Introdução: Lidando com Exceções em APIs

Em APIs REST, é fundamental retornar **códigos de status HTTP apropriados** para cada situação. Isso melhora a experiência do desenvolvedor que consome a API e facilita a depuração.

500

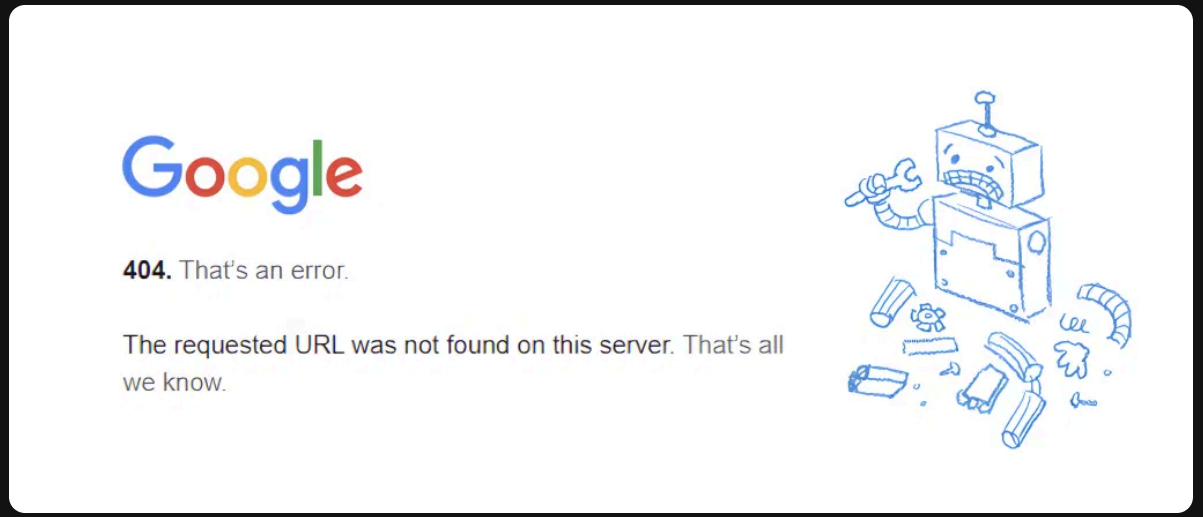
Internal Server Error
Indica um problema no servidor

404

Not Found
Recurso não encontrado

Boa Prática:

Quando um recurso não é encontrado (como um registro no banco de dados), a API deve retornar o código 404 (Not Found) em vez do código 500 (Internal Server Error).

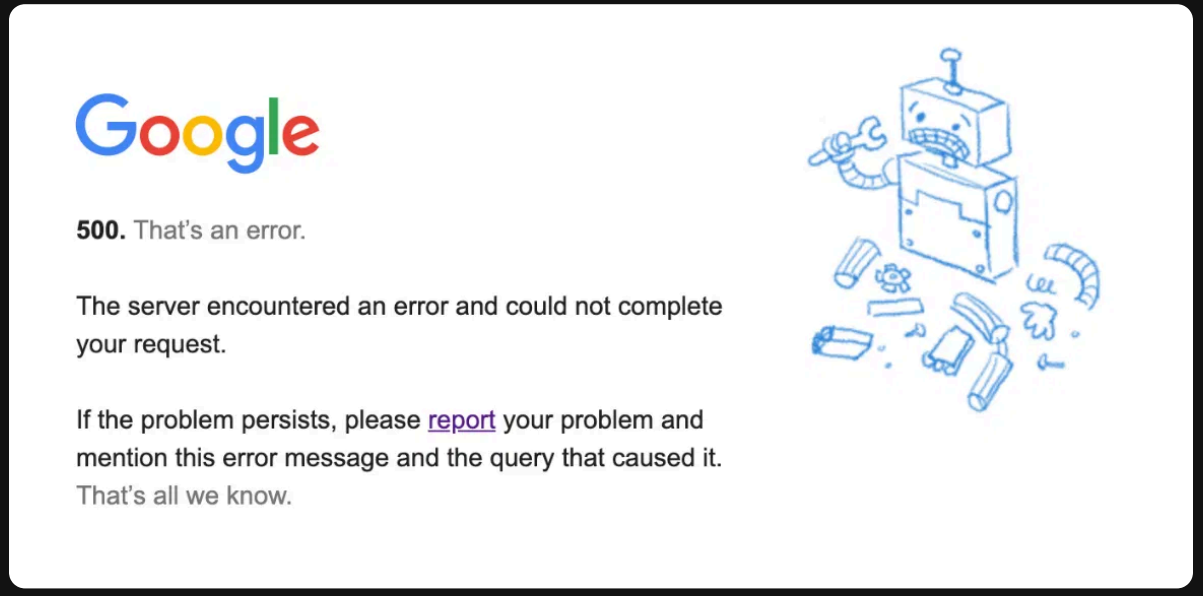


O Problema do getReferenceById()

No método **detalhar** de um controller, quando buscamos um registro inexistente usando **getReferenceById()**, uma exceção **EntityNotFoundException** é lançada.

```
// MedicoController.java
@GetMapping("/{id}")
public ResponseEntity detalhar(@PathVariable Long id) {
  var medico = repository.getReferenceById(id);
  return ResponseEntity.ok(new
DadosDetalhamentoMedico(medico));
}
```

Busca por ID inexistente → EntityNotFoundException → Spring retorna erro 500



Por padrão, o Spring Boot trata exceções não capturadas como **erro 500**, quando o ideal seria retornar **erro 404** para recursos não encontrados.

Centralizando o Tratamento de Erros

Abordagem Tradicional

- Blocos try-catch em cada método do controller
- Duplicação de código de tratamento de erros
- Controllers com código não relacionado ao negócio

Abordagem Centralizada

- Classe dedicada para tratamento de exceções
- Reutilização de código em toda a aplicação
- Controllers limpos e focados na lógica de negócio

Solução Spring Boot:

O Spring Boot oferece a anotação **@RestControllerAdvice** para criar uma classe que centraliza o tratamento de exceções em toda a aplicação, mantendo os controllers limpos e o código mais organizado.

Organização do Projeto (Refatoração)

Estrutura Original

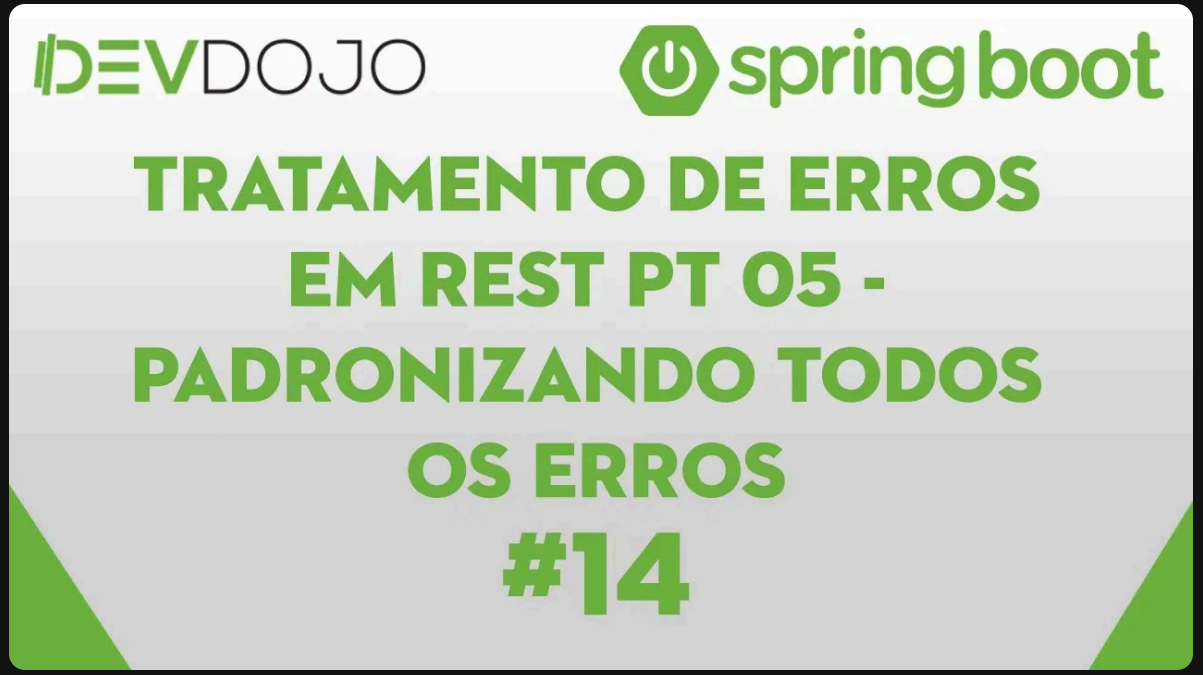
```
med.voll.api
  ├── controller
  ├── endereco
  ├── medico
  └── paciente
```

→

Estrutura Refatorada

```
med.voll.api
  ├── controller
  ├── domain
  │   ├── endereco
  │   ├── medico
  │   └── paciente
  └── infra
      └──
        TratadorDeErros.java
```

1. Agrupar pacotes relacionados ao domínio (**endereco**, **medico**, **paciente**) em um único pacote **domain**
2. Criar um novo pacote **infra** para componentes de infraestrutura, como o tratamento de erros
3. Atualizar os imports nos controllers para apontar para os novos caminhos dos pacotes



Implementando o TratadorDeErros

```java
// Pacote de infraestrutura
package med.voll.api.infra;

import jakarta.persistence.EntityNotFoundException;
import org.springframework.http.ResponseEntity;
import
org.springframework.web.bind.annotation.ExceptionHandler;
import
org.springframework.web.bind.annotation.RestControllerAdvice;

@RestControllerAdvice
public class TratadorDeErros {

  @ExceptionHandler(EntityNotFoundException.class)
  public ResponseEntity tratarErro404() {
    return ResponseEntity.notFound().build();
  }
}
```

1. Criar um pacote **infra** para código de infraestrutura
2. Criar a classe **TratadorDeErros** com a anotação **@RestControllerAdvice**
3. Implementar método com **@ExceptionHandler** especificando a exceção a ser tratada
4. Retornar o **ResponseEntity** com o status HTTP adequado

> **Nota:** A anotação **@RestControllerAdvice** indica ao Spring que esta classe deve ser carregada automaticamente e utilizada para tratar exceções em toda a aplicação.

Tratando EntityNotFoundException

Benefícios da Abordagem Centralizada

A centralização do tratamento de erros com **@RestControllerAdvice** traz diversos benefícios para o desenvolvimento e manutenção de APIs.

Código Limpo

Controllers focados apenas na lógica de negócio, sem código de tratamento de erros

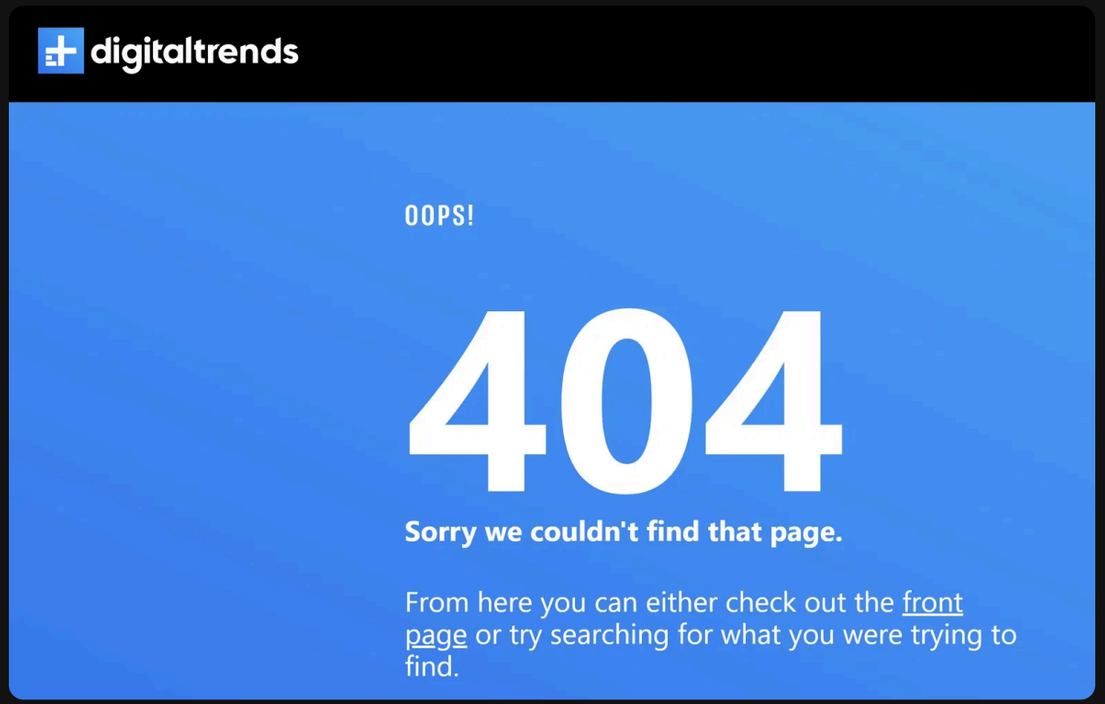
Reutilização

Tratamento de exceções aplicado automaticamente em todos os controllers da aplicação

Manutenibilidade

Alterações no tratamento de erros são feitas em um único lugar, facilitando a manutenção

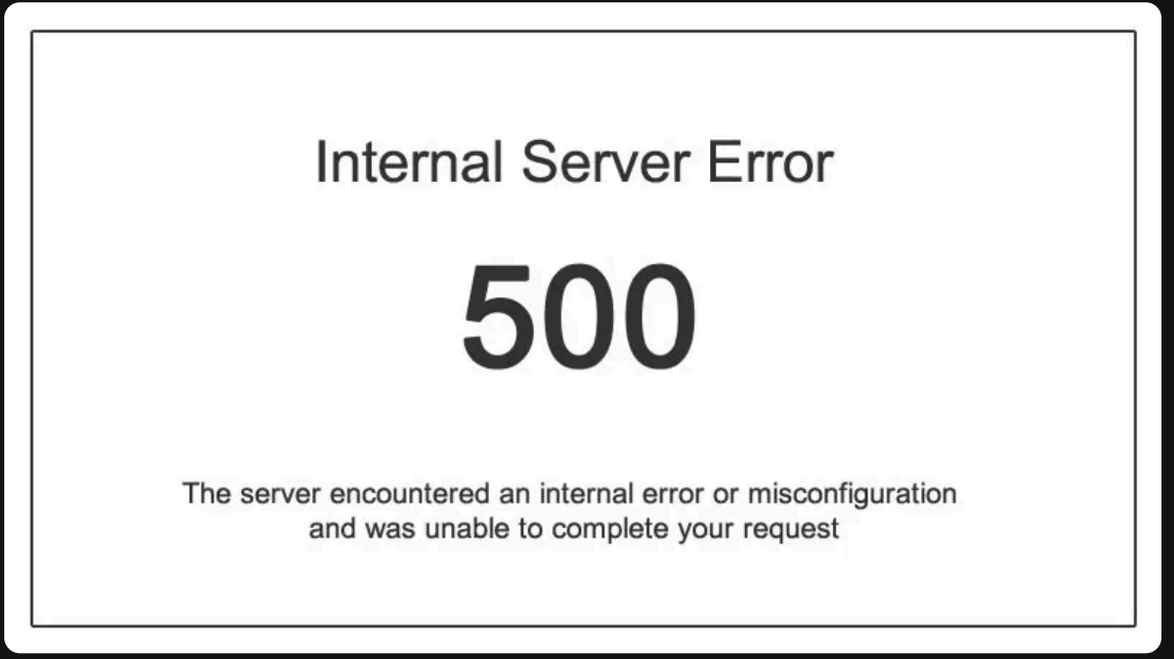
> *"Código limpo não é escrito seguindo um conjunto de regras. Você não se torna um artesão de software aprendendo uma lista de heurísticas. Profissionalismo e artesanato vêm de valores que impulsionam disciplinas."*
>
> *— Robert C. Martin, Clean Code*



Verificação e Próximos Passos

Verificação da Solução

- Requisição para um ID inexistente agora retorna **404 Not Found** em vez de 500 Internal Server Error
- Tratamento centralizado aplicado a todos os controllers sem alteração de código
- Código mais limpo e organizado, sem blocos try-catch espalhados



Próximos Passos

1. Implementar tratamento para erro 400 (Bad Request) para falhas de validação
2. Adicionar tratamento para outras exceções comuns (403 Forbidden, 401 Unauthorized)
3. Incluir mensagens de erro personalizadas para melhorar a experiência do desenvolvedor