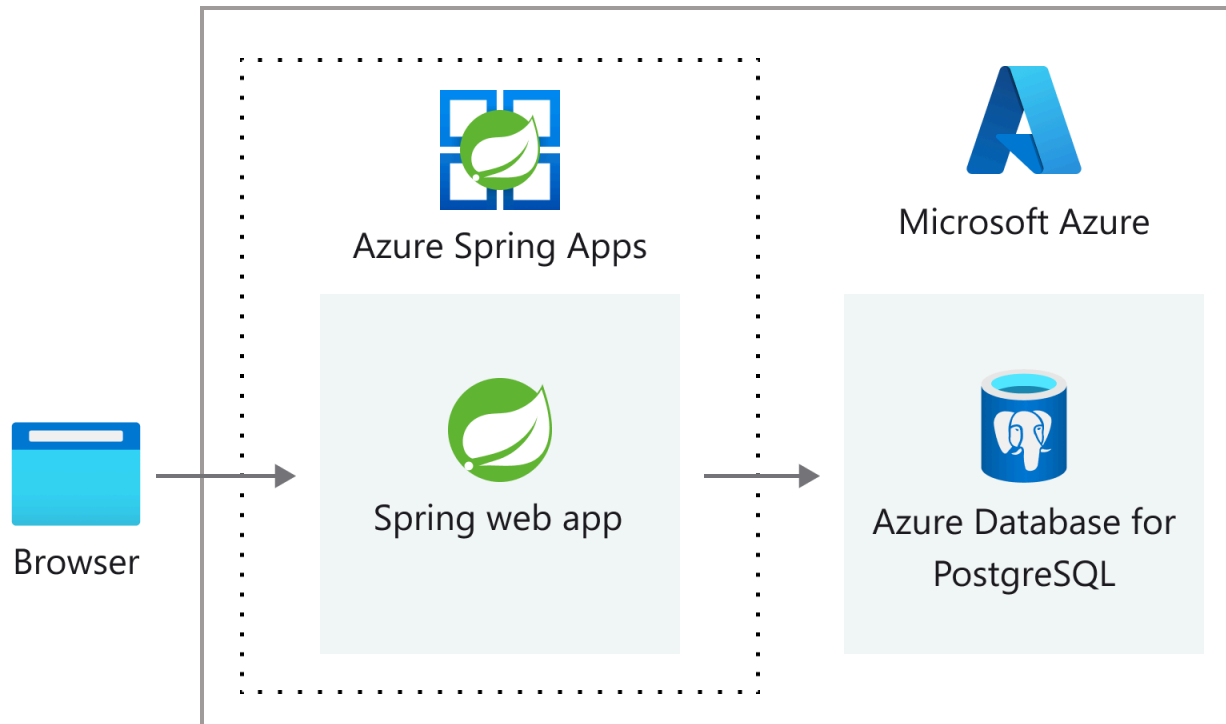


Database

Gerenciadores de dados persistentes.

Um banco de dados é uma coleção organizada de informações estruturadas, ou dados, armazenados em um computador.





Deploy

Disponibilização a um público (ambiente de produção). Requer a configuração de infraestruturas específicas.

Testes Unitários

Teste de funções usando JUnit.

Uma das principais vantagens da injeção de dependência é que ela deve facilitar o teste de unidade do seu código. Você pode usar objetos simulados em vez de dependências reais.

```
@RunWith(SpringRunner.class)
@DataJpaTest
public class ItemRepositoryTest {

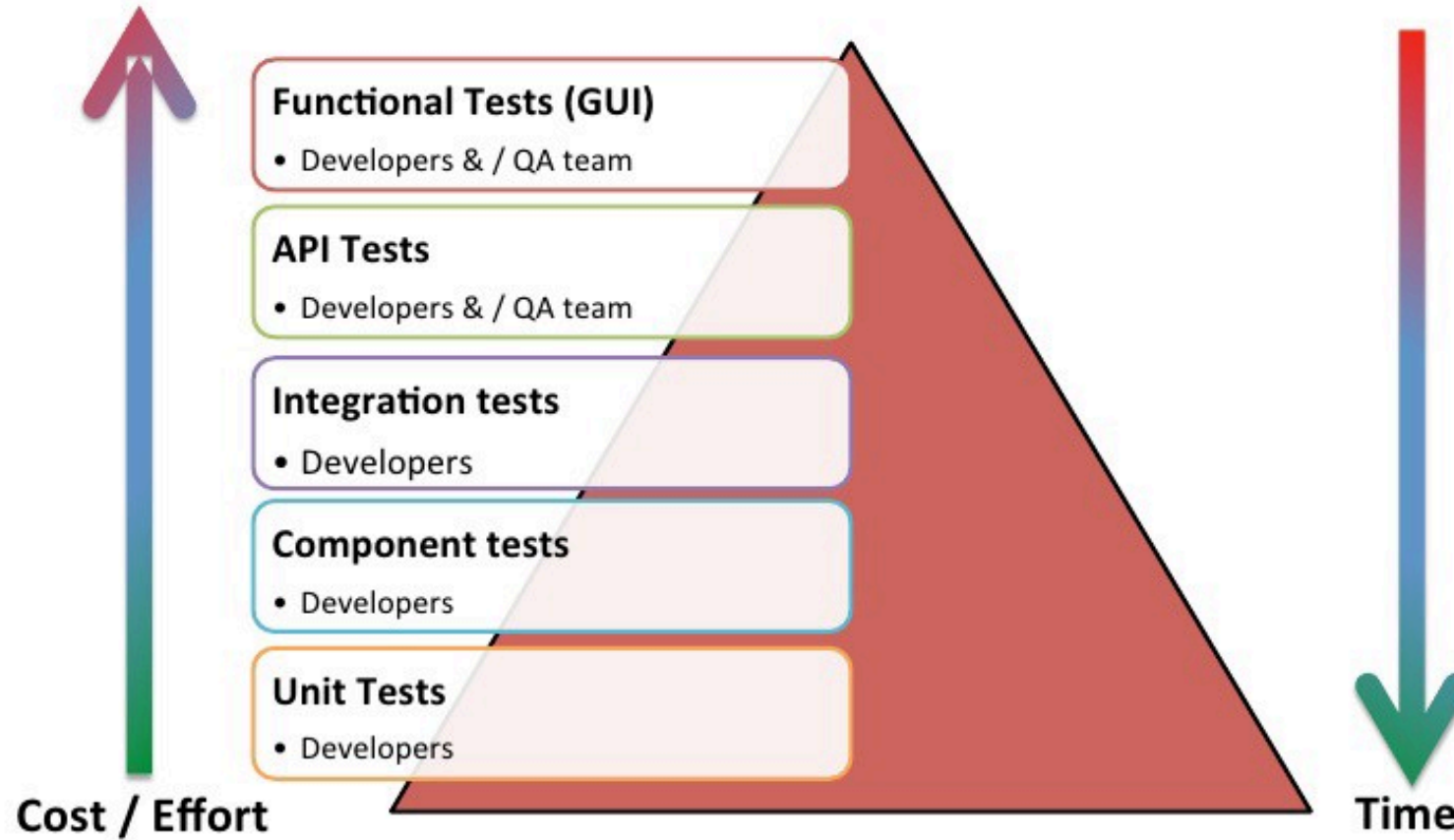
    @Autowired
    private ItemRepository itemRepository;

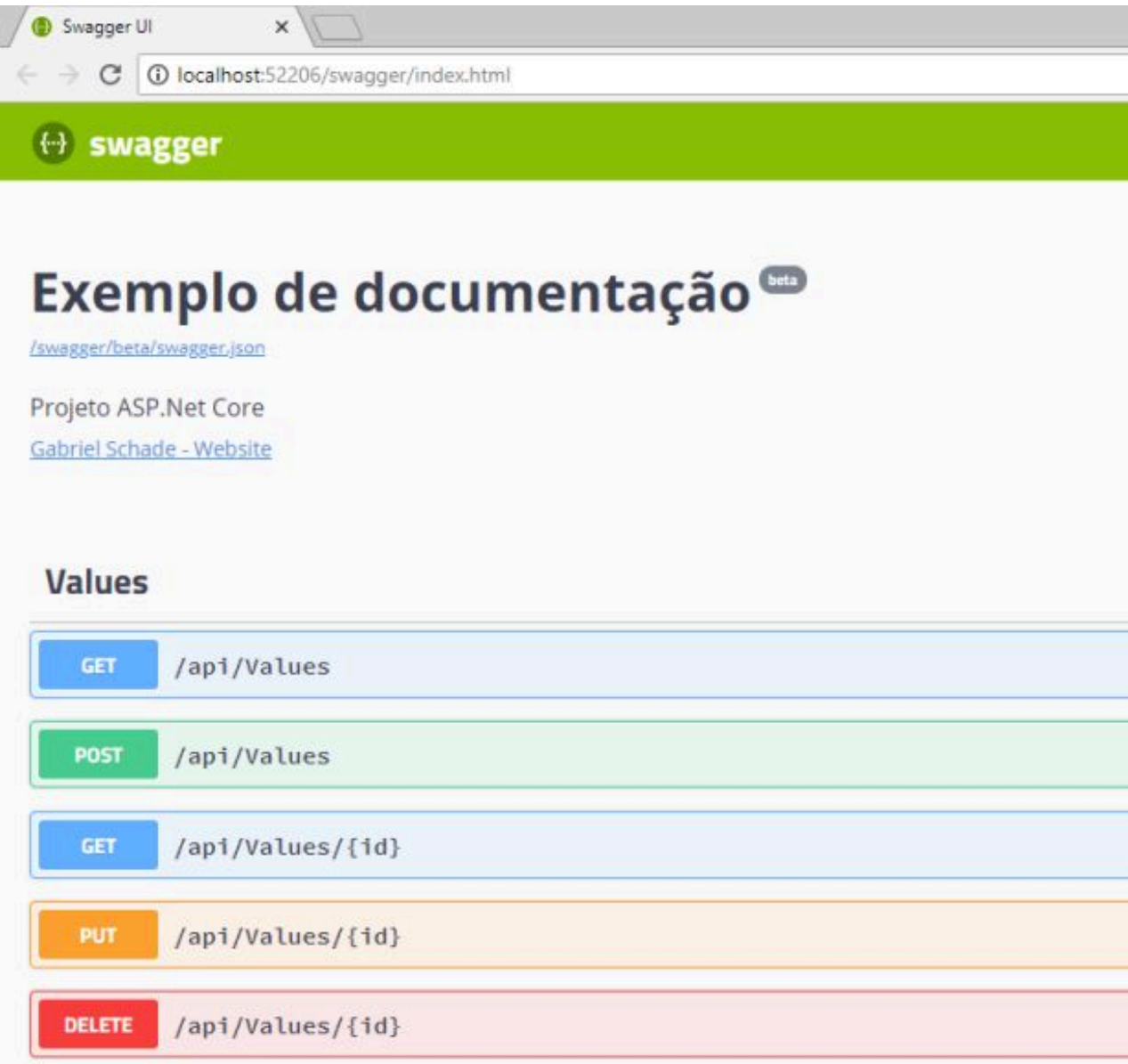
    @Before
    public void setUp() throws Exception {
        List<Item> items = Arrays.asList(
            new Item(1, "Ball 1", 100, 10),
            new Item(2, "Ball 2", 100, 20),
            new Item(3, "Ball 3", 100, 30)
        );
        itemRepository.saveAll(items);
    }

    @After
    public void tearDown() throws Exception {
        itemRepository.deleteAll();
    }

    // TEST METHOD IS OMITTED
}
```

Ideal Test Pyramid





Documentação

A padronização da API ajuda na sua auto-documentação. Porém podemos utilizar de ferramentas para aprimorar a documentação dos detalhes da API